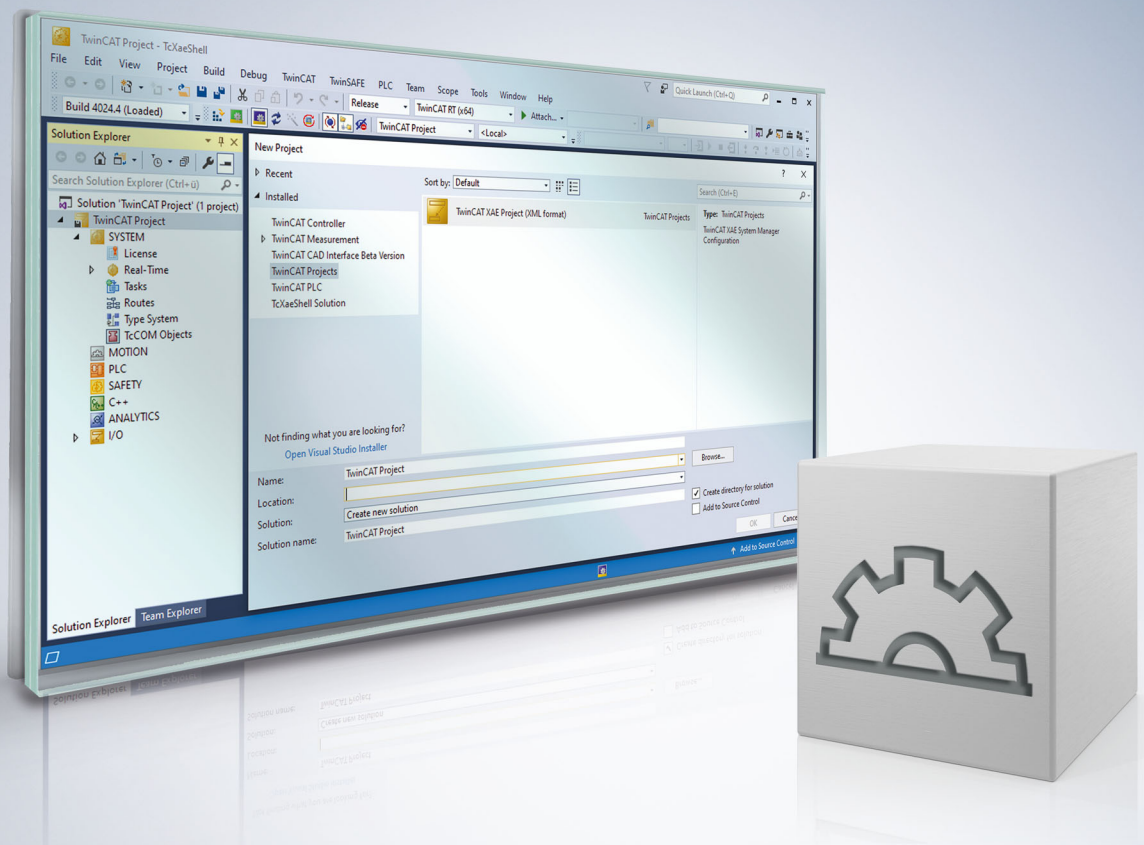
BECKHOFF New Automation Technology

Handbuch | DE

TE1000

TwinCAT 3 | PLC-Bibliothek: Tc2_MC2



04.08.2025 | Version: 2.15.4

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®, ATRO®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, MX-System®, Safety over EtherCAT®, TC/BSD®, TwinCAT®, TwinCAT/BSD®, TwinSAFE®, XFC®, XPlanar® und XTS® 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 Kennzeichnungen führen.

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

Copyright

Fremdmarken

In dieser Dokumentation können Marken Dritter verwendet werden. Die zugehörigen Markenvermerke finden Sie unter: https://www.beckhoff.com/trademarks.

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.

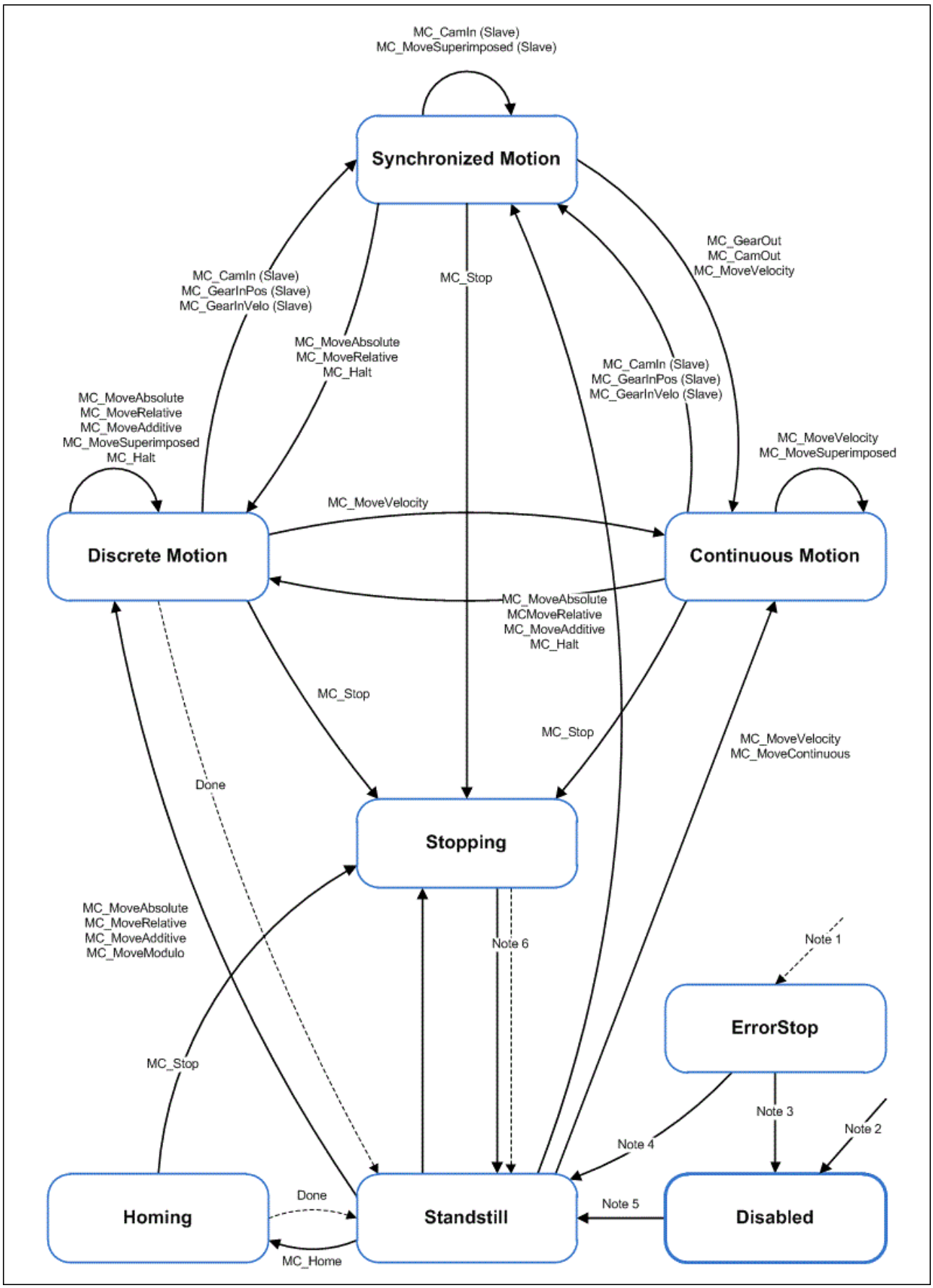
2 Übersicht

Die TwinCAT Motion Control SPS-Bibliothek Tc2_MC2 enthält Funktionsbausteine zur Programmierung von Maschinenapplikationen. Sie richtet sich nach der PLCopen-Spezifikation für Motion-Control-Funktionsbausteine V2.0 (www.PLCopen.org).

Dies ist eine konvertierte TwinCAT-2-TcMC2-Bibliothek. Bestehende Projekte, die noch die TwinCAT-2-TcMC-Bibliothek verwenden, müssen zunächst an die TcMC2-Bibliothek angepasst werden, bevor sie für TwinCAT 3 konvertiert werden können.

3 Zustandsdiagramm

Das folgende Zustandsdiagramm definiert das Verhalten einer Achse für den Fall, dass mehrere Funktionsbausteine für diese Achse gleichzeitig aktiv sind. Die Kombination mehrerer Funktionsbausteine ist nützlich, um komplexere Bewegungsprofile zu erzeugen oder um in einem Programmablauf Ausnahmezustände zu behandeln.



Note 1	Aus irgendeinem Zustand, in dem ein Fehler auftritt
Note 2	Aus irgendeinem Zustand, wenn MC_Power.Enable = FALSE und 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, wiederspiegelt. Der Zustand der Achse wechselt sofort mit dem auslösenden Kommando.

Das Zustandsdiagramm bezieht sich auf Einzelachsen. Multi-Achsbausteine, wie MC_CamIn oder MC_GearIn, beeinflussen die Zustände mehrerer Achsen, die jedoch immer auf einen Einzelachszustand der beteiligten Achsen zurückgeführt werden können. Beispielsweise kann ein Kurvenscheiben-Master im Zustand „Continuous Motion“ und der zugehörige Slave im Zustand „Synchronized Motion“ sein. Das Ankoppeln eines Slaves hat keinen Einfluss auf den Zustand des Masters.

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 nicht (MC_ReadStatus, MC_ReadAxisError, MC_ReadParameter, MC_ReadBoolParameter, MC_WriteParameter, MC_WriteBoolParameter, MC_ReadActualPosition und MC_CamTableSelect).

Der Zustand „Stopping“ zeigt an, dass sich die Achse in einer Stopprampe befindet. Der Zustand wechselt nach dem vollständigen Stopp nach „Standstill“.

Bewegungskommandos wie MC_MoveAbsolute, die aus dem Zustand „Synchronized Motion“ herausführen, sind nur dann möglich, wenn sie in den Achsparametern explizit erlaubt werden. Abkoppelkommandos wie MC_GearOut sind unabhängig davon möglich.

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 [▶ 90]. 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“, „InGear“, „InVelocity“, „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 oder es muss während der Kommandoausführung eine zweite Instanz des Funktionsbausteins mit neuen Parametern getriggert werden.

Wenn ein Eingangsparameter nicht an den Funktionsbaustein übergeben wird, bleibt der zuletzt an diesen Baustein übergebene Wert gültig. Beim ersten Aufruf ist ein sinnvoller Default-Wert gültig.

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
Jerk	mm/s3

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“, „InVelocity“, „InGear“ und „InSync“ 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). Kommunikationsfehler 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.
- Achsfehler (logische NC-Achse) treten üblicherweise während der Fahrt auf (z. B. Schleppabstandsfehler) und führen dazu, dass die Achse in den Fehlerzustand geht. Ein Achsfehler muss mit dem Bewegungskommando MC_Reset [▶ 19] zurückgesetzt werden.
- Antriebsfehler (Regelgerät) bewirken evtl. einen Achsfehler, also ein Fehler der logischen NC-Achse. In vielen Fällen können Achsfehler und Antriebsfehler gemeinsam durch das Bewegungskommando MC_Reset [▶ 19] zurückgesetzt werden. Abhängig vom Antriebsregler kann aber auch ein separater Reset-Mechanismus notwendig sein (z. B. Schalten einer Reset-Leitung zum Regelgerät).

Verhalten des Done-Ausgangs

Der Ausgang „Done“ (oder auch alternativ „InVelocity“, „InGear“, „InSync“ etc.) wird gesetzt, wenn ein Kommando erfolgreich ausgeführt wurde. Wenn mit mehreren Funktionsbausteinen an einer Achse gearbeitet wird und das laufende Kommando durch einen weiteren Baustein unterbrochen wird, wird der Ausgang „Done“ des ersten Bausteins nicht gesetzt.

Verhalten des CommandAborted-Ausgangs

Der Ausgang „CommandAborted“ wird gesetzt, wenn ein Kommando durch einen anderen Baustein 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

Wenn die Bewegung einer Achse durch mehrere Funktionsbausteine gesteuert wird, so zeigt der Ausgang „Active“ eines Bausteins an, dass das Kommando durch die Achse ausgeführt wird. Der Zustand Busy = TRUE und Active = FALSE bedeutet, dass das Kommando noch nicht oder nicht mehr ausgeführt wird.

Enable-Eingang und Valid-Ausgang

Im Gegensatz zum Eingang „Execute“ führt der Eingang „Enable“ dazu, dass eine Aktion permanent und wiederholt ausgeführt wird, solange „Enable“ TRUE ist. Der Funktionsbaustein MC_ReadStatus [▶ 30] aktualisiert beispielsweise zyklisch den Zustand einer Achse, solange „Enable“ TRUE ist. Ein Funktionsbaustein mit einem Eingang „Enable“ zeigt durch den Ausgang „Valid“ an, dass die an den Ausgängen angezeigten Daten gültig sind. Solange der Ausgang „Valid“ TRUE ist, können die Daten jedoch ständig aktualisiert werden.

BufferMode

Einige Funktionsbausteine haben einen Eingang „BufferMode“, über den der Kommandofluss mit mehreren Funktionsbausteinen geregelt wird. Der BufferMode kann beispielsweise festlegen, dass ein Kommando ein anderes unterbricht (ungepufferter Betrieb) oder dass das nachfolgende Kommando erst im Anschluss an

das vorherige Kommando ausgeführt wird (gepufferter Betrieb). Im gepufferten Betrieb kann über den BufferMode ebenfalls festgelegt werden, wie die Bewegung am Übergang von einem Kommando zum nächsten aussehen soll. Man spricht hier von einem „Blending“, das die Geschwindigkeit am Übergangspunkt festlegt.

Um den BufferMode zu verwenden, ist immer ein zweiter Funktionsbaustein nötigt. Es ist nicht möglich, einen Move-Baustein mit neuen Parametern zu triggern, während er noch aktiv ist.

Im ungepufferten Betrieb führt ein nachfolgendes Kommando zu einem Abbruch eines laufenden Kommandos. Das vorherige Kommando setzt dann den Ausgang „CommandAborted“. Im gepufferten Betrieb wartet ein nachfolgendes Kommando, bis ein laufendes Kommando beendet wurde. Hier ist zu beachten, dass eine Endlosbewegung kein gepuffertes Folgekommando zulässt. Gepufferte Kommandos führen immer sofort zum Abbruch einer Endlosbewegung wie im ungepufferten Betrieb.

Es wird nur ein Kommando gepuffert, während ein anderes ausgeführt wird. Wird mehr als ein Kommando während eines laufenden Kommandos getriggert, so wird das zuletzt gestartete und zu puffernde Kommando mit einem Fehler abgewiesen (Fehler 0x4292 Buffer Full). Falls jedoch das letzte Kommando ungepuffert gestartet wird, so wird es in jedem Fall aktiv und unterbricht das laufende und ein bereits gepuffertes Kommando.

BufferModes

- Aborting: Default-Modus ohne Pufferung. Das Kommando wird sofort ausgeführt und unterbricht gegebenenfalls ein laufendes Kommando.
- Buffered: Das Kommando wird ausgeführt, sobald die Achse kein anderes Kommando mehr ausführt. Die vorherige Bewegung wird bis zum Stopp ausgeführt und das nachfolgende Kommando wird aus dem Stillstand gestartet.
- BlendingLow: Das Kommando wird ausgeführt, sobald die Achse kein anderes Kommando mehr ausführt. Im Gegensatz zu Buffered stoppt die Achse an der vorherigen Zielposition nicht, sondern durchläuft diese Position mit der niedrigeren Geschwindigkeit zweier Kommandos.
- BlendingHigh Das Kommando wird ausgeführt, sobald die Achse kein anderes Kommando mehr ausführt. Im Gegensatz zu Buffered stoppt die Achse an der vorherigen Zielposition nicht, sondern durchläuft diese Position mit der höheren Geschwindigkeit zweier Kommandos.
- BlendingNext: Das Kommando wird ausgeführt, sobald die Achse kein anderes Kommando mehr ausführt. Im Gegensatz zu Buffered stoppt die Achse an der vorherigen Zielposition nicht, sondern durchläuft diese Position mit der Geschwindigkeit des zuletzt beauftragten Kommandos.
- BlendingPrevious: Das Kommando wird ausgeführt, sobald die Achse kein anderes Kommando mehr ausführt. Im Gegensatz zu Buffered stoppt die Achse an der vorherigen Zielposition nicht, sondern durchläuft diese Position mit der Geschwindigkeit des zuerst beauftragten Kommandos.

Siehe auch: Grafische Darstellung der BufferModes [▶ 138]

Optionale Blending-Position

Das Blending bei den verschiedenen BufferModes wird jeweils an der Zielposition des gerade laufenden Kommandos durchgeführt. Im Falle des Bewegungskommandos MC_MoveVelocity [▶ 82] ist keine Zielposition definiert und in anderen Fällen kann es sinnvoll sein, die Blending-Position zu ändern. Dazu kann über den Eingang „Options“ des Funktionsbausteins (siehe Options-Eingang [▶ 16]) eine Blending-Position festgelegt werden, die dann für das neue Kommando verwendet wird. Die optionale Blending-Position muss vor der Zielposition des vorherigen Kommandos liegen, anderenfalls wird das neue Kommando mit einer Fehlermeldung abgewiesen (0x4296). Wenn die optionale Blending-Position bereits überfahren worden ist, wird das neue Kommando instantan umgesetzt. Das Kommando verhält sich also wie ein Aborting-Kommando.

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 offen bleiben kann. Nur wenn in der Dokumentation ausdrücklich auf bestimmte Optionen hingewiesen wird, muss die Options-Datenstruktur vom Anwender belegt werden.

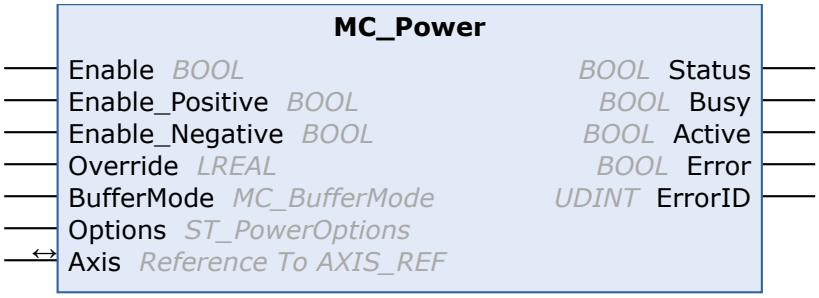
Slaveachsen

Bewegungskommandos können auf gekoppelte Slaveachsen angewendet werden, wenn diese Option in den Parametern der Achse explizit aktiviert worden ist. Ein Bewegungskommando wie MC_MoveAbsolute [▶ 67] führt dann automatisch zum Abkoppeln der Achse und das Kommando wird anschließend ausgeführt. In diesem Fall ist ausschließlich der BufferMode „Aborting“ möglich.

5 Organisationsbausteine

5.1 Achsfunktionen

5.1.1 MC_Power



Mit dem Funktionsbaustein wird die Software-Freigabe einer Achse geschaltet. Die Freigabe kann für beide oder nur für eine bestimmte Fahrtrichtung zugeschaltet werden. Am Ausgang „Status“ wird die Betriebsbereitschaft der Achse signalisiert.

Ein Geschwindigkeits-Override beeinflusst prozentual die Geschwindigkeit aller Fahrkommandos.

Abhängig vom Antriebstypen signalisiert „Status“ auch die Betriebsbereitschaft des Antriebs. Insbesondere digitale Antriebe melden die Betriebsbereitschaft zurück, wogegen analog angeschlossene Antriebe ihre Betriebsbereitschaft nicht zurückmelden können. Im letzten Fall signalisiert Status nur die steuerungsseitige Betriebsbereitschaft.

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; (* B *)
    Enable_Positive : BOOL; (* E *)
    Enable_Negative : BOOL; (* E *)
    Override        : LREAL (* V *) := 100.0; (* in percent - Beckhoff proprietary input *)
    BufferMode      : MC_BufferMode; (* V *)
    Options         : ST_PowerOptions;
END_VAR
```

Name	Typ	Beschreibung
Enable	BOOL	Allgemeine Software-Freigabe für die Achse.
Enable_Positive	BOOL	Vorschubfreigabe in positive Richtung. Wirkt nur, wenn Enable = TRUE ist.
Enable_Negative	BOOL	Vorschubfreigabe in negative Richtung. Wirkt nur, wenn Enable = TRUE ist.
Override	LREAL	Geschwindigkeits-Override in % für alle Fahrbefehle. (0 ≤ Override ≤ 100.0)
BufferMode	MC_BufferMode [▶ 138]	Wird ausgewertet, wenn „Enable“ zurückgesetzt wird. Der Mode „MC_Aborting“ bewirkt ein sofortiges Abschalten der Achsfreigabe. Anderenfalls, z. B. im Mode „MC_Buffered“, wartet der Baustein, bis die Achse keinen Auftrag mehr ausführt.
Options	ST_PowerOptions	Nicht implementiert.

Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 14]

Ein-/Ausgänge

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

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	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; (* B *)
    Busy    : BOOL; (* V *)
    Active  : BOOL; (* V *)
    Error   : BOOL; (* B *)
    ErrorID : UDINT; (* E *)
END_VAR
```

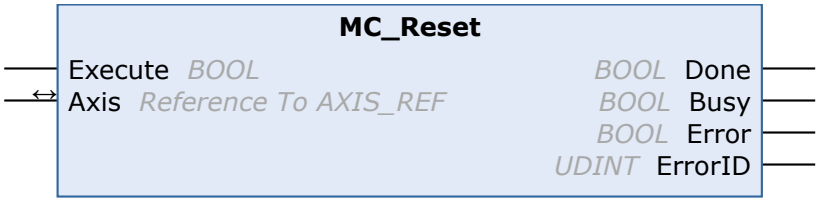
Name	Typ	Beschreibung
Status	BOOL	TRUE, wenn die Achse betriebsbereit ist.
Busy	BOOL	TRUE, solange der Funktionsbaustein mit Enable = TRUE aufgerufen wird.
Active	BOOL	Zeigt an, dass das Kommando ausgeführt wird.
Error	BOOL	TRUE, wenn ein Fehler auftritt.
ErrorID	UDINT	Liefert bei einem gesetzten Error-Ausgang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

Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 14]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

5.1.2 MC_Reset



Mit dem Funktionsbaustein wird ein Reset der NC-Achse durchgeführt. 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 [▶ 124]	Achsdatenstruktur, welche eine Achse eindeutig im System adressiert. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.

Ausgänge

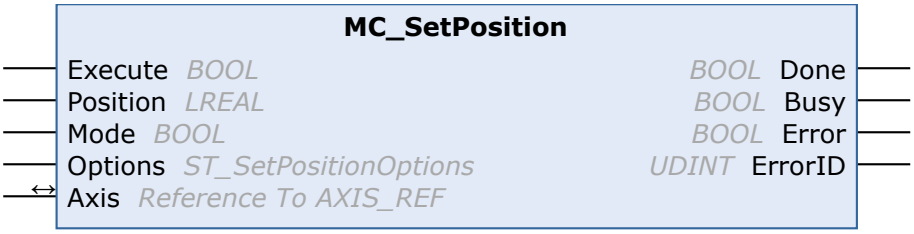
```
VAR_OUTPUT
    Done    : BOOL;
    Busy    : BOOL;
    Error   : BOOL;
    ErrorID : UDINT;
END_VAR
```

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	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

5.1.3 MC_SetPosition



Mit dem Funktionsbaustein wird die aktuelle Achsposition auf einen parametrierbaren Wert gesetzt.

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.

Ab TwinCAT 3.1.4024.51 und Tc2_MC2 3.3.56 ist es über den Schalter `Options.ClearPositionOffset` möglich, den Positionsoffset zu löschen und die Position wieder auf das ursprünglich referenzierte Koordinatensystem zurückzusetzen. Zudem ist es ab der Version möglich, den Offset über ADS auszulesen und in die Variable UserData der AXIS_REF-Struktur einzublenden, sodass der Wert zyklusaktuell vorliegt.

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 [▶ 156]	Datenstruktur, die zusätzliche selten benötigte Parameter enthält. Im Normalfall kann der Eingang offen bleiben. • **ClearPositionLag:** Kann optional gesetzt werden, wenn Soll- und Istposition auf den gleichen Wert gesetzt werden sollen. Damit wird der Schleppfehler gelöscht. • **SelectEncoderIndex:** Kann optional gesetzt werden, wenn eine Achse mit mehreren Encodern verwendet wird und die Position eines bestimmten Encoders (Options.EncoderIndex) gesetzt werden soll. • **EncoderIndex:** Gibt den Encoder (0..n) an, wenn SelectEncoderIndex = TRUE ist. • **ClearPositionOffset:** Kann gesetzt werden, um die mit MC_SetPosition gesetzten Positionsoffsets zu löschen und so die Position wieder zurück auf das ursprünglich referenzierte Koordinatensystem zu setzen; verfügbar ab TwinCAT 3.1.4024.51 und Tc2_MC2 3.3.56.

Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 14]

Ein-/Ausgänge

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

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	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.

Name	Typ	Beschreibung
Error	BOOL	TRUE, wenn ein Fehler auftritt.
ErrorID	UDINT	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 14]

Auslesen des MC_SetPosition-Offsets

Ab TwinCAT 3.1.4024.51 kann das durch MC_SetPosition eingestellte Positionsoffset über ADS ausgelesen werden:

Funktion	ADS-Read	
Port	501 (dez)	
Index Group	0x4100 + Achs-ID	0x5100 + Achs-ID
Index Offset	0x00n10017	0x0017
Daten	REAL64	

Einblenden des MC_SetPosition-Offsets in das AXIS_REF

Die AXIS_REF-Achsstruktur enthält eine Variable UserData, die so konfiguriert werden kann, dass hier ab der TwinCAT Version 3.1.4024.51 der aktuelle Wert des MC_SetPosition-Offsets eingeblendet werden kann. Der Wert liegt dann zyklusaktuell vor.

Die Variable UserData ist hierfür über ADS wie folgt zu konfigurieren:

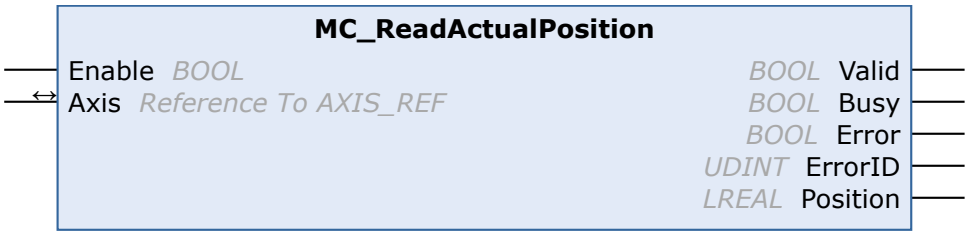
Funktion	ADS-Write
Port	501 (dez)
Index Group	0x4000 + Achs-ID
Index Offset	0x010D
Daten	0x00000000: Deaktiviert (Default) 0x00010017: Aktiviert den Wert MC_SetPosition-Offsets

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

5.2 Status und Parameter

5.2.1 MC_ReadActualPosition



Mit dem Funktionsbaustein kann der Positions-Istwert der Achse gelesen werden. Der Positions-Istwert ist der Positionswert, der vom Antriebsverstärker an die Achse zurückgeht.

Eingänge

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

Name	Typ	Beschreibung
Enable	BOOL	Liest den Parameter in Abhängigkeit vom ReadMode einmalig oder zyklisch.

Ein-/Ausgänge

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

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	Achsdatenstruktur, welche eine Achse eindeutig im System adressiert. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.

Ausgänge

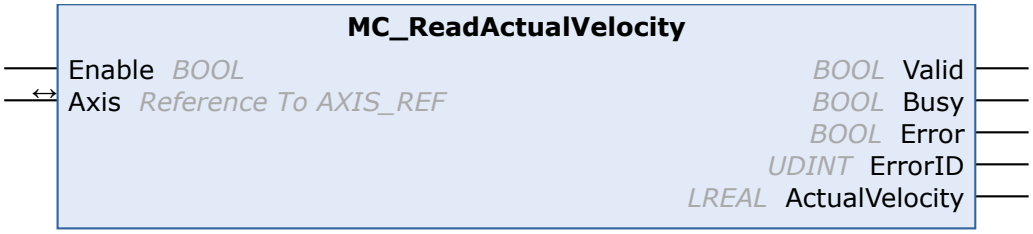
```
VAR_OUTPUT
    Valid    : BOOL;
    Busy     : BOOL;
    Error    : BOOL;
    ErrorID  : UDINT;
    Position : LREAL;
END_VAR
```

Name	Typ	Beschreibung
Valid	BOOL	TRUE, wenn der Ausgang „Position“ einen gültigen Wert hat.
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.
Position	LREAL	Augenblickliche Position der Achse

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

5.2.2 MC_ReadActualVelocity



Mit dem Funktionsbaustein kann der Geschwindigkeits-Istwert der Achse gelesen werden. Der Geschwindigkeits-Istwert ist der Geschwindigkeitswert, der vom Antriebsverstärker an die Achse zurückgeht.

Eingänge

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

Name	Typ	Beschreibung
Enable	BOOL	Liest den Parameter in Abhängigkeit vom ReadMode einmalig oder zyklisch.

Ein-/Ausgänge

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

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	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
    Valid          : BOOL;
    Busy           : BOOL;
    Error          : BOOL;
    ErrorID        : UDINT;
    ActualVelocity : LREAL;
END_VAR
```

Name	Typ	Beschreibung
Valid	BOOL	Signalisiert mit TRUE, dass der am Ausgang Value gelesene Wert gültig ist.
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.
ActualVelocity	LREAL	Augenblickliche Geschwindigkeit der Achse

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

5.2.3 MC_ReadAxisComponents

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

Mit „Achse" sind in diesem Fall die TwinCAT-NC-Achse und deren Parameter und nicht der Antrieb gemeint.

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
    AxisComponents : ST_AxisComponents;
    Axis           : AXIS_REF;
END_VAR
```

Name	Typ	Beschreibung
AxisComponents	ST_AxisComponents [▶ 152]	Datenstrukur, mit der die Komponenten (Encoders, Kontrollers und Antriebe) der Achse zurückgegeben werden.
Axis	AXIS_REF [▶ 124]	Achsdatenstruktur, welche eine Achse eindeutig im System adressiert. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.

Ausgänge

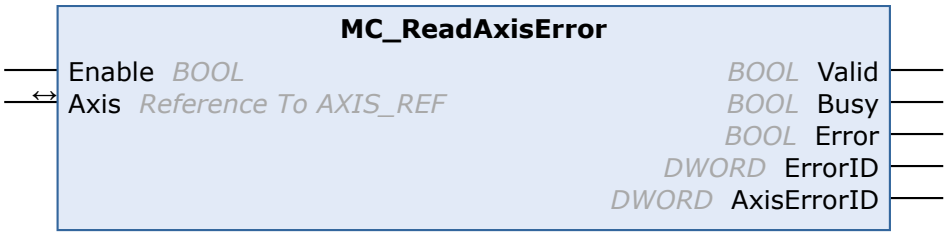
```
VAR_OUTPUT
    Done    : BOOL;
    Busy    : BOOL;
    Error   : BOOL;
    ErrorID : UDINT;
END_VAR
```

Name	Typ	Beschreibung
Done	BOOL	TRUE, wenn die Komponenten erfolgreich gelesen wurden.
Busy	BOOL	TRUE, sobald der Eingang Execute des Funktionsbausteins auf TRUE gesetzt wurde und das Lesen der Komponenten noch nicht beendet wurde. Anschließend ist entweder der Ausgang Done oder Error TRUE und Busy wieder auf FALSE.
Error	BOOL	TRUE, wenn ein Fehler auftritt.
ErrorID	UDINT	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

5.2.4 MC_ReadAxisError



Mit dem Funktionsbaustein wird der Achsfehler einer Achse ausgelesen.

Eingänge

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

Name	Typ	Beschreibung
Enable	BOOL	Wenn Enable = TRUE ist, wird der Achsfehler am Ausgang „AxisErrorID“ ausgegeben.

Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 14]

Ein-/Ausgänge

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

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	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
    Valid       : BOOL; (* B *)
    Busy        : BOOL; (* E *)
    Error       : BOOL; (* B *)
    ErrorID     : UDINT; (* B *)
    AxisErrorID : DWORD; (* B *)
END_VAR
```

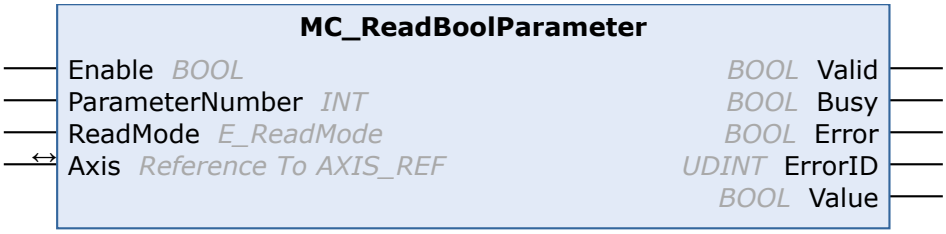
Name	Typ	Beschreibung
Valid	BOOL	TRUE, wenn der am Ausgang „AxisErrorID“ signalisierte Fehler gültig ist.
Busy	BOOL	TRUE, sobald das Kommando mit Enable gestartet wird und solange der Befehl abgearbeitet wird. Wenn Busy FALSE ist, ist der Funktionsbaustein bereit für einen neuen Auftrag.
Error	BOOL	TRUE, wenn ein Fehler auftritt.
ErrorID	UDINT	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.
AxisErrorID	DWORD	Fehlernummer der Achse

Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 14]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

5.2.5 MC_ReadBoolParameter



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

Mit „Achse" sind in diesem Fall die TwinCAT-NC-Achse und deren Parameter und nicht der Antrieb gemeint.

Eingänge

```
VAR_INPUT
    Enable          : BOOL;       (* B *)
    ParameterNumber : MC_AxisParameter;       (* B *)
    ReadMode        : E_ReadMode (* V *)
END_VAR
```

Name	Typ	Beschreibung
Enable	BOOL	Liest den Parameter in Abhängigkeit vom ReadMode einmalig oder zyklisch.
ParameterNumber	MC_AxisParameter [▶ 149]	Nummer des zu lesenden Parameters
ReadMode	E_ReadMode [▶ 148]	Lesemodus des zu lesenden Parameters (einmalig oder zyklisch)

Ein-/Ausgänge

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

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	Achsdatenstruktur, welche eine Achse eindeutig im System adressiert. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.

Ausgänge

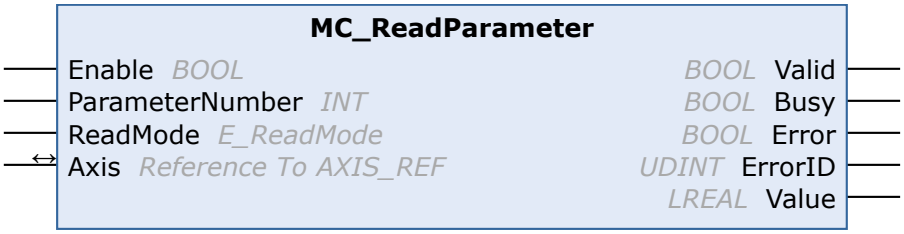
```
VAR_OUTPUT
    Valid   : BOOL;  (* B *)
    Busy    : BOOL;  (* E *)
    Error   : BOOL;  (* B *)
    ErrorID : UDINT; (* E *)
    Value   : BOOL;  (* B *)
END_VAR
```

Name	Typ	Beschreibung
Valid	BOOL	Signalisiert mit TRUE, dass der am Ausgang Value gelesene Wert gültig ist.
Busy	BOOL	TRUE, sobald das Kommando mit Enable gestartet wird und solange der Befehl abgearbeitet wird. Wenn Busy FALSE ist, ist der Funktionsbaustein bereit für einen neuen Auftrag.
Error	BOOL	TRUE, wenn ein Fehler auftritt.
ErrorID	UDINT	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.
Value	BOOL	Zeigt den gelesenen booleschen Wert an.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

5.2.6 MC_ReadParameter



Mit dem Funktionsbaustein MC_ReadParameter wird ein Parameter der Achse gelesen.

Mit „Achse" sind in diesem Fall die TwinCAT-NC-Achse und deren Parameter und nicht der Antrieb gemeint.

Eingänge

```
VAR_INPUT
    Enable          : BOOL;       (* B *)
    ParameterNumber : MC_AxisParameter;       (* B *)
    ReadMode        : E_ReadMode (* V *)
END_VAR
```

Name	Typ	Beschreibung
Enable	BOOL	Liest den Parameter in Abhängigkeit vom ReadMode einmalig oder zyklisch.
ParameterNumber	MC_AxisParameter [▶ 149]	Nummer des zu lesenden Parameters
ReadMode	E_ReadMode [▶ 148]	Lesemodus des zu lesenden Parameters (einmalig oder zyklisch)

Ein-/Ausgänge

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

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	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
    Valid   : BOOL;  (* B *)
    Busy    : BOOL;  (* E *)
    Error   : BOOL;  (* B *)
    ErrorID : DWORD; (* E *)
    Value   : LREAL; (* B *)
END_VAR
```

Name	Typ	Beschreibung
Valid	BOOL	Signalisiert mit TRUE, dass der am Ausgang Value gelesene Wert gültig ist.
Busy	BOOL	TRUE, sobald das Kommando mit Enable gestartet wird und solange der Befehl abgearbeitet wird. Wenn Busy FALSE ist, ist der Funktionsbaustein bereit für einen neuen Auftrag.
Error	BOOL	TRUE, wenn ein Fehler auftritt.

Name	Typ	Beschreibung
ErrorID	DWORD	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.
Value	LREAL	Zeigt den gelesenen Wert an.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

5.2.7 MC_ReadParameterSet

Mit dem Funktionsbaustein MC_ReadParameterSet kann der gesamte Parametersatz einer Achse gelesen werden.

Mit „Achse" sind in diesem Fall die TwinCAT-NC-Achse und deren Parameter und nicht der Antrieb gemeint.

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
    Parameter : ST_AxisParameterSet;
    Axis : AXIS_REF;
END_VAR
```

Name	Typ	Beschreibung
Parameter	ST_AxisParameterSet [▶ 152]	Parameter-Datenstruktur, in die die Parameter eingelesen werden.
Axis	AXIS_REF [▶ 124]	Achsdatenstruktur, welche eine Achse eindeutig im System adressiert. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.

Ausgänge

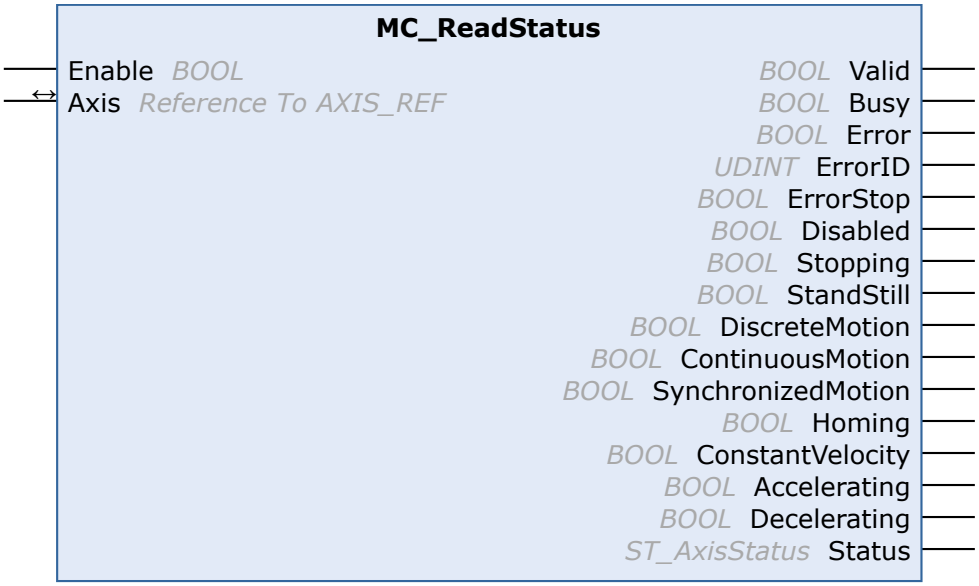
```
VAR_OUTPUT
    Done    : BOOL;
    Busy    : BOOL;
    Error   : BOOL;
    ErrorID : UDINT;
END_VAR
```

Name	Typ	Beschreibung
Done	BOOL	TRUE, wenn die Parameter erfolgreich gelesen wurden.
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	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

5.2.8 MC_ReadStatus



Mit dem Funktionsbaustein MC_ReadStatus wird der aktuelle Betriebszustand einer Achse ermittelt und signalisiert ihn an den Ausgängen des Bausteins.

Der aktualisierte Betriebszustand wird zusätzlich in der Ausgangsdatenstruktur Status und in der Achsdatenstruktur Axis.Status abgelegt. Dadurch ist es ausreichend, den Betriebszustand einmalig am Anfang jedes SPS-Zyklus zu lesen und im weiteren Programmverlauf auf Axis.Status zuzugreifen.

Die Axis-Variable (Typ: AXIS_REF [▶ 124]) enthält bereits eine Instanz des Funktionsbausteins MC_ReadStatus. Dadurch ist es einfach möglich, den Betriebszustand einer Achse am Anfang eines SPS-Zyklus durch den Aufruf von Axis.ReadStatus zu aktualisieren.

Beispiel:

```
PROGRAM MAIN
VAR
    Axis1 : AXIS_REF
END_VAR

(* call the read status function *)
Axis1.ReadStatus;
```

Eingänge

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

Name	Typ	Beschreibung
Enable	BOOL	Wenn Enable = TRUE ist, wird der Betriebszustand der Achse mit jedem Aufruf des Bausteins aktualisiert.

Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 14]

Ein-/Ausgänge

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

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	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
    Valid              : BOOL;
    Busy               : BOOL;
    Error              : BOOL;
    ErrorId            : UDINT;
    (* motion control statemachine states: *)
    ErrorStop          : BOOL;
    Disabled           : BOOL;
    Stopping           : BOOL;
    StandStill         : BOOL;
    DiscreteMotion     : BOOL;
    ContinuousMotion    : BOOL;
    SynchronizedMotion : BOOL;
    Homing             : BOOL;
    (* additional status *)
    ConstantVelocity   : BOOL;
    Accelerating       : BOOL;
    Decelerating       : BOOL;
    (* status data structure *)
    Status             : ST_AxisStatus;
END_VAR
```

Name	Typ	Beschreibung
Valid	BOOL	Zeigt an, dass der an den weiteren Ausgängen angezeigte Betriebszustand der Achse gültig ist.
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.
ErrorStop	BOOL	Zustand der Achse gemäß des PlcOpen-Zustandsdiagramms [▶ 11]
Disabled	BOOL	Zustand der Achse gemäß des PlcOpen-Zustandsdiagramms [▶ 11]
Stopping	BOOL	Zustand der Achse gemäß des PlcOpen-Zustandsdiagramms [▶ 11]
StandStill	BOOL	Zustand der Achse gemäß des PlcOpen-Zustandsdiagramms [▶ 11]
DiscreteMotion	BOOL	Zustand der Achse gemäß des PlcOpen-Zustandsdiagramms [▶ 11]
ContinuousMotion	BOOL	Zustand der Achse gemäß des PlcOpen-Zustandsdiagramms [▶ 11]

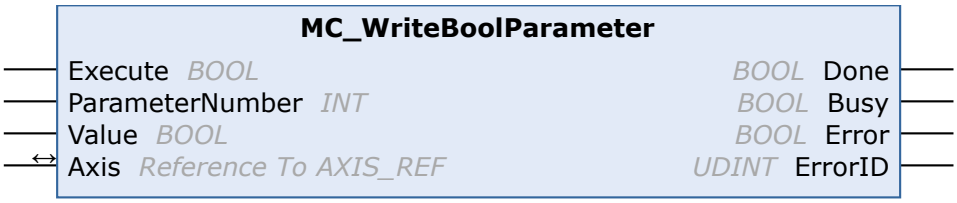
Name	Typ	Beschreibung
SynchronizedMotion	BOOL	Zustand der Achse gemäß des PlcOpen-Zustandsdiagramms [▶ 11]
Homing	BOOL	Zustand der Achse gemäß des PlcOpen-Zustandsdiagramms [▶ 11]
ConstantVelocity	BOOL	Die Achse fährt mit konstanter Geschwindigkeit.
Accelerating	BOOL	Die Achse beschleunigt.
Decelerating	BOOL	Die Achse bremst ab.
Status	ST_AxisStatus	Erweiterte Status-Datenstruktur [▶ 154] mit weiteren Status-Informationen. (Typ: ST_AxisStatus [▶ 154])

Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 14]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

5.2.9 MC_WriteBoolParameter



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

Mit „Achse" sind in diesem Fall die TwinCAT-NC-Achse und deren Parameter und nicht der Antrieb gemeint.

Eingänge

```
VAR_INPUT
    Execute         : BOOL;
    ParameterNumber : MC_AxisParameter;
    Value           : BOOL;
END_VAR
```

Name	Typ	Beschreibung
Execute	BOOL	Mit einer steigenden Flanke wird das Kommando ausgeführt.
ParameterNumber	MC_AxisParameter [▶ 149]	Nummer des zu schreibenden Parameters.
Value	BOOL	Boolesche Wert, der geschrieben wird.

Ein-/Ausgänge

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

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	Achsdatenstruktur, welche eine Achse eindeutig im System adressiert. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.

Ausgänge

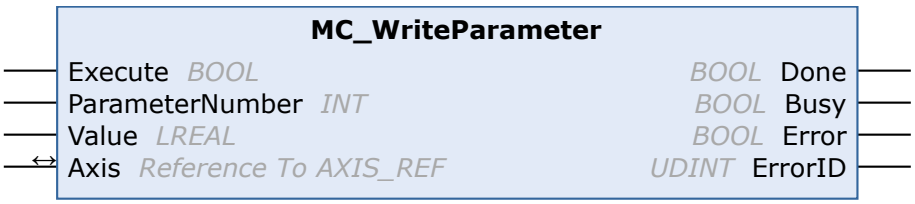
```
VAR_OUTPUT
    Done    : BOOL;
    Busy    : BOOL;
    Error   : BOOL;
    ErrorID : UDINT;
END_VAR
```

Name	Typ	Beschreibung
Done	BOOL	TRUE, wenn die Parameter erfolgreich geschrieben wurden.
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.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

5.2.10 MC_WriteParameter



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

Mit „Achse" sind in diesem Fall die TwinCAT-NC-Achse und deren Parameter und nicht der Antrieb gemeint.

Eingänge

```
VAR_INPUT
    Execute         : BOOL;
    ParameterNumber : MC_AxisParameter;
    Value           : LREAL;
END_VAR
```

Name	Typ	Beschreibung
Execute	BOOL	Mit einer steigenden Flanke wird das Kommando ausgeführt.
ParameterNumber	MC_AxisParameter [▶ 149]	Nummer des zu schreibenden Parameters.

Name	Typ	Beschreibung
Value	LREAL	LREAL-Wert, der geschrieben wird.

Ein-/Ausgänge

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

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	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 Parameter erfolgreich geschrieben wurden.
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.

