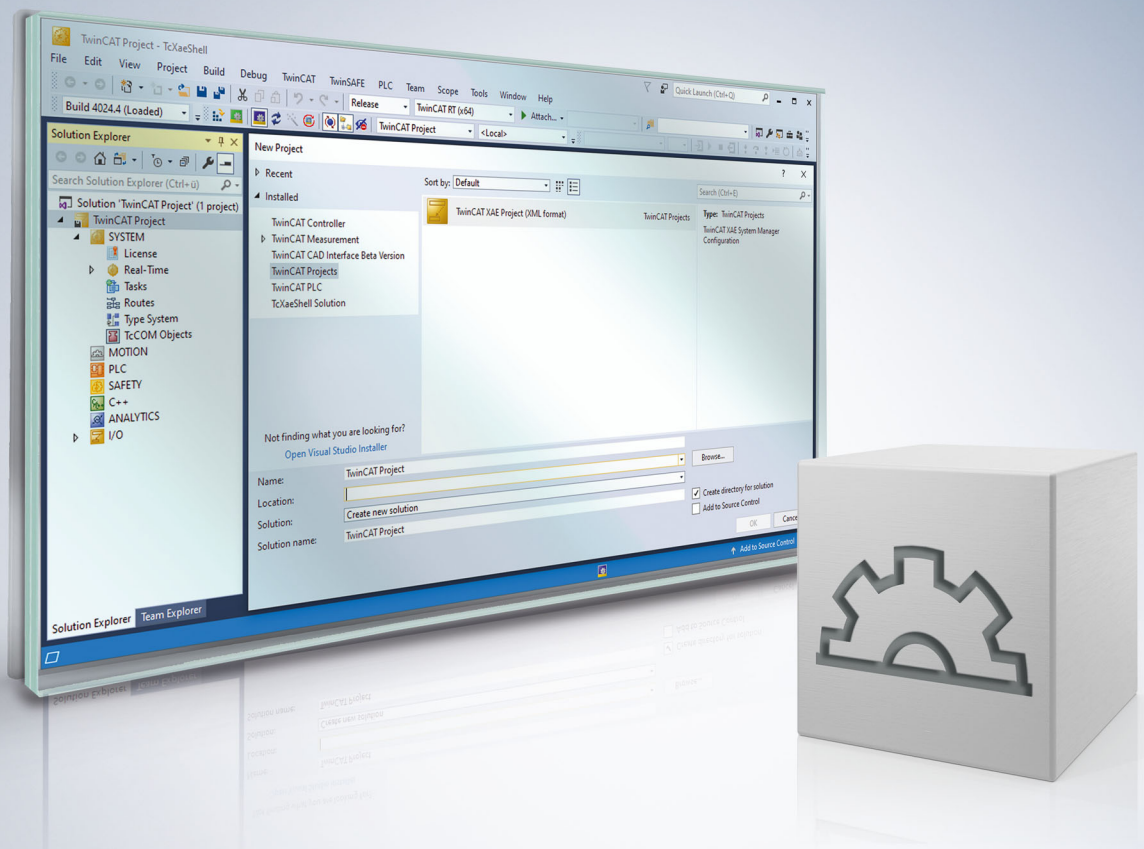
BECKHOFF New Automation Technology

Handbuch | DE

TE1000

TwinCAT 3 | PLC-Bibliothek: Tc3_DriveMotionControl



03.11.2023 | Version: 1.5.2

Inhaltsverzeichnis

1 Vorwort

1.1 Hinweise zur Dokumentation

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs- und Automatisierungstechnik, das mit den geltenden nationalen Normen vertraut ist.
Zur Installation und Inbetriebnahme der Komponenten ist die Beachtung der Dokumentation und der nachfolgenden Hinweise und Erklärungen unbedingt notwendig.
Das Fachpersonal ist verpflichtet, stets die aktuell gültige Dokumentation zu verwenden.

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

Disclaimer

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

Marken

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® und XPlanar® sind eingetragene und lizenzierte Marken der Beckhoff Automation GmbH.
Die Verwendung anderer in dieser Dokumentation enthaltenen Marken oder Kennzeichen durch Dritte kann zu einer Verletzung von Rechten der Inhaber der entsprechenden Bezeichnungen führen.

Patente

Die EtherCAT-Technologie ist patentrechtlich geschützt, insbesondere durch folgende Anmeldungen und Patente:
EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
mit den entsprechenden Anmeldungen und Eintragungen in verschiedenen anderen Ländern.

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

Copyright

1.2 Zu Ihrer Sicherheit

Sicherheitsbestimmungen

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

Haftungsausschluss

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

Qualifikation des Personals

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

Signalwörter

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

Warnungen vor Personenschäden

⚠ GEFAHR

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

⚠ WARNUNG

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

⚠ VORSICHT

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

Warnung vor Umwelt- oder Sachschäden

HINWEIS

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

Information zum Umgang mit dem Produkt

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

1.3 Hinweise zur Informationssicherheit

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

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

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

Um stets über Hinweise zur Informationssicherheit zu Produkten von Beckhoff informiert zu sein, abonnieren Sie den RSS Feed unter https://www.beckhoff.de/secinfo.

2 Übersicht

Die TwinCAT Motion Control SPS-Bibliothek Tc3_DriveMotionControl enthält Funktionsbausteine zur Programmierung von einfachen Maschinenapplikationen auf Basis der Beckhoff Servoklemmen-Technologie. Sie richtet sich nach der PLCopen-Spezifikation für Motion-Control-Funktionsbausteine V2.0 (www.PLCopen.org).

Diese Bibliothek stellt eine alternative Möglichkeit, ohne Verwendung der TwinCAT NC PTP, zur Steuerung von einfachen Bewegungen dar. Der Funktionsumfang ist gegenüber der TwinCAT NC PTP reduziert.

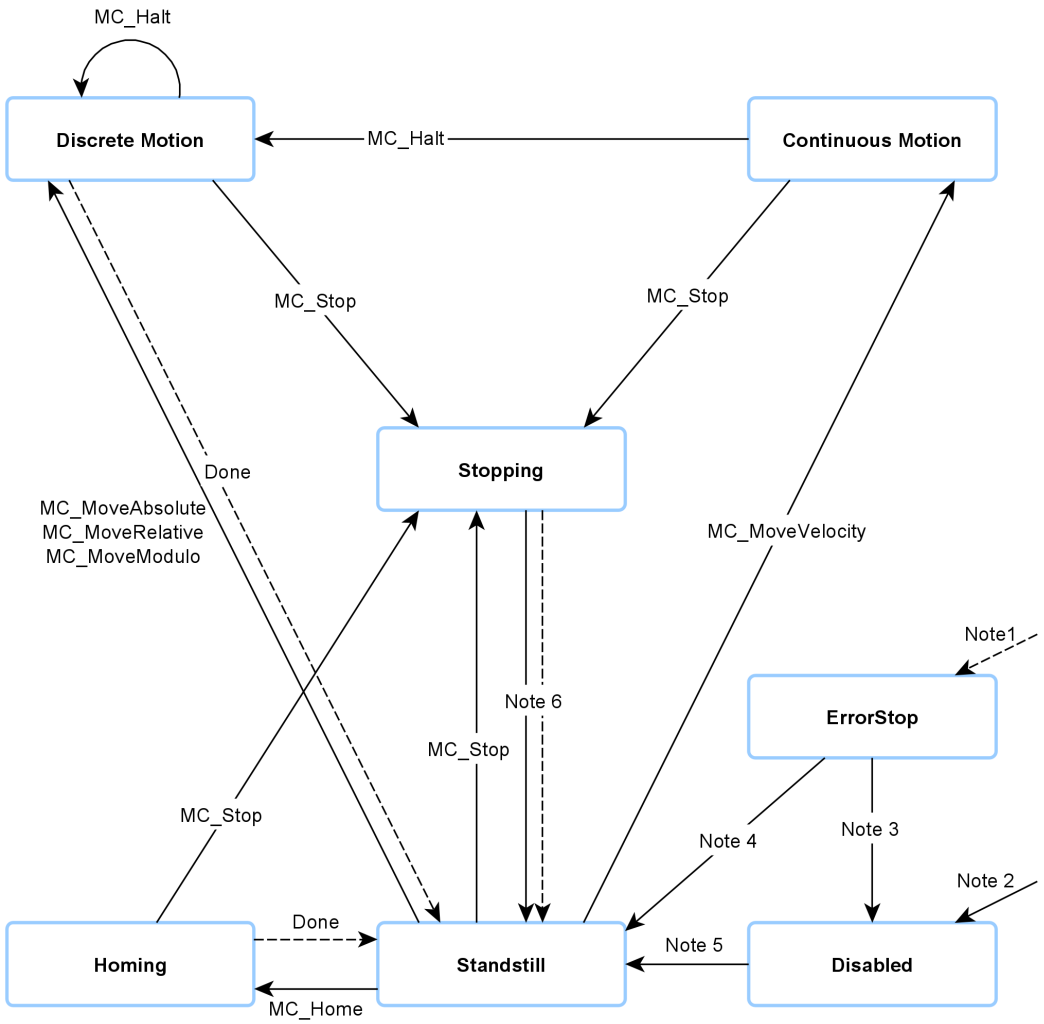
Soll die Bibliothek in Parallelität zur Tc2_MC2 verwendet werden, muss bei einer Bibliothek die Option „*Qualified access only*“ = TRUE gesetzt werden.

Diese Bibliothek kann dann über den entsprechenden Namespace angesprochen werden z.B.: Tc2_MC2.MC_Power.

In gewohnter, PLCopen konformer Art und Weise können Fahraufträge direkt an eine Servoklemme kommandiert werden.

3 Zustandsdiagramm

Das folgende Zustandsdiagramm definiert das Verhalten einer Achse.



Note 1	Aus undefiniertem Zustand, in dem ein Fehler auftritt.
Note 2	Aus undefiniertem Zustand, wenn MC_Power.Enable = FALSE. Die Achse hat keinen Fehler.
Note 3	MC_Reset und MC_Power.Status = FALSE
Note 4	MC_Reset und MC_Power.Status = TRUE und MC_Power.Enable = TRUE
Note 5	MC_Power.Status = TRUE und MC_Power.Enable = TRUE
Note 6	MC_Stop.Done = TRUE und MC_Stop.Execute = FALSE

Bewegungskommandos werden grundsätzlich sequentiell abgearbeitet. Alle Kommandos arbeiten in dem beschriebenen Zustandsdiagramm.

Die Achse befindet sich immer in einem der definierten Zustände. Jedes Bewegungskommando, das eine Transition verursacht, ändert den Zustand der Achse und damit das Bewegungsprofil. Das Zustandsdiagramm ist eine Abstraktionsebene, die den realen Zustand der Achse, vergleichbar zum Prozessabbild von E/A-Punkten, widerspiegelt. Der Zustand der Achse wechselt sofort mit dem auslösenden Kommando.

Das Zustandsdiagramm bezieht sich auf Einzelachsen.

Der Zustand „Disabled“ beschreibt den Grundzustand einer Achse. In diesem Zustand kann die Achse durch keinen Funktionsbaustein bewegt werden. Wenn der Funktionsbaustein MC_Power mit Enable = TRUE aufgerufen wird, wechselt die Achse in den Zustand „Standstill“ oder im Fehlerfall in den Zustand „ErrorStop“. Wenn der Funktionsbaustein MC_Power mit Enable = FALSE aufgerufen wird, wechselt der Zustand nach „Disabled“.

Zweck des Zustands „ErrorStop“ ist es, die Achse zu stoppen und anschließend keine weiteren Kommandos anzunehmen, bis ein Reset ausgelöst wurde. Der Zustandsübergang „Error“ bezieht sich nur auf tatsächliche Achsfehler und nicht auf Ausführungsfehler eines Funktionsbausteins. Achsfehler können aber auch am Fehlerausgang eines Funktionsbausteins angezeigt werden.

Funktionsbausteine, die im Zustandsdiagramm nicht aufgeführt werden, beeinflussen den Zustand der Achse.

Der Zustand „Stopping“ zeigt an, dass sich die Achse in einer Stopprampe befindet. Der Zustand wechselt nach dem vollständigen Stopp erst nach „Standstill“, wenn MC_Stop mit Execute = FALSE aufgerufen wird. Andernfalls bleibt die Achse für weitere Bewegungskommandos gesperrt.

4 Allgemeine Regeln für MC-Funktionsbausteine

Für alle MC-Funktionsbausteine gelten die nachfolgend beschriebenen Regeln. Diese stellen eine definierte Abarbeitung durch das SPS-Programm sicher.

Ausschließlichkeit der Ausgänge

Die Ausgänge „Busy“, „Done“, „Error“ und „CommandAborted“ schließen sich gegenseitig aus, d. h. an einem Funktionsbaustein kann zur gleichen Zeit nur einer dieser Ausgänge TRUE sein. Sobald der Eingang „Execute“ TRUE wird, muss einer der Ausgänge TRUE werden. Zur gleichen Zeit kann dabei jedoch nur einer der Ausgänge „Active“, „Done“, „Error“ und „CommandAborted“ TRUE sein.

