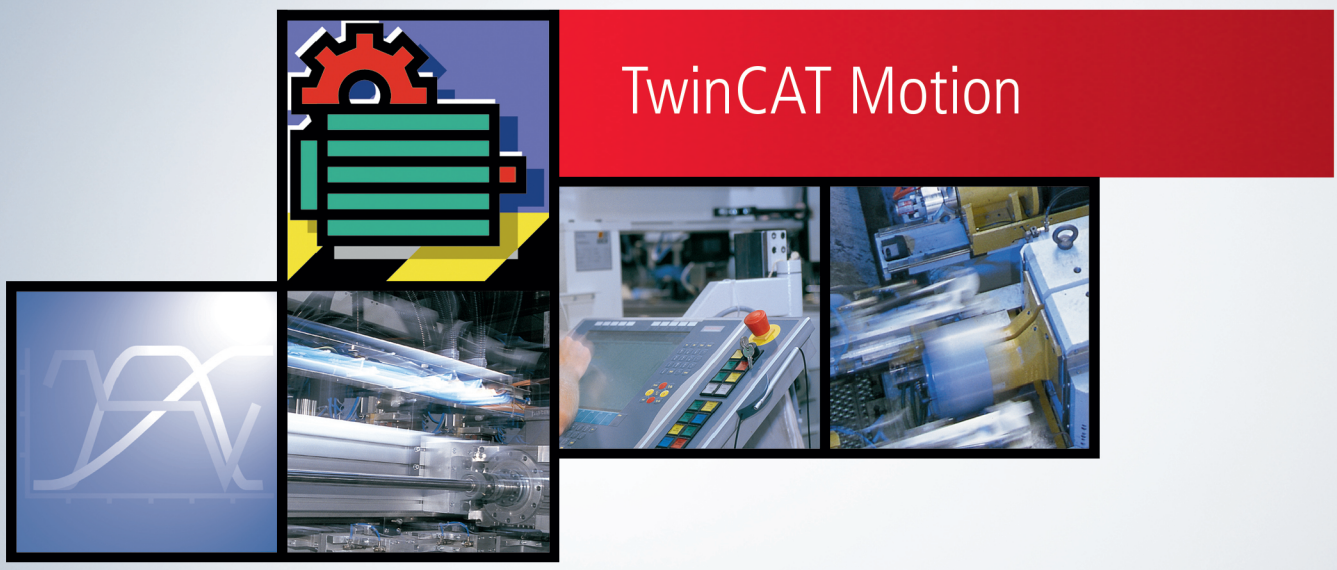
BECKHOFF New Automation Technology

Handbuch | DE

PLC-Bibliothek: TcMC

TwinCAT 2 | TX1200



06.10.2021 | Version: 1.0

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, für jede Installation und Inbetriebnahme die zu dem betreffenden Zeitpunkt veröffentliche 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 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®

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

Copyright

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.

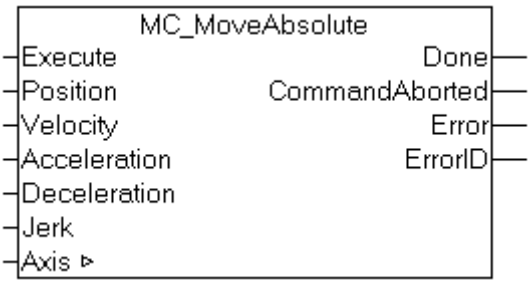
2 Übersicht

In dieser Bibliothek sind die von der Nutzerorganisation der IEC61131 Anwender PLCopen standardisierten Bausteine enthalten.

Die Spezifikation kann unter www.PLCopen.org abgerufen werden. In der TcMC.lib sind die Basisbausteine enthalten. Die Camming-Bausteine werden gesondert mit dem Camming-Paket ausgeliefert.

3 Funktionsbausteine Bewegung

3.1 MC_MoveAbsolute



Mit dem Funktionsbaustein MC_MoveAbsolute wird eine absolute Positionierung durchgeführt.

VAR_INPUT

```
VAR_INPUT
    Execute      : BOOL;
    Position     : LREAL;
    Velocity     : LREAL;
    Acceleration : LREAL;
    Deceleration : LREAL;
    Jerk         : LREAL;
END_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt.

Position : Absolute Zielposition auf die positioniert werden soll.

Velocity : Maximale Geschwindigkeit mit der gefahren werden soll (>0).

Acceleration : Beschleunigung (≥0). Bei einem Wert von 0 wirkt die Standardbeschleunigung aus der Achskonfiguration im System Manager.

Deceleration : Verzögerung (≥0). Bei einem Wert von 0 wirkt die Standardverzögerung aus der Achskonfiguration im System Manager.

Jerk : Ruck (≥0). Bei einem Wert von 0 wirkt die Standardruck aus der Achskonfiguration im System Manager.

VAR_OUTPUT

```
VAR_OUTPUT
    Done           : BOOL;
    CommandAborted : BOOL;
    Error          : BOOL;
    ErrorID        : UDINT;
END_VAR
```

Done : Wird TRUE, wenn die Positionierung abgeschlossen ist.
Das *Done* Signal wird frühestens nach Ende der logischen Sollwert-Positionierung gesetzt (Sollwertgenerator, HasJob-Flag). Abhängig von eingeschalteten Überwachungsfunktionen, wie der *Zielpositionsüberwachung*, wird *Done* erst gesetzt, wenn sich auch die Ist-Position in einem definierten *Zielpositionsfenster* befindet (Lageregelfehler). Siehe auch im Anhang [▶ 77].

CommandAborted : Wird TRUE, wenn das Kommando nicht vollständig ausgeführt werden konnte. Der Grund kann ein Stopp oder ein überlagerter Fahrbefehl sein.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

VAR_IN_OUT

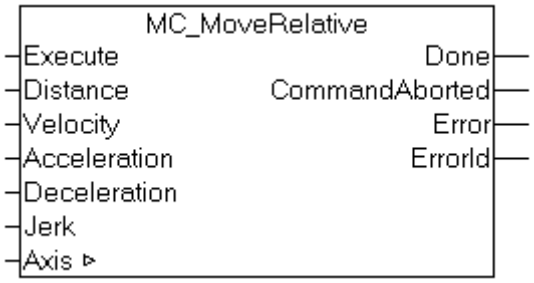
```
VAR_IN_OUT
    Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Achsstruktur.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcMC.Lib

3.2 MC_MoveRelative



Mit dem Funktionsbaustein MC_MoveRelative wird eine relative Positionierung durchgeführt.

VAR_INPUT

```
VAR_INPUT
    Execute      : BOOL;
    Distance     : LREAL;
    Velocity     : LREAL;
    Acceleration : LREAL;
    Deceleration : LREAL;
    Jerk         : LREAL;
END_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt.

Distance : Relative Wegstrecke um die positioniert werden soll.

Velocity : Maximale Geschwindigkeit mit der gefahren werden soll (>0).

Acceleration : Beschleunigung (≥0). Bei einem Wert von 0 wirkt die Standardbeschleunigung aus der Achskonfiguration im System Manager

Deceleration : Verzögerung (≥0). Bei einem Wert von 0 wirkt die Standardverzögerung aus der Achskonfiguration im System Manager

Jerk : Ruck (≥0). Bei einem Wert von 0 wirkt die Standardruck aus der Achskonfiguration im System Manager

VAR_OUTPUT

```
VAR_OUTPUT
    Done           : BOOL;
    CommandAborted : BOOL;
    Error          : BOOL;
    ErrorID        : UDINT;
END_VAR
```

Done : Wird TRUE, wenn die Positionierung abgeschlossen ist.
Das *Done* Signal wird frühestens nach Ende der logischen Sollwert-Positionierung gesetzt (Sollwertgenerator, HasJob-Flag). Abhängig von eingeschalteten Überwachungsfunktionen, wie der *Zielpositionsüberwachung*, wird *Done* erst gesetzt, wenn sich auch die Ist-Position in einem definierten *Zielpositionsfenster* befindet (Lageregelfehler). Siehe auch im Anhang [▶ 77].

CommandAborted : Wird TRUE, wenn das Kommando nicht vollständig ausgeführt werden konnte. Der Grund kann ein Stopp oder ein überlagerter Fahrbefehl sein.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

VAR_IN_OUT

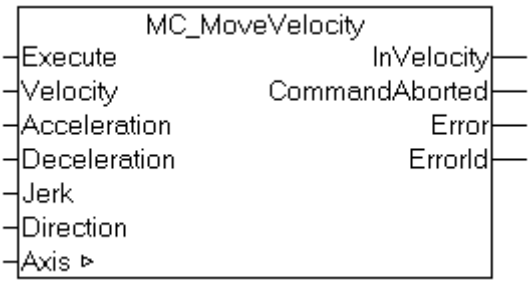
```
VAR_IN_OUT
    Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Achsstruktur.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcMC.Lib

3.3 MC_MoveVelocity



Mit dem Funktionsbaustein MC_MoveVelocity wird eine Endlosfahrt durchgeführt.

VAR_INPUT

```
VAR_INPUT
    Execute      : BOOL;
    Velocity     : LREAL;
    Acceleration : LREAL;
    Deceleration : LREAL;
    Jerk         : LREAL;
    Direction    : MC_Direction;
ND_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt.

Velocity : Maximale Geschwindigkeitsüberhöhung mit der gefahren werden soll (>0).

Acceleration : Beschleunigung (≥0). Bei einem Wert von 0 wirkt die Standardbeschleunigung aus der Achskonfiguration im System Manager

Deceleration : Verzögerung (≥0). Bei einem Wert von 0 wirkt die Standardverzögerung aus der Achskonfiguration im System Manager

Jerk : Ruck (≥0). Bei einem Wert von 0 wirkt die Standardruck aus der Achskonfiguration im System Manager

Direction : Richtung [▶ 72] für die Bewegung.

VAR_OUTPUT

```
VAR_OUTPUT
    InVelocity     : BOOL;
    CommandAborted : BOOL;
    Error          : BOOL;
    ErrorID        : UDINT;
END_VAR
```

InVelocity : Wird TRUE, wenn die Zielgeschwindigkeit erreicht wurde.

CommandAborted : Wird TRUE, wenn das Kommando nicht vollständig ausgeführt werden konnte. Der Grund kann ein Stopp oder ein überlagerter Fahrbefehl sein.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

VAR_IN_OUT

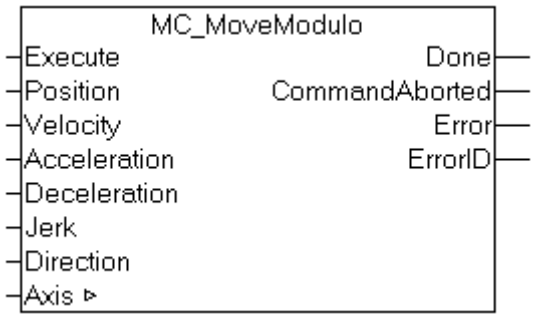
```
VAR_IN_OUT
    Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Achsstruktur.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcMC.Lib

3.4 MC_MoveModulo



Mit dem Funktionsbaustein MC_MoveModulo wird eine Positionierung durchgeführt, die sich auf die Moduloposition einer Achse bezieht. Grundlage für eine Moduloumdrehung ist dabei der einstellbare Achsparameter *Modulofaktor* (z. B. 360). Abhängig vom *Direction* Eingang werden drei mögliche Starttypen unterschieden.

- Positionierung in *positive Richtung*
- Positionierung in *negative Richtung*
- Positionierung *auf kürzestem Weg*

Sonderfälle: Besonders zu beachten ist das Verhalten bei Anforderung einer oder mehrerer vollständiger Moduloumdrehungen. Befindet sich die Achse auf einer exakten Sollposition von beispielsweise 90 Grad und wird sie auf 90 Grad positioniert, so wird keine Bewegung ausgeführt. Bei Anforderung von 450 Grad in positiver Richtung fährt sie eine Umdrehung. Nach einem Achsreset kann das Verhalten anders sein, weil durch den Reset die aktuelle Istposition in die Sollposition übernommen wird. Damit steht die Achse nicht

mehr exakt bei 90 Grad, sondern ein wenig darunter oder darüber. In diesen beiden Fällen wird entweder nur eine minimale Positionierung auf 90 Grad oder aber eine ganze Umdrehung ausgeführt. Auf dieses Problematik wird in den Erläuterungen [▶ 13] näher eingegangen.

Je nach Anwendungsfall kann es für volle Moduloumdrehungen günstiger sein, die gewünschte Zielposition auf Grund der aktuellen absoluten Position zu berechnen und mit dem Baustein MC_MoveAbsolute zu positionieren.

Hinweis: Die Modulopositionierung ebenso wie die Absolutpositionierung steht für alle Achsen unabhängig von der *Modulo* Einstellung im TwinCAT SystemManager zur Verfügung. Für jede Achse kann die aktuelle Absolutposition *fPosSoll* aus dem zyklischen Achsinterface NCTOPLC_AXLESTRUCT ausgelesen werden.

Wichtig: Nähere Erläuterungen zu Modulo Bewegungen [▶ 13]

VAR_INPUT

```
VAR_INPUT
    Execute       : BOOL;
    Position      : LREAL;
    Velocity      : LREAL;
    Acceleration  : LREAL;
    Deceleration  : LREAL;
    Jerk          : LREAL;
    Direction     : MC_Direction;
END_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt.

Position : Absolute Modulozielposition auf die positioniert werden soll.

Die Zielposition muss für den Modulo-Starttypen *MC_Shortest_Way* (*Direction*) kleiner als die im TwinCAT SystemManager eingestellte Modulo-Periode sein (z. B. 0 ≤ Position < 360°). Für andere Starttypen sind auch größere Positionen erlaubt, die dann zu ein oder mehreren Umdrehungen führen bevor das Ziel angefahren wird.

Velocity : Maximale Geschwindigkeit mit der gefahren werden soll (>0).

Acceleration : Beschleunigung (≥0). Bei einem Wert von 0 wirkt die Standardbeschleunigung aus der Achskonfiguration im System Manager

Deceleration : Verzögerung (≥0). Bei einem Wert von 0 wirkt die Standardverzögerung aus der Achskonfiguration im System Manager

Jerk : Ruck (≥0). Bei einem Wert von 0 wirkt die Standardruck aus der Achskonfiguration im System Manager

Direction : Positionierrichtung MC_Direction [▶ 72] (positive Richtung, negative Richtung, auf kürzestem Weg)

VAR_OUTPUT

```
VAR_OUTPUT
    Done           : BOOL;
    CommandAborted : BOOL;
    Error          : BOOL;
    ErrorID        : UDINT;
END_VAR
```

Done : Wird TRUE, wenn die Positionierung abgeschlossen ist.
Das *Done* Signal wird frühestens nach Ende der logischen Sollwert-Positionierung gesetzt (Sollwertgenerator, HasJob-Flag). Abhängig von eingeschalteten Überwachungsfunktionen, wie der *Zielpositionsüberwachung*, wird *Done* erst gesetzt, wenn sich auch die Ist-Position in einem definierten *Zielpositionsfenster* befindet (Lageregelfehler). Siehe auch im Anhang [▶ 77].

CommandAborted : Wird TRUE, wenn das Kommando nicht vollständig ausgeführt werden konnte. Der Grund kann ein Stopp oder ein überlagerter Fahrbefehl sein.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

VAR_IN_OUT

```
VAR_IN_OUT
    Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Achsstruktur.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcMC.Lib

3.5 Hinweise zur Modulo-Positionierung

Die Modulo-Positionierung (MC_MoveModulo [▶ 11]) ist unabhängig vom Achstyp möglich. Sie kann also bei linearen ebenso wie bei rotatorischen Achsen angewendet werden, da TwinCAT nicht zwischen diesen Typen unterscheidet. Auch eine Modulo-Achse hat eine fortlaufende absolute Position im Bereich ±∞. Die Modulo-Position der Achse ist einfach eine zusätzliche Information zur absoluten Achsposition und die Modulo-Positionierung stellt die gewünschte Zielposition auf eine andere Art dar. Im Gegensatz zur absoluten Positionierung, bei der der Benutzer das Ziel eindeutig vorgibt, birgt die Modulo-Positionierung einige Tücken, da die gewünschte Zielposition unterschiedlich interpretiert werden kann.

Einstellungen im TwinCAT SystemManager

Die Modulo-Positionierung bezieht sich grundsätzlich auf eine im TwinCAT SystemManager einstellbare *Modulo-Periode*. In den Beispielen auf dieser Seite wird von einer rotatorischen Achse mit einer *Modulo-Periode* von 360 Grad ausgegangen.

Das *Modulo-Toleranzfenster* definiert ein Positionsfenster um die aktuelle Modulo-Sollposition der Achse herum. Die Fensterbreite entspricht dem doppelten angegebenen Wert (Sollposition ± Toleranzwert). Auf das Toleranzfenster wird im Folgenden näher eingegangen.

Besonderheiten beim Reset einer Achse

Die Positionierung einer Achse bezieht sich immer auf deren Sollposition. Die Sollposition der Achse ist im Normalfall die Position, die mit dem letzten Fahrauftrag angefahren wurde. Durch einen Achsreset (MC_Reset [▶ 43], Zuschalten der Reglerfreigabe mit MC_Power [▶ 42]) kann sich eine vom Anwender nicht erwartete Sollposition einstellen, da in diesem Fall die aktuelle Istposition als Sollposition übernommen wird. Der Achsreset setzt durch diesen Vorgang einen eventuell aufgetretenen Schleppfehler zurück. Wenn dieser Umstand nicht berücksichtigt wird, kann sich eine nachfolgende Positionierung unerwartet verhalten.

Beispiel: Eine Achse wird auf 90° positioniert, wodurch die Sollposition der Achse anschließend exakt 90° beträgt. Ein weiterer Modulo-Fahrauftrag auf 450° in *positive Richtung* führt zu einer vollen Umdrehung und die Modulo-Position der Achse ist anschließend wieder exakt 90°. Wird jetzt ein Achsreset durchgeführt, so kann die Sollposition zufällig etwas kleiner oder etwas größer als 90° sein. Der neue Wert ist abhängig vom Istwert der Achse zum Zeitpunkt des Reset. In beiden Fällen verhält sich das nächste Fahrkommando unterschiedlich. Liegt die Sollposition leicht unter 90°, so führt ein neues Fahrkommando auf 90° in *positive Richtung* nur zu einer minimalen Bewegung. Die durch den Reset entstandene Abweichung wird ausgeglichen und die Sollposition ist anschließend wieder exakt 90°. Liegt aber die Sollposition nach dem Achsreset leicht über 90°, so führt dasselbe Fahrkommando zu einer vollen Umdrehung um wieder die exakte Sollposition von 90° zu erreichen. Diese Problematik tritt auf, wenn volle Umdrehungen um 360° oder ein Vielfaches von 360° beauftragt werden. Bei Positionierungen auf einen von der aktuellen Modulo-Position entfernten Winkel ist der Fahrauftrag eindeutig.

Um das Problem zu lösen, kann ein *Modulo-Toleranzfenster* im TwinCAT SystemManager parametriert werden. Kleine Abweichungen der Position, die innerhalb des Fensters liegen, führen damit nicht mehr zu einem unterschiedlichen Verhalten der Achse. Wird beispielsweise ein Fenster von 1° parametriert, so verhält sich die Achse im oben beschriebenen Fall gleich, solange die Sollposition zwischen 89° und 91° liegt. Wenn jetzt die Sollposition weniger als 1° über 90° liegt, wird die Achse bei einem Modulo-Start in *positive Richtung* zurückpositioniert. Bei einer Zielposition von 90° wird also in beiden Fällen eine Minimalbewegung auf exakt 90° ausgeführt und bei einer Zielposition von 450° wird in beiden Fällen eine ganze Umdrehung gefahren.

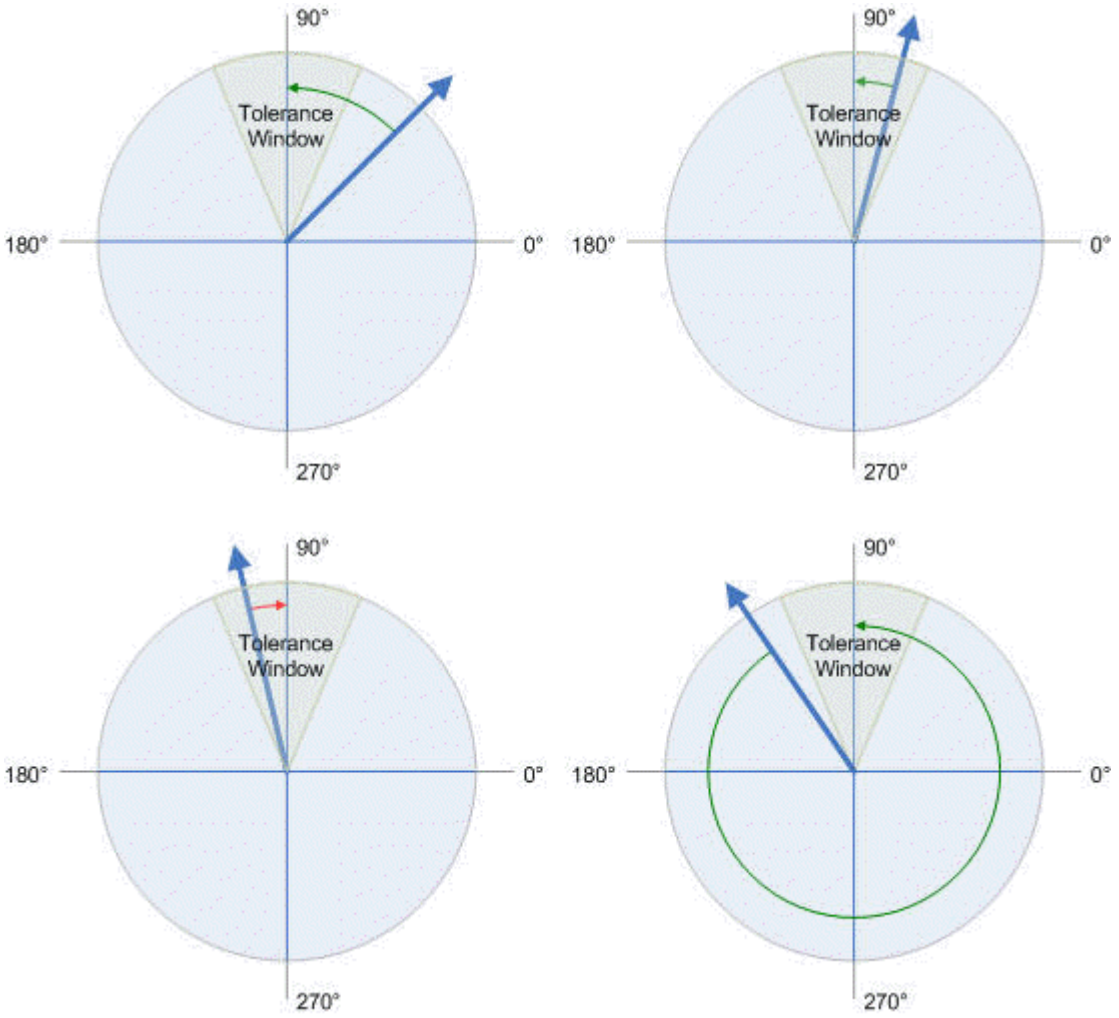


Abb. 1: Abbildung: Wirkung des Modulo-Toleranzfensters - Modulo-Zielposition 90° in positive Richtung

Das Modulo-Toleranzfenster kann also innerhalb des Fensters zu Bewegungen gegen die beauftragte Richtung führen. Bei einem kleinen Fenster ist das normalerweise unproblematisch, weil auch Regelabweichungen zwischen Soll- und Istposition in beide Richtungen ausgeglichen werden. Das Toleranzfenster lässt sich also auch bei Achsen verwenden, die konstruktionsbedingt nur in einer Richtung verfahren werden dürfen.

Modulo-Positionierung um weniger als eine Umdrehung

Die Modulo-Positionierung von einer Ausgangsposition auf eine nicht identische Zielposition ist eindeutig und birgt keine Besonderheiten. Eine Modulo-Zielposition im Bereich [0 ≤ Position < 360] führt in weniger als einer ganzen Umdrehung zum gewünschten Ziel. Ist die Zielposition mit der Ausgangsposition identisch, so wird keine Bewegung ausgeführt. Bei Zielpositionen ab 360 Grad aufwärts werden ein oder mehr vollständige Umdrehungen ausgeführt, bevor die Achse auf die gewünschte Zielposition fährt.