Beispiel

```
VAR
    Axis1           : Axis_REF;
    fbWriteParameter: MC_WriteParameter;
END_VAR
```

```
fbWriteParameter(
    Execute := TRUE;
    Axis:= Axis1 ,
    ParameterNumber:= MC_AxisParameter.SwLimitPos,
    Value:= 2000
);
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

5.2.11 MC_WriteBoolParameterPersistent

Mit dem Funktionsbaustein MC_WriteBoolParameterPersistent können boolesche Parameter für die Achse persistent geschrieben werden.

Der zu schreibende persistente Parameter wird in einer Initialisierungsliste gespeichert. Bei Systemstart startet das System zunächst mit den ursprünglich konfigurierten Werten und überschreibt diese noch vor dem Anlaufen der Tasks durch die persistenten Daten aus der Initialisierungsliste. Die Initialisierungsliste wird beim Registrieren einer neuen Systemkonfiguration gelöscht. Das System startet dann mit den unveränderten Daten aus der neuen Konfiguration.

Mit „Achse" sind in diesem Fall die TwinCAT-NC-Achse und deren Parameter und nicht der Antrieb gemeint.

Eingänge

```
VAR_INPUT
    Execute         : BOOL;
    ParameterNumber : MC_AxisParameter;
    Value           : BOOL;
END_VAR
```

Name	Typ	Beschreibung
Execute	BOOL	Mit einer steigenden Flanke wird das Kommando ausgeführt.
ParameterNumber	MC_AxisParameter [▶ 149]	Nummer des zu schreibenden Parameters.
Value	BOOL	Boolesche Wert, der geschrieben wird.

Ein-/Ausgänge

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

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	Achsdatenstruktur, welche eine Achse eindeutig im System adressiert. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.

Ausgänge

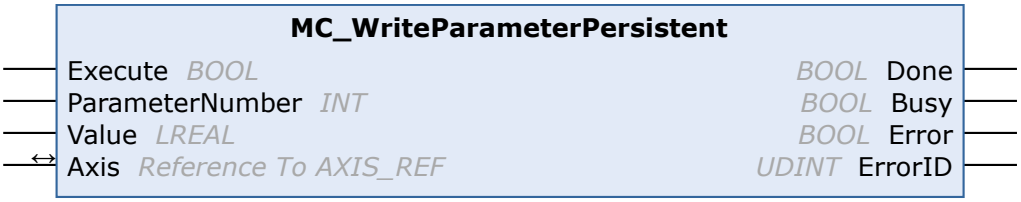
```
VAR_OUTPUT
    Done    : BOOL;
    Busy    : BOOL;
    Error   : BOOL;
    ErrorID : UDINT;
END_VAR
```

Name	Typ	Beschreibung
Done	BOOL	TRUE, wenn die Parameter erfolgreich geschrieben wurden.
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.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

5.2.12 MC_WriteParameterPersistent



Mit dem Baustein MC_WriteParameterPersistent können Parameter für die Achse persistent geschrieben werden.

Der zu schreibende persistente Parameter wird in einer Initialisierungsliste gespeichert. Bei Systemstart startet das System zunächst mit den ursprünglich konfigurierten Werten und überschreibt diese noch vor dem Anlaufen der Tasks durch die persistenten Daten aus der Initialisierungsliste. Die Initialisierungsliste wird beim Registrieren einer neuen Systemkonfiguration gelöscht. Das System startet dann mit den unveränderten Daten aus der neuen Konfiguration.

Mit „Achse" sind in diesem Fall die TwinCAT-NC-Achse und deren Parameter und nicht der Antrieb gemeint.

Eingänge

```
VAR_INPUT
    Execute         : BOOL;
    ParameterNumber : MC_AxisParameter;
    Value           : LREAL;
END_VAR
```

Name	Typ	Beschreibung
Execute	BOOL	Mit einer steigenden Flanke wird das Kommando ausgeführt.
ParameterNumber	MC_AxisParameter [▶ 149]	Nummer des zu schreibenden Parameters.
Value	LREAL	LREAL-Wert, der geschrieben wird.

Ein-/Ausgänge

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

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	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
```

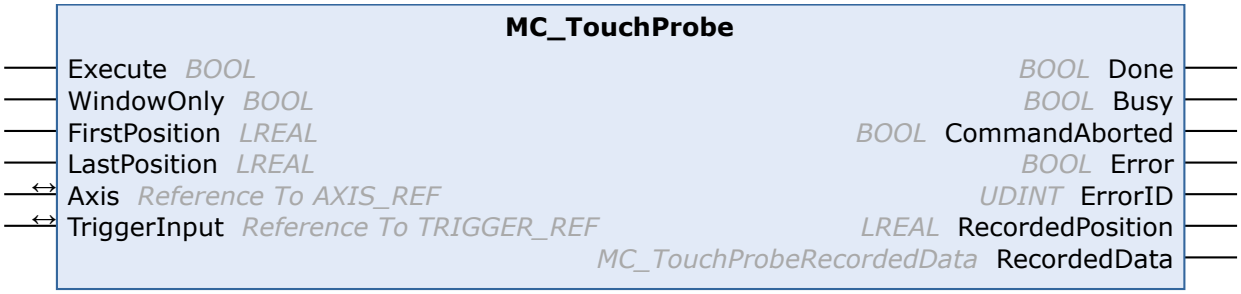
Name	Typ	Beschreibung
Done	BOOL	TRUE, wenn die Parameter erfolgreich geschrieben wurden.
Busy	BOOL	TRUE, sobald das Kommando mit „Execute“ gestartet wird und solange der Befehl abgearbeitet wird. Wenn „Busy“ FALSE ist, ist der Funktionsbaustein bereit für einen neuen Auftrag. Gleichzeitig ist einer der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

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

5.3 Touch Probe

5.3.1 MC_TouchProbe



Mit dem Funktionsbaustein MC_TouchProbe wird eine Achsposition zum Zeitpunkt eines digitalen Signals (Messtasterfunktion) erfasst. Die Position wird üblicherweise nicht direkt in der SPS-Umgebung, sondern 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.

Der Funktionsbaustein wurde gegenüber TwinCAT 2 erweitert und entspricht der Funktion des bisherigen Bausteins MC_TouchProbe_V2.

Voraussetzungen

Voraussetzung für die Positionserfassung ist eine geeignete Encoder-Hardware, die in der Lage ist, die erfasste Position zu latchen. Unterstützt werden:

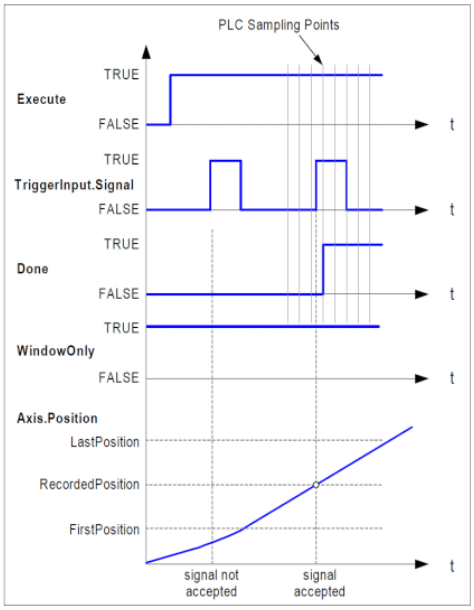
- SERCOS Antriebe
  Der Antrieb muss abweichend von MC_TouchProbe mit einer erweiterten Schnittstelle konfiguriert werden, bei der die Parameter S-0-0405 bzw. S-0-0406 im Prozessabbild enthalten sind.
- EtherCAT SoE Antriebe (z. B. AX5000)
  Der Antrieb muss abweichend von MC_TouchProbe mit einer erweiterten Schnittstelle konfiguriert werden, bei der die Parameter S-0-0405 bzw. S-0-0406 im Prozessabbild enthalten sind.
- EtherCAT CoE Antriebe
  Der Antrieb muss mit dem Parameter 0x60B9 (Touch Probe Status) im Prozessabbild konfiguriert werden.
- EL5101, KL5101
  Latchen der C-Spur und des digitalen Eingangs ist möglich. Mit dieser Hardware kann zeitgleich nur ein Signal bzw. eine Flanke erfasst werden. Der Continuous-Mode wird nicht unterstützt.

Das 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. Beckhoff-EtherCAT-Antriebe können mit dem System Manager konfiguriert werden. Hierbei muss beachtet werden, dass die Probe-Unit mit dem erweiterten Interface „Extended NC Probe Unit“ konfiguriert werden muss.

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 [▶ 40] mit derselben TriggerInput [▶ 162]-Datenstruktur aufgerufen werden. Anderenfalls kann kein neuer Zyklus gestartet werden.

Signalverlauf



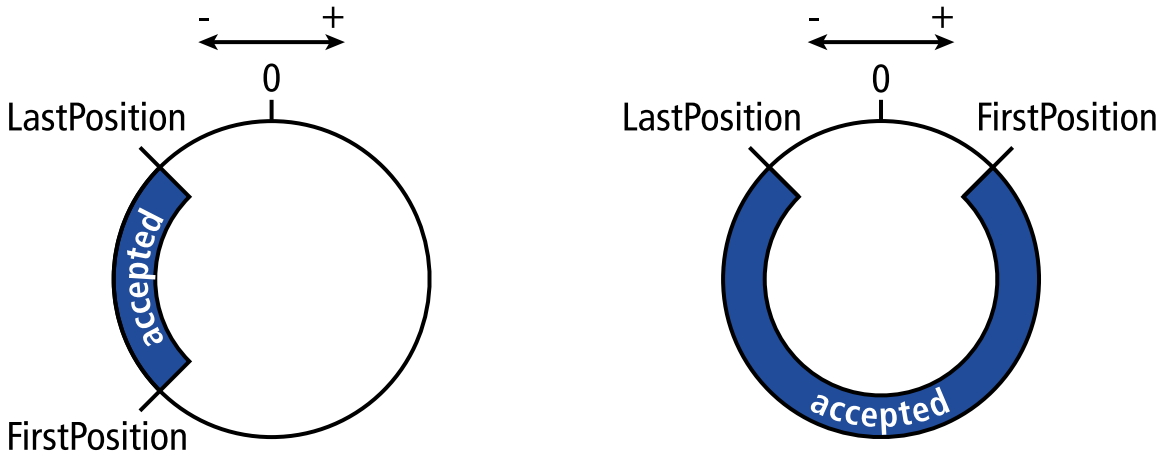
Eingänge

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

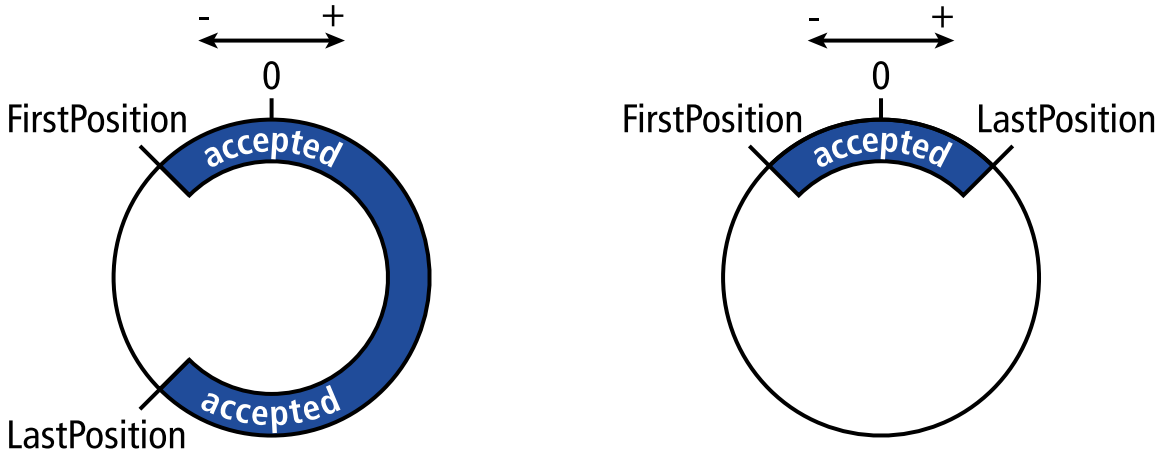
Name	Typ	Beschreibung
Execute	BOOL	Mit einer steigenden Flanke wird das Kommando ausgeführt.
WindowOnly	BOOL	Wenn WindowOnly = TRUE ist, wird nur eine Position innerhalb des Fensters zwischen „FirstPosition“ und „LastPosition“ erfasst. Positionen außerhalb des Fensters werden verworfen und das externe 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 [▶ 162]-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 Grad).

Name	Typ	Beschreibung
FirstPosition	LREAL	Anfangsposition des Erfassungsfensters, wenn „WindowOnly“ TRUE ist. Diese Position kann absolut oder modulo interpretiert werden. Dazu muss in der TriggerInput [▶ 162]-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 [▶ 162]-Datenstruktur entsprechend gesetzt werden.

A. FirstPosition < LastPosition



B. FirstPosition > LastPosition



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

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

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 [▶ 40], abgebrochen wurde.
Error	BOOL	TRUE, wenn ein Fehler auftritt.
ErrorID	UDINT	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.
RecordedPosition	LREAL	Erfasste Achsposition zum Zeitpunkt des Trigger-Signals
RecordedData	MC_TouchProbeRecordedData [▶ 161]	Datenstruktur mit ergänzenden Informationen zur erfassten Achsposition zum Zeitpunkt des Trigger-Signals

Ein-/Ausgänge

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

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	Achsdatenstruktur
TriggerInput	TRIGGER_REF [▶ 162]	Datenstruktur zur Beschreibung der Trigger-Quelle. Diese Datenstruktur muss vor dem ersten Aufruf des Funktionsbausteins parametriert werden.

5.3.2 MC_AbortTrigger

Mit dem Funktionsbaustein MC_AbortTrigger wird ein durch MC_TouchProbe gestarteter Messtasterzyklus abgebrochen. MC_TouchProbe startet einen Messtasterzyklus, indem ein Positionslatch in einer externen Encoder- oder 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 [▶ 124]	Achsdatenstruktur
TriggerInput	TRIGGER_REF [▶ 162]	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
```

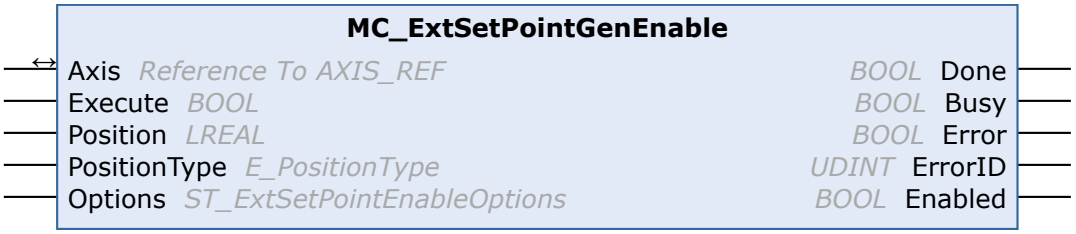
Name	Typ	Beschreibung
Done	BOOL	TRUE, sobald der Messtasterzyklus erfolgreich abgebrochen wurde.
Busy	BOOL	TRUE, sobald der Baustein aktiv ist. FALSE, wenn er sich im Grundzustand befindet.
Error	BOOL	TRUE, wenn ein Fehler auftritt.
ErrorID	UDINT	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

5.4 Externer Sollwertgenerator

5.4.1 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 (Axis.PlcToNc.ExtSetPos, ExtSetVelo, ExtSetAcc und ExtSetDirection).

Ein sogenannter externer Sollwertgenerator ist üblicherweise ein SPS-Funktionsbaustein, der zyklische Sollwerte für eine Achse berechnet und somit den in einer NC-Achse enthaltenen internen Sollwertgenerator ersetzen kann.

Ergänzende Informationen finden Sie unter MC_ExtSetPointGenDisable [▶ 43] und MC_ExtSetPointGenFeed [▶ 44].

Eingänge

```
VAR_INPUT
    Execute      : BOOL;
    Position     : LREAL;
    PositionType : E_PositionType;
    Options      : ST_ExtSetPointEnableOptions;
END_VAR
```

Name	Typ	Beschreibung
Execute	BOOL	Mit einer steigenden Flanke wird das Kommando ausgeführt.
Position	LREAL	Position für Zielpositionsüberwachung. Das Setzen dieser Position bedeutet nicht, dass die Achse zu dieser Position verfährt, dafür ist ausschließlich der externe Sollwertgenerator verantwortlich. Vielmehr wird durch das Setzen dieser Position die Zielpositionsüberwachung aktiviert. Das Flag InTargetPosition wird TRUE, sobald diese Position erreicht wird.
PositionType	E_PositionType [▶ 138]	Positionstyp POSITIONTYPE_ABSOLUTE oder POSITIONTYPE_RELATIVE
Options	ST_ExtSetPointEnableOptions	**UseTorqueOffset:** Muss auf TRUE gesetzt werden, damit bei Verwendung des MC_ExtSetPointGenFeedWithTorque auch der TorqueOffset zyklisch an den Antriebsregler übertragen wird.

Ein-/Ausgänge

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

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	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;
    Enabled : BOOL;
END_VAR
```

Name	Typ	Beschreibung
Done	BOOL	TRUE, wenn der Befehl erfolgreich ausgeführt wurde.
Busy	BOOL	TRUE, sobald der Baustein aktiv ist. FALSE, wenn er sich im Grundzustand befindet.

Name	Typ	Beschreibung
Error	BOOL	TRUE, wenn ein Fehler auftritt.
ErrorID	UDINT	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.
Enabled	BOOL	Zeigt, unabhängig von der Funktionsausführung, den aktuellen Zustand des externen Sollwertgenerators.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

5.4.2 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 (Axis.PlcToNc.ExtSetPos, ExtSetVelo, ExtSetAcc und ExtSetDirection).

Ein sogenannter externer Sollwertgenerator ist üblicherweise ein SPS-Funktionsbaustein, der zyklische Sollwerte für eine Achse berechnet und somit den in einer NC-Achse enthaltenen internen Sollwertgenerator ersetzen kann.

Ergänzende Informationen finden Sie unter MC_ExtSetPointGenEnable [▶ 41] und MC_ExtSetPointGenFeed [▶ 44].

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 [▶ 124]	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;
```

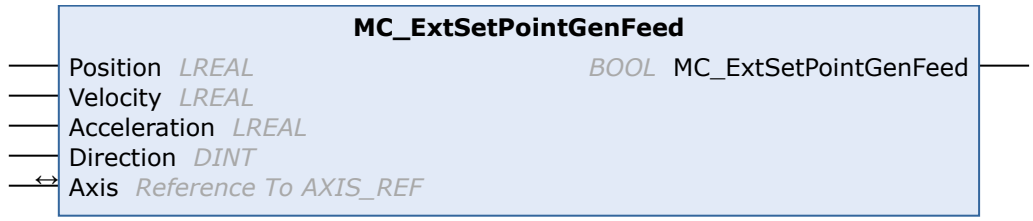
```
    ErrorID : UDINT;
    Enabled : BOOL;
END_VAR
```

Name	Typ	Beschreibung
Done	BOOL	TRUE, wenn der Befehl erfolgreich ausgeführt 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.
Enabled	BOOL	Zeigt, unabhängig von der Funktionsausführung, den aktuellen Zustand des externen Sollwertgenerators.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

5.4.3 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.

Ein sogenannter externer Sollwertgenerator ist üblicherweise ein SPS-Funktionsbaustein, der zyklische Sollwerte für eine Achse berechnet und somit den in einer NC-Achse enthaltenen internen Sollwertgenerator ersetzen kann.

Ergänzende Informationen finden Sie unter MC_ExtSetPointGenEnable [▶ 41] und MC_ExtSetPointGenDisable [▶ 43].

Eingänge

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

Name	Typ	Beschreibung
Position	LREAL	Sollposition aus einem externen Sollwertgenerator
Velocity	LREAL	Sollgeschwindigkeit aus einem externen Sollwertgenerator
Acceleration	LREAL	Sollbeschleunigung aus einem externen Sollwertgenerator
Direction	DINT	Sollrichtung aus einem externen Sollwertgenerator. (-1 = negative Richtung, 0 = Stillstand, 1 = positive Richtung)

Ein-/Ausgänge

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

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	Achsdatenstruktur, welche eine Achse eindeutig im System adressiert. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

5.4.4 MC_ExtSetPointGenFeedWithTorque

Mit der Funktion MC_ExtSetPointGenFeedWithTorque werden die Sollwerte eines externen Sollwertgenerators in eine Achse eingespeist. Diese Funktion ist gegenüber der Funktion MC_ExtSetPointGenFeed um die Übergabe eines TorqueOffset erweitert. Damit auch der TorqueOffset von der NC zyklisch an den Antrieb übergeben wird, muss dieses explizit beim Aufruf des FB_ExtSetPointGenEnable über Options.UseTorqueOffset aktiviert werden. Die Funktion kopiert die Daten instantan in das zyklische Achsinterface (ExtSetPos, ExtSetVelo, ExtSetAcc, ExtSetDirection und ExtTorque) der Achse. Das Funktionsergebnis MC_ExtSetPointGenFeedWithTorque ist ungenutzt und daher immer FALSE.

Ein sogenannter externer Sollwertgenerator ist üblicherweise ein SPS-Funktionsbaustein, der zyklische Sollwerte für eine Achse berechnet und somit den in einer NC-Achse enthaltenen internen Sollwertgenerator ersetzen kann.

Ergänzende Informationen finden Sie unter MC_ExtSetPointGenEnable [▶ 41] und MC_ExtSetPointGenDisable [▶ 43].

Eingänge

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

Name	Typ	Beschreibung
Position	LREAL	Sollposition aus einem externen Sollwertgenerator
Velocity	LREAL	Sollgeschwindigkeit aus einem externen Sollwertgenerator
Acceleration	LREAL	Sollbeschleunigung aus einem externen Sollwertgenerator
TorqueOffset	LREAL	TorqueOffset aus einem externen Sollwertgenerator
Direction	DINT	Sollrichtung aus einem externen Sollwertgenerator. (-1 = negative Richtung, 0 = Stillstand, 1 = positive Richtung)

Ein-/Ausgänge

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

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	Achsdatenstruktur, welche eine Achse eindeutig im System adressiert. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.

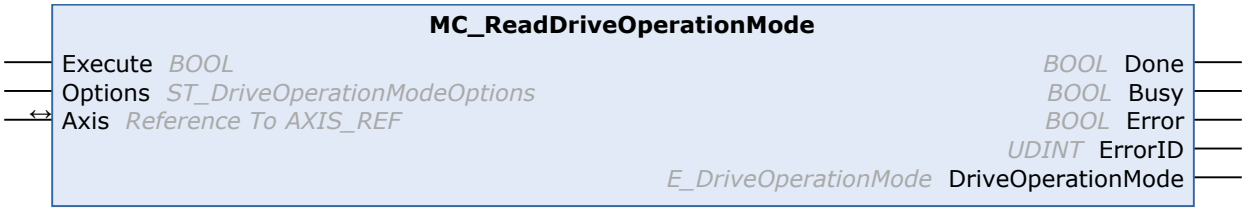
Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4024	PC oder CX (x86 oder x64)	Tc2_MC2 (ab v3.3.68.0)

5.5 Spezielle Erweiterungen

5.5.1 DriveOperationMode

5.5.1.1 MC_ReadDriveOperationMode



Mit dem Funktionsbaustein MC_ReadDriveOperationMode wird die aktuell aktive Betriebsart des mit der NC-Achse verknüpften Antriebsgerätes gelesen.

Eingänge

```
VAR_INPUT
    Execute             : BOOL;
    Options             : ST_DriveOperationModeOptions;
END_VAR
```

Name	Typ	Beschreibung
Execute	BOOL	Mit einer steigenden Flanke wird das Kommando ausgeführt.
Options	ST_DriveOperationModeOptions	Nicht implementiert.

Ein-/Ausgänge

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

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	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;
    DriveOperationMode : E_DriveOperationMode;
END_VAR
```

Name	Typ	Beschreibung
Done	BOOL	Wird TRUE, wenn das Kommando erfolgreich beendet wurde.
Busy	BOOL	Der Busy-Ausgang wird TRUE, sobald das Kommando mit Execute gestartet wird und bleibt TRUE, solange der Befehl abgearbeitet wird. Wenn Busy wieder FALSE wird, so ist der Funk-tionsbaustein bereit für einen neuen Auftrag. Gleichzeitig ist einer der Ausgänge Done, CommandAborted oder Error gesetzt.
Error	BOOL	Wird TRUE, sobald ein Fehler eintritt.
ErrorID	UDINT	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.
DriveOperationMode	E_DriveOperationMode	Aktuell aktive Betriebsart des angeschlossenen Antriebsgerätes (E_DriveOperationMode [▶ 133]).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

5.5.1.2 MC_WriteDriveOperationMode

Mit dem Funktionsbaustein MC_WriteDriveOperationMode wird der Wechsel in die parametrierte Betriebsart für ein mit der NC-Achse verknüpftes Antriebsgerät veranlasst.

Eingänge

```
VAR_INPUT
    Execute            : BOOL;
    DriveOperationMode : E_DriveOperationMode;
    Options            : ST_DriveOperationModeOptions;
END_VAR
```

Name	Typ	Beschreibung
Execute	BOOL	Mit einer steigenden Flanke wird das Kommando ausgeführt.
DriveOperationMode	E_DriveOperationMode [▶ 133]	Zu aktivierende Betriebsart des angeschlossenen Antriebsgerätes.
Options	ST_DriveOperationModeOptions	Nicht implementiert.

Ein-/Ausgänge

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

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	Achsdatenstruktur, welche eine Achse eindeutig im System adressiert. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.

Ausgänge

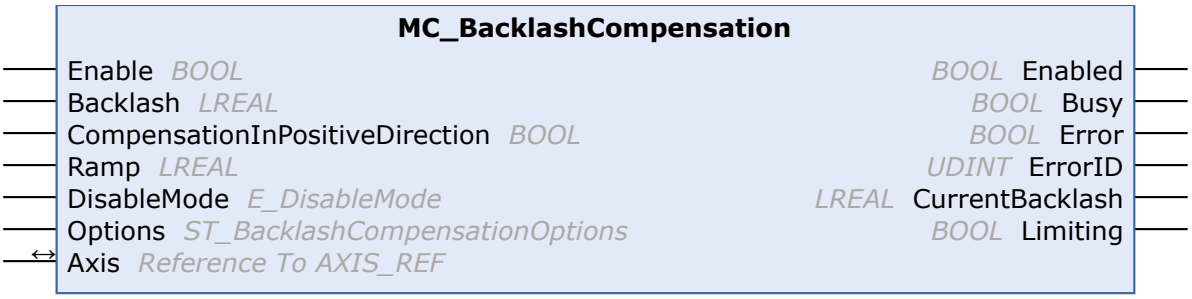
```
VAR_OUTPUT
    Done                : BOOL;
    Busy                : BOOL;
    Error               : BOOL;
    ErrorID             : UDINT;
END_VAR
```

Name	Typ	Beschreibung
Done	BOOL	TRUE, wenn das Kommando fehlerfrei ausgeführt wurde.
Busy	BOOL	Der Busy-Ausgang wird TRUE, sobald das Kommando mit Execute gestartet wird und bleibt TRUE, solange der Befehl abgearbeitet wird. Wenn Busy wieder FALSE wird, so ist der Funk-tionsbaustein bereit für einen neuen Auftrag.
Error	BOOL	Wird TRUE, sobald ein Fehler eintritt.
ErrorID	UDINT	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

5.5.2 MC_BacklashCompensation



Mit dem Funktionsbaustein MC_BacklashCompensation werden zusammen mit dem Achs-Parameter *Position Correction* Lose kompensiert. Es ist möglich, negative sowie positive Lose zu kompensieren.

Verwendungshinweise und nötige Einstellungen in TwinCAT:

- Die hier beschriebene Funktionalität arbeitet mit dem Parameter *Position Correction*. Dieser Parameter muss zwingend im TwinCAT XAE oder per ADS aktiviert werden (siehe auch TwinCAT 3 NC PTP Axes). Die im XAE angebotene Funktion Backlash Compensation ist lediglich aus Kompatibilitätsgründen noch vorhanden. Verwenden Sie diese nicht bei neuen Projekten.
- Wenn der Anwender zusätzlich parallel zum Funktionsbaustein *MC_BacklashCompensation* (deren Implementierung auf der Funktion *Position Correction* basiert) für andere Zwecke ebenfalls die Funktion *Position Correction* verwenden will, dann ist diese „doppelte Nutzung“ des Parameters durch einen Workaround in der SPS lösbar, indem die Summe aus beiden Anforderungen unter Verwendung des Funtkionsbausteins *MC_PositionCorrectionLimiter* eingespeist wird.

Eingänge

```
VAR_INPUT
    Enable                          : BOOL;
    Backlash                        : LREAL;
    CompensationInPositiveDirection : BOOL;
    Ramp                            : LREAL;
    DisableMode                     : E_DisableMode;
    Options                         : ST_BacklashCompensationOptions;
END_VAR
```

Name	Typ	Beschreibung
Enable	BOOL	Das Kommando wird so lange ausgeführt, wie Enable aktiv ist. Die Losekompensation wird hiermit aktiviert.
Backlash	LREAL	Kompensations-Wert mit Vorzeichen [mm]. Wenn der Default-Wert genutzt wird, wird der interne NC Backlash-Wert per ADS gelesen und in diesem Funktionsblock genutzt.
CompensationInPositiveDirection	BOOL	Die Kompensation findet in der ausgewählten Arbeitsrichtung statt. FALSE (Default): Losekompensation muss ausgeführt werden, wenn in negative Richtung gefahren wird. TRUE: Losekompensation muss ausgeführt werden, wenn in positive Richtung gefahren wird.
Ramp	LREAL	Geschwindigkeitslimit für eingegebene Losekompensation (konstante Geschwindigkeit und lineare Position als Unterprofile für Losekompensation [mm/s]).
DisableMode	E_DisableMode	Abschaltmodus: (0)=HOLD, (1)=RESET
Options	ST_BacklashCompensationOptions	Optionale Parameter (nicht implementiert)

Ein-/Ausgänge

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

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	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
    Enabled                 : BOOL;
    Busy                    : BOOL;
    Error                   : BOOL;
    ErrorId                 : UDINT;
    CurrentBacklash         : LREAL;
    Limiting                : BOOL;
END_VAR
```

Name	Typ	Beschreibung
Enabled	BOOL	Dieser Ausgang wird TRUE, wenn die Losekompensation ohne Fehler aktiviert wurde.
Busy	BOOL	Dieser Ausgang wird TRUE, wenn das Kommando mit Enable gestartet wird, und bleibt es dann so lange, wie der Funktionsbaustein das Kommando ausführt.
Error	BOOL	Dieser Ausgang wird TRUE, wenn bei der Ausführung des Kommandos ein Fehler aufgetreten ist.

Name	Typ	Beschreibung
ErrorId	UDINT	Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Kommandos. Der Fehlercode kann in der ADS-Fehlerdokumentation oder in der NC-Fehlerdokumentation (Fehlercodes 0x4nnn und 0x8nnn) nachgeschlagen werden.
CurrentBacklash	LREAL	Aktueller Kompensations-Wert [mm].
Limiting	BOOL	Dieser Ausgang wird TRUE, wenn der Funktionsblock aktuell die Losekompensation limitiert.

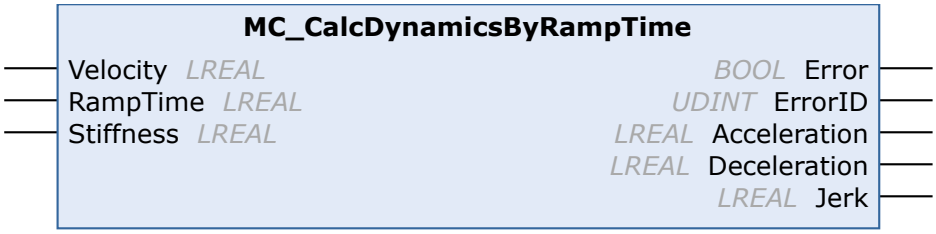
TYPE E_DisableMode:

```
(
    DisableModeHold         :=0, (* hold on the last output value (default) *)
    DisableModeReset        :=1 (* reset the output value to zero (jump to zero) *));
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.0.0	PC or CX (x86 or x64)	Tc2_MC2

5.5.3 MC_CalcDynamicsByRampTime



Mit dem Funktionsbaustein MC_CalcDynamicsByRampTime werden die Dynamikparameter Beschleunigung, Verzögerung und Ruck berechnet, welche notwendig sind, um eine vorgegebene Geschwindigkeit in einer definierten Zeit zu erreichen.

Der Funktionsbaustein geht davon aus, dass die vorgegebene Geschwindigkeit tatsächlich erreicht werden kann. Falls die berechneten Dynamikparameter auf einer kurzen Fahrstrecke verwendet werden, können die erreichte Geschwindigkeit und die dazugehörige Hochlaufzeit kleiner sein. Der Funktionsbaustein berechnet die Beschleunigungsrampe und gibt den identischen Wert als Verzögerung aus.

Die berechneten Dynamikparameter Beschleunigung, Verzögerung und Ruck können mit allen MC_Move…-Bausteinen sowie MC_Halt und MC_Stop verwendet werden.

Eingänge

```
VAR_INPUT
    Velocity      : LREAL;
    RampTime      : LREAL;
    Stiffness     : LREAL;
END_VAR
```

Name	Typ	Beschreibung
Velocity	LREAL	Zu erreichende Geschwindigkeit
RampTime	LREAL	Zeit, die zum Erreichen der Geschwindigkeit benötigt werden soll.
Stiffness	LREAL	Steifheit des Beschleunigungsprofils [0..1]. Ein großer Wert bedeutet einen höheren Ruck des Profils.

Ausgänge

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

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

Name	Typ	Beschreibung
Error	BOOL	TRUE, wenn ein Fehler auftritt.
ErrorID	UDINT	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.
Acceleration	LREAL	Berechnete Beschleunigung
Deceleration	LREAL	Berechnete Verzögerung (Deceleration=Acceleration)
Jerk	LREAL	Berechneter Ruck

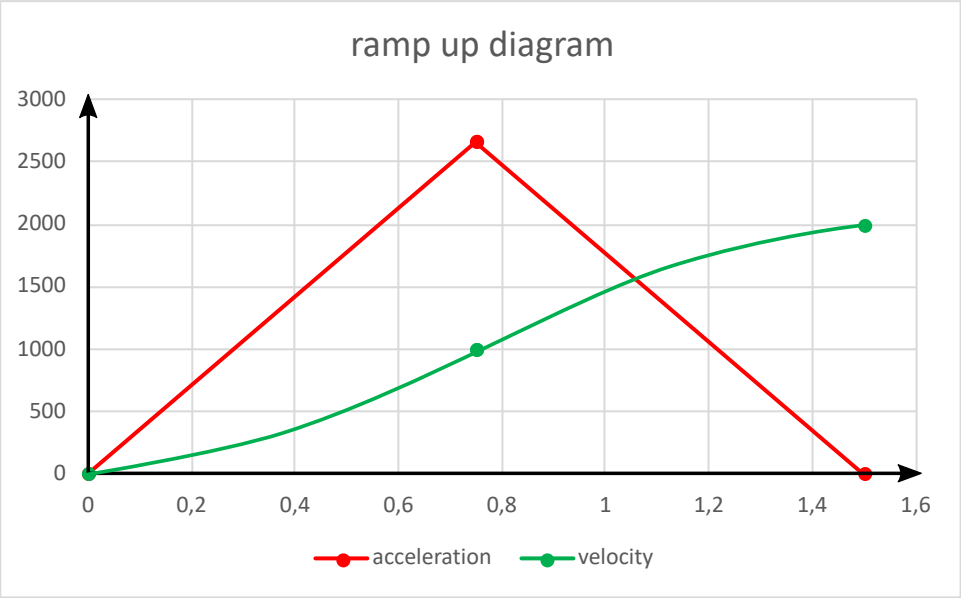
Beispiele

t_1: interne Beschleunigungsrampenzeit/Ruckzeit

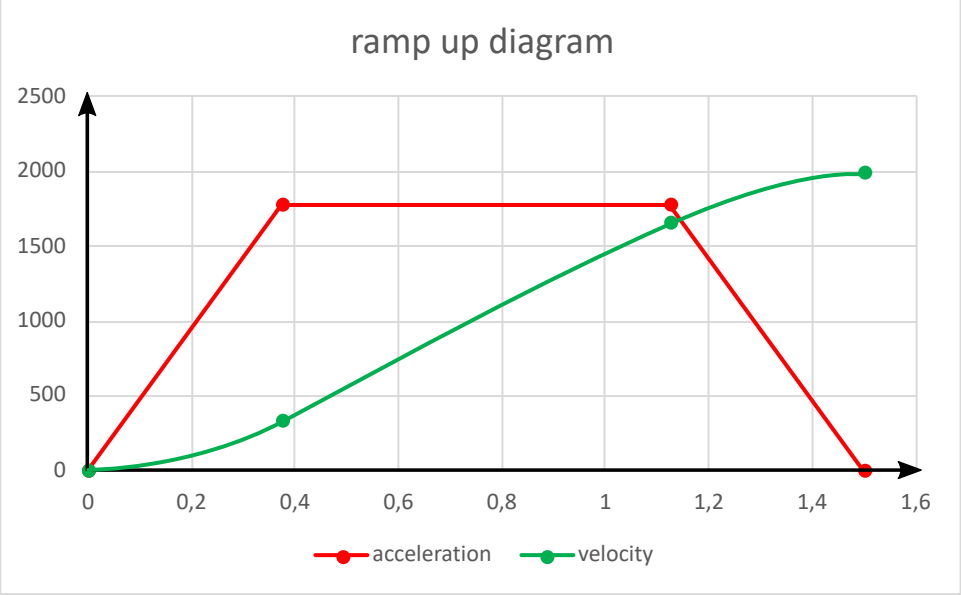
t_2: interne konstante Beschleunigungszeit

t : Beschleunigungsrampenzeit, $t = 2\ t_1 + t_2$

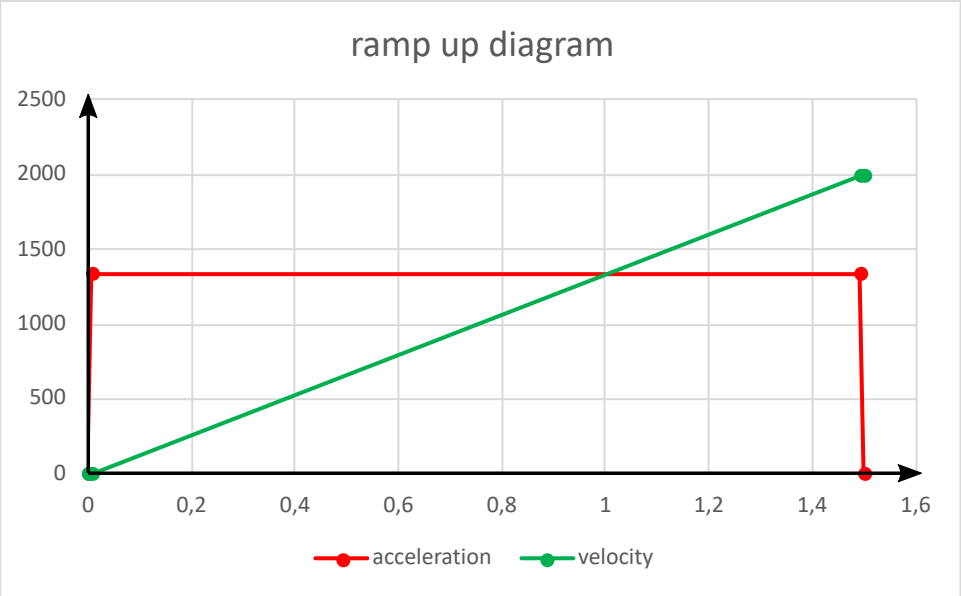
$c_s = 0\%$ → $t_1 = ½\ t$ $t_2 = 0$



$c_s = 50\%$ → $t_1 = t_2 = 1/3\ t$



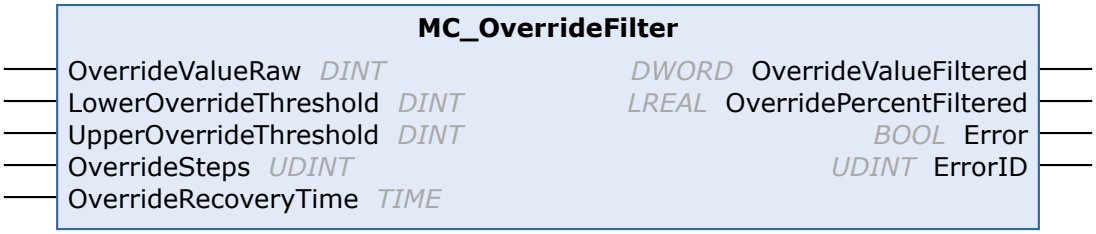
c_s = 99% → t_1 = 0 t_2 = t



Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4024.42	PC or CX (x86 or x64)	Tc2_MC2

5.5.4 MC_OverrideFilter



Mit dem Funktionsbaustein MC_OverrideFilter kann ein ungefilterter Override-Wert in Digits (z. B. ein Spannungswert einer Analogeingangsklemme) in einen gefilterten, für das zyklische Achsinterface (PlcToNc) passenden, Override-Wert (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 „UpperOverrideThreshold“ auf auf einen Gültigkeitsbereich begrenzt und in eine parametrierbare Stufung (Auflösung) umgesetzt („OverrideSteps“). Nach jeder Override-Änderung am Ausgang des FBs wird intern eine Mindestruhezeit abgewartet („OverrideRecoveryTime“), bevor wieder ein neuer Override-Wert übernommen werden kann. Die einzigen Ausnahmen stellen die Override-Werte 0% und 100%, die aus Sicherheitsgründen immer ohne Zeitverzögerung umgesetzt werden.

Aufgrund der Stufung des ausgegebenen Override-Wertes („OverrideValueFiltered“) kann der gefilterte Override bei sehr kleinen Override-Eingangswerten („OverrideValueRaw“) Null werden. Ein Override Null führt zum Stillstand der Achse. Wenn ein vollständiger Stillstand nicht erwünscht ist, sollte „OverrideValueRaw“ nicht unter die kleineste Stufung fallen.

Eingänge

```
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
```

Name	Typ	Beschreibung
OverrideValueRaw	DINT	Roher, ungefilterter Override-Wert (z. B. ein Spannungswert einer Analogeingangsklemme).
LowerOverrideThreshold	DINT	Untere Schranke, die den rohen Override-Wert begrenzt.
UpperOverrideThreshold	DINT	Obere Schranke, die den rohen Override-Wert begrenzt.
OverrideSteps	UDINT	Die vorgegebene Stufung (Overrideauflösung).
OverrideRecoveryTime	TIME	Mindestruhezeit nach der ein neuer gefilterter Override-Wert auf den Ausgang gelegt wird. Die Override-Werte 0% und 100% werden allerdings ohne Zeitverzögerung umgesetzt.

Ausgänge

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

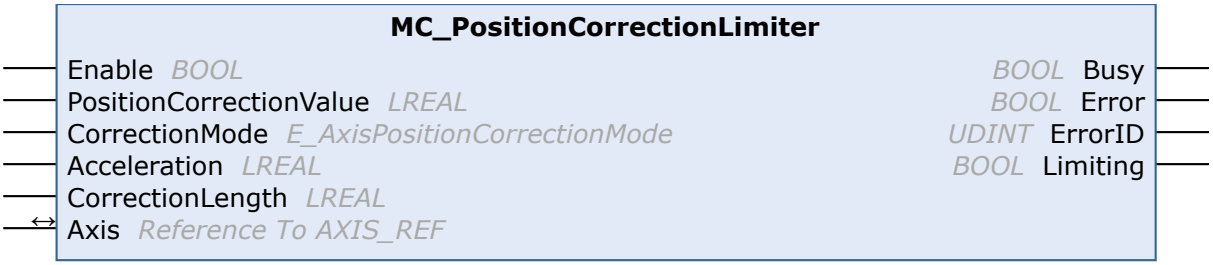
Name	Typ	Beschreibung
OverrideValueFiltered	DWORD	Gefilterter Override-Wert in Digits (der Datentyp passt zum Override im zyklischen Achsinterface 0..1000000).
OverridePercentFiltered	LREAL	Gefilterter Override-Wert in Prozent (0..100%).
Error	BOOL	TRUE, wenn ein Fehler auftritt.
ErrorID	UDINT	Liefert 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 v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

5.5.5 MC_PositionCorrectionLimiter



Mit dem Funktionsbaustein MC_PositionCorrectionLimiter wird ein Korrekturwert (PositionCorrectionValue) auf die Istposition einer Achse geschrieben. Abhängig vom Korrekturmodus werden die Daten entweder unmittelbar oder gefiltert in die Achse eingespeist.

Um diesen Funktionsbaustein erfolgreich nutzen zu können, muss der Parameter „Positionskorrektur“ im System Manager aktiviert werden. Der Funktionsbaustein sollte nur auf freigegebenen Achsen ausgeführt werden.

Eingänge

```
VAR_INPUT
    Enable                  :    BOOL;
    PositionCorrectionValue :    LREAL;
    CorrectionMode          :    E_AxisPositionCorrectionMode;
    Acceleration            :    LREAL;
    CorrectionLength        :    LREAL;
END_VAR
```

Name	Typ	Beschreibung
Enable	BOOL	Aktiviert das kontinuierliche Schreiben des Korrekturwerts „PositionCorrectionValue“. Muss TRUE sein, solange neue Korrekturwerte akzeptiert werden sollen.
PositionCorrectionValue	LREAL	Korrekturwert, der zum Istwert der Achse addiert werden soll.
CorrectionMode	E_AxisPositionCorrectionMode [▶ 144]	Abhängig von diesem Modus wird der Korrekturwert „PositionCorrectionValue“ entweder unmittelbar oder gefiltert geschrieben.
Acceleration	LREAL	Abhängig vom „CorrectionMode“ wird hier die maximale Beschleunigung zum Erreichen des neuen Korrekturwerts vorgegeben. Bei PositionCorrectionMode_Fast hat dieser Wert einen unmittelbaren Einfluss auf das Positionsdelta per PLC-Tick. Maximal zulässiger Korrekturwert Positionsdelta = Beschleunigung * (SPS-Zykluszeit)2. Wenn Acceleration gleich 0.0 parametriert wird, wird die Positionskorrektur nicht begrenzt.
CorrectionLength	LREAL	Wird aktiv, wenn der „CorrectionMode“ mit PositionCorrectionMode_FullLength [▶ 144] übereinstimmt. Eine Änderung beim „PositonCorrectionValue“ wird auf dieser Korrekturlänge aufgeteilt.