Eine Ausnahme ist das Bewegungskommando MC Stop [▶ 37]. Dieses setzt den Ausgang „Done“ auf TRUE, sobald die Achse gestoppt ist. Dennoch bleiben auch die Ausgänge „Busy“ und „Active“ TRUE, da die Achse verriegelt wird. Erst nachdem der Eingang „Execute“ auf FALSE gesetzt wird, wird die Achse entriegelt und die Ausgänge „Busy“ sowie „Active“ werden auf FALSE gesetzt.

Ausgangszustand

Wenn der Funktionsbaustein nicht aktiv ist, werden die Ausgänge „Done“, „Error“, „ErrorID“ und „CommandAborted“ mit einer fallenden Flanke am Eingang „Execute“ zurückgesetzt. Die fallende Flanke am Eingang „Execute“ beeinflusst jedoch nicht die Kommandoausführung.

Wenn der Eingang „Execute“ bereits während der Kommandoausführung zurückgesetzt wird, ist sichergestellt, dass einer der Ausgänge am Ende des Kommandos für einen SPS-Zyklus gesetzt wird. Erst danach werden die Ausgänge zurückgesetzt.

Wenn der Eingang „Execute“ während der Ausführung eines Kommandos mehrfach getriggert wird, gibt der Funktionsbaustein keinerlei Rückmeldung, führt aber auch kein weiteres Kommando aus.

Eingangsparameter

Die Eingangsparameter werden mit steigender Flanke übernommen. Um die Parameter zu ändern, muss das Kommando neu getriggert werden, nachdem es beendet wurde. Wenn ein Eingangsparameter nicht an den Funktionsbaustein übergeben wird, bleibt der zuletzt an diesen Baustein übergebene Wert gültig. Beim ersten Aufruf sind sinnvoller Werte zu parametrieren.

Position und Distanz

Der Eingang „Position“ bezeichnet einen definierten Wert innerhalb eines Koordinatensystems. Der Eingang „Distance“ ist dagegen ein relatives Maß, also der Abstand zweier Positionen. Sowohl „Position“, als auch „Distance“ werden in technischen Einheiten, z. B. mm oder °, entsprechend der Skalierung der Achse angegeben.

Dynamikparameter

Die Dynamikparameter für Move-Funktionen werden in technischen Einheiten mit der Zeitbasis Sekunde angegeben. Ist eine Achse beispielsweise in Millimetern skaliert, haben die Parameter folgende Einheiten:

Velocity	mm/s
Acceleration	mm/s2
Deceleration	mm/s2

Fehlerbehandlung

Alle Funktionsbausteine haben zwei Fehlerausgänge, um Fehler während der Kommandoausführung anzuzeigen. Der Ausgang „Error“ zeigt den Fehler an und der Ausgang „ErrorID“ gibt eine ergänzende Fehlernummer aus. Die Ausgänge „Done“ und „InVelocity“ bezeichnen eine erfolgreiche Kommandoausführung und werden nicht gesetzt, wenn der Ausgang „Error“ TRUE ist.

Am Ausgang der Funktionsbausteine werden Fehler unterschiedlichen Typs signalisiert. Der Fehlertyp wird nicht explizit angegeben, sondern hängt von der Fehlernummer ab, die systemweit eindeutig vergeben ist.

Fehlertypen

- Funktionsbausteinfehler sind Fehler, die ausschließlich den Funktionsbaustein und nicht die Achse betreffen (z. B. fehlerhafte Parametrierung). Funktionsbausteinfehler müssen nicht explizit zurückgesetzt werden, sondern werden selbständig zurückgesetzt, wenn der Eingang „Execute“ zurückgesetzt wird.
- Kommunikationsfehler (z. B. kann der Funktionsbaustein die Achse nicht adressieren) deuten oft auf eine fehlerhafte Konfiguration oder Parametrierung hin. Ein Reset ist nicht möglich. Der Funktionsbaustein kann neu getriggert werden, nachdem die Konfiguration korrigiert wurde.
- Antriebsfehler (Regelgerät) können in vielen Fällen durch das Bewegungskommando MC_Reset [▶ 14] zurückgesetzt werden.

Verhalten des Done-Ausgangs

Der Ausgang „Done“ (oder auch alternativ „InVelocity“) wird gesetzt, wenn ein Kommando erfolgreich ausgeführt wurde.

Verhalten des CommandAborted-Ausgangs

Der Ausgang „CommandAborted“ wird gesetzt, wenn ein Kommando unterbrochen wird.

Verhalten des Busy-Ausgangs

Der Ausgang „Busy“ zeigt an, dass der Funktionsbaustein aktiv ist. Der Baustein kann nur dann mit einer steigenden Flanke am Eingang „Execute“ getriggert werden, wenn der Ausgang „Busy“ FALSE ist. „Busy“ wird sofort mit der steigenden Flanke am Eingang „Execute“ gesetzt und erst zurückgesetzt, wenn das Kommando erfolgreich oder auch nicht erfolgreich beendet wurde. Solange der Ausgang „Busy“ TRUE ist, muss der Funktionsbaustein zyklisch aufgerufen werden, um das Kommando ausführen zu können.

Verhalten des Active-Ausgangs

Der Ausgang „Active“ eines Bausteins zeigt an, dass der Baustein die Kontrolle über die Achse hat.

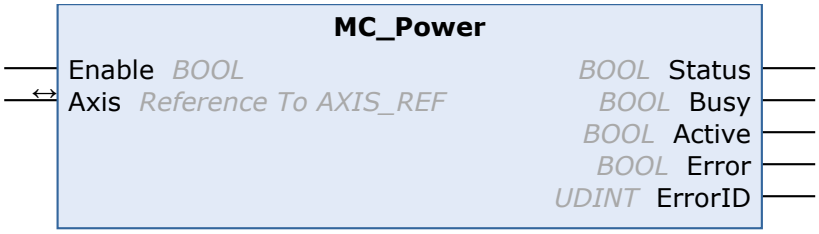
Options-Eingang

Viele Funktionsbausteine haben einen Eingang „Options“, der in einer Datenstruktur zusätzliche selten benötigte Optionen enthält. Um die Grundfunktion des Funktionsbausteins auszuführen, werden die Optionen oft nicht benötigt, sodass der Eingang unbelegt bleiben kann. Nur wenn in der Dokumentation ausdrücklich auf bestimmte Optionen hingewiesen wird, muss die Options-Datenstruktur vom Anwender belegt werden.

5 Organisationsbausteine

5.1 Achsfunktionen

5.1.1 MC_Power



Der Funktionsbaustein MC_Power schaltet die Software-Freigabe einer Achse. Am Ausgang Status wird die Betriebsbereitschaft der Achse signalisiert.

Zusätzlich zur Software-Freigabe kann es notwendig sein, ein Hardware-Freigabesignal zu schalten, um einen Antrieb freizugeben. Dieses Signal wird nicht durch MC_Power beeinflusst und muss durch die SPS separat geschaltet werden.

Eingänge

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

Name	Typ	Beschreibung
Enable	BOOL	Allgemeine Software-Freigabe für die Achse.

Ein-/Ausgänge

```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 40]	Achsdatenstruktur, welche eine Achse eindeutig im System adressiert. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.

Ausgänge

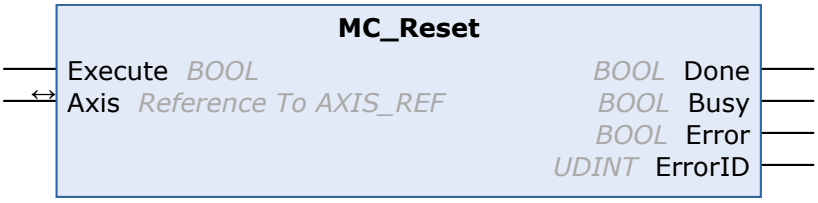
```
VAR_OUTPUT
    Status  : BOOL;
    Busy    : BOOL;
    Active  : BOOL;
    Error   : BOOL;
    ErrorID : UDINT;
END_VAR
```

Name	Typ	Beschreibung
Status	BOOL	TRUE, wenn die Achse betriebsbereit ist.
Busy	BOOL	TRUE, solange der Funktionsbaustein mit Enable = TRUE aufgerufen wird.
Active	BOOL	Zeigt an, dass das Kommando ausgeführt wird.
Error	BOOL	TRUE, wenn ein Fehler auftritt.
ErrorID	UDINT	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

Voraussetzungen

Entwicklungsumgebung	Einzubindende SPS-Bibliotheken
TwinCAT 3.1.4024.11	Tc3_DriveMotionControl

5.1.2 MC_Reset



Der Funktionsbaustein MC_Reset führt einen Reset der NC-Achse durch. In vielen Fällen führt dies auch zu einem Reset eines angeschlossenen Antriebsgerätes. Abhängig vom Bussystem oder Antriebstypen kann in einigen Fällen ein separater Reset des Antriebsgerätes notwendig sein.

Eingänge

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

Name	Typ	Beschreibung
Execute	BOOL	Mit einer steigenden Flanke wird das Kommando ausgeführt.

Ein-/Ausgänge

```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 40]	Achsdatenstruktur, welche eine Achse eindeutig im System adressiert. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.

Ausgänge

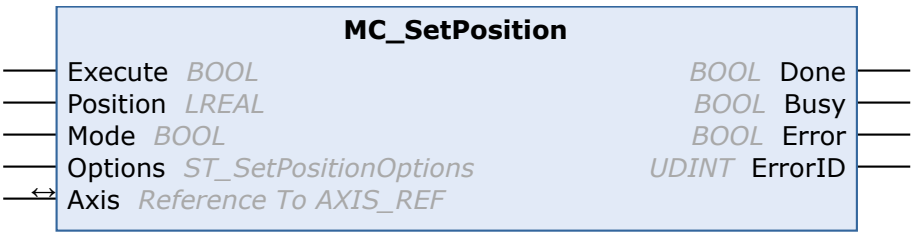
```
VAR_OUTPUT
    Done    : BOOL;
    Busy    : BOOL;
    Error   : BOOL;
    ErrorID : UDINT;
END_VAR
```

Name	Typ	Beschreibung
Done	BOOL	TRUE, wenn der Reset erfolgreich ausgeführt wurde.
Busy	BOOL	TRUE, solange der Funktionsbaustein mit Enable = TRUE aufgerufen wird.
Error	BOOL	TRUE, wenn ein Fehler auftritt.
ErrorID	UDINT	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

Voraussetzungen

Entwicklungsumgebung	Einzubindende SPS-Bibliotheken
TwinCAT 3.1.4024.11	Tc3_DriveMotionControl

5.1.3 MC_SetPosition



Der Funktionsbaustein MC_SetPosition setzt die aktuelle Achsposition auf einen parametrierbaren Wert.

Im absoluten Modus wird die Istposition auf den parametrierten absoluten Wert Position gesetzt. Im relativen Modus wird die Istposition um den parametrierten Wert Position verschoben. In beiden Fällen wird die Sollposition der Achse so gesetzt, dass ein eventuell vorhandener Schleppfehler erhalten bleibt. Wenn der Schleppfehler gelöscht werden soll, ist das über den Schalter Options.ClearPositionLag möglich.

Der relative Modus ist geeignet, die Achsposition während der Fahrt zu ändern.