Für eine Bewegung von 270° auf 0° darf demnach nicht 360°, sondern es muss 0° als Modulo-Zielposition abgegeben werden, da 360 außerhalb des Grundbereiches liegt und zu einer zusätzlichen Umdrehung führen würde.

Die Modulo-Positionierung unterscheidet drei Richtungsvorgaben, *positive Richtung*, *negative Richtung* und *auf kürzestem Weg* (MC_Direction [▶ 72]). Bei der Positionierung auf kürzestem Weg sind Zielpositionen ab 360° nicht sinnvoll, da das Ziel immer direkt angefahren wird. Im Gegensatz zur positiven oder negativen Richtung können also nicht mehrere Umdrehungen ausgeführt werden, bevor das Ziel angefahren wird.

Wichtig: Bei Modulo-Positionierungen mit dem Start-Typ *auf kürzestem Weg* sind nur Modulo-Zielpositionen in der Grundperiode (z. B. kleiner als 360°) erlaubt, anderenfalls wird ein Fehler zurückgegeben.

Die folgende Tabelle zeigt einige Positionierungsbeispiele:

Richtung (Modulo-Start-typ)	Absolute An-fangsposition	Modulo-Zielpo-sition	Relativer Ver-fahrweg	absolute End-position	Modulo Endpo-sition
positive Richtung	90,00	0,00	270,00	360,00	0,00
positive Richtung	90,00	360,00	630,00	720,00	0,00
positive Richtung	90,00	720,00	990,00	1080,00	0,00
negative Richtung	90,00	0,00	-90,00	0,00	0,00
negative Richtung	90,00	360,00	-450,00	-360,00	0,00
negative Richtung	90,00	720,00	-810,00	-720,00	0,00
auf kürzestem Weg	90,00	0,00	-90,00	0,00	0,00

Modulo-Positionierung um ganze Umdrehungen

Modulo-Positionierungen um ein oder mehrere ganze Umdrehungen verhalten sich grundsätzlich nicht anders als Positionierungen auf von der Ausgangsposition entfernt liegende Winkel. Wenn die beauftragte Zielposition gleich der Ausgangsposition ist, so wird keine Bewegung ausgeführt. Für eine ganze Umdrehung muss zur Ausgangsposition 360° addiert werden.

Das weiter oben beschriebene Reset-Verhalten zeigt , dass Positionierungen mit ganzzahligen Umdrehungen besonders beachtet werden müssen. Die nachfolgende Tabelle zeigt Positionierbeispiele für eine Ausgangsposition von ungefähr 90°. Das Modulo-Toleranzfenster TF ist hier auf 1° eingestellt. Besondere Fälle, in denen die Ausgangsposition außerhalb dieses Fensters liegt, sind gekennzeichnet.

Richtung (Modulo-Starttyp)	Absolute An-fangsposition	Modulo-Ziel-position	Relativer Ver-fahrweg	absolute Endposition	Modulo End-position	Anmerkung
positive Richtung	90,00	90,00	0,00	90,00	90,00	
positive Richtung	90,90	90,00	-0,90	90,00	90,00	
positive Richtung	91,10	90,00	358,90	450,00	90,00	außerhalb TF
positive Richtung	89,10	90,00	0,90	90,00	90,00	
positive Richtung	88,90	90,00	1,10	90,00	90,00	außerhalb TF
positive Richtung	90,00	450,00	360,00	450,00	90,00	
positive Richtung	90,90	450,00	359,10	450,00	90,00	
positive Richtung	91,10	450,00	718,90	810,00	90,00	außerhalb TF
positive Richtung	89,10	450,00	360,90	450,00	90,00	
positive Richtung	88,90	450,00	361,10	450,00	90,00	außerhalb TF
positive Richtung	90,00	810,00	720,00	810,00	90,00	
positive Richtung	90,90	810,00	719,10	810,00	90,00	
positive Richtung	91,10	810,00	1078,90	1170,00	90,00	außerhalb TF
positive Richtung	89,10	810,00	720,90	810,00	90,00	
positive Richtung	88,90	810,00	721,10	810,00	90,00	außerhalb TF
negative Richtung	90,00	90,00	0,00	90,00	90,00	
negative Richtung	90,90	90,00	-0,90	90,00	90,00	
negative Richtung	91,10	90,00	-1,10	90,00	90,00	außerhalb TF
negative Richtung	89,10	90,00	0,90	90,00	90,00	
negative Richtung	88,90	90,00	-358,90	-270,00	90,00	außerhalb TF
negative Richtung	90,00	450,00	-360,00	-270,00	90,00	
negative Richtung	90,90	450,00	-360,90	-270,00	90,00	
negative Richtung	91,10	450,00	-361,10	-270,00	90,00	außerhalb TF
negative Richtung	89,10	450,00	-359,10	-270,00	90,00	

Richtung (Modulo-Starttyp)	Absolute Anfangsposition	Modulo-Zielposition	Relativer Verfahrweg	absolute Endposition	Modulo Endposition	Anmerkung
negative Richtung	88,90	450,00	-718,90	-630,00	90,00	außerhalb TF
negative Richtung	90,00	810,00	-720,00	-630,00	90,00	
negative Richtung	90,90	810,00	-720,90	-630,00	90,00	
negative Richtung	91,10	810,00	-721,10	-630,00	90,00	außerhalb TF
negative Richtung	89,10	810,00	-719,10	-630,00	90,00	
negative Richtung	88,90	810,00	-1078,90	-990,00	90,00	außerhalb TF

Modulo-Berechnungen im SPS Programm

Alle Positionieraufträge an eine Achse werden in TwinCAT NC auf der Basis der *Sollposition* durchgeführt. Die aktuelle Istposition wird nur zur Regelung herangezogen. Wenn in einem SPS-Programm eine neue Zielposition ausgehend von der aktuellen Position berechnet werden soll, so muss diese Berechnung mit der aktuellen Sollposition der Achse durchgeführt werden. Im Gegensatz zur absoluten Sollposition fPosSoll, wird die Modulo-Sollposition nicht explizit im zyklischen Interface NCTOPLC_AXLESTRUCT der Achse zur Verfügung gestellt. Die Modulo-Sollposition und auch die Anzahl der Modulo-Umdrehungen lassen sich aber mit Hilfe von Bibliotheksfunktionen aus der Bibliothek *TcMath.lib* berechnen.

```
ModuloSollposition := MODABS( NcToPlc.fPosSoll, 360 );

ModuloSollUmdrehungen := MODTURNS( NcToPlc.fPosSoll, 360 );
```

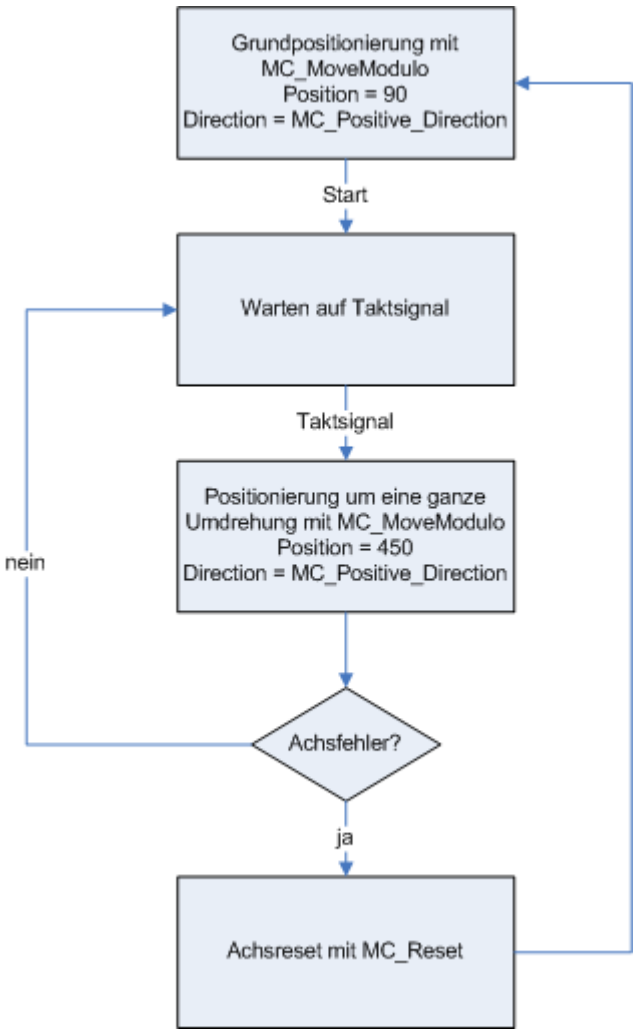
Es ist nicht zu empfehlen, Auftragsberechnungen auf Basis der Modulo-Istposition durchzuführen, die im zyklischen Achsinterface (*fModuloPosIst* und *nModuloTurns*) zur Verfügung steht. Wegen der mehr oder weniger großen Regelabweichung der Achse könnten sich Fehler im programmierten Ablauf, wie z. B. unerwünschte Umdrehungen, ergeben.

Anwendungsbeispiel

In einer Anlage führt eine Rotationsachse einen Arbeitsschritt aus. Die Ausgangsposition für jeden Arbeitsschritt ist 90° und mit jedem Takt soll die Achse um 360° in positive Richtung positioniert werden. Eine Rückwärtspositionierung ist aus mechanischen Gründen nicht erlaubt. Kleine Rückwärtspositionierungen im Rahmen der Lageregelung sind zulässig.

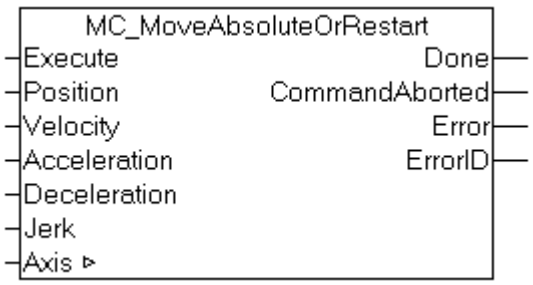
Das Modulo-Toleranzfenster wird im SystemManager auf 1,5° eingestellt. Damit werden unerwünschte Umdrehungen der Achse nach einem Achsreset vermieden. Da die Achse nur vorpositioniert werden darf, wird das Kommando MC_MoveModulo [▶ 11] mit dem Modulo-Starttyp *positive Richtung (MC_Positive_Direction)* verwendet. Die Modulo-Zielposition wird mit 450° angegeben, da die Ausgangsorientierung nach einer vollen Umdrehung um 360° wieder erreicht werden soll. Eine Modulo-Zielposition von 90° würde hier keine Bewegung ausführen.

Der Ablauf startet zunächst mit einer Grundpositionierung (MC_MoveModulo [▶ 11]), mit der die exakte Ausgangsposition sichergestellt wird. Anschließend wechselt die Schrittkette in einen Bearbeitungszyklus. Im Fehlerfall wird die Achse mit MC_Reset [▶ 43] zurückgesetzt und anschließend, am Anfang der Schrittkette, auf ihre gültige Ausgangsposition gefahren. In diesem Fall wird 90° als Zielposition angegeben, damit diese Position schnellst möglich angefahren wird. Steht die Achse bereits an der Ausgangsposition, so wird keine Bewegung ausgeführt.



Der Reset-Schritt kann alternativ auch am Anfang der Schrittkette ausgeführt werden um die Achse auch zu Beginn des Ablaufs zu initialisieren.

3.6 MC_MoveAbsoluteOrRestart



Mit dem Funktionsbaustein MC_MoveAbsoluteOrRestart kann einerseits eine Bewegung aus dem Stillstand, wie mit dem Baustein MC_MoveAbsolute [▶ 8], gestartet werden. Andererseits werden Zielposition und Geschwindigkeit geändert, wenn der Baustein während einer Bewegung nachgetriggert wird.

VAR_INPUT

```
VAR_INPUT
    Execute       : BOOL;   (* Start axis. Toggle while axis is moving to restart axis *)
    Position      : LREAL;  (* target position *)
    Velocity      : LREAL;  (* velocity *)
    Acceleration  : LREAL;  (* optional on axis start. not used for restart *)
    Deceleration  : LREAL;  (* optional on axis start. not used for restart *)
    Jerk          : LREAL;  (* optional on axis start. not used for restart *)
ND_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt. Während der Bewegung kann der Baustein nachgetriggert werden.

Position : Absolute Zielposition auf die positioniert werden soll.

Velocity : Maximale Geschwindigkeit mit der gefahren werden soll (>0).

Acceleration : Beschleunigung (≥0). Bei einem Wert von 0 wirkt die Standardbeschleunigung aus der Achskonfiguration im System Manager. Die Beschleunigung wird nur bei einem Start der Achse aus dem Stillstand verwendet.

Deceleration : Verzögerung (≥0). Bei einem Wert von 0 wirkt die Standardverzögerung aus der Achskonfiguration im System Manager. Die Verzögerung wird nur bei einem Start der Achse aus dem Stillstand verwendet.

Jerk : Ruck (≥0). Bei einem Wert von 0 wirkt die Standardruck aus der Achskonfiguration im System Manager. Der Ruck wird nur bei einem Start der Achse aus dem Stillstand verwendet.

VAR_OUTPUT

```
VAR_OUTPUT
    Done            : BOOL;   (* move completed *)
    CommandAborted  : BOOL;   (* move aborted *)
    Error           : BOOL;
    ErrorID         : UDINT;
END_VAR
```

Done : Wird TRUE, wenn die Positionierung abgeschlossen ist.
Das *Done* Signal wird frühestens nach Ende der logischen Sollwert-Positionierung gesetzt (Sollwertgenerator, HasJob-Flag). Abhängig von eingeschalteten Überwachungsfunktionen, wie der *Zielpositionsüberwachung*, wird *Done* erst gesetzt, wenn sich auch die Ist-Position in einem definierten *Zielpositionsfenster* befindet (Lageregelfehler). Siehe auch im Anhang [▶ 77].

CommandAborted : Wird TRUE, wenn das Kommando nicht vollständig ausgeführt werden konnte. Der Grund kann ein Stopp oder ein überlagerter Fahrbefehl sein.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

VAR_IN_OUT

```
VAR_IN_OUT
    Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Achsstruktur.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8 Build	PC (i386)	TcMC.Lib

3.7 MC_NewPos

Mit dem Funktionsbaustein MC_NewPos kann einer sich bewegenden Achse eine neue Zielposition vorgegeben werden.

VAR_INPUT

```
VAR_INPUT
    Execute : BOOL;
    NewPos  : LREAL;
ND_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt.

NewPos : Neue Zielposition. Die neue Zielposition kann größer oder kleiner der alten Zielposition sein.

VAR_OUTPUT

```
VAR_OUTPUT
    Done           : BOOL;
    CommandAborted : BOOL;
    Error          : BOOL;
    ErrorID        : UDINT;
END_VAR
```

Done : Wird TRUE, wenn der Befehl erfolgreich abgesetzt wurde.

CommandAborted : Wird TRUE, wenn der Auftrag unterbrochen wurde.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

VAR_IN_OUT

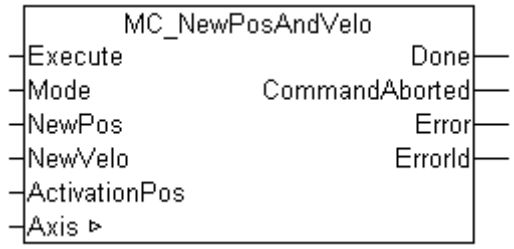
```
VAR_IN_OUT
    Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Achsstruktur.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcMC.Lib

3.8 MC_NewPosAndVelo



Mit dem Funktionsbaustein MC_NewPosAndVelo kann einer sich bewegenden Achse eine neue Zielposition mit neuer Geschwindigkeit vorgegeben werden.

VAR_INPUT

```
VAR_INPUT
    Execute       : BOOL;
    Mode          : E_CmdTypeNewTargPosAndVelo;
    NewPos        : LREAL;
    NewVelo       : LREAL;
    ActivationPos : LREAL;
ND_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt.

Mode : Mode für die neue Zielposition/Geschwindigkeit

NewPos : Neue Zielposition. Die neue Zielposition kann größer oder kleiner der alten Zielposition sein.

NewVelo : Neue Geschwindigkeit.

ActivationPos : Position an der bei nicht instantaner Übernahme die neue Position oder Geschwindigkeit aktiviert wird. Wenn der

Mode REACH_VELO_AT_POS verwendet wird, wird die neue Geschwindigkeit an dieser Position erreicht. Die Änderung beginnt also vorher.

VAR_OUTPUT

```
VAR_OUTPUT
    Done           : BOOL;
    CommandAborted : BOOL;
    Error          : BOOL;
    ErrorID        : UDINT;
END_VAR
```

Done : Wird TRUE, wenn der Befehl erfolgreich abgesetzt wurde.

CommandAborted : Wird TRUE, wenn der Auftrag unterbrochen wurde.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

VAR_IN_OUT

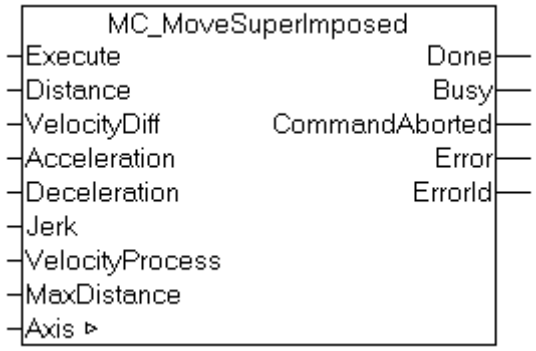
```
VAR_IN_OUT
    Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Achsstruktur.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcMC.Lib

3.9 MC_MoveSuperImposed



Mit dem Funktionsbaustein MC_MoveSuperImposed wird eine überlagerte relative Positionierung durchgeführt. Die Achse legt eine, im Vergleich zur unbeeinflussten Bewegung, zusätzliche Wegstrecke (*Distance*) zurück. Bei negativer *Distance* legt die Achse eine um diesen Betrag kürzere Strecke zurück.

Alternativ kann der erweiterte Funktionsbaustein MC_MoveSuperImposedExt [▶ 23] verwendet werden.

VAR_INPUT

```
VAR_INPUT
    Execute         : BOOL;
    Distance        : LREAL;
    VelocityDiff    : LREAL;
    Acceleration    : LREAL;
    Deceleration    : LREAL;
    Jerk            : LREAL;
    VelocityProcess : LREAL;
    MaxDistance     : LREAL;
END_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt.

Distance : Relative Wegstrecke die aufgeholt werden soll. Ein positiver Wert bedeutet eine betragsmäßige Geschwindigkeitserhöhung, um diese Strecke zusätzlich zur unbeeinflussten Bewegung zurückzulegen. Ein negativer Wert bedeutet ein Abbremsen und Zurückfallen um diese Strecke.

VelocityDiff : Maximale Geschwindigkeitsüberhöhung mit der gefahren werden soll (Betrag > 0).

Acceleration : Beschleunigung (≥0). Bei einem Wert von 0 wirkt die Standardbeschleunigung aus der Achskonfiguration im System Manager.

Deceleration : Verzögerung (≥0). Bei einem Wert von 0 wirkt die Standardverzögerung aus der Achskonfiguration im System Manager.

Jerk : Der Ruck wird nicht ausgewertet.

VelocityProcess : Mittlere Prozessgeschwindigkeit in der Konstantgeschwindigkeitsphase (Betrag > 0).

MaxDistance : Wegstrecke in der der Aufholprozess stattfinden darf (Betrag > 0).

VAR_OUTPUT

```
VAR_OUTPUT
    Done           : BOOL;
    Busy           : BOOL;
    CommandAborted : BOOL;
    Error          : BOOL;
    ErrorID        : UDINT;
END_VAR
```

Done : Wird TRUE, wenn die überlagerte Bewegung beendet ist.

Busy : Wird TRUE, während der Aufholfahrt.

CommandAborted : Wird TRUE, wenn das Kommando nicht vollständig ausgeführt werden konnte. Der Grund kann ein Fehler, ein Stopp oder ein überlagerter Fahrbefehl sein.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

Hinweis: Der Baustein liefert den Fehler 4243_{hex} (16963) wenn die Ausgleichsfahrt aufgrund der Parametrierung (Strecke, Geschwindigkeit etc.) nicht vollständig durchgeführt werden kann. In diesem Fall wird die Ausgleichsfahrt soweit wie möglich ausgeführt. Der Anwender muss entscheiden, ob er diese Fehlermeldung in seiner Applikation als echten Fehler oder eher als Warnung versteht.

VAR_IN_OUT

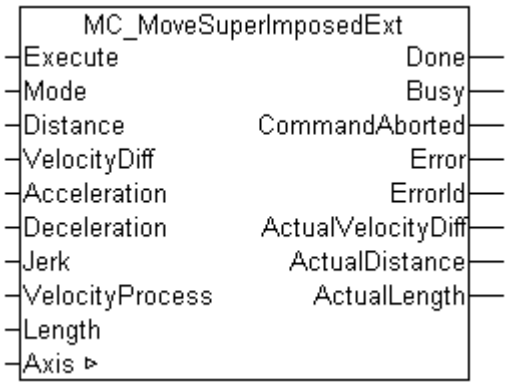
```
VAR_IN_OUT
    Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Achsstruktur.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcMC.Lib

3.10 MC_MoveSuperImposedExt



Mit dem Funktionsbaustein MC_MoveSuperImposed wird eine überlagerte relative Positionierung durchgeführt. Die Achse legt eine, im Vergleich zur unbeeinflussten Bewegung, zusätzliche Wegstrecke *Distance* zurück (Aufholfahrt). Bei negativer *Distance* legt die Achse eine um diesen Betrag kürzere Strecke zurück.

Dieser Baustein ist eine erweiterte Version des Funktionsbausteins MC_MoveSuperImposed [▶ 22].

Anwendungsbeispiele zu MC_MoveSuperimposed [▶ 25]

VAR_INPUT

```
VAR_INPUT
    Execute         : BOOL;
    Mode            : E_SuperpositionMode;
    Distance        : LREAL;
    VelocityDiff    : LREAL;
    Acceleration    : LREAL;
    Deceleration    : LREAL;
    Jerk            : LREAL;
    VelocityProcess : LREAL;
    Length          : LREAL;
END_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt.

Mode : Mode [▶ 74] legt fest, wie die überlagerte Bewegung auszuführen ist.

Distance : Relative Wegstrecke die aufgeholt werden soll. Ein positiver Wert bedeutet eine betragsmäßige Geschwindigkeitserhöhung, um diese Strecke zusätzlich zur unbeeinflussten Bewegung zurückzulegen. Ein negativer Wert bedeutet ein Abbremsen und Zurückfallen um diese Strecke.

VelocityDiff : Maximale Geschwindigkeitsüberhöhung mit der gefahren werden soll (Betrag > 0).

Acceleration : Beschleunigung (≥0). Bei einem Wert von 0 wirkt die Standardbeschleunigung aus der Achskonfiguration im System Manager.

Deceleration : Verzögerung (≥0). Bei einem Wert von 0 wirkt die Standardverzögerung aus der Achskonfiguration im System Manager.

Jerk : Der Ruck wird nicht ausgewertet.

VelocityProcess : Mittlere Prozessgeschwindigkeit in der Konstantgeschwindigkeitsphase (Betrag > 0).

Length : Wegstrecke in der der Aufholprozess stattfinden darf (Betrag > 0). Mode [▶ 74] legt fest, wie diese Strecke interpretiert wird.

VAR_OUTPUT

```
VAR_OUTPUT
    Done               : BOOL;
    Busy               : BOOL;
    CommandAborted     : BOOL;
    Error              : BOOL;
    ErrorId            : UDINT;
    ActualVelocityDiff : LREAL;
    ActualDistance     : LREAL;
    ActualLength       : LREAL;
END_VAR
```

Done : Wird TRUE, wenn die überlagerte Bewegung beendet ist.

Busy : Wird TRUE, während der Aufholfahrt.

CommandAborted : Wird TRUE, wenn das Kommando nicht vollständig ausgeführt werden konnte. Der Grund kann ein Fehler, ein Stopp oder ein überlagerter Fahrbefehl sein.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

Hinweis: Der Baustein liefert den Fehler 4243_{hex} (16963) wenn die Ausgleichsfahrt aufgrund der Parametrierung (Strecke, Geschwindigkeit etc.) nicht vollständig durchgeführt werden kann. In diesem Fall wird die Ausgleichsfahrt soweit wie möglich ausgeführt. Der Anwender muss entscheiden, ob er diese Fehlermeldung in seiner Applikation als echten Fehler oder eher als Warnung versteht.

ActualVelocityDiff : Tatsächlich genutzte Geschwindkeitsüberhöhung während der überlagerten Bewegung (ActualVelocityDiff ≤ VelocityDiff).

ActualDistance : Tatsächlich überlagerte Distanz. Der Baustein versucht die vorgegebene Distanz *Distance* vollständig zu erreichen. Abhängig von der Parametrierung (*VelocityDiff, Acceleration, Deceleration, Length, Mode*) kann diese Distanz evtl. nicht vollständig erreicht werden. In diesen Fällen wird die maximal mögliche Distanz überlagert. (*ActualDistance≤Distance*).

ActualLength : Tatsächlich zurückgelegte Strecke während der überlagerten Bewegung (*ActualLenght≤Length*).

VAR_IN_OUT

```
VAR_IN_OUT
    Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Achsstruktur.