Ein-/Ausgänge

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

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	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
    Busy     :    BOOL;
    Error    :    BOOL;
    ErrorID  :    UDINT;
    Limiting :    BOOL;
END_VAR
```

Name	Typ	Beschreibung
Busy	BOOL	TRUE, sobald der Funktionsbaustein aktiv ist. FALSE, wenn er in den ursprünglichen Zustand zurückkehrt.

Name	Typ	Beschreibung
Error	BOOL	TRUE, wenn ein Fehler auftritt.
ErrorID	UDINT	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.
Limiting	BOOL	TRUE, wenn der geforderte Korrekturwert „PositionCorrectionValue“ noch nicht vollständig akzeptiert ist.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

5.5.6 MC_ReadDriveAddress

Mit dem Funktionsbaustein MC_ReadDriveAddress werden die ADS-Zugriffdaten für ein an die Achse angeschlossenes Antriebsgerät gelesen. Diese Informationen werden für den Zugriff auf das Gerät, zum Beispiel zur speziellen Parametrierung, benötigt.

Eingänge

```
VAR_INPUT
    Execute : BOOL; (* B *)
END_VAR
```

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

Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 14]

Ein-/Ausgänge

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

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	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; (* B *)
    Busy         : BOOL; (* E *)
    Error        : BOOL; (* B *)
    ErrorID      : DWORD; (* B *)
    DriveAddress : ST_DriveAddress; (* B *)
END_VAR
```

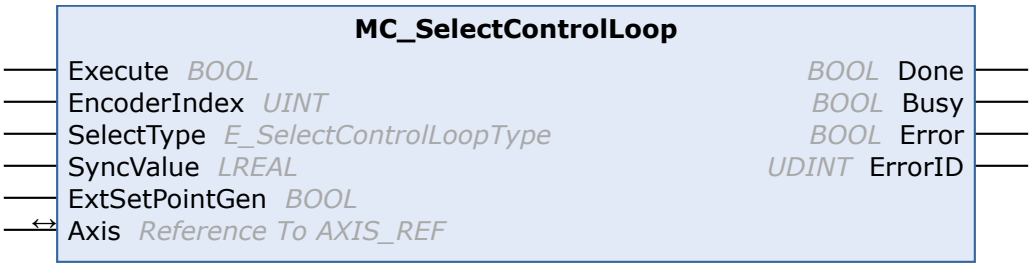
Name	Typ	Beschreibung
Done	BOOL	TRUE, wenn das Kommando fehlerfrei ausgeführt wurde.
Busy	BOOL	TRUE, sobald das Kommando mit Execute gestartet wird und solange der Befehl abgearbeitet wird. Wenn „Busy“ FALSE ist, ist der Funktionsbaustein bereit für einen neuen Auftrag.
Error	BOOL	TRUE, wenn ein Fehler auftritt.
ErrorID	DWORD	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.
DriveAddress	ST_DriveAddress	ADS-Zugriffsdaten eines mit der Achse verbunden Antriebsgerätes. (Typ: ST_DriveAddress [▶ 155])

Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 14]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

5.5.7 MC_SelectControlLoop



Mit dem Funktionsbaustein MC_SelectControlLoop wird zwischen Regelkreisen einer Achse umgeschaltet. Voraussetzung ist, dass in der Systemkonfiguration zwei oder mehr Regelkreise unterhalb der Achse angelegt wurden.

Siehe Beispiel „Regelkreisumschaltung bei einem AX5000 mit zwei vorhandenen Encodern [▶ 165]“.

Eingänge

```
VAR_INPUT
Execute : BOOL;
EncoderIndex : UINT;
SelectType : E_SelectControlLoopType;
SyncValue : LREAL;
ExtSetPointGen : BOOL;
END_VAR
```

Name	Typ	Beschreibung
Execute	BOOL	Mit einer steigenden Flanke wird das Kommando ausgeführt.
EncoderIndex	UINT	Die laufende Nummer [0..9] des zu aktivierenden Regelkreises der Achse.
SelectType	E_SelectControlLoopType [▶ 145]	Legt fest, wie die Umschaltung der Regelkreise durchgeführt wird. Derzeit nur mit dem Wert SelectControlLoopType_Standard anwendbar.
SyncValue	LREAL	Derzeit nicht anwendbar.
ExtSetPointGen	BOOL	Derzeit nicht anwendbar.

Ein-/Ausgänge

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

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	Achsdatenstruktur, welche eine Achse eindeutig im System adressiert. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.

Ausgänge

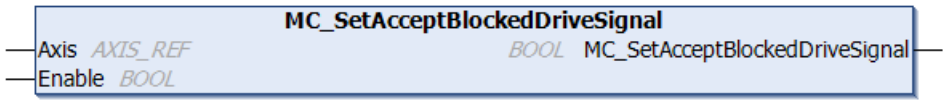
```
VAR_OUTPUT
Done : BOOL;
Busy : BOOL;
Error : BOOL;
ErrorID : UDINT;
END_VAR
```

Name	Typ	Beschreibung
Done	BOOL	TRUE, wenn das Kommando fehlerfrei ausgeführt 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.
Error	BOOL	TRUE, wenn ein Fehler auftritt.
ErrorID	UDINT	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4024	PC oder CX (x86)	Tc2_MC2

5.5.8 MC_SetAcceptBlockedDriveSignal



Es gibt Situationen, in denen ein Antrieb nicht mehr den NC-Sollwerten folgt, beispielsweise wenn eine Achse auf einen Endschalter fährt. Eine solche Situation wird von der NC als Fehler erkannt und der Antrieb wird stillgesetzt. Der Anwender kann aber auch bewusst eine solche Situation provozieren wollen, z. B. um für einen Referenzierfahrt auf einen Endschalter zu fahren.

Mit der Funktion MC_SetAcceptBlockedDriveSignal kann der Anwender vorübergehend verhindern, dass die NC-Achse in einen Fehler läuft, weil der Antrieb den Sollwerten der NC nicht mehr folgt.

- Siehe auch Bit 8 des ControlDWords im AXIS_REF [▶ 124].
- Ein SERCOS/SoE-Antrieb meldet über das Statusbit 3 des Antriebsstatuswortes S-0-0135 "Drive follows the command values".
- Ein CanOpen/CoE-Antrieb meldet über das Statusbit 12 des Objekts 6041h "Drive follows the command values".

FUNCTION MC_SetAcceptBlockedDriveSignal: BOOL

Eingänge

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

Name	Typ	Beschreibung
Enable	BOOL	NC-Reglerfreigabe für die Achse

Ein-/Ausgänge

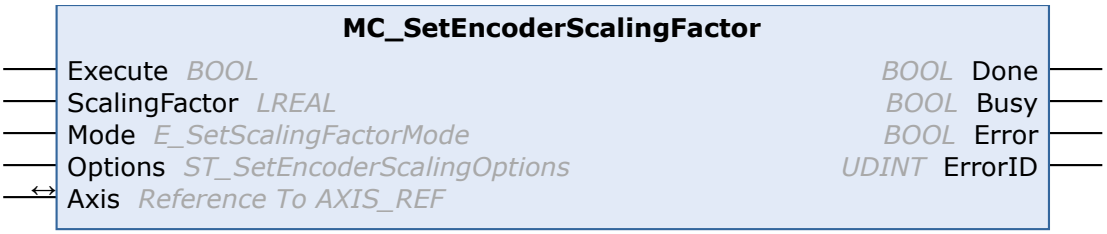
```
VAR_IN_OUT
    Axis : AXIS_REF;
END_VAR
```

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	Achsdatenstruktur, welche eine Achse eindeutig im System adressiert. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

5.5.9 MC_SetEncoderScalingFactor



Mit dem Funktionsbaustein MC_SetEncoderScalingFactor wird der Skalierungsfaktor des aktiven Encoders einer Achse entweder im Stillstand oder in Bewegung geändert.

Die Änderung kann absolut oder relativ durchgeführt werden. Da sich durch Änderung des Skalierungsfaktors im absoluten Modus ein Positionssprung ergibt, ist dieser Mode nur im Stilstand geeignet. Im relativen Modus wird gleichzeitig ein interner Positionsoffset so angepasst, dass es keinen Sprung gibt. Es muss beachtet werden, dass sich durch einen Eingriff während der Fahrt bei gleichbleibender realer Geschwindigkeit die Ist-Geschwindigkeit der Achse ändert. Daher können nur kleine Änderungen während der Fahrt durchgeführt werden.

Eingänge

```
VAR_INPUT
    Execute       : BOOL;
    ScalingFactor : LREAL;
    Mode          : E_SetScalingFactorMode;
    Options       : ST_SetEncoderScalingOptions;
END_VAR
```

Name	Typ	Beschreibung
Execute	BOOL	Mit einer steigenden Flanke wird das Kommando ausgeführt.
ScalingFactor	LREAL	Skalierungsfaktor des aktiven Encoders einer Achse. Der Skalierungsfaktor wird in physikalischen Positionseinheiten [u] dividiert durch die Anzahl der Encoder-Inkremente angegeben.
Mode	E_SetScalingFactorMode [▶ 148]	Der Skalierungsfaktor kann im absoluten oder relativen Modus gesetzt werden (ENCODERSCALINGMODE_ABSOLUTE, ENCODERSCALINGMODE_RELATIVE). Im absoluten Modus beginnt die Zählung im Nullpunkt des Achskoordinatensystems, daher ergibt sich ein Positionssprung, wenn der Skalierungsfaktor geändert

Name	Typ	Beschreibung
		wird. Im relativen Modus ändert sich die Istposition der Achse nicht. Dieser Mode ist daher auch für die Änderung in Bewegung geeignet.
Options	ST_SetEncoderScalingOptions [▶ 156]	Datenstruktur, die zusätzliche, selten benötigte Parameter enthält. Im Normalfall kann der Eingang offen bleiben. • **SelectEncoderIndex:** Kann gesetzt werden, wenn eine Achse mit mehreren Encodern verwendet wird und die Position eines bestimmten Encoders (Options.EncoderIndex) gesetzt werden soll. • **EncoderIndex:** Gibt den Encoder (0..n) an, wenn „SelectEncoderIndex“ TRUE ist.

Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 14]

Ein-/Ausgänge

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

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	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
```

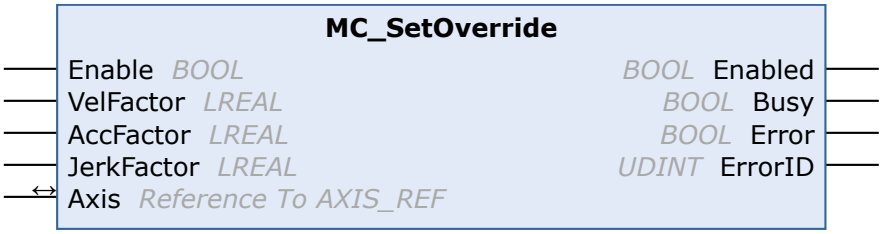
Name	Typ	Beschreibung
Done	BOOL	TRUE, wenn die Position erfolgreich gesetzt wurde.
Busy	BOOL	TRUE, sobald das Kommando mit Execute gestartet wird und solange der Befehl abgearbeitet wird. Wenn „Busy“ FALSE ist, ist der Funktionsbaustein bereit für einen neuen Auftrag. Gleichzeitig ist einer der Ausgänge „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 [▶ 14]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

5.5.10 MC_SetOverride



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

Eingänge

```
VAR_INPUT
    Enable     : BOOL; (* B *)
    VelFactor  : LREAL (* B *) := 1.0; (* 1.0 = 100% *)
    AccFactor  : LREAL (* E *) := 1.0; (* 1.0 = 100% *) (* not supported *)
    JerkFactor : LREAL (* E *) := 1.0; (* 1.0 = 100% *) (* not supported *)
END_VAR
```

Name	Typ	Beschreibung
Enable	BOOL	TRUE, solange das Kommando ausgeführt wird.
VelFactor	LREAL	Geschwindigkeits-Override-Faktor
AccFactor	LREAL	Nicht unterstützt
JerkFactor	LREAL	Nicht unterstützt

Ein-/Ausgänge

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

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	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
    Enabled : BOOL;
    Busy    : BOOL;
    Error   : BOOL;
    ErrorID : UDINT;
END_VAR
```

Name	Typ	Beschreibung
Enabled	BOOL	Der parametrierte Override wird gesetzt.
Busy	BOOL	TRUE, sobald das Kommando mit Enable gestartet wird und solange der Befehl abgearbeitet wird. Wenn „Busy“ FALSE ist, ist der Funktionsbaustein bereit für einen neuen Auftrag.
Error	BOOL	TRUE, wenn ein Fehler auftritt.
ErrorID	UDINT	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

5.5.11 MC_WriteNcIoOutput

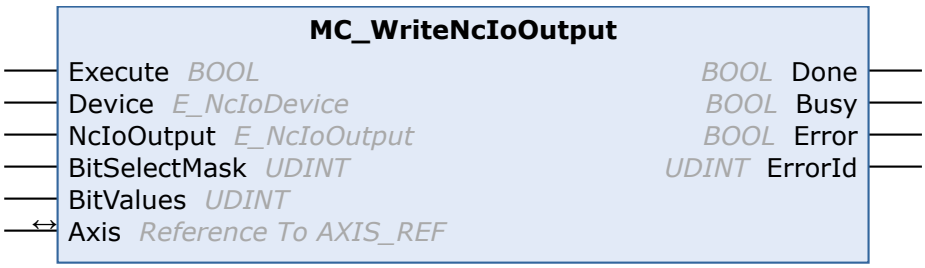


Abb. 1:

Mit dem Funktionsbaustein MC_WriteNcIoOutput können nicht verwendete IO-Ausgänge der Achse beschrieben werden.

Eingänge

```
VAR_INPUT
    Execute           : BOOL;
    Device            : E_NcIoDevice := E_NcIoDevice.NcIoDeviceDrive;
    NcIoOutput        : E_NcIoOutput := E_NcIoOutput.NcIoOutputnCtrl1;
    BitSelectMask     : UDINT := 16#0;
    BitValues         : UDINT;
END_VAR
```

Name	Typ	Beschreibung
Execute	BOOL	Mit einer steigenden Flanke wird das Kommando ausgeführt.
Device	E_NcIoDevice [▶ 143]	Auswahl der Nc-Achskomponente, dessen IO-Objekt modifiziert werden soll (Encoder oder Drive).
NcIoOutput	E_NcIoOutput [▶ 143]	Auswahl des Unterobjektes, dessen Wert modifiziert werden soll (z. B. nCtrl1).
BitSelectMask	UDINT	Maske zum Selektieren, welche Bits modifiziert werden sollen.
BitValues	UDINT	Werte der entsprechenden zu modifizierenden Bits.

Ein-/Ausgänge

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

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	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
```

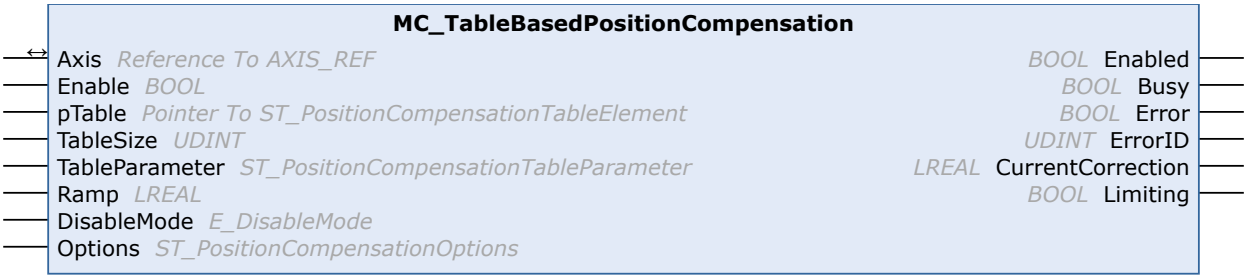
Name	Typ	Beschreibung
Done	BOOL	TRUE, wenn das Kommando fehlerfrei ausgeführt wurde.

Name	Typ	Beschreibung
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.
Error	BOOL	TRUE, wenn ein Fehler auftritt.
ErrorID	UDINT	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4024	PC oder CX (x86)	Tc2_MC2

5.5.12 MC_TableBasedPositionCompensation



Mit dem Funktionsbaustein MC_TableBasedPositionCompensation erfolgt eine Korrektur der Achsposition mit einem Korrekturfaktor abhängig von der aktuellen Achsposition. Die Korrekturwerte müssen dazu in einer äquidistanten, monoton ansteigenden Tabelle abgelegt werden.

Eingänge

```
VAR_INPUT
    Enable          : BOOL;
    pTable          : POINTER To ST_PositionCompensationTableElement;
    TableSize       : UDINT;
    TableParameter  : ST_PositionCompensationTableParameter;
    Ramp            : LREAL;
    DisableMode     : E_DisableMode;
    Options         : ST_PositionCompensationOptions;
END_VAR
```

Name	Typ	Beschreibung
Enable	BOOL	Das Kommando wird so lange ausgeführt, wie Enable aktiv ist.
pTable	POINTER To ST_PositionCompensationTableElement [▶ 144]	Zeiger auf die Kompensationstabelle, welche ein Array des Typen St_PositionCompensationTableElement ist.
TableSize	UDINT	Größe der Kompensationstabelle
TableParameter	ST_PositionCompensationTableParameter [▶ 144]	Datenstruktur mit zusätzlichen Parametern zur Kompensationstabelle.
Ramp	LREAL	Geschwindigkeitslimit für eingegebene Kompensation (konstante Geschwindigkeit und lineare Position als Unterprofile für die Tabellenkompensation [mm/s]).
DisableMode	E_DisableMode	Abschaltmodus: DisableModeHold = die letzte Korrektur wird bei behalten DisableModeReset = die Korrektur wird auf 0 gesetzt
Options	ST_PositionCompensationOptions	Optionale Parameter (nicht implementiert)

Ein-/Ausgänge

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

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	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
    Enabled           : BOOL;
    Busy              : BOOL;
    Error             : BOOL;
    ErrorID           : UDINT;
    CurrentCorrection : LREAL;
    Limiting          : BOOL;
END_VAR
```

Name	Typ	Beschreibung
Enabled	BOOL	Dieser Ausgang wird TRUE, wenn die Tabellenkompensation ohne Fehler aktiviert wurde.
Busy	BOOL	Dieser Ausgang wird TRUE, wenn das Kommando mit Enable gestartet wird, und bleibt es dann so lange, wie der Funktionsbaustein das Kommando ausführt.
Error	BOOL	Dieser Ausgang wird TRUE, wenn bei der Ausführung des Kommandos ein Fehler aufgetreten ist.
ErrorID	UDINT	Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Kommandos. Der Fehlercode kann in der ADS-Fehlerdokumentation oder in der NC-Fehlerdokumentation (Fehlercodes 0x4nnn und 0x8nnn) nachgeschlagen werden.
CurrentCorrection	LREAL	Aktueller Kompensations-Wert, in der Einheit der Achse.
Limiting	BOOL	Dieser Ausgang ist TRUE, wenn der zur Position gehörende Korrekturwert noch nicht vollständig übernommen ist.

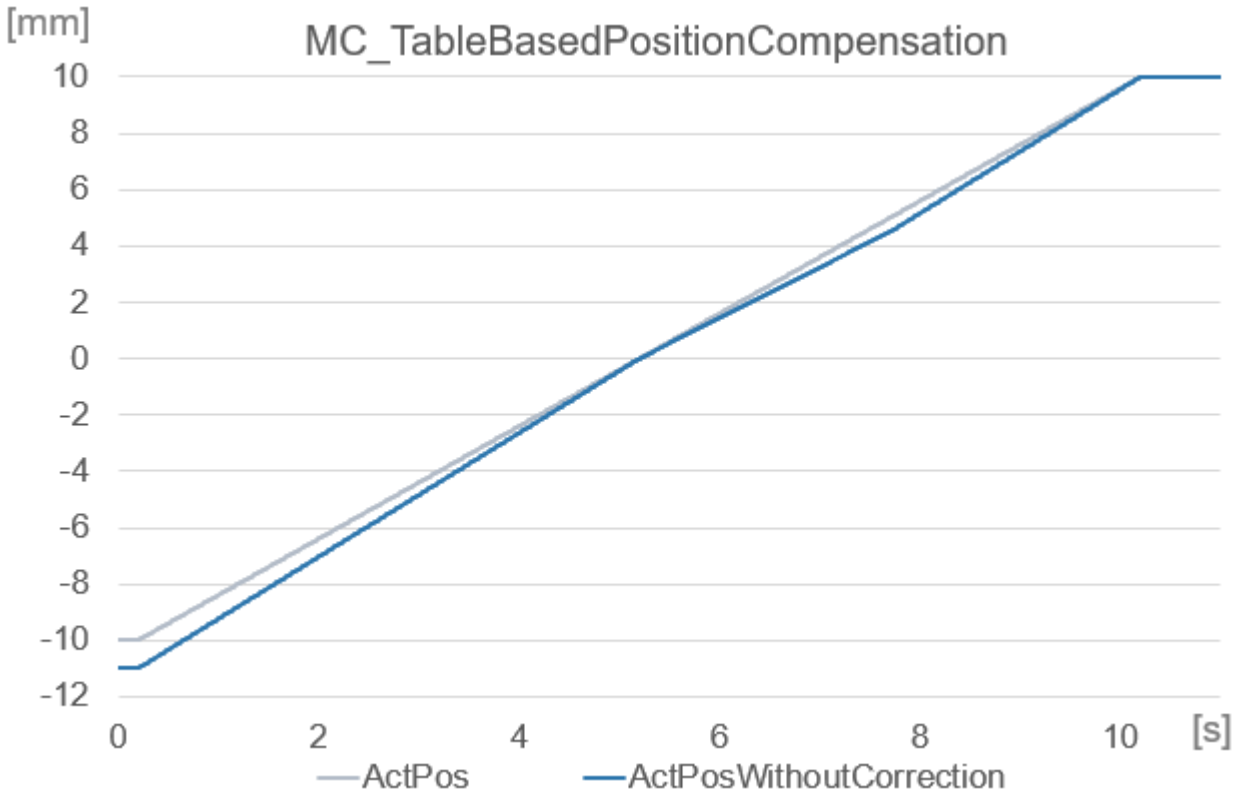
Weiterführende Informationen

Nachfolgend beispielhaft eine Positionstabelle und die dazugehörigen Tabellenparameter:

```
VAR_INPUT
    …
    stParameter : ST_PositionCompensationTableParameter
                  := (MinPosition := -10.0, MaxPosition := 10.0,
                      NoOfTableElements := 21, Direction := WorkDirectionBoth);
    stPosTable  : ARRAY[0..20] OF ST_PositionCompensationTableElement
                  := [ ( Position := -10.0, Compensation := 1.0 ),
                       ( Position :=  -9.0, Compensation := 0.9 ),
                       ( Position :=  -8.0, Compensation := 0.8 ),
                       ( Position :=  -7.0, Compensation := 0.7 ),
                       ( Position :=  -6.0, Compensation := 0.6 ),
                       ( Position :=  -5.0, Compensation := 0.5 ),
                       ( Position :=  -4.0, Compensation := 0.4 ),
                       ( Position :=  -3.0, Compensation := 0.3 ),
                       ( Position :=  -2.0, Compensation := 0.2 ),
                       ( Position :=  -1.0, Compensation := 0.1 ),
                       ( Position :=   0.0, Compensation := 0.0 ),
                       ( Position :=   1.0, Compensation := 0.1 ),
                       ( Position :=   2.0, Compensation := 0.2 ),
                       ( Position :=   3.0, Compensation := 0.3 ),
                       ( Position :=   4.0, Compensation := 0.4 ),
                       ( Position :=   5.0, Compensation := 0.5 ),
                       ( Position :=   6.0, Compensation := 0.4 ),
                       ( Position :=   7.0, Compensation := 0.3 ),
                       ( Position :=   8.0, Compensation := 0.2 ),
                       ( Position :=   9.0, Compensation := 0.1 ),
                       ( Position :=  10.0, Compensation := 0.0 )
```

```
                        ];
    …
END_VAR
```

Grafisch dargestellt ergibt sich mit der Tabelle folgendes Korrekturverhalten:



Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4024	PC oder CX (x86/x64)	Tc2_MC2

5.5.13 MC_ConvertIncPosToPos

Der Funktionsblock MC_ConvertIncPosToPos konvertiert die inkrementelle Encoderposition in eine korrespondierende NC-Achsposition.

Die inkrementelle Position wird mit dem Skalierungsfaktor skaliert und mit dem derzeit gültigen Positionsoffset beaufschlagt. Die ermittelte NC-Achsposition befindet sich im selben Koordinatensystem wie die Istposition der Achse.

Zu beachten: Bei der inkrementellen Position ist es so, dass die inkrementelle Encoderposition regelmäßig überläuft. Die Umrechnung berücksichtigt keine vergangenen oder zukünftigen Überläufe und das Ergebnis bezieht sich auf das aktuelle Encoder-Feedback.

Eingänge

```
VAR_INPUT
    Axis    : AXIS_REF;
    IncPos  : UDINT;
    Options : ST_NcPositionConversionOptions;
END_VAR
```

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	Achsdatenstruktur, welche eine Achse eindeutig im System adressiert. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.
InPos	UDINT	Inkrementelle Encoderposition
Options	ST_NcPositionConversionOptions [▶ 156]	Optionale Parameter

Ausgänge

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

Name	Typ	Beschreibung
Valid	BOOL	Die berechnete Position ist gültig.
Position	LREAL	Position der NC-Achse Liefert die zur IncPos korrespondierende NC-Achsposition. Hierbei handelt es sich um eine um die Skalierung und Offset verrechnete NC-Achsposition mit beispielsweise der physikalischen Einheit Grad oder mm.
Error	BOOL	TRUE, wenn ein Fehler auftritt.
ErrorID	UDINT	Liefert im Fehlerfall eine Fehlernummer.

Voraussetzungen

Bibliothek	Version
Tc2_MC2	3.3.59

5.5.14 MC_ConvertPosToIncPos

Der Funktionsblock MC_ConvertPosToIncPos konvertiert eine NC-Achsposition in eine korrespondierende inkrementelle Encoderposition.

Die NC-Achsposition wird mit dem Skalierungsfaktor skaliert und mit dem derzeit gültigen Positionsoffset beaufschlagt.

Zu beachten: Bei der inkrementellen Position ist es so, dass die inkrementelle Encoderposition regelmäßig überläuft. Wenn die NC-Achsposition zu weit von der Istposition der NC-Achse entfernt ist, liefert der Funktionsbaustein einen Fehler.

Eingänge

```
VAR_INPUT
    Axis     : AXIS_REF;
    Position : LREAL;
    Options  : ST_NcPositionConversionOptions;
END_VAR
```

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	Achsdatenstruktur, welche eine Achse eindeutig im System adressiert. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.
Position	LREAL	Position der NC-Achse
Options	ST_NcPositionConversionOptions [▶ 156]	Optionale Parameter

Ausgänge

```
VAR_OUTPUT
    Valid   : BOOL;
    IncPos  : UDINT;
    Error   : BOOL;
    ErrorID : UDINT;
END_VAR
```

Name	Typ	Beschreibung
Valid	BOOL	Die berechnete IncPos ist gültig.
IncPos	UDINT	Liefert die zur Position korrespondierende inkrementelle Encoderposition.
Error	BOOL	TRUE, wenn ein Fehler auftritt.
ErrorID	UDINT	Liefert im Fehlerfall eine Fehlernummer.

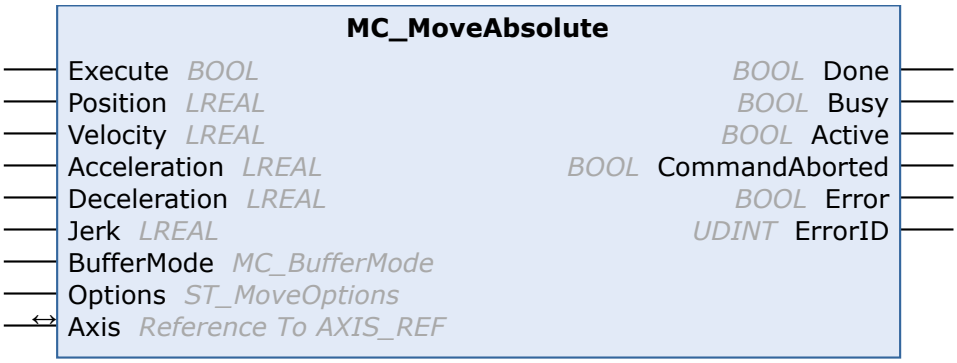
Voraussetzungen

Bibliothek	Version
Tc2_MC2	3.3.59

6 Motion-Bausteine

6.1 Point to Point Motion

6.1.1 MC_MoveAbsolute



Mit dem Funktionsbaustein MC_MoveAbsolute wird eine Positionierung auf eine absolute Zielposition gestartet und die Achsbewegung über den gesamten Fahrweg überwacht. 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 [▶ 73] verwendet werden.

Bewegungskommandos können auf gekoppelte Slave-Achsen angewendet werden, wenn diese Option in den Parametern der Achse explizit aktiviert worden ist. Ein Bewegungskommando wie MC_MoveAbsolute führt dann automatisch zum Abkoppeln der Achse und das Kommando wird anschließend ausgeführt. In diesem Fall ist ausschließlich der BufferMode „Aborting“ möglich.

Eingänge

```
VAR_INPUT
    Execute      : BOOL;
    Position     : LREAL;
    Velocity     : LREAL;
    Acceleration : LREAL;
    Deceleration : LREAL;
    Jerk         : LREAL;
    BufferMode   : MC_BufferMode;
    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) Bei einem Wert von 0 wirkt die Standardbeschleunigung aus der Achskonfiguration im System Manager.
Deceleration	LREAL	Verzögerung (≥0) Bei einem Wert von 0 wirkt die Standardverzögerung aus der Achskonfiguration im System Manager.

Name	Typ	Beschreibung
Jerk	LREAL	Ruck (≥0) Bei einem Wert von 0 wirkt der Standard-Ruck aus der Achskonfiguration im System Manager.
BufferMode	MC_BufferMode [▶ 138]	Wird ausgewertet, wenn die Achse bereits ein anderes Kommando ausführt. MC_MoveAbsolute wird nach dem laufenden Kommando aktiv oder bricht dieses ab. Übergangsbedingungen vom laufenden zum nächsten Kommando werden ebenfalls durch den BufferMode festgelegt. Wird das Kommando auf eine gekoppelte Slave-Achse angewendet, ist nur der BufferMode „Aborting“ möglich. Um den BufferMode zu verwenden, ist immer ein zweiter Funktionsbaustein nötigt. Es ist nicht möglich, einen Move-Baustein mit neuen Parametern zu triggern, während er noch aktiv ist.
Options	ST_MoveOptions [▶ 142]	Datenstruktur, die zusätzliche selten benötigte Parameter enthält. Im Normalfall kann der Eingang offenbleiben.

Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 14]

Ein-/Ausgänge

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

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	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.

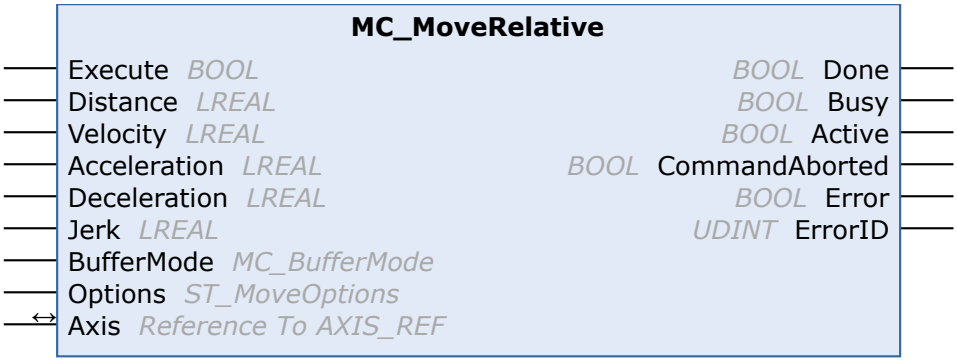
Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 14]

- „Target Position Monitoring“ ist aktiviert (Standard-Fall): Wenn das logische Positionierende erreicht ist (die NC Sollwertgenerierung ist beendet und das HasJob-Bit ist FALSE) UND danach das Bit InTargetPosition TRUE geworden ist, dann wird der Done-Ausgang auf TRUE gesetzt.
- „Position Range Monitoring“ ist aktiviert („Target Position Monitoring“ ist nicht aktiv): Wenn das logische Positionierende erreicht ist (die NC Sollwertgenerierung ist beendet und das HasJob-Bit ist FALSE) UND danach das Bit InPositionArea TRUE geworden ist, dann wird der Done-Ausgang auf TRUE gesetzt.
- Weder „Target Position Monitoring“ noch „Position Range Monitoring“ ist aktiviert: Der Done-Ausgang wird sofort gesetzt, wenn das logische Positionierende erreicht ist (die NC Sollwertgenerierung ist beendet und das HasJob-Bit ist FALSE).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

6.1.2 MC_MoveRelative



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

Bewegungskommandos können auf gekoppelte Slave-Achsen angewendet werden, wenn diese Option in den Parametern der Achse explizit aktiviert worden ist. Ein Bewegungskommando wie MC_MoveRelative führt dann automatisch zum Abkoppeln der Achse und das Kommando wird anschließend ausgeführt. In diesem Fall ist ausschließlich der BufferMode „Aborting“ möglich.

Eingänge

```
VAR_INPUT
    Execute      : BOOL;
    Distance     : LREAL;
    Velocity     : LREAL;
    Acceleration : LREAL;
    Deceleration : LREAL;
    Jerk         : LREAL;
    BufferMode   : MC_BufferMode;
    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).

Name	Typ	Beschreibung
Acceleration	LREAL	Beschleunigung (≥0) Bei einem Wert von 0 wirkt die Standardbeschleunigung aus der Achskonfiguration im System Manager.
Deceleration	LREAL	Verzögerung (≥0) Bei einem Wert von 0 wirkt die Standardverzögerung aus der Achskonfiguration im System Manager.
Jerk	LREAL	Ruck (≥0) Bei einem Wert von 0 wirkt der Standard-Ruck aus der Achskonfiguration im System Manager.
BufferMode	MC_BufferMode [▶ 138]	Wird ausgewertet, wenn die Achse bereits ein anderes Kommando ausführt. MC_MoveRelative wird nach dem laufenden Kommando aktiv oder bricht dieses ab. Übergangsbedingungen vom laufenden zum nächsten Kommando werden ebenfalls durch den BufferMode festgelegt. Wird das Kommando auf eine gekoppelte Slave-Achse angewendet, ist nur der BufferMode „Aborting“ möglich. Um den BufferMode zu verwenden, ist immer ein zweiter Funktionsbaustein nötigt. Es ist nicht möglich, einen Move-Baustein mit neuen Parametern zu triggern, während er noch aktiv ist.
Options	ST_MoveOptions [▶ 142]	Datenstruktur, die zusätzliche selten benötigte Parameter enthält. Im Normalfall kann der Eingang offenbleiben.

Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 14]

Ein-/Ausgänge

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

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	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.

Name	Typ	Beschreibung
Error	BOOL	TRUE, wenn ein Fehler auftritt.
ErrorID	UDINT	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

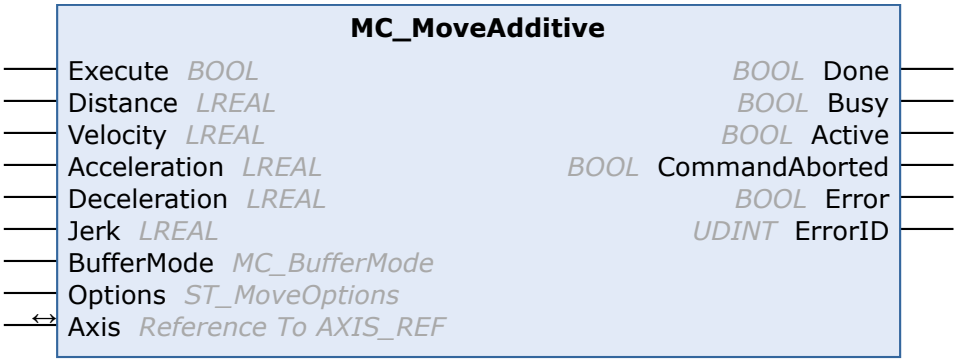
Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 14]

- „Target Position Monitoring“ ist aktiviert (Standard-Fall): Wenn das logische Positionierende erreicht ist (die NC Sollwertgenerierung ist beendet und das HasJob-Bit ist FALSE) UND danach das Bit InTargetPosition TRUE geworden ist, dann wird der Done-Ausgang auf TRUE gesetzt.
- „Position Range Monitoring“ ist aktiviert („Target Position Monitoring“ ist nicht aktiv): Wenn das logische Positionierende erreicht ist (die NC Sollwertgenerierung ist beendet und das HasJob-Bit ist FALSE) UND danach das Bit InPositionArea TRUE geworden ist, dann wird der Done-Ausgang auf TRUE gesetzt.
- Weder „Target Position Monitoring“ noch „Position Range Monitoring“ ist aktiviert: Der Done-Ausgang wird sofort gesetzt, wenn das logische Positionierende erreicht ist (die NC Sollwertgenerierung ist beendet und das HasJob-Bit ist FALSE).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

6.1.3 MC_MoveAdditive



Mit dem Funktionsbaustein MC_MoveAdditive wird eine relative Positionierung ausgehend von der letzten beauftragten Zielposition gestartet, unabhängig davon, ob diese erreicht wurde. Der Ausgang „Done“ wird gesetzt, wenn die Zielposition angefahren wurde. Anderenfalls wird der Ausgang „CommandAborted“ oder im Fehlerfall der Ausgang „Error“ gesetzt.

Ist keine letzte Zielposition bekannt oder bewegt sich die Achse endlos, wird die Bewegung ausgehend von der aktuellen Sollposition der Achse ausgeführt.

Für Eil-/Schleichachsen ist MC_MoveAdditive nicht implementiert.

Eingänge

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

```
    BufferMode   : MC_BufferMode;
    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) Bei einem Wert von 0 wirkt die Standardbeschleunigung aus der Achskonfiguration im System Manager.
Deceleration	LREAL	Verzögerung (≥0) Bei einem Wert von 0 wirkt die Standardverzögerung aus der Achskonfiguration im System Manager.
Jerk	LREAL	Ruck (≥0) Bei einem Wert von 0 wirkt der Standard-Ruck aus der Achskonfiguration im System Manager.
BufferMode	MC_BufferMode [▶ 138]	Wird ausgewertet, wenn die Achse bereits ein anderes Kommando ausführt. MC_MoveAdditive wird nach dem laufenden Kommando aktiv oder bricht dieses ab. Übergangsbedingungen vom laufenden zum nächsten Kommando werden ebenfalls durch den BufferMode festgelegt. Um den BufferMode zu verwenden, ist immer ein zweiter Funktionsbaustein nötigt. Es ist nicht möglich, einen Move-Baustein mit neuen Parametern zu triggern, während er noch aktiv ist.
Options	ST_MoveOptions [▶ 142]	Datenstruktur, die zusätzliche selten benötigte Parameter enthält. Im Normalfall kann der Eingang offenbleiben.

Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 14]

Ein-/Ausgänge

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

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	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.

Name	Typ	Beschreibung
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.

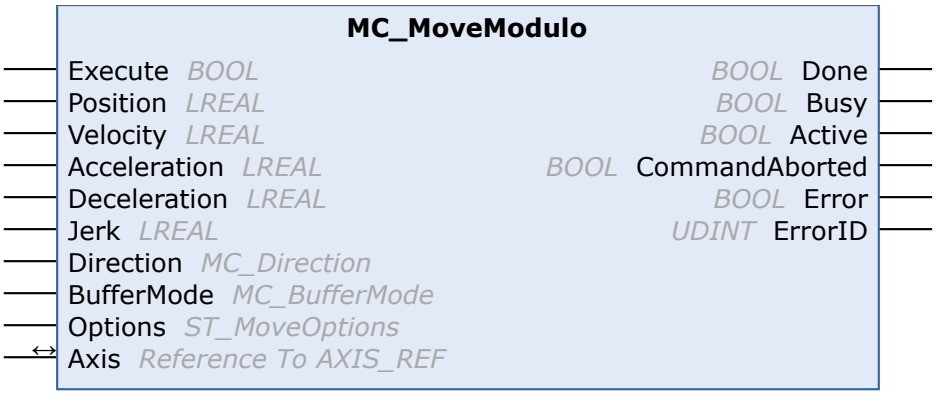
Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 14]

- „Target Position Monitoring“ ist aktiviert (Standard-Fall): Wenn das logische Positionierende erreicht ist (die NC Sollwertgenerierung ist beendet und das HasJob-Bit ist FALSE) UND danach das Bit InTargetPosition TRUE geworden ist, dann wird der Done-Ausgang auf TRUE gesetzt.
- „Position Range Monitoring“ ist aktiviert („Target Position Monitoring“ ist nicht aktiv): Wenn das logische Positionierende erreicht ist (die NC Sollwertgenerierung ist beendet und das HasJob-Bit ist FALSE) UND danach das Bit InPositionArea TRUE geworden ist, dann wird der Done-Ausgang auf TRUE gesetzt.
- Weder „Target Position Monitoring“ noch „Position Range Monitoring“ ist aktiviert: Der Done-Ausgang wird sofort gesetzt, wenn das logische Positionierende erreicht ist (die NC Sollwertgenerierung ist beendet und das HasJob-Bit ist FALSE).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

6.1.4 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 Achsparameter Modulo-Faktor (z. B. 360°). Abhängig vom Eingang „Direction“ werden drei mögliche Starttypen unterschieden.

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

Bewegungskommandos können auf gekoppelte Slave-Achsen angewendet werden, wenn diese Option in den Parametern der Achse explizit aktiviert worden ist. Ein Bewegungskommando wie MC_MoveModulo führt dann automatisch zum Abkoppeln der Achse und das Kommando wird anschließend ausgeführt. In diesem Fall ist ausschließlich der BufferMode „Aborting“ möglich.

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. Gleiches gilt für einen Start mit dem BufferMode „MC_Buffered“.

Start einer Achse aus der Bewegung

Wenn sich eine Achse bereits in Bewegung befindet, sind einige Besonderheiten zu beachten.

Die Anzahl zusätzlicher Umdrehungen kann nicht vom Anwender bestimmt werden. Das System rechnet selbständig, wie die Achse auf möglichst kurzem Weg auf die Zielposition positioniert werden kann.

Der Fehlerausgang muss zwingend ausgewertet werden, da unter bestimmten Bedingungen ein orientierter Stopp nicht möglich ist. Beispielsweise könnte kurz zuvor ein Standard-Stopp 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.

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 Grad und wird sie auf 90 Grad positioniert, wird keine Bewegung ausgeführt. Bei Anforderung von 450 Grad 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 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.

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 [▶ 67] zu positionieren.

> **i** Die Modulo-Positionierung steht ebenso wie die Absolut-Positionierung für alle Achsen unabhängig von der Modulo-Einstellung im TwinCAT System Manager zur Verfügung. Für jede Achse kann die aktuelle Absolutposition „SetPos“ aus dem zyklischen Achsinterface Datentyp NCTOPLC_AXIS_REF [▶ 125] ausgelesen werden.

Siehe auch: Hinweise zur Modulo-Positionierung [▶ 76]

Eingänge

```
VAR_INPUT
    Execute      : BOOL;
    Position     : LREAL;
    Velocity     : LREAL;
    Acceleration : LREAL;
    Deceleration : LREAL;
    Jerk         : LREAL;
    Direction    : MC_Direction;
    BufferMode   : MC_BufferMode;
    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).

Name	Typ	Beschreibung
Acceleration	LREAL	Beschleunigung (≥0) Bei einem Wert von 0 wirkt die Standardbeschleunigung aus der Achskonfiguration im System Manager.
Deceleration	LREAL	Verzögerung (≥0) Bei einem Wert von 0 wirkt die Standardverzögerung aus der Achskonfiguration im System Manager.
Jerk	LREAL	Ruck (≥0) Bei einem Wert von 0 wirkt der Standard-Ruck aus der Achskonfiguration im System Manager.
Direction	MC_Direction [▶ 142]	Positive oder negative Fahrtrichtung. Falls die Achse aus der Bewegung heraus gestartet wird, darf die Richtung nicht umgekehrt werden.
BufferMode	MC_BufferMode [▶ 138]	Wird ausgewertet, wenn die Achse bereits ein anderes Kommando ausführt. MC_MoveModulo wird nach dem laufenden Kommando aktiv oder bricht dieses ab. Übergangsbedingungen vom laufenden zum nächsten Kommando werden ebenfalls durch den BufferMode festgelegt. Um den BufferMode zu verwenden, ist immer ein zweiter Funktionsbaustein nötigt. Es ist nicht möglich, einen Move-Baustein mit neuen Parametern zu triggern, während er noch aktiv ist.
Options	ST_MoveOptions [▶ 142]	Datenstruktur, die zusätzliche selten benötigte Parameter enthält. Im Normalfall kann der Eingang offen bleiben.

Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 14]

Ein-/Ausgänge

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

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	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 [▶ 14]

- „Target Position Monitoring“ ist aktiviert (Standard-Fall): Wenn das logische Positionierende erreicht ist (die NC Sollwertgenerierung ist beendet und das HasJob-Bit ist FALSE) UND danach das Bit InTargetPosition TRUE geworden ist, dann wird der Done-Ausgang auf TRUE gesetzt.
- „Position Range Monitoring“ ist aktiviert („Target Position Monitoring“ ist nicht aktiv): Wenn das logische Positionierende erreicht ist (die NC Sollwertgenerierung ist beendet und das HasJob-Bit ist FALSE) UND danach das Bit InPositionArea TRUE geworden ist, dann wird der Done-Ausgang auf TRUE gesetzt.
- Weder „Target Position Monitoring“ noch „Position Range Monitoring“ ist aktiviert: Der Done-Ausgang wird sofort gesetzt, wenn das logische Positionierende erreicht ist (die NC Sollwertgenerierung ist beendet und das HasJob-Bit ist FALSE).