Eingänge

```
VAR_INPUT
    Execute  : BOOL;
    Position : LREAL;
    Mode     : BOOL; (* RELATIVE=True, ABSOLUTE=False (Default) *)
    Options  : ST_SetPositionOptions;
END_VAR
```

Name	Typ	Beschreibung
Execute	BOOL	Mit einer steigenden Flanke wird das Kommando ausgeführt.
Position	LREAL	Positionswert, auf den die Achsposition gesetzt werden soll. Im absoluten Modus wird die Istposition auf diesen Wert gesetzt, im relativen Modus wird sie um diesen Wert verschoben.
Mode	BOOL	Wenn Mode = FALSE ist, wird die Achsposition auf einen absoluten Wert gesetzt. Anderenfalls wird die Achsposition relativ um den angegebenen Wert Position verändert. Der relative Modus ist geeignet, um die Position einer Achse während der Fahrt anzupassen.
Options	ST_SetPositionOptions	Momentan nicht verwendet

Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 11]

Ein-/Ausgänge

```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 40]	Achsdatenstruktur, welche eine Achse eindeutig im System adressiert. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.

Ausgänge

```
VAR_OUTPUT
    Done    : BOOL;
    Busy    : BOOL;
```

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

Name	Typ	Beschreibung
Done	BOOL	TRUE, wenn die Position erfolgreich gesetzt wurde.
Busy	BOOL	TRUE, sobald das Kommando mit „Execute“ gestartet wird und solange der Befehl abgearbeitet wird. Wenn „Busy“ FALSE ist, ist der Funktionsbaustein bereit für einen neuen Auftrag. Gleichzeitig ist einer der Ausgänge „Done“ oder „Error“ gesetzt.
Error	BOOL	TRUE, wenn ein Fehler auftritt.
ErrorID	UDINT	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 11]

Voraussetzungen

Entwicklungsumgebung	Einzubindende SPS-Bibliotheken
TwinCAT 3.1.4024.11	Tc3_DriveMotionControl

5.2 Touch Probe

5.2.1 MC_AbortTrigger

Der Funktionsbaustein MC_AbortTrigger bricht einen durch MC_TouchProbe gestarteten Messtasterzyklus ab. MC_TouchProbe startet einen Messtasterzyklus, indem ein Positionslatch in der Antriebs-Hardware aktiviert wird. Um den Vorgang zu beenden, bevor das Trigger-Signal das Positionslatch aktiviert hat, kann der Baustein MC_AbortTrigger verwendet werden. Wurde der Messtasterzyklus erfolgreich beendet, ist es nicht notwendig, diesen Baustein aufzurufen.

Eingänge

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

Name	Typ	Beschreibung
Execute	BOOL	Mit einer steigenden Flanke wird das Kommando ausgeführt und das externe Positionslatch wird deaktiviert.

Ein-/Ausgänge

```
VAR_IN_OUT
    Axis         : AXIS_REF;
    TriggerInput : TRIGGER_REF;
END_VAR
```

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 40]	Achsdatenstruktur
TriggerInput	TRIGGER_REF [▶ 45]	Datenstruktur zur Beschreibung der Trigger-Quelle. Diese Datenstruktur muss vor dem ersten Aufruf des Funktionsbausteins parametriert werden.

Ausgänge

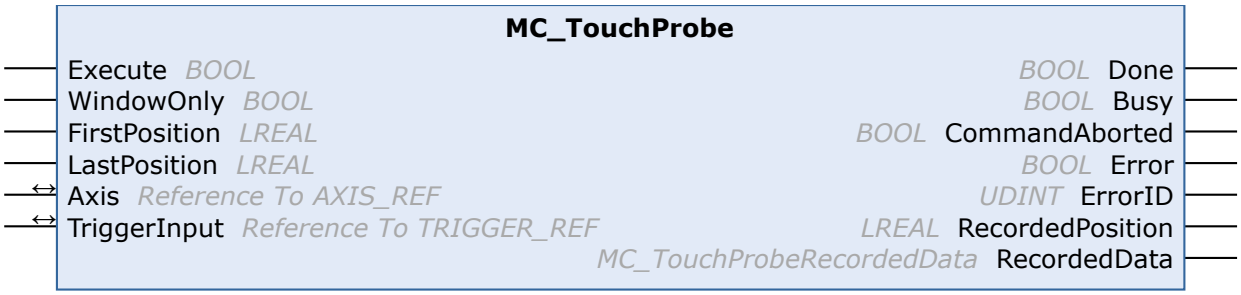
```
VAR_OUTPUT
    Done    : BOOL;
    Busy    : BOOL;
    Error   : BOOL;
    ErrorID : UDINT;
END_VAR
```

Name	Typ	Beschreibung
Done	BOOL	TRUE, sobald der Messtasterzyklus erfolgreich abgebrochen wurde.
Busy	BOOL	TRUE, sobald der Baustein aktiv ist. FALSE, wenn er sich im Grundzustand befindet.
Error	BOOL	TRUE, wenn ein Fehler auftritt.
ErrorID	UDINT	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

Voraussetzungen

Entwicklungsumgebung	Einzubindende SPS-Bibliotheken
TwinCAT 3.1.4024.11	Tc3_DriveMotionControl

5.2.2 MC_TouchProbe

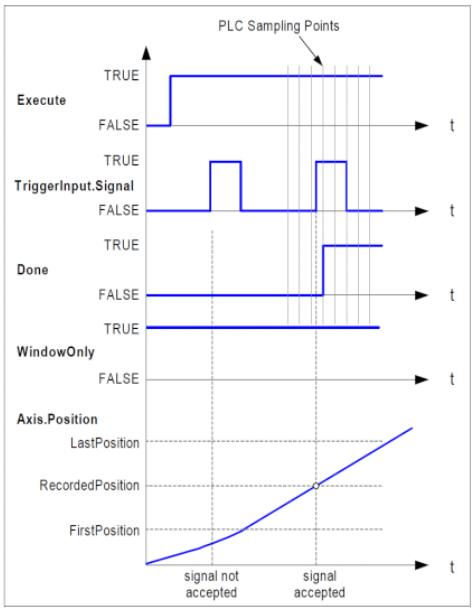


Der Funktionsbaustein MC_TouchProbe erfasst eine Achsposition zum Zeitpunkt eines digitalen Signals (Messtasterfunktion). Die Position wird über ein externes Hardware-Latch erfasst und ist damit hochgenau und von der Zykluszeit unabhängig. Der Baustein steuert diesen Mechanismus und ermittelt die extern erfasste Position.

Beteiligte Parameter müssen unter Umständen in den Parametern des Drive eingestellt werden. Für die Servoklemme sind die Parameter z. B. in den Objekten DMC Setting (0x8030) bzw. DMC Features (0x8031) zu finden.

Nachdem ein Messtasterzyklus durch eine steigende Flanke am Eingang „Execute“ gestartet wurde, wird dieser erst beendet, wenn die Ausgänge „Done“, „Error“ oder „CommandAborted“ TRUE werden. Soll der Vorgang zwischenzeitlich abgebrochen werden, muss der Funktionsbaustein MC_AbortTrigger [▶ 16] mit derselben TriggerInput [▶ 45]-Datenstruktur aufgerufen werden. Anderenfalls kann kein neuer Zyklus gestartet werden.

Signalverlauf

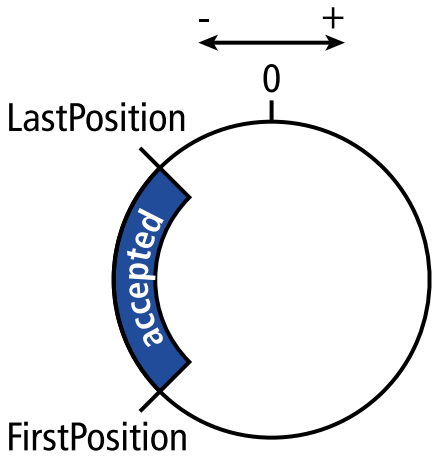


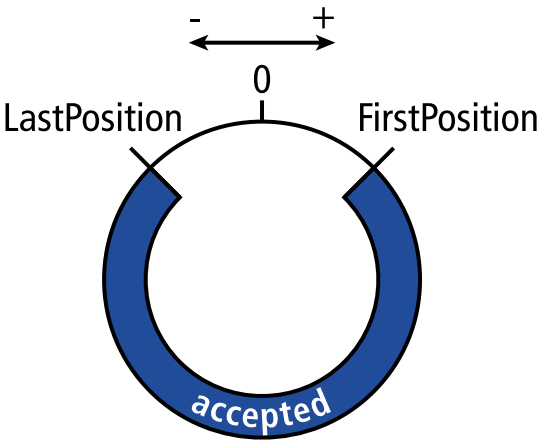
Eingänge

```
VAR_INPUT
    Execute       : BOOL;
    WindowOnly    : BOOL;
    FirstPosition : LREAL;
    LastPosition  : LREAL;
END_VAR
```

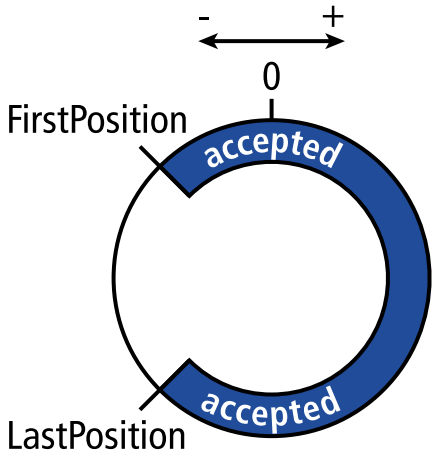
Name	Typ	Beschreibung
Execute	BOOL	Mit einer steigenden Flanke wird das Kommando ausgeführt und das externe Positionslatch aktiviert.
WindowOnly	BOOL	Wenn WindowOnly = TRUE ist, wird nur eine Position innerhalb des Fensters zwischen „FirstPosition“ und „LastPosition“ erfasst. Positionen außerhalb des 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 muss das Flag „ModuloPositions“ in der TriggerInput [▶ 45]-Datenstruktur entsprechend gesetzt werden. Bei absoluten Positionen gibt es exakt ein Fenster. Bei Modulo-Positionen wiederholt sich das Fenster innerhalb des in den Achsparametern festgelegten Modulo-Zyklus (z. B. 0 bis 360°).
FirstPosition	LREAL	Anfangsposition des Erfassungsfensters, wenn „WindowOnly“ TRUE ist. Diese Position kann absolut oder modulo interpretiert werden. Dazu muss in der TriggerInput [▶ 45]-Datenstruktur das Flag „ModuloPositions“ entsprechend gesetzt werden.
LastPosition	LREAL	Endposition des Erfassungsfensters, wenn „WindowOnly“ TRUE ist. Diese Position kann absolut oder modulo interpretiert werden. Dazu muss das Flag „ModuloPositions“ in der TriggerInput [▶ 45]-Datenstruktur entsprechend gesetzt werden.

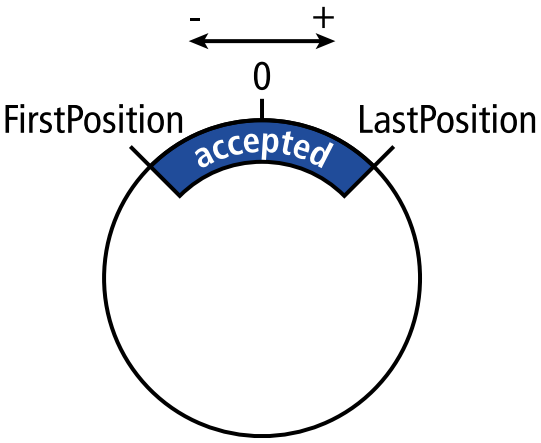
A. FirstPosition < LastPosition





B. FirstPosition > LastPosition





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

Ein-/Ausgänge

```
VAR_IN_OUT
    Axis         : AXIS_REF;
    TriggerInput : TRIGGER_REF;
END_VAR
```

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 40]	Achsdatenstruktur
TriggerInput	TRIGGER_REF [▶ 45]	Datenstruktur zur Beschreibung der Trigger-Quelle. Diese Datenstruktur muss vor dem ersten Aufruf des Funktionsbausteins parametriert werden.

Ausgänge

```
VAR_OUTPUT
    Done              : BOOL;
    Busy              : BOOL;
    CommandAborted    : BOOL;
    Error             : BOOL;
    ErrorId           : UDINT;
    RecordedPosition  : LREAL;
    RecordedData      : MC_TouchProbeRecordedData;
END_VAR
```

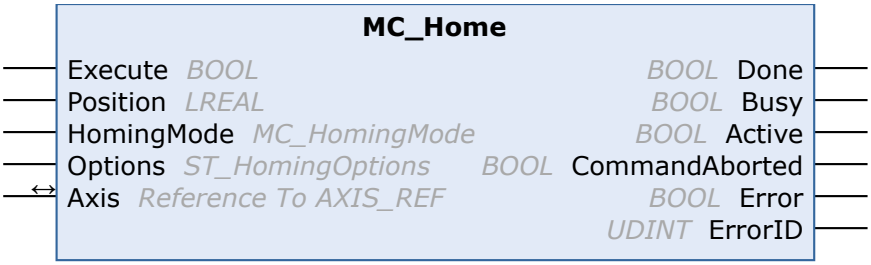
Name	Typ	Beschreibung
Done	BOOL	TRUE, wenn eine Achsposition erfolgreich erfasst wurde. Die Position wird am Ausgang „RecordedPosition“ ausgegeben.
Busy	BOOL	TRUE, sobald der Baustein aktiv ist. FALSE, wenn er sich im Grundzustand befindet.
CommandAborted	BOOL	TRUE, wenn der Vorgang von außen, z. B. durch den Aufruf von MC_AbortTrigger [▶ 16], abgebrochen wurde.
Error	BOOL	TRUE, wenn ein Fehler auftritt.
ErrorID	UDINT	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.
RecordedData	MC_TouchProbeRecordedData	Datenstruktur mit ergänzenden Informationen zur erfassten Achsposition zum Zeitpunkt des Trigger-Signals

Voraussetzungen

Entwicklungsumgebung	Einzubindende SPS-Bibliotheken
TwinCAT 3.1.4024.11	Tc3_DriveMotionControl

6 Motion-Bausteine

6.1 MC_Home (Homing)



Mit dem Funktionsbaustein MC_Home wird eine Referenzierfahrt der Achse durchgeführt.

Der Referenziermodus wird in den Options mit dem Parameter „ReferenceMode“ eingestellt.

> Beteiligte Parameter müssen unter Umständen in den Parametern des Drive eingestellt werden. Für die Servoklemme sind die Parameter z.B. in den Objekten DMC Setting (0x8030) bzw. DMC Features (0x8031) zu finden.

Eingänge