Voraussetzungen

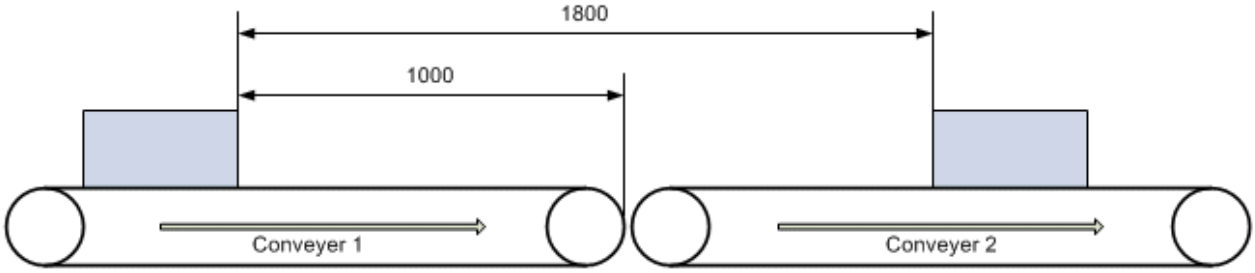
Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9 ab Build 1025	PC (i386)	TcMC.Lib

3.11 Anwendungsbeispiele zu MC_MoveSuperImposedExt

Der Funktionsbaustein MC_MoveSuperImposedExt [▶ 23] startet einen überlagerte Bewegung auf einer bereits fahrenden Achse. Für diese Überlagerung gibt es verschiedene Anwendungsfälle, die im Folgenden beschrieben werden.

Abstandskorrektur für Produkte auf einem Förderband

Ein Förderband besteht aus einzelnen Segmenten, die jeweils durch eine Achse angetrieben werden. Auf dem Förderband werden Pakete transportiert, deren Abstand korrigiert werden soll. Dazu muss ein Fördersegment im Vergleich zu einem nachfolgenden Segment kurzzeitig schneller oder langsamer laufen.



Der gemessene Abstand ist 1800 mm und soll auf 1500 mm reduziert werden. Dazu soll Transportband Conveyer 1 beschleunigt werden, um den Abstand zu reduzieren. Die Korrektur muss bis zum Ende des Bandes 1 abgeschlossen sein, damit das Paket nicht auf das langsamer laufende Band 2 geschoben wird.

Da in dieser Situation Conveyer 1 beschleunigt werden muss, benötigt das Antriebssystem eine Geschwindigkeitsreserve, die hier mit 500 mm/s angenommen wird. In der Praxis lässt sich dieser Wert aus der Differenz zwischen maximaler Transportgeschwindigkeit und aktueller Sollgeschwindigkeit ermitteln.

Für die Parametrierung des Funktionsbausteins MC_MoveSuperImposedExt [▶ 23] bedeutet das:

Distance = 1800 mm - 1500 mm = 300 mm (Abstandskorrektur)

Length = 1000 mm (zur Verfügung stehende Strecke bis zum Ende des Bandes 1)

Mode = SUPERPOSITIONMODE_VELOREDUCTION_LIMITEDMOTION

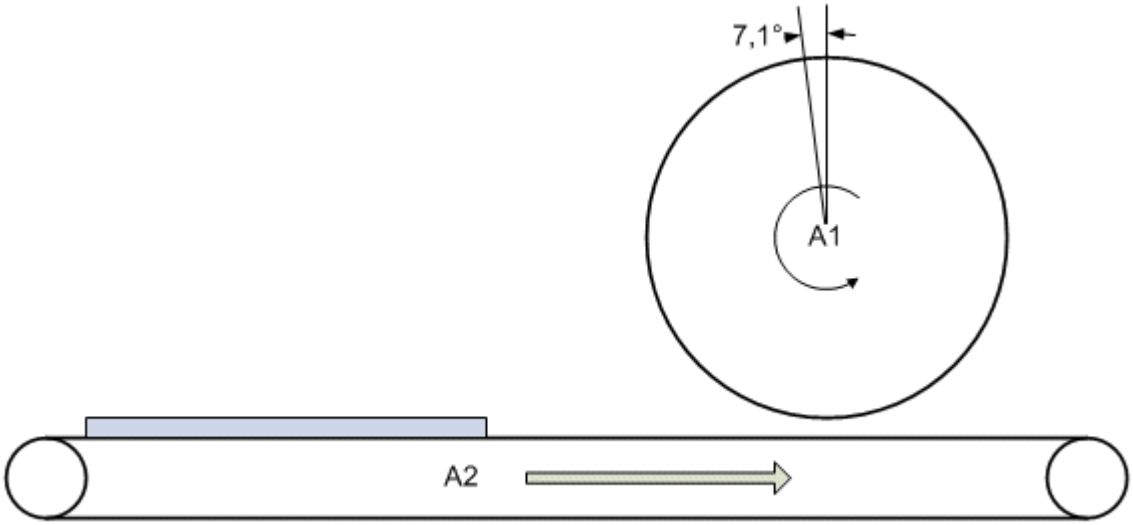
VelocityDiff = 500 mm/s

Der Mode legt fest, dass die Strecke *Length* bis zum Ende des Förderbandes zur Korrektur genutzt wird und dass die Korrektur an dieser Stelle abgeschlossen ist. Als Freiheitsgrad nutzt das System die Geschwindigkeit, die intern berechnet wird. *VelocityDiff* ist in diesem Fall also die Obergrenze für die Geschwindigkeitsänderung.

Alternativ könnte die Korrektur auch durch Abbremsen des Bandes 2 erreicht werden. In diesem Fall muss *Distance* negativ angegeben werden und die zur Verfügung stehende Korrekturstrecke *Length* ist der Abstand des rechten Paketes bis zum Bandende. Die maximal mögliche Geschwindigkeitsänderung *VelocityDiff* entspricht der aktuellen Sollgeschwindigkeit; das Band 2 könnte also, falls notwendig, bis zum Stillstand abgebremst werden.

Phasenverschiebung einer Druckwalze

Eine Druckwalze dreht mit konstanter Umfangsgeschwindigkeit gleich schnell wie ein Transportband auf dem ein zu bedruckendes Werkstück gefördert wird. Die Druckwalze soll zur Synchronisierung mit dem Werkstück um einen Winkel vorpositioniert werden (Phasenverschiebung).



Es gibt hier zwei Möglichkeiten, die Phasenverschiebung durchzuführen. Der Winkel kann so schnell wie möglich korrigiert werden wodurch die Geschwindigkeit der Druckwalze möglichst kurzzeitig und stark erhöht wird. Oder aber man definiert eine Korrekturstrecke innerhalb der die Korrektur stattfinden kann, z. B. eine volle Walzenumdrehung. Daraus ergeben sich folgende möglichen Parametrierungen des Funktionsbausteins MC_MoveSuperImposedExt [▶ 23]:

1. Schnelle Korrektur:

Distance = 7,1°

Length = 360° (maximal mögliche Korrekturstrecke)

Mode = SUPERPOSITIONMODE_LENGTHREDUCTION_LIMITEDMOTION

VelocityDiff = 30°/s (Geschwindigkeitsreserve)

Der *Mode* bestimmt, dass die Korrekturstrecke so weit wie möglich gekürzt wird. Der angegebene Wert für *Length* ist also eine Obergrenze, die frei aber nicht zu knapp gewählt werden kann.

Als *Mode* kann hier alternativ auch SUPERPOSITIONMODE_LENGTHREDUCTION_ADDITIVEMOTION verwendet werden. Die Gesamtstrecke für die Korrektur würde dann bei bis zu 367,1° liegen. Da die Strecke aber möglichst gekürzt wird, sind in diesem Fall beide Modi gleichwertig.

2. Langsame Korrektur:

Distance = 7,1°

Length = 360° (Korrekturstrecke)

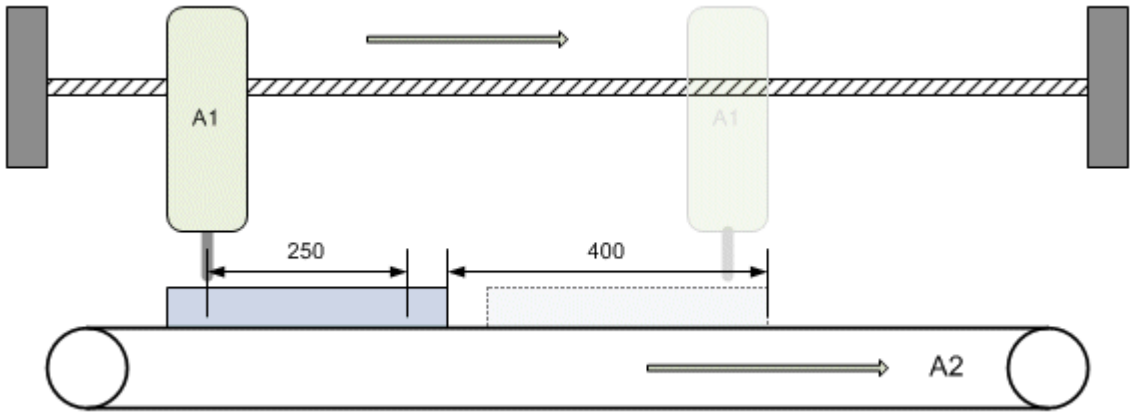
Mode = SUPERPOSITIONMODE_VELOREDUCTION_LIMITEDMOTION

VelocityDiff = 30°/s (Geschwindigkeitsreserve)

Der *Mode* bestimmt, dass die Korrekturstrecke voll genutzt wird und die Geschwindigkeitsänderung möglichst klein gehalten wird. Der angegebene Wert für *VelocityDiff* ist also eine Obergrenze, die frei aber nicht zu knapp gewählt werden kann.

Bohraggregat

Ein Bohraggregat soll während des Transportes in ein Werkstück zwei Bohrungen setzen. Die Synchronisierung für die erste Bohrung für die erste Bohrung soll hier nicht betrachtet werde und wurde z. B. mit der fliegenden Säge durchgeführt (MC_GearInPos). Nach der ersten Bearbeitung muss das Aggregat um eine bestimmte Distanz relativ zum fahrenden Werkstück versetzt werden.



Das Bohraggregat soll nach der ersten Bohrung um 250 mm relativ zum Werkstück vorpositioniert werden. Das Werkstück fährt in der Zeit eine Strecke von 400 mm ab. Ab dieser Position fährt das Bohraggregat wieder mit dem Werkstück synchron und die zweite Bearbeitung kann stattfinden.

Auch hier gibt es zwei mögliche Verfahren, die sich in der Geschwindigkeitsänderung des Bohraggregates und damit auch in der mechanischen Belastung unterscheiden.

Parametrierung des Funktionsbausteins MC_MoveSuperImposedExt [▶ 23]:

1. Schnelle Korrektur:

Distance = 250 mm

Length = 400 mm

Mode = SUPERPOSITIONMODE_LENGTHREDUCTION_ADDITIVEMOTION

VelocityDiff = 500 mm/s (Geschwindigkeitsreserve des Bohraggregats)

Der *Mode* bestimmt, dass die Strecke, auf der die Korrektur durchgeführt wird, so weit wie möglich gekürzt wird. Der angegebene Wert für *Length* ist also eine Obergrenze, die frei aber nicht zu knapp gewählt werden kann. Das Bohraggregat kann eine größere Strecke abfahren, da sich *Length* auf das Werkstück bezieht und relative Positionsänderung hinzukommt.

2. Langsame Korrektur:

Distance = 250 mm

Length = 400 mm

Mode = SUPERPOSITIONMODE_VELOREDUCTION_ADDITIVEMOTION

VelocityDiff = 500 mm/s (Geschwindigkeitsreserve des Bohraggregats)

Der *Mode* bestimmt, dass die Korrekturstrecke voll genutzt wird und die Geschwindigkeitsänderung möglichst klein gehalten wird. Der angegebene Wert für *VelocityDiff* ist also eine Obergrenze, die frei aber nicht zu knapp gewählt werden kann. Das Werkstück fährt während der Positionsänderung des Bohraggregates die Strecke *Length* ab, das Aggregat positioniert wegen der zusätzlichen Korrekturstrecke um 650 mm (*Length* + *Distance*).

3.12 MC_AbortSuperposition

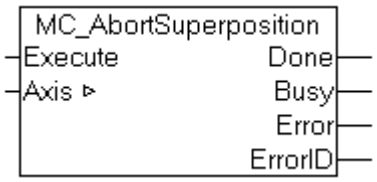


Abb. 2: MC_AbortSuperposition

Der Baustein MC_AbortSuperposition bricht eine durch MC_MoveSuperImposed [▶ 22] oder MC_MoveSuperImposedExt [▶ 23] gestartete überlagerte Bewegung ab, ohne die unterlagerte Achsbewegung zu stoppen.

Ein vollständiger Achsstopp kann gegebenenfalls mit MC_Stop [▶ 32] durchgeführt werden. Ein Aufruf von MC_AbortSuperposition ist dann nicht notwendig.

VAR_INPUT

```
VAR_INPUT
    Execute     : BOOL;
END_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt und das externe Positionslatch wird deaktiviert.

VAR_OUTPUT

```
VAR_OUTPUT
    Done     : BOOL;
    Busy     : BOOL; (* function block is currently busy *)
    Error    : BOOL; (* Signals that an error has occurred within Function Block *)
    ErrorID  : UDINT;    (* Error identification *)
END_VAR
```

Done : Wird TRUE, sobald die überlagerte Bewegung erfolgreich abgebrochen wurde.

Busy : Wird TRUE sobald der Baustein aktiv ist und wird FALSE nachdem er sich wieder im Grundzustand befindet.

Error : Wird TRUE, sobald ein Fehler auftritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

VAR_IN_OUT

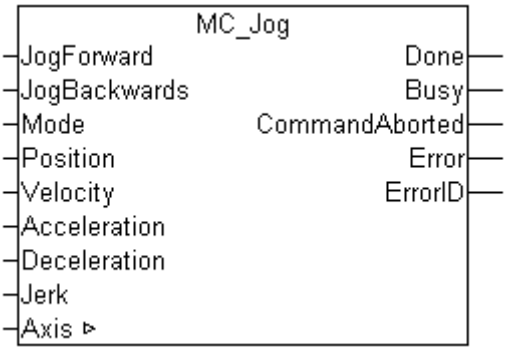
```
VAR_IN_OUT
    Axis        : NCTOPLC_AXLESTRUCT;   (* Identifies the axis which position should be recorded at
a defined event at the trigger input *)
END_VAR
```

Axis : Achsstruktur.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
ab TwinCAT v2.10	PC (i386)	TcMC.Lib

3.13 MC_Jog



Der Funktionsbaustein MC_Jog ermöglicht es, eine Achse mit Handbedientasten zu fahren. Das Tastensignal kann direkt mit den beiden Eingängen JogForward und JogBackwards verbunden werden. Über den Mode-Eingang wird die gewünschte Betriebsart festgelegt. So steht beispielsweise auch ein Inching-Mode zur Verfügung, in dem die Achse mit jedem Tastendruck um einen festgelegten Distanz-Schritt fährt. Je nach Betriebsart können Geschwindigkeit und Dynamik der Bewegung bestimmt werden.

VAR_INPUT

```
VAR_INPUT
    JogForward    : BOOL;
    JogBackwards  : BOOL;
    Mode          : E_JogMode;
    Position      : LREAL;
    Velocity      : LREAL;
    Acceleration  : LREAL;
    Deceleration  : LREAL;
    Jerk          : LREAL;
END_VAR
```

JogForward : Mit der steigenden Flanke wird das Kommando ausgeführt und die Achse wird in positiver Fahrtrichtung bewegt.

Je nach Betriebsart (siehe Mode), fährt die Achse solange das Signal TRUE bleibt oder stoppt automatisch nach einer festgelegten Distanz. Während der Bewegung werden keine weiteren Signalflanken angenommen, auch nicht am Eingang JogBackwards. Bei gleichzeitiger Signalflanke an den Eingängen JogForward und JogBackwards hat JogForward Vorrang.

JogBackwards : Mit der steigenden Flanke wird das Kommando ausgeführt und die Achse wird in negativer Fahrtrichtung bewegt.

JogForward und JogBackwards sollten alternativ getriggert werden, sind aber auch intern gegeneinander verriegelt.

Mode : Der Mode-Eingang legt die Betriebsart fest, in der die Handfunktion ausgeführt wird.

MC_JOGMODE_STANDARD_SLOW

Die Achse wird solange verfahren, wie das Signal an einem der Jog-Eingänge TRUE ist. Dabei wird die im TwinCAT SystemManager festgelegte niedrige Geschwindigkeit für Handfunktionen und die Standarddynamik verwendet. In dieser Betriebsart haben die am Funktionsbaustein angelegten Positions-, Geschwindigkeits- und Dynamikdaten keine Bedeutung.

MC_JOGMODE_STANDARD_FAST

Die Achse wird solange verfahren, wie das Signal an einem der Jog-Eingänge TRUE ist. Dabei wird die im TwinCAT SystemManager festgelegte hohe Geschwindigkeit für Handfunktionen und die Standarddynamik verwendet. In dieser Betriebsart haben die am Funktionsbaustein angelegten Positions-, Geschwindigkeits- und Dynamikdaten keine Bedeutung.

MC_JOGMODE_CONTINOUS

Die Achse wird solange verfahren, wie das Signal an einem der Jog-Eingänge TRUE ist. Dabei werden die vom Anwender angegebenen Geschwindigkeits- und Dynamikdaten verwendet. Die Position hat keine Bedeutung.

MC_JOGMODE_INCHING

Die Achse wird mit steigender Flanke an einem der Jog-Eingänge um eine bestimmte Distanz verfahren, die über den Positions-Eingang festgelegt wird. Die Achse stoppt automatisch, unabhängig vom Zustand der Jog-Eingänge. Erst mit einer weiteren steigenden Flanke wird ein neuer Bewegungsschritt ausgeführt. Mit jedem Start werden die vom Anwender angegebenen Geschwindigkeits- und Dynamikdaten verwendet.

Position : relative Distanz, um die in der Betriebsart *MC_JOGMODE_INCHING* verfahren wird.

Velocity : Maximale Geschwindigkeit mit der gefahren werden soll (>0).

Acceleration : Beschleunigung (≥0). Bei einem Wert von 0 wirkt die Standardbeschleunigung aus der Achskonfiguration im System Manager.

Deceleration : Verzögerung (≥0). Bei einem Wert von 0 wirkt die Standardverzögerung aus der Achskonfiguration im System Manager.

Jerk : Ruck (≥0). Bei einem Wert von 0 wirkt die Standardruck aus der Achskonfiguration im System Manager.

Hinweis:

Die Parameter *Position*, *Velocity*, *Acceleration*, *Deceleration* und *Jerk* werden in den Betriebsarten *MC_JOGMODE_STANDARD_SLOW* und *MC_JOGMODE_STANDARD_FAST* nicht verwendet und können frei bleiben.

VAR_OUTPUT

```
VAR_OUTPUT
    Done            : BOOL;
    Busy            : BOOL;
    CommandAborted  : BOOL;
    Error           : BOOL;
    ErrorID         : UDINT;
END_VAR
```

Done : Wird TRUE wenn eine Bewegung erfolgreich abgeschlossen wurde.

Busy : Wird TRUE sobald der Baustein aktiv ist und wird FALSE nachdem er sich wieder im Grundzustand befindet. Erst dann kann eine weitere Flanke an den Jog-Eingängen angenommen werden.

CommandAborted : Wird TRUE wenn der Vorgang von außen, z. B. durch den Aufruf von MC_Stop [▶ 32], abgebrochen wurde.

Error : Wird TRUE, sobald ein Fehler auftritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

VAR_IN_OUT

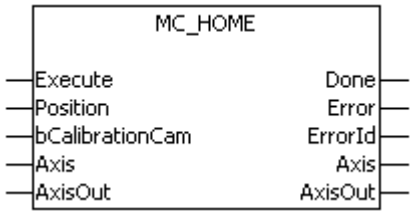
```
VAR_IN_OUT
    Axis        : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Achsstruktur.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
ab TwinCAT v2.10 Build 1303	PC (i386)	TcMC.Lib

3.14 MC_Home



Mit dem Funktionsbaustein MC_Home wird eine Kalibrierung der Achse (Referenzieren) durchgeführt.

Der Referenziermodus wird im TwinCAT SystemManager auf dem Encoder-Reiter *Incremental* eingestellt. Abhängig vom angeschlossenen Encoder-System sind verschiedene Abläufe möglich (siehe auch Referenziermodus für Inkrementalencoder)

VAR_INPUT

```
VAR_INPUT
    Execute         : BOOL;
    Position        : LREAL;
    bCalibrationCam : BOOL;
END_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt.

Position : Position bei Erreichen der Referenznocke. Alternativ kann hier die Konstante DEFAULT_HOME_POSITION verwendet werden. Dadurch wird der im TwinCAT System Manager festgelegte *Referenzposition für Referenzierfahrt* verwendet.

bCalibrationCam : Referenznocke.

VAR_OUTPUT

```
VAR_OUTPUT
    Done    : BOOL;
    Error   : BOOL;
    ErrorID : UDINT;
END_VAR
```

Done : Wird TRUE, wenn die Referenzierung abgeschlossen wurde.

CommandAborted : Wird TRUE, wenn das Kommando nicht vollständig ausgeführt werden konnte. Der Grund kann ein Fehler, ein Stopp oder ein überlagerter Fahrbefehl sein.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

VAR_IN_OUT

```
VAR_IN_OUT
    Axis    : NCTOPLC_AXLESTRUCT;
    AxisOut : PLCTONC_AXLESTRUCT;
END_VAR
```

Axis : Achsstruktur.

AxisOut : Achsstruktur von PLC zur NC.

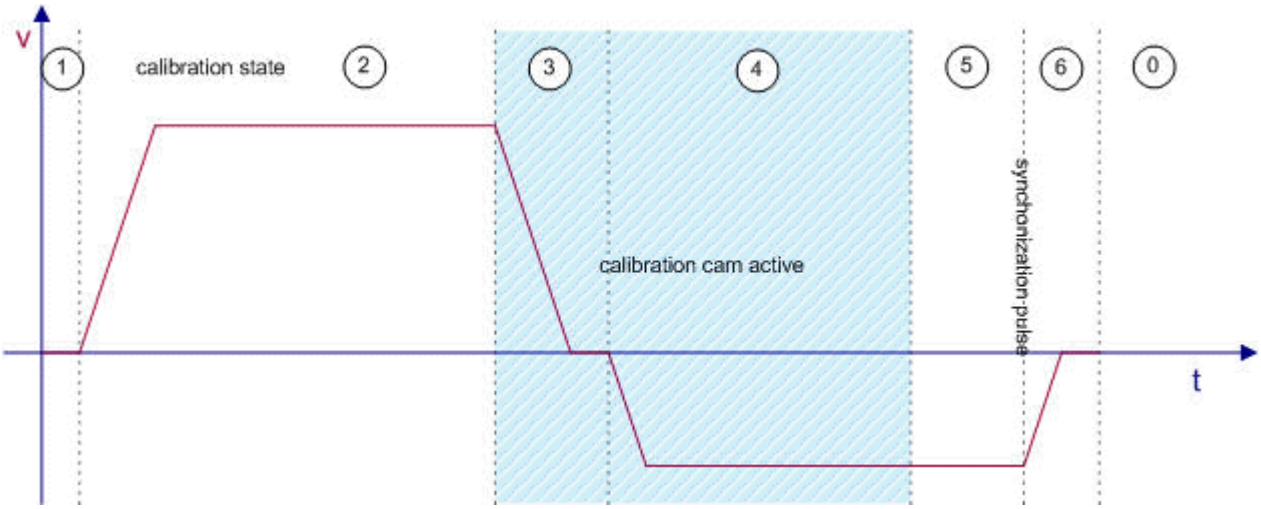
Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcMC.Lib

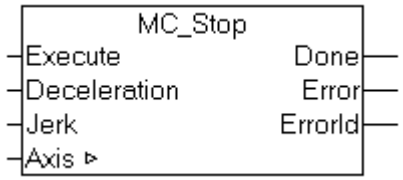
Anmerkung

Der Referenziervorgang läuft in mehreren Phasen ab. Der Referenzierstatus (Calibration State) wird im zyklischen Interface der Achse signalisiert (Axis.nCalibrationState). Im nachfolgenden Bild ist der Ablauf nach dem Start des MC_Home Bausteins mit den einzelnen Phasen schematisch dargestellt.

Soll eine Achse ohne Referenznocke, also nur auf den Sync-Impuls des Gebers, referenziert werden, so kann die Referenznocke durch das SPS-Programm simuliert werden. Das Signal bCalibrationCam wird zunächst aktiviert und dann zurückgenommen, wenn Axis.nCalibrationState größer oder gleich 4 ist.



3.15 MC_Stop



Mit dem Funktionsbaustein MC_Stop wird Achse gestoppt.

VAR_INPUT

```
VAR_INPUT
    Execute      : BOOL;
    Deceleration : LREAL;
    Jerk         : LREAL;
ND_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt.