Voraussetzungen

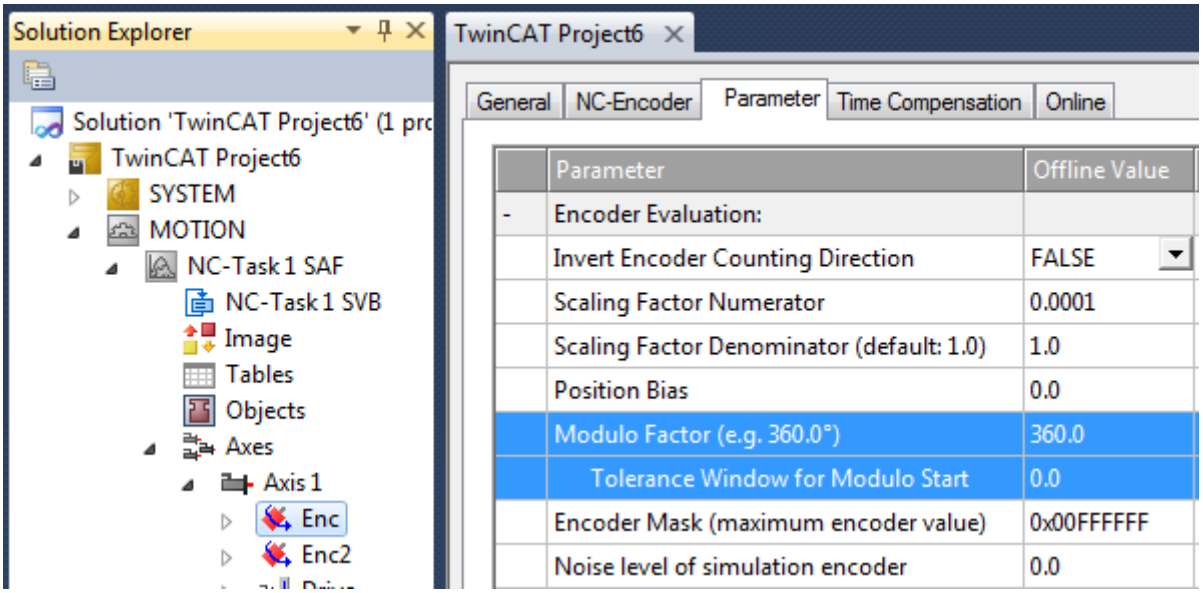
Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

6.1.5 Hinweise zur Modulo-Positionierung

Die Modulo-Positionierung (MC_MoveModulo [▶ 73]) 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 im TwinCAT System Manager

Die Modulo-Positionierung bezieht sich grundsätzlich auf eine im TwinCAT System Manager 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

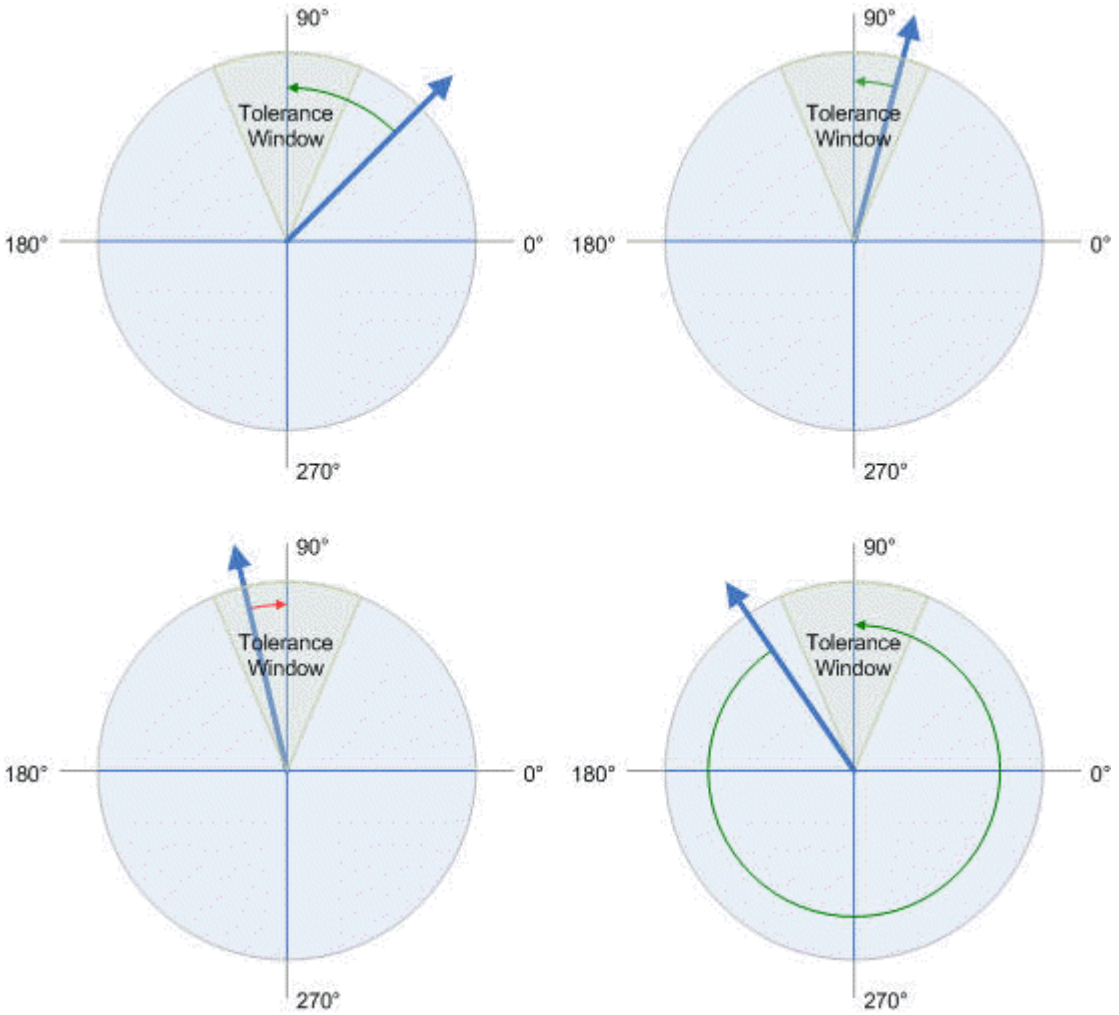
Die Positionierung einer Achse bezieht sich immer auf deren Sollposition. Die Sollposition der Achse ist im Normalfall die Position, die mit dem letzten Fahrauftrag angefahren wurde. Durch einen Achs-Reset (MC_Reset [▶ 19], Zuschalten der Reglerfreigabe mit MC_Power [▶ 18]) kann sich eine vom Anwender 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.

Um das Problem zu lösen, kann ein Modulo-Toleranzfenster im TwinCAT System Manager 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.

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 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 [▶ 142]). 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°. Das Modulo-Toleranzfenster (TF) ist hier auf 1° eingestellt. Besondere Fälle, in denen die Ausgangsposition außerhalb dieses Fensters liegt, sind gekennzeichnet.

Richtung (Modulo-Start-typ)	Absolute An-fangsposition	Modulo-Zielposi-tion	Relativer Ver-fahrweg	Absolute Endpo-sition	Modulo-Endposi-tion	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	
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	

Richtung (Modulo-Start-typ)	Absolute An-fangsposition	Modulo-Zielposi-tion	Relativer Ver-fahrweg	Absolute Endpo-sition	Modulo-Endposi-tion	Anmerkung
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, muss diese Berechnung mit der aktuellen Sollposition der Achse durchgeführt werden (Axis.NcToPlc.ModuloSetPos und Axis.NcToPlc.ModuloSetTurns).

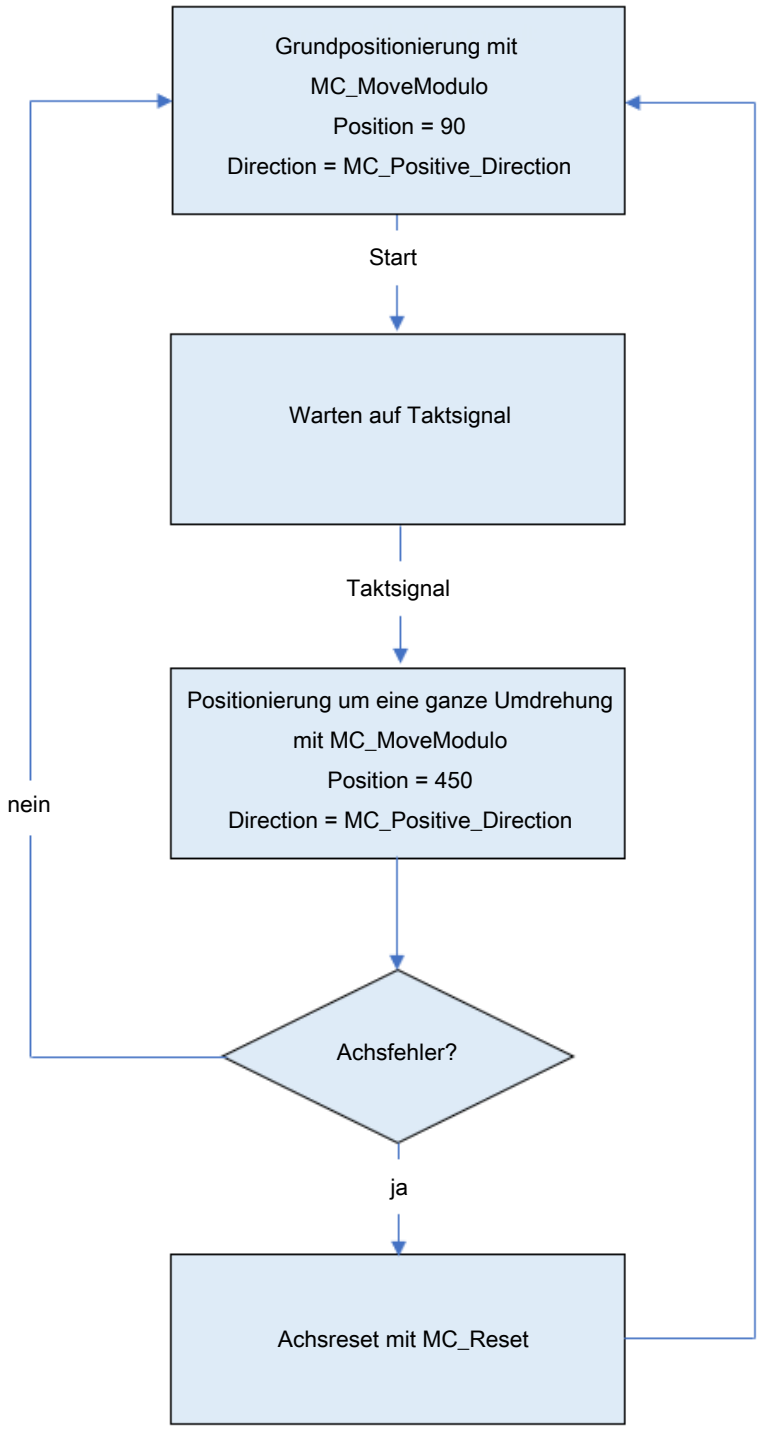
Es ist nicht zu empfehlen, Auftragsberechnungen auf Basis der Modulo-Istposition durchzuführen, die im zyklischen Achsinterface (ModuloActPos und ModuloActTurns) zur Verfügung steht. Wegen der mehr oder weniger großen Regelabweichung der Achse könnten sich Fehler im programmierten Ablauf, wie z. B. unerwünschte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

Das Modulo-Toleranzfenster wird im System Manager auf 1,5° eingestellt. Damit werden unerwünschte Umdrehungen der Achse nach einem Achs-Reset vermieden. Da die Achse nur vorpositioniert werden darf, wird das Bewegungskommando MC_MoveModulo [▶ 73] 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 [▶ 73]), mit der die exakte Ausgangsposition sichergestellt wird. Anschließend wechselt die Schrittkette in einen Bearbeitungszyklus. Im Fehlerfall wird die Achse mit MC_Reset [▶ 19] 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.

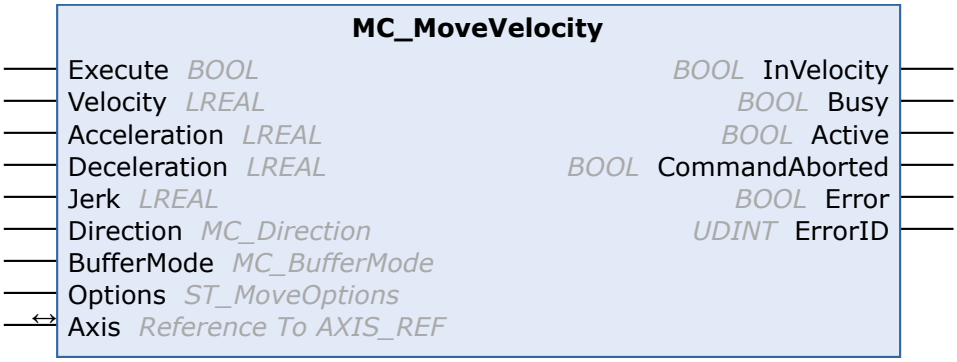


Der Reset-Schritt kann alternativ auch am Anfang der Schrittkette ausgeführt werden, um die Achse auch zu Beginn des Ablaufs zu initialisieren.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

6.1.6 MC_MoveVelocity



Mit dem Funktionsbaustein MC_MoveVelocity wird eine Endlosfahrt mit vorgegebener Geschwindigkeit und Richtung gestartet. 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.

Bewegungskommandos können auf gekoppelte Slave-Achsen angewendet werden, wenn diese Option in den Parametern der Achse explizit aktiviert worden ist. Ein Bewegungskommando wie MC_MoveAbsolute führt dann automatisch zum Abkoppeln der Achse und das Kommando wird anschließend ausgeführt. In diesem Fall ist ausschließlich der BufferMode „Aborting“ möglich.

Eingänge

```
VAR_INPUT
    Execute      : BOOL; (* B *)
    Velocity     : LREAL; (* E *)
    Acceleration : LREAL; (* E *)
    Deceleration : LREAL; (* E *)
    Jerk         : LREAL; (* E *)
    Direction    : MC_Direction := MC_Positive_Direction; (* E *)
    BufferMode   : MC_BufferMode; (* E *)
    Options      : ST_MoveOptions; (* V *)
END_VAR
```

Name	Typ	Beschreibung
Execute	BOOL	Mit einer steigenden Flanke wird das Kommando ausgeführt.
Velocity	LREAL	Maximale Geschwindigkeit mit der gefahren werden soll (>0).
Acceleration	LREAL	Beschleunigung (≥0) Bei einem Wert von 0 wirkt die Standardbeschleunigung aus der Achskonfiguration im System Manager.
Deceleration	LREAL	Verzögerung (≥0) Bei einem Wert von 0 wirkt die Standardverzögerung aus der Achskonfiguration im System Manager.
Jerk	LREAL	Ruck (≥0) Bei einem Wert von 0 wirkt der Standard-Ruck aus der Achskonfiguration im System Manager.
Direction	MC_Direction [▶ 142]	Positive oder negative Fahrtrichtung.
BufferMode	MC_BufferMode [▶ 138]	Wird ausgewertet, wenn die Achse bereits ein anderes Kommando ausführt. MC_MoveVelocity wird nach dem laufenden Kommando aktiv oder bricht dieses ab. Übergangsbedingungen vom laufenden zum nächsten Kommando werden ebenfalls durch den BufferMode festgelegt. Wird das Kommando auf eine gekoppelte Slave-Achse angewendet, ist nur der BufferMode

Name	Typ	Beschreibung
		„Aborting“ möglich. Um den BufferMode zu verwenden, ist immer ein zweiter Funktionsbaustein nötigt. Es ist nicht möglich, einen Move-Baustein mit neuen Parametern zu triggern, während er noch aktiv ist.
Options	ST_MoveOptions [▶ 142]	Datenstruktur, die zusätzliche selten benötigte Parameter enthält. Im Normalfall kann der Eingang offen bleiben.

Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 14]

Ein-/Ausgänge

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

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	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. Der InVelocity-Ausgang bleibt auf dem Wert TRUE, bis die Achsgeschwindigkeit von einem anderen Kommando geändert wird. Sobald eine Geschwindigkeitsabweichung festgestellt wird, signalisiert der Ausgang FALSE.
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. 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.

InVelocity und Override

Wenn der Override-Wert verändert wird, signalisiert der InVelocity-Ausgang den Wert FALSE. Gleichzeitig nimmt nach einer Geschwindigkeitsänderung der CommandAborted-Ausgang den Wert TRUE an und der Funktionsbaustein zeigt nicht mehr den Status „busy“ an.

InVelocity bei einer Geschwindigkeitsüberlagerung mit MC_MoveSuperImposed

Eine Geschwindigkeitsänderung durch den Funktionsbaustein MC_MoveSuperImposed ist erlaubt und führt auch nicht zu einem Abbruch der Aktivität des Funktionsbausteins. Obwohl sich die Geschwindigkeit verändert, bleibt der InVelocity-Ausgang TRUE, weil sich die Gesamtgeschwindigkeit aus der Grundgeschwindigkeit, die sich nicht verändert hat, und einer dieser Grundgeschwindigkeit überlagerten Geschwindigkeit (Superposition) zusammensetzt.

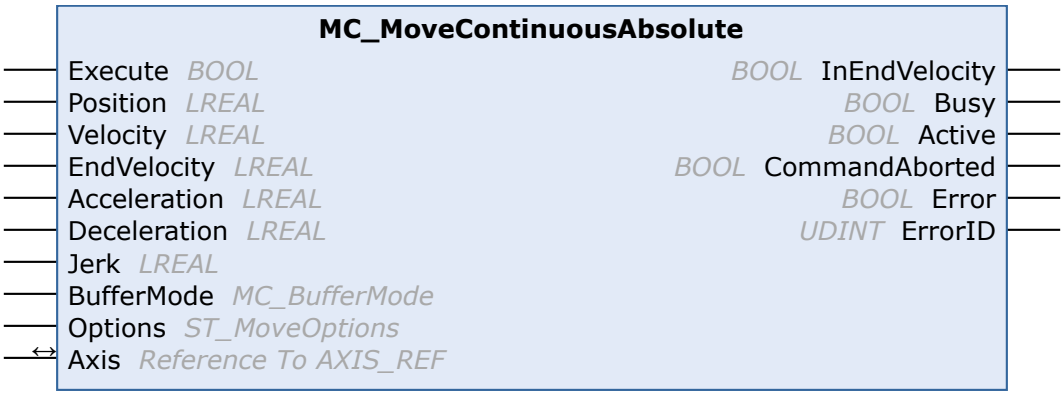
Der Funktionsbaustein bleibt „Busy“ und „Active“, bis das Kommando abgelöst wird.

Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 14]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

6.1.7 MC_MoveContinuousAbsolute



Mit dem Funktionsbaustein MC_MoveContinuousAbsolute wird eine Positionierung auf eine absolute Zielposition gestartet und die Achsbewegung über den gesamten Fahrweg überwacht. An der Zielposition wird eine konstante Endgeschwindigkeit erreicht, die beibehalten wird. Der Ausgang „InEndVelocity“ wird gesetzt, wenn die Zielposition angefahren wurde. Anderenfalls wird der Ausgang „CommandAborted“ oder im Fehlerfall der Ausgang „Error“ gesetzt.

Nachdem die Zielposition erreicht wurde, wird die Funktion des Bausteins abgeschlossen und die Achse nicht weiter überwacht.

Für Eil-/Schleichachsen ist MC_MoveContinuousAbsolute nicht implementiert.

Eingänge

```
VAR_INPUT
    Execute      : BOOL;
    Position     : LREAL;
    Velocity     : LREAL;
    EndVelocity  : LREAL;
    Acceleration : LREAL;
    Deceleration : LREAL;
    Jerk         : LREAL;
    BufferMode   : MC_BufferMode;
    Options      : ST_MoveOptions;
END_VAR
```

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

Name	Typ	Beschreibung
Position	LREAL	Absolute Zielposition
Velocity	LREAL	Maximale Geschwindigkeit, mit der die Zielposition angefahren werden soll (>0).
EndVelocity	LREAL	Endgeschwindigkeit, die nach Erreichen die Zielposition beibehalten werden soll.
Acceleration	LREAL	Beschleunigung (≥0) Bei einem Wert von 0 wirkt die Standardbeschleunigung aus der Achskonfiguration im System Manager.
Deceleration	LREAL	Verzögerung (≥0) Bei einem Wert von 0 wirkt die Standardverzögerung aus der Achskonfiguration im System Manager.
Jerk	LREAL	Ruck (≥0) Bei einem Wert von 0 wirkt der Standard-Ruck aus der Achskonfiguration im System Manager.
BufferMode	MC_BufferMode [▶ 138]	Wird ausgewertet, wenn die Achse bereits ein anderes Kommando ausführt. MC_MoveContinuousAbsolute wird nach dem laufenden Kommando aktiv oder bricht dieses ab. Übergangsbedingungen vom laufenden zum nächsten Kommando werden ebenfalls durch den BufferMode festgelegt. Um den BufferMode zu verwenden, ist immer ein zweiter Funktionsbaustein nötigt. Es ist nicht möglich, einen Move-Baustein mit neuen Parametern zu triggern, während er noch aktiv ist.
Options	ST_MoveOptions [▶ 142]	Datenstruktur, die zusätzliche selten benötigte Parameter enthält. Im Normalfall kann der Eingang offen bleiben.

Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 14]

Ein-/Ausgänge

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

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	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
    InEndVelocity  : BOOL;
    Busy           : BOOL;
    Active         : BOOL;
    CommandAborted : BOOL;
    Error          : BOOL;
    ErrorID        : UDINT;
END_VAR
```

Name	Typ	Beschreibung
InEndVelocity	BOOL	TRUE, wenn die Zielposition erreicht wurde. Der Funktionsbaustein bleibt „Busy“ und „Active“ bis das Kommando abgelöst wird.
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 „CommandAborted“ oder „Error“ gesetzt.

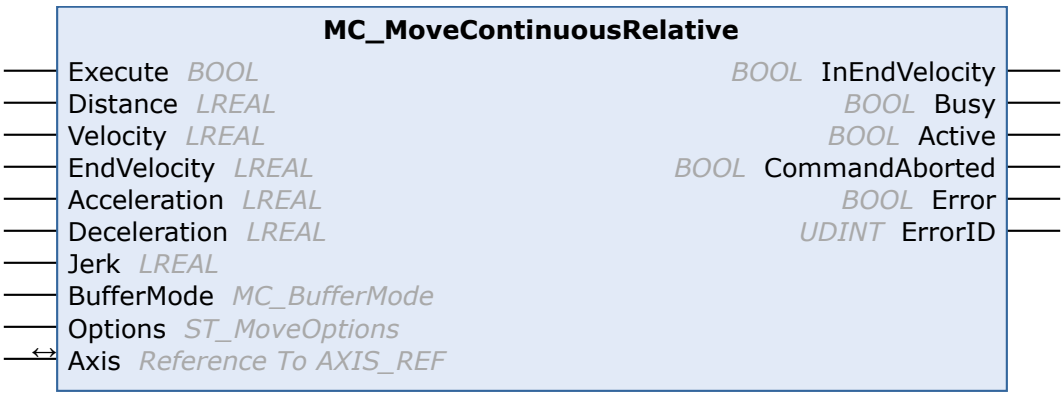
Name	Typ	Beschreibung
Active	BOOL	Zeigt an, dass das Kommando ausgeführt wird. Wenn das Kommando gepuffert wurde, wird es evtl. erst aktiv, nachdem ein laufendes Kommando beendet ist.
CommandAborted	BOOL	TRUE, wenn das Kommando nicht vollständig ausgeführt werden konnte. Die Achse wurde gestoppt oder 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 [▶ 14]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

6.1.8 MC_MoveContinuousRelative



Mit dem Funktionsbaustein MC_MoveContinuousRelative wird eine Positionierung um eine relative Distanz gestartet und die Achsbewegung über den gesamten Fahrweg überwacht. An der Zielposition wird eine konstante Endgeschwindigkeit erreicht, die beibehalten wird. Der Ausgang „InEndVelocity“ wird gesetzt, wenn die Zielposition angefahren wurde. Anderenfalls wird der Ausgang „CommandAborted“ oder im Fehlerfall der Ausgang „Error“ gesetzt.

Nachdem die Zielposition erreicht wurde, wird die Funktion des Bausteins abgeschlossen und die Achse wird nicht weiter überwacht.

Für Eil-/Schleichachsen ist MC_MoveContinuousRelative nicht implementiert.

Eingänge

```
VAR_INPUT
    Execute      : BOOL;
    Distance     : LREAL;
    Velocity     : LREAL;
    EndVelocity  : LREAL;
    Acceleration : LREAL;
    Deceleration : LREAL;
    Jerk         : LREAL;
    BufferMode   : MC_BufferMode;
    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 „Distance“ abgefahren werden soll (>0).
EndVelocity	LREAL	Endgeschwindigkeit, die nach Abfahren der relativen Strecke „Distance“ beibehalten werden soll.
Acceleration	LREAL	Beschleunigung (≥0) Bei einem Wert von 0 wirkt die Standardbeschleunigung aus der Achskonfiguration im System Manager.
Deceleration	LREAL	Verzögerung (≥0) Bei einem Wert von 0 wirkt die Standardverzögerung aus der Achskonfiguration im System Manager.
Jerk	LREAL	Ruck (≥0) Bei einem Wert von 0 wirkt der Standard-Ruck aus der Achskonfiguration im System Manager.
BufferMode	MC_BufferMode [▶ 138]	Wird ausgewertet, wenn die Achse bereits ein anderes Kommando ausführt. MC_MoveContinuousRelative wird nach dem laufenden Kommando aktive oder bricht dieses ab. Übergangsbedingungen vom laufenden zum nächsten Kommando werden ebenfalls durch den BufferMode festgelegt. Um den BufferMode zu verwenden, ist immer ein zweiter Funktionsbaustein nötigt. Es ist nicht möglich, einen Move-Baustein mit neuen Parametern zu triggern, während er noch aktiv ist.
Options	ST_MoveOptions [▶ 142]	Datenstruktur, die zusätzliche selten benötigte Parameter enthält. Im Normalfall kann der Eingang offen bleiben.

Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 14]

Ein-/Ausgänge

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

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	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
    InEndVelocity  : BOOL;
    Busy           : BOOL;
    Active         : BOOL;
    CommandAborted : BOOL;
    Error          : BOOL;
    ErrorID        : UDINT;
END_VAR
```

Name	Typ	Beschreibung
InEndVelocity	BOOL	TRUE, wenn die Zielposition erreicht wurde. Der Funktionsbaustein bleibt „Busy“ und „Active“ bis das Kommando abgelöst wird.

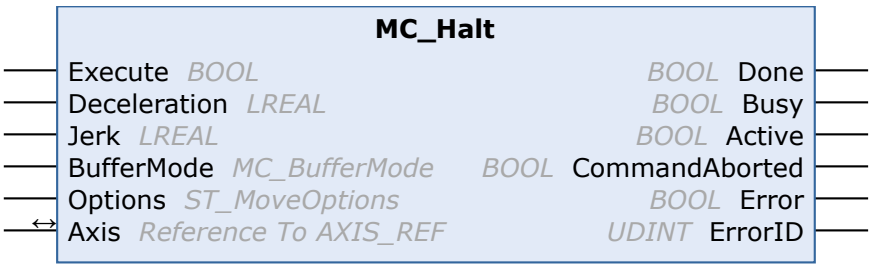
Name	Typ	Beschreibung
Busy	BOOL	TRUE, sobald das Kommando mit „Execute“ gestartet wird und solange der Fahrbefehl abgearbeitet wird. Wenn „Busy“ FALSE ist, ist der Funktionsbaustein bereit für einen neuen Auftrag. Gleichzeitig ist einer der Ausgänge „CommandAborted“ oder „Error“ gesetzt.
Active	BOOL	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 [▶ 14]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

6.1.9 MC_Halt



Mit dem Funktionsbaustein MC_Halt wird eine Achse mit einer definierten Bremsrampe angehalten.

Im Gegensatz zu MC_Stop [▶ 90] wird die Achse nicht gegen weitere Fahrbefehle verriegelt. Die Achse kann also sowohl während der Bremsrampe als auch nach dem Halt durch ein anderes Kommando gestartet werden.

Bewegungskommandos können auf gekoppelte Slave-Achsen angewendet werden, wenn diese Option in den Parametern der Achse explizit aktiviert worden ist. Ein Bewegungskommando wie MC_Halt führt dann automatisch zum Abkoppeln der Achse und das Kommando wird anschließend ausgeführt. In diesem Fall ist ausschließlich der BufferMode „Aborting“ möglich.

Eingänge

```
VAR_INPUT
    Execute      : BOOL;
    Deceleration : LREAL;
    Jerk         : LREAL;
    BufferMode   : MC_BufferMode;
    Options      : ST_MoveOptions;
END_VAR
```

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

Name	Typ	Beschreibung
Deceleration	LREAL	Verzögerung Bei einem Wert ≤ 0 wirkt die mit dem letzten Move-Kommando parametrierte Verzögerung. MC_Halt und auch MC_Stop [▶ 90] können aus Sicherheitsgründen nicht mit schwächerer Dynamik ausgeführt werden, als der gerade aktive Fahrauftrag. Die Parametrierung wird gegebenenfalls automatisch angepasst.
Jerk	LREAL	Ruck Bei einem Wert ≤ 0 wirkt der mit dem letzten Move-Kommando parametrierte Ruck. MC_Halt und auch MC_Stop [▶ 90] können aus Sicherheitsgründen nicht mit schwächerer Dynamik ausgeführt werden als der gerade aktive Fahrauftrag. Die Parametrierung wird gegebenenfalls automatisch angepasst.
BufferMode	MC_BufferMode [▶ 138]	Wird ausgewertet, wenn die Achse bereits ein anderes Kommando ausführt. MC_Halt wird nach dem laufenden Kommando aktiv oder bricht dieses ab. Übergangsbedingungen vom laufenden zum nächsten Kommando werden ebenfalls durch den BufferMode festgelegt. Wird das Kommando auf eine gekoppelte Slave-Achse angewendet, so ist nur der BufferMode „Aborting“ möglich. Besonderheiten bei MC_Halt: Der BufferMode „Buffered“ zeigt keinen Effekt, da das Kommando erst im Stillstand ausgeführt wird. Die Blending-Modes „MC_BlendingNext“ und „MC_BlendingLow“ verändern die letzte Zielposition nicht, können aber zu einer veränderten Dynamik (Deceleration) der Anhalterampe führen. Die Modes „MC_BlendingPrevious“ und „MC_BlendingHigh“ verlängern die Fahrt bis zur ursprünglichen Zielposition und leiten erst dort die Anhalterampe ein (definierter Bremspunkt).
Options	ST_MoveOptions [▶ 142]	Datenstruktur, die zusätzliche selten benötigte Parameter enthält. Im Normalfall kann der Eingang offen bleiben.

Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 14]

Ein-/Ausgänge

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

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	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
```

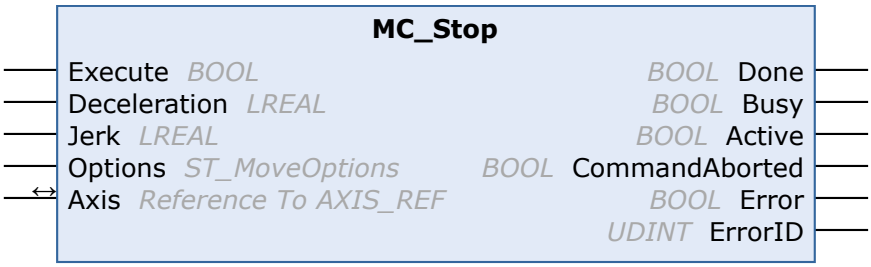
Name	Typ	Beschreibung
Done	BOOL	TRUE, wenn die Achse gestoppt wurde und im Stillstand ist.
Busy	BOOL	TRUE, sobald das Kommando mit „Execute“ gestartet wird und solange der Funktionsbaustein aktiv ist. Wenn „Busy“ FALSE ist, ist der Funktionsbaustein bereit für einen neuen Auftrag. Gleichzeitig ist einer der Ausgänge „CommandAborted“ oder „Error“ gesetzt.
Active	BOOL	Zeigt an, dass das Kommando ausgeführt wird. 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 [▶ 14]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

6.1.10 MC_Stop



Mit dem Funktionsbaustein MC_Stop wird eine Achse mit einer definierten Bremsrampe angehalten und die Achse gegen andere Bewegungskommandos verriegelt. 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 [▶ 19] wird die Verriegelung der Achse aufgehoben.

Alternativ kann die Achse mit MC_Halt [▶ 88] ohne Verriegelung angehalten werden. MC_Halt für normale Bewegungsabläufe bevorzugen.

Bewegungskommandos können auf gekoppelte Slave-Achsen angewendet werden, wenn diese Option in den Parametern der Achse explizit aktiviert worden ist. Ein Bewegungskommando wie MC_Stop führt dann automatisch zum Abkoppeln der Achse und das Kommando wird anschließend ausgeführt.

Eingänge

```
VAR_INPUT
    Execute      : BOOL;
    Deceleration : LREAL;
    Jerk         : 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.
Jerk	LREAL	Ruck Bei einem Wert ≤ 0 wirkt der mit dem letzten Move-Kommando parametrierte Ruck. 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 [▶ 142]	Datenstruktur, die zusätzliche, selten benötigte Parameter enthält. Im Normalfall kann der Eingang offen bleiben.

Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 14]

Ein-/Ausgänge

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

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	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

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

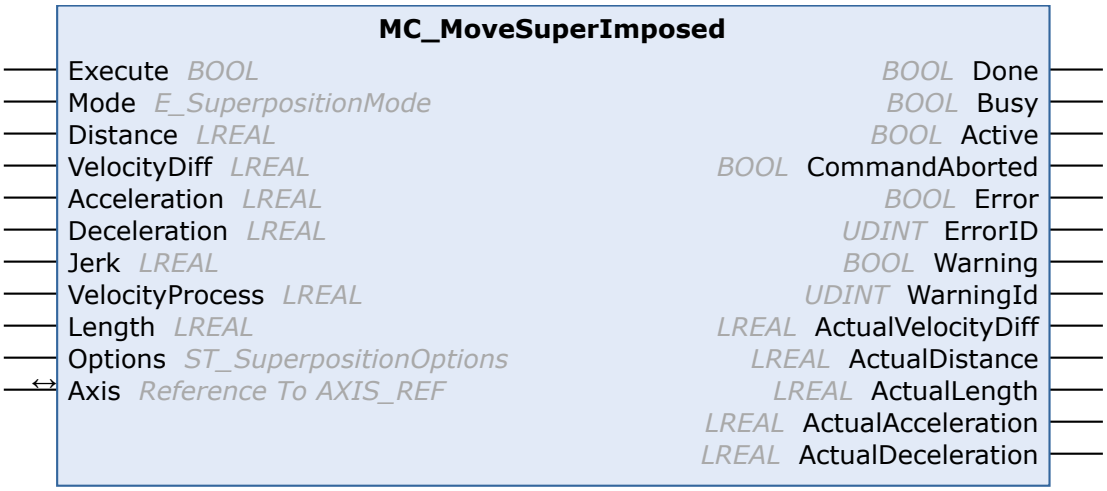
Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 14]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

6.2 Superposition

6.2.1 MC_MoveSuperImposed



Mit dem Funktionsbaustein MC_MoveSuperImposed wird eine relative überlagerte Bewegung gestartet, während eine Achse bereits in Bewegung ist. Die aktuelle Bewegung wird nicht unterbrochen. Der Ausgang „Done“ wird gesetzt, wenn die überlagerte Bewegung beendet ist. Die ursprüngliche unterlagerte Bewegung kann weiterhin aktiv sein und wird durch den zugehörigen Move-Funktionsbaustein überwacht.

Die Funktion der Überlagerung wird deutlich, wenn zwei gleich schnell laufende Achsen betrachtet werden. Wird eine der Achsen durch MC_MoveSuperImposed überlagert, eilt sie um den Parameter „Distance“ vor oder nach. Nachdem die überlagerte Bewegung beendet ist, bleibt die Streckendifferenz „Distance“ zwischen beiden Achsen erhalten.

MC_MoveSuperImposed kann sowohl auf Einzelachsen, als auch auf Master- oder Slaveachsen ausgeführt werden. Bei einer Slaveachse wirkt die überlagerte Bewegung allein auf die Slaveachse. Wird die Funktion auf eine Masterachse angewendet, so macht der Slave aufgrund der Achskopplung die überlagerte Bewegung des Masters mit.

Da MC_MoveSuperImposed eine relative überlagerte Bewegung ausführt, verändert sich auch die Zielposition des unterlagerten Fahrkommandos um Distance.

Die überlagerte Bewegung wird abhängig von der Position der Hauptbewegung ausgeführt. Das heißt, dass bei einer Geschwindigkeitsänderung der Hauptbewegung auch die überlagerte Bewegung entsprechend langsamer oder schneller wird und bei einem Stillstand der Hauptbewegung ist auch die überlagerte Bewegung nicht mehr aktiv. Über den Parameter „Options“ kann festgelegt werden, ob die überlagerte Bewegung nach einem Stillstand der Hauptbewegung abgebrochen oder fortgesetzt wird.

Siehe auch: Anwendungsbeispiele zu MC_MoveSuperImposed [▶ 95]

Eingänge

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

Name	Typ	Beschreibung
Execute	BOOL	Mit einer steigenden Flanke wird das Kommando ausgeführt.
Mode	E_SuperpositionMode	Legt fest, wie die überlagerte Bewegung auszuführen ist. (Typ: E_SuperpositionMode [▶ 157])
Distance	LREAL	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	LREAL	Maximale Geschwindigkeitsdifferenz zur aktuellen Geschwindigkeit (Grundgeschwindigkeit) der Achse (>0). Bei diesem Parameter ist eventuell abhängig von der Überlagerungsrichtung (Beschleunigen oder Verzögern) eine Fallunterscheidung notwendig. Ist beispielsweise eine Fahrtrichtungsumkehr nicht erlaubt, kann nur bis zur Maximalgeschwindigkeit beschleunigt oder bis zum Stopp abgebremst werden. Demnach gibt es zwei Fälle für den maximal möglichen Wert von „VelocityDiff“: • Distance > 0 (Achse beschleunigt) VelocityDiff = Maximalgeschwindigkeit – Grundgeschwindigkeit • Distance < 0 (Achse verzögert) VelocityDiff = Grundgeschwindigkeit
Acceleration	LREAL	Beschleunigung (≥0). Bei einem Wert von 0 wirkt die Standardbeschleunigung aus der Achskonfiguration im System Manager.
Deceleration	LREAL	Verzögerung (≥0). Bei einem Wert von 0 wirkt die Standardverzögerung aus der Achskonfiguration im System Manager.
Jerk	LREAL	Der Parameter Jerk ist nicht implementiert.
VelocityProcess	LREAL	Mittlere Prozessgeschwindigkeit in der Achse (>0). Bei konstanter Grundgeschwindigkeit während der Überlagerung kann hier die Sollgeschwindigkeit der Achse angegeben werden.
Length	LREAL	Fahrstrecke, die für die überlagerte Bewegung zur Verfügung steht. Der Parameter „Mode“ legt fest, wie diese Strecke interpretiert wird.

Name	Typ	Beschreibung
Options	ST_SuperpositionOptions [▶ 159]	Datenstruktur, die zusätzliche selten benötigte Parameter enthält. Im Normalfall kann der Eingang offen bleiben. • **AbortOption:** Legt das Verhalten im Stillstand der unterlagerten Bewegung fest. Die überlagerte Bewegung kann dann abgebrochen oder später fortgesetzt werden.

Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 14]

Ein-/Ausgänge

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

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	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;
    Warning            : BOOL;
    WarningID          : UDINT;
    ActualVelocityDiff : LREAL;
    ActualDistance     : LREAL;
    ActualLength       : LREAL;
    ActualAcceleration : LREAL;
    ActualDeceleration : LREAL;
END_VAR
```

Name	Typ	Beschreibung
Done	BOOL	TRUE, wenn die überlagerte Bewegung erfolgreich abgeschlossen 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.
CommandAborted	BOOL	TRUE, wenn das Kommando durch ein anderes Kommando abgebrochen wurde und daher nicht vollständig ausgeführt werden konnte.
Error	BOOL	TRUE, wenn ein Fehler auftritt.
ErrorID	UDINT	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.
Warning	BOOL	TRUE, wenn die Aktion nicht vollständig ausgeführt werden kann.
WarningID	UDINT	Der Baustein liefert die Warnung 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

Name	Typ	Beschreibung
		Anwender muss entscheiden, ob er diese Warnmeldung in seiner Applikation als echten Fehler oder eher als Warnung versteht.
ActualVelocityDiff	LREAL	Tatsächlich genutzte Geschwindigkeitsüberhöhung während der überlagerten Bewegung (ActualVelocityDiff ≤ VelocityDiff).
ActualDistance	LREAL	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	LREAL	Tatsächlich zurückgelegte Strecke während der überlagerten Bewegung (ActualLenght ≤ Length).
ActualAcceleration	LREAL	Tatsächlich verwendete Beschleunigung der überlagerten Bewegung (ActualAcceleration ≤ Acceleration).
ActualDeceleration	LREAL	Tatsächlich verwendete Verzögerung der überlagerten Bewegung (ActualDeceleration ≤ Deceleration).

Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 14]

Voraussetzungen

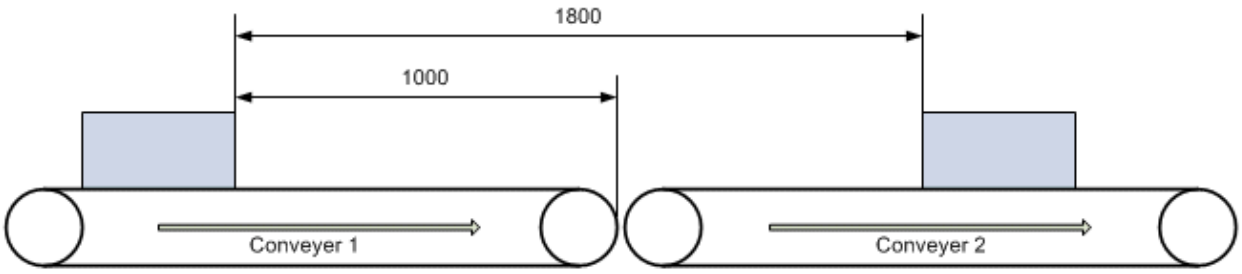
Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

6.2.2 Anwendungsbeispiele zu MC_MoveSuperImposed

Mit dem Funktionsbaustein MC_MoveSuperImposed [▶ 92] wird eine überlagerte Bewegung auf einer bereits fahrenden Achse gestartet. 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_MoveSuperImposed [▶ 92] 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

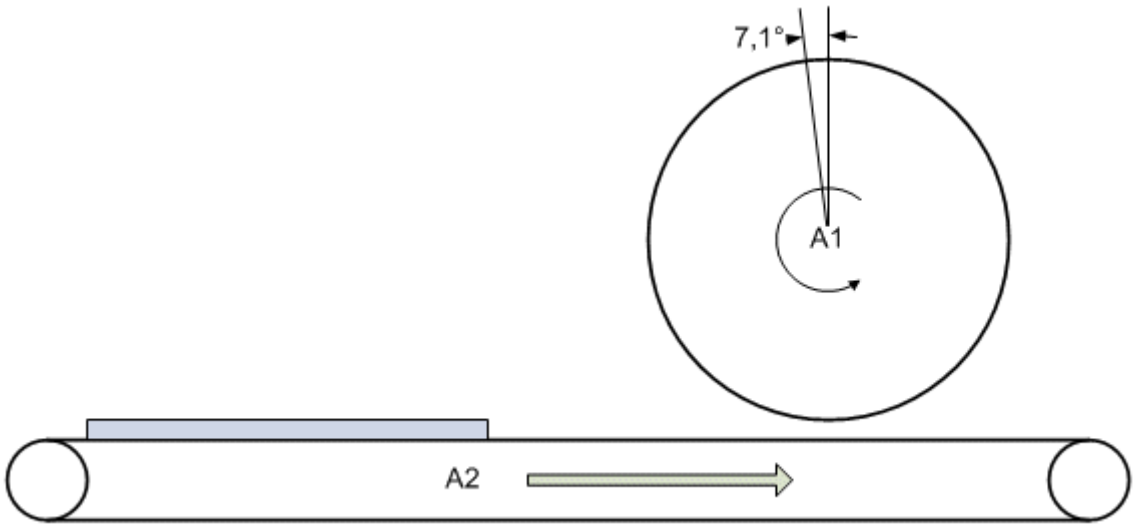
VelocityDiff = 500 mm/s

„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 kann 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 Band-Ende. 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.
- Eine Korrekturstrecke kann definiert werden, innerhalb der die Korrektur stattfinden kann, z. B. eine volle Walzenumdrehung. Daraus ergeben sich folgende Möglichkeiten der Parametrierung des Funktionsbausteins MC_MoveSuperImposed [▶ 92].

Schnelle Korrektur:

Distance = 7,1°

Length = 360° (maximal mögliche Korrekturstrecke)

Mode = SUPERPOSITIONMODE_LENGTHREDUCTION_LIMITEDMOTION

VelocityDiff = 30°/s (Geschwindigkeitsreserve)

„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.

Langsame Korrektur:

Distance = 7,1°

Length = 360° (Korrekturstrecke)

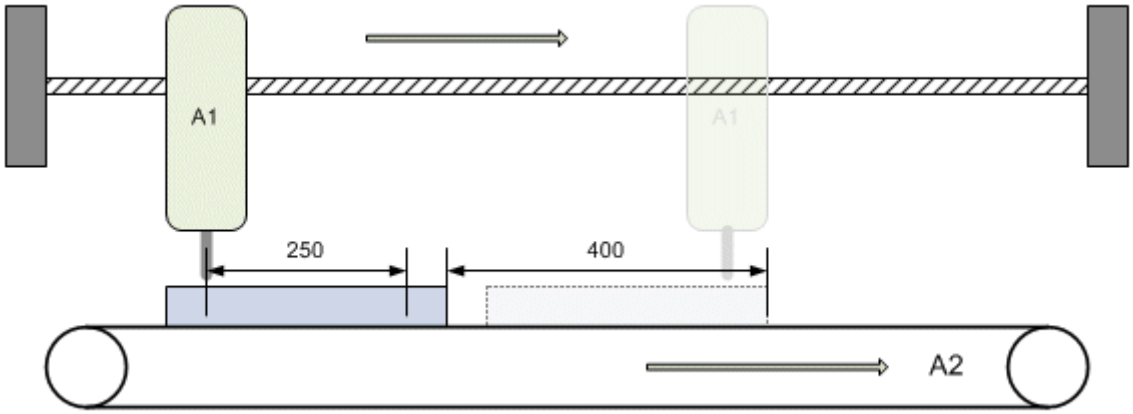
Mode = SUPERPOSITIONMODE_VELOREDUCTION_LIMITEDMOTION

VelocityDiff = 30°/s (Geschwindigkeitsreserve)

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 wurde z. B. mit der fliegenden Säge durchgeführt (MC_GearInPos) und soll hier nicht betrachtet werden und. 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_MoveSuperImposed [▶ 92]:

Schnelle Korrektur:

Distance = 250 mm

Length = 400 mm

Mode = SUPERPOSITIONMODE_LENGTHREDUCTION_ADDITIVEMOTION

VelocityDiff = 500 mm/s (Geschwindigkeitsreserve des Bohraggregats)

„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.

Langsame Korrektur:

Distance = 250 mm

Length = 400 mm

Mode = SUPERPOSITIONMODE_VELOREDUCTION_ADDITIVEMOTION

VelocityDiff = 500 mm/s (Geschwindigkeitsreserve des Bohraggregats)

„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).

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

6.2.3 MC_AbortSuperposition

Mit dem Funktionsbaustein MC_AbortSuperposition wird eine durch MC_MoveSuperImposed [▶ 92] gestartete überlagerte Bewegung abgebrochen, ohne die unterlagerte Achsbewegung zu stoppen.

Ein vollständiger Achsstopp kann gegebenenfalls mit MC_Stop [▶ 90] oder MC_Halt [▶ 88] durchgeführt werden. Ein Aufruf von MC_AbortSuperposition ist dann nicht notwendig.

Eingänge

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

Name	Typ	Beschreibung
Execute	BOOL	Mit einer steigenden Flanke wird das Kommando ausgeführt und die überlagerte Bewegung beendet.