```
VAR_INPUT
    Execute         : BOOL;
    Position        : LREAL         := DEFAULT_HOME_POSITION;
    HomingMode      : MC_HomingMode;
    Options         : ST_HomingOptions;
END_VAR
```

Name	Typ	Beschreibung
Execute	BOOL	Mit einer steigenden Flanke wird das Kommando ausgeführt.
Position	LREAL	Absolute Referenzposition, auf die die Achse nach der Referenzfahrt gesetzt wird. Alternativ kann hier die Konstante DEFAULT_HOME_POSITION verwendet werden. Dadurch wird die im TwinCAT System Manager festgelegte „Referenzposition für Referenzierfahrt“ verwendet.
HomingMode	MC_HomingMode	Bestimmt, auf welche Weise die Kalibrierung durchgeführt wird. (Typ: MC_HomingMode [▶ 41]) • MC_DefaultHoming Führt die Standard-Referenzierfahrt aus. • MC_Direct Setzt die Position der Achse direkt auf Position ohne eine Bewegung auszuführen. • MC_Block Führt ein Referenzieren auf einen mechanischen Endanschlag durch. • MC_ForceCalibration Erzwingt den Zustand „Achse ist kalibriert“. Es wird keine Bewegung ausgeführt und die Position bleibt unverändert. • MC_ResetCalibration Setzt den Kalibrierungszustand der Achse zurück. Es wird keine Bewegung ausgeführt und die Position bleibt unverändert.
Options	ST_HomingOptions	Datenstruktur, die zusätzliche Parameter enthält.

Name	Typ	Beschreibung
		• SearchDirection: Richtung in der die Referenznocke gesucht werden soll • SearchVelocity: Geschwindigkeit mit der die Referenznocke gesucht werden soll • SyncDirection: Richtung in der die fallende Flanke der Referenznocke gesucht wird, nachdem die Referenznocke erkannt wurde • SyncVelocity: Geschwindigkeit mit der die fallende Flanke der Referenznocke gesucht wird, nachdem die Referenznocke erkannt wurde • ReferenceMode: Mode der Referenzierung (Momentan nur ENCODERREFERENCEMODE_CAMATDIGITALINPUT) • Acceleration: Beschleunigung für die Referenzierfahrt • Deceleration: Verzögerung für die Referenzierfahrt Das Signal einer Referenznocke, muss auf einen digitalen Eingang der Klemme geführt werden (HomingMode = MC_DefaultHoming).

Da die Referenzposition üblicherweise noch während der Fahrt gesetzt wird, bleibt die Achse nicht exakt an dieser Position stehen. Die Stillstandsposition weicht um den Bremsweg der Achse ab, dennoch ist die Kalibrierung exakt.

Ein-/Ausgänge

```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 40]	Achsdatenstruktur, welche eine Achse eindeutig im System adressiert. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.

Ausgänge

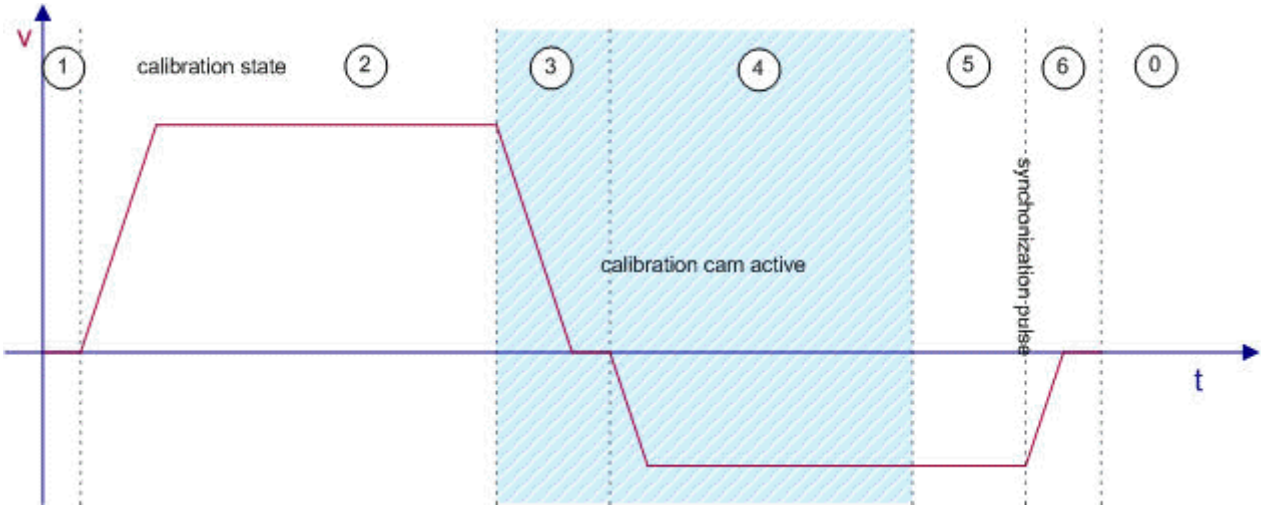
```
VAR_OUTPUT
    Done           : BOOL;
    Busy           : BOOL;
    Active         : BOOL;
    CommandAborted : BOOL;
    Error          : BOOL;
    ErrorID        : UDINT;
END_VAR
```

Name	Typ	Beschreibung
Done	BOOL	TRUE, wenn die Achse kalibriert wurde und die Bewegung beendet ist.
Busy	BOOL	TRUE, sobald das Kommando mit „Execute“ gestartet wird und solange der Fahrbefehl abgearbeitet wird. Wenn „Busy“ FALSE ist, ist der Funktionsbaustein bereit für einen neuen Auftrag. Gleichzeitig ist einer der Ausgänge „Done“, „CommandAborted“ oder „Error“ gesetzt.

Name	Typ	Beschreibung
Active	BOOL	Zurzeit nicht implementiert – „Active“ zeigt an, dass das Kommando ausgeführt wird. Wenn das Kommando gepuffert wurde, wird es evtl. erst aktiv, nachdem ein laufendes Kommando beendet worden ist.
CommandAborted	BOOL	TRUE, wenn das Kommando nicht vollständig ausgeführt werden konnte.
Error	BOOL	TRUE, wenn ein Fehler auftritt.
ErrorID	UDINT	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

Anmerkung

Der Referenziervorgang läuft in mehreren Phasen ab. Im nachfolgenden Bild ist der Ablauf nach dem Start des Funktionsbausteins MC_Home mit den einzelnen Phasen schematisch für den Fall HomingMode = MC_DefaultHoming dargestellt.



Voraussetzungen

Entwicklungsumgebung	Einzubindende SPS-Bibliotheken
TwinCAT 3.1.4024.11	Tc3_DriveMotionControl

6.2 MC_Jog (Manual Motion)

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.

Eingänge

```
VAR_INPUT
    JogForward   : BOOL;
    JogBackwards : BOOL;
    Velocity     : LREAL;
```

```
    Acceleration : LREAL;
    Deceleration : LREAL;
END_VAR
```

Name	Typ	Beschreibung
JogForward	BOOL	Mit einer steigenden Flanke wird das Kommando ausgeführt und die Achse in positiver Fahrtrichtung bewegt. Während der Bewegung werden keine weiteren Signalflanken angenommen, auch nicht am Eingang „JogBackwards“.
JogBackwards	BOOL	Mit einer steigenden Flanke wird das Kommando ausgeführt und die Achse in negativer Fahrtrichtung bewegt. „JogForward“ und „JogBackwards“ sollten alternativ getriggert werden, sind aber auch intern gegeneinander verriegelt.
Velocity	LREAL	Maximale Geschwindigkeit, mit der gefahren werden soll (>0).
Acceleration	LREAL	Beschleunigung (≥0).
Deceleration	LREAL	Verzögerung (≥0).

Ein-/Ausgänge

```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 40]	Achsdatenstruktur, welche eine Achse eindeutig im System adressiert. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.

Ausgänge

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

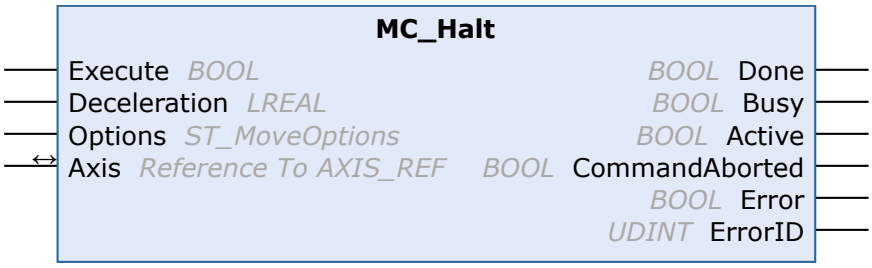
Name	Typ	Beschreibung
Done	BOOL	TRUE, wenn eine Bewegung erfolgreich abgeschlossen wurde.
Busy	BOOL	TRUE, sobald der Baustein aktiv ist. FALSE, wenn er sich im Grundzustand befindet. Erst dann kann eine weitere Flanke an den Jog-Eingängen angenommen werden.
Active	BOOL	Zeigt an, dass die Achse durch die Jog-Funktion bewegt wird.
CommandAborted	BOOL	TRUE, wenn der Vorgang von außen, z. B. durch den Aufruf von MC_Stop [▶ 37], abgebrochen wurde.
Error	BOOL	TRUE, wenn ein Fehler auftritt.
ErrorID	UDINT	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

Voraussetzungen

Entwicklungsumgebung	Einzubindende SPS-Bibliotheken
TwinCAT 3.1.4024.11	Tc3_DriveMotionControl

6.3 Point to Point Motion

6.3.1 MC_Halt



Der Funktionsbaustein MC_Halt hält eine Achse mit einer definierten Bremsrampe an.

Im Gegensatz zu MC_Stop [▶ 37] wird die Achse nicht gegen weitere Fahrbefehle verriegelt. Die Achse kann also nach dem Halt durch ein anderes Kommando gestartet werden.

Eingänge

```
VAR_INPUT
    Execute      : BOOL;
    Deceleration : LREAL;
    Options      : ST_MoveOptions;
END_VAR
```

Name	Typ	Beschreibung
Execute	BOOL	Mit einer steigenden Flanke wird das Kommando ausgeführt.
Deceleration	LREAL	Verzögerung Bei einem Wert ≤ 0 wirkt die mit dem letzten Move-Kommando parametrierte Verzögerung. MC_Halt und auch MC_Stop [▶ 37] können aus Sicherheitsgründen nicht mit schwächerer Dynamik ausgeführt werden, als der gerade aktive Fahrauftrag. Die Parametrierung wird gegebenenfalls automatisch angepasst.
Options	ST_MoveOptions	Datenstruktur (ST_MoveOptions [▶ 42]), die zusätzliche selten benötigte Parameter enthält. Im Normalfall kann der Eingang unbelegt bleiben.

Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 11].

Ein-/Ausgänge

```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 40]	Achsdatenstruktur, welche eine Achse eindeutig im System adressiert. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.

Ausgänge

```
VAR_OUTPUT
    Done         : BOOL;
    Busy         : BOOL;
    Active       : BOOL;
```

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

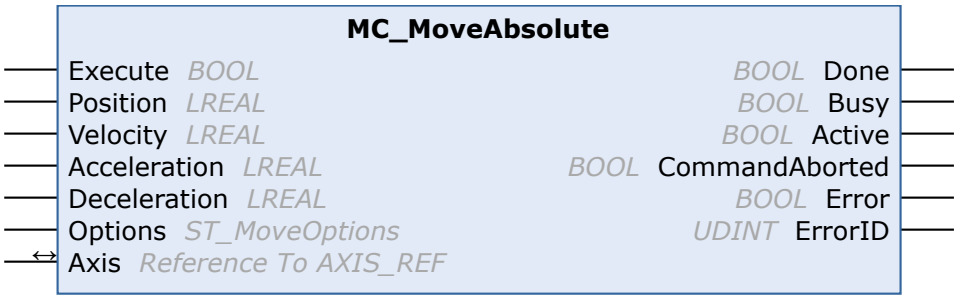
Name	Typ	Beschreibung
Done	BOOL	TRUE, wenn die Achse gestoppt wurde und im Stillstand ist.
Busy	BOOL	TRUE, sobald das Kommando mit „Execute“ gestartet wird und solange das Kommando abgearbeitet wird. Wenn „Busy“ FALSE ist, ist der Funktionsbaustein bereit für einen neuen Auftrag. Gleichzeitig ist einer der Ausgänge „Done“, „CommandAborted“ oder „Error“ gesetzt.
Active	BOOL	Zeigt an, dass das Kommando ausgeführt wird. Wenn das Kommando gepuffert wurde, wird es evtl. erst aktiv, nachdem ein laufendes Kommando beendet ist.
CommandAborted	BOOL	Wird TRUE, wenn das Kommando nicht vollständig ausgeführt werden konnte. Das laufende Kommando wurde eventuell durch ein Move-Kommando abgelöst.
Error	BOOL	TRUE, wenn ein Fehler auftritt.
ErrorID	UDINT	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 11].