Deceleration : Verzögerung während der Bremsfahrt.

Ist der Wert 0, so wird die Bremsverzögerung übernommen, mit der die Achse gestartet wurde. Ein Wert ungleich 0 wird nur akzeptiert, wenn gleichzeitig ein Wert für *Jerk* angegeben wird.

Jerk : Ruck während der Bremsfahrt.

Ist der Wert 0, so wird der Ruck übernommen, mit dem die Achse gestartet wurde. Ein Wert ungleich 0 wird nur akzeptiert, wenn gleichzeitig ein Wert für *Deceleration* angegeben wird.

Anmerkungen:

Die Dynamikparameter Deceleration und Jerk werden einer Achse bereits mit dem Startkommando mitgegeben oder, falls diese nicht angegeben wurden, werden die Default-Werte aus dem Dynamikmenü der Achse im TwinCAT SystemManager verwendet. Bis zur TwinCAT Version 2.10 Build 1242 kann die Bremsdynamik durch MC_Stop nur verstärkt werden, d. h. sowohl Deceleration als auch Jerk müssen größer oder gleich der aktuell gültigen Dynamik sein.

Ab TwinCAT 2.10 Build 1243 wurde ein neuer Sollwertgenerator (7 phases optimized) eingeführt welcher in den Global-Parametern einer Achse ausgewählt werden kann. Damit kann beim Aufruf von MC_Stop auch eine schwächere Bremsdynamik vorgegeben werden, um langsamer als ursprünglich vorgesehen anzuhalten. Dabei gibt es aus Sicherheitsgründen Ausnahmen, bei denen die schwächeren Parameter nicht verwendet werden:

- Wenn die anvisierte Zielposition überfahren würde
- Wenn bereits zuvor ein Stopp mit härterer Dynamik ausgelöst wurde
- Wenn die Achse sich in einer Beschleunigungsphase befindet und die Geschwindigkeit sich durch schwächere Dynamikparameter über die parametrierte Fahrgeschwindigkeit hinaus erhöhen würde (Beschleunigung würde zu langsam abgebaut)
- Wenn die Achse sich in einer Bremsphase befindet und sich die Bewegungsrichtung durch schwächere Dynamikparameter umkehren würde (Bremsbeschleunigung kann nicht rechtzeitig abgebaut werden)

In allen Fällen wird die Achse sicher und ohne Fehler gestoppt, auch wenn schwächere Dynamik-Parameter in den oben genannten Fällen nicht akzeptiert werden und die derzeit gültigen härteren Parameter verwendet werden.

VAR_OUTPUT

```
VAR_OUTPUT
    Done    : BOOL;
    Error   : BOOL;
    ErrorID : UDINT;
END_VAR
```

Done : Wird TRUE, wenn die Achse ihren Auftrag beendet hat und steht.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

VAR_IN_OUT

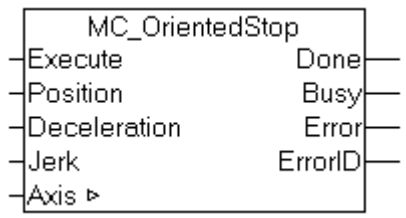
```
VAR_IN_OUT
    Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Achsstruktur.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcMC.Lib

3.16 MC_OrientedStop



Mit dem Funktionsbaustein MC_OrientedStop wird eine Achse an der angegebenen Modulo-Position "orientiert " gestoppt. Je nach Geschwindigkeit und Dynamikparametern der Achse stoppt diese an der nächst möglichen orientierten Position. Im Vergleich zu einem Standard-Stopp mit gleichen Dynamikdaten kann also die Anhalteposition um bis zu eine Modulo-Periode weiter entfernt liegen.

Hinweis: Der Fehlerausgang muss zwingend ausgewertet werden, da unter bestimmten Bedingungen ein orientierter Stopp nicht möglich ist. Beispielsweise könnte kurz zuvor ein Standardstopp ausgelöst worden sein oder es würde im Falle eines orientierten Stopps ein aktiver Software-Endschalter überfahren. In allen Fehlerfällen wird die Achse sicher gestoppt, steht dann aber anschließend nicht an der gewünschten orientierten Position.

VAR_INPUT

```
VAR_INPUT
    Execute         : BOOL;
    Position        : LREAL;
    Deceleration    : LREAL;
    Jerk            : LREAL;
D_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt.

Position : Absolute Modulozielposition an der gestoppt werden soll.
Die Position muss im Bereich [0 ≤ Position < Moduloperiode] liegen. Bei einer Moduloperiode von z. B. 360 Grad ist also 360 keine gültige Position. Stattdessen muss der Wert 0 angegeben werden.

Deceleration : Verzögerung während der Bremsfahrt. Dieser Eingang wird zur Zeit nicht unterstützt. Es wird mit der beim Start der Achse angegebenen Verzögerung gestoppt.

Jerk : Ruck während der Bremsfahrt. Dieser Eingang wird zur Zeit nicht unterstützt. Es wird mit dem beim Start der Achse angegebenen Ruck gestoppt.

VAR_OUTPUT

```
VAR_OUTPUT
    Done        : BOOL;
    Busy        : BOOL;
    Error       : BOOL;
    ErrorID     : UDINT;
END_VAR
```

Done : Wird TRUE, wenn die Achse ihren Auftrag beendet hat und steht.

Busy : Wird TRUE, sobald der Baustein aktiv ist. Busy wird mit der steigenden Flanke am Execute-Eingang TRUE und wird FALSE, wenn die Funktion beendet oder abgebrochen wurde und der Baustein eine neue steigende Flanke am Execute-Eingang akzeptiert.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

VAR_IN_OUT

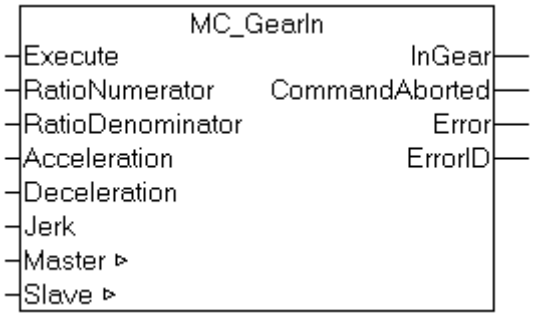
```
VAR_IN_OUT
    Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Achsstruktur.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9 ab Build 1003	PC (i386)	TcMC.Lib

3.17 MC_GearIn



Mit dem Funktionsbaustein MC_GearIn wird ein lineare Master-Slave-Kopplung (Getriebekopplung) aktiviert. Der Baustein akzeptiert einen festen Getriebefaktor im Zähler-Nenner-Format. Alternativ stehen die Bausteine MC_GearInFloat [▶ 36], mit festem Getriebefaktor als LREAL-Wert, und MC_GearInDyn [▶ 37] mit dynamisch änderbarem Getriebefaktor zur Verfügung.

Anmerkung:

Die Kopplung kann nur im Stillstand durchgeführt werden. Die Eingänge Acceleration, Deceleration und Jerk werden daher nicht genutzt und können frei bleiben.

Bewegte Achsen können mit der optionalen Funktion *Fliegende Säge(*MC_GearInPos*, MC_GearInVelo)* durchgeführt werden.

VAR_INPUT

```
VAR_INPUT
    Execute          : BOOL;
    RatioNumerator   : INT;
    RatioDenominator : UINT;
    Acceleration     : LREAL;
    Deceleration     : LREAL;
    Jerk             : LREAL;
END_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt.

RatioNumerator : Getriebefaktor Numerator

RatioDenominator : Getriebefaktor Denominator.

Acceleration : Beschleunigung (nicht verwendet)

Deceleration : Verzögerung (nicht verwendet)

Jerk : Ruck (nicht verwendet)

Bei einem Verhältnis 1:3 muss der Numerator 1 sein und der Denominator 3. Der Numerator kann auch negativ werden.

VAR_OUTPUT

```
VAR_OUTPUT
    InGear         : BOOL;
    CommandAborted : BOOL;
    Error          : BOOL;
    ErrorID        : UDINT;
END_VAR
```

InGear : Wird TRUE, wenn die Kopplung erfolgreich durchgeführt wurde.

CommandAborted : Wird TRUE, wenn die Kopplung nicht durchgeführt werden konnte.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer

VAR_IN_OUT

```
VAR_IN_OUT
    Master : NCTOPLC_AXLESTRUCT;
    Slave  : NCTOPLC_AXLESTRUCT;
END_VAR
```

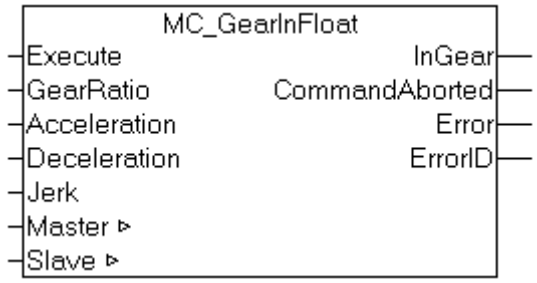
Master : Achsstruktur des Masters.

Slave : Achsstruktur des Slaves.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcMC.Lib

3.18 MC_GearInFloat



Mit dem Funktionsbaustein MC_GearIn wird ein lineare Master-Slave-Kopplung (Getriebekopplung) aktiviert. Der Baustein akzeptiert einen festen Getriebefaktor im LREAL-Format. Alternativ stehen die Bausteine MC_GearIn [▶ 35], mit festem Getriebefaktor im Zähler-Nenner-Format, und MC_GearInDyn [▶ 37] mit dynamisch änderbarem Getriebefaktor zur Verfügung.

Anmerkung:

Die Kopplung kann nur im Stillstand durchgeführt werden. Die Eingänge Acceleration, Deceleration und Jerk werden daher nicht genutzt und können frei bleiben.

Bewegte Achsen können mit der optionalen Funktion *Fliegende Säge(*MC_GearInPos*, MC_GearInVelo)* durchgeführt werden.

VAR_INPUT

```
VAR_INPUT
    Execute      : BOOL;
    GearRatio    : LREAL;
    Acceleration : LREAL;
    Deceleration : LREAL;
    Jerk         : LREAL;
END_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt.

GearRatio : Getriebefaktor als Fließkommazahl

Acceleration : Beschleunigung (nicht verwendet)

Deceleration : Verzögerung (nicht verwendet)

Jerk : Ruck (nicht verwendet)

VAR_OUTPUT

```
VAR_OUTPUT
    InGear         : BOOL;
    CommandAborted : BOOL;
```

```
    Error          : BOOL;
    ErrorID        : UDINT;
END_VAR
```

InGear : Wird TRUE, wenn die Kopplung erfolgreich durchgeführt wurde.

CommandAborted : Wird TRUE, wenn die Kopplung nicht durchgeführt werden konnte.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer

VAR_IN_OUT

```
VAR_IN_OUT
    Master : NCTOPLC_AXLESTRUCT;
    Slave  : NCTOPLC_AXLESTRUCT;
END_VAR
```

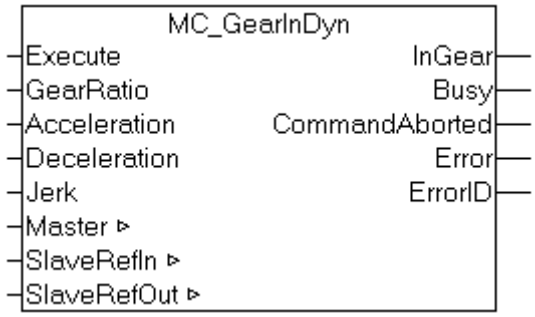
Master : Achsstruktur des Masters.

Slave : Achsstruktur des Slaves.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcMC.Lib

3.19 MC_GearInDyn



Mit dem Funktionsbaustein MC_GearInDyn wird ein lineare Master-Slave-Kopplung (Getriebekopplung) aktiviert. Der Getriebefaktor kann dynamisch, d. h. in jedem SPS-Zyklus angepasst werden. Somit lässt sich eine geregelte Master-Slave-Kopplung aufbauen. Der Parameter Acceleration wirkt begrenzend, falls die Änderungen des Getriebefaktors sehr groß sind.

Anmerkung:

Die Kopplung kann nur im Stillstand durchgeführt werden. Bewegte Achsen können mit der optionalen Funktion *Fliegende Säge(*MC_GearInPos*, MC_GearInVelo)* durchgeführt werden.

VAR_INPUT

```
VAR_INPUT
    Execute      : BOOL;
    GearRatio    : LREAL;
    Acceleration : LREAL;
    Deceleration : LREAL;
    Jerk         : LREAL;
END_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt. Während Execute TRUE ist wird der Getriebefaktor in jedem SPS-Zyklus übernommen. Erst mit fallender Flanke am Execute wird das Kommando beendet.

GearRatio : Getriebefaktor. Der Getriebefaktor kann in jedem SPS-Zyklus geändert werden, solange Execute TRUE ist.

Acceleration : Beschleunigung. Der Parameter begrenzt die Beschleunigung des Slaves bei großen Änderungen des Getriebefaktors. Die maximale Beschleunigung wird erst bei maximaler Master-Geschwindigkeit erreicht, anderenfalls liegt die Slave-Beschleunigung bei großen Getriebefaktoränderungen unterhalb dieses Wertes.

Deceleration : Verzögerung. (Wird zur Zeit nicht genutzt)

Jerk : Ruck. (Wird zur Zeit nicht genutzt)

VAR_OUTPUT

```
VAR_OUTPUT
    InGear          : BOOL;
    Busy            : BOOL;
    CommandAborted : BOOL;
    Error           : BOOL;
    ErrorID         : UDINT;
END_VAR
```

InGear : Wird TRUE, wenn die Kopplung erfolgreich durchgeführt wurde.

Busy : Wird TRUE, sobald der Baustein aktiv ist. Busy wird mit der steigenden Flanke am Execute-Eingang TRUE und wird FALSE, wenn die Funktion beendet oder abgebrochen wurde und der Baustein eine neue steigende Flanke am Execute-Eingang akzeptiert.

CommandAborted : Wird TRUE, wenn die Kopplung aufgelöst wird, während Execute TRUE ist.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer

VAR_IN_OUT

```
VAR_IN_OUT
    Master      : NCTOPLC_AXLESTRUCT;
    SlaveRefIn  : NCTOPLC_AXLESTRUCT;
    SlaveRefOut : PLCTONC_AXLESTRUCT;
END_VAR
```

Master : Achsstruktur des Masters.

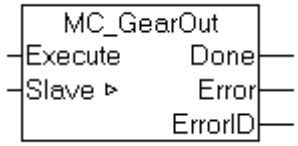
SlaveRefIn : Achsstruktur des Slaves - Daten von der NC and die SPS.

SlaveRefOut : Achsstruktur des Slaves - Daten von der SPS and die NC.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8 ab Build 748 TwinCAT v2.9 ab Build 1000	PC (i386)	TcMC.Lib

3.20 MC_GearOut



Mit dem Funktionsbaustein MC_GearOut wird ein Master-Slave-Kopplung deaktiviert.

Anmerkung: Wenn eine Slave-Achse in der Bewegung abgekoppelt wird, so wird sie nicht automatisch gestoppt, sondern sie erreicht eine konstante Geschwindigkeit mit der sie endlos weiterfährt. Die Achse kann mit einem Stopp-Kommando angehalten werden. (siehe auch MC_GearOutExt [▶ 40])

HINWEIS

Migration zu TwinCAT 2.11

Wenn der Sollwertgeneratortype der Achse auf "7 Phasen (optimiert)" eingestellt ist, wird die Slaveachse nach dem Abkoppeln beschleunigungsfrei gefahren und mit der sich einstellenden konstanten Geschwindigkeit weitergefahren.
Es erfolgt keine Positionierung um den mit dem Koppelfaktor umgerechneten Masterverfahrweg, sondern es stellt sich ein Verhalten wie nach einem MC_MoveVelocity ein. In TwinCAT 2.10 ist der Sollwertgeneratortyp wählbar.
Ab TwinCAT 2.11 ist der Sollwertgeneratortype fest auf "7 Phasen (optimiert)" eingestellt.
Bei der Umstellung eines Projektes von TwinCAT 2.10 auf TwinCAT 2.11 ergibt sich damit das hier beschriebene Verhalten. Ein Update bestehender Applikationen auf Version 2.11 kann daher, je nach Anwendung, eine Anpassung des SPS-Programms erforderlich machen.

VAR_INPUT

```
VAR_INPUT
    Execute : BOOL;
END_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt.

VAR_OUTPUT

```
VAR_OUTPUT
    Done    :       BOOL;
    Error   :          BOOL;
    ErrorID :        UDINT;
END_VAR
```

Done : Wird TRUE, wenn die Entkopplung erfolgreich durchgeführt wurde.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer

VAR_IN_OUT

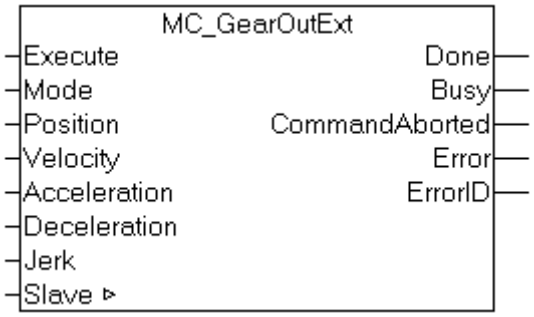
```
VAR_IN_OUT
    Slave : NCTOPLC_AXLESTRUCT;
END_VAR
```

Slave : Achsstruktur des Slaves.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcMC.Lib

3.21 MC_GearOutExt



HINWEIS

Ab TwinCAT 2.11 wird bei einem Abkoppeln die Slaveachse beschleunigungsfrei gefahren und mit der sich einstellenden konstanten Geschwindigkeit weitergefahren. Es erfolgt keine Positionierung um den mit dem Koppelfaktor umgerechneten Masterverfahrweg, sondern es stellt sich ein Verhalten wie nach einem MC_MoveVelocity ein.

MC_GearOutExt ist ein erweitertes universelles Abkoppelkommando, das die Slave-Achse gleich nach dem Abkoppeln mit einem Folgeauftrag versorgt. Somit ist ein nahtloser Übergang aus einer gekoppelten in eine eigenständige Bewegung möglich.

Anmerkung: Das Folgekommando kann nur ausgeführt werden, wenn die Slave-Achse in der Bewegung abgekoppelt wird. Im Stillstand ist kein Folgekommando möglich.

HINWEIS

Migration zu TwinCAT 2.11

Wenn der Sollwertgeneratortype der Achse auf *"7 Phasen (optimiert)"* eingestellt ist, wird die Slaveachse nach dem Abkoppeln beschleunigungsfrei gefahren und mit der sich einstellenden konstanten Geschwindigkeit weitergefahren.
Es erfolgt keine Positionierung um den mit dem Koppelfaktor umgerechneten Masterverfahrweg, sondern es stellt sich ein Verhalten wie nach einem MC_MoveVelocity ein.
In TwinCAT 2.10 ist der Sollwertgeneratortyp wählbar.
Ab TwinCAT 2.11 ist der Sollwertgeneratortype fest auf *"7 Phasen (optimiert)"* eingestellt.
Bei der Umstellung eines Projektes von TwinCAT 2.10 auf TwinCAT 2.11 ergibt sich damit das hier beschriebene Verhalten. Ein Update bestehender Applikationen auf Version 2.11 kann daher, je nach Anwendung, eine Anpassung des SPS-Programms erforderlich machen.

VAR_INPUT

```
VAR_INPUT
    Execute         : BOOL;
    Mode            : E_DecoupleMode;
    Position        : LREAL;
    Velocity        : LREAL;
    Acceleration    : LREAL;
    Deceleration    : LREAL;
    Jerk            : LREAL;
END_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt.

Mode: beschreibt das Folgekommando, das unmittelbar nach dem Abkoppeln der Slave-Achse ausgeführt wird.

```
TYPE E_DecoupleMode :
(
    MC_DECOUPLEMODE_STOP,        (* 0 Stop after decoupling *)
    MC_DECOUPLEMODE_ORIENTEDSTOP,    (* 1 Oriented stop after decoupling *)
    MC_DECOUPLEMODE_ENDLESS,         (* 2 endless motion at actual velocity after decoupling *)
    MC_DECOUPLEMODE_ENDLESS_NEWVELO,     (* 3 endless motion at parameterized velocity after decoupli
ng *)
    MC_DECOUPLEMODE_NEWPOS,          (* 4 stop at parameterized position after decoupling *)
    MC_DECOUPLEMODE_NEWPOSANDVELO,   (* 5 move at parameterized velocity after decoupling and
```

```
                        stop at parameterized position *)
    MC_DECOUPLEMODE_INSTANTANEOUSSTOP       (* 6 instantaneous stop (jump to velocity zero) *)
);
END_TYPE
```

Position : Absolute Zielposition auf die positioniert werden soll. Wenn der Mode *OrientedStop* verwendet wird, ist *Position* eine Modulo-Position

Velocity : Maximale Geschwindigkeit mit der gefahren werden soll (>0).

Acceleration : Beschleunigung (≥0). Bei einem Wert von 0 wirkt die Standardbeschleunigung aus der Achskonfiguration im System Manager.

Deceleration : Verzögerung (≥0). Bei einem Wert von 0 wirkt die Standardverzögerung aus der Achskonfiguration im System Manager.

Jerk : Ruck (≥0). Bei einem Wert von 0 wirkt die Standardruck aus der Achskonfiguration im System Manager.

VAR_OUTPUT

```
VAR_OUTPUT
    Done            : BOOL;
    Busy            : BOOL;
    CommandAborted  : BOOL;
    Error           : BOOL;
    ErrorID         : UDINT;
END_VAR
```

Done : Wird TRUE, wenn die Entkopplung erfolgreich durchgeführt wurde und je nach Folgekommando die Zielposition oder eine konstante Geschwindigkeit erreicht wurde.

Busy : Wird TRUE, sobald der Baustein aktiv ist. Busy wird mit der steigenden Flanke am Execute-Eingang TRUE und wird FALSE, wenn die Funktion beendet oder abgebrochen wurde und der Baustein eine neue steigende Flanke am Execute-Eingang akzeptiert.

CommandAborted : Wird TRUE, wenn das Kommando nicht vollständig ausgeführt werden konnte. Der Grund kann ein Stopp oder ein überlagerter Fahrbefehl sein.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

VAR_IN_OUT

```
VAR_IN_OUT
    Slave       : NCTOPLC_AXLESTRUCT;
END_VAR
```

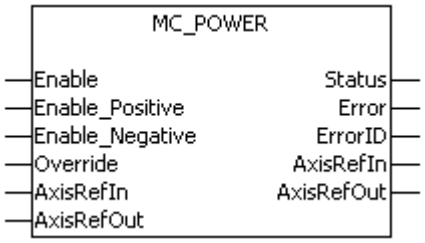
Slave : Achsstruktur des Slaves.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10 Build 1313	PC (i386)	TcMC.Lib

4 Funktionsbausteine Organisation

4.1 MC_Power



Mit dem Funktionsbaustein MC_Power werden die Freigaben für eine Achse gesetzt.