Ein-/Ausgänge

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

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	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
```

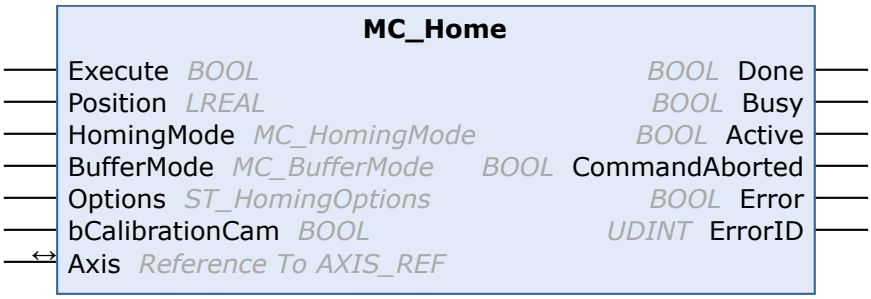
Name	Typ	Beschreibung
Done	BOOL	TRUE, wenn die überlagerte Bewegung erfolgreich abgeschlossen wurde.
Busy	BOOL	TRUE, sobald der Baustein aktiv ist. FALSE, wenn er sich wieder im Grundzustand befindet.
Error	BOOL	TRUE, sobald ein Fehler auftritt.
ErrorID	UDINT	Liefert bei einem gesetzten Error-Ausgang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

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

6.3 Homing

6.3.1 MC_Home



Mit dem Funktionsbaustein MC_Home wird eine Referenzierfahrt der Achse durchgeführt.

Der Referenziermodus wird im TwinCAT System Manager mit dem Encoderparameter „Reference Mode“ eingestellt. Abhängig vom angeschlossenen Encoder-System sind verschiedene Abläufe möglich (siehe auch Referenziermodus für Inkrementalencoder in der Dokumentation TwinCAT 3 ADS Interface NC).

Eingänge

```
VAR_INPUT
    Execute         : BOOL;
    Position        : LREAL         := DEFAULT_HOME_POSITION;
    HomingMode      : MC_HomingMode;
    BufferMode      : MC_BufferMode;
    Options         : ST_HomingOptions;
    bCalibrationCam : BOOL;
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 [▶ 136]	Bestimmt, auf welche Weise die Kalibrierung durchgeführt wird. • MC_DefaultHoming Führt die Standard-Referenzierfahrt aus.

Name	Typ	Beschreibung
		• MC_Direct Setzt die Position der Achse direkt auf Position ohne eine Bewegung auszuführen. • 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.
BufferMode	MC_BufferMode	Zurzeit nicht implementiert.
Options	ST_HomingOptions [▶ 136]	Datenstruktur, die zusätzliche selten benötigte Parameter. Im Normalfall kann der Eingang offen bleiben. • **ClearPositionLag:** Wirkt nur im Mode „MC_Direct“. Kann optional gesetzt werden, falls Soll- und Istposition auf den gleichen Wert gesetzt werden sollen. Damit wird der Schleppfehler gelöscht.
bCalibrationCam	BOOL	Spiegelt das Signal einer Referenznocke wieder, das über einen digitalen Eingang in die Steuerung kommen kann.

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.

Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 14]

Ein-/Ausgänge

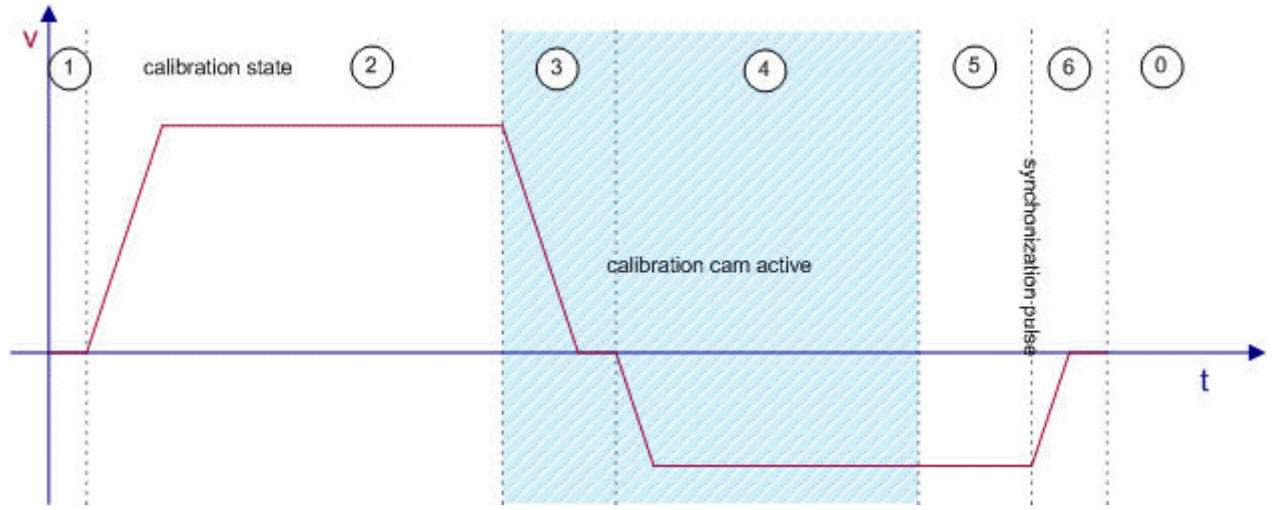
```
VAR_IN_OUT
    Axis      : AXIS_REF;
END_VAR
```

Name	Typ	Beschreibung
Axis	AXIS_REF	Achsdatenstruktur vom Typ AXIS_REF [▶ 124], welche eine Achse eindeutig im System adressiert. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.

Referenziervorgang

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

Soll eine Achse ohne Referenznocke, also nur auf den Sync-Impuls des Gebers, referenziert werden, kann die Referenznocke durch das SPS-Programm simuliert werden. Das Signal „bCalibrationCam“ wird zunächst aktiviert und dann zurückgenommen, wenn Axis.NcToPlc.HomingState [▶ 125] größer oder gleich 4 ist.



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.
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, sobald ein Fehler auftritt.
ErrorID	UDINT	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

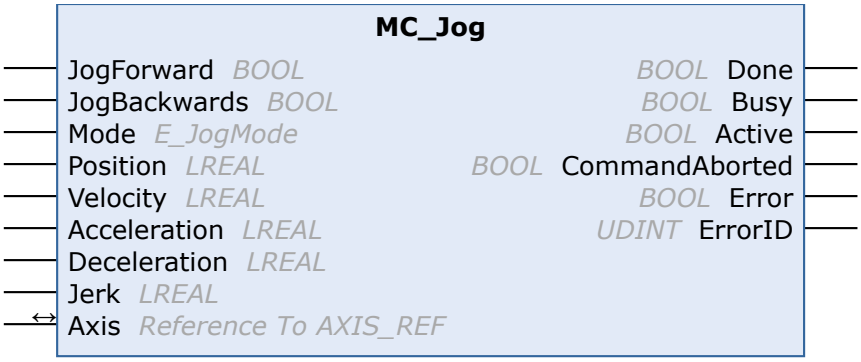
Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 14]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

6.4 Manual Motion

6.4.1 MC_Jog



Mit dem Funktionsbaustein MC_Jog wird es ermöglicht, eine Achse mit Handbedientasten zu fahren. Das Tastensignal kann direkt mit den beiden Eingängen „JogForward“ und „JogBackwards“ verbunden werden. Über den Eingang „Mode“ 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.

Eingänge

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

Name	Typ	Beschreibung
JogForward	BOOL	Mit einer steigenden Flanke wird das Kommando ausgeführt und die Achse in positiver Fahrtrichtung bewegt. Je nach Betriebsart (siehe Eingang „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	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.
Mode	E_JogMode [▶ 137]	Legt die Betriebsart fest, in der die Handfunktion ausgeführt wird. • MC_JOGMODE_STANDARD_SLOW Die Achse wird so lange verfahren, wie das Signal an einem der Jog-Eingänge TRUE ist. Dabei wird die im TwinCAT System Manager festgelegte „niedrige Geschwindigkeit für Handfunktionen“ und die Standarddynamik verwendet. In dieser Betriebsart

Name	Typ	Beschreibung
		haben die am Funktionsbaustein angelegten Positions-, Geschwindigkeits- und Dynamikdaten keine Bedeutung. • MC_JOGMODE_STANDARD_FAST Die Achse wird so lange verfahren, wie das Signal an einem der Jog-Eingänge TRUE ist. Dabei wird die im TwinCAT System Manager 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 so lange 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 einer steigenden Flanke an einem der Jog-Eingänge um eine bestimmte Distanz verfahren, die über den Eingang „Position“ 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. • MC_JOGMODE_INCHING_MODULO Die Achse wird mit einer steigenden Flanke an einem der Jog-Eingänge um eine bestimmte Distanz verfahren, die über den Eingang „Positions“ festgelegt wird. Die Achsposition rastet dabei auf ein ganzzahliges Vielfaches des Positionsparameters ein. 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	LREAL	Relative Distanz, um die in der Betriebsart MC_JOGMODE_INCHING verfahren wird.
Velocity	LREAL	Maximale Geschwindigkeit, mit der gefahren werden soll (>0).
Acceleration	LREAL	Beschleunigung (≥0). Bei einem Wert von 0 wirkt die Standardbeschleunigung aus der Achskonfiguration im System Manager.
Deceleration	LREAL	Verzögerung (≥0). Bei einem Wert von 0 wirkt die Standardverzögerung aus der Achskonfiguration im System Manager.
Jerk	LREAL	Ruck (≥0). Bei einem Wert von 0 wirkt der Standard-Ruck aus der Achskonfiguration im System Manager.

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.

Ein-/Ausgänge

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

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	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
```

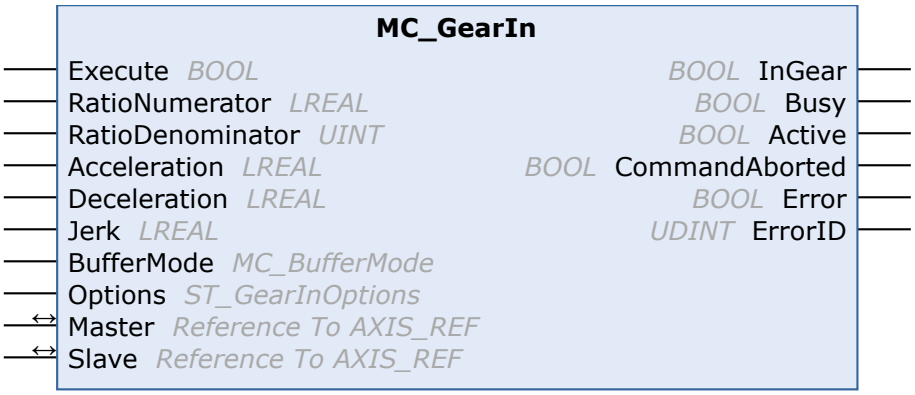
Name	Typ	Beschreibung
Done	BOOL	TRUE, wenn eine Bewegung erfolgreich abgeschlossen wurde.
Busy	BOOL	TRUE, sobald der Baustein aktiv ist. FALSE, wenn er sich im Grundzustand befindet. Erst dann kann eine weitere Flanke an den Jog-Eingängen angenommen werden.
Active	BOOL	Zeigt an, dass die Achse durch die Jog-Funktion bewegt wird.
CommandAborted	BOOL	TRUE, wenn der Vorgang von außen, z. B. durch den Aufruf von MC_Stop [▶ 90], abgebrochen wurde.
Error	BOOL	TRUE, wenn ein Fehler auftritt.
ErrorID	UDINT	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

6.5 Achskopplung

6.5.1 MC_GearIn



Mit dem Funktionsbaustein MC_GearIn wird eine lineare Master-Slave-Kopplung (Getriebekopplung) aktiviert. Der Baustein akzeptiert einen festen Getriebefaktor im Zähler-Nenner-Format.

Die Slaveachse kann im Stillstand an die Masterachse gekoppelt werden. Es ist mit diesem Baustein nicht möglich, auf eine fahrende Masterachse aufzusynchronisieren. Zu diesem Zweck können die Fliegende-Säge-Bausteine MC_GearInVelo oder MC_GearInPos verwendet werden.

Die Slaveachse kann mit dem Funktionsbaustein MC_GearOut [▶ 108] abgekoppelt werden. Wird der Slave während der Fahrt abgekoppelt, behält er seine Geschwindigkeit bei und kann mit MC_Stop [▶ 90] oder MC_Halt [▶ 88] angehalten werden.

Alternativ steht der Funktionsbaustein MC_GearInDyn [▶ 106] mit dynamisch änderbarem Getriebefaktor zur Verfügung.

Eingänge

```
VAR_INPUT
    Execute          : BOOL;
    RatioNumerator   : LREAL;
    RatioDenominator : UINT;
    Acceleration     : LREAL;
    Deceleration     : LREAL;
    Jerk             : LREAL;
    BufferMode       : MC_BufferMode;
    Options          : ST_GearInOptions;
END_VAR
```

Name	Typ	Beschreibung
Execute	BOOL	Mit einer steigenden Flanke wird das Kommando ausgeführt.
RatioNumerator	LREAL	Getriebefaktor Zähler. Alternativ kann der Getriebefaktor im Zähler als Fließkommawert angegeben werden, wenn der Nenner 1 ist.
RatioDenominator	UINT	Getriebefaktor Nenner
Acceleration	LREAL	Beschleunigung (≥0). (Zurzeit nicht implementiert)
Deceleration	LREAL	Verzögerung (≥0). (Zurzeit nicht implementiert)
Jerk	LREAL	Ruck (≥0). (Zurzeit nicht implementiert)
BufferMode	MC_BufferMode	Zurzeit nicht implementiert
Options	ST_GearInOptions	Zurzeit nicht implementiert

Bei einem Verhältnis 1:4 muss der RatioNumerator 1 sein und der RatioDenominator 4. Alternativ kann der RatioDenominator 1 sein und der Getriebefaktor wird im RatioNumerator als Fließkommazahl 0.25 angegeben. Der RatioNumerator darf negativ sein.

Ein-/Ausgänge

```
VAR_IN_OUT
    Master : AXIS_REF;
    Slave  : AXIS_REF;
END_VAR
```

Name	Typ	Beschreibung
Master	AXIS_REF	Achsdatenstruktur des Masters
Slave	AXIS_REF	Achsdatenstruktur des Slaves

Die Achsdatenstruktur vom Typ AXIS_REF [▶ 124] adressiert eine Achse eindeutig im System. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.

Ausgänge

```
VAR_OUTPUT
    InGear         : BOOL;
    Busy           : BOOL;
    Active         : BOOL;
    CommandAborted : BOOL;
```

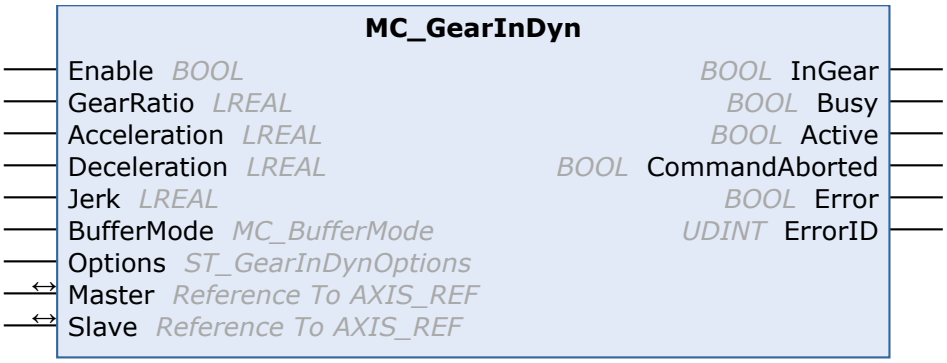
```
    Error          : BOOL;
    ErrorID        : UDINT;
END_VAR
```

Name	Typ	Beschreibung
InGear	BOOL	TRUE, wenn die Kopplung erfolgreich durchgeführt 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 „InGear“, „CommandAborted“ oder „Error“ gesetzt.
Active	BOOL	Zeigt an, dass das Kommando ausgeführt wird. (Zurzeit ist Active = Busy)
CommandAborted	BOOL	TRUE, wenn das Kommando nicht vollständig ausgeführt werden konnte. Die Achse kann während des Koppelvorgangs entkoppelt worden sein (gleichzeitige Kommandoausführung).
Error	BOOL	TRUE, wenn ein Fehler auftritt.
ErrorID	UDINT	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

6.5.2 MC_GearInDyn



Mit dem Funktionsbaustein MC_GearInDyn wird eine 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.

Die Slaveachse kann mit dem Funktionsbaustein MC_GearOut [▶ 108] abgekoppelt werden. Wird der Slave während der Fahrt abgekoppelt, behält er seine Geschwindigkeit bei und kann mit MC_Stop [▶ 90] oder MC_Halt [▶ 88] angehalten werden.

Alternativ steht der Baustein MC_GearIn [▶ 104] mit festem Getriebefaktor zur Verfügung.

Eingänge

```
VAR_INPUT
    Enable       : BOOL;
    GearRatio    : LREAL;
    Acceleration : LREAL;
    Deceleration : LREAL;
    Jerk         : LREAL;
```

```
    BufferMode   : MC_BufferMode;
    Options      : ST_GearInDynOptions;
END_VAR
```

Name	Typ	Beschreibung
Enable	BOOL	Ist „Enable“ TRUE, stellt der Baustein eine Kopplung her. Wird ein MC_GearOut ausgeführt, während der MC_GearInDyn weiterhin mit „Enable“ TRUE aufgerufen wird, wird nur kurzzeitig abgekoppelt und danach direkt versucht, die Kopplung wieder herzustellen. Solange „Enable“ TRUE ist, kann der Getriebefaktor zyklisch geändert werden. Wenn „Enable“ nach der Kopplung FALSE wird, wird das Kommando beendet. Der Getriebefaktor wird auf seinem letzten Wert eingefroren, aber der Slave wird nicht entkoppelt.
GearRatio	LREAL	Getriebefaktor als Fließkommawert. Der Getriebefaktor kann zyklisch geändert werden, solange „Enable“ TRUE ist. Wenn „Enable“ FALSE ist, bleibt der Getriebefaktor unverändert.
Acceleration	LREAL	Beschleunigung (≥0). Bei einem Wert von 0 wirkt die Standardbeschleunigung aus der Achskonfiguration im System Manager. 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	LREAL	Verzögerung (≥0). (Nicht implementiert)
Jerk	LREAL	Ruck (≥0). (Nicht implementiert)
BufferMode	MC_BufferMode	Zurzeit nicht implementiert
Options	ST_GearInDynOptions	Zurzeit nicht implementiert

Ein-/Ausgänge

```
VAR_IN_OUT
    Master : AXIS_REF;
    Slave  : AXIS_REF;
END_VAR
```

Name	Typ	Beschreibung
Master	AXIS_REF	Achsdatenstruktur des Masters
Slave	AXIS_REF	Achsdatenstruktur des Slaves

Die Achsdatenstruktur vom Typ AXIS_REF [▶ 124] adressiert eine Achse eindeutig im System. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.

Ausgänge

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

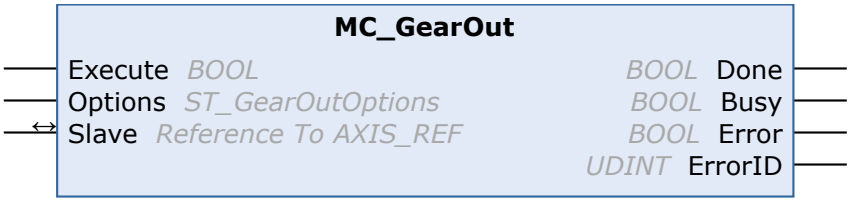
Name	Typ	Beschreibung
InGear	BOOL	TRUE, wenn die Kopplung erfolgreich durchgeführt wurde.

Name	Typ	Beschreibung
Busy	BOOL	TRUE, sobald das Kommando 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 „InGear“, „CommandAborted“ oder „Error“ gesetzt.
Active	BOOL	Zeigt an, dass das Kommando ausgeführt wird. (Zurzeit ist Active = Busy)
CommandAborted	BOOL	TRUE, wenn das Kommando nicht vollständig ausgeführt werden konnte. Die Achse kann während des Koppelvorgangs entkoppelt worden sein (gleichzeitige Kommandoausführung).
Error	BOOL	TRUE, wenn ein Fehler auftritt.
ErrorID	UDINT	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

6.5.3 MC_GearOut



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

⚠ WARNUNG

Kein Stillstand der Achse durch Abkopplung

Wenn eine Slaveachse in der Bewegung abgekoppelt wird, wird sie nicht automatisch gestoppt, sondern erreicht eine konstante Geschwindigkeit mit der sie endlos weiterfährt.

Sie können die Achse mit den Bausteinen MC_Halt [▶ 88] oder MC_Stop [▶ 90] anhalten.

Sollwertgeneratortyp

Wenn der Sollwertgeneratortyp 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.

Eingänge

```
VAR_INPUT
    Execute : BOOL;
    Options : ST_GearOutOptions;
END_VAR
```

Name	Typ	Beschreibung
Execute	BOOL	Mit einer steigenden Flanke wird das Kommando ausgeführt.
Options	ST_GearOutOptions	Zurzeit nicht implementiert

Ein-/Ausgänge

```
VAR_IN_OUT
    Slave : AXIS_REF;
END_VAR
```

Name	Typ	Beschreibung
Slave	AXIS_REF	Achsdatenstruktur des Slaves

Die Achsdatenstruktur vom Typ AXIS_REF [▶ 124] adressiert eine Achse eindeutig im System. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.

Ausgänge

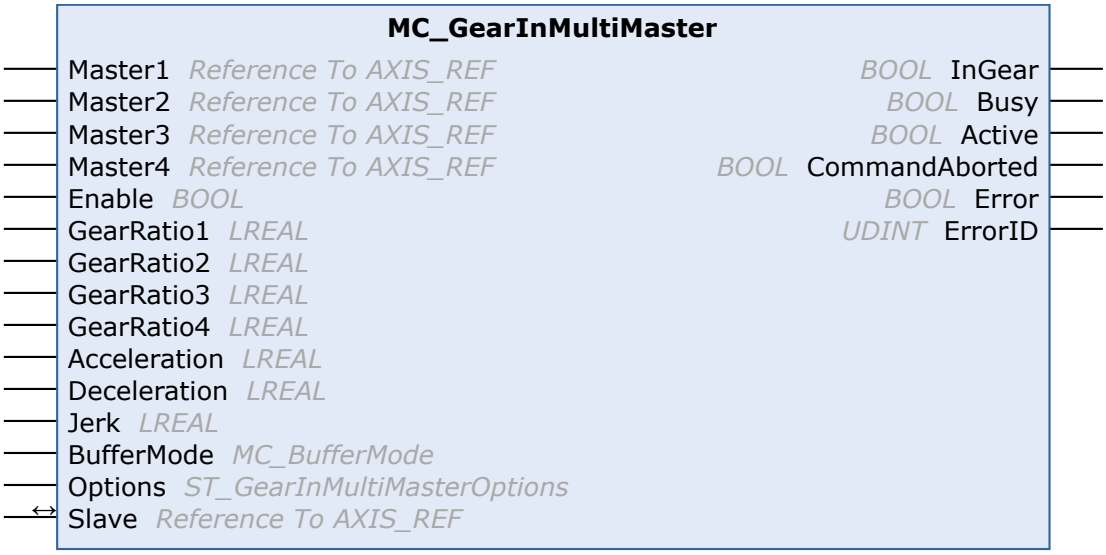
```
VAR_OUTPUT
    Done    : BOOL;
    Busy    : BOOL;
    Error   : BOOL;
    ErrorID : UDINT;
END_VAR
```

Name	Typ	Beschreibung
Done	BOOL	TRUE, wenn die Achse erfolgreich abgekoppelt wurde.
Busy	BOOL	TRUE, sobald das Kommando 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.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

6.5.4 MC_GearInMultiMaster



Mit dem Funktionsbaustein MC_GearInMultiMaster wird eine lineare Master-Slave-Kopplung (Getriebekopplung) zu bis zu vier verschiedenen Masterachsen aktiviert. Der Getriebefaktor kann jeweils dynamisch, d. h. in jedem SPS-Zyklus angepasst werden. Die Slave-Bewegung ergibt sich aus der Überlagerung der Bewegungen der Master. Der Parameter „Acceleration“ wirkt begrenzend, falls die Änderungen des Getriebefaktors sehr groß sind.

Die Slaveachse kann mit dem Funktionsbaustein MC_GearOut [▶ 108] abgekoppelt werden. Wird der Slave während der Fahrt abgekoppelt, behält er seine Geschwindigkeit bei und kann mit MC_Stop [▶ 90] angehalten werden.

Wenn weniger als vier Master verwendet werden, kann für die Parameter „Master2“ bis „Master4“ jeweils eine leere Datenstruktur übergeben werden (die Achs-ID muss 0 sein).

Eingänge

```
VAR_INPUT
    Master1      : Reference To AXIS_REF;
    Master2      : Reference To AXIS_REF;
    Master3      : Reference To AXIS_REF;
    Master4      : Reference To AXIS_REF;
    Enable       : BOOL;
    GearRatio1   : LREAL;
    GearRatio2   : LREAL;
    GearRatio3   : LREAL;
    GearRatio4   : LREAL;
    Acceleration : LREAL;
    Deceleration : LREAL;
    Jerk         : LREAL;
    BufferMode   : MC_BufferMode;
    Options      : ST_GearInMultiMasterOptions;
END_VAR
```

Name	Typ	Beschreibung
Master1	Reference To AXIS_REF [▶ 124]	Achsdatenstruktur des ersten Masters
Master2	Reference To AXIS_REF [▶ 124]	Achsdatenstruktur des zweiten Masters
Master3	Reference To AXIS_REF [▶ 124]	Achsdatenstruktur des dritten Masters
Master4	Reference To AXIS_REF [▶ 124]	Achsdatenstruktur des vierten Masters
Enable	BOOL	Ist „Enable“ TRUE, stellt der Baustein eine Kopplung her.

Name	Typ	Beschreibung
		Wird ein MC_GearOut ausgeführt, während der MC_GearInMultiMaster weiterhin mit „Enable“ TRUE aufgerufen wird, wird nur kurzzeitig abgekoppelt und danach direkt versucht, die Kopplung wieder herzustellen. Solange „Enable“ TRUE ist, können die Getriebefaktoren zyklisch geändert werden. Wenn „Enable“ nach der Kopplung FALSE wird, wird das Kommando beendet. Die Getriebefaktoren werden auf ihre letzten Werten eingefroren, aber der Slave wird nicht entkoppelt.
GearRatio1	LREAL	Getriebefaktor als Fließkommawert für die erste Masterachse. Der Getriebefaktor kann zyklisch geändert werden, solange „Enable“ TRUE ist. Wenn „Enable“ FALSE ist, bleibt der Getriebefaktor unverändert.
GearRatio2	LREAL	Getriebefaktor als Fließkommawert für die zweite Masterachse. Der Getriebefaktor kann zyklisch geändert werden, solange „Enable“ TRUE ist. Wenn „Enable“ FALSE ist, bleibt der Getriebefaktor unverändert.
GearRatio3	LREAL	Getriebefaktor als Fließkommawert für die dritte Masterachse. Der Getriebefaktor kann zyklisch geändert werden, solange „Enable“ TRUE ist. Wenn „Enable“ FALSE ist, bleibt der Getriebefaktor unverändert.
GearRatio4	LREAL	Getriebefaktor als Fließkommawert für die vierte Masterachse. Der Getriebefaktor kann zyklisch geändert werden, solange „Enable“ TRUE ist. Wenn „Enable“ FALSE ist, bleibt der Getriebefaktor unverändert.
Acceleration	LREAL	Beschleunigung (≥0). Bei einem Wert von 0 wirkt die Standardbeschleunigung aus der Achskonfiguration im System Manager. Der Parameter begrenzt die Beschleunigung des Slaves bei großen Änderungen des Getriebefaktors.
Deceleration	LREAL	Verzögerung (≥0). Bei einem Wert von 0 wirkt die Standardverzögerung aus der Achskonfiguration im System Manager. Der Parameter begrenzt die Verzögerung des Slaves bei großen Änderungen des Getriebefaktors. **Nur für die Option „AdvancedSlaveDynamics“ verwendet.**
Jerk	LREAL	Ruck (≥0). Bei einem Wert von 0 wirkt der Standardruck aus der Achskonfiguration im System Manager. Der Parameter begrenzt den Ruck des Slaves bei großen Änderungen des Getriebefaktors. **Nur für die Option „AdvancedSlaveDynamics“ verwendet.**
BufferMode	MC_BufferMode	Zurzeit nicht implementiert
Options	ST_GearInMultiMasterOptions [▶ 135]	• **AdvancedSlaveDynamics:** Tauscht den internen Algorithmus des Funktionsbausteins. Damit ist es möglich, auf bereits in Bewegung befindliche Master aufzusynchronisieren. Dabei sollten „Acceleration“ und „Deceleration“ ausschließlich symmetrisch parametriert werden. Bei zu großen Ruck-Vorgaben wird dieser soweit reduziert, dass eine Änderung von Null auf die parametrierte Beschleunigung/Verzögerung in einem NC-Zyklus erfolgen kann. Die Auflösung der Beschleunigung/Verzögerung hängt somit direkt von der geeigneten Parametrierung des Ruck-Wertes ab. • **SyncMode (verfügbar ab TC3.1.4024.11):** Der SyncMode legt fest, wie sich die Achskopplung verhält, falls die Bewegung der Slave-Achse im Modus

Name	Typ	Beschreibung
		AdvancedSlaveDynamics begrenzt wird (Begrenzung von Geschwindigkeit, Beschleunigung und Ruck). Folgende Einstellungen sind möglich: ◦ VELOSYNC: Die Slave-Achse fährt geschwindigkeitssynchron zu den Master-Achsen. Im Falle einer Dynamikbegrenzung baut der Slave eine Positionsdifferenz auf, welche nicht wieder aufgeholt wird. ◦ POSSYNC1: Die Slave-Achse fährt geschwindigkeits- und positionssynchron zu den Master-Achsen. Im Falle einer Dynamikbegrenzung baut der Slave eine Positionsdifferenz auf, welche aber zu einem späteren Zeitpunkt aufgeholt wird. Änderungen der Getriebefaktoren während der Fahrt werden unter Berücksichtigung der Dynamik ausgeführt. Die daraus resultierende Positionsdifferenz zwischen Master und Slave wird nicht herausgefahren. ◦ POSSYNC2: Die Slave-Achse fährt geschwindigkeits- und positionssynchron zu den Master-Achsen. Im Falle einer Dynamikbegrenzung baut der Slave eine Positionsdifferenz auf, welche aber zu einem späteren Zeitpunkt aufgeholt wird. Änderungen der Getriebefaktoren während der Fahrt werden sofort intern verrechnet und der Slave synchronisiert sich kontinuierlich auf die neue Position auf.

Übersicht: Welches Anwenderkommando bewirkt in Abhängigkeit vom SyncMode eine bleibende Positionsdifferenz bzw. holt diese Positionsdifferenz wieder auf.

		Anwenderkommandos		
		Ankoppeln während der Fahrt	Änderung Getriebefaktor	Dynamiklimitierung (Velo, Acc, Jerk)
SyncMode	VELOSYNC (default)	Positionsdifferenz bleibend	Positionsdifferenz bleibend	Positionsdifferenz bleibend
	POSSYNC1	Positionsdifferenz bleibend	Positionsdifferenz bleibend	Positionsdifferenz aufholen
	POSSYNC2	Positionsdifferenz aufholen	Positionsdifferenz aufholen	Positionsdifferenz aufholen

Wenn im Stillstand der Achsen mit einem konstanten Getriebefaktor gekoppelt wird (und der Getriebefaktor im Verlauf der Kopplung nicht mehr geändert wird), dann verhalten sich die beiden SyncModes POSSYNC1 und POSSYNC2 identisch.

Ein-/Ausgänge

```
VAR_IN_OUT
    Slave   :  AXIS_REF;
END_VAR
```

Name	Typ	Beschreibung
Slave	AXIS_REF	Achsdatenstruktur des Slaves

Die Achsdatenstruktur vom Typ AXIS_REF [▶ 124] adressiert eine Achse eindeutig im System. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.

Ausgänge

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

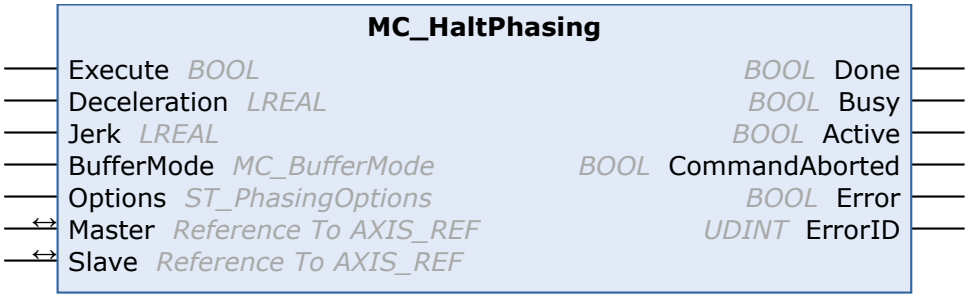
Name	Typ	Beschreibung
InGear	BOOL	TRUE, wenn die Kopplung erfolgreich durchgeführt wurde.
Busy	BOOL	TRUE, sobald das Kommando mit „Enable“ gestartet wird und solange der Befehl abgearbeitet wird. Wenn „Busy“ FALSE ist, ist der Funktionsbaustein bereit für einen neuen Auftrag. Gleichzeitig ist einer der Ausgänge „InGear“, „CommandAborted“ oder „Error“ gesetzt.
Active	BOOL	Zeigt an, dass das Kommando ausgeführt wird. (Zurzeit ist Active = Busy)
CommandAborted	BOOL	TRUE, wenn das Kommando nicht vollständig ausgeführt werden konnte. Die Achse kann während des Koppelvorgangs entkoppelt worden sein (gleichzeitige Kommandoausführung).
Error	BOOL	TRUE, wenn ein Fehler auftritt.
ErrorID	UDINT	Liefert bei einem gesetzten Error-Ausgang die Fehlernummer.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

6.6 Phasing

6.6.1 MC_HaltPhasing



Mit dem Funktionsbaustein MC_HaltPhasing wird ein gesteuerter Halt der Phasenverschiebung einer Slaveachse gegenüber der Masterachse herbeigeführt. Der „Halt“ wird immer ruckbegrenzt mit dem in „Jerk“ eingestellten konstanten Ruck für den Aufbau der Bremsverzögerung ausgeführt. MC_HaltPhasing bricht eine laufende überlagerte Bewegung durch MC_PhasingAbsolute oder MC_PhasingRelative ab.

Eingänge

```
VAR_INPUT
    Execute      : BOOL;
    Deceleration : LREAL;
    Jerk         : LREAL;
    BufferMode   : MC_BufferMode;
    Options      : ST_PhasingOptions;
END_VAR
```

Name	Typ	Beschreibung
Execute	BOOL	Mit einer steigenden Flanke wird das Kommando ausgeführt.
Deceleration	LREAL	Maximaler Verzögerungswert
Jerk	LREAL	Maximaler Ruckwert
BufferMode	MC_BufferMode	Nur MC_Aborting
Options	ST_PhasingOptions	Nicht implementiert

Ein-/Ausgänge

```
VAR_IN_OUT
    Master : AXIS_REF;
    Slave  : AXIS_REF;
END_VAR
```

Name	Typ	Beschreibung
Master	AXIS_REF	Achsdatenstruktur des Masters
Slave	AXIS_REF	Achsdatenstruktur des Slaves

Die Achsdatenstruktur vom Typ AXIS_REF [▶ 124] adressiert eine Achse eindeutig im System. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.

Ausgänge

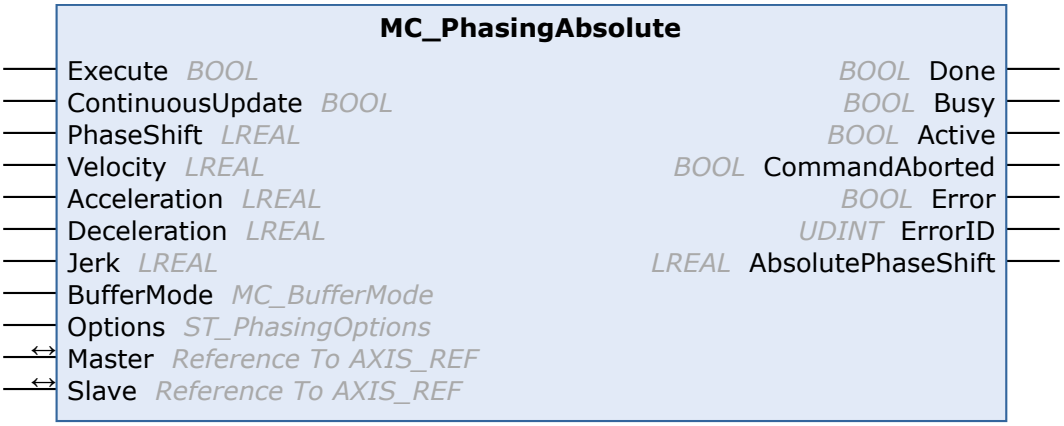
```
VAR_OUTPUT
    Done           :    BOOL;
    Busy           :    BOOL;
    Active         :    BOOL;
    CommandAborted :    BOOL;
    Error          :    BOOL;
    ErrorId        :    UDINT;
END_VAR
```

Name	Typ	Beschreibung
Done	BOOL	TRUE, wenn die Geschwindigkeit = 0 erreicht wird.
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. Gleichzeit 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.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

6.6.2 MC_PhasingAbsolute



Mit dem Funktionsbaustein MC_PhasingAbsolute kann eine Phasenverschiebung zwischen einer Master- und einer Slaveachse eingestellt werden. Der Funktionsbaustein führt eine überlagerte Bewegung der Slaveachse aus und stellt dadurch eine Positionsdifferenz „PhaseShift“ zwischen Master und Slave ein.

Die Dynamikwerte „Velocity“, „Acceleration“ und „Decelaration“ beziehen sich auf die überlagerte Bewegung, mit der die Phasenverschiebung durchgeführt wird. Die Bewegung wird immer ruckbegrenzt mit dem in „Jerk“ eingestellten konstanten Ruck ausgeführt. Der Wert gilt sowohl für die Beschleunigung mit „Acceleration“ als auch für das Bremsen mit „Deceleration“.

Die Phasenverschiebung kann sowohl für eine einfache MC_GearIn [▶ 104]-Kopplung, als auch für eine Kopplung mit dynamischem Koppelfaktor verwendet werden. Im letzteren Fall ist zu beachten, dass MC_GearInMultimaster [▶ 110] mit Options.AdvancedSlaveDynamics = TRUE (Gearing) oder MC_CamIn_V2 (Camming) verwendet wird.

MC_GearInDyn [▶ 106] wird nicht unterstützt.

Eingänge

```
VAR_INPUT
    Execute             : BOOL;
    ContinuousUpdate    : BOOL;
    PhaseShift          : LREAL;
    Velocity            : LREAL;
    Acceleration        : LREAL;
    Deceleration        : LREAL;
    Jerk                : LREAL;
    BufferMode          : MC_BufferMode;
    Options             : ST_PhasingOptions;
END_VAR
```

Name	Typ	Beschreibung
Execute	BOOL	Mit einer steigenden Flanke wird das Kommando ausgeführt.
ContinuousUpdate	BOOL	Ist dieser Eingang TRUE bei der steigenden Flanke am Eingang „Execute“, so können während der Abarbeitung des Kommandos die Eingänge „PhaseShift“, „Velocity“, „Acceleration“, „Deceleration“ und „Jerk“ verändert und schnellstmöglich zur Wirkung gebracht werden.
PhaseShift	LREAL	Einzustellende Phasenverschiebung zwischen Master- und Slaveachse
Velocity	LREAL	Wert der maximalen Geschwindigkeit, die bei der Phasenverschiebung erreicht werden darf (>=0.01).

Name	Typ	Beschreibung
Acceleration	LREAL	Wert der maximalen Beschleunigung
Deceleration	LREAL	Wert der maximalen Verzögerung
Jerk	LREAL	Wert des maximalen Rucks
BufferMode	MC_BufferMode	Nur MC_Aborting
Options	ST_PhasingOptions	Nicht implementiert

Ein-/Ausgänge

```
VAR_IN_OUT
    Master : AXIS_REF;
    Slave  : AXIS_REF;
END_VAR
```

Name	Typ	Beschreibung
Master	AXIS_REF	Achsdatenstruktur des Masters
Slave	AXIS_REF	Achsdatenstruktur des Slaves

Die Achsdatenstruktur vom Typ AXIS_REF [▶ 124] adressiert eine Achse eindeutig im System. Sie enthält unter anderem den aktuellen Status der Achse, wie Position, Geschwindigkeit oder Fehlerzustand.

Ausgänge

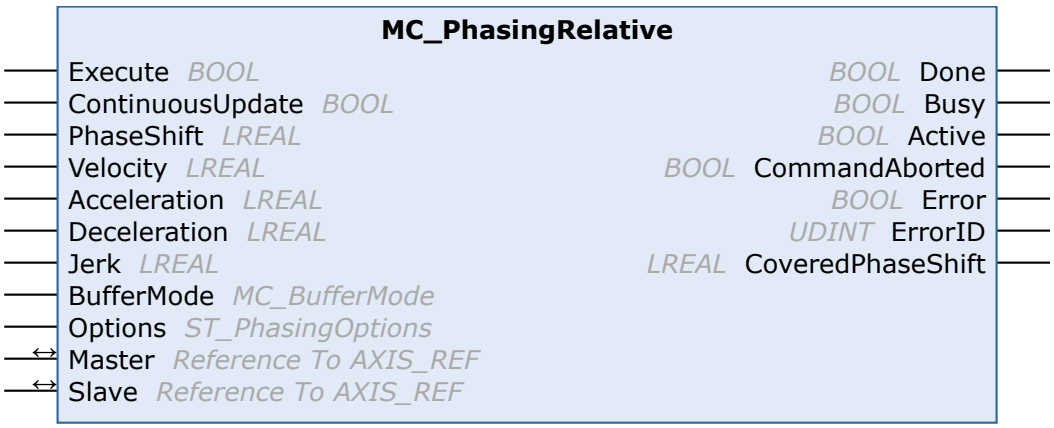
```
VAR_OUTPUT
    Done               : BOOL;
    Busy               : BOOL;
    Active             : BOOL;
    CommandAborted     : BOOL;
    Error              : BOOL;
    ErrorId            : UDINT;
    AbsolutePhaseShift : LREAL;
END_VAR
```

Name	Typ	Beschreibung
Done	BOOL	TRUE, wenn die absolute Phasenverschiebung hergestellt ist.
Busy	BOOL	TRUE, sobald das Kommando mit Execute gestartet wird und solange die Phasenverschiebung erfolgt. 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.
AbsolutePhaseShift	LREAL	Absolute Phasenverschiebung

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

6.6.3 MC_PhasingRelative



Mit dem Funktionsbaustein MC_PhasingRelative kann eine Phasenverschiebung zwischen einer Master- und einer Slaveachse eingestellt werden. Der Funktionsbaustein führt eine überlagerte Bewegung der Slaveachse aus und verändert dadurch die Positionsdifferenz zwischen Master und Slave um die Distanz „PhaseShift“.

Die Dynamikwerte „Velocity“, „Acceleration“ und „Decelaration“ beziehen sich auf die überlagerte Bewegung, mit der die Phasenverschiebung durchgeführt wird. Die Bewegung wird immer ruckbegrenzt mit dem in „Jerk“ eingestellten konstanten Ruck ausgeführt. Der Wert gilt sowohl für die Beschleunigung mit „Acceleration“ als auch für das Bremsen mit „Deceleration“.

Die Phasenverschiebung kann sowohl für eine einfache MC_GearIn [▶ 104]-Kopplung, als auch für eine Kopplung mit dynamischem Koppelfaktor verwendet werden. Im letzteren Fall ist zu beachten, dass MC_GearInMultimaster [▶ 110] mit Options.AdvancedSlaveDynamics = TRUE (Gearing) oder MC_CamIn_V2 (Camming) verwendet wird.

MC_GearInDyn [▶ 106] wird nicht unterstützt.

Eingänge

```
VAR_INPUT
    Execute            : BOOL;
    ContinuousUpdate   : BOOL;
    PhaseShift         : LREAL;
    Velocity           : LREAL;
    Acceleration       : LREAL;
    Deceleration       : LREAL;
    Jerk               : LREAL;
    BufferMode         : MC_BufferMode;
    Options            : ST_PhasingOptions;
END_VAR
```

Name	Typ	Beschreibung
Execute	BOOL	Mit einer steigenden Flanke wird das Kommando ausgeführt.
ContinuousUpdate	BOOL	Ist dieser Eingang TRUE bei der steigenden Flanke am Eingang Execute, so können während der Abarbeitung des Kommandos die Eingänge „PhaseShift“, „Velocity“, „Acceleration“, „Deceleration“ und „Jerk“ verändert und schnellstmöglich zur Wirkung gebracht werden.
PhaseShift	LREAL	Betrag, um den die Phasenverschiebung zwischen Master- und Slaveachse verändert wird.
Velocity	LREAL	Wert der maximalen Geschwindigkeit, die bei der Phasenverschiebung erreicht werden darf (>=0.01).
Acceleration	LREAL	Wert der maximalen Beschleunigung
Deceleration	LREAL	Wert der maximalen Verzögerung
Jerk	LREAL	Wert des maximalen Rucks
BufferMode	MC_BufferMode	Nur MC_Aborting

Name	Typ	Beschreibung
Options	ST_PhasingOptions	Nicht implementiert

Ein-/Ausgänge

```
VAR_IN_OUT
    Master : AXIS_REF;
    Slave  : AXIS_REF;
END_VAR
```

Name	Typ	Beschreibung
Master	AXIS_REF	Achsdatenstruktur des Masters
Slave	AXIS_REF	Achsdatenstruktur des Slaves

Die Achsdatenstruktur vom Typ AXIS_REF [▶ 124] adressiert eine Achse eindeutig im System. 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;
    CoveredPhaseShift : LREAL;
END_VAR
```

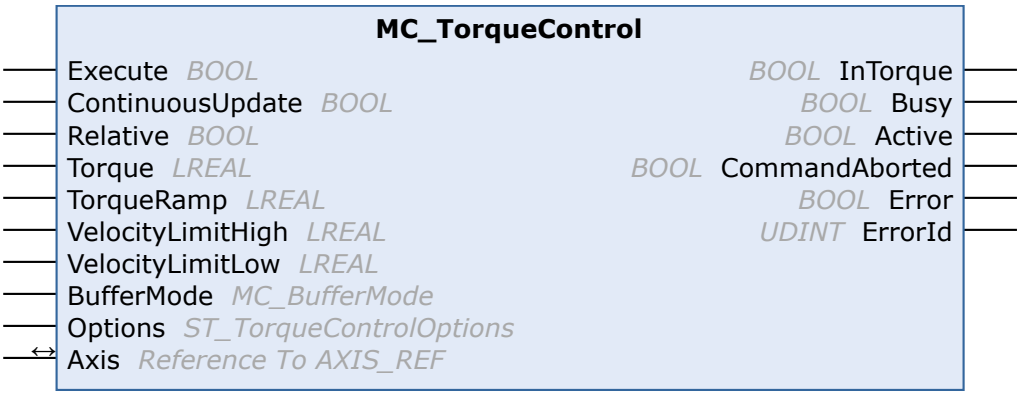
Name	Typ	Beschreibung
Done	BOOL	TRUE, wenn die absolute Phasenverschiebung hergestellt ist.
Busy	BOOL	TRUE, sobald das Kommando mit „Execute“ gestartet wird und solange die Phasenverschiebung erfolgt. Wenn „Busy“ FALSE ist, ist der Funktionsbaustein bereit für einen neuen Auftrag. Gleichzeit ist einer 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.
CoveredPhaseShift	LREAL	Absolute Phasenverschiebung

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

6.7 Torque Control

6.7.1 MC_TorqueControl



Mit dem Funktionsbaustein MC_TorqueControl wird eine Achse NC geführt in den „Cyclic Synchronous Torque Mode“ (CST) geschaltet und gibt dieser einen Torque-Sollwert vor. In dieser Betriebsart stellt die NC ebenfalls die parametrierten „VelocityLimitHigh“ & „VelocityLimitLow“ als verknüpfbare Objekte im zyklischen Interface zum Antriebsregler zur Verfügung (Axis->Drive->Outputs->nDataOut5/nDataOut6).