Voraussetzungen

Entwicklungsumgebung	Einzubindende SPS-Bibliotheken
TwinCAT 3.1.4024.11	Tc3_DriveMotionControl

6.3.2 MC_MoveAbsolute



Der Funktionsbaustein MC_MoveAbsolute startet eine Positionierung auf eine absolute Zielposition und überwacht die Achsbewegung über den gesamten Fahrweg. Der Ausgang „Done“ wird gesetzt, wenn die Zielposition angefahren wurde. Anderenfalls wird der Ausgang „CommandAborted“ oder im Fehlerfall der Ausgang „Error“ gesetzt.

MC_MoveAbsolute wird in erster Linie für lineare Achssysteme eingesetzt. Bei Modulo-Achsen wird die Position nicht als Modulo-Position, sondern ebenfalls als absolute Position in einem endlosen absoluten Koordinatensystem interpretiert. Zur Modulo-Positionierung kann alternativ der Funktionsbaustein MC_MoveModulo [▶ 28] verwendet werden.

Eingänge

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

```
    Acceleration : LREAL;
    Deceleration : LREAL;
    Options      : ST_MoveOptions;
END_VAR
```

Name	Typ	Beschreibung
Execute	BOOL	Mit einer steigenden Flanke wird das Kommando ausgeführt.
Position	LREAL	Absolute Zielposition, auf die positioniert werden soll.
Velocity	LREAL	Maximale Geschwindigkeit, mit der gefahren werden soll (>0).
Acceleration	LREAL	Beschleunigung (>0)
Deceleration	LREAL	Verzögerung (>0)
Options	ST_MoveOptions	Datenstruktur (ST_MoveOptions [▶ 42]), die zusätzliche selten benötigte Parameter enthält. Im Normalfall kann der Eingang unbelegt bleiben.

Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 11].

Ein-/Ausgänge

```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 40]	Achsdatenstruktur, welche eine Achse eindeutig im System adressiert. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.

Ausgänge

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

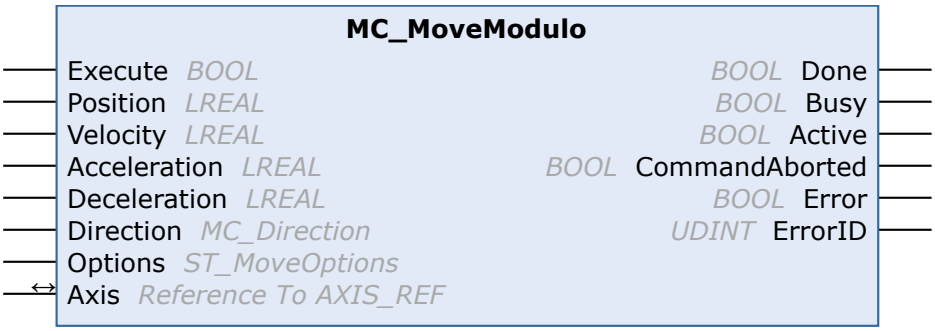
Name	Typ	Beschreibung
Done	BOOL	TRUE, wenn die Zielposition erreicht wurde.
Busy	BOOL	TRUE, sobald das Kommando mit „Execute“ gestartet wird und solange der Fahrbefehl abgearbeitet wird. Wenn „Busy“ FALSE ist, ist der Funktionsbaustein bereit für einen neuen Auftrag. Gleichzeitig ist einer der Ausgänge „Done“, „CommandAborted“ oder „Error“ gesetzt.
Active	BOOL	Zeigt an, dass das Kommando ausgeführt wird. Wenn das Kommando gepuffert wurde, wird es evtl. erst aktiv, nachdem ein laufendes Kommando beendet ist.
CommandAborted	BOOL	TRUE, wenn das Kommando nicht vollständig ausgeführt werden konnte. Die Achse wurde gestoppt oder das laufende Kommando wurde durch ein weiteres Move-Kommando abgelöst.
Error	BOOL	TRUE, wenn ein Fehler auftritt.
ErrorID	UDINT	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 11].

Voraussetzungen

Entwicklungsumgebung	Einzubindende SPS-Bibliotheken
TwinCAT 3.1.4024.11	Tc3_DriveMotionControl

6.3.3 MC_MoveModulo



Mit dem Funktionsbaustein MC_MoveModulo wird eine Positionierung durchgeführt, die sich auf die Modulo-Position einer Achse bezieht. Grundlage für eine Modulo-Umdrehung ist dabei der einstellbare Parameter „Modulo-Faktor“ der AXIS_REF-Struktur(Axis.Parameter.ModuloFactor z. B. 360°). Abhängig vom Eingang „Direction“ werden drei mögliche Starttypen unterschieden.

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

Start einer Achse aus dem Stillstand

Wird eine Achse mit MC_MoveModulo aus dem Stillstand gestartet, so können auch Positionen größer oder gleich 360° angegeben werden, um zusätzliche volle Umdrehungen auszuführen.

Sonderfälle

Besonders zu beachten ist das Verhalten bei Anforderung einer oder mehrerer vollständiger Modulo-Umdrehungen. Befindet sich die Achse auf einer exakten Sollposition von beispielsweise 90° und wird sie auf 90° positioniert, wird keine Bewegung ausgeführt. Bei Anforderung von 450° in positiver Richtung fährt sie eine Umdrehung. Nach einem Achs-Reset 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°, sondern ein wenig darunter oder darüber. In diesen beiden Fällen wird entweder nur eine minimale Positionierung auf 90° oder aber eine ganze Umdrehung ausgeführt.

Je nach Anwendungsfall kann es für volle Modulo-Umdrehungen günstiger sein, die gewünschte Zielposition auf Grund der aktuellen absoluten Position zu berechnen und mit dem Baustein MC_MoveAbsolute [▶ 26] zu positionieren.

Siehe auch: Die Modulo-Positionierung [▶ 30]

Eingänge

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

```
    Direction    : MC_Direction;
    Options      : ST_MoveOptions;
END_VAR
```

Name	Typ	Beschreibung
Execute	BOOL	Mit einer steigenden Flanke wird das Kommando ausgeführt.
Position	LREAL	Modulo-Zielposition, auf die positioniert werden soll. Falls die Achse aus dem Stillstand gestartet wird, führen Positionen ab 360° zu zusätzlichen Umdrehungen. Negative Positionen sind nicht zulässig.
Velocity	LREAL	Maximale Geschwindigkeit mit der gefahren werden soll (>0).
Acceleration	LREAL	Beschleunigung (≥0)
Deceleration	LREAL	Verzögerung (≥0)
Direction	MC_Direction	Positive oder negative Fahrtrichtung (Typ: MC_Direction [▶ 42]). Falls die Achse aus der Bewegung heraus gestartet wird, darf die Richtung nicht umgekehrt werden.
Options	ST_MoveOptions	Datenstruktur (ST_MoveOptions [▶ 42]), die zusätzliche selten benötigte Parameter enthält. Im Normalfall kann der Eingang unbelegt bleiben.

Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 11].

Ein-/Ausgänge

```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 40]	Achsdatenstruktur, welche eine Achse eindeutig im System adressiert. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.

Ausgänge

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

Name	Typ	Beschreibung
Done	BOOL	TRUE, wenn die Zielposition erreicht wurde.
Busy	BOOL	TRUE, sobald das Kommando mit „Execute“ gestartet wird und solange der Fahrbefehl abgearbeitet wird. Wenn „Busy“ FALSE ist, ist der Funktionsbaustein bereit für einen neuen Auftrag. Gleichzeitig ist einer der Ausgänge „Done“, „CommandAborted“ oder „Error“ gesetzt.
Active	BOOL	Zeigt an, dass das Kommando ausgeführt wird. Wenn das Kommando gepuffert wurde, wird es evtl. erst aktiv, nachdem ein laufendes Kommando beendet ist.

Name	Typ	Beschreibung
CommandAborted	BOOL	TRUE, wenn das Kommando nicht vollständig ausgeführt werden konnte. Die Achse wurde gestoppt oder das laufende Kommando wurde durch ein weiteres Move-Kommando abgelöst.
Error	BOOL	TRUE, wenn ein Fehler auftritt.
ErrorID	UDINT	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

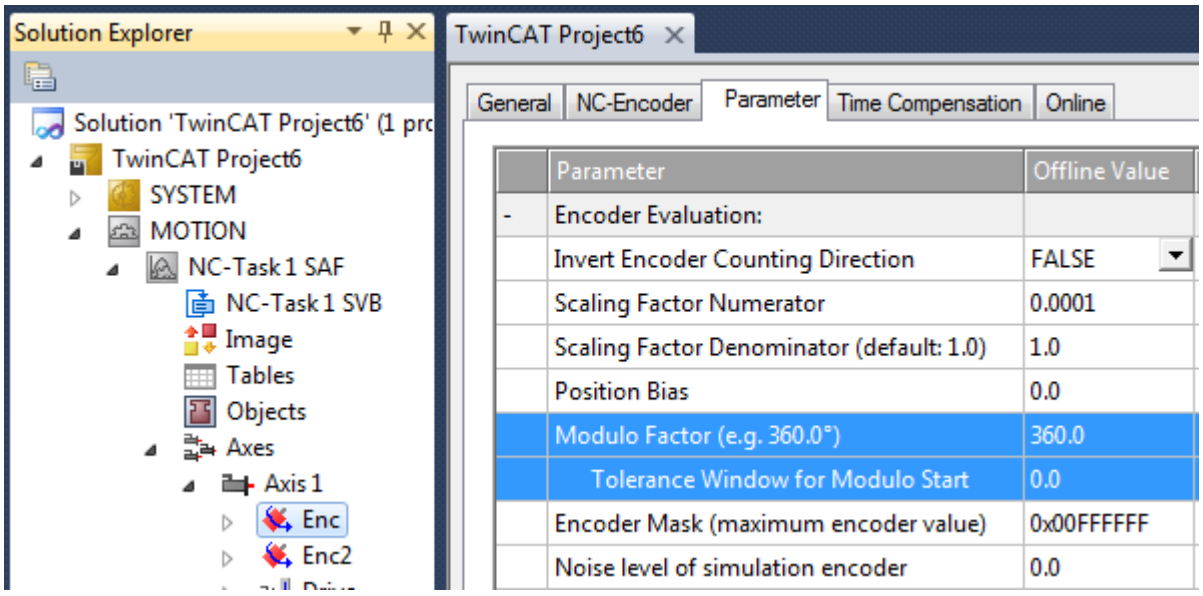
Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 11].

6.3.4 Die Modulo-Positionierung

Die Modulo-Positionierung (MC_MoveModulo [▶ 28]) 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 Risiken, da die gewünschte Zielposition unterschiedlich interpretiert werden kann.

Einstellungen in den Parametern der AXIS_REF

Die Modulo-Positionierung bezieht sich grundsätzlich auf eine unter *Axis-Parameter. ModuloFactor* des entsprechenden AXIS_REF [▶ 40] einzustellenden Modulo-Periode. Zusätzlich sind entsprechende Einstellungen in den Parametern des Drive (z. B. EL72xx) zu machen. In den Beispielen auf dieser Seite wird von einer rotatorischen Achse mit einer Modulo-Periode von 360° ausgegangen.