VAR_INPUT

```
VAR_INPUT
    Enable          : BOOL;
    Enable_Positive : BOOL;
    Enable_Negative : BOOL;
    Override        : LREAL;
END_VAR
```

Enable : Solange der Eingang TRUE ist wird das Kommando ausgeführt

Enable_Positive :

Enable_Negative :

Override : Overridewert in Prozent (z. B. als 68.123%)

VAR_OUTPUT

```
VAR_OUTPUT
    Status  : BOOL;
    Error   : BOOL;
    ErrorID : UDINT;
END_VAR
```

Status : Wird TRUE, wenn die Freigaben erfolgreich gesetzt wurden. Dabei wird, soweit möglich, die Betriebsbereitschaft des Antriebs mit einbezogen.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

VAR_IN_OUT

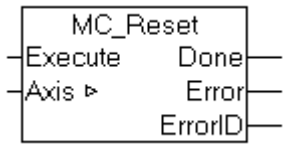
```
VAR_IN_OUT
    AxisRefIn  : NCTOPLC_AXLESTRUCT;
    AxisRefOut : PLCTONC_AXLESTRUCT;
EEND_VAR
```

Axis : Achsstruktur.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcMC.Lib

4.2 MC_Reset



Mit dem Funktionsbaustein MC_Reset wird ein Reset für eine Achse durchgeführt.

VAR_INPUT

```
VAR_INPUT
    Execute :       BOOL;
END_VAR
```

Execute : Mit einer steigenden Flanke wird das Kommando ausgeführt.

VAR_OUTPUT

```
VAR_OUTPUT
    Done    : BOOL;
    Error   : BOOL;
    ErrorID : UDINT;
END_VAR
```

Done : Wird TRUE, sobald das Reset-Kommando fehlerfrei quittiert wurde.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

VAR_IN_OUT

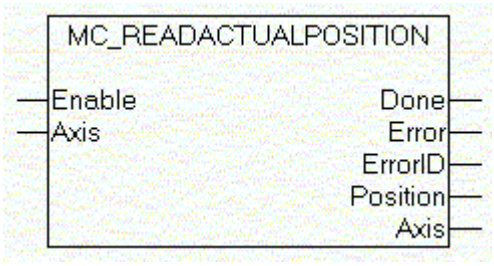
```
VAR_IN_OUT
    Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Achsstruktur.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcMC.Lib

4.3 MC_ReadActualPosition



Mit dem Funktionsbaustein MC_ReadActualPosition kann die aktuelle Position der Achse gelesen werden.

VAR_INPUT

```
VAR_INPUT
    Enable : BOOL;
END_VAR
```

Enable : Solange der Eingang TRUE ist wird das Kommando ausgeführt.

VAR_OUTPUT

```
VAR_OUTPUT
    Done     : BOOL;
    Error    : BOOL;
    ErrorID  : UDINT;
    Position : LREAL;
END_VAR
```

Done : Wird TRUE, wenn die Stati erfolgreich ermittelt wurden.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

Position : Augenblickliche Position der Achse

VAR_IN_OUT

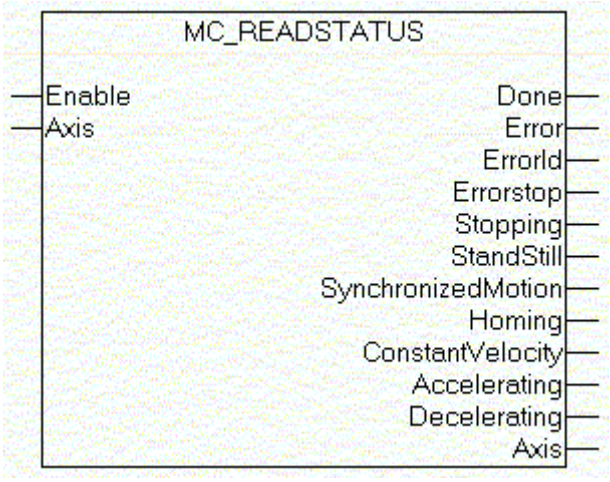
```
VAR_IN_OUT
    Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Achsstruktur.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcMC.Lib

4.4 MC_ReadStatus



Mit dem Funktionsbaustein MC_ReadStatus werden die Stati für eine Achse angezeigt.

VAR_INPUT

```
VAR_INPUT
    Enable : BOOL;
END_VAR
```

Enable : Solange der Eingang TRUE ist wird das Kommando ausgeführt.

VAR_OUTPUT

```
VAR_OUTPUT
    Done                : BOOL;
    Error               : BOOL;
```

```
    ErrorID            : UDINT;
    Errorstop          : BOOL;
    Stopping           : BOOL;
    StandStill         : BOOL;
    DiscreteMotion     : BOOL;
    ContinousMotion    : BOOL;
    SynchronizedMotion : BOOL;
    Homing             : BOOL;
    ConstantVelocity   : BOOL;
    Accelerating       : BOOL;
    Decelerating       : BOOL;
END_VAR
```

Done : Wird TRUE, wenn die Stati erfolgreich ermittelt wurden.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

Errorstop : Wird TRUE, sobald ein Achs-Fehler eintritt.

Stopping : Wird TRUE, sobald die Achse in der Bremsphase ist.

StandStill : Wird TRUE, sobald die Achse in steht.

DiscreteMotion : Wird TRUE, sobald die Achse durch eine diskrete Bewegung verfahren wird.

ContinousMotion : Wird TRUE, sobald die Achse durch eine kontinuierliche Bewegung verfahren wird.

SynchronizedMotion : Wird TRUE, sobald die Achse gekoppelt ist.

Homing : Wird TRUE, sobald die Achse referenziert wird.

ConstantVelocity : Wird TRUE, sobald die Achse mit konstanter Geschwindigkeit fährt.

Acceleration : Wird TRUE, sobald die Achse beschleunigt.

Deceleration : Wird TRUE, sobald die Achse verzögert.

VAR_IN_OUT

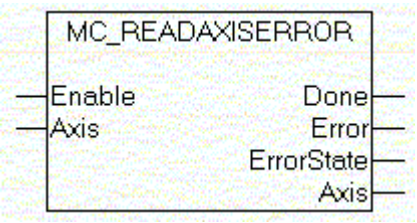
```
VAR_IN_OUT
    Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Achsstruktur.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcMC.Lib

4.5 MC_ReadAxisError



Mit dem Funktionsbaustein MC_ReadAxisError wird der Fehlercode für eine Achse angezeigt.

VAR_INPUT

```
VAR_INPUT
    Enable : BOOL;
END_VAR
```

Enable : Solange der Eingang TRUE ist wird das Kommando ausgeführt.

VAR_OUTPUT

```
VAR_OUTPUT
    Done    : BOOL;
    Error   : BOOL;
    ErrorID : UDINT;
END_VAR
```

Done : Wird TRUE, wenn die Stati erfolgreich ermittelt wurden.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

VAR_IN_OUT

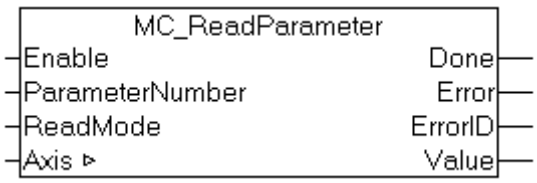
```
VAR_IN_OUT
    Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Achsstruktur.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcMC.Lib

4.6 MC_ReadParameter



Mit dem Funktionsbaustein MC_ReadParameter wird ein Parameter der Achse gelesen.

Hinweis: Mit "Achse" ist in diesem Fall die TwinCAT NC Achse und deren Parameter und nicht der Antrieb gemeint!

VAR_INPUT

```
VAR_INPUT
    Enable          : BOOL;
    ParameterNumber : INT;
    ReadMode        : E_ReadMode
END_VAR
```

Enable : Solange der Eingang TRUE ist wird das Kommando ausgeführt.

ParameterNumber : Nummer [▶ 71] des zu lesenden Parameters.

ReadMode : Lesemodus [▶ 74] des zu lesenden Parameters (einmalig oder zyklisch).

VAR_OUTPUT

```
VAR_OUTPUT
    Done    : BOOL;
    Error   : BOOL;
    ErrorID : UDINT;
    Value   : LREAL;
END_VAR
```

Done : Wird TRUE, wenn die Stati erfolgreich ermittelt wurden.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

Value : Hier wird der gelesene Wert angezeigt.

VAR_IN_OUT

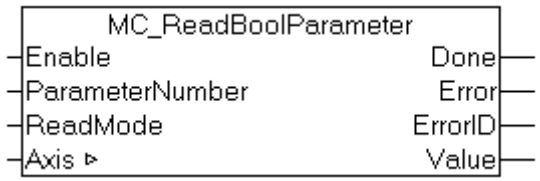
```
VAR_IN_OUT
    Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Achsstruktur.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcMC.Lib

4.7 MC_ReadBoolParameter



Mit dem Funktionsbaustein MC_ReadBoolParameter wird ein boolescher Parameter der Achse gelesen.

Hinweis: Mit "Achse" ist in diesem Fall die TwinCAT NC Achse und deren Parameter und nicht der Antrieb gemeint!

VAR_INPUT

```
VAR_INPUT
    Enable          : BOOL;
    ParameterNumber : INT;
    ReadMode        : E_ReadMode
END_VAR
```

Enable : Solange der Eingang TRUE ist wird das Kommando ausgeführt.

ParameterNumber : Nummer [▶ 71] des zu lesenden Parameters.

ReadMode : Lesemodus [▶ 74] des zu lesenden Parameters (einmalig oder zyklisch).

VAR_OUTPUT

```
VAR_OUTPUT
    Done    : BOOL;
    Error   : BOOL;
    ErrorID : UDINT;
    Value   : LREAL;
END_VAR
```

Done : Wird TRUE, wenn die Stati erfolgreich ermittelt wurden.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

Value : Hier wird der gelesene boolesche Wert angezeigt.

VAR_IN_OUT

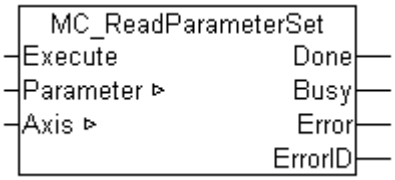
```
VAR_IN_OUT
    Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Achsstruktur.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcMC.Lib

4.8 MC_ReadParameterSet



Mit dem Funktionsbaustein MC_ReadParameterSet kann der gesamte Parameter-Satz einer Achse gelesen werden.

Hinweis: Mit "Achse" ist in diesem Fall die TwinCAT NC Achse und deren Parameter und nicht der Antrieb gemeint!

VAR_INPUT

```
VAR_INPUT
    Execute :       BOOL;
END_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt.

VAR_OUTPUT

```
VAR_OUTPUT
    Done    : BOOL;
    Busy    : BOOL;
    Error   : BOOL;
    ErrorID : UDINT;
END_VAR
```

Done : Wird TRUE, wenn die Parameter erfolgreich gelesen wurden.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

VAR_IN_OUT

```
VAR_IN_OUT
    Parameter : ST_AxisParameterSet;
    Axis      : NCTOPLC_AXLESTRUCT;
END_VAR
```

Parameter : Parameterdatenstruktur vom Typ *ST_AxisParameterSet*.

Axis : Achsstruktur.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10 Build 1251	PC (i386)	TcMC.Lib

4.9 MC_ReadAxisComponents

Mit dem Funktionsbaustein *MC_ReadAxisComponents* werden Informationen zu den Unterelementen Encoder, Drive und Controller einer Achse gelesen.

Hinweis: Mit "Achse" ist in diesem Fall die TwinCAT NC Achse und nicht der Antrieb gemeint!

VAR_INPUT

```
VAR_INPUT
    Execute :       BOOL;
END_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt.

VAR_OUTPUT

```
VAR_OUTPUT
    Done    : BOOL;
    Busy    : BOOL;
    Error   : BOOL;
    ErrorID : UDINT;
END_VAR
```

Done : Wird TRUE, wenn die Parameter erfolgreich gelesen wurden.

Busy : Wird TRUE, sobald der Baustein aktiv ist. Busy wird mit der steigenden Flanke am Execute-Eingang TRUE und wird FALSE, wenn die Funktion beendet oder abgebrochen wurde und der Baustein eine neue steigende Flanke am Execute-Eingang akzeptiert.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

VAR_IN_OUT

```
VAR_IN_OUT
    AxisComponents : ST_AxisComponents;
    Axis           : NCTOPLC_AXLESTRUCT;
END_VAR
```

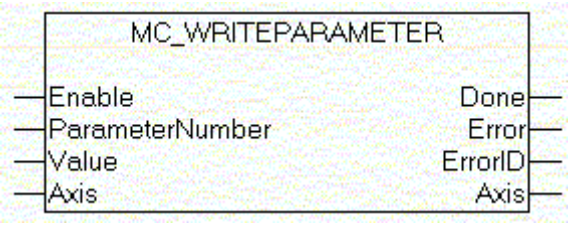
AxisComponents : Parameterdatenstruktur vom Typ *ST_AxisComponents*.

Axis : Achsstruktur.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10 Build 1251	PC (i386)	TcMC.Lib

4.10 MC_WriteParameter



Mit dem Funktionsbaustein MC_WriteParameter können Parameter für die Achse geschrieben werden.

Hinweis: Mit "Achse" ist in diesem Fall die TwinCAT NC Achse und deren Parameter und nicht der Antrieb gemeint!

VAR_INPUT

```
VAR_INPUT
    Enable          : BOOL;
    ParameterNumber : INT;
    Value           : LREAL;
END_VAR
```

Enable : Solange der Eingang TRUE ist wird das Kommando ausgeführt.

ParameterNumber : Nummer [▶ 71] des zu lesenden Parameters.

Value : Dieser LREAL Wert wird geschrieben.

VAR_OUTPUT

```
VAR_OUTPUT
    Done    : BOOL;
    Error   : BOOL;
    ErrorID : UDINT;
END_VAR
```

Done : Wird TRUE, wenn die Stati erfolgreich ermittelt wurden.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

VAR_IN_OUT

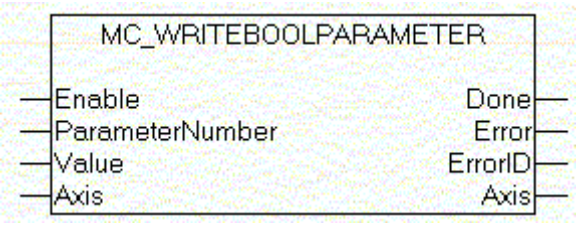
```
VAR_IN_OUT
    Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Achsstruktur.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcMC.Lib

4.11 MC_WriteBoolParameter



Mit dem Funktionsbaustein MC_WriteBoolParameter können boolsche Parameter für die Achse geschrieben werden.

Hinweis: Mit "Achse" ist in diesem Fall die TwinCAT NC Achse und deren Parameter und nicht der Antrieb gemeint!

VAR_INPUT

```
VAR_INPUT
    Enable          : BOOL;
    ParameterNumber : INT;
    Value           : BOOL;
END_VAR
```

Enable : Solange der Eingang TRUE ist wird das Kommando ausgeführt.

ParameterNumber : Nummer [▶ 71] des zu lesenden Parameters.

Value : Dieser BOOL Wert wird geschrieben.

VAR_OUTPUT

```
VAR_OUTPUT
    Done    : BOOL;
    Error   : BOOL;
    ErrorID : UDINT;
END_VAR
```

Done : Wird TRUE, wenn die Stati erfolgreich ermittelt wurden.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

VAR_IN_OUT

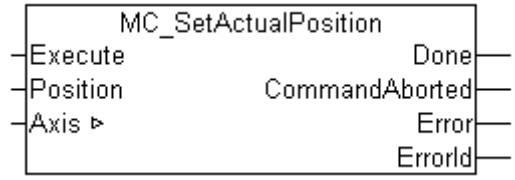
```
VAR_IN_OUT
    Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Achsstruktur.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcMC.Lib

4.12 MC_SetActualPosition



Mit dem Funktionsbaustein MC_SetActualPosition kann die Istposition einer Achse vorgegeben werden.

VAR_INPUT

```
VAR_INPUT
    Execute  : BOOL;
    Position : LREAL;
END_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt.

Position : Ist-Position die gesetzt werden soll.

VAR_OUTPUT

```
VAR_OUTPUT
    Done           : BOOL;
    CommandAborted : BOOL;
    Error          : BOOL;
    ErrorID        : UDINT;
END_VAR
```

Done : Wird TRUE, wenn der Befehl erfolgreich abgesetzt wurde.

CommandAborted : Wird TRUE, wenn das Kommando abgebrochen wurde.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

VAR_IN_OUT

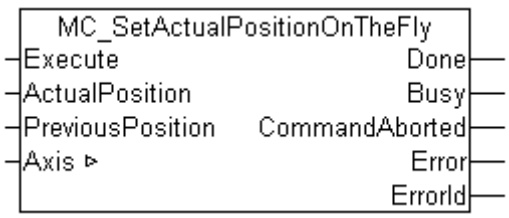
```
VAR_IN_OUT
    Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Achsstruktur.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcMC.Lib

4.13 MC_SetActualPositionOnTheFly



Mit dem Funktionsbaustein MC_SetActualPositionOnTheFly kann die Istposition einer Achse während der Fahrt gesetzt werden. Der aktive Fahrauftrag wird angepasst, so dass das ursprüngliche Fahrziel nicht überfahren wird.

Dieser Baustein kommt z. B. bei speziellen Referenzierfahrten zur Anwendung. Eine genau vermessene Position wird beispielsweise durch ein Hardware-Latch erkannt und das Latch liefert die derzeitige aktuelle Position der Achse an dieser Stelle. Ohne die Achse zu stoppen, kann nun auch zu einem späteren Zeitpunkt die Achsposition an dieser Stelle gesetzt werden, indem neben der gewünschten Position (*ActualPosition*) die Latch-Position als bisherige Referenzposition angegeben wird (*PreviousPosition*)

Anmerkung:

Es ist zu beachten, dass auch die aktuelle Sollposition und die Zielposition der aktuellen Bewegung um dieselbe Differenz (ActualPosition - PreviousPosition) angepasst werden. Dadurch wird exakt die vorher anvisierte Distanz abgefahren aber die Zielposition weicht durch die Verschiebung des Koordinatensystems von der beauftragten Zielposition ab.

VAR_INPUT

```
VAR_INPUT
    Execute          : BOOL;
    ActualPosition   : LREAL;
    PreviousPosition : LREAL;
ND_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt.

ActualPosition : Ist-Position die gesetzt werden soll.

PreviousPosition : Bisherige Ist-Position, die mit der neuen Position korrespondiert.

VAR_OUTPUT

```
VAR_OUTPUT
    Done           : BOOL;
    Busy           : BOOL;
    CommandAborted : BOOL;
    Error          : BOOL;
    ErrorID        : UDINT;
END_VAR
```

Done : Wird TRUE, wenn der Befehl erfolgreich abgesetzt wurde.

Busy : Wird TRUE, sobald der Baustein aktiv ist. Busy wird mit der steigenden Flanke am Execute-Eingang TRUE und wird FALSE, wenn die Funktion beendet oder abgebrochen wurde und der Baustein eine neue steigende Flanke am Execute-Eingang akzeptiert.

CommandAborted : Wird TRUE, wenn das Kommando abgebrochen wurde.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

VAR_IN_OUT

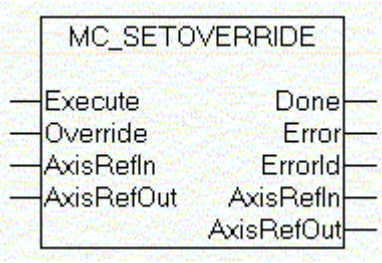
```
VAR_IN_OUT
    Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Achsstruktur.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.10 Build 1251	PC (i386)	TcMC.Lib

4.14 MC_SetOverride



Mit dem Funktionsbaustein MC_SetOverride kann der Override einer Achse vorgegeben werden.

VAR_INPUT

```
VAR_INPUT
    Execute  : BOOL;
    Override : LREAL;
END_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt.

Override : Neuer Overridewert.

VAR_OUTPUT

```
VAR_OUTPUT
    Done    : BOOL;
    Error   : BOOL;
    ErrorID : UDINT;
END_VAR
```

Done : Wird TRUE, wenn der Befehl erfolgreich abgesetzt wurde.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

VAR_IN_OUT

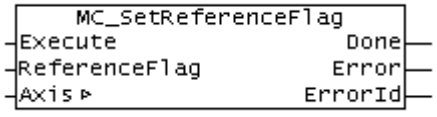
```
VAR_IN_OUT
    AxisRefIn  : NCTOPLC_AXLESTRUCT;
    AxisRefOut : PLCTONC_AXLESTRUCT;
END_VAR
```

Axis : Achsstruktur.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcMC.Lib

4.15 MC_SetReferenceFlag



Mit dem Funktionsbaustein MC_SetReferenceFlag kann der Zustand *Achse ist referenziert* geändert werden. Dieser Baustein wird benötigt, wenn die Achse nicht mit mit dem Baustein MC_Home referenziert wird, sondern über andere Mechanismen eine Referenzposition festgestellt wird (z. B. Fahrt auf Anschlag). Die Referenzposition kann in einem solchen Fall mit dem Funktionsbaustein MC_SetActualPosition [▶ 52] gesetzt werden.

VAR_INPUT

```
VAR_INPUT
    Execute       : BOOL;
    ReferenceFlag : BOOL;
END_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt.

ReferenceFlag : Neuer Referenzierzustand der Achse. TRUE = Achse ist referenziert, FALSE = Achse ist nicht referenziert.

VAR_OUTPUT

```
VAR_OUTPUT
    Done    : BOOL;
    Error   : BOOL;
    ErrorID : UDINT;
END_VAR
```

Done : Wird TRUE, wenn der Befehl erfolgreich abgesetzt wurde.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

VAR_IN_OUT

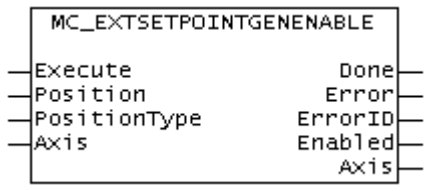
```
VAR_IN_OUT
    Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Achsstruktur.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9	PC (i386)	TcMC.Lib

4.16 MC_ExtSetPointGenEnable



Mit dem Funktionsbaustein MC_ExtSetPointGenEnable kann der externe Sollwertgenerator einer Achse eingeschaltet werden. Anschließend übernimmt die Achse die Sollwertvorgaben aus ihrem zyklischen Achsinterface (fExtSetPos, fExtSetVelo, fExtSetAcc und nExtSetDirection)

VAR_INPUT

```
VAR_INPUT
    Execute      : BOOL;
    Position     : LREAL;
    PositionType : E_TargPosType;
END_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt.

Position : Position für Zielpositionsüberwachung. Das Setzten dieser Position bedeutet nicht, dass die Achse zu diese Position verfährt, dafür ist ausschließlich der externe Sollwertgenerator verantwortlich. Vielmehr wird durch das Setzen dieser Position die Zielpositionsüberwachung aktiviert und das Flag In Zielposition wird TRUE sobald diese Position erreicht wird.

PositionType : Positionstyp

VAR_OUTPUT

```
VAR_OUTPUT
    Done    : BOOL;
    Error   : BOOL;
    ErrorID : UDINT;
    Enabled : BOOL;
END_VAR
```

Done : Wird TRUE, wenn der Befehl erfolgreich abgesetzt wurde.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

Enabled : Enabled zeigt unabhängig von der Funktionsausführung den aktuellen Zustand des externen Sollwertgenerators an.

VAR_IN_OUT

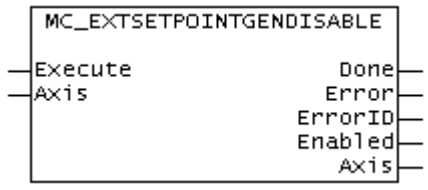
```
VAR_IN_OUT
    Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Achsstruktur.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcMC.Lib

4.17 MC_ExtSetPointGenDisable



Mit dem Funktionsbaustein MC_ExtSetPointGenDisable kann der externe Sollwertgenerator einer Achse ausgeschaltet werden. Anschließend übernimmt die Achse nicht mehr die Sollwertvorgaben aus ihrem zyklischen Achsinterface (fExtSetPos, fExtSetVelo, fExtSetAcc und nExtSetDirection)

VAR_INPUT

```
VAR_INPUT
    Execute : BOOL;
END_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt.

VAR_OUTPUT

```
VAR_OUTPUT
    Done    : BOOL;
    Error   : BOOL;
```

```
    ErrorID : UDINT;
    Enabled : BOOL;
END_VAR
```

Done : Wird TRUE, wenn der Befehl erfolgreich abgesetzt wurde.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

Enabled : Enabled zeigt unabhängig von der Funktionsausführung den aktuellen Zustand des externen Sollwertgenerators an.