Für eine ruckfreie Umschaltung der Betriebsart, ist die Totzeitkompensation in der NC-Achse zu aktivieren.

Unterstützte Beckhoff Hardware		
AX5xxx	**AX8xxx / AMP8xxx / MD8xxx**	**Kompakte Antriebstechnik (Servo)**
✔ ab FW v2.14 b0001	✔ ab FW v1.05 b0001	✔ ab FW v01 (ELM72xx & AMI8xxx)

⚠ GEFAHR

Lebensgefahr oder Gefahr von schweren Verletzungen oder Sachschäden durch unbeabsichtigte Bewegungen der Achse

Bei Verwendung des Funktionsbausteins wird die Achse in den CST-Mode geschaltet. Nach Verwendung des Funktionsbausteins (insbesondere nach Fehlersituationen), kann es dazu kommen, dass sich die Achse weiterhin im CST-Mode befindet. Dieses kann beim Freigeben der Achse zu plötzlichen und ungeplanten Bewegungen (insbesondere bei Hubachsen) führen.

- Stellen Sie sicher, dass keine Gefahr im Sinne der Risikobewertung entsteht.
- Prüfen Sie die aktuelle Betriebsart über den Funktionsbaustein MC_ReadDriveOperationMode [▶ 46].
- Wenn sich die Achse nicht in einer positionsbezogenen Betriebsart (CSV/CSP) befindet, überführen Sie diese vor einer Freigabe:
  – *direkt* mit MC_WriteDriveOperationMode [▶ 47] in die gewünschte positionsbezogene Betriebsart (CSV/CSP) oder
  – *indirekt* mit MC_Halt [▶ 88] / MC_Stop [▶ 90] in die gewünschte positionsbezogene Betriebsart (CSV/CSP) (ab TwinCAT 3.1.4024.40)
  Weitere Funktionsbausteine, die die Achse indirekt in eine positionsbezogene Betriebsart schalten, können dies nur eingeschränkt und sind daher nicht für einen bewussten Betriebsartwechsel zu verwenden.

⇨ Anschließend ist ein erneutes Prüfen nötig, ob sich die Achse auch wirklich in einer positionsbezogenen Betriebsart (CSV/CSP) befindet, falls nicht, ist ein Abbruch mit Fehlerbehandlung erforderlich.

Damit dieser Baustein sinnvoll verwendet werden kann, muss der Antriebsregler ein „fliegendes“ Umschalten der Betriebsarten sowie eine Drehzahlbegrenzung im Torque-Mode unterstützen.

Für die Skalierung des Ist- und Soll-Torque stehen unter den Parametern der Nc-Achse im Bereich Drive entsprechende Parameter zur Verfügung.

Eingänge

```
VAR_INPUT
    Execute             : BOOL;
    ContinuousUpdate    : BOOL;
    Relative            : BOOL;
    Torque              : LREAL;
    TorqueRamp          : LREAL;
    VelocityLimitHigh   : LREAL;
    VelocityLimitLow    : LREAL;
    BufferMode          : MC_BufferMode;
    Options             : ST_TorqueControlOptions;
END_VAR
```

Name	Typ	Beschreibung
Execute	BOOL	Mit einer steigenden Flanke wird das Kommando ausgeführt.
ContinuousUpdate	BOOL	Ist dieser Eingang TRUE bei der steigenden Flanke am Eingang „Execute“, so können während der Abarbeitung des Kommandos die Eingänge „Torque“, „TorqueRamp“, „VelocityLimitHigh“ und „VelocityLimitLow“ verändert und schnellstmöglich zur Wirkung gebracht werden.
Relative	BOOL	Ist dieser Eingang TRUE, wird der Torque-Wert relativ um den angegebenen Wert „Torque“ verändert.
Torque	LREAL	Drehmomentsollwert, mit dem der Antrieb betrieben („Relative“ = FALSE), bzw. um den das Drehmoment verändert werden soll („Relative“ = TRUE). (z. B. %) Hierbei bedeutet eine positiver Torque-Vorgabe, Torque in logisch positive Bewegungsrichtung.
TorqueRamp	LREAL	Änderungsgeschwindigkeit des Drehmomentsollwertes (z. B. %/s)
VelocityLimitHigh	LREAL	Oberes Geschwindigkeitslimit zur Begrenzung im CST-Mode (z. B. mm/s). Diese Limitierung muss entsprechend im Prozessabbild konfiguriert werden.
VelocityLimitLow	LREAL	Unteres Geschwindigkeitslimit zur Begrenzung im CST-Mode (z. B. mm/s). Diese Limitierung muss entsprechend im Prozessabbild konfiguriert werden.
BufferMode	MC_BufferMode [▶ 138]	Es wird nur „Aborting“ unterstützt.
Options	ST_TorqueControlOptions [▶ 160]	Datenstruktur, die zusätzliche, selten benötigte Parameter enthält. Im Normalfall kann der Eingang offen bleiben.

Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 14]

Ein-/Ausgänge

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

Name	Typ	Beschreibung
Axis	AXIS_REF [▶ 124]	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
    InTorque            : BOOL;
    Busy                : BOOL;
```

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

Name	Typ	Beschreibung
InTorque	BOOL	TRUE, wenn die Achse das Solldrehmoment aufgebaut hat und keine Limitirung der Geschwindigkeit aktiv 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.
Active	BOOL	Zeigt an, dass der Funktionsbaustein die Achse kontrolliert.
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.

Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 14]

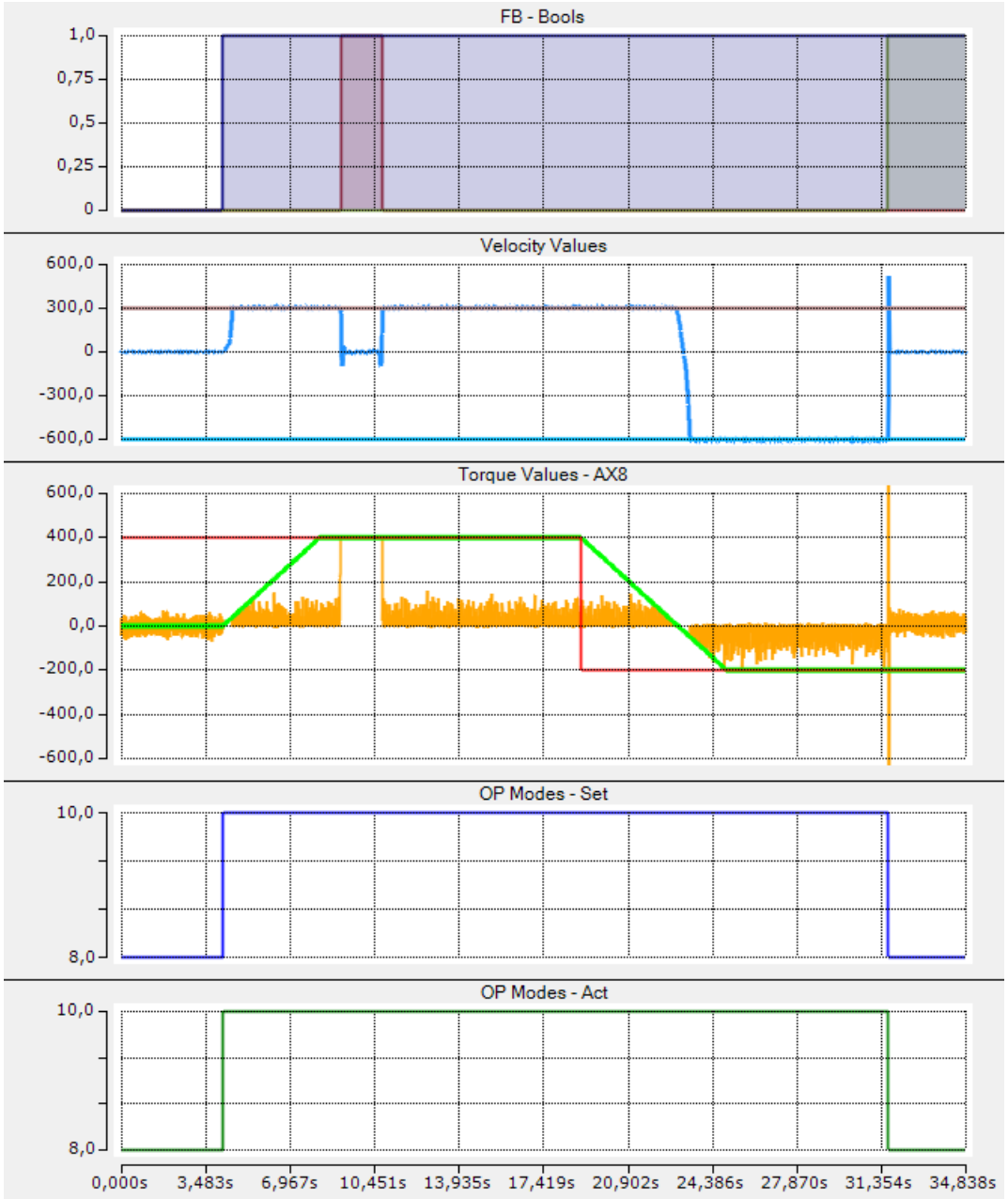
Beispiel MC_TorqueControl beim AX8xxx

```
VAR
    Axis_Main               : AXIS_REF;
    bExecute_TorqueControl  : BOOL;
    fTorque                 : LREAL;
END_VAR
```

```
fTorque := 35;

fbTorqueControl(
    Axis                := Axis_Main,
    Execute             := bExecute_TorqueControl,
    ContinuousUpdate    := TRUE,
    Relative            := FALSE,
    Torque              := fTorque,
    TorqueRamp          :=   10,
    VelocityLimitHigh   :=  400,
    VelocityLimitLow    := -500,
    BufferMode          := ,
    Options             := ,
    InTorque            => ,
    Busy                => ,
    Active              => ,
    CommandAborted      => ,
    Error               => ,
    ErrorId             => );
```

- YT Chart
  - FB - Bools
    - fbTorqueControl.Execute
    - fbTorqueControl.InTorque
    - fbStop.Execute
  - Velocity Values
    - fbTorqueControl.VelocityLimitHigh
    - fbTorqueControl.VelocityLimitLow
    - Ch A Velocity actual value (scaled*0,0006)
  - Torque Values - AX8
    - Ch A Torque/force actual value
    - Ch A Target torque/force
    - fbTorqueControl.Torque (scaled *10)
  - OP Modes - Set
    - Ch A Modes of operation
  - OP Modes - Act
    - Ch A Modes of operation display



Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4024	PC oder CX (x86)	Tc2_MC2

7 Datentypen

7.1 Achsinterface

7.1.1 ST_AdsAddress

ADS-Datenstruktur, die im AXIS_REF [▶ 124] verwendet wird und die ADS-Kommunikationsparameter einer Achse enthält, die für eine direkte ADS-Kommunikation benötigt werden. Im Normalfall muss diese Struktur nicht belegt werden. Erst wenn eine Achse auf einem anderen Zielsystem, oder über eine besondere Port-Nummer angesprochen werden soll, kann der Anwender hier die entsprechende Information hinterlegen.

```
TYPE ST_AdsAddress
STRUCT
    NetId   : STRING(23);
    Port    : UINT;
    Channel : UINT;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
NetId	STRING(23)	AmsNetId des Zielsystems
Port	UINT	Port-Nummer
Channel	UINT	Kanal-Nummer

7.1.2 AXIS_REF

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

```
TYPE AXIS_REF :
VAR_INPUT
    PlcToNc AT %Q* : PLCTONC_AXIS_REF;
END_VAR
VAR_OUTPUT
    NcToPlc AT %I* : NCTOPLC_AXIS_REF;
    ADS            : ST_AdsAddress;
    Status         : ST_AxisStatus;
    DriveAddress   : ST_DriveAddress;
END_VAR
END_TYPE
```

Elemente von AXIS_REF	Beschreibung
PlcToNc	Datenstruktur, die zyklisch zwischen SPS und NC ausgetauscht wird. Über diese Datenstruktur kommunizieren die MC-Funktionsbausteine mit der NC und senden Kontrollinformation von der SPS zur NC. Die Datenstruktur wird automatisch im Ausgangsprozessabbild der SPS platziert und muss im TwinCAT System Manager mit dem Eingangsprozessabbild einer NC-Achse verbunden werden. (Typ: PLCTONC_AXIS_REF [▶ 131])
NcToPlc	Datenstruktur, die zyklisch zwischen SPS und NC ausgetauscht wird. Über diese Datenstruktur kommunizieren die MC-Funktionsbausteine mit der NC und empfangen Statusinformationen von der NC. Die Datenstruktur wird automatisch im Eingangsprozessabbild der SPS platziert und muss im TwinCAT System Manager mit dem Ausgangsprozessabbild einer NC-Achse verbunden werden. Die NCTOPLC-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. (Typ: NCTOPLC_AXIS_REF [▶ 125])

Elemente von AXIS_REF	Beschreibung
ADS	ADS-Datenstruktur, die die ADS-Kommunikationsparameter einer Achse enthält, die für eine direkte ADS-Kommunikation benötigt werden. Im Normalfall muss diese Struktur nicht belegt werden. Erst wenn eine Achse auf einem anderen Zielsystem, oder über eine besondere Port-Nummer angesprochen werden soll, kann der Anwender hier die entsprechende Information hinterlegen.
Status	Datenstruktur, die zusätzliche oder aufbereitete Statusinformation zu einer Achse enthält (Typ: ST_AxisStatus [▶ 154]). Diese Datenstruktur wird nicht zyklisch aufgefrischt, sondern muss durch das SPS-Programm aktualisiert werden. Dazu kann MC_ReadStatus [▶ 30] oder alternativ die Aktion „ReadStatus“ von AXIS_REF aufgerufen werden:
DriveAddress	Datenstruktur, die die ADS-Zugriffdaten eines Antriebsgerätes enthält. Diese Daten werden erst gefüllt nachdem implizit oder explizit der Funktionsbaustein MC_ReadDriveAddress [▶ 55] aufgerufen wurde.

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.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

7.1.3 NCTOPLC_AXIS_REF

Die Datenstruktur NCTOPLC_AXIS_REF ist Teil der Datenstruktur AXIS_REF [▶ 124] und wird automatisch von der NC aktualisiert, sodass die Informationen in jedem SPS-Zyklus aktuell vorliegen. NCTOPLC_AXIS_REF wird auch als Achsinterface zwischen NC und SPS bezeichnet.

```
TYPE NCTOPLC_AXIS_REF
STRUCT
    StateDWord                  : NCTOPLC_AXIS_REF_STATE; (* Status double word *)
    ErrorCode                   : DWORD; (* Axis error code *)
    AxisState                   : DWORD; (* Axis moving status *)
    AxisModeConfirmation        : DWORD; (* Axis mode confirmation (feedback from NC) *)
    HomingState                 : DWORD; (* State of axis calibration (homing) *)
    CoupleState                 : DWORD; (* Axis coupling state *)
    SvbEntries                  : DWORD; (* SVB entries/orders (SVB = Set preparation task) *)
    SafEntries                  : DWORD; (* SAF entries/orders (SAF = Set execution task) *)
    AxisId                      : DWORD; (* Axis ID *)
    OpModeDWord                 : NCTPPLC_AXIS_REF_OPMODE; (* Current operation mode *)
    ActPos                      : LREAL; (* Actual position (absolut value from NC) *)
    ModuloActPos                : LREAL; (* Actual modulo position *)
    ActiveControlLoopIndex      : WORD; (* Active control loop index *)
    ControlLoopIndex            : WORD; (* Axis control loop index (0, 1, 2, when multiple control
loops are used) *)
    ModuloActTurns              : DINT; (* Actual modulo turns *)
    ActVelo                     : LREAL; (* Actual velocity *)
    PosDiff                     : LREAL; (* Position difference (lag distance) *)
    SetPos                      : LREAL; (* Setpoint position *)
    SetVelo                     : LREAL; (* Setpoint velocity *)
    SetAcc                      : LREAL; (* Setpoint acceleration *)
    TargetPos                   : LREAL; (* Estimated target position *)
    ModuloSetPos                : LREAL; (* Setpoint modulo position *)
    ModuloSetTurns              : DINT; (* Setpoint modulo turns *)
```

```
    CmdNo                       : WORD; (* Continuous actual command number *)
    CmdState                    : WORD; (* Command state *)
    SetJerk                     : LREAL;
    SetTorque                   : LREAL;
    ActTorque                   : LREAL;
    StateDWord2                 : NCTOPLC_AXIS_REF_STATE2;
    StateDWord3                 : DWORD;
    TouchProbeState             : DWORD;
    TouchProbeCounter           : DWORD;
    CamCouplingState            : ARRAY [0..7] OF NCTOPLC_AXIS_REF_CAMCOUPLINGSTATE;
    CamCouplingTableID          : ARRAY [0..7] OF UINT;
    ActTorqueDerivative         : LREAL;
    SetTorqueDerivative         : LREAL;
    AbsPhasingPos               : LREAL;
    TorqueOffset                : LREAL
    ActPosWithoutPosCorrection  : LREAL;
    ActAcc                      : LREAL;
    DcTimeStamp                 : UDINT;
    {attribute 'hide'}
    _reserved2                  : ARRAY [1..4] OF USINT;
    UserData                    : LREAL;
END_TYPE
```

Variablenname	Datentyp	Definitionsbereich	Beschreibung
StateDWord	NCTOPLC_AXIS_REF_STATE [▶ 129]	-	Status-Doppelwort.
ErrorCode	DWORD	≥0	Fehlercode Achse
AxisState	DWORD	ENUM [▶ 127]	Bewegungszustand der Achse
AxisModeConfirmation	DWORD	ENUM	Betriebsart der Achse (Rückmeldung der NC)
HomingState	DWORD	ENUM [▶ 128]	Referenzierstatus der Achse („Eichstatus“)
CoupleState	DWORD	ENUM [▶ 128]	Koppelstatus der Achse
SvbEntries	DWORD	≥0	SVB-Einträge/Aufträge
SafEntries	DWORD	≥0	SAF-Einträge/Aufträge (NC-Interpreter, Fifo-Gruppe)
AxisId	DWORD	>0	Achs-ID
OpModeDWord.	NCTOPLC_AXIS_REF_OPMODE [▶ 128]	-	Achs-Betriebsart-Doppelwort
ActPos	LREAL	±∞	Istposition (verrechneter Absolutwert)
ModuloActPos	LREAL	≥0	Modulo-Istposition (verrechneter Wert z. B. in Grad)
ActiveControlLoopIndex	WORD	≥0	Aktiver Achsregelkreis Index
ControlLoopIndex	WORD	≥0	Achsregelkreis Index (0, 1, 2, … wenn mehrere Achsregelkreise verwendet werden)
ModuloActTurns	DINT	±∞	Modulo-Ist-Umdrehungen
ActVelo	LREAL	±∞	Istgeschwindigkeit (optional)
PosDiff	LREAL	±∞	Schleppabstand (Position)
SetPos	LREAL	±∞	Sollposition (verrechneter Absolutwert)
SetVelo	LREAL	±∞	Sollgeschwindigkeit
SetAcc	LREAL	±∞	Sollbeschleunigung
TargetPos	LREAL	±∞	Voraussichtliche Zielposition der Achse
ModuloSetPos	LREAL	≥0	Modulo-Sollposition (verrechneter Wert z. B. in Grad)
ModuloSetTurns	DINT	≥0	Modulo-Soll-Umdrehungen
CmdNo	WORD	≥0	Kommandonummer des aktiven Auftrags der Achse (siehe BufferMode)

Variablenname	Datentyp	Definitionsbereich	Beschreibung
CmdState	WORD	≥0	Kommando Statusinformation (siehe BufferMode)
SetJerk	LREAL		Soll-Ruck
SetTorque	LREAL		Soll-Torque
ActTorque	LREAL		Ist-Torque
StateDWord2	NCTOPLC_AXIS_REF_STATE2 [▶ 130]		Status-Doppelwort2
StateDWord3	DWORD		Status-Doppelwort3
TouchProbeState	DWORD		TouchProbe-Status
TouchProbeCounter	DWORD		TouchProbe-Zähler
CamCouplingState	ARRAY [0..7] OF NCTOPLC_AXIS_REF_CAMCOUPLINGSTATE [▶ 131]		Kurvenkoppelinformation für Multitabellen (ab TwinCAT 3.1.4020.0)
CamCouplingTableId	ARRAY [0..7] OF UINT		Kurven-Koppel-Id für Multitabellen (ab TwinCAT 3.1.4020.0)
ActTorqueDerivative	LREAL		Erste Ableitung des Ist-Torque
SetTorqueDerivative	LREAL		Erste Ableitung des Soll-Torque
AbsPhasingPos	LREAL		Absoluter Phasing-Offset
TorqueOffset	LREAL		Additiver Torqueanteil
ActPosWithoutPosCorrection	LREAL		Istposition ohne Positionskorrektur
ActAcc	LREAL		Istbeschleunigung
DcTimeStamp	UDINT		Aktueller NC-Zeitstempel
UserData	LREAL		Konfigurierbarer Achszustandsparameter

Define	Master: Bewegungszustand / Fahrphase der kontinuierlichen Masterachse (Servo) (AxisState)
0	Sollwertgenerator nicht aktiv (INACTIVE)
1	Sollwertgenerator aktiv (RUNNING)
2	Geschwindigkeits-Override ist Null (OVERRIDE_ZERO)
3	Konstante Geschwindigkeit (PHASE_VELOCONST)
4	Beschleunigungsphase (PHASE_ACCPOS)
5	Verzögerungsphase (PHASE_ACCNEG)
Define	Master: Bewegungszustand / Fahrphase der diskreten Masterachse (Eil/Schleich) (AxisState)
0	Sollwertgenerator nicht aktiv
1	Fahrphase (Eilgang bzw. Schleichgang)
2	Umschaltverzögerung von Eil- auf Schleichgang
3	Schleichfahrt (innerhalb vom Schleichweg)
4	Bremszeit (beginnend ab Bremsweg vor dem Ziel)
Define	Slave: Bewegungszustand / Fahrphase der kontinuierlichen Slaveachse (Servo) (AxisState)
0	Slavegenerator nicht aktiv (INACTIVE)
11	Slave befindet sich in einer Bewegungs-Vorphase (PREPHASE)
12	Slave ist am Aufsynchronisieren (SYNCHRONIZING)
13	Slave ist aufsynchronisiert und fährt synchron (SYNCHRON)

Define	Referenzierstatus der Achse (HomingState)
0	Referenziervorgang fertig (READY)
1	Endlosstart in Richtung der Referenziernocke. **Wenn die Nocke zu Beginn belegt ist, dann wird direkt mit Referenzierstatus 3 begonnen.**
2	Warten auf positive Flanke der Referenziernocke und Achsstopp einleiten
3	Warten bis Achse im Stillstand (Prüfung ob Nocke noch belegt ist) und dann Endlosstart vom Referenziernocken in Richtung Syncimpuls
4	Warten auf fallende Flanke der Referenziernocke
5	Latch aktivieren, warten bis Latch gültig geworden ist und dann den Achsstopp einleiten
6	Wenn Achse im Stillstand dann Istposition setzen (Istposition = Referenzposition + Bremsweg)

Siehe auch Bausteinbeschreibung und Anmerkungen des MC_Home [▶ 99]

Define	Koppelstatus der Achse (CoupleState)
0	Singleachse, die weder Master noch Slave ist (SINGLE)
1	Masterachse mit beliebiger Anzahl Slaves (MASTER)
2	Slaveachse, die Master eines anderen Slaves ist (MASTER-SLAVE)
3	Nur Slaveachse (SLAVE)

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

7.1.4 NCTOPLC_AXIS_REF_OPMODE

Die Struktur NCTOPLC_AXIS_REF_OPMODE ist Teil der Struktur NCTOPLC_AXIS_REF [▶ 125].

```
TYPE NCTOPLC_AXIS_REF_OPMODE :
    DWORD;
END_TYPE
```

Die einzelnen Informationen werden in der Status-Struktur des AXIS_REF an folgenden Stellen zur Verfügung gestellt:

Bit	Variablenname	Beschreibung
0	Status.Opmode.PositionAreaMonitoring	Positionsbereichsüberwachung
1	Status.Opmode.TargetPositionMonitoring	Zielpositionsfensterüberwachung
2	Status.Opmode.LoopMode	Schleifenweg
3	Status.Opmode.MotionMonitoring	Physikalische Bewegungsüberwachung
4	Status.Opmode.PEHTimeMonitoring	PEH-Zeitüberwachung
5	Status.Opmode.BacklashCompensation	Losekompensation
6	Status.Opmode.DelayedErrorReaction	Verzögerte Fehlerreaktion der NC
7	Status.Opmode.Modulo	Modulo-Achse (Modulo-Anzeige)
8	Status.Opmode.SimulationAxis	Simulationsachse
9-11		
12	Status.Opmode.StopMonitoring	Stillstandsüberwachung
13-15		
16	Status.Opmode.PositionLagMonitoring	Schleppabstandsüberwachung Position
17	Status.Opmode.VeloLagMonitoring	Schleppabstandsüberwachung Geschwindigkeit
18	Status.Opmode.SoftLimitMinMonitoring	Endlagenüberwachung Min.
19	Status.Opmode.SoftLimitMaxMonitoring	Endlagenüberwachung Max.
20	Status.Opmode.PositionCorrection	Positionskorrektur („Messsystemfehlerkompensation“)

Bit	Variablenname	Beschreibung
21	Status.Opmode.AllowSlaveCommands	Erlaube Bewegungskommandos an Slaveachsen
22	Status.Opmode.AllowExtSetAxisCommands	Erlaube Bewegungskommandos an Achse die über einen externen Sollwertgenerator gespeist wird
23	Status.NcApplicationRequest	Anforderungs-Bit für die Anwendersoftware (SPS-Code) für z. B. einen „ApplicationHomingRequest“
24-31	Status.NcCycleCounter	NC-Zykluszähler

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

7.1.5 NCTOPLC_AXIS_REF_STATE

Die Struktur NCTOPLC_AXIS_REF_STATE ist Teil der Struktur NCTOPLC_AXIS_REF [▶ 125].

```
TYPE NCTOPLC_AXIS_REF_STATE :
    DWORD;
END_TYPE
```

Die einzelnen Informationen werden in der Status-Struktur des AXIS_REF an folgenden Stellen zur Verfügung gestellt:

Bit	Variablenname	Beschreibung
0	Status.Operational	Achse ist betriebsbereit
1	Status.Homed	Achse ist referenziert („Achse geeicht“)
2	Status.NotMoving	Achse ist im logischen Stillstand („Achse fährt nicht“)
3	Status.InPositionArea	Achse ist im Positionsbereichsfenster (physikalische Rückmeldung)
4	Status.InTargetPosition	Achse ist in Zielposition (PEH) (physikalische Rückmeldung)
5	Status.Protected	Achse ist in geschützter Betriebsart (z. B. Slaveachse)
6	Status.ErrorPropagationDelayed	Achse signalisiert eine Fehlervorwarnung (ab TC 2.11)
7	Status.HasBeenStopped	Achse ist gestoppt worden bzw. führt einen Stopp aus
8	Status.HasJob	Achse hat Auftrag, führt Auftrag aus
9	Status.PositiveDirection	Achse fährt logisch größer
10	Status.NegativeDirection	Achse fährt logisch kleiner
11	Status.HomingBusy	Achse referenziert („Achse wird geeicht“)
12	Status.ConstantVelocity	Achse hat V-Konst bzw. Drehzahl erreicht
13	Status.Compensating	Streckenkompensation passiv[0]/aktiv[1] (siehe MC_MoveSuperImposed)
14	Status.ExtSetPointGenEnabled	Externe Sollwertgenerierung freigegeben
15		Betriebsart noch nicht ausgeführt (Busy). Noch nicht freigegeben!
16	Status.ExternalLatchValid	Externer Latchwert bzw. Messtaster gültig geworden
17	Status.NewTargetPos	Achse hat neue Endposition bzw. neue Geschwindigkeit erhalten
18		Achse nicht in Zielposition bzw. kann/wird diese nicht erreichen (z. B. Achs-Stopp). Noch nicht freigegeben!
19	Status.ContinuousMotion	Achse führt Endlos-Positionierauftrag aus
20	Status.ControlLoopClosed	Achse betriebsbereit und Achsregelkreis geschlossen (z. B. Lageregelung)
21	Status.CamTableQueued	Neue Tabelle steht für „Online Change“ bereit und wartet auf Aktivierung

Bit	Variablenname	Beschreibung
22	Status.CamDataQueued	Tabellendaten (MF) stehen für „Online Change“ bereit und warten auf Aktivierung
23	CamScalingPending	Tabellenskalierungen stehen für „Online Change“ bereit und warten auf Aktivierung
24	Status.CmdBuffered	Nachfolgekommando liegt im Auftragspuffer bereit (siehe BufferMode) (ab TwinCAT V2.10 Build 1311)
25	Status.PTPmode	Achse in PTP Betriebsart (kein Slave, keine NCI-Achse, keine FIFO-Achse) (ab TC 2.10 Build 1326)
26	Status.SoftLimitMinExceeded	Software Endlage Minimum ist aktiv/belegt (ab TC 2.10 Build 1327)
27	Status.SoftLimitMaxExceeded	Software Endlage Maximum ist aktiv/belegt (ab TC 2.10 Build 1327)
28	Status.DriveDeviceError	Antriebshardware hat einen Fehler (keine Warnung), Interpretation nur möglich wenn Antrieb im IO-Datenaustausch, z.B. EtherCAT "OP"-State (ab TC 2.10 Build 1326)
29	Status.MotionCommandsLocked	Achse ist gesperrt für Bewegungskommandos (TcMc2)
30	Status.IoDataInvalid	IO Daten ungültig (z. B. 'WcState' oder 'CdlState' des Feldbusses)
31	Error	Achse ist im Fehlerzustand

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

7.1.6 NCTOPLC_AXIS_REF_STATE2

Die Struktur NCTOPLC_AXIS_REF_STATE2 ist Teil der Struktur NCTOPLC_AXIS_REF [▶ 125].

```
TYPE NCTOPLC_AXIS_REF_STATE2 :
UNION
    Value     : DWORD;
    Flags     : NCTOPLC_AXIS_REF_STATE2_FLAGS;
END_TYPE
```

Siehe auch: NCTOPLC_AXIS_REF_STATE2_FLAGS [▶ 130]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

7.1.7 NCTOPLC_AXIS_REF_STATE2_FLAGS

Die Struktur NCTOPLC_AXIS_REF_STATE2_FLAGS ist Teil der Struktur NCTOPLC_AXIS_REF_STATE2 [▶ 130].

```
TYPE NCTOPLC_AXIS_REF_STATE2_FLAGS :
STRUCT
    AvoidingCollision       : BIT;
    {attribute 'hide'}
    _reserved1              : BIT;
    {attribute 'hide'}
    _reserved2              : BIT;
    {attribute 'hide'}
    _reserved3              : BIT;
    {attribute 'hide'}
    _reserved4              : BIT;
    {attribute 'hide'}
    _reserved5              : BIT;
```

```
    {attribute 'hide'}
    _reserved6              : BIT;
    {attribute 'hide'}
    _reserved7              : BIT;
    {attribute 'hide'}
    _reserved8              : ARRAY [1..3] OF USINT;
END_STRUCT;
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

7.1.8 NCTOPLC_AXIS_REF_CAMCOUPLINGSTATE

```
TYPE NCTOPLC_AXIS_REF_CAMCOUPLINGSTATE :
STRUCT
    CamActivationPending    : BIT;
    CamDeactivationPending  : BIT;
    CamActive               : BIT;
    {attribute 'hide'}
    _reserved1              : BIT;
    {attribute 'hide'}
    _reserved2              : BIT;
    {attribute 'hide'}
    _reserved3              : BIT;
    CamDataQueued           : BIT;
    CamScalingPending       : BIT;
END_STRUCT
END_TYPE
```

Bit	Variablenname	Beschreibung
0	CamActivationPending	Tabelle wartet auf Aktivierung
1	CamDeactivationPending	Tabelle wartet auf Deaktivierung
2	CamActive	Tabelle ist aktiv
3-5		RESERVE
6	CamDataQueued	Tabellendaten (MF) stehen für „Online Change“ bereit und warten auf Aktivierung
7	CamScalingPending	Tabellenskalierungen stehen für „Online Change“ bereit und warten auf Aktivierung

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4020	PC oder CX (x86)	Tc2_MC2

7.1.9 PLCTONC_AXIS_REF

Die Datenstruktur PLCTONC_AXIS_REF ist Bestandteil der Datenstruktur AXIS_REF [▶ 124] und übermittelt zyklisch Informationen an die NC. PLCTONC_AXIS_REF wird auch als Achsinterface zwischen SPS und NC bezeichnet.

```
TYPE PLCTONC_AXIS_REF
STRUCT
    ControlDWord        : PLCTONC_AXIS_REF_CTRL; (* Control double word *)
    Override            : UDINT; (* Velocity override *)
    AxisModeRequest     : UDINT; (* Axis operating mode (PLC request) *)
    AxisModeDWord       : UDINT; (* optional mode parameter *)
    AxisModeLReal       : LREAL; (* optional mode parameter *)
    PositionCorrection  : LREAL; (* Correction value for current position *)
    ExtSetPos           : LREAL; (* external position setpoint *)
    ExtSetVelo          : LREAL; (* external velocity setpoint *)
    ExtSetAcc           : LREAL; (* external acceleration setpoint *)
    ExtSetDirection     : DINT; (* external direction setpoint *)
    {attribute 'hide'}
    _reserved1          : UDINT; (* reserved *)
    ExtControllerOutput : LREAL; (* external controller output *)
```

```
    GearRatio1          : LREAL; (* Gear ratio for dynamic multi master coupling modes *)
    GearRatio2          : LREAL; (* Gear ratio for dynamic multi master coupling modes *)
    GearRatio3          : LREAL; (* Gear ratio for dynamic multi master coupling modes *)
    GearRatio4          : LREAL; (* Gear ratio for dynamic multi master coupling modes *)
    MapState            : BOOL; (* reserved - internal use *)
    PlcCycleControl     : BYTE;
    PlcCycleCount       : BYTE;
    {attribute 'hide'}
    _reserved2          : ARRAY [1..5] OF USINT;
    ExtTorque           : LREAL;
    {attribute 'hide'}
    _reserved3          : ARRAY [1..8] OF USINT;
END_STRUCT
END_TYPE
```

Variablenname	Datentyp	Definitionsbereich	Beschreibung
ControlDWord	PLCTONC_AXSI_REF_CTRL [▶ 132]	0/1	Kontroll-Doppelwort
Override	UDINT	0...1000000	Geschwindigkeits-Override (0% bis 100%)
AxisModeRequest	UDINT		Betriebsart der Achse. Nur für interne Verwendung vorgesehen!
AxisModeDWord	UDINT		Nur für interne Verwendung vorgesehen!
AxisModeLReal	LREAL		Nur für interne Verwendung vorgesehen !
PositionCorrection	LREAL		Istpositionskorrekturwert
ExtSetPos	LREAL		Externe Sollposition
ExtSetVelo	LREAL		Externe Sollgeschwindigkeit
ExtSetAcc	LREAL		Externe Sollbeschleunigung
ExtSetDirection	DINT		Externe Sollfahrrichtung [-1,0,1]
ExtControllerOutput	LREAL		Externe Regler Ausgabe. Noch nicht freigegeben!
GearRatio1	LREAL	±∞	Getriebefaktor (Koppelfaktor) 1
GearRatio2	LREAL	±∞	Getriebefaktor (Koppelfaktor) 2
GearRatio3	LREAL	±∞	Getriebefaktor (Koppelfaktor) 3
GearRatio4	LREAL	±∞	Getriebefaktor (Koppelfaktor) 4
MapState	BOOL		Internal use only
PlcCycleControl	BYTE		Internal use only
PlcCycleCount	BYTE		Internal use only
ExtTorque	LREAL		Torque für MC_TorqueControl mit “ContinuousUpdate”

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

7.1.10 PLCTONC_AXIS_REF_CTRL

Die Struktur PLCTONC_AXIS_REF_CTRL ist Teil der Struktur PLCTONC_AXIS_REF [▶ 131].

```
TYPE PLCTONC_AXIS_REF_CTRL :
    DWORD;
END_TYPE
```

Bit	Variablenname	Beschreibung
0	Enable	Reglerfreigabe
1	FeedEnablePlus	Vorschubfreigabe Plus

Bit	Variablenname	Beschreibung
2	FeedEnableMinus	Vorschubfreigabe Minus
3-4	-	RESERVE
5	HomingSensor	Referenziernocke bzw. Referenziersensor
6-7	-	RESERVE
8	AcceptBlockedDrive	Akzeptiere Sperre der Sollwertübernahme des Drives (z. B. Hardware Endlagen) ab TwinCAT V2.10 Build 1311
9	BlockedDriveDetected	Anwendersignal „Achse ist blockiert“ (z. B. mechanischer Festanschlag). Noch nicht freigegeben!
10-29	-	RESERVE
30	PlcDebugFlag	Debug-Funktion PLC. Nur für internen Gebrauch!

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

7.2 DriveOperationMode

7.2.1 E_DriveOperationMode

Der Datentyp E_DriveOperationMode wird in Verbindung mit den Funktionsbausteinen MC_ReadDriveOperationMode [▶ 46] und MC_WriteDriveOperationMode [▶ 47] verwendet, um die aktuell aktive Betriebsart auszulesen oder festzulegen.

Dieser *Enum* abstrahiert die Umschaltung der Betriebsart für CoE- & SoE-Antriebe.

Für CoE wird die Umschaltung der Betriebsart über das gemappte Objekt 0x6060 „Drive operation mode“ angestoßen und über das gemappte Objekt 0x6061 „Drive operation mode display“ an die nc zurückgemeldet. Ein mapping dieser beiden Objekte ist zwingend für die Funktion erforderlich.

Für SoE erfolgt die Umschaltung der Betriebsarten über das gemappte „Drive control word“.

```
TYPE E_DriveOperationMode :
(
    DriveOperationMode_Default                      := 0,   // default => reactivate the nc default
op mode (if default mode is known!)

    // general operation modes
    DriveOperationMode_Torque                       := 1,   // CoE: CST = 10 cyclic synchronous torq
ue mode / SoE: torque/force control (1)
    DriveOperationMode_Velo1                        := 2,   // CoE: CSV = 9 cyclic synchronous veloc
ity mode / SoE: velo control (2) with feedback 1
    DriveOperationMode_Velo2                        := 3,   // CoE: CSV2 = 16 cyclic synchronous vel
ocity mode with feedback 2
    DriveOperationMode_Pos1                         := 4,   // CoE: CSP = 8 cyclic synchronous posit
ion mode / SoE: pos control (11) with feedback 1, lag less
    DriveOperationMode_Pos2                         := 5,   // CoE: CSP2 = 15 cyclic synchronous pos
ition mode with feedback 2 / SoE: pos control (12) with feedback 2, lag less
    DriveOperationMode_TorqueAndCommutationAngle    := 6,   // CoE: CSTCA = 11 cyclic synchronous to
rque mode with commutation angle

    DriveOperationMode_VelocityMode                 := 34,  // CoE: VL = 2 velocity mode
    DriveOperationMode_DriveBasedHoming             := 38,  // CoE: HM = 6 homing mode

    // special indirect index based operation modes (only for SoE). In most cases the general operat
ion modes shall be used
    DriveOperationMode_Index0                       := 100, // SoE primary operation mode, s. S-0-00
32)
    DriveOperationMode_Index1                       := 101, // SoE secondary operation mode 1, s. S-
0-0033)
    DriveOperationMode_Index2                       := 102, // SoE secondary operation mode 2, s. S-
0-0034)
    DriveOperationMode_Index3                       := 103 // SoE secondary operation mode 3, s. S-0
-0035)
```

```
) DINT;
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

7.3 Externer Sollwertgenerator

7.3.1 ST_ExtSetPointEnableOptions

```
TYPE ST_ExtSetPointEnableOptions
STRUCT
    UseTorqueOffset : BOOL;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
UseTorqueOffset	BOOL	Muss auf TRUE gesetzt werden, wenn ein TorqueOffset mit Hilfe der Funktion MC_ExtSetPointGenFeedWithTorque zyklisch zum Antriebsregler übertragen werden soll.

7.4 Gearing

7.4.1 E_GearInMultiMasterSyncMode

E_GearInMultiMasterSyncMode definiert die möglichen Synchronisationsmodus für die Achskopplung beim MC_GearInMultiMaster [▶ 110], falls die Bewegung der Slave-Achse im Modus AdvancedSlaveDynmaics begrenzt wird (Begrenzung von Geschwindigkeit, Beschleunigung und Ruck).

```
TYPE E_GearInMultiMasterSyncMode :
(
    VELOSYNC := 0,
    POSSYNC1 := 1,
    POSSYNC2 := 2
)INT;
END_TYPE
```

E_GearInMultiMasterSyncMode	Beschreibung
VELOSYNC	Die Slave-Achse fährt geschwindigkeitssynchron zu den Master-Achsen. Im Falle einer Dynamikbegrenzung baut der Slave eine Positionsdifferenz auf, welche nicht wieder aufgeholt wird. (Standard)
POSSYNC1	Die Slave-Achse fährt geschwindigkeits- und positionssynchron zu den Master-Achsen. Im Falle einer Dynamikbegrenzung baut der Slave eine Positionsdifferenz auf, welche aber zu einem späteren Zeitpunkt aufgeholt wird. Änderungen der Getriebefaktoren während der Fahrt werden unter Berücksichtigung der Dynamik ausgeführt. Die daraus resultierende Positionsdifferenz zwischen Master und Slave wird nicht herausgefahren.
POSSYNC2	Die Slave-Achse fährt geschwindigkeits- und positionssynchron zu den Master-Achsen. Im Falle einer Dynamikbegrenzung baut der Slave eine Positionsdifferenz auf, welche aber zu einem späteren Zeitpunkt aufgeholt wird. Änderungen der

E_GearInMultiMasterSyncMode	Beschreibung
	Getriebefaktoren während der Fahrt werden sofort intern verrechnet und der Slave synchronisiert sich kontinuierlich auf die neue Position auf.

7.4.2 ST_GearInDynOptions

```
TYPE ST_GearInDynOptions
STRUCT
    CCVmode : BOOL;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
CCVmode	BOOL	Constant circumference velocity mode

7.4.3 ST_GearInMultiMasterOptions

Die Datenstruktur erweitert den Funktionsbaustein MC_GearInMultiMaster [▶ 110] um zusätzliche Eingangsparameter.

```
TYPE ST_GearInMultiMasterOptions :
STRUCT
    AdvancedSlaveDynamics : BOOL;
    SyncMode              : E_GearInMultiMasterSyncMode;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
AdvancedSlaveDynamics	BOOL	Tauscht den internen Algorithmus des Funktionsbausteins. Damit ist es möglich, auf bereits in Bewegung befindliche Master aufzusynchronisieren. Dabei sollten „Acceleration“ und „Deceleration“ ausschließlich symmetrisch parametriert werden. Bei zu großen Ruck-Vorgaben wird dieser soweit reduziert, dass eine Änderung von Null auf die parametrierte Beschleunigung/ Verzögerung in einem NC-Zyklus erfolgen kann. Die Auflösung der Beschleunigung/Verzögerung hängt somit direkt von der geeigneten Parametrierung des Ruck-Wertes ab.
SyncMode	E_GearInMultiMasterSyncMode [▶ 134]	Der SyncMode legt fest, wie sich die Achskopplung verhält, falls die Bewegung der Slave-Achse im Modus AdvancedSlaveDynamics begrenzt wird (Begrenzung von Geschwindigkeit, Beschleunigung und Ruck). (verfügbar ab TwinCAT 3.1.4024.11)

7.4.4 ST_GearInOptions

```
TYPE ST_GearInOptions
STRUCT
    SlaveType : _E_TcNC_SlaveTypes;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
SlaveType	_E_TcNC_SlaveTypes	Nicht verwendet

7.5 Homing

7.5.1 E_EncoderReferenceMode

Dieser Datentyp wird in Verbindung mit dem Funktionstbaustein MC_Home [▶ 99] und der Strukur … verwendet, um bei Bedarf einen anderen Referenz Mode auszuwählen als in der Konfiguration am Encoder eingestellt ist.

```
Type E_EncoderReferenceMode :
(
    ENCODERREFERENCEMODE_DEFAULT           :=  0,
    ENCODERREFERENCEMODE_PLCCAM            :=  1,
    ENCODERREFERENCEMODE_HARDWARESYNC      :=  2,
    ENCODERREFERENCEMODE_HARDWARELATCHPOS  :=  3,
    ENCODERREFERENCEMODE_HARDWARELATCHNEG  :=  4,
    ENCODERREFERENCEMODE_SOFTWARESYNC      :=  5,
    ENCODERREFERENCEMODE_SOFTDRIVELATCHPOS :=  6,
    ENCODERREFERENCEMODE_SOFTDRIVELATCHNEG :=  7,
)UDINT;
END_TYPE
```

7.5.2 MC_HomingMode

Dieser Datentyp wird zur Parametrierung des Funktionsbausteins MC_Home [▶ 99] verwendet.

```
TYPE MC_HomingMode :
(
    MC_DefaultHoming,    (* default homing as defined in the SystemManager encoder parameters *)
    MC_Direct,           (* Static Homing forcing position from user reference *)
    MC_ForceCalibration, (* set the calibration flag without performing any motion or changing the p
osition *)
    MC_ResetCalibration  (* resets the calibration flag without performing any motion or changing th
e position *)
);
END_TYPE
```

MC_HomingMode	Beschreibung
MC_DefaultHoming	Führt die Standard-Referenzierfahrt aus.
MC_Direct	Setzt die Position der Achse direkt auf Position, ohne eine Bewegung auszuführen.
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.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

7.5.3 ST_HomingOptions

Die Datenstruktur erweitert den Funktionsbaustein MC_Home [▶ 99] um zusätzliche Eingangsparameter.

```
TYPE ST_HomingOptions :
STRUCT
    ClearPositionLag : BOOL;
    SearchDirection  : MC_Direction := MC_Direction.MC_Undefined_Direction;
    SearchVelocity   : LREAL;
    SyncDirection    : MC_Direction := MC_Direction.MC_Undefined_Direction;
    SyncVelocity     : LREAL;
    ReferenceMode    : E_EncoderReferenceMode :=
```

```
E_EncoderReferenceMode.ENCODERREFERENCEMODE_DEFAULT;
END_STRUCT
END_TYPE
```

Name	Typ	Default	Beschreibung
ClearPositionLag	BOOL		Wirkt nur wenn MC_HomingMode [▶ 136] = MC_Direct. Kann optional gesetzt werden, falls Soll- und Istposition auf den gleichen Wert gesetzt werden sollen. Der Schleppfehler wird damit gelöscht.
SearchDirection	MC_Direction [▶ 142]	MC_Undefined_Direction	Die Suche der Eichnocke kann in positiver oder negativer Fahrtrichtung erfolgen. Wird nichts vorgegeben erfolgt die Suche in positiver Fahrtrichtung.
SearchVelocity	LREAL		Achsgeschwindigkeit, mit der die Eichnocke gesucht wird.
SyncDirection	MC_Direction [▶ 142]	MC_Undefined_Direction	Die Suche des Sync-Impulses kann in positiver oder negativer Fahrtrichtung erfolgen. Wird nichts vorgegeben erfolgt die Suche in positiver Fahrtrichtung.
SyncVelocity	LREAL		Achsgeschwindigkeit, mit der der Sync-Impuls gesucht wird.
ReferenceMode	E_EncoderReferenceMode [▶ 136]		Parametriert, welches Signal für die Sync-Impuls-Suche verwendet wird.