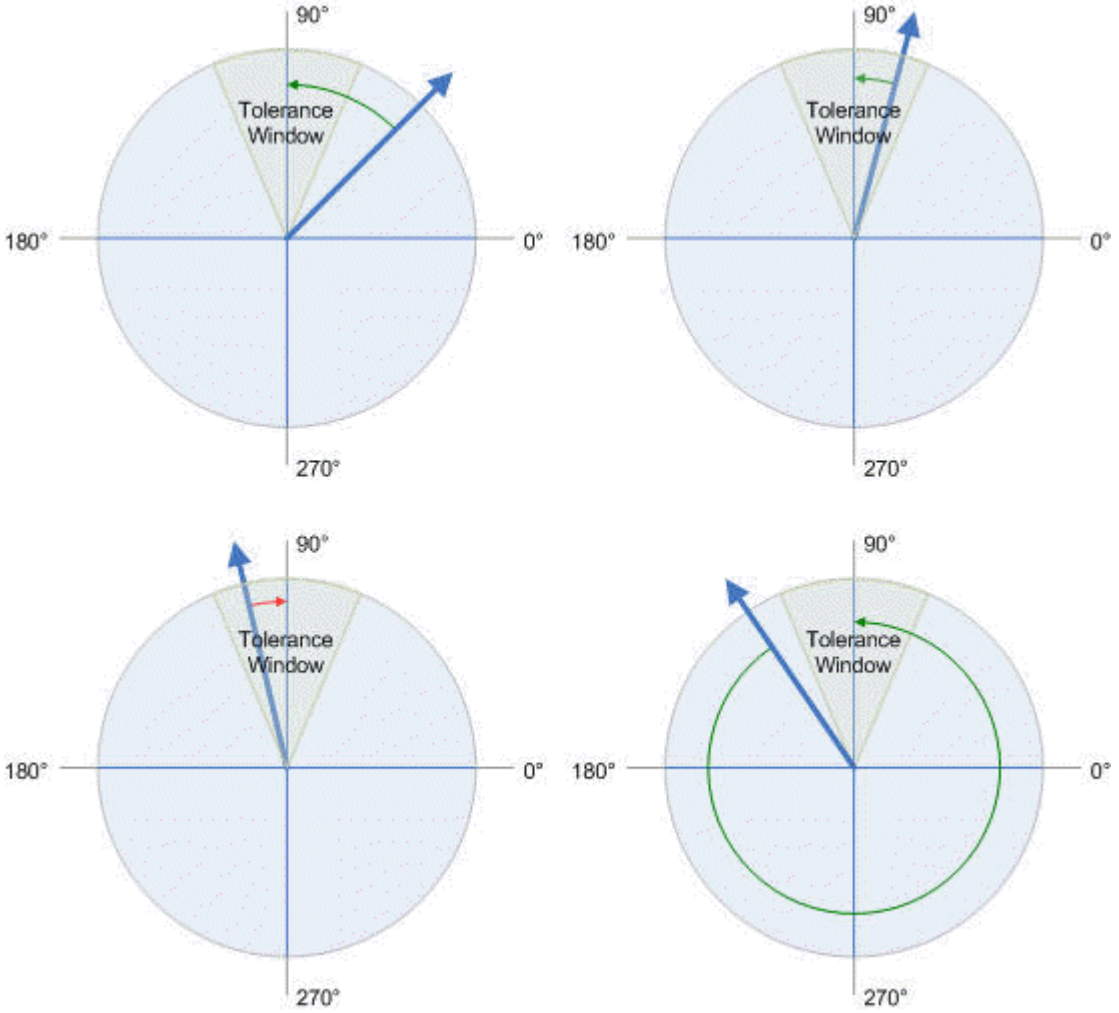


Besonderheiten beim Reset einer Achse

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 Achs-Reset (MC_Reset [▶ 14], Zuschalten der Reglerfreigabe mit MC_Power [▶ 13]) kann sich eine vom Anwender nicht erwartete Sollposition einstellen, da in diesem Fall die aktuelle Istposition als Sollposition übernommen wird. Der Achs-Reset 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 anschließend ein Achs-Reset durchgeführt, 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°, 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 Achs-Reset leicht über 90°, 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.



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° 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 muss 0° als Modulo-Zielposition angegeben 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 [▶ 42]). 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.

Bei Modulo-Positionierungen mit dem Starttyp „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-Starttyp)	Absolute Anfangs-position	Modulo-Zielposition	Relativer Verfahr-weg	Absolute Endposition	Modulo-Endposition
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, 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°.

Richtung (Modulo-Starttyp)	Absolute Anfangs-position	Modulo-Zielposition	Relativer Verfahr-weg	Absolute Endposition	Modulo-Endposition
Positive Richtung	90,00	90,00	0,00	90,00	90,00
Positive Richtung	91,10	90,00	358,90	450,00	90,00
Positive Richtung	88,90	90,00	1,10	90,00	90,00
Positive Richtung	90,00	450,00	360,00	450,00	90,00
Positive Richtung	91,10	450,00	718,90	810,00	90,00
Positive Richtung	88,90	450,00	361,10	450,00	90,00
Positive Richtung	90,00	810,00	720,00	810,00	90,00
Positive Richtung	91,10	810,00	1078,90	1170,00	90,00
Positive Richtung	88,90	810,00	721,10	810,00	90,00
Negative Richtung	90,00	90,00	0,00	90,00	90,00
Negative Richtung	91,10	90,00	-1,10	90,00	90,00
Negative Richtung	88,90	90,00	-358,90	-270,00	90,00
Negative Richtung	90,00	450,00	-360,00	-270,00	90,00
Negative Richtung	91,10	450,00	-361,10	-270,00	90,00
Negative Richtung	88,90	450,00	-718,90	-630,00	90,00
Negative Richtung	90,00	810,00	-720,00	-630,00	90,00
Negative Richtung	91,10	810,00	-721,10	-630,00	90,00
Negative Richtung	88,90	810,00	-1078,90	-990,00	90,00

Modulo-Berechnungen im SPS-Programm

Alle Positionieraufträge an einer Achse werden 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, muss diese Berechnung mit der aktuellen Sollposition der Achse durchgeführt werden (Axis.Status.ModuloSetPos und Axis.Status.ModuloSetTurns).

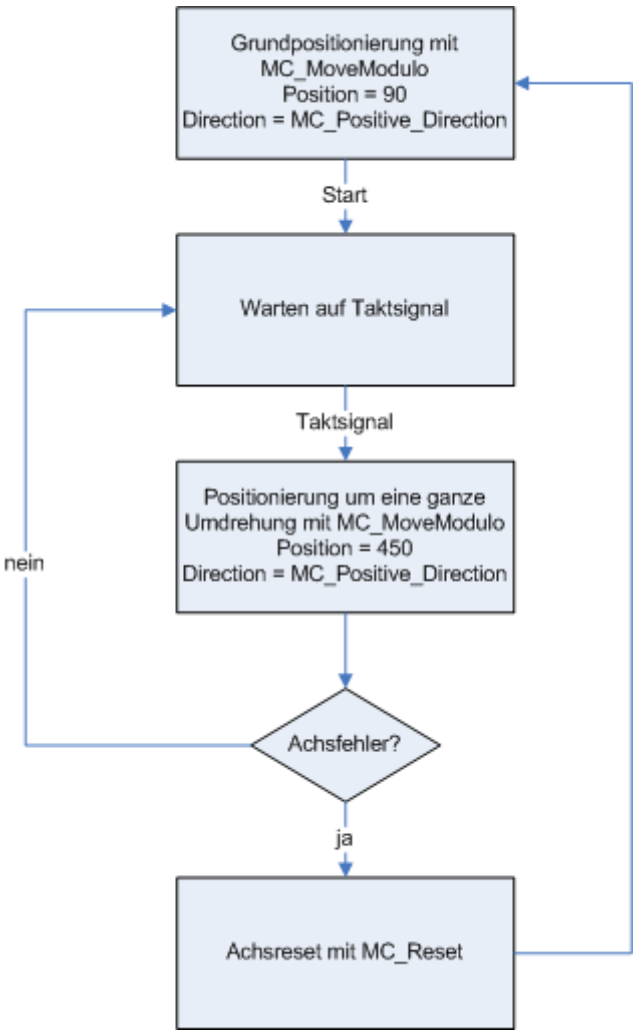
Es ist nicht zu empfehlen, Auftragsberechnungen auf Basis der Modulo-Istposition durchzuführen, die im Achsstatus (ModuloActPos und ModuloActTurns) zur Verfügung steht. Wegen der mehr oder weniger großen Regelabweichung der Achse könnten sich Fehler im programmierten Ablauf, wie z. B. unerwünschte Umdrehungen,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

Da die Achse nur vorpositioniert werden darf, wird das Bewegungskommando MC_MoveModulo [▶ 28] 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 [▶ 28]), mit der die exakte Ausgangsposition sichergestellt wird. Anschließend wechselt die Schrittkette in einen Bearbeitungszyklus. Im Fehlerfall wird die Achse mit MC_Reset [▶ 14] zurückgesetzt und anschließend, am Anfang der Schrittkette, auf ihre gültige Ausgangsposition gefahren. In diesem Fall wird 90° als Zielposition angegeben, damit diese Position schnellstmöglich angefahren wird. Steht die Achse bereits an der Ausgangsposition, so wird keine Bewegung ausgeführt.

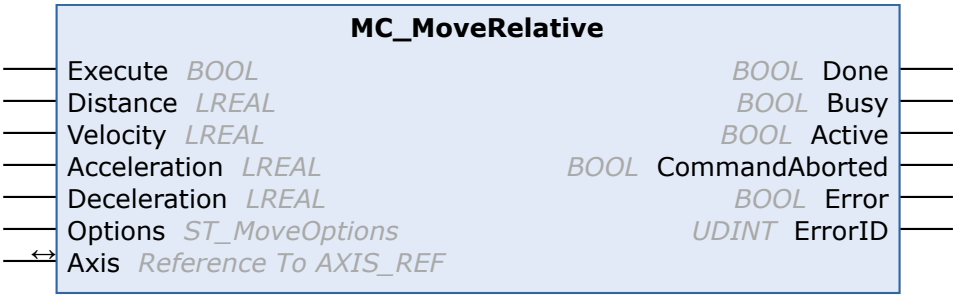


Der Reset-Schritt kann alternativ auch am Anfang der Schrittkette ausgeführt werden, um die Achse auch zu Beginn des Ablaufs zu initialisieren.

Voraussetzungen

Entwicklungsumgebung	Einzubindende SPS-Bibliotheken
TwinCAT 3.1.4024.11	Tc3_DriveMotionControl

6.3.5 MC_MoveRelative



Der Funktionsbaustein MC_MoveRelative startet eine relative Positionierung ausgehend von der aktuellen Sollposition und überwacht die Achsbewegung über den gesamten Fahrweg. Der Ausgang „Done“ wird gesetzt, wenn die Zielposition angefahren wurde. Anderenfalls wird der Ausgang „CommandAborted“ oder im Fehlerfall der Ausgang „Error“ gesetzt.

Eingänge

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

Name	Typ	Beschreibung
Execute	BOOL	Mit einer steigenden Flanke wird das Kommando ausgeführt.
Distance	LREAL	Relative Fahrstrecke, um die positioniert werden soll.
Velocity	LREAL	Maximale Geschwindigkeit mit der gefahren werden soll (>0).
Acceleration	LREAL	Beschleunigung (≥0)
Deceleration	LREAL	Verzögerung (≥0)
Options	ST_MoveOptions	Datenstruktur (ST_MoveOptions [▶ 42]), die zusätzliche selten benötigte Parameter enthält. Im Normalfall kann der Eingang unbelegt bleiben.

Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 11].

Ein-/Ausgänge

```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 40]	Achsdatenstruktur, welche eine Achse eindeutig im System adressiert. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.

Ausgänge

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

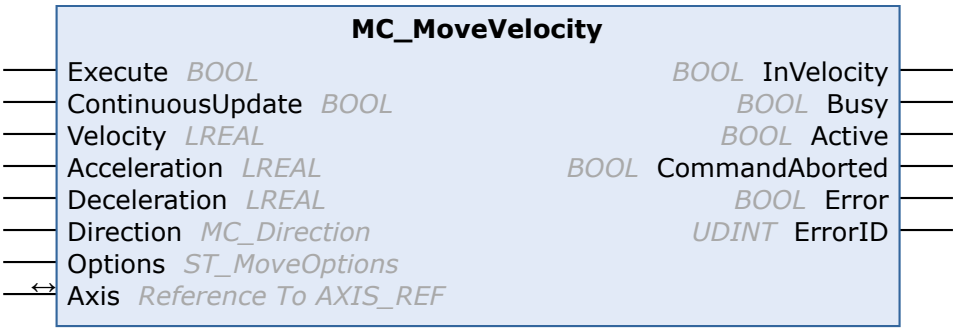
Name	Typ	Beschreibung
Done	BOOL	TRUE, wenn die Zielposition erreicht wurde.
Busy	BOOL	TRUE, sobald das Kommando mit „Execute“ gestartet wird und solange der Fahrbefehl abgearbeitet wird. Wenn „Busy“ FALSE ist, ist der Funktionsbaustein bereit für einen neuen Auftrag. Gleichzeitig ist einer der Ausgänge „Done“, „CommandAborted“ oder „Error“ gesetzt.
Active	BOOL	Zeigt an, dass das Kommando ausgeführt wird. Wenn das Kommando gepuffert wurde, wird es evtl. erst aktiv, nachdem ein laufendes Kommando beendet ist.
CommandAborted	BOOL	TRUE, wenn das Kommando nicht vollständig ausgeführt werden konnte. Die Achse wurde gestoppt oder das laufende Kommando wurde durch ein weiteres Move-Kommando abgelöst.
Error	BOOL	TRUE, wenn ein Fehler auftritt.
ErrorID	UDINT	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 11].

Voraussetzungen

Entwicklungsumgebung	Einzubindende SPS-Bibliotheken
TwinCAT 3.1.4024.11	Tc3_DriveMotionControl

6.3.6 MC_MoveVelocity



Der Funktionsbaustein MC_MoveVelocity startet eine Endlosfahrt mit vorgegebener Geschwindigkeit und Richtung. Die Bewegung kann durch ein Stopp-Kommando angehalten werden.

Der InVelocity-Ausgang wird gesetzt, sobald die konstante Geschwindigkeit erreicht ist. Mit Erreichen der Konstantfahrt ist die Funktion des Bausteins abgeschlossen, es findet also keine weitere Überwachung der Bewegung statt. Wenn das Kommando noch während der Beschleunigungsphase abgebrochen wird, wird der Ausgang „CommandAborted“ oder im Fehlerfall der Ausgang „Error“ gesetzt.

Eingänge

```
VAR_INPUT
    Execute      : BOOL;
    Velocity     : LREAL;
    Acceleration : LREAL;
    Deceleration : LREAL;
    Direction    : MC_Direction := MC_Positive_Direction;
    Options      : ST_MoveOptions;
END_VAR
```

Name	Typ	Beschreibung
Execute	BOOL	Mit einer steigenden Flanke wird das Kommando ausgeführt.
ContinuousUpdate	BOOL	Ist ContinuousUpdate TRUE, so können während der Abarbeitung des Kommandos, mit einer einer steigenden Flanke am Eingang Execute, die Dynamiken entsprechend der Eingänge Velocity, Acceleration und Deceleration verändert und schnellstmöglich zur Wirkung gebracht werden.
Velocity	LREAL	Maximale Geschwindigkeit mit der gefahren werden soll (>0).
Acceleration	LREAL	Beschleunigung (≥0)
Deceleration	LREAL	Verzögerung (≥0)
Direction	MC_Direction	Positive oder negative Fahrtrichtung (Typ: MC_Direction [▶ 42])
Options	ST_MoveOptions	Datenstruktur (ST_MoveOptions [▶ 42]), die zusätzliche selten benötigte Parameter enthält. Im Normalfall kann der Eingang unbelegt bleiben.

Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 11].

Ein-/Ausgänge

```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 40]	Achsdatenstruktur, welche eine Achse eindeutig im System adressiert. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.