VAR_IN_OUT

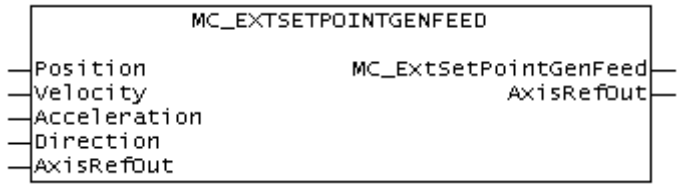
```
VAR_IN_OUT
    Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Achsstruktur.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcMC.Lib

4.18 MC_ExtSetPointGenFeed



Mit der Funktion MC_ExtSetPointGenFeed werden die Sollwerte eines externen Sollwertgenerators in eine Achse eingespeist. Die Funktion kopiert die Daten instantan in das zyklische Achsinterface (fExtSetPos, fExtSetVelo, fExtSetAcc und nExtSetDirection) der Achse. Das Funktionsergebnis MC_ExtSetPointGenFeed ist ungenutzt und daher immer FALSE.

VAR_INPUT

```
VAR_INPUT
    Position     : LREAL;
    Velocity     : LREAL;
    Acceleration : LREAL;
    Direction    : DINT;
END_VAR
```

Position : Sollposition aus einem externen Sollwertgenerator

Velocity : Sollgeschwindigkeit aus einem externen Sollwertgenerator

Acceleration : Sollbeschleunigung aus einem externen Sollwertgenerator

Direction : Sollrichtung aus einem externen Sollwertgenerator. (-1 = negative Richtung, 0 = Stillstand, 1 = positive Richtung)

VAR_IN_OUT

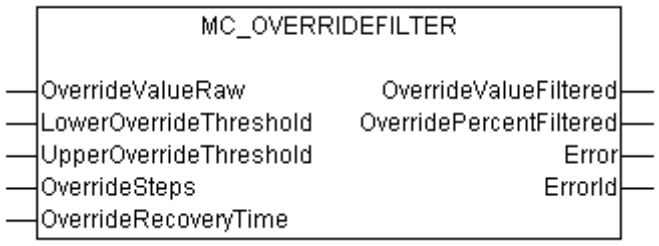
```
VAR_IN_OUT
    AxisRefOut : PLCTONC_AXLESTRUCT;
END_VAR
```

Axis : Achsstruktur.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcMC.Lib

4.19 MC_OverrideFilter



Mit dem Funktionsbaustein MC_OverrideFilter kann ein ungefilterter Overridewert in Digits (z.B. ein Spannungswert einer Analogeingangsklemme) in einen gefilterten, für das zyklische Achsinterface (PlcToNc) passenden, Overidewert (DWORD im Wertebereich 0...1000000) umgerechnet werden. Ebenfalls steht dieser gefilterte Override auch in Prozent zur Verfügung (LREAL im Wertebereich 0...100%).
Der rohe Eingangswert wird dabei durch *LowerOverrideThreshold* und *UpperOverrideThresholdauf* auf einen Gültigkeitsbereich begrenzt und in eine parametrierbare Stufung (Auflösung) umgesetzt (*OverrideSteps*). Nach jeder Overrideänderung am Ausgang des FB's wird intern eine Mindestruhezeit abgewartet (*OverrideRecoveryTime*), bevor wieder ein neuer Overridewert übernommen werden kann. Die einzigen Ausnahmen stellen die Overridewerte 0% und 100%, die aus Sicherheitsgründen immer ohne Zeitverzögerung umgesetzt werden.

Hinweis: Aufgrund der Stufung des ausgegebenen Override-Wertes (*OverrideValueFiltered*) kann es bei sehr kleinen Override-Eingangswerten (*OverrideValueRaw*) vorkommen, dass der gefilterte Override zu Null wird. Ein Override Null führt zum Stillstand der Achse. Falls ein vollständiger Stillstand nicht erwünscht ist, sollte *OverrideValueRaw* nicht unter die kleineste Stufung fallen.

VAR_INPUT

```
VAR_INPUT
    OverrideValueRaw       : DINT;
    LowerOverrideThreshold : DINT  := 0;       (* 0...32767 digits *)
    UpperOverrideThreshold : DINT  := 32767;          (* 0...32767 digits *)
    OverrideSteps          : UDINT := 200;     (* 200 steps => 0.5 percent *)
    OverrideRecoveryTime   : TIME  := T#150ms;     (* 150 ms *)
END_VAR
```

OverrideValueRaw: Rohe, ungefilterter Overridewert (z.B. ein Spannungswert einer Analogeingangsklemme).

LowerOverrideThreshold: Die untere Schranke, die den rohen Overridewert begrenzt.

UpperOverrideThreshold: Die obere Schranke, die den rohen Overridewert begrenzt.

OverrideSteps: Die vorgegebene Stufung (Overrideauflösung).

OverrideRecoveryTime: Mindestruhezeit nach der ein neuer gefilterter Overridewert auf den Ausgang gelegt wird. Die Overridewerte 0% und 100% werden allerdings ohne Zeitverzögerung umgesetzt.

VAR_OUTPUT

```
VAR_OUTPUT
    OverrideValueFiltered   : DWORD;  (* 0...1000000 counts *)
    OverridePercentFiltered : LREAL;  (* 0...100 % *)
    Error                   : BOOL;
    ErrorId                 : UDINT;
END_VAR
```

OverrideValueFiltered: Der gefilterte Overridewert in Digits (der Datentyp passt zum Override im zyklischen Achsinterface 0..1000000).

OverridePercentFiltered: Der gefilterte Overridewert in Prozent (0..100%).

Error: Wird TRUE, sobald ein Fehler eintritt.

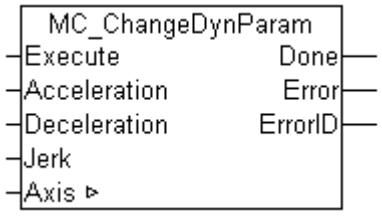
ErrorID: Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

Mögliche Fehlernummer	Mögliche Ursachen
MC_ERROR_PARAMETER_NOT_CORRECT	• OverrideSteps <= 1 • LowerOverrideThreshold >= UpperOverrideThreshold

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8 Build > 739	PC (i386)	TcMC.Lib

4.20 MC_ChangeDynParam



MC_ChangeDynParam ändert die Beschleunigungs- und Verzögerungsparameter für eine aktive Positionierung.

Voraussetzungen

Das Ändern der aktuellen Dynamik ist möglich, wenn der Sollwertgenerator-Typ der Achse auf "7 Phasen (optimized)" eingestellt ist (Globale Achsparameter). Anderenfalls ist eine Änderung nur sehr eingeschränkt möglich.

Einschränkungen

In einigen Situationen können aus Sicherheitsgründen keine neuen Dynamikparameter akzeptiert werden und es werden die aktuellen Dynamikparameter beibehalten:

- Wenn die anvisierte Zielposition überfahren würde (Fehler 16#427F)
- Wenn bereits zuvor ein Stopp mit härterer Dynamik ausgelöst wurde (Fehler 16#425D)
- Wenn die Achse sich in einer Beschleunigungsphase befindet und die Geschwindigkeit sich durch schwächere Dynamikparameter über die parametrierte Fahrgeschwindigkeit hinaus erhöhen würde (Beschleunigung würde zu langsam abgebaut) (Fehler 16#423A)
- Wenn die Achse sich in einer Bremsphase befindet und sich die Bewegungsrichtung durch schwächere Dynamikparameter umkehren würde (Bremsbeschleunigung kann nicht rechtzeitig abgebaut werden) (Fehler 16#4289)
- Sonderfall: interne Optimierung ist nicht möglich (Fehler 16#427E)

VAR_INPUT

```
VAR_INPUT
    Execute         : BOOL;
    Acceleration    : LREAL;
    Deceleration    : LREAL;
    Jerk        : LREAL;
END_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt. Während der Bewegung kann der Baustein nachgetriggert werden.

Acceleration : Beschleunigung (≥0). Bei einem Wert von 0 wirkt die Standardbeschleunigung aus der Achskonfiguration im System Manager

Deceleration : Verzögerung (≥0). Bei einem Wert von 0 wirkt die Standardbeschleunigung aus der Achskonfiguration im System Manager

Jerk : Ruck (≥0). Bei einem Wert von 0 wirkt die Standardruck aus der Achskonfiguration im System Manager.

Anmerkung: Der Ruck kann nur geändert werden, wenn der Sollwertgenerator-Typ der Achse auf "7 Phasen (optimized)" eingestellt ist. (ab TwinCAT 2.10 Build 1251)

VAR_OUTPUT

```
VAR_OUTPUT
    Done        : BOOL;
    Error       : BOOL;
    ErrorID     : UDINT;
END_VAR
```

Done : Wird TRUE, wenn die Achse im Ziel ist und alle aktivierten Überwachungen (z. B. Zielpositionsüberwachung) positiv sind.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

VAR_IN_OUT

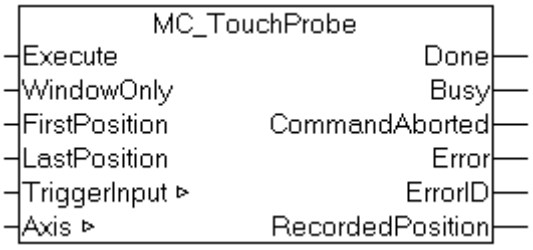
```
VAR_IN_OUT
    Axis : NCTOPLC_AXLESTRUCT;
END_VAR
```

Axis : Achsstruktur.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8 Build	PC (i386)	TcMC.Lib

4.21 MC_TouchProbe



Der Baustein MC_TouchProbe erfasst eine Achsposition zum Zeitpunkt eines digitalen Signals (Messtasterfunktion). Die Position wird üblicherweise nicht direkt in der SPS-Umgebung, sondern über ein externes Hardware-Latch erfasst und ist damit hochgenau und Zykluszeit-unabhängig. Der Baustein steuert diesen Mechanismus und ermittelt die extern erfasste Position.

Voraussetzungen

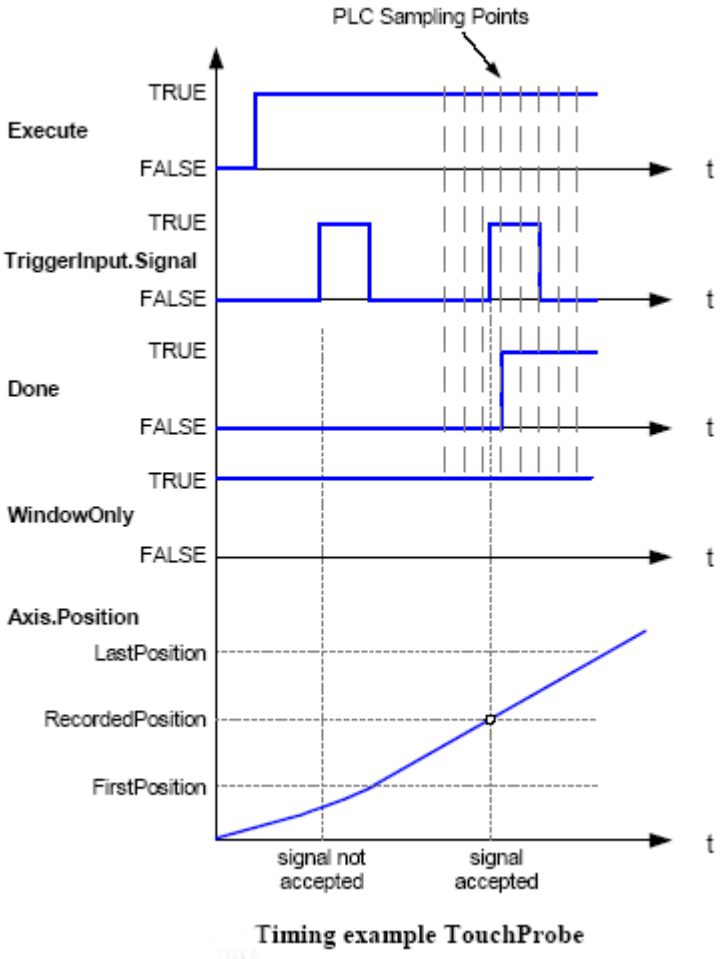
Vorraussetzung für die Positionserfassung ist eine geeignete Encoder-Hardware, die in der Lage ist, die erfasste Position zu latchen. Unterstützt werden beipielsweise SERCOS-Antriebe, Beckhoff AX2000 mit SERCOS- und Lightbus-Schnittstelle und Beckhoff Encoder-Klemmen KL5101. Dass digitale Trigger-Signal wird mit dieser Hardware verdrahtet und führt unabhängig vom SPS-Zyklus zum Aufzeichnen der aktuellen Achsposition.

Teilweise müssen diese Endgeräte konfiguriert werden, damit eine Positionserfassung möglich ist. Lesen Sie dazu Messtasterauswertung mit AX2xxx-B200 (Lightbus) und Messtasterauswertung mit AX2xxx-B750 (SERCOS).

Hinweis

Nachdem ein Messtasterzyklus durch eine steigende Flanke am *Execute* Eingang gestartet wurde, wird dieser erst beendet, wenn die Ausgänge *Done*, *Error* oder *CommandAborted* TRUE werden. Soll der Vorgang zwischenzeitlich abgebrochen werden, so muss der Baustein MC_AbortTrigger [▶ 63] mit derselben TriggerInput [▶ 63] Datenstruktur aufgerufen werden. Anderenfalls kann kein neuer Zyklus gestartet werden.

Signalverlauf



Timing example TouchProbe

VAR_INPUT

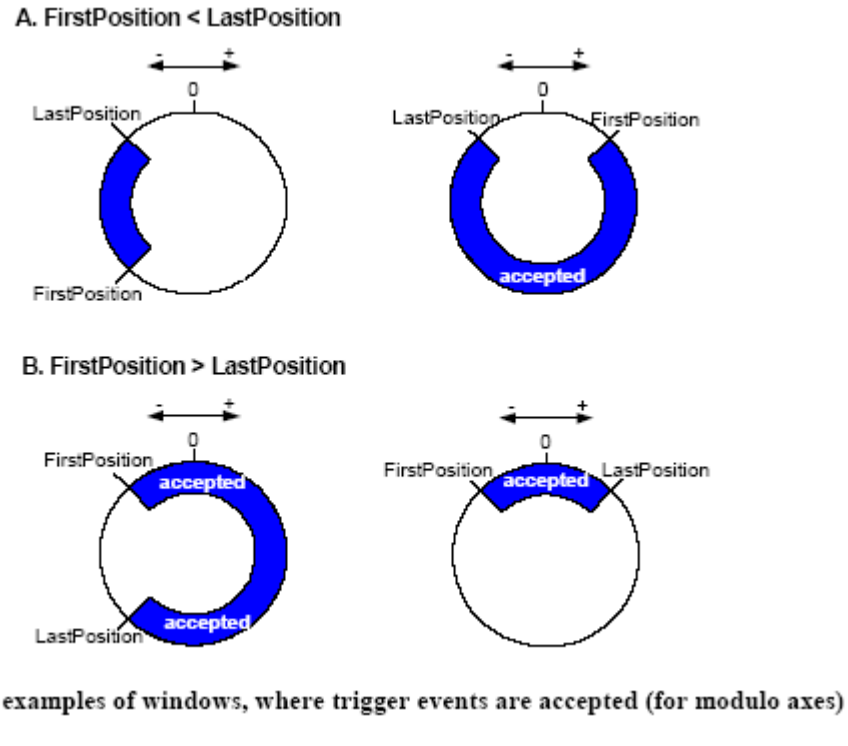
```
VAR_INPUT
    Execute         : BOOL; (* Starts touch probe recording *)
    WindowOnly      : BOOL; (* Enables the monitoring window *)
    FirstPosition   : LREAL; (* Beginning of the monitoring window *)
    LastPosition    : LREAL; (* Ending of the monitoring window *)
END_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt und das externe Positionslatch wird aktiviert.

WindowOnly : Wenn diese Option aktiv ist, wird nur eine Position innerhalb des Fensters zwischen *FirstPosition* und LastPosition erfasst. Positionen außerhalb des Fensters werden verworfen und das externe Positionslatch wird automatisch neu aktiviert. Erst wenn die erfasste Position innerhalb des Fensters liegt, wird *Done* TRUE.
Das Erfassungsfenster kann absolut oder modulo interpretiert werden. Dazu ist das Flag ModuloPositions [▶ 72] in der Struktur TriggerInput [▶ 72] entsprechend zu setzen. Bei absoluten Positionen gibt es exakt ein Fenster. Bei Modulo-Positionen wiederholt sich das Fenster innerhalb des in den Achsparametern festgelegten Modulozyklus (z. B. 0 bis 360 Grad).

FirstPosition : Anfangsposition des Erfassungsfensters, wenn WindowOnly TRUE ist. Diese Position kann absolut oder modulo interpretiert werden. Dazu ist in der Struktur *TriggerInput* (siehe unten) das Flag ModuloPositions [▶ 72] entsprechend zu setzen.

LastPosition : Endposition des Erfassungsfensters, wenn WindowOnly TRUE ist. Diese Position kann absolut oder modulo interpretiert werden. Dazu ist in der Struktur *TriggerInput* (siehe unten) das Flag ModuloPositions [▶ 72] entsprechend zu setzen.



examples of windows, where trigger events are accepted (for modulo axes)

VAR_OUTPUT

```
VAR_OUTPUT
    Done              : BOOL; (* move completed *)
    Busy              : BOOL; (* function block is currently busy *)
    CommandAborted    : BOOL;
    Error             : BOOL; (* Signals that an error has occured within Function Block *)
    ErrorID           : UDINT; (* Error identification *)
    RecordedPosition  : LREAL; (* Position where the trigger event occured *)
END_VAR
```

Done : Wird TRUE, wenn eine Achsposition erfolgreich erfasst wurde. Die Position wird am Ausgang *RecordedPosition* ausgegeben.

Busy : Wird TRUE sobald der Baustein aktiv ist und wird FALSE nachdem er sich wieder im Grundzustand befindet.

CommandAborted : Wird TRUE wenn der Vorgang von außen, z. B. durch den Aufruf von MC_AbortTrigger [▶ 63], abgebrochen wurde.

Error : Wird TRUE, sobald ein Fehler auftritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

RecordedPosition : Erfasste Achsposition zum Zeitpunkt des Trigger-Signals

VAR_IN_OUT

```
VAR_IN_OUT
    TriggerInput : MC_InputRef;         (* Reference to the trigger signal source. *)
    Axis         : NCTOPLC_AXLESTRUCT;  (* Identifies the axis which position should be recorded at
a defined event at the trigger input *)
END_VAR
```

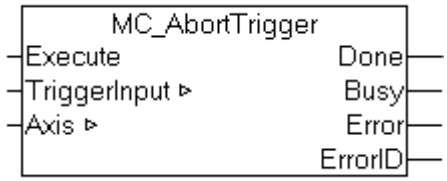
TriggerInput : Datenstruktur vom Typ MC_InputRef [▶ 72], die den Trigger-Eingang für die Erfassung der Achsposition beschreibt.

Axis : Achsstruktur.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
ab TwinCAT v2.9 Build 1000	PC (i386)	TcMC.Lib

4.22 MC_AbortTrigger



Der Baustein MC_AbortTrigger bricht einen durch MC_TouchProbe [▶ 60] gestarteten Messtasterzyklus ab. MC_TouchProbe startet einen Messtasterzyklus, indem ein Positionslatch in einer externen Encoder-Hardware aktiviert wird. Soll der Vorgang beendet werden, bevor das Trigger-Signal das Positionslatch aktiviert hat, so kann dazu MC_AbortTrigger verwendet werden. Wurde der Messtasterzyklus erfolgreich beendet, so ist es nicht notwendig diesen Baustein aufzurufen.

VAR_INPUT

```
VAR_INPUT
    Execute     : BOOL; (* Starts touch probe recording *)
END_VAR
```

Execute : Mit der steigenden Flanke wird das Kommando ausgeführt und das externe Positionslatch wird deaktiviert.

VAR_OUTPUT

```
VAR_OUTPUT
    Done     : BOOL; (* move completed *)
    Busy     : BOOL; (* function block is currently busy *)
    Error    : BOOL; (* Signals that an error has occured within Function Block *)
    ErrorID  : UDINT;    (* Error identification *)
END_VAR
```

Done : Wird TRUE, sobald der Messtasterzyklus erfolgreich abgebrochen wurde.

Busy : Wird TRUE sobald der Baustein aktiv ist und wird FALSE nachdem er sich wieder im Grundzustand befindet.

Error : Wird TRUE, sobald ein Fehler auftritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

VAR_IN_OUT

```
VAR_IN_OUT
    TriggerInput : MC_InputRef;          (* Reference to the trigger signal source. *)
    Axis         : NCTOPLC_AXLESTRUCT;   (* Identifies the axis which position should be recorded at
 a defined event at the trigger input *)
END_VAR
```

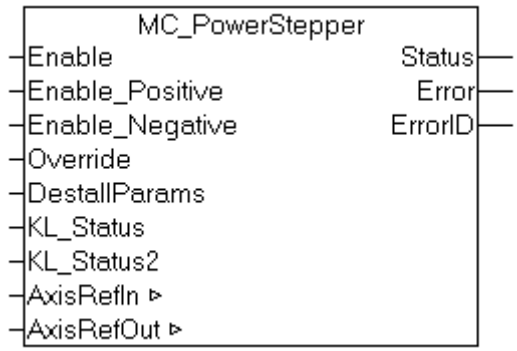
TriggerInput : Datenstruktur vom Typ MC_InputRef [▶ 72], die den Trigger-Eingang für die Erfassung der Achsposition beschreibt. Hier muss dieselbe Datenstruktur wie am korrespondierenden MC_TouchProbe [▶ 60] Baustein verwendet werden.

Axis : Achsstruktur.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
ab TwinCAT v2.9 Build 1000	PC (i386)	TcMC.Lib

4.23 MC_PowerStepper



Mit dem Funktionsbaustein MC_PowerStepper werden die Freigaben für eine Achse gesetzt. Dazu wird intern ein MC_Power Baustein verwendet. Zusätzlich erkennt der MC_PowerStepper die bei Schrittmotoren im Überlastfall auftretenden Stall-Situationen und bietet geeignete Gegenmaßnahmen an. Die Stausbits einer KL2531 oder KL2541 Klemme werden überwacht und die dort signalisierten Fehler an die NC gemeldet.

Weitere Erläuterungen im Anhang [▶ 65].

VAR_INPUT

```
VAR_INPUT
    Enable          : BOOL;
    Enable_Positive : BOOL;
    Enable_Negative : BOOL;
    Override        : LREAL;
    DestallParams   : ST_PowerStepperStruct;
    KL_Status       :    USINT;
    KL_Status2      : UINT;
END_VAR
```

Enable : NC-Reglerfreigabe für die Achse.

Enable_Positive : NC-Vorschubfreigabe in positiver Richtung.

Enable_Negative : NC-Vorschubfreigabe in negativer Richtung.

Override : Overridewert in Prozent (z. B. als 68.123%)

DestallParams : Hier werden die Funktionen des Bausteins freigegeben und ihre Arbeitsregeln [▶ 75] festgelegt.

KL_Status : Das Statusbyte einer Klemme des Typs KL2531 oder KL2541.

KL_Status2 : Das Statuswort einer Klemme des Typs KL2531 oder KL2541.

VAR_OUTPUT

```
VAR_OUTPUT
    Status  : BOOL;
    Error   : BOOL;
    ErrorID : UDINT;
END_VAR
```

Status : Wird TRUE, wenn die Freigaben erfolgreich gesetzt wurden.

Error : Wird TRUE, sobald ein Fehler eintritt.

ErrorID : Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

VAR_IN_OUT