7.6 Motion

7.6.1 E_JogMode

Dieser Datentyp wird in Verbindung mit dem Funktionsbaustein MC_Jog [▶ 102] verwendet.

```
TYPE E_JogMode :
(
    MC_JOGMODE_STANDARD_SLOW, (* motion with standard jog parameters for slow motion *)
    MC_JOGMODE_STANDARD_FAST, (* motion with standard jog parameters for fast motion *)
    MC_JOGMODE_CONTINOUS, (* axis moves as long as the jog button is pressed using parameterized dyn
amics *)
    MC_JOGMODE_INCHING, (* axis moves for a certain relative distance *)
    MC_JOGMODE_INCHING_MODULO (* axis moves for a certain relative distance -
 stop position is rounded to the distance value *)
);
END_TYPE
```

E_JogMode	Beschreibung
MC_JOGMODE_STANDARD_SLOW	Die Achse wird so lange verfahren, wie das Signal an einem der Jog-Eingänge TRUE ist. Dabei wird die im TwinCAT System Manager 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 so lange verfahren, wie das Signal an einem der Jog-Eingänge TRUE ist. Dabei wird die im TwinCAT System Manager festgelegte „hohe Geschwindigkeit für Handfunktionen“ und die Standarddynamik verwendet. In dieser Betriebsart haben die am Funktionsbaustein angelegten Positions-, Geschwindigkeits- und Dynamikdaten keine Bedeutung.

E_JogMode	Beschreibung
MC_JOGMODE_CONTINOUS	Die Achse wird so lange 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 einer steigenden Flanke an einem der Jog-Eingänge um eine bestimmte Distanz verfahren, die über den Eingang „Position“ 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.
MC_JOGMODE_INCHING_MODULO	Die Achse wird mit einer steigenden Flanke an einem der Jog-Eingänge um eine bestimmte Distanz verfahren, die über den Eingang „Positions“ festgelegt wird. Die Achsposition rastet dabei auf ein ganzzahliges Vielfaches des Positionsparameters ein. 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.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

7.6.2 E_PositionType

E_PositionType wird bei der externen Sollwertgenerierung MC_ExtSetPointGenEnable [▶ 41] um zu definieren wie die vorgegeben Position zu interpretieren ist.

```
TYPE E_PositionType :
(
    POSITIONTYPE_ABSOLUTE := 1, (*Absolute position*)
    POSITIONTYPE_RELATIVE       (*Relative position*)
);
END_TYPE
```

E_PositionType	Beschreibung
POSITIONTYPE_ABSOLUTE	Absolute Position
POSITIONTYPE_RELATIVE	Relative Position

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

7.6.3 MC_BufferMode

Dieser Datentyp wird mit verschiedenen Funktionsbausteinen der Motion-Control-Bibliothek verwendet. Über BufferMode wird festgelegt, wie aufeinanderfolgende Bewegungskommandos abgearbeitet werden sollen.

```
TYPE MC_BufferMode :
 (
    MC_Aborting,
    MC_Buffered,
    MC_BlendingLow,
    MC_BlendingPrevious,
```

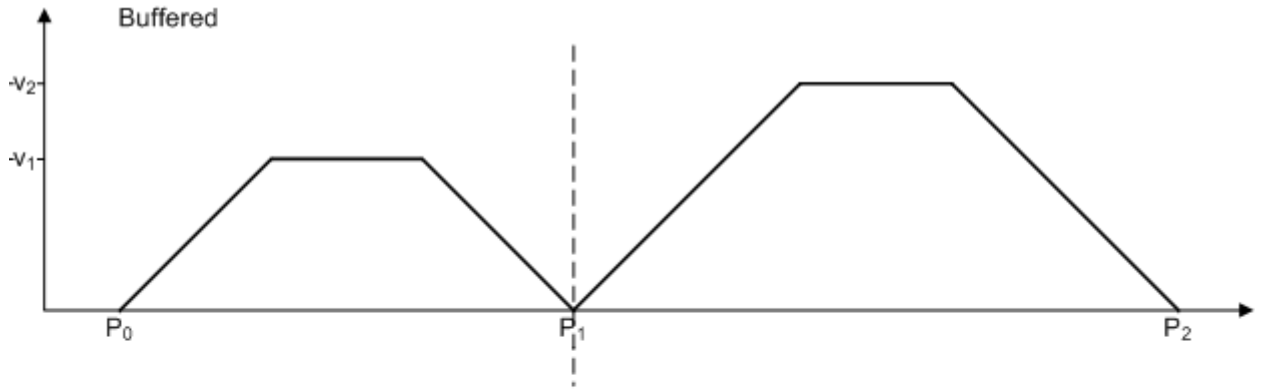
```
    MC_BlendingNext,
    MC_BlendingHigh
 );
END_TYPE
```

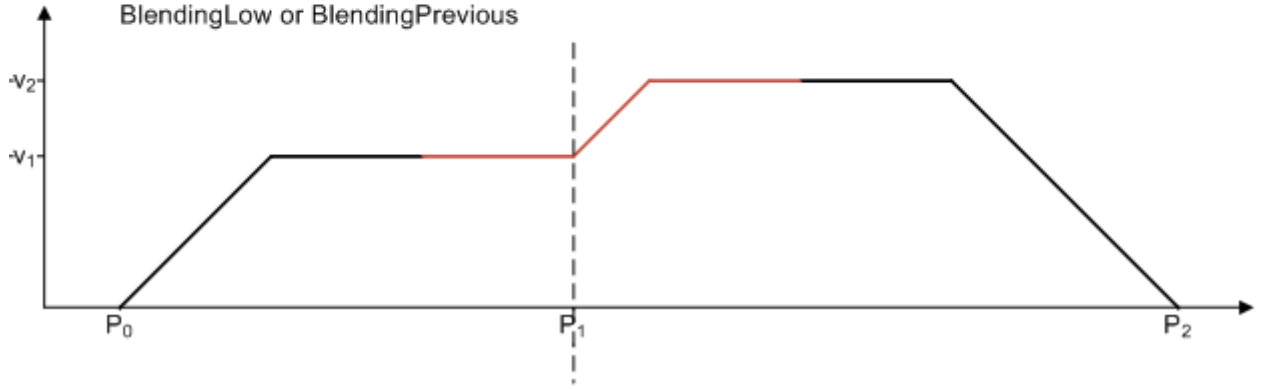
Um den BufferMode zu verwenden, ist immer ein zweiter Funktionsbaustein nötigt. Es ist nicht möglich, einen Move-Baustein mit neuen Parametern zu triggern, während er noch aktiv ist.

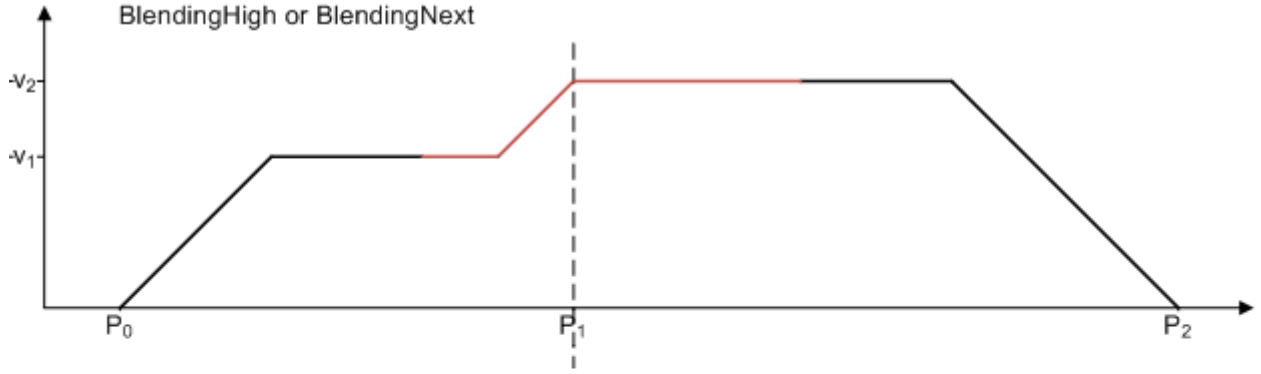
Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 14]

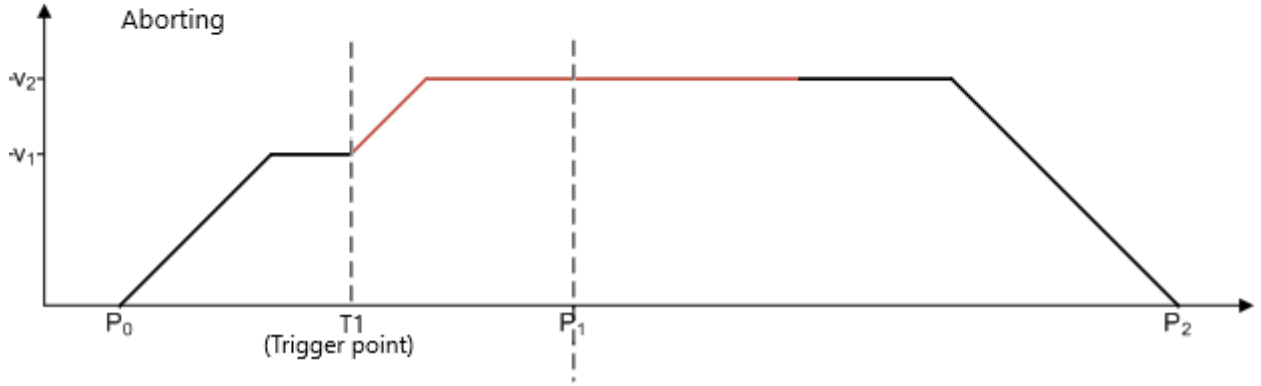
Beispiele:

Im folgenden Beispiel wird eine Achse mit zwei Move-Kommandos zunächst von Position P_0 auf P_1 und anschließend auf P_2 gefahren. Das zweite Kommando wird während der Fahrt nach P_1 aber noch vor der Bremsrampe mit verschieden BufferModes beauftragt. Bezugspunkt für die verschiedenen Geschwindigkeitsprofile ist in jedem Fall P_1. Der Mode legt die Geschwindigkeit v_1 oder v_2 in diesem Punkt fest.



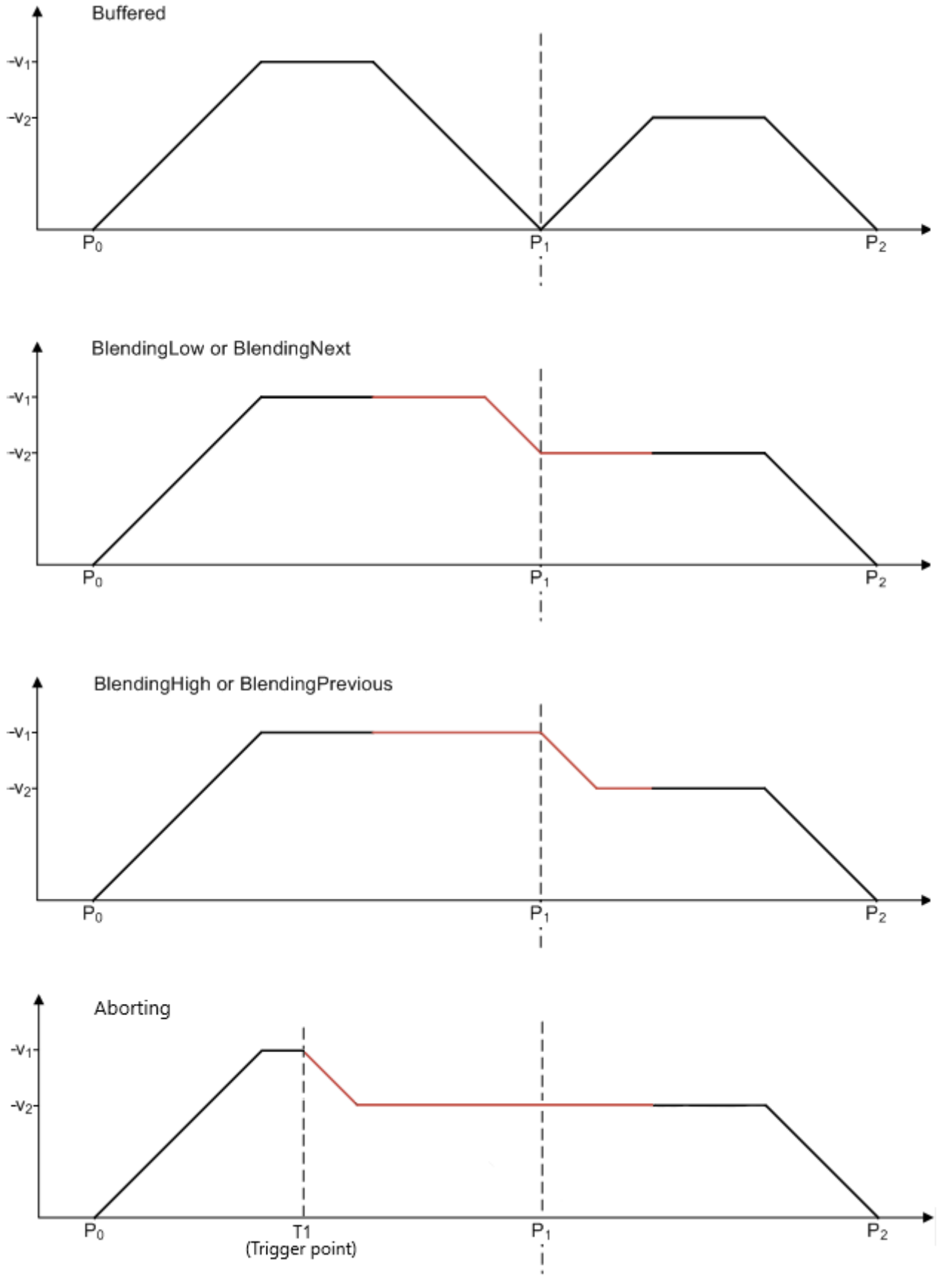






Da die Geschwindkeit des ersten Kommandos niedriger ist als die des zweiten, führen die Modes BlendingLow und BlendingPrevious bzw. die Modes BlendingHigh und BlendingNext jeweils zum selben Ergebnis.

Das nächste Beispiel unterscheidet sich dadurch, dass die Geschwindigkeit des zweiten Kommandos niedriger ist als die des ersten Kommandos. Hier sind nun die Modes BlendingLow und BlendingNext bzw. die Modes BlendingHigh und BlendingPrevious jeweils gleichwertig.



Die gezeigten Geschwindigkeitsprofile setzen voraus, dass das nachfolgende Kommando rechtzeitig, also noch vor der Bremsrampe des ersten Kommandos beauftragt wird. Anderenfalls wird das Blending bestmöglich umgesetzt.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

7.6.4 MC_Direction

Dieser Aufzählungstyp enthält die möglichen Bewegungsrichtungen für verschiedene Funktionsbausteine z.B. MC_MoveVelocity [▶ 82] und MC_MoveModulo [▶ 73].

```
TYPE MC_Direction :
(
    MC_Positive_Direction := 1,
    MC_Shortest_Way ,
    MC_Negative_Direction,
    MC_Current_Direction,
    MC_Undefined_Direction := 128
);
END_TYPE
```

MC_Direction	Beschreibung
MC_Positive_Direction	Positive Bewegungsrichtung
MC_Shortest_Way	Kürzester Weg
MC_Negative_Direction	Negative Bewegungsrichtung
MC_Current_Direction	Aktuelle Bewegungsrichtung wird beibehalten
MC_Undefined_Direction	Undefinierte Bewegungsrichtung

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

7.6.5 ST_MoveOptions

Dieser Datentyp enthält optionale Einstellungen für Fahrkommandos, wie MC_MoveAbsolute [▶ 67] oder MC_Halt [▶ 88].

```
TYPE ST_MoveOptions :
STRUCT
// Command activation at defined ActivationPosition
// Extends the buffer mode when enabled
     EnableBlendingPosition : BOOL;
     BlendingPosition : LREAL;

// PositionAreaMonitoring, TargetPositionMonitoring
// and StopMonitoring can be ignored using this flag
     IgnorePositionMonitoring : BOOL;

// PositionAreaMonitoring, TargetPositionMonitoring
// can be enabled for MC_Stop and MC_Halt commands
     EnableStopPositionMonitoring : BOOL;
END_STRUCT
END_TYPE
```

ST_MoveOptions	Typ	Beschreibung
EnableBlendingPosition	BOOL	Das Flag kann aktiviert werden, um eine abweichende Blending-Position für ein Kommando zu definieren. Im Normalfall ist die Blending-Position das Ziel des laufenden Kommandos. Bei einem aktiven MC_MoveVelocity [▶ 82] ist zunächst kein Ziel definiert.

ST_MoveOptions	Typ	Beschreibung
BlendingPosition	LREAL	Blending-Position, die bei aktiven Flag EnableBlendingPosition verwendet wird.
IgnorePositionMonitoring	BOOL	Die Zielfensterüberwachung kann mit diesem Flag für ein einzelnes Fahrkommando abgeschaltet werden. Die Zielfensterüberwachung wartet nach Erreichen der Zielposition bis auch der Istwert in einem definierten Zielfenster ist. Die Zielfensterüberwachung kann in den Achsparametern aktiviert werden.
EnableStopPositionMonitoring	BOOL	MC_Stop [▶ 90] und MC_Halt [▶ 88]-Kommandos definieren keine Zielposition. Mit diesem Flag kann für diese Kommandos dennoch eine Zielfensterüberwachung aktiviert werden. Das Fahrkommando wartet dann ab, bis sich auch die Istposition der Achse in einem definierten Fenster an der Anhalteposition befindet. Die Zielfensterüberwachung muss dazu in den Achsparametern aktiviert sein.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4024	PC oder CX (x86)	Tc2_MC2

7.7 Nc Process Data

7.7.1 E_NcIoDevice

```
TYPE E_NcIoDevice :
(
    NcIoDeviceDrive := 0,
    NcIoDeviceEncoder := 1
)INT;
END_TYPE
```

7.7.2 E_NcIoOutput

```
TYPE E_NcIoOutput :
(
    NcIoOutputnCtrl1    :=  0,
    NcIoOutputnCtrl2    :=  1,
    NcIoOutputnCtrl3    :=  2,
    NcIoOutputnCtrl4    :=  3,
    NcIoOutputnCtrl5    :=  4,
    NcIoOutputnCtrl6    :=  5,
    NcIoOutputnCtrl7    :=  6,
    NcIoOutputnCtrl8    :=  7,
    NcIoOutputnDataOut1 :=  8,
    NcIoOutputnDataOut2 :=  9,
    NcIoOutputnDataOut3 := 10,
    NcIoOutputnDataOut4 := 11,
    NcIoOutputnDataOut5 := 12,
    NcIoOutputnDataOut6 := 13
);
END_TYPE
```

7.8 Phasing

7.8.1 E_PhasingType

```
TYPE E_PhasingType :
(
    NC_PHASINGTYPE_ABSOLUTE := 1,
    NC_PHASINGTYPE_RELATIVE := 2,
```

```
    NC_PHASINGTYPE_STOP := 4096
)UINT;
END_TYPE
```

7.9 Position Correction

7.9.1 E_AxisPositionCorrectionMode

```
TYPE E_PositionCorrectionMode:
(
    POSITIONCORRECTION_MODE_UNLIMITED, (* no limitation - pass correction immediately *)
    POSITIONCORRECTION_MODE_FAST,      (* limitatation to maximum position change per cycle *)
    POSITIONCORRECTION_MODE_FULLLENGTH (* limitation uses full length to adapt to correction in smal
l steps *)
);
END_TYPE
```

E_AxisPositionCorrectionMode	Beschreibung
POSITIONCORRECTION_MODE_UNLIMITED	Keine Filterung, die Korrektur wird unmittelbar ausgeführt. Es ist zu beachten, das große Korrekturwertänderungen zu hohen Beschleunigungen führen können.
POSITIONCORRECTION_MODE_FAST	Die Positionskorrektur wird soweit begrenzt, dass eine maximale Beschleunigung nicht überschritten wird. Die Korrektur wird aber möglichst schnell vollständig ausgeführt.
POSITIONCORRECTION_MODE_FULLLENGTH	Die Positionskorrektur wird über eine Fahrstrecke der Achse (CorrectionLength) verteilt durchgeführt. Dadurch ergeben sich kleinere Änderungen pro Zeiteinheit.

7.9.2 ST_PositionCompensationTableElement

```
TYPE ST_PositionCompensationTableElement :
STRUCT
    Position     : LREAL;
    Compensation : LREAL;
END_STRUCT
END_TYPE
```

Name	Typ	Beschreibung
Position	LREAL	Unkorrigierter Positionswert (z. B. SetPos)
Compensation	LREAL	Additive Auslgeichswert

7.9.3 ST_PositionCompensationTableParameter

```
TYPE ST_PositionCompensationTableParameter :
STRUCT
    MinPosition       : LREAL;
    MaxPosition       : LREAL;
    NoOfTableElements : UDINT;
    Direction         : E_WorkDirection := WorkDirectionBoth;
    Modulo            : BOOL;
END_STRUCT
END_TYPE
```

Name	Typ	Default	Beschreibung
MinPosition	LREAL		Startposition für den Positionsausgleich
MaxPosition	LREAL		Endposition für den Positionsausgleich
NoOfTableElements	UDINT		Anzahl der Tabelleneinträge

Name	Typ	Default	Beschreibung
Direction	E_WorkDirection [▶ 149]	WorkDirectionBoth	Der Positionsausgleich erfolgt nur in die ausgewählte Richtung.
Modulo	BOOL		FALSE: lineare Achse TRUE: Modulo Achse mit zyklischer Periode

7.10 SelectControlLoop

7.10.1 E_SelectControlLoopType

Definiert wie der Funktionsbaustein MC_SelectControlLoop [▶ 56] die Regelkreisumschaltung durchführt.

```
Type E_SelectControlLoop :
(
    SelectControlLoopType_Undefined,
    SelectControlLoopType_Standard,
    SelectControlLoopType_InternalSyncValue,
    SelectControlLoopType_UserSyncValue
)UDINT;
END_TYPE
```

E_SelectControlLoop	Beschreibung
SelectControlLoopType_Undefined	Nicht definiert.
SelectControlLoopType_Standard	Einfaches Umschalten (ähnlich einem Achsreset)
SelectControlLoopType_InternalSyncValue	Umschalten/Aufsynchronisieren mittels I/D-Anteil des Reglers auf einen internen Initialwert (ruckfrei/ stoßfrei).
SelectControlLoopType_UserSyncValue	Umschalten/Aufsynchronisieren mittels I/D-Anteil des Reglers auf einen parametrierbaren Initialwert.

7.11 Status und Parameter

7.11.1 E_AxisErrorCodes

Achsfehlercodes und Fehlercodes die von den Funktionsbausteinen der Tc2_MC2 zurück gegeben werden können sind im Enum E_AxisErrorCodes definiert. Eine Beschreibung ist in den NC Fehlercodes zu finden.

7.11.2 E_NcAxisType

TwinCAT unterstützt verschieden Achstypen, diese sind im Enum E_NcAxisType definiert.

```
TYPE E_NcAxisType
(
    NcAxisType_undefined            := 0,
    NcAxisType_Continious           := 1,
    NcAxisType_Discrete_TwoSpeed    := 2,
    NcAxisType_LowCostStepper_DigIO := 3,
    NcAxisType_Encoder              := 5,
    NcAxisType_Hydraulic            := 6,
    NcAxisType_TimeGenerator        := 7,
    NcAxisType_Specific             := 100
)DWORD;
END_TYPE
```

E_NcAxisType	Beschreibung
NcAxisType_undefined	
NcAxisType_Continious	Kontinuierliche Achse (auch SERCOS)
NcAxisType_Discrete_TwoSpeed	Diskrete Achse (Eil/Schleich-Achse)

E_NcAxisType	Beschreibung
NcAxisType_LowCostStepper_DigIO	Schrittmotor Achse (ohne PWM Klemme KL2502/30 und ohne Pulse-Train KL2521)
NcAxisType_Encoder	Encoder Achse
NcAxisType_Hydraulic	Kontinuierliche Achse mit Betriebsartumschaltung für Positions-/Druck-Regelung
NcAxisType_TimeGenerator	Time Base Generator
NcAxisType_Specific	

7.11.3 E_NcDriveType

TwinCAT unterstützt verschiedene Antriebe, diese sind im Enum E_NcDriveType definiert.

```
TYPE E_NcDriveType :
(
    NcDriveType_undefined            := 0,
    NcDriveType_M2400_DAC1           := 1,
    NcDriveType_M2400_DAC2           := 2,
    NcDriveType_M2400_DAC3           := 3,
    NcDriveType_M2400_DAC4           := 4,
    NcDriveType_KL4xxx               := 5,
    NcDriveType_KL4xxx_NonLinear     := 6,
    NcDriveType_Discete_TwoSpeed     := 7,
    NcDriveType_Stepper              := 8,
    NcDriveType_Sercos               := 9,
    NcDriveType_KL5051               := 10,
    NcDriveType_AX2000_B200          := 11,
    NcDriveType_ProfilDrive          := 12,
    NcDriveType_Universal            := 13,
    NcDriveType_NcBackplane          := 14,
    NcDriveType_CANopen_Lenze        := 15,
    NcDriveType_CANopen_DS402_MDP742 := 16,
    NcDriveType_AX2000_B900          := 17,
    NcDriveType_KL2531_Stepper       := 20,
    NcDriveType_KL2532_DC            := 21,
    NcDriveType_TCOM                 := 22,
    NcDriveType_MDP_733              := 23,
    NcDriveType_MDP_703              := 24
)DWORD;
END_TYPE
```

E_NcDriveType	Beschreibung
NcDriveType_undefined	
NcDriveType_M2400_DAC1	
NcDriveType_M2400_DAC2	
NcDriveType_M2400_DAC3	
NcDriveType_M2400_DAC4	
NcDriveType_KL4xxx	MDP 252/253: KL4xxx, PWM KL2502_30K (Frq-Cnt-Impuls-Modus), KL4132 (16 Bit), Pulse-Train KL2521, IP2512
NcDriveType_KL4xxx_NonLinear	MDP 252/253: Analog-Typ für nichtlineare Kennlinien
NcDriveType_Discete_TwoSpeed	
NcDriveType_Stepper	
NcDriveType_Sercos	
NcDriveType_KL5051	MDP 510: BiSSI Drive KL5051 mit 32 Bit (siehe KL4xxx)
NcDriveType_AX2000_B200	AX2000-B200 Lightbus, Inkremental mit 32Bit (AX2000)
NcDriveType_ProfilDrive	Inkremental mit 32 Bit
NcDriveType_Universal	Variable Bitmask (max. 32 Bit, signed value)
NcDriveType_NcBackplane	Variable Bitmask (max. 32 Bit, signed value)
NcDriveType_CANopen_Lenze	CANopen Lenze (max. 32 Bit, signed value)

E_NcDriveType	Beschreibung
NcDriveType_CANopen_DS402_MDP742	MDP 742 (DS402): CANopen und EtherCAT (AX2000-B510, AX2000-B1x0, EL7201, AX8000)
NcDriveType_AX2000_B900	AX2000-B900 Ethernet (max. 32 Bit, signed value)
NcDriveType_KL2531_Stepper	Schrittmotorklemme KL2531/KL2541
NcDriveType_KL2532_DC	2-Kanla-DC-Motor-Endstufe (2-channel DC motor stage) KL2532/KL2542, 2-Kanal-PWM-DC-Motorenstufe KL2535/KL2545
NcDriveType_TCOM	TCOM Drive -> Interface to Soft Drive
NcDriveType_MDP_733	MDP 733: Modular Device Profile MDP 733 for DC (e.g. EL7332/EL7342)
NcDriveType_MDP_703	MDP 703: Modular Device Profile MDP 703 for stepper (e.g. EL7031/EL7041)

7.11.4 E_NcEncoderType

TwinCAT unterstützt verschiedene Encoder, diese sind im Enum E_NcEncoderType definiert.

```
TYPE E_NcEncoderType :
(
    NcEncoderType_undefined,
    NcEncoderType_Simulation,
    NcEncoderType_ABS_M3000,
    NcEncoderType_INC_M31X0,
    NcEncoderType_INC_KL5101,
    NcEncoderType_ABS_KL5001_SSI,
    NcEncoderType_INC_KL5051,
    NcEncoderType_ABS_KL30XX,
    NcEncoderType_INC_Sercos_P,
    NcEncoderType_INC_Sercos_PV,
    NcEncoderType_INC_Binary,
    NcEncoderType_ABS_M2510,
    NcEncoderType_ABS_FOX50,
    NcEncoderType_HYDRAULIC_FORCE,
    NcEncoderType_AX2000_B200,
    NcEncoderType_PROFIDRIVE,
    NcEncoderType_UNIVERSAL,
    NcEncoderType_NCBACKPLANE,
    NcEncoderType_CANOPEN_LENZE,
    NcEncoderType_CANOPEN_DS402_MDP513_MDP742,
    NcEncoderType_AX2000_B900,
    NcEncoderType_KL5151,
    NcEncoderType_IP5209,
    NcEncoderType_KL2531_Stepper,
    NcEncoderType_KL2532_DC,
    NcEncoderType_TIMEBASEGENERATOR,
    NcEncoderType_INC_TCOM,
    NcEncoderType_CANOPEN_MDP513_64BIT,
    NcEncoderType_SPECIFIC
)DWORD;
END_TYPE
```

E_NcEncoderType	Beschreibung
NcEncoderType_undefined	
NcEncoderType_Simulation	Simulation
NcEncoderType_ABS_M3000	Absolute mit 24 und 25 Bit sowie 12 und 13 Bit Single Turn Encoder (M3000)
NcEncoderType_INC_M31X0	Inkremental mit 24 Bit (M31x0, M3200, M3100, M2000)
NcEncoderType_INC_KL5101	MDP 511: Inkremental mit 16 Bit und Latch (MDP511: EL7041, EL5101, EL5151, EL2521, EL5021(SinCos); KL5101, IP5109, KL5111)
NcEncoderType_ABS_KL5001_SSI	MDP 500/501: Absolut SSI mit 24 Bit (KL5001, IP5009) (MDP 501: EL5001)
NcEncoderType_INC_KL5051	MDP 510: Absolute/Inkremental BiSSI mit 16 Bit (KL5051, PWM KL2502_30K(Frq-Cnt-Impuls-Modus), Pulse-Train KL2521, IP2512)

E_NcEncoderType	Beschreibung
NcEncoderType_ABS_KL30XX	Absolut analog Eingang mit 16 Bit (KL30xx)
NcEncoderType_INC_Sercos_P	SERCOS „Encoder“ Position
NcEncoderType_INC_Sercos_PV	SERCOS „Encoder“ Position und Geschwindigkeit
NcEncoderType_INC_Binary	Binaerer Inkremental Encoder (0/1)
NcEncoderType_ABS_M2510	Absolut analog Eingang mit 12 Bit (M2510)
NcEncoderType_ABS_FOX50	T&R Fox 50 Modul (24 Bit absolut (SSI))
NcEncoderType_HYDRAULIC_FORCE	MMW-Typ: Kraftermittlung aus Pa, Pb, Aa, Ab
NcEncoderType_AX2000_B200	Inkremental AX2000-B200 Lightbus mit 16/20 Bit (AX2000)
NcEncoderType_PROFIDRIVE	Inkremental mit 32 Bit
NcEncoderType_UNIVERSAL	Inkremental mit variabler Bitmaske (max. 32 Bit)
NcEncoderType_NCBACKPLANE	Inkremental NC Rückwand
NcEncoderType_CANOPEN_LENZE	Inkremental CANopen Lenze
NcEncoderType_CANOPEN_DS402_MDP513_MDP742	MDP 513 / MDP 742 (DS402): CANopen und EtherCAT (AX2000-B510, AX2000-B1x0, EL7201, EL5032/32 Bit)
NcEncoderType_AX2000_B900	Inkremental AX2000-B900 Ethernet
NcEncoderType_KL5151	Inkremental mit 16 Bit Zähler und int. + ext 32 Bit Latch (KL5151_0000) (nur umschaltbar), die 2- kanalige KL5151_0050 hat kein Latch.
NcEncoderType_IP5209	Inkremental mit 16 Bit Zähler und int.32 Bit Latch (IP5209)
NcEncoderType_KL2531_Stepper	Inkremental mit 16 Bit Zähler und int. + ext. 15 Bit Latch (nur umschaltbar) (Schrittmotorklemme KL2531/KL2541)
NcEncoderType_KL2532_DC	Inkremental mit 16 Bit Zähler und ext. 16 Bit Latch (nur umschaltbar) (2-Kanal-DC-Motor-Endstufe KL2532/ KL2542), 2-Kanal-PWM-DC-Motorendstufe KL2535/ KL2545
NcEncoderType_TIMEBASEGENERATOR	Time Base Generator
NcEncoderType_INC_TCOM	TCOM Encoder -> Interface to Soft Drive Encoder
NcEncoderType_CANOPNE_MDP513_64BIT	MDP 513 (DS402, EnDat2.2, 64 Bit): EL5032/64 Bit
NcEncoderType_SPECIFIC	

7.11.5 E_ReadMode

Dieser Datentyp wird in Verbindung mit dem Funktionsbausteinen MC_ReadParameter [▶ 28] und MC_ReadBoolParameter [▶ 26] verwendet, um einen einmaligen oder zyklischen Betrieb festzulegen.

```
TYPE E_ReadMode :
(
    READMODE_ONCE := 1,
    READMODE_CYCLIC
);
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

7.11.6 E_SetScalingFactorMode

Das Enum E_SetScalingFactorMode wird in Verbindung mit dem Funktionsbaustein MC_SetEncoderScalingFactor [▶ 58] verwendet, um den Skalierungsfaktor des aktiven Encoders zu ändern.

```
TYPE E_SetScalingFactorMode :
(
    ENCODERSCALINGMODE_ABSOLUTE,
    ENCODERSCALINGMODE_RELATIVE
)UINT;
END_TYPE
```

E_SetScalingFactorMode	Beschreibung
ENCODERSCALINGMODE_ABSOLUTE	
ENCODERSCALINGMODE_RELATIVE	

7.11.7 E_WorkDirection

Das Enum E_WorkDirection wird verwendet um bei einem Kommando ein Wirkrichtung vorzugeben.

```
TYPE E_WorkDirection :
(
    WorkDirectionNone     := 0,
    WorkDirectionBoth     := 1,
    WorkDirectionPositive := 2,
    WorkDirectionNegative := 3
)INT;
END_TYPE
```

E_WorkDirection	Beschreibung
WorkDirectionNone	Keine Richtungsvorgabe
WorkDirectionBoth	Positive und negative Wirkrichtung
WorkDirectionPositive	Positive Wirkrichtung
WorkDirectionNegative	Negative Wirkrichtung

7.11.8 MC_AxisParameter

Dieser Datentyp wird in Verbindung mit Funktionsbausteinen zum Lesen und Schreiben von Achs-Parametern verwendet.

```
TYPE MC_AxisParameter : (
(* PLCopen specific parameters *) (* Index-Group 0x4000 + ID*)
 CommandedPosition := 1,             (* lreal *) (* taken from NcToPlc *)
 SWLimitPos,                         (* lreal *) (* IndexOffset= 16#0001_000E *)
 SWLimitNeg,                         (* lreal *) (* IndexOffset= 16#0001_000D *)
 EnableLimitPos,                     (* bool  *) (* IndexOffset= 16#0001_000C *)
 EnableLimitNeg,                     (* bool  *) (* IndexOffset= 16#0001_000B *)
 EnablePosLagMonitoring,             (* bool  *) (* IndexOffset= 16#0002_0010 *)
 MaxPositionLag,                     (* lreal *) (* IndexOffset= 16#0002_0012 *)
 MaxVelocitySystem,                  (* lreal *) (* IndexOffset= 16#0000_0027 *)
 MaxVelocityAppl,                    (* lreal *) (* IndexOffset= 16#0000_0027 *)
 ActualVelocity,                     (* lreal *) (* taken from NcToPlc *)
 CommandedVelocity,                  (* lreal *) (* taken from NcToPlc *)
 MaxAccelerationSystem,              (* lreal *) (* IndexOffset= 16#0000_0101 *)
 MaxAccelerationAppl,                (* lreal *) (* IndexOffset= 16#0000_0101 *)
 MaxDecelerationSystem,              (* lreal *) (* IndexOffset= 16#0000_0102 *)
 MaxDecelerationAppl,                (* lreal *) (* IndexOffset= 16#0000_0102 *)
 MaxJerkSystem,                      (* lreal *) (* IndexOffset= 16#0000_0103 *)
 MaxJerkAppl,                        (* lreal *) (* IndexOffset= 16#0000_0103 *)

(* Beckhoff specific parameters *) (* Index-Group 0x4000 + ID*)
 AxisId := 1000,                     (* lreal *) (* IndexOffset= 16#0000_0001 *)
 AxisVeloManSlow,                    (* lreal *) (* IndexOffset= 16#0000_0008 *)
 AxisVeloManFast,                    (* lreal *) (* IndexOffset= 16#0000_0009 *)
 AxisVeloMax,                        (* lreal *) (* IndexOffset= 16#0000_0027 *)
 AxisAcc,                            (* lreal *) (* IndexOffset= 16#0000_0101 *)
 AxisDec,                            (* lreal *) (* IndexOffset= 16#0000_0102 *)
 AxisJerk,                           (* lreal *) (* IndexOffset= 16#0000_0103 *)
 MaxJerk,                            (* lreal *) (* IndexOffset= 16#0000_0103 *)
 AxisMaxVelocity,                    (* lreal *) (* IndexOffset= 16#0000_0027 *)
 AxisRapidTraverseVelocity,          (* lreal *) (* IndexOffset= 16#0000_000A *)
 AxisManualVelocityFast,             (* lreal *) (* IndexOffset= 16#0000_0009 *)
 AxisManualVelocitySlow,             (* lreal *) (* IndexOffset= 16#0000_0008 *)
 AxisCalibrationVelocityForward,     (* lreal *) (* IndexOffset= 16#0000_0006 *)
```

```
 AxisCalibrationVelocityBackward,     (* lreal *) (* IndexOffset= 16#0000_0007 *)
 AxisJogIncrementForward,             (* lreal *) (* IndexOffset= 16#0000_0018 *)
 AxisJogIncrementBackward,            (* lreal *) (* IndexOffset= 16#0000_0019 *)
 AxisEnMinSoftPosLimit,               (* bool  *) (* IndexOffset= 16#0001_000B *)
 AxisMinSoftPosLimit,                 (* lreal *) (* IndexOffset= 16#0001_000D *)
 AxisEnMaxSoftPosLimit,               (* bool  *) (* IndexOffset= 16#0001_000C *)
 AxisMaxSoftPosLimit,                 (* lreal *) (* IndexOffset= 16#0001_000E *)
 AxisEnPositionLagMonitoring,         (* bool  *) (* IndexOffset= 16#0002_0010 *)
 AxisMaxPosLagValue,                  (* lreal *) (* IndexOffset= 16#0002_0012 *)
 AxisMaxPosLagFilterTime,             (* lreal *) (* IndexOffset= 16#0002_0013 *)
 AxisEnPositionRangeMonitoring,       (* bool  *) (* IndexOffset= 16#0000_000F *)
 AxisPositionRangeWindow,             (* lreal *) (* IndexOffset= 16#0000_0010 *)
 AxisEnTargetPositionMonitoring,      (* bool  *) (* IndexOffset= 16#0000_0015 *)
 AxisTargetPositionWindow,            (* lreal *) (* IndexOffset= 16#0000_0016 *)
 AxisTargetPositionMonitoringTime,    (* lreal *) (* IndexOffset= 16#0000_0017 *)
 AxisEnInTargetTimeout,               (* bool  *) (* IndexOffset= 16#0000_0029 *)
 AxisInTargetTimeout,                 (* lreal *) (* IndexOffset= 16#0000_002A *)
 AxisEnMotionMonitoring,              (* bool  *) (* IndexOffset= 16#0000_0011 *)
 AxisMotionMonitoringWindow,          (* lreal *) (* IndexOffset= 16#0000_0028 *)
 AxisMotionMonitoringTime,            (* lreal *) (* IndexOffset= 16#0000_0012 *)
 AxisDelayTimeVeloPosition,           (* lreal *) (* IndexOffset= 16#0000_0104 *)
 AxisEnLoopingDistance,               (* bool  *) (* IndexOffset= 16#0000_0013 *)
 AxisLoopingDistance,                 (* lreal *) (* IndexOffset= 16#0000_0014 *)
 AxisBacklash,                        (* lreal *) (* IndexOffset= 16#0000_002C *)
 AxisEnDataPersistence,               (* bool  *) (* IndexOffset= 16#0000_0030 *)
 AxisRefVeloOnRefOutput,              (* lreal *) (* IndexOffset= 16#0003_0101 *)
 AxisOverrideType,                    (* lreal *) (* IndexOffset= 16#0000_0105 *)
 (* new since 4/2007 *)
 AxisEncoderScalingFactor,            (* lreal *) (* IndexOffset= 16#0001_0006 *)
 AxisEncoderOffset,                   (* lreal *) (* IndexOffset= 16#0001_0007 *)
 AxisEncoderDirectionInverse,         (* bool  *) (* IndexOffset= 16#0001_0008 *)
 AxisEncoderMask,                     (* dword *) (* IndexOffset= 16#0001_0015 *)
 AxisEncoderModuloValue,              (* lreal *) (* IndexOffset= 16#0001_0009 *)
 AxisModuloToleranceWindow,           (* lreal *) (* IndexOffset= 16#0001_001B *)
 AxisEnablePosCorrection,             (* bool  *) (* IndexOffset= 16#0001_0016 *)
 AxisPosCorrectionFilterTime,         (* lreal *) (* IndexOffset= 16#0001_0017 *)
 (* new since 1/2010 *)
 AxisUnitInterpretation,              (* lreal *) (* IndexOffset= 16#0000_0026 *)
 AxisMotorDirectionInverse,           (* bool  *) (* IndexOffset= 16#0003_0006 *)
 (* new since 1/2011 *)
 AxisCycleTime,                       (* lreal *) (* IndexOffset= 16#0000_0004 *)
 (* new since 5/2011 *)
 AxisFastStopSignalType,              (* dword *) (* IndexOffset= 16#0000_001E *)
 AxisFastAcc,                         (* lreal *) (* IndexOffset= 16#0000_010A *)
 AxisFastDec,                         (* lreal *) (* IndexOffset= 16#0000_010B *)
 AxisFastJerk,                        (* lreal *) (* IndexOffset= 16#0000_010C *)
(* new since 1/2012 *)
 AxisEncoderScalingNumerator,         (* lreal *) (* IndexOffset= 16#0001_0023 - available in Tc3 *)
 AxisEncoderScalingDenominator,       (* lreal *) (* IndexOffset= 16#0001_0024 - available in Tc3 *)
(* new since 7/2016 *)
 AxisMaximumAcceleration,             (* lreal *) (* IndexOffset= 16#0000_00F1 - available in Tc3 *)
 AxisMaximumDeceleration,             (* lreal *) (* IndexOffset= 16#0000_00F2 - available in Tc3 *)
 AxisVeloJumpFactor,                  (* lreal *) (* IndexOffset= 16#0000_0106 *)
 AxisToleranceBallAuxAxis,            (* lreal *) (* IndexOffset= 16#0000_0108 *)
 AxisMaxPositionDeviationAuxAxis,     (* lreal *) (* IndexOffset= 16#0000_0109 *)
 AxisErrorPropagationMode,            (* dword *) (* IndexOffset= 16#0000_001A *)
 AxisErrorPropagationDelay,           (* lreal *) (* IndexOffset= 16#0000_001B *)
 AxisCoupleSlaveToActualValues,       (* bool  *) (* IndexOffset= 16#0000_001C *)
 AxisAllowMotionCmdToSlaveAxis,       (* bool  *) (* IndexOffset= 16#0000_0020 *)
 AxisAllowMotionCmdToExtSetAxis,      (* bool  *) (* IndexOffset= 16#0000_0021 *)
 AxisEncoderSubMask,                  (* dword *) (* IndexOffset= 16#0001_0108 *)
 AxisEncoderReferenceSystem,          (* dword *) (* IndexOffset= 16#0001_0019 *)
 AxisEncoderPositionFilterPT1,        (* lreal *) (* IndexOffset= 16#0001_0010 *)
 AxisEncoderVelocityFilterPT1,        (* lreal *) (* IndexOffset= 16#0001_0011 *)
 AxisEncoderAccelerationFilterPT1,    (* lreal *) (* IndexOffset= 16#0001_0012 *)
 AxisEncoderMode,                     (* dword *) (* IndexOffset= 16#0001_000A *)
 AxisEncoderHomingInvDirCamSearch,    (* bool  *) (* IndexOffset= 16#0001_0101 *)
 AxisEncoderHomingInvDirSyncSearch,   (* bool  *) (* IndexOffset= 16#0001_0102 *)
 AxisEncoderHomingCalibValue,         (* lreal *) (* IndexOffset= 16#0001_0103 *)
 AxisEncoderReferenceMode,            (* dword *) (* IndexOffset= 16#0001_0107 *)
 AxisRefVeloOutputRatio,              (* lreal *) (* IndexOffset= 16#0003_0102 *)
 AxisDrivePositionOutputScaling,      (* lreal *) (* IndexOffset= 16#0003_0109 *)
 AxisDriveVelocityOutputScaling,      (* lreal *) (* IndexOffset= 16#0003_0105 *)
 AxisDriveVelocityOutputDelay,        (* lreal *) (* IndexOffset= 16#0003_010D *)
 AxisDriveMinOutputLimitation,        (* lreal *) (* IndexOffset= 16#0003_000B *)
 AxisDriveMaxOutputLimitation,        (* lreal *) (* IndexOffset= 16#0003_000C *)
 AxisTorqueInputScaling,              (* lreal *) (* IndexOffset= 16#0003_0031 - available in Tc3 *)
 AxisTorqueInputFilterPT1,            (* lreal *) (* IndexOffset= 16#0003_0032 - available in Tc3 *)
 AxisTorqueDerivationInputFilterPT1,  (* lreal *) (* IndexOffset= 16#0003_0033 - available in Tc3 *)
 AxisTorqueOutputScaling,             (* lreal *) (* IndexOffset= 16#0003_010B *)
```

```
 AxisTorqueOutputDelay,              (* lreal *) (* IndexOffset= 16#0003_010F *)
 AxisAccelerationOutputScaling,      (* lreal *) (* IndexOffset= 16#0003_010A *)
 AxisAccelerationOutputDelay,        (* lreal *) (* IndexOffset= 16#0003_010E *)
 AxisDrivePosOutputSmoothFilterType, (* dword *) (* IndexOffset= 16#0003_0110 *)
 AxisDrivePosOutputSmoothFilterTime, (* lreal *) (* IndexOffset= 16#0003_0111 *)
 AxisDrivePosOutputSmoothFilterOrder,(* dword *) (* IndexOffset= 16#0003_0112 *)
 AxisDriveMode,                      (* dword *) (* IndexOffset= 16#0003_000A *)
 AxisDriftCompensationOffset,        (* lreal *) (* IndexOffset= 16#0003_0104 *)
 AxisPositionControlKv,              (* lreal *) (* IndexOffset= 16#0002_0102 *)
 AxisCtrlVelocityPreCtrlWeight,      (* lreal *) (* IndexOffset= 16#0002_000B *)
 AxisControllerMode,                 (* dword *) (* IndexOffset= 16#0002_000A *)
 AxisCtrlAutoOffset,                 (* bool  *) (* IndexOffset= 16#0002_0110 *)
 AxisCtrlAutoOffsetTimer,            (* lreal *) (* IndexOffset= 16#0002_0115 *)
 AxisCtrlAutoOffsetLimit,            (* lreal *) (* IndexOffset= 16#0002_0114 *)
 AxisSlaveCouplingControlKcp,        (* lreal *) (* IndexOffset= 16#0002_010F *)
 AxisCtrlOutputLimit,                (* lreal *) (* IndexOffset= 16#0002_0100 *)

(* Beckhoff specific axis status information - READ ONLY *) (* Index-Group 0x4100 + ID*)
 AxisTargetPosition := 2000,         (* lreal *) (* IndexOffset= 16#0000_0013 *)
 AxisRemainingTimeToGo,              (* lreal *) (* IndexOffset= 16#0000_0014 *)
 AxisRemainingDistanceToGo,          (* lreal *) (* IndexOffset= 16#0000_0022, 16#0000_0042 *)
 AxisMcSetPositionOffset,            (* lreal *) (* IndexOffset= 16#00n1_0017 *)

(* Beckhoff specific axis functions *)
 (* read/write gear ratio of a slave *)
 AxisGearRatio := 3000,              (* lreal *) (* read: IndexGroup=0x4100+ID, IdxOffset=16#0000_00
22, *)
 (* write:IndexGroup=0x4200+ID, IdxOffset=16#0000_0042 *)

(* Beckhoff specific other parameters *)
 (* new since 1/2011 *)
 NcSafCycleTime := 4000,             (* lreal *) (* IndexOffset= 16#0000_0010 *)
 NcSvbCycleTime                      (* lreal *) (* IndexOffset= 16#0000_0012 *)
);
END_TYPE
```

Der Parameter AxisGearRatio kann nur gelesen oder geschrieben werden, wenn die Achse als Slave gekoppelt ist. Während der Fahrt sind nur sehr kleine Änderungen erlaubt.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

7.11.9 MC_AxisStates

Dieser Datentyp beschreibt die Betriebszustände nach dem PlcOpen-Zustandsdiagramm [▶ 11].