Ausgänge

```
VAR_OUTPUT
    InVelocity     : BOOL; (* B *)
    Busy           : BOOL; (* E *)
    Active         : BOOL; (* E *)
    CommandAborted : BOOL; (* E *)
    Error          : BOOL; (* B *)
    ErrorID        : UDINT; (* E *)
END_VAR
```

Name	Typ	Beschreibung
InVelocity	BOOL	Nach der Achsbeschleunigung nimmt der InVelocity-Ausgang den Wert TRUE an, sobald die angeforderte Sollgeschwindigkeit erreicht worden ist.

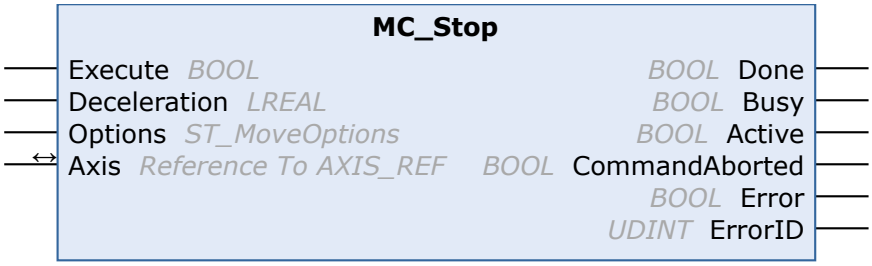
Name	Typ	Beschreibung
Busy	BOOL	TRUE, sobald das Kommando mit „Execute“ gestartet wird und solange der Funktionsbaustein aktiv ist. Wenn „Busy“ FALSE ist, ist der Funktionsbaustein bereit für einen neuen Auftrag. Gleichzeitig ist einer der Ausgänge „CommandAborted“ oder „Error“ gesetzt.
Active	BOOL	Zeigt an, dass das Kommando ausgeführt wird.
CommandAborted	BOOL	TRUE, wenn das Kommando nicht vollständig ausgeführt werden konnte. Die Achse wurde gestoppt oder das laufende Kommando wurde durch ein weiteres Move-Kommando abgelöst.
Error	BOOL	TRUE, wenn ein Fehler auftritt.
ErrorID	UDINT	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 11].

Voraussetzungen

Entwicklungsumgebung	Einzubindende SPS-Bibliotheken
TwinCAT 3.1.4024.11	Tc3_DriveMotionControl

6.3.7 MC_Stop



Der Funktionsbaustein MC_Stop hält eine Achse mit einer definierten Bremsrampe an und verriegelt die Achse gegen andere Bewegungskommandos. Der Baustein eignet sich daher für Stopps in besonderen Situationen, in denen eine weitere Bewegung der Achse unterbunden werden soll.

Die Achse wird gleichzeitig für andere Bewegungskommandos gesperrt. Erst wenn nach dem vollständigen Stopp das Execute-Signal auf FALSE gesetzt wird, kann die Achse wieder gestartet werden. Die Entriegelung der Achse nach der fallenden Flanke von Execute benötigt wenige Zyklen. Während dieser Phase bleibt der Ausgang „Busy“ TRUE und der Funktionsbaustein muss weiter aufgerufen werden, bis „Busy“ FALSE wird.

Mit einem MC_Reset [▶ 14] wird die Verriegelung der Achse aufgehoben.

Alternativ kann die Achse mit MC_Halt [▶ 25] ohne Verriegelung angehalten werden. MC_Halt ist für normale Bewegungsabläufe zu bevorzugen.

Eingänge

```
VAR_INPUT
    Execute      : BOOL;
    Deceleration : LREAL;
    Options      : ST_MoveOptions;
END_VAR
```

Name	Typ	Beschreibung
Execute	BOOL	Mit einer steigenden Flanke wird das Kommando ausgeführt. Die Achse wird während des Stopps verriegelt. Erst wenn nach dem vollständigen Stopp das Execute-Signal auf FALSE gesetzt wird, kann die Achse wieder gestartet werden.
Deceleration	LREAL	Verzögerung Bei einem Wert = 0 wirkt die mit dem letzten Move-Kommando parametrierte Verzögerung. MC_Stop und MC_Halt können aus Sicherheitsgründen nicht mit schwächerer Dynamik ausgeführt werden als der gerade aktive Fahrauftrag. Die Parametrierung wird gegebenenfalls automatisch angepasst.
Options	ST_MoveOptions	Datenstruktur (ST_MoveOptions [▶ 42]), die zusätzliche selten benötigte Parameter enthält. Im Normalfall kann der Eingang unbelegt bleiben.

Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 11].

Ein-/Ausgänge

```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 40]	Achsdatenstruktur, welche eine Achse eindeutig im System adressiert. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.

Ausgänge

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

Name	Typ	Beschreibung
Done	BOOL	TRUE, wenn die Achse gestoppt wurde und im Stillstand ist.
Busy	BOOL	TRUE, sobald das Kommando mit Execute gestartet wird und solange das Kommando abgearbeitet wird. Wenn „Busy“ FALSE ist, ist der Funktionsbaustein bereit für einen neuen Auftrag. „Busy“ bleibt TRUE, solange die Achse gesperrt ist. Erst nachdem Execute = FALSE gesetzt wird, wird die Achse entriegelt und „Busy“ wird FALSE.
Active	BOOL	Zeigt an, dass der Funktionsbaustein die Achse kontrolliert. Bleibt TRUE, solange die Achse gesperrt ist. Erst nachdem Execute = FALSE gesetzt wird, wird die Achse entriegelt und „Active“ FALSE.
CommandAborted	BOOL	TRUE, wenn das Kommando nicht vollständig ausgeführt werden konnte.
Error	BOOL	TRUE, wenn ein Fehler auftritt.

Name	Typ	Beschreibung
ErrorID	UDINT	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 11].

Voraussetzungen

Entwicklungsumgebung	Einzubindende SPS-Bibliotheken
TwinCAT 3.1.4024.11	Tc3_DriveMotionControl

7 Datentypen

7.1 Achsinterface

7.1.1 AXIS_REF

Der Datentyp AXIS_REF enthält Information zu einer Achse. AXIS_REF ist eine Schnittstelle zwischen der SPS und der DRIVE und wird den MC-Funktionsbausteinen als Referenz auf eine Achse mitgegeben.

```
TYPE AXIS_REF :
VAR_INPUT
    PlcToDrive AT %Q* : PLCTODRIVE_AXIS_REF;
    Parameter         : ST_AxisParameter;
END_VAR
VAR_OUTPUT
    DriveToPlc AT %I* : DRIVETOPLC_AXIS_REF;
    WcState    AT %I* : DRIVETOPLC_WCSTATE;
    InfoData   AT %I* : DRIVETOPLC_INFODATA;
    Status            : ST_AxisStatus;
END_VAR
END_TYPE
```

Elemente von AXIS_REF

Name	Beschreibung
PlcToDrive	Datenstruktur, die zyklisch zwischen SPS und DRIVE ausgetauscht wird. Über diese Datenstruktur kommunizieren die MC-Funktionsbausteine mit dem DRIVE und senden Kontrollinformation und Kommandos von der SPS zum DRIVE. Die Datenstruktur wird automatisch im Ausgangsprozessabbild der SPS platziert und muss in der TwinCAT Entwicklungsumgebung mit dem Eingangsprozessabbild eines DRIVE verbunden werden.
Parameter	Datenstruktur zur Parametrierung der Achse (Typ: ST_AxisParameters [▶ 43]).
DriveToPlc	Datenstruktur, die zyklisch zwischen SPS und DRIVE ausgetauscht wird. Über diese Datenstruktur kommunizieren die MC-Funktionsbausteine mit dem DRIVE und empfangen Statusinformationen vom DRIVE. Die Datenstruktur wird automatisch im Eingangsprozessabbild der SPS platziert und muss im TwinCAT System Manager mit dem Ausgangsprozessabbild eines DRIVE verbunden werden. Die DRIVETOPLC-Struktur enthält alle wesentlichen Zustandsinformationen einer Achse wie Position, Geschwindigkeit und Auftragszustand. Da der Datenaustausch zyklisch stattfindet, kann die SPS jederzeit ohne zusätzlichen Kommunikationsaufwand auf den aktuellen Achszustand zugreifen.
WcState	Datenstruktur, die zyklisch zwischen SPS und DRIVE ausgetauscht wird und den WcState des DRIVE beinhaltet. Die Datenstruktur wird automatisch im Eingangsprozessabbild der SPS platziert und muss in der TwinCAT Entwicklungsumgebung mit dem Ausgangsprozessabbild eines DRIVE verbunden werden (Typ: DRIVETOPLC_WCSTATE [▶ 41]).
InfoData	Datenstruktur, die zyklisch zwischen SPS und DRIVE ausgetauscht wird und ADS-Informationen für den Zugriff auf DRIVE-Parameter enthält. Die Datenstruktur wird automatisch im Eingangsprozessabbild der SPS platziert und muss in der TwinCAT Entwicklungsumgebung mit dem Ausgangsprozessabbild eines DRIVE verbunden werden (Typ: DRIVETOPLC_INFODATA [▶ 41]).

Name	Beschreibung
Status	Datenstruktur, die zusätzliche oder aufbereitete Statusinformation zu einer Achse enthält (Typ: ST_AxisStatus [▶ 43]). Diese Datenstruktur wird nicht zyklisch aufgefrischt, sondern muss durch das SPS-Programm aktualisiert werden. Dazu kann MC_ReadStatus oder alternativ die Aktion „ReadStatus“ von AXIS_REF aufgerufen werden (siehe Beispiel).

Beispiel:

```
VAR
 Axis1 : AXIS_REF (* axis data structure for Axis-1 *)
END_VAR

(* program code at the beginning of each PLC cycle *)
Axis1.ReadStatus();

(* alternative program code at the beginning of each PLC cycle *)
Axis1();
```

Der Aufruf von „ReadStatus“ bzw. „Axis1“ sollte einmalig am Anfang jedes SPS-Zyklus getätigt werden. Anschließend kann innerhalb des gesamten SPS-Programms auf die aktuelle Statusinformation in AXIS_REF zugegriffen werden. Innerhalb eines Zyklus ändert sich der Status nicht.

7.1.2 DRIVETOPLC_INFODATA

Die Struktur DRIVETOPLC_INFODATA ist Teil der Struktur AXIS_REF [▶ 40].

```
TYPE DRIVETOPLC_INFODATA :
    State      : UINT;
    AdsAddr    : AMSADDR;
END_TYPE
```

7.1.3 DRIVETOPLC_WCSTATE

Die Struktur DRIVETOPLC_WCSTATE ist Teil der Struktur AXIS_REF [▶ 40].

```
TYPE DRIVETOPLC_WCSTATE :
    WcState     : BIT;
    InputToggle : BIT;
END_TYPE
```

7.2 Homing

7.2.1 E_EncoderReferenceMode

```
TYPE E_EncoderReferenceMode :
(
    EncoderReferenceMode_CamAtDigitalInput,    (* Cam is connected to  digital input*)
);
END_TYPE
```

7.2.2 MC_HomingMode

Dieser Datentyp wird zur Parametrierung des Funktionsbausteins MC_Home [▶ 21] verwendet.