```
VAR_IN_OUT
    AxisRefIn  : NCTOPLC_AXLESTRUCT;
    AxisRefOut : PLCTONC_AXLESTRUCT;
EEND_VAR
```

AxisRefIn : Die von der NC bereitgestellte Achsstruktur.

AxisRefOut : Die von der SPS bereitgestellte Achsstruktur.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9 ab Build 1026	PC (i386)	TcMC.Lib

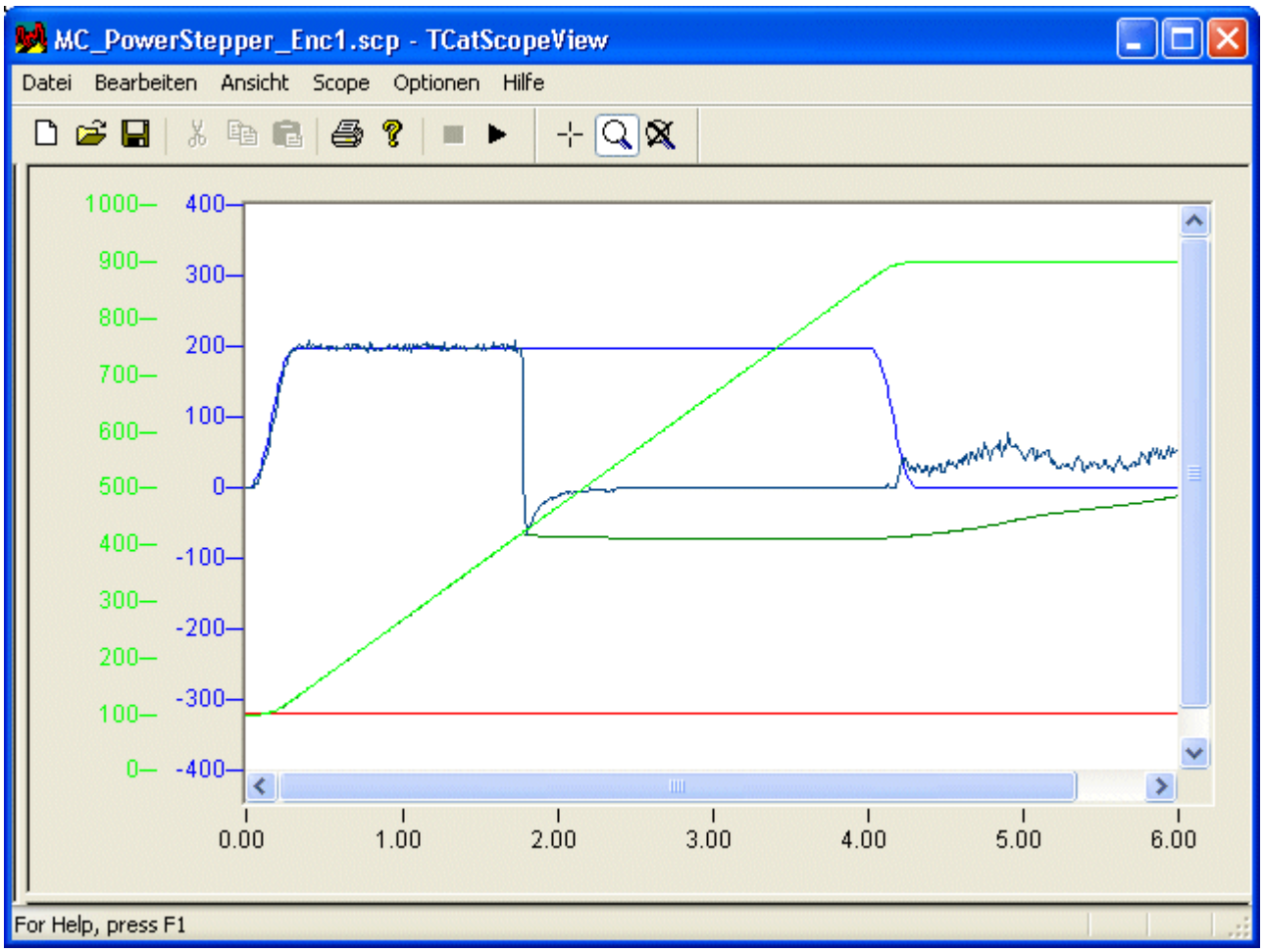
4.23.1 Hinweise zu MC_PowerStepper

Mit dem Funktionsbaustein MC_PowerStepper [▶ 64] werden die Freigaben und der Override für eine Achse gesetzt. Dazu wird intern ein MC_Power Baustein verwendet. Zusätzlich erkennt der MC_PowerStepper die bei Schrittmotoren im Überlastfall auftretenden Stall-Situationen und bietet geeignete Gegenmaßnahmen an. Die Stausbits einer KL2531 oder KL2541 Klemme werden überwacht und die dort signalisierten Fehler an die NC gemeldet.

Schrittmotor und Synchron-Servo: Gemeinsamkeiten und Unterschiede

Beide Motortypen verwenden ein elektromagnetisches und ein permanentmagnetisches Feld, um durch deren Zusammenspiel eine Antriebskraft zu erzeugen. Während jedoch beim Servo eine kostenaufwändige Sensorik eingesetzt wird, um die Ausrichtung der Felder gezielt zu kontrollieren (rotorlageorientierte Bestromung) wird beim Schrittmotor auf diese Orientierung verzichtet. Dadurch ist eine deutliche Kosteneinsparung erzielbar. Allerdings entsteht so die Möglichkeit, dass der Motor durch eine äußere Kraft über den Punkt des maximalen von ihm erzeugten Moments hinaus bewegt wird. Da das elektromagnetisch erzeugte Feld dies nicht berücksichtigt wird er bei weiter steigender Auslenkung jetzt ein absinkendes Rückstellmoment erzeugen. Dies führt dazu, dass ab einer Auslenkung von mehr als einem halben Polspiel das Rückstellmoment im Vorzeichen wechselt und den Motor zum nächsten Pol mit entsprechender Richtung zieht. Je nach jetzt geltenden Bedingungen kann der Motor jetzt in der neuen Orientierung einrasten (der so genannte Schrittverlust ist eingetreten) oder auch hier den gleichen Vorgang wiederholen. Letzteres wird als Stalling bezeichnet und wird vor allem dann auftreten, wenn der Motor mit typischen Frequenzen des aktiven Fahrbetriebs bestromt wird.

Beispiel 1: Ein mit einem Encoder ausgerüsteter Schrittmotor wird mit für Servos üblichen Parametern mit der NC PTP verfahren.

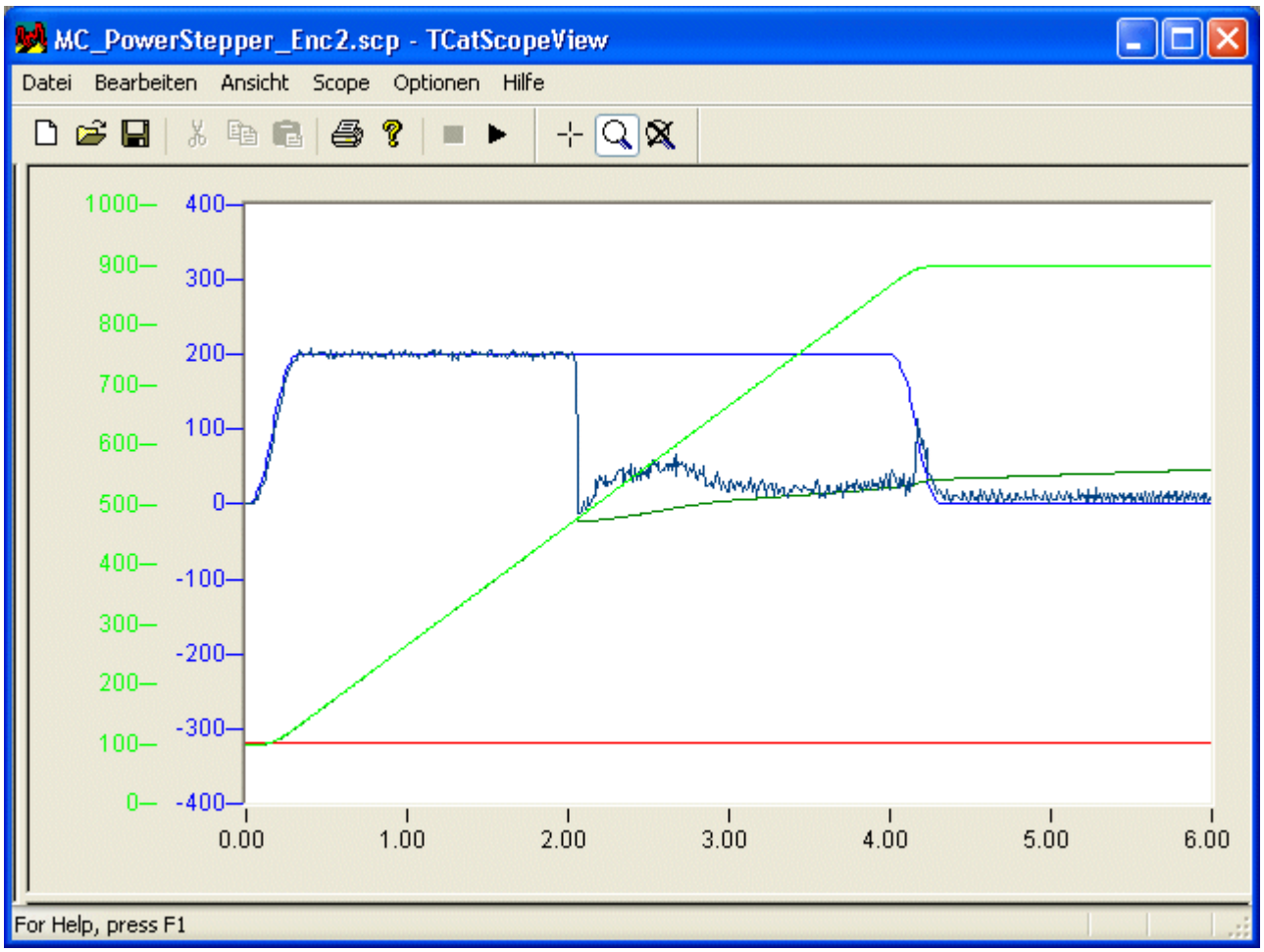


bStalled
SetVelo
ActualPos
ActualVelo
SetPos

Bei etwa 1.8 Sekunden wird die Achse von einem Hindernis kurzzeitig blockiert. Obwohl die Achse anschließend frei beweglich ist kann sie dem Geschwindigkeitssollwert nicht folgen, sondern steht mit erheblicher Geräuschentwicklung und ohne ein erkennbares Moment zu erzeugen still. Erst nachdem der Profilgenerator sein Fahrziel erreicht hat sinkt die Summe aus Soll- und Korrekturgeschwindigkeit ab. Der Motor setzt sich in diesem Beispiel in unregelmäßiger Weise in Bewegung. Schon ein geringes Lastmoment würde dies jedoch verhindern. Die einzige Lösung würde hier ein MC_Reset [▶ 43] und eine angemessene Beruhigungszeit sein. Anschließend müsste die Achse durch die Applikation erneut gestartet werden. Dabei würden im Achsinterface diverse Zustandsbits reagieren. Dies ist in der Applikation entsprechend zu berücksichtigen, da es sonst zu fehlerhaften Reaktionen im Maschinenablauf kommen kann.

Erste Maßnahme: Reglerbegrenzung

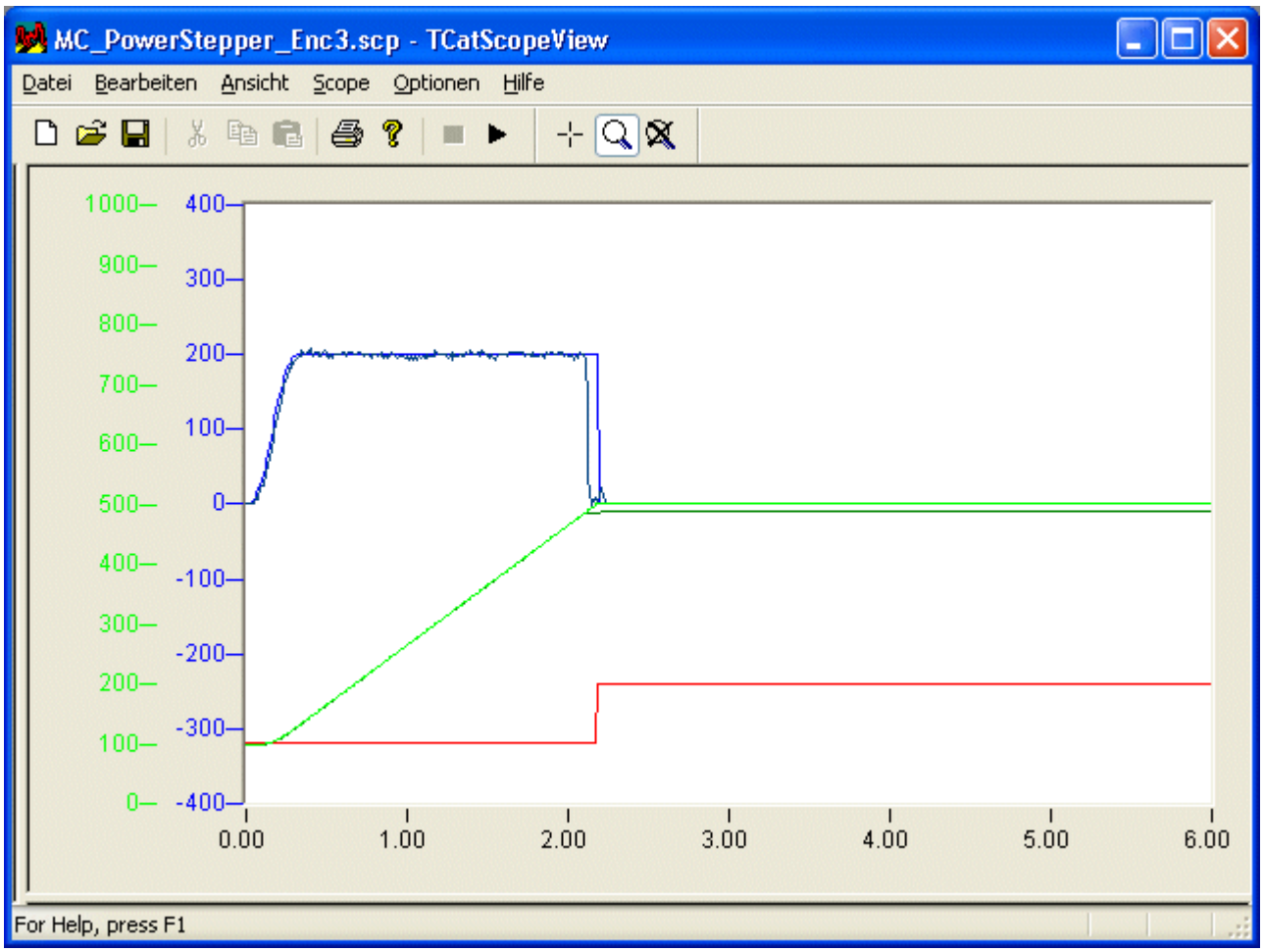
Wenn in der oben beschriebenen Situation die Ausgabe des Lagereglers auf einen ausreichend kleinen Wert wie z.B. 2% begrenzt wird ergibt sich das nachfolgende Bild.



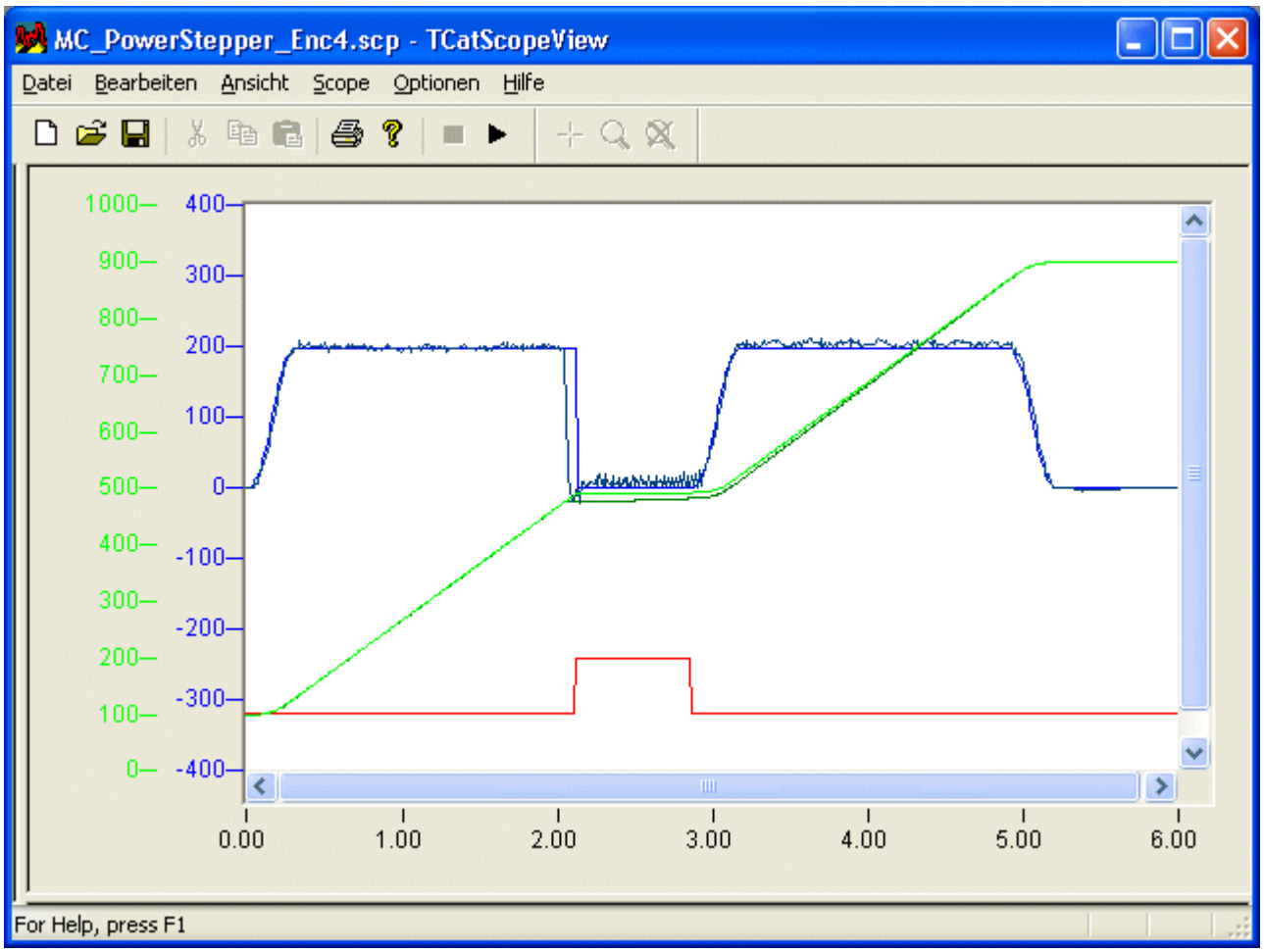
Auch hier ist für die restliche Zeit der Profilgenerierung die Sollgeschwindigkeit zu hoch, als dass der Schrittmotor der Sollbewegung angemessen folgen könnte. Nach dem das Ende des Sollprofils erreicht ist wird der Schrittmotor jetzt durch den Lageregler mit einer geringen Arbeitsfrequenz zum Ziel geführt, der er ohne Rampe folgen kann. Dabei erzeugt er ein recht hohes Moment. Allerdings ist der Zeitbedarf für diese Korrekturmaßnahme sehr hoch.

Erkennung und Handhabung von Stall-Situationen unter Verwendung eines Encoders

Um geeignete Gegenmaßnahmen ergreifen zu können ist zunächst ein Erkennen des Problems nötig. Das nachfolgende Bild ergibt sich, wenn ein MC_PowerStepper Baustein verwendet wird. In seiner Parameterstruktur des Typen ST_PowerStepperStruct [▶ 75] ist als DestallDetectMode PwStDetectMode_Lagging eingetragen. Der Baustein verwendet als Entscheidungsgrundlage den Schleppabstand der Achse und verwendet dabei den Grenzwert und die Filterzeit der hier zu deaktivierenden Schleppüberwachung aus den NC-Achsdaten. In diesem Beispiel ist PwStMode_SetError als DestallMode eingetragen. Das Ergebnis ist zunächst nur durch den anderen Fehlercode vom Schleppabstands-Alarm zu unterscheiden.



Wenn PwStMode_UseOverride als DestallMode eingetragen ist verwendet der MC_PowerStepper Baustein den Override, um das Profil unmittelbar zu stoppen. Da dieses Anhalten jedoch keinen Abbruch des Profils bewirkt, gleichzeitig aber das Erreichen des Profilendes verhindert werden keine Statusbits beeinflusst. Die hier auf 2% begrenzte Reglerausgabe führt die Achse bis auf die Schleppabstandsgrenze an die aktuelle Sollposition des Profils heran. Dann wird der Override wieder auf den von der Applikation vorgegebenen Wert gesetzt.



Im Ergebnis wird ein erheblich größerer Teil des Gesamt-Profils mit der vorgegebenen Geschwindigkeit gefahren und die Zielposition korrekt erreicht. Die Statusinformationen der Achse werden korrekt erzeugt.

Kombinationen von Stall-Erkennung und -Handhabung

Die nachfolgende Tabelle stellt Kombinationen der unterstützten Modi zur Stall-Erkennung und -Handhabung dar.

	PwStMode_SetError	PwStMode_SetErrNon-Ref	PwStMode_UseOverride
PwStDetectMode_Encoderless	Anmerkung 1	Anmerkung 2	nicht geeignet
PwStDetectMode_Lagging	Anmerkung 3	nicht sinnvoll	Anmerkung 4

Anmerkung 1: Sinnvoll bei Achsen ohne Encoder, die nicht referenziert werden.

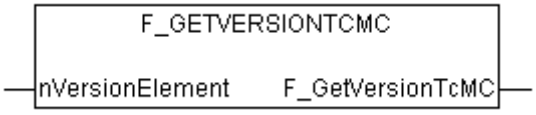
Anmerkung 2: Sinnvoll bei Achsen ohne Encoder, die unter Verwendung des Pulszählers der Klemme und z. B. eines externen Sensors referenziert werden.

Anmerkung 3: Das erzeugte Verhalten entspricht weitgehend der Schleppüberwachung.

Anmerkung 4: Sinnvoll bei Achsen mit Encoder.

5 Funktionen

5.1 F_GetVersionTcMC



Mit dieser Funktion können Versionsinformationen der SPS-Bibliothek ausgelesen werden.

FUNCTION F_GetVersionTcMC : UINT

```
VAR_INPUT
    nVersionElement : INT;
END_VAR
```

nVersionElement : Versionselement, das gelesen werden soll. Mögliche Parameter:

- 1 : major number;
- 2 : minor number;
- 3 : revision number;

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcMC.Lib

6 Datentypen

6.1 MC_AxisPara

```
TYPE MC_AxisPara : (
(* PLCopen specific parameters *)
   CommandedPosition := 1,    (* lreal *)
   SWLimitPos,                (* lreal *)
   SWLimitNeg,                (* lreal *)
   EnableLimitPos,            (* bool *)
   EnableLimitNeg,            (* bool *)
   EnablePosLagMonitoring,    (* bool *)
   MaxPositionLag,            (* lreal *)
   MaxVelocitySystem,         (* lreal *)
   MaxVelocityAppl,           (* lreal *)
   ActualVelocity,            (* lreal *)
   CommandedVelocity,         (* lreal *)
   MaxAccelerationSystem,     (* lreal *)
   MaxAccelerationAppl,       (* lreal*)
   MaxDecelerationSystem,     (* lreal *)
   MaxDecelerationAppl,       (* lreal*)
   MaxJerk,                   (* lreal *)