```
TYPE MC_HomingMode :
(
    MC_DefaultHoming       :=0,  (* default homing as defined in the SystemManager encoder parameter
s *)
    MC_Direct              :=4,  (* Static Homing forcing position from user reference *)
    MC_Block               :=6,  (* Homing against hardware parts blocking movement *)
    MC_ForceCalibration    :=7,  (* set the calibration flag without perfomring any motion or changi
ng the position *)
    MC_ResetCalibration    :=8   (* resets the calibration flag without perfomring any motion or cha
```

```
nging the position *)
);
END_TYPE
```

7.2.3 ST_HomingOptions

```
TYPE ST_HomingOptions :
STRUCT
(
    SearchDirection   : MC_Direction := MC_Direction.MC_Undefined_Direction;
                (* search direction *)
    SearchVelocity    : LREAL;
                (* search velocity *)
    SyncDirection     : MC_Direction := MC_Direction.MC_Undefined_Direction;
                (* synchronization direction *)
    SyncVelocity      : LREAL;
                (* synchronization velocity *)
    ReferenceMode     : E_EncoderReferenceMode := E_EncoderReferenceMode.ENCODERREFERENCEMODE_CAMATD
IGITALINPUT;    (* Mode of reference sequence *)

    Acceleration      : LREAL;
                (* user defined acceleration *)
    Deceleration      : LREAL;
                (* user defined deceleration *)
);
END_TYPE
```

Name	Datentyp	Beschreibung
SearchDirection	MC_Direction	Richtung, in der die Referenznocke gesucht werden soll.
SearchVelocity	LREAL	Geschwindigkeit, mit der die Referenznocke gesucht werden soll.
SyncDirection	MC_Direction	Richtung, in der die fallende Flanke der Referenznocke gesucht wird, nachdem die Referenznocke erkannt wurde.
ReferenceMode	E_EncoderReferenceMode	Mode der Referenzierung (Momentan nur ENCODERREFERENCEMODE_CAMATDIGITALINPUT).
Acceleration	LREAL	Beschleunigung für die Referenzierfahrt
Deceleration	LREAL	Verzögerung für die Referenzierfahrt

7.3 Motion

7.3.1 MC_Direction

Dieser Aufzählungstyp enthält die möglichen Bewegungsrichtungen für die Funktionsbausteine MC_MoveVelocity [▶ 35] und MC_MoveModulo [▶ 28].

```
TYPE MC_Direction :
(
    MC_Undefined_Direction  := 0,
    MC_Positive_Direction   := 1,
    MC_Shortest_Way         := 2,
    MC_Negative_Direction   := 3
);
END_TYPE
```

7.3.2 ST_MoveOptions

Dieser Datentyp ist für mögliche, zukünftige optionale Einstellungen für Fahrkommandos wie MC_MoveAbsolute oder MC_Halt vorgesehen.

```
TYPE ST_MoveOptions :
STRUCT
END_STRUCT
END_TYPE
```

7.4 Status und Parameter

7.4.1 MC_AxisStates

Dieser Datentyp beschreibt die Betriebszustände nach dem PlcOpen-Zustandsdiagramm [▶ 9].

```
TYPE MC_AxisStates :
(
    MC_AXISSTATE_UNDEFINED,
    MC_AXISSTATE_DISABLED,
    MC_AXISSTATE_STANDSTILL,
    MC_AXISSTATE_ERRORSTOP,
    MC_AXISSTATE_STOPPING,
    MC_AXISSTATE_HOMING,
    MC_AXISSTATE_DISCRETEMOTION,
    MC_AXISSTATE_CONTINOUSMOTION
);
END_TYPE
```

Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 11].

7.4.2 ST_AxisParameters

Dieser Datentyp enthält grundlegende, notwendige Parameter einer Achse.

```
TYPE ST_AxisParameters :
STRUCT
    EncoderScalingFactor    : LREAL := 360.0/4096.0;  // Default for 360° and 4096 increments per
revolution
    MaxVelocity             : LREAL := 3600.0;       // Default for 10 revolutions per second

    ModuloFactor            : LREAL := 360.0;
END_STRUCT
END_TYPE
```

Formel zur Berechnung der Parameter:

$$EncoderScalingFactor = \frac{Feed\ constant}{Position\ Increments} = \frac{360°}{32\ Bit} = \frac{360°}{2^{32}} = 8.381903173490871 * 10^{-8}$$

$$MaxVelocity = Motor\ speed\ limitation * Feed\ constant = \frac{1615}{60}\frac{Revolutions}{s} * 360\frac{°}{Revolution} = 9690.0\frac{°}{s}$$

Motor speed Limitation

„Motor speed Limitation“ ist jeweils abhängig von der konfigurierten Spannung und dem verwendeten Motor. Für die Servoklemmen kann dieser Wert z. B. aus dem CoE-Objekt 0x8011:1B abgelesen werden.

7.4.3 ST_AxisStatus

Dieser Datentyp enthält umfangreiche Statusinformationen über eine Achse. Die Datenstruktur muss in jedem SPS-Zyklus durch einen Aufruf von MC_ReadStatus oder durch einen Aufruf der Aktion Axis.ReadStatus() oder Axis() (AXIS_REF [▶ 40]) aktualisiert werden.

```
TYPE ST_AxisStatus :
STRUCT
    AxisId              : UDINT;
    AxisName            : STRING;

    ActPos              : LREAL;
    ModuloActPos        : LREAL;
    ActVelo             : LREAL;
    ActAcceleration     : LREAL;
```

```
    SetPos               : LREAL;
    ModuloSetPos         : LREAL;
    SetVelo              : LREAL;
    SetAcceleration      : LREAL;

    PosDiff              : LREAL;

    TargetPosition       : LREAL;
    TargetVelocity       : LREAL;
    TargetAcceleration   : LREAL;
    TargetDeceleration   : LREAL;

    InfoData1            : LREAL;
    InfoData2            : LREAL;
    InfoData3            : LREAL;

    DigitalInput1        : BOOL;
    DigitalInput2        : BOOL;

    CmdNo                : UINT;
    CmdState             : UINT;

    MotionState          : MC_AxisStates; (* motion state in the PLCopen state diagram *)

    Error                : BOOL;  (* axis error state *)
    ErrorId              : UDINT; (* axis error code *)

    (* statemachine states: *)
    ErrorStop            : BOOL;
    Disabled             : BOOL;
    Stopping             : BOOL;
    StandStill           : BOOL;
    DiscreteMotion       : BOOL;
    ContinuousMotion     : BOOL;
    Homing               : BOOL;

    (* additional status *)
    ConstantVelocity     : BOOL;
    Accelerating         : BOOL;
    Decelerating         : BOOL;

    (* Status *)
    Operational          : BOOL;
    ControlLoopClosed    : BOOL; (* operational and position control active *)
    HasJob               : BOOL;
    HasBeenStopped       : BOOL;
    InTargetPosition     : BOOL;
    Protected            : BOOL;
    Homed                : BOOL;
    HomingBusy           : BOOL;
    MotionCommandsLocked : BOOL; (* stop 'n hold *)

    Moving               : BOOL;
    PositiveDirection    : BOOL;
    NegativeDirection    : BOOL;
    NotMoving            : BOOL;

    PTPmode              : BOOL;
    DriveDeviceError     : BOOL;
    DrivePositioningError: BOOL;
    IoDataInvalid        : BOOL;
END_STRUCT
END_TYPE
```

7.5 Touch Probe

7.5.1 E_SignalEdge

`Edge` legt fest, ob die steigende oder fallende Flanke des Trigger-Signals ausgewertet wird.

```
TYPE E_SignalEdge :
(
RisingEdge,
```

```
FallingEdge
) UINT;
END_TYPE
```

Name	Beschreibung
RisingEdge	Steigende Signalflanke
FallingEdge	Fallende Signalflanke

7.5.2 E_SignalSource

```
TYPE E_SignalSource :
(
    SignalSource_Default,    (* undefined or externally configured *)
    SignalSource_ZeroPulse := 128, (* encoder zero pulse *)
);
END_TYPE
```

Name	Beschreibung
SignalSource	Definiert optional die Signalquelle, soweit diese über die Steuerung gewählt werden kann. In vielen Fällen wird die Signalquelle fest im Antrieb konfiguriert und sollte dann auf den Standardwert „SignalSource_Default“ eingestellt sein.

7.5.3 E_TouchProbe

```
TYPE E_TouchProbe :
(
    TouchProbe1 := 1, (* 1st hardware probe unit with Sercos, CanOpen, KL5xxx and others *)
    PlcEvent := 10    (* simple PLC signal TRUE/FALSE *)
);
END_TYPE
```

7.5.4 MC_TouchProbeRecordedData

```
TYPE MC_TouchProbeRecordedData :
STRUCT
    Counter             : LREAL;
    RecordedPosition    : LREAL;
    AbsolutePosition    : LREAL;
    ModuloPosition      : LREAL;
END_STRUCT
END_TYPE
```

Name	Datentyp	Beschreibung
Counter	LREAL	Zähler, der anzeigt, wie viele gültige Flanken im letzten Zyklus erkannt wurden. Eine Erkennung mehrerer Flanken ist nur im Mode = TOUCHPROBEMODE_CONTINIOUS unter SERCOS/SOE implementiert und muss explizit von der Hardware unterstützt werden (z. B. AX5000).
RecordedPosition	LREAL	Erfasste Achsposition zum Zeitpunkt des Trigger-Signals. Diese entspricht je nach Parametrierung der absoluten Achsposition oder der Modulo-Achsposition.
AbsolutePosition	LREAL	Erfasste absolute Achsposition zum Zeitpunkt des Trigger-Signals.
ModuloPosition	LREAL	Erfasste Modulo-Achsposition zum Zeitpunkt des Trigger-Signals.

7.5.5 TRIGGER_REF

```
TYPE TRIGGER_REF :
STRUCT
    TouchProbe      : E_TouchProbe; (* probe unit definition *)
    SignalSource    : E_SignalSource; (* optional physical signal source used by the probe unit *)
```

```
    Edge            : E_SignalEdge; (* rising or falling signal edge *)
    Mode            : E_TouchProbeMode; (* single shot or continuous monitoring *)
    PlcEvent        : BOOL; (* PLC trigger signal input when TouchProbe signal source is set to 'Plc
Event' *)
    ModuloPositions : BOOL; (* interpretation of FirstPosition, LastPosition and RecordedPosition as
 modulo positions when TRUE *)
END_STRUCT
END_TYPE
```

TouchProbe: Definiert die verwendete Latch-Einheit (Probe-Unit) innerhalb der verwendeten Encoder-Hardware.

```
TYPE E_TouchProbe :
(
    TouchProbe1  := 1, (* 1st hardware probe unit *)
    PlcEvent     := 10    (* simple PLC signal TRUE/FALSE *)
);
END_TYPE
```

SignalSource: Definiert optional die Signalquelle, soweit diese über die Steuerung gewählt werden kann. In vielen Fällen wird die Signalquelle fest im Antrieb konfiguriert und sollte dann auf den Standardwert „SignalSource_Default“ eingestellt sein.

```
TYPE E_SignalSource :
(
    SignalSource_Default,          (* undefined or externally configured *)
    SignalSource_ZeroPulse := 128, (* encoder zero pulse *)
);
END_TYPE
```

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

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

Mode: Legt die Betriebsart der Latch-Einheit fest. Im Single-Mode wird nur die erste Flanke erfasst.

```
TYPE E_TouchProbeMode :
(
    TOUCHPROBEMODE_SINGLE := 1
);
END_TYPE
```

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

ModuloPositions: Wenn die Variable „ModuloPositions“ FALSE ist, wird die Achsposition in einem absoluten linearen Bereich von - ∞ bis + ∞ interpretiert. Die Positionen „FirstPosition“, „LastPosition“ und „RecordedPosition“ des Funktionsbausteins MC_TouchProbe [▶ 17] 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.

8 Globale Konstanten

8.1 Bibliotheksversion

Alle Bibliotheken haben eine bestimmte Version. Diese Version ist u. a. im SPS-Bibliotheks-Repository zu sehen. Eine globale Konstante enthält die Information über die Bibliotheksversion:

Global_Version

```
VAR_GLOBAL CONSTANT
    stLibVersion_Tc3_DriveMotionControl : ST_LibVersion;
END_VAR
```

stLibVersion_Tc3_DriveMotionControl: Versionsinformation der Tc3_DriveMotionControl-Bibliothek (Typ: ST_LibVersion).

Um zu sehen, ob die Version, die Sie haben auch die Version ist, die Sie brauchen, benutzen Sie die Funktion F_CmpLibVersion (definiert in der Tc2_System-Bibliothek).

Versionen vergleichen

Alle anderen Möglichkeiten Bibliotheksversionen zu vergleichen, die Sie von TwinCAT 2 kennen, sind veraltet.

9 Support und Service

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

Downloadfinder

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

Die Downloads sind in verschiedenen Formaten erhältlich.

Beckhoff Niederlassungen und Vertretungen

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

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

Dort finden Sie auch weitere Dokumentationen zu Beckhoff Komponenten.

Beckhoff Support

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

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

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

Beckhoff Service

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

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

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

Beckhoff Unternehmenszentrale

Beckhoff Automation GmbH & Co. KG

Hülshorstweg 20
33415 Verl
Deutschland

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

Mehr Informationen:
www.beckhoff.com/te1000

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