(* Beckhoff specific parameters *)
   AxisId :=1000,                    (* lreal *)
   AxisVeloManSlow,                  (* lreal *)
   AxisVeloManFast,                  (* lreal *)
   AxisVeloMax,                      (* lreal *)
   AxisAcc,                          (* lreal *)
   AxisDec,                          (* lreal *)
   AxisJerk,                         (* lreal *)
   AxisMaxVelocity,                  (* lreal *)
   AxisRapidTraverseVelocity,        (* lreal *)
   AxisManualVelocityFast,           (* lreal *)
   AxisManualVelocitySlow,           (* lreal *)
   AxisCalibrationVelocityForward,   (* lreal *)
   AxisCalibrationVelocityBackward,  (* lreal *)
   AxisJogIncrementForward,          (* lreal *)
   AxisJogIncrementBackward,         (* lreal *)
   AxisEnMinSoftPosLimit,            (* bool *)
   AxisMinSoftPosLimit,              (* lreal *)
   AxisEnMaxSoftPosLimit,            (* bool *)
   AxisMaxSoftPosLimit,              (* lreal *)
   AxisEnPositionLagMonitoring,      (* bool *)
   AxisMaxPosLagValue,               (* lreal *)
   AxisMaxPosLagFilterTime,          (* lreal *)
   AxisEnPositionRangeMonitoring,    (* bool *)
   AxisPositionRangeWindow,          (* lreal *)
   AxisEnTargetPositionMonitoring,   (* bool *)
   AxisTargetPositionWindow,         (* lreal *)
   AxisTargetPositionMonitoringTime, (* lreal*)
   AxisEnInTargetTimeout,            (* bool *)
   AxisInTargetTimeout,              (* lreal *)
   AxisEnMotionMonitoring,           (* bool *)
   AxisMotionMonitoringWindow,       (* lreal *)
   AxisMotionMonitoringTime,         (* lreal *)
   AxisDelayTimeVeloPosition,        (* lreal *)
   AxisEnLoopingDistance,            (* bool *)
   AxisLoopingDistance,              (* lreal *)
   AxisEnBacklashCompensation,       (* bool *)
   AxisBacklash,                     (* lreal *)
   AxisEnDataPersistence,            (* bool *)
   AxisRefVeloOnRefOutput,           (* lreal *)
   AxisOverrideType,                 (* lreal *)
   AxisEncoderScalingFactor,         (* lreal *)
   AxisEncoderOffset,                (* lreal *)
   AxisEncoderDirectionInverse,      (* bool *)
   AxisEncoderEncoderMask,           (* dword *)
   AxisEncoderModuloValue,           (* lreal *)
   AxisModuloToleranceWindow,        (* lreal *)
   AxisEnActPosCorrection,           (* bool *)
   AxisActPosCorrectionFilterTime,   (* lreal *)
   AxisUnitInterpretation,           (* lreal - Bit 2 of this word is the modulo flag *)

(* Beckhoff specific axis status information - READ ONLY *)
   AxisTargetPosition := 2000,       (* lreal *)
```

```
   AxisRemainingTimeToGo,            (* lreal *)
   AxisRemainingDistanceToGo         (* lreal *)
);
END_TYPE
```

In diesem Enum werden die Parameter für die Bausteine MC_ReadParameter, MC_ReadBoolParameter, MC_WriteParameter und MC_WriteBoolParameter vorgegeben. Stimmen die Datentypen nicht überein oder wird ein nicht vorhandener Parameter versucht zu lesen so wird ein Fehler generiert.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcMC.Lib

6.2 MC_Direction

```
TYPE MC_Direction :
(
    MC_Positive_Direction := 1,
    MC_Shortest_Way ,
    MC_Negative_Direction,
    MC_Current_Direction
);
END_TYPE
```

In diesem Enum werden die Richtungen für den Baustein MC_MoveVelocity und MC_MoveModulo vorgegeben.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcMC.Lib

6.3 MC_TouchProbe Datenstrukturen

MC_InputRef

MC_InputRef

```
TYPE MC_InputRef :
STRUCT
    EncoderID        : UDINT;        (* 1..255 *)
    TouchProbe       : E_TouchProbe;    (* Signal source *)
    Edge             : E_SignalEdge;    (* rising or falling signal edge *)
    PlcEvent         : BOOL;         (*
PLC trigger signal input when TouchProbe signal source is set to 'PlcEvent' *)
    ProbeState       : E_TouchProbeState; (* internal state of the touch probe sequence *)
    ModuloPositions  : BOOL;  (*
interpretation of FirstPosition, LastPosition and RecordedPosition as modulo positions when TRUE *)
END_STRUCT
END_TYPE
```

EncoderID : Die ID des Encoders kann im TwinCAT SystemManager abgelesen werden.

TouchProbe : Definiert die Signalquelle [▶ 73] innerhalb der verwendeten Encoder-Hardware, die die Messtasterfunktion triggert.

Edge : Legt fest, ob die steigende oder fallende Flanke [▶ 73] des Trigger-Signals ausgewertet wird.

PlcEvent : Wenn die Signalquelle TouchProbe [▶ 72] auf den Typ PlcEvent [▶ 72] eingestellt ist, so führt eine steigende Flanke an dieser Variablen zum Aufzeichnen der aktuellen Achsposition. Das PlcEvent ist keine echte Latch-Funktion, sondern Zykluszeit-abhängig.

ProbeState : Interner Zustand [▶ 73] der Messtasterfunktion. Diese Variable darf nicht beschrieben werden.

ModuloPositions : Wenn die Variable *ModuloPositions* FALSE ist, so wird die Achsposition in einem absoluten linearen Bereich von -∞ bis +∞ interpretiert. Die Positionen *FirstPosition*, *LastPosition* und *RecordedPosition* des Funktionsbausteins MC_TouchProbe [▶ 60] sind dann ebenfalls absolut.
Wenn ModuloPositions TRUE ist, werden alle Positionen modulo im Modulo-Bereich der verwendeten Achse interpretiert (z. B. 0..359.9999). Gleichzeitig bedeutet das, dass ein definiertes Trigger-Fenster sich zyklisch wiederholt.

E_TouchProbe

E_TouchProbe

Der Datentyp *E_TouchProbe* beschreibt, welches Signal von einer Signalquelle für die Messtasterfunktion verwendet wird.

```
TYPE E_TouchProbe :
(
    TouchProbe1     := 1,
    TouchProbe2,
    TouchProbe3,
    TouchProbe4,
    PlcEvent         := 10
);
END_TYPE
```

TouchProbe1..TouchProbe4 : Signalquelle innerhalb der Encoder-Hardware (z. B. Antrieb AX2000, KL5001). Hardwareabhängig können bis zu vier Trigger-Signale ausgewertet werden, es wird aber zur Zeit nur das erste Signal, also *TouchProbe1* unterstützt.

PlcEvent : *PlcEvent* definiert ein Trigger-Signal, das nicht mit einer Encoder-Hardware verschaltet ist, sonder direkt in der SPS erzeugt wird, also eine normale BOOL Variable. Dieses SPS-Trigger-Signal wird in die Datenstruktur MC_InputRef [▶ 72] in die Variable PlcEvent [▶ 72] eingespeist. Es führt dazu, dass die derzeit aktuelle Achsposition aufgezeichnet wird. Das PlcEvent ist keine echte Latch-Funktion, sondern Zykluszeit-abhängig.

E_SignalEdge

E_SignalEdge

```
TYPE E_SignalEdge :
(
    RisingEdge,
    FallingEdge
);
END_TYPE
```

Steigende oder fallende Flanke des Trigger-Signals.

E_TouchProbeState

E_TouchProbeState

```
TYPE E_TouchProbeState :
(
    TouchProbeInactive,
    TouchProbeActivated,
    TouchProbeAborted
);
END_TYPE
```

Interner Zustand der Messtasterfunktion.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
ab TwinCAT v2.9 Build 1000	PC (i386)	TcMC.Lib

6.4 E_ReadMode

```
TYPE E_ReadMode :
 (
       ReadMode_Once     := 1,
       ReadMode_Cyclic
 );

END_TYPE
```

In diesem Enum wird der Lesemodus für die Bausteine MC_ReadParameter und MC_ReadBoolParameter vorgegeben.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.8	PC (i386)	TcMC.Lib

6.5 E_SuperpositionMode

```
TYPE E_SuperpositionMode :
(
   SUPERPOSITIONMODE_VELOREDUCTION_ADDITIVEMOTION:= 1,
   SUPERPOSITIONMODE_VELOREDUCTION_LIMITEDMOTION,
   SUPERPOSITIONMODE_LENGTHREDUCTION_ADDITIVEMOTION,
   SUPERPOSITIONMODE_LENGTHREDUCTION_LIMITEDMOTION
);
END_TYPE
```

Die Struktur E_SuperpositionMode legt fest, wie eine überlagerte Bewegung mit dem Funktionsbaustein MC_MoveSuperImposedExt [▶ 23] durchgeführt wird.

SUPERPOSITIONMODE_VELOREDUCTION_ADDITIVEMOTION:

Die überlagerte Bewegung wird über die gesamte Strecke *Length* durchgeführt. Um die Distanz *Distance* auf dieser Strecke zu erreichen. wird die vorgegebene maximale Geschwindigkeitänderung *VelocityDiff* reduziert. Die Länge *Length* bezieht sich auf eine Vergleichs-Achse ohne überlagerte Bewegung (beispielsweise Master-Achse). Die Achse auf die die Ausgleichsfahrt wirkt legt die Strecke *Length+Distance* zurück.

SUPERPOSITIONMODE_VELOREDUCTION_LIMITEDMOTION:

Die überlagerte Bewegung wird über die gesamte Strecke *Length* durchgeführt. Um die Distanz *Distance* auf dieser Strecke zu erreichen. wird die vorgegebene maximale Geschwindigkeitänderung *VelocityDiff* reduziert. Die Länge *Length* bezieht sich auf die Achse, auf die die Ausgleichsfahrt wirkt. Diese legt während der Ausgleichsfahrt die Strecke *Length* zurück.

SUPERPOSITIONMODE_LENGTHREDUCTION_ADDITIVEMOTION:

Die überlagerte Bewegung wird auf möglichst kurzer Strecke mit möglichst hoher Geschwindigkeit ausgeführt. Dabei wird weder die maximale Geschwindigkeitänderung *VelocityDiff* noch die maximale Strecke *Length* überschritten. Die Länge *Length* bezieht sich auf eine Vergleichs-Achse ohne überlagerte Bewegung (beispielsweise Master-Achse). Die Achse auf die die Ausgleichsfahrt wirkt legt maximal die Strecke *Length+Distance* zurück.

SUPERPOSITIONMODE_LENGTHREDUCTION_LIMITEDMOTION:

Die überlagerte Bewegung wird auf möglichst kurzer Strecke mit möglichst hoher Geschwindigkeit ausgeführt. Dabei wird weder die maximale Geschwindigkeitänderung *VelocityDiff* noch die maximale Strecke *Length* überschritten. Die Länge *Length* bezieht sich auf die Achse, auf die die Ausgleichsfahrt wirkt. Diese legt während der Ausgleichsfahrt maximal die Strecke *Length* zurück.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9	PC (i386)	TcMC.Lib

6.6 ST_PowerStepperStruct

```
TYPE ST_PowerStepperStruct :
STRUCT
    DestallDetectMode : E_DestallDetectMode;
    DestallMode       : E_DestallMode;
    DestallEnable     : BOOL;
    StatusMonEnable   : BOOL;
END_STRUCT
END_TYPE
```

In dieser Struktur werden die Parameter für den Baustein MC_PowerStepper [▶ 64] vorgegeben.

DestallDetectMode : Hier wird mit einem kodierten Wert [▶ 76] festgelegt, auf welche Weise eine Stall-Situation erkannt werden soll.
DestallMode : Hier wird mit einem kodierten Wert [▶ 75] festgelegt, auf welche Weise eine erkannte Stall-Situation behandelt werden soll.
DestallEnable : Dieser Parameter legt fest, ob die Stall-Erkennung und -Handhabung entsprechend den oben genannten Regeln aktiv sein soll.
StatusMonEnable : Dieser Parameter legt fest, ob die Statusbits der E/A-Klemme überwacht und auftretende Fehler gemeldet werden sollen.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9 ab Build 1026	PC (i386)	TcMC.Lib

6.7 E_DestallMode

```
TYPE E_DestallMode :
(
    PwStMode_None := 0,
    PwStMode_SetError,
    PwStMode_SetErrNonRef,
    PwStMode_UseOverride
);
END_TYPE
```

In diesem Enum werden die Arbeitsweisen für den Baustein MC_PowerStepper [▶ 64] vorgegeben.

PwStMode_None : Eine aufgetretene Stall-Situation wird nicht behandelt.
PwStMode_SetError : Bei Auftreten einer Stall-Situation wird die Achse in einen Fehlerzustand mit Code 16#4636 versetzt. Eine Klartextmeldung wie z.B. "**MC_PowerStepper(AxId:=3) reporting error 0x4636 (stall error) !**" wird im Logbuch eingetragen und erscheint bei gestartetem SystemManager im Logger View.
PwStMode_SetErrNonRef : Bei Auftreten einer Stall-Situation wird die Achse wie in der Einstellung **PwStMode_SetError** in einen Fehlerzustand versetzt und eine Klartextmeldung ausgelöst. Zusätzlich wird die Achse in den nicht referenzierten Zustand versetzt.
PwStMode_UseOverride : Bei Auftreten einer Stall-Situation wird der Profilgenerator der Achse durch Nullsetzen des Overrides schnellstmöglich gestoppt. Nachdem der Lageregler die Achse wieder an die aktuelle Sollposition herangeführt hat wird der Profilgenerator durch Hochsetzen des Overrides veranlasst, den verbliebenen Restweg auszufahren.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9 ab Build 1026	PC (i386)	TcMC.Lib

6.8 E_DestallDetectMode

```
TYPE E_DestallDetectMode :
(
    PwStDetectMode_None := 0,
    PwStDetectMode_Encoderless,
    PwStDetectMode_Lagging
);
END_TYPE
```

In diesem Enum werden die Methoden der Stall-Erkennung für den Baustein MC_PowerStepper [▶ 64] vorgegeben.

PwStDetectMode_None : Es wird keine Stall-Erkennung durchgeführt.
PwStDetectMode_Encoderless : Zur Stall-Erkennung werden die Bits 1 bis 3 des Statusbyte einer Klemme des Typs KL2531 oder KL2541 ausgewertet. Das Byte ist als **KL_Status** an den MC_PowerStepper Baustein zu übergeben.
PwStDetectMode_Lagging : Zur Stall-Erkennung wird der Schleppfehler der Achse ausgewertet. Für die Entscheidung werden dabei die Schleppfehlergrenze und die Schleppfehlerfilterzeit aus den NC-Parametern benutzt.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS Bibliotheken
TwinCAT v2.9 ab Build 1026	PC (i386)	TcMC.Lib

7 Appendix

7.1 MC_Move.....Done

Die erfolgreiche Ausführung eines Fahrbefehls für die Achse wird durch die Meldung „Done“ (Beendet) des entsprechenden Funktionsbausteins bestätigt. Es werden drei Bedingungen geprüft, bevor die erfolgreiche Ausführung eines Fahrbefehls angezeigt wird. Das Flag AxisHasJob (Achse hat Befehl erhalten) wird kontrolliert, bevor MC_Move...Done auf TRUE (OK) gestellt wird. Wird die Option *Position Range Monitoring* (Positionsbereichsüberwachung) UND/ODER die Option *Target Position Monitoring* (Zielpositionsüberwachung) in der Global Section des entsprechenden Achsparameters auf TRUE gesetzt, wird der Status der Flags AxisInPositionWindow (Achse in Zielpunktfenster) und AxisIsAtTargetPosition (Achse auf Zielposition) überprüft. Er muss auf TRUE stehen, wenn AxisHasJob von TRUE auf FALSE gewechselt hat.

Für eine erfolgreiche Ausführung eines Fahrbefehls werden also folgende Kombinationen überprüft:

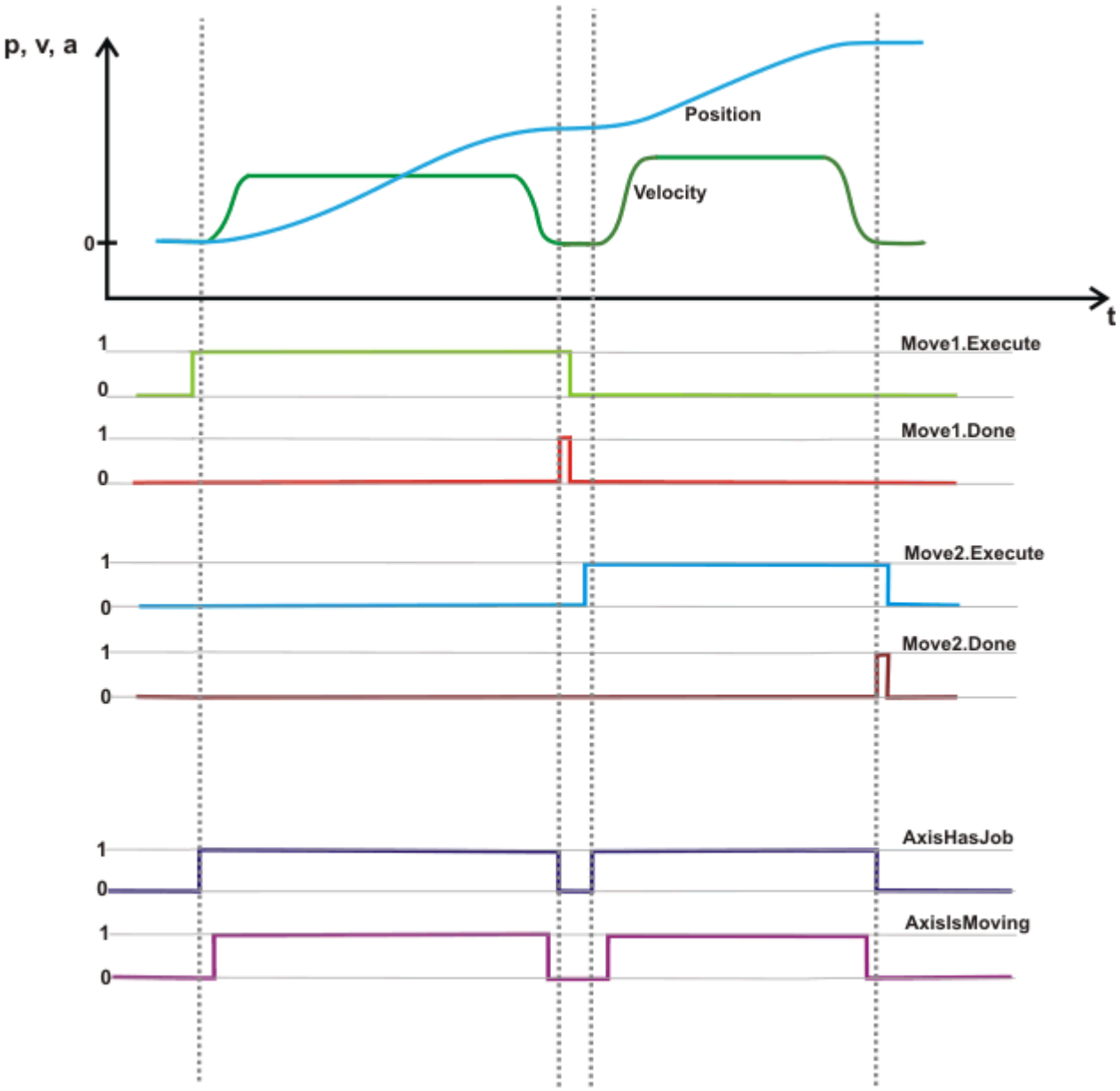
1. Wenn *Position Range Monitoring* = TRUE und *Target Position Monitoring* = TRUE, Überprüfung ob Status AxisHasJob = FALSE UND Status AxisIsAtTargetPosition = TRUE (implizite Überprüfung des TRUE-Status von AxisInPositionWindow).
2. Wenn *Position Range Monitoring* = TRUE und *Target Position Monitoring* = FALSE, Überprüfung ob AxisHasJob = FALSE UND AxisInPositionWindow = TRUE.
3. Wenn *Position Range Monitoring* = FALSE und *Target Position Monitoring* = FALSE, Überprüfung ob AxisHasJob = FALSE.

HINWEIS:

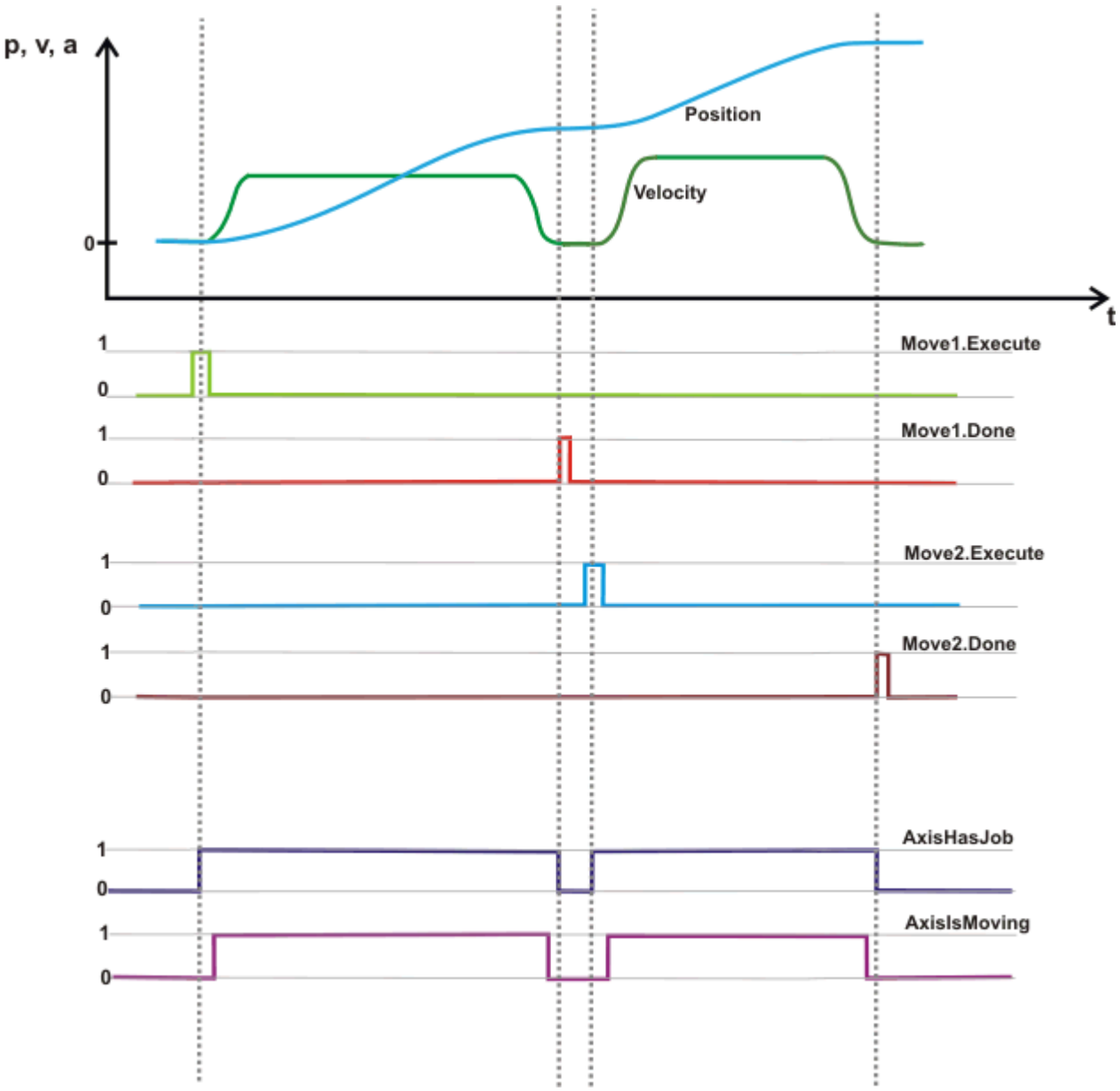
Position Range Monitoring und *Target Position Monitoring* sind Optionen der Global Section der entsprechenden Achsparameter im TwinCAT System Manager.

Mit den nachfolgenden Zahlen wird das Verhalten der Meldung MC_Move...Done auf Grundlage verschiedener MC_Move...Execute-Meldungen dargestellt. Die hier dargestellte Sequenz wird durch zwei aufeinanderfolgende Fahrbefehle gebildet.

In der folgenden Abbildung ist MC_Move...Execute während der Positionierung der Achse auf TRUE gesetzt und wird nur auf FALSE gesetzt, wenn die MC_Move...Done- Meldung TRUE lautet. MC_Move...Done bleibt TRUE, bis MC_Move...Execute auf TRUE gesetzt wird, wodurch angezeigt wird, dass der Befehl erfolgreich ausgeführt wurde.



In der folgenden Abbildung bleibt MC_Move...Execute nur während der ersten Zyklen nach dem Positionierungsbefehl für eine Achse TRUE. MC_Move...Done bleibt einen Zyklus TRUE. Dadurch wird angezeigt, dass der Befehl erfolgreich ausgeführt wurde.


p, v, a
Position
Velocity
0
t
1
0
Move1.Execute
1
0
Move1.Done
1
0
Move2.Execute
1
0
Move2.Done
1
0
AxisHasJob
1
0
AxisIsMoving

Mehr Informationen:
www.beckhoff.de/tx1200

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