```
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,
    MC_AXISSTATE_SYNCHRONIZEDMOTION
);
END_TYPE
```

Siehe auch: Allgemeine Regeln für MC-Funktionsbausteine [▶ 14].

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

7.11.10 ST_AxisComponents

```
TYPE ST_AxisComponents
STRUCT
    EncoderIDs    : ARRAY[1..9] OF DWORD;
    ControllerIDs : ARRAY[1..9] OF DWORD;
    DriveIDs      : ARRAY[1..9] OF DWORD;
END_STRUCT
END_TYPE
```

7.11.11 ST_AxisOpModes

Dieser Datentyp enthält Informationen über die Parametrierung der Betriebsarten einer Achse.

```
TYPE ST_AxisOpModes :
STRUCT
    PositionAreaMonitoring   : BOOL; (* bit 0 - OpModeDWord *)
    TargetPositionMonitoring : BOOL; (* bit 1 - OpModeDWord *)
    LoopMode                 : BOOL; (* bit 2 - OpModeDWord - loop mode for two speed axes *)
    MotionMonitoring         : BOOL; (* bit 3 - OpModeDWord *)
    PEHTimeMonitoring        : BOOL; (* bit 4 - OpModeDWord *)
    BacklashCompensation     : BOOL; (* bit 5 - OpModeDWord *)
    DelayedErrorReaction     : BOOL; (* bit 6 - OpModeDWord *)
    Modulo                   : BOOL; (* bit 7 - OpModeDWord -
 axis is parameterized as modulo axis *)
    SimulationAxis           : BOOL; (* bit 8 - OpModeDWord *)
    PositionLagMonitoring    : BOOL; (* bit 16 - OpModeDWord *)
    VelocityLagMonitoring    : BOOL; (* bit 17 - OpModeDWord *)
    SoftLimitMinMonitoring   : BOOL; (* bit 18 - OpModeDWord *)
    SoftLimitMaxMonitoring   : BOOL; (* bit 19 - OpModeDWord *)
    PositionCorrection       : BOOL; (* bit 20 - OpModeDWord *)
    AllowSlaveCommands       : BOOL; (* bit 21 - OpModeDWord *)
    AllowExtSetAxisCommands  : BOOL; (* bit 22 - OpModeDWord *)
END_STRUCT
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

7.11.12 ST_AxisParameterSet

Dieser Datentyp enthält den gesamten Parameterdatensatz einer Achse, der mit dem Funktionsbaustein MC_ReadParameterSet [▶ 29] gelesen werden kann.

Einzelne Parameter, die zur Laufzeit änderbar sind, können mit MC_WriteParameter [▶ 33] geschrieben werden. Es ist nicht möglich, den Parameterdatensatz als Ganzes zurückzuschreiben.

Die einzelnen Parameter sind in der NC-ADS-Dokumentation beschrieben.

```
TYPE ST_AxisParameterSet :
STRUCT
    (* AXIS: *)
    AxisId                     : DWORD;          (* 0x00000001 *)
    nAxisType                  : E_NcAxisType;   (* 0x00000003 *)
    sAxisName                  : STRING(31);     (* 0x00000002 *)
    fAxisCycleTime             : LREAL;          (* 0x00000004 *)
    bEnablePositionAreaControl : WORD;           (* 0x0000000F *)
    fPositionAreaControlRange  : LREAL;          (* 0x00000010 *)
    bEnableMotionControl       : WORD;           (* 0x00000011 *)
    fMotionControlTime         : LREAL;          (* 0x00000012 *)
    bEnableLoop                : WORD;           (* 0x00000013 *)
    fLoopDistance              : LREAL;          (* 0x00000014 *)
    bEnableTargetPosControl    : WORD;           (* 0x00000015 *)
    fTargetPosControlRange     : LREAL;          (* 0x00000016 *)
    fTargetPosControlTime      : LREAL;          (* 0x00000017 *)
    fVeloMaximum               : LREAL;          (* 0x00000027 *)
    fRefVeloSearch             : LREAL;          (* 0x00000006 *)
    fRefVeloSync               : LREAL;          (* 0x00000007 *)
    fVeloSlowManual            : LREAL;          (* 0x00000008 *)
    fVeloFastManual            : LREAL;          (* 0x00000009 *)
    fMotionControlRange        : LREAL;          (* 0x00000028 *)
```

```
    bEnablePEHTimeControl       : WORD;           (* 0x00000029 *)
    fPEHControlTime             : LREAL;          (* 0x0000002A *)
    bEnableBacklashCompensation : WORD;           (* 0x0000002B *)
    fBacklash                   : LREAL;          (* 0x0000002C *)
    sAmsNetId                   : T_AmsNetId;     (* 0x00000031 *)
    nPort                       : WORD;           (* 0x00000031 *)
    nChnNo                      : WORD;           (* 0x00000031 *)
    fAcceleration               : LREAL;          (* 0x00000101 *)
    fDeceleration               : LREAL;          (* 0x00000102 *)
    fJerk                       : LREAL;          (* 0x00000103 *)

    (* ENCODER: *)
    nEncId                      : DWORD;          (* 0x00010001 *)
    nEncType                    : E_NcEncoderType;(* 0x00010003 *)
    sEncName                    : STRING(31);     (* 0x00010002 *)
    fEncScaleFactorNumerator    : LREAL;          (* 0x00010023 *)
    fEncScaleFactorDenominator  : LREAL;          (* 0x00010024 *)
    fEncScaleFactor             : LREAL;          (* 0x00010006 *)
    fEncOffset                  : LREAL;          (* 0x00010007 *)
    bEncIsInverse               : WORD;           (* 0x00010008 *)
    fEncModuloFactor            : LREAL;          (* 0x00010009 *)
    nEncMode                    : DWORD;          (* 0x0001000A *)
    bEncEnableSoftEndMinControl : WORD;           (* 0x0001000B *)
    bEncEnableSoftEndMaxControl : WORD;           (* 0x0001000C *)
    fEncSoftEndMin              : LREAL;          (* 0x0001000D *)
    fEncSoftEndMax              : LREAL;          (* 0x0001000E *)
    nEncMaxIncrement            : DWORD;          (* 0x00010015 *)
    nEncRefSoftSyncMask         : DWORD;          (* 0x00010108 *)
    bEncEnablePosCorrection     : WORD;           (* 0x00010016 *)
    nEncReferenceSystem         : DWORD;          (* 0x00010019 *)
    fEncPosCorrectionFilterTime : LREAL;          (* 0x00010017 *)
    bEncRefSearchInverse        : UINT;           (* 0x00010101 *)
    bEncRefSyncInverse          : UINT;           (* 0x00010102 *)
    nEncRefMode                 : UDINT;          (* 0x00010107 *)
    fEncRefPosition             : LREAL;          (* 0x00010103 *)

    (* CONTROLLER: *)
    nCtrlId                     : DWORD;        (* 0x00020001 *)
    nCtrlType                   : DWORD;        (* 0x00020003 *)
    sCtrlName                   : STRING(31);   (* 0x00020002 *)
    bCtrlEnablePosDiffControl   : WORD;         (* 0x00020010 *)
    bCtrlEnableVeloDiffControl  : WORD;         (* 0x00020011 *)
    fCtrlPosDiffMax             : LREAL;        (* 0x00020012 *)
    fCtrlPosDiffMaxTime         : LREAL;        (* 0x00020013 *)
    fCtrlPosKp                  : LREAL;        (* 0x00020102 *)
    fCtrlPosTn                  : LREAL;        (* 0x00020103 *)
    fCtrlPosTv                  : LREAL;        (* 0x00020104 *)
    fCtrlPosTd                  : LREAL;        (* 0x00020105 *)
    fCtrlPosExtKp               : LREAL;        (* 0x00020106 *)
    fCtrlPosExtVelo             : LREAL;        (* 0x00020107 *)
    fCtrlAccKa                  : LREAL;        (* 0x00020108 *)

    (* DRIVE: *)
    nDriveId                    : DWORD;        (* 0x00030001 *)
    nDriveType                  : E_NcDriveType;(* 0x00030003 *)
    sDriveName                  : STRING(31);   (* 0x00030002 *)
    bDriveIsInverse             : WORD;         (* 0x00030006 *)
    nDriveControlDWord          : DWORD;        (* 0x00030010 *)
    fDriveVeloReferenz          : LREAL;        (* 0x00030101 *)
    fDriveOutputReferenz        : LREAL;        (* 0x00030102 *)
    fDriveOutputScalingAcc      : LREAL;        (* 0x0003000A *)
    fDriveOutputScalingTorque   : LREAL;        (* 0x0003000B *)
    fDriveInputScalingTorque    : LREAL;        (* 0x00030031 *)
    fDriveInputFiltertimeTorque : LREAL;        (* 0x00030032 *)
    fDriveInputFiltertimeTorqueDerivative: LREAL;(* 0x00030033 *)

    fAccelerationMax            : LREAL;        (* 0x000300F1 *)
    fDecelerationMax            : LREAL;        (* 0x000300F2 *)
END_STRUCT
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

7.11.13 ST_AxisStatus

Dieser Datentyp enthält umfangreiche Statusinformationen über eine Achse. Die Datenstruktur muss in jedem SPS-Zyklus durch einen Aufruf von MC_ReadStatus [▶ 30] oder durch einen Aufruf der Aktion Axis.ReadStatus (AXIS_REF [▶ 124]) aktualisiert werden.

```
TYPE ST_AxisStatus :
STRUCT
    UpdateTaskIndex          : BYTE;  (* Task-Index of the task that updated this data set *)
    UpdateCycleTime          : LREAL; (* task cycle time of the task which calls the status function
 *)
    CycleCounter             : UDINT; (* PLC cycle counter when this data set updated *)
    NcCycleCounter           : UDINT; (* NC cycle counter incremented after NC task updated NcToPlc
data structures *)

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

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

    (* PLCopen motion control statemachine states: *)
    ErrorStop                : BOOL;
    Disabled                 : BOOL;
    Stopping                 : BOOL;
    StandStill               : BOOL;
    DiscreteMotion           : BOOL;
    ContinuousMotion         : BOOL; (* StateDWord bit 19 *)
    SynchronizedMotion       : BOOL;
    Homing                   : BOOL;

    (* additional status - (PLCopen definition)*)
    ConstantVelocity         : BOOL; (* StateDWord bit 12 *)
    Accelerating             : BOOL;
    Decelerating             : BOOL;

    (* Axis.NcToPlc.StateDWord *)
    Operational              : BOOL; (* StateDWord bit 0 *)
    ControlLoopClosed        : BOOL; (* StateDWord bit 20 -
 operational and position control active *)
    HasJob                   : BOOL; (* StateDWord bit 8 *)
    HasBeenStopped           : BOOL; (* StateDWord bit 7 *)
    NewTargetPosition        : BOOL; (* StateDWord bit 17 *-
* new target position commanded during move *)
    InPositionArea           : BOOL; (* StateDWord bit 3 *)
    InTargetPosition         : BOOL; (* StateDWord bit 4 *)
    ProtectedMode            : BOOL; (* StateDWord bit 5 *)
    Homed                    : BOOL; (* StateDWord bit 1 *)
    HomingBusy               : BOOL; (* StateDWord bit 11 *)
    MotionCommandsLocked     : BOOL; (* StateDWord bit 29 *-* stop 'n hold *)
    SoftLimitMinExceeded     : BOOL; (* StateDWord bit 26 *-* reverse soft travel limit exceeded *)
    SoftLimitMaxExceeded     : BOOL; (* StateDWord bit 27 *-* forward soft travel limit exceeded *)

    Moving                   : BOOL; (* StateDWord bit 9 or 10 *)
    PositiveDirection        : BOOL; (* StateDWord bit 9 *)
    NegativeDirection        : BOOL; (* StateDWord bit 10 *)
    NotMoving                : BOOL; (* StateDWord bit 2 *)
    Compensating             : BOOL; (* StateDWord bit 13 *-* superposition - overlayed motion *)

    ExtSetPointGenEnabled    : BOOL; (* StateDWord bit 14 *)
    PhasingActive            : BOOL; (* StateDWord bit 15 *)
    ExternalLatchValid       : BOOL; (* StateDWord bit 16 *)
    CamDataQueued            : BOOL; (* StateDWord bit 22 *)
    CamTableQueued           : BOOL; (* StateDWord bit 21 *)
    CamScalingPending        : BOOL; (* StateDWord bit 23 *)
    CmdBuffered              : BOOL; (* StateDWord bit 24 *)
    PTPmode                  : BOOL; (* StateDWord bit 25 *)
    DriveDeviceError         : BOOL; (* StateDWord bit 28 *)
    IoDataInvalid            : BOOL; (* StateDWord bit 30 *)
    ErrorPropagationDelayed  : BOOL; (* StateDWord bit 6 *)
    DriveLimitActive         : BOOL; (* StateDWord bit 18 *)

    (* Axis.NcToPlc.CoupleState *)
    Coupled                  : BOOL;

    (* axis operation mode feedback from NcToPlc *)
    OpMode                   : ST_AxisOpModes;
```

```
    NcApplicationRequest     : BOOL; (* OpModeDWord bit 23 *)
END_STRUCT
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

7.11.14 ST_DriveAddress

Dieser Datentyp enthält die ADS-Zugriffdaten eines Antriebsgerätes. Die Daten werden mit dem Funktionsbaustein MC_ReadDriveAddress [▶ 55] gelesen.

```
TYPE ST_DriveAddress :
STRUCT
    NetID               : T_AmsNetId;     (* AMS NetID of the drive as a string *)
    NetIdBytes          : T_AmsNetIdArr;  (* AMS NetID of the drive as a byte array (same informatio
n as NetID) *)
    SlaveAddress        : T_AmsPort;      (* slave address of the drive connected to a bus master *)
    Channel             : BYTE;           (* channel number of the drive *)
    (* new since 2013-04-04 - just available with versions after this date, otherwise zero *)
    NcDriveId           : DWORD;          (* ID [1..255] of the NC software drive of an axis *)
    NcDriveIndex        : DWORD;          (* index [0..9] of the NC software drive of an axis *)
    NcDriveType         : E_NcDriveType;  (* type enumeration of the NC software drive of an axis *)
    NcEncoderId         : DWORD;          (* ID [1..255] of the NC software encoder of an axis *)
    NcEncoderIndex      : DWORD;          (* index [0..9] of the NC software encoder of an axis *)
    NcEncoderType       : E_NcEncoderType;(* type enumeration of the NC encoder drive of an axis *)
    NcAxisId            : DWORD;          (* ID [1..255] of the NC axis *)
    NcAxisType          : E_NcAxisType;   (* type enumeration of the NC axis *)
    (* new since 2016-04-11 - just available with versions after this date, otherwise zero *)
    TcDriveObjectId     : DWORD;
    TcEncoderObjectId   : DWORD;
    TcAxisObjectId      : DWORD;
END_STRUCT
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

7.11.15 ST_McOutputs

```
TYPE ST_McOutputs
STRUCT
    Done            : BOOL;
    Busy            : BOOL;
    Active          : BOOL;
    CommandAborted : BOOL;
    Error           : BOOL;
    ErrorID         : UDINT;
END_STRUCT
END_TYPE
```

7.11.16 ST_NcApplicationRequest

```
TYPE ST_NcApplicationRequest
STRUCT
    bApplRequestBit     : UINT;
    nApplRequestType    : UINT;
    nApplRequestCounter : UDINT;
    nApplRequestVersion : UDINT;
END_STRUCT
END_TYPE
```

7.11.17 ST_SetEncoderScalingOptions

```
TYPE ST_SetEncoderScalingOptions
STRUCT
    SelectEncoderIndex : BOOL;
    EncoderIndex       : UINT;
END_STRUCT
END_TYPE
```

7.11.18 ST_SetPositionOptions

Dieser Datentyp enthält die optionalen Einstellungen des Funktionsbausteins MC_SetPosition [▶ 20].

```
TYPE ST_SetPositionOptions
STRUCT
    ClearPositionLag    : BOOL;
    SelectEncoderIndex  : BOOL;
    EncoderIndex        : UINT;
    ClearPositionOffset : BOOL;
END_STRUCT
END_TYPE
```

ClearPositionOffset - Löschen des MC_SetPosition-Offsets

Der durch den MC_SetPosition [▶ 20]-Aufruf aufsummierte Positionsoffset kann mit diesem Funktionsbaustein gelöscht werden. Dazu wird in den Optionen der Schalter ClearPositionOffset gesetzt; die am Baustein übergebene Position ist in diesem Fall nicht relevant.

Diese Option steht ab TwinCAT 3.1.4024.51 und Tc2_MC2 3.3.56 zur Verfügung.

```
VAR
    mcSetPositionClear: MC_SetPosition;
END_VAR
```

```
mcSetPositionClear.Options.ClearPositionOffset := TRUE;
mcSetPositionClear (
    Axis:= Axis,
    Position:= , // not relevant
    Execute:= TRUE);
```

Ein Homing der Achse, z. B. durch MC_Home, führt zu einem neu referenzierten Koordinatensystem der Achse und löscht den Offset ebenfalls ab.

7.11.19 ST_NcPositionConversionOptions

Dieser Datentyp enthält die optionalen Einstellungen für die Funktionsbausteine MC_ConvertIncPosToPos [▶ 64] und MC_ConvertPosToIncPos [▶ 65].

```
TYPE ST_NcPositionConversionOptions
STRUCT
    SubIndex   : UINT;
    CtrlMask   : UINT;
END_STRUCT
END_TYPE
```

7.12 Stepper

7.12.1 E_DestallDetectMode

```
TYPE E_DestallDetectMode :
(
    PwStDetectMode_None := 0,
    PwStDetectMode_Encoderless,
    PwStDetectMode_Lagging(* use encoder signals for stall detection *)
);
END_TYPE
```

E_DestallDetectMode	Beschreibung
PwStDetectMode_None	Keine Blockiererkennung
PwStDetectMode_Encoderless	Verwendung von Statussignalen der KL2531/KL2541 zur Blockiererkennung
PwStDetectMode_Lagging	Verwendung von Encodersignalen zur Blockiererkennung

7.12.2 E_DestallMode

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

7.12.3 ST_PowerStepperStruct

```
TYPE ST_PowerStepperStruct :
STRUCT
    DestallDetectMode   : E_DestallDetectMode;
    DestallMode         : E_DestallMode;
    DestallEnable       : BOOL;
    StatusMonEnable     : BOOL;
    Retries             : INT;
    Timeout             : TIME;
END_STRUCT
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

7.13 Superposition

7.13.1 E_SuperpositionMode

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

E_SuperpositionMode legt fest, wie eine überlagerte Bewegung mit dem Funktionsbaustein MC_MoveSuperImposed [▶ 92] durchgeführt wird.

Die mit „Veloreduction“ benannten Modi führen die überlagerte Bewegung mit einer möglichst geringen Geschwindigkeitsänderung durch und nutzen bevorzugt die gesamte parametrierte Ausgleichstrecke. Umgekehrt nutzen die mit „Lengthreduction“ benannten Modi die maximal mögliche Geschwindigkeitsänderung und verkürzen damit die benötigte Fahrstrecke. In beiden Fällen wird gleiche Distanz ausgeglichen.

In den mit „Additivemotion“ benannten Fällen führt die überlagerte Achse eine längere oder kürzere Bewegung aus, als durch „Length“ angegeben, da die überlagerte Distanz hinzukommt. Diese Modi werden beispielsweise verwendet, wenn sich der Length-Parameter auf eine Vergleichsachse bezieht und die überlagerte Achse im Vergleich dazu eine längere oder kürzere Fahrtstrecke zurücklegen darf.

In den mit „Limitedmotion“ benannten Fällen, wird die Überlagerung innerhalb der parametrierten Distanz abgeschlossen. Diese Modi werden beispielsweise verwendet, wenn sich der Length-Parameter auf die überlagerte Achse selbst bezieht. Bei diesen Modi ist zu beachten, dass die überlagerte Distanz deutlich kürzer sein muss als die zur Verfügung stehende Fahrstrecke „Length“.

E_SuperpositionMode	Beschreibung
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 Geschwindigkeitsänderung „VelocityDiff“ reduziert. Die Länge „Length“ bezieht sich auf eine Vergleichsachse ohne überlagerte Bewegung (beispielsweise Masterachse). 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 Geschwindigkeitsänderung „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 Geschwindigkeitsänderung „VelocityDiff“ noch die maximale Strecke „Length“ überschritten. Die Länge „Length“ bezieht sich auf eine Vergleichsachse 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 Geschwindigkeitsänderung „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.
SUPERPOSITIONMODE_ACCREDUCTION_ADDITIVEMOTION **(ab TwinCAT 2.11)**	Die überlagerte Bewegung wird über die gesamte Strecke „Length“ durchgeführt. Um die Distanz „Distance“ auf dieser Strecke zu erreichen, wird die vorgegebene maximale Beschleunigung „Acceleration“ bzw. „Deceleration“ soweit wie möglich reduziert. Die Länge „Length“ bezieht sich auf eine Vergleichsachse ohne überlagerte Bewegung (beispielsweise Masterachse). Die Achse auf die die Ausgleichsfahrt wirkt, legt die Strecke „Length + Distance“ zurück.
SUPERPOSITIONMODE_ACCREDUCTION_LIMITEDMOTION **(ab TwinCAT 2.11)**	Die überlagerte Bewegung wird über die gesamte Strecke „Length“ durchgeführt. Um die Distanz „Distance“ auf dieser Strecke zu erreichen, wird die vorgegebene maximale Beschleunigung „Acceleration“ bzw. „Deceleration“ soweit wie möglich reduziert.

E_SuperpositionMode	Beschreibung
	Die Länge „Length“ bezieht sich auf die Achse, auf die die Ausgleichsfahrt wirkt. Diese legt während der Ausgleichsfahrt die Strecke „Length“ zurück.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

7.13.2 E_SuperpositionAbortOption

```
TYPE E_SuperpositionAbortOption :
(
    SUPERPOSITIONOPTION_ABORTATSTANDSTILL := 0,
    SUPERPOSITIONOPTION_RESUMEAFTERSTANDSTILL,
    SUPERPOSITIONOPTION_RESUMEAFTERMOTIONSTOP
) UINT;
END_TYPE
```

Beschreibung, siehe ST_SuperpositionOptions [▶ 159].

7.13.3 ST_SuperpositionOptions

```
TYPE ST_SuperpositionOptions :
STRUCT
   AbortOption : E_SuperpositionAbortOption;
END_STRUCT
END_TYPE
```

```
TYPE E_SuperpositionAbortOption :
 (
    SUPERPOSITIONOPTION_ABORTATSTANDSTILL := 0,
    SUPERPOSITIONOPTION_RESUMEAFTERSTANDSTILL,
    SUPERPOSITIONOPTION_RESUMEAFTERMOTIONSTOP
 );
END_TYPE
```

AbortOption ist ein optionaler Parameter des Funktionsbausteins MC_MoveSuperImposed [▶ 92], der das Verhalten einer überlagerten Bewegung bei einem Stillstand der Hauptbewegung festlegt.

AbortOption	Beschreibung
SUPERPOSITIONOPTION_ABORTATSTANDSTILL	Die überlagerte Bewegung wird abgebrochen, sobald die unterlagerte Bewegung zu einem Stillstand der Achse führt. Einzige Ausnahme ist ein Stillstand, der durch einen Geschwindigkeits-Override von Null herbeigeführt wird. In diesem Fall wird auch die überlagerte Bewegung fortgesetzt, sobald der Override ungleich Null ist. AbortAtStandstill ist das Standardverhalten, falls die Option vom Anwender nicht belegt wird.
SUPERPOSITIONOPTION_RESUMEAFTERSTANDSTILL	Die überlagerte Bewegung wird bei einem temporären Stillstand der Hauptbewegung nicht abgebrochen, sondern wird fortgesetzt, sobald sich die Achse wieder bewegt. Dieser Fall kann insbesondere bei einer Richtungsumkehr oder bei Kurvenscheibenbewegungen eintreten. Erst wenn die Zielposition der Achse erreicht ist oder die Achse gestoppt wurde, wird auch die überlagerte Bewegung beendet.
SUPERPOSITIONOPTION_RESUMEAFTERMOTIONSTOP	Die überlagerte Bewegung wird bei einem Stillstand der Hauptbewegung nicht abgebrochen, auch wenn die Achse ihre Zielposition erreicht hat oder gestoppt wurde. In diesem Fall wird die überlagerte Bewegung nach einem erneuten Start der Achse fortgesetzt.

AbortOption	Beschreibung
	Dieser Fall ist nicht von Bedeutung, falls die überlagerte Bewegung auf eine Slaveachse angewendet wird, da diese nicht aktiv gestartet oder gestoppt werden kann. Bei Slaveachsen sind die Betriebsarten RESUMEAFTERSTANDSTILL und RESUMEAFTERMOTIONSTOP gleichwertig. Die überlagerte Bewegung würde also auch nach einem erneuten Start der Master-Achse fortgesetzt.

Übersicht über die Abbruchbedingungen einer überlagerten Bewegung (MC_MoveSuperImposed)

	ABORTATSTAND STILL	RESUMEAFTER STANDSTILL	RESUMEAFTER MOTIONSTOP
1. Override = 0%	wird fortgesetzt	wird fortgesetzt	wird fortgesetzt
2. temporärer Stillstand der Hauptbewegung	Abbruch	wird fortgesetzt	wird fortgesetzt
3. Bewegungsumkehr	Abbruch	wird fortgesetzt	wird fortgesetzt
4. Achse hat Zielposition erreicht oder wird gestoppt	Abbruch	Abbruch	wird fortgesetzt
5. Achs-Reset oder Abschalten des Enable-Signals	Abbruch	Abbruch	Abbruch
6. Bei Slave-Achsen: Abkoppeln	Abbruch	Abbruch	Abbruch

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

7.14 Torque Control

7.14.1 ST_TorqueControlOptions

Dieser Datentyp enthält optionale Einstellungen für MC_TorqueControl.

```
TYPE ST_TorqueControlOptions :
STRUCT
    EnableManualTorqueStartValue : BOOL;
    ManualTorqueStartValue       : LREAL;
END_STRUCT
END_TYPE
```

ST_TorqueControlOptions	Beschreibung
EnableManualTorqueStartValue	Das Flag kann aktiviert werden, um einen vom aktuellen Torque abweichenden Startwert vorzugeben.
ManualTorqueStartValue	Startwert für den Torque, der bei aktiver Flag EnableManualTorqueStartValue verwendet wird.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.1.4024	PC oder CX (x86)	Tc2_MC2

7.15 Touch Probe

7.15.1 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

7.15.2 E_SignalSource

```
TYPE E_SignalSource :
(
    SignalSource_Default,     (* undefined or externally configured *)
    SignalSource_Input1,      (* digital drive input 1 *)
    SignalSource_Input2,      (* digital drive input 2 *)
    SignalSource_Input3,      (* digital drive input 3 *)
    SignalSource_Input4,      (* digital drive input 4 *)
    SignalSource_ZeroPulse := 128, (* encoder zero pulse *)
    SignalSource_DriveDefined (* defined by drive parameters - e. g. CAN object 0x60D0 *)
) UINT;
END_TYPE
```

Name	Beschreibung
SignalSource	Definiert optional die Signalquelle, soweit diese über die Steuerung gewählt werden kann. **Beispiele:** • AX5000 (SoE): SignalSource_Default • AX8000 (CoE DS402): SignalSource_DriveDefined

7.15.3 MC_TouchProbeRecordedData

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

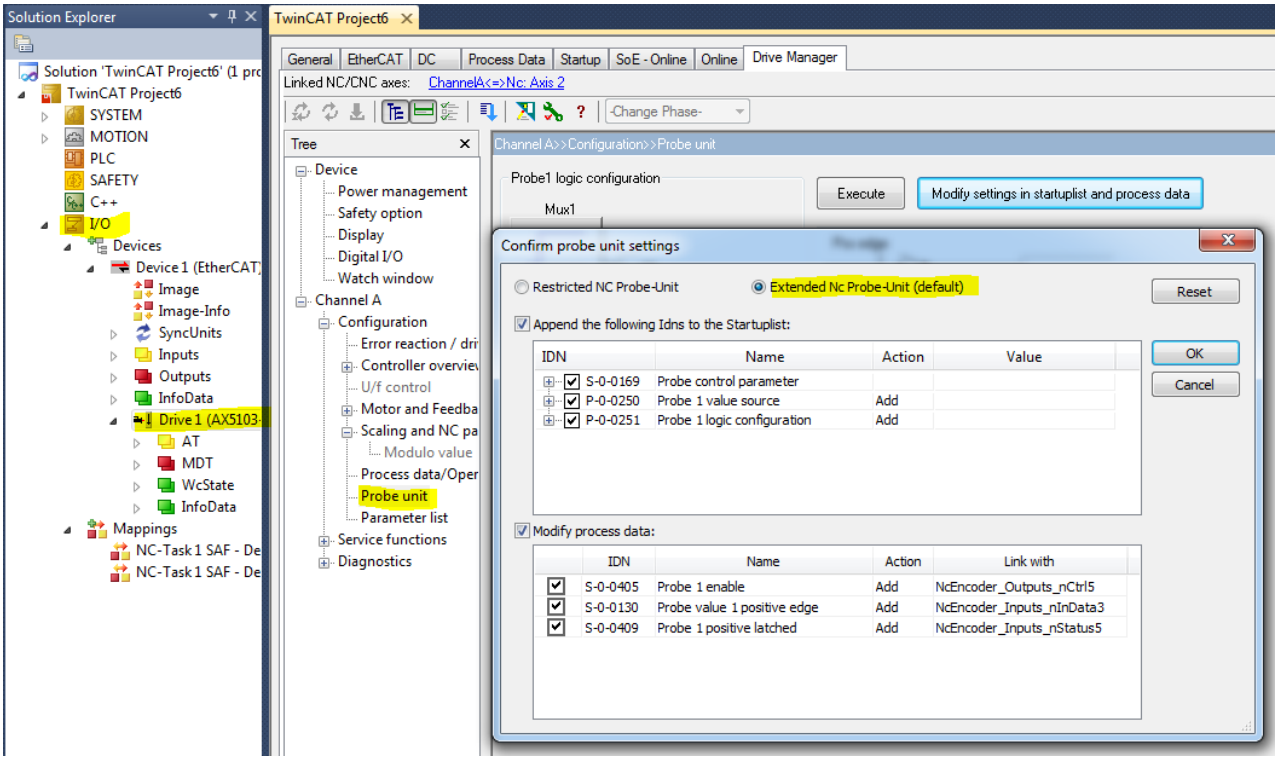
MC_TouchProbeRecordedData	Datentyp	Beschreibung
Counter	LREAL	Zähler, der anzeigt, wie viele gültige Flanken im letzten Zyklus erkannt wurden. Eine Erkennung mehrerer Flanken ist nur im Mode = TOUCHPROBEMODE_CONTINOUS unter SERCOS/SOE implementiert und muss explizit von der Hardware unterstützt werden (z. B. AX5000).

MC_TouchProbeRecordedData	Datentyp	Beschreibung
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.

7.15.4 TRIGGER_REF

```
TYPE TRIGGER_REF :
STRUCT
    EncoderID       : UDINT; (* 1..255 *)
    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 continous 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
```

Name	Datentyp	Beschreibung
EncoderID	UDINT	Die ID des Encoders kann im TwinCAT System Manager abgelesen werden.
TouchProbe	E_TouchProbe [▶ 163]	Definiert die verwendete Latch-Einheit (Probe-Unit) innerhalb der verwendeten Encoder-Hardware.
SignalSource	E_SignalSource [▶ 161]	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.
Edge	E_SignalEdge [▶ 161]	Edge legt fest, ob die steigende oder fallende Flanke des Trigger-Signals ausgewertet wird.
Mode	E_TouchProbeMode [▶ 163]	Legt die Betriebsart der Latch-Einheit fest. Im Single-Mode wird nur die erste Flanke erfasst. Im Continous-Mode wird jede Flanke für einen SPS-Zyklus signalisiert.
PlcEvent	BOOL	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	BOOL	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 [▶ 37] 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.



Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

7.15.5 E_TouchProbeMode

Legt die Betriebsart der Latch-Einheit fest. Im Single-Mode wird nur die erste Flanke erfasst. Im Continous-Mode wird jede Flanke für einen SPS-Zyklus signalisiert.

```
TYPE E_TouchProbeMode :
(    TOUCHPROBEMODE_SINGLE_COMPATIBILITYMODE, (* for TwinCAT 2.10 and 2.11 before Build 2022 *)
    TOUCHPROBEMODE_SINGLE, (* multi probe interface - from 2.11 Build 2022 *)
    TOUCHPROBEMODE_CONTINOUS (* multi probe interface - from 2.11 Build 2022 *)
) UINT;
END_TYPE
```

Für die Modes SINGLE bzw. CONTINOUS muss die Probe Unit als „Extended Nc Probe Unit“ konfiguriert werden.

7.15.6 E_Touch Probe

```
TYPE E_TouchProbe :
(
    TouchProbe1 := 1, (* 1st hardware probe unit with Sercos, CanOpen, KL5xxx and others *)
    TouchProbe2,      (* 2nd probe unit *)
    TouchProbe3,      (* currently not available *)
    TouchProbe4,      (* currently not available *)
    PlcEvent := 10    (* simple PLC signal TRUE/FALSE *)
) UINT;
END_TYPE
```

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_Tc2_MC2 : ST_LibVersion;
END_VAR
```

stLibVersion_Tc2_MC2: Versionsinformation der Tc2_MC2-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).

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

9 Beispiele

Die Beispielprogramme verwenden die Tc2_MC2-Bibliothek und laufen vollständig im Simulationsmodus.

Der Ablauf kann im TwinCAT Scope View mit beiliegender Konfiguration verfolgt werden.

PTP – Punkt-zu-Punkt-Bewegung

Das Beispielprogramm verwaltet und bewegt eine Achse im PTP-Modus. Die Achse wird mit zwei Instanzen eines MC_MoveAbsolute-Funktionsbausteins in der gepufferten Betriebsart über mehrere Zwischenpositionen und Geschwindigkeitsebenen bewegt.

Download: https://infosys.beckhoff.com/content/1031/TcPlcLib_Tc2_MC2/Resources/2386997515.zip

Master-Slave-Kopplung

Das Beispielprogramm koppelt zwei Achsen und verfährt sie gemeinsam. Die Slave-Achse wird während der Fahrt abgekoppelt und positioniert.

Download: https://infosys.beckhoff.com/content/1031/TcPlcLib_Tc2_MC2/Resources/2386995851.zip

Tänzerregelung (Dancer Control)

Das Beispielprogram zeigt, wie die Geschwindigkeit einer Slave-Achse anhand der Position eines Tänzers geregelt werden kann.

Download: https://infosys.beckhoff.com/content/1031/TcPlcLib_Tc2_MC2/Resources/2386992523.zip

Überlagerte Bewegung (Superposition)

Das Beispiel zeigt die Überlagerung einer Bewegung während der Fahrt einer Achse.

Download: https://infosys.beckhoff.com/content/1031/TcPlcLib_Tc2_MC2/Resources/2386999179.zip

Kompensation der Umkehrlose einer Achse

Das Beispielprogramm zeigt, wie die Umkehrlose einer Achse kompensiert werden kann.

Download: https://infosys.beckhoff.com/content/1031/TcPlcLib_Tc2_MC2/Resources/2386989195.zip

Externe Sollwertgenerierung

Das Beispiel zeigt, wie eine Achse über die externe Sollwertgenerierung verfahren werden kann. Dabei wird die Bewegung der Nc-Achse „Axis“ als Summe der Einzelbewegung der anderen beiden NC-Achsen erzeugt.

Download: https://infosys.beckhoff.com/content/1031/TcPlcLib_Tc2_MC2/Resources/2386994187.zip

Regelkreisumschaltung bei einem AX5000 mit zwei vorhandenen Encodern

Das Beispiel zeigt, wie zwischen zwei Regelkreisen einer Achse umgeschaltet werden kann. Für dieses Beispiel wird entsprechende Hardware benötigt.

Download: https://infosys.beckhoff.com/content/1031/TcPlcLib_Tc2_MC2/Resources/2386990859.zip

Schreiben von Outputs im IO-Interface des Enkoder oder Drive einer-Achse

Das Beispiel zeigt, wie die nicht von der NC verwendeten IO-Outputs aus der PLC geschrieben werden können.

Download: https://infosys.beckhoff.com/content/1031/TcPlcLib_Tc2_MC2/Resources/13977016971.zip

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT v3.0.0	PC oder CX (x86 oder x64)	Tc2_MC2

10 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

11 Anhang

11.1 NC Backlash Compensation

11.1.1 Mechanische Lose

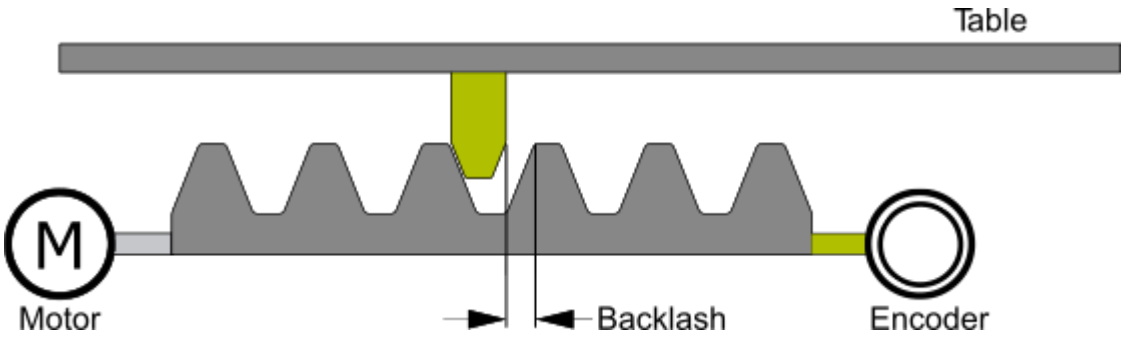
Als mechanische Lose wird das Spiel zwischen Antrieb und einem bewegten Maschinenteil oder zwischen einem Geber und einem bewegten Maschinenteil bezeichnet. Durch die mechanische Lose ergibt sich für ein bewegtes Maschinenteil eine Abweichung zwischen kommandierter Position und der tatsächlichen Istposition. Dies wirkt sich insbesondere bei der Bewegungsrichtungsumkehr aus.

Es wird bei der mechanischen Lose zwischen den nachfolgend erläuterten Arten unterschieden:

Positive Lose

Die positive Lose tritt bei Systemen auf, bei denen das Messsystem direkt mit dem Antrieb verbunden ist und die Lose zwischen Antrieb und bewegtem Maschinenteil auftritt. Bei einer Bewegungsrichtungsumkehr wird das Messsystem eine Positionsänderung detektieren, obwohl sich das Maschinenteil bedingt durch die Lose noch nicht bewegt. Dies führt dazu, dass das Maschinenteil nicht die kommandierte Position erreicht, sondern um den Betrag der Lose zu kurz verfährt, da der Geber, der indirekt die Position des Maschinenteils misst, der tatsächlichen Istposition des Maschinenteils vorauseilt.

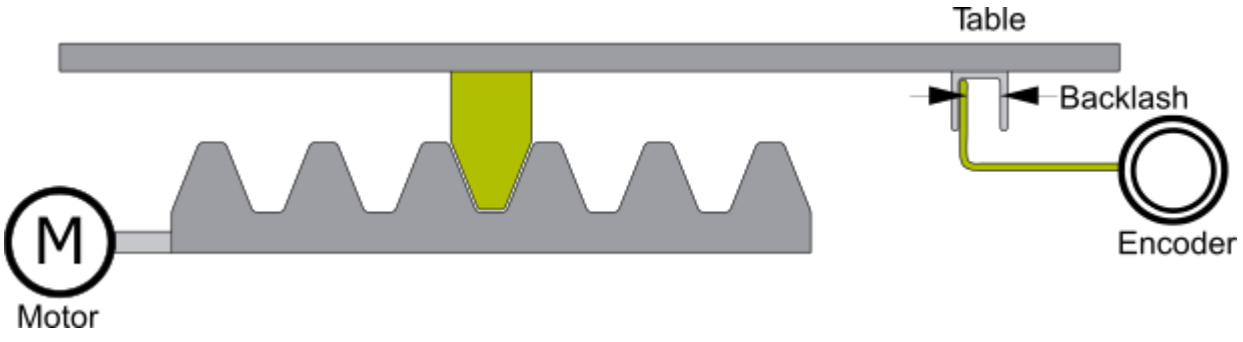
In den folgenden Abbildungen wird eine Bewegung von links nach rechts als positive Fahrtrichtung (Normalfall) festgelegt.



Der Geber eilt dem Maschinenteil (z. B. Tisch) voraus. Da damit auch die vom Geber erfasste Istposition der tatsächlichen Istposition des Tisches vorauseilt, fährt der Tisch zu kurz. Der Lose-Korrekturwert ist hier positiv einzugeben (= Normalfall).

Negative Lose

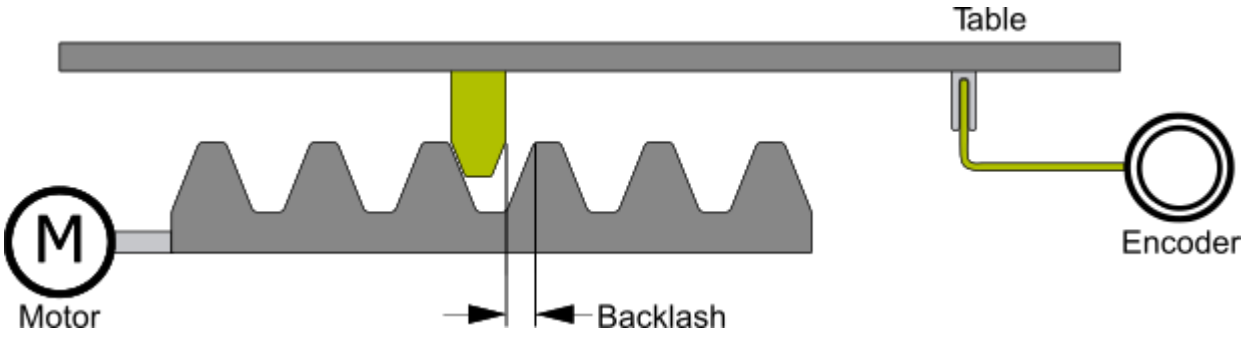
Die negative Lose tritt bei Systemen auf, bei denen die Lose zwischen dem bewegten Maschinenteil und dem Messsystem auftritt. Bei einer Bewegungsumkehr verfährt das Maschinenteil unmittelbar in die neue Richtung, ohne dass das Messsystem eine Positionsänderung detektiert. In diesem Fall verfährt das Maschinenteil um die Lose weiter als durch die Kommandierung erforderlich ist, da der Geber, der die Position des Maschinenteils direkt misst, der tatsächlichen Position des Maschinenteils nacheilt.



Der Geber hinkt dem Maschinenteil (z. B. Tisch) nach; der Tisch fährt zu weit. Der Korrekturwert ist **negativ** einzugeben.

Neutrale Lose

Die „neutrale Lose“ tritt als Sonderfall bei Systemen auf, bei denen das Messsystem am Werkstücktisch befestigt und direkt mit dem Antrieb verbunden ist. Die Lose tritt nun zwischen Antrieb und bewegtem Maschinenteil auf. Bei einer Bewegungsrichtungsumkehr wird diese Lose nun automatisch ausgefahren, da der Positionsregelkreis über das direkte Messsystem am Werkstücktisch geschlossen wird und die kommandierte Position erreicht wird, ohne besondere Maßnahmen der Losekompensation vorzunehmen.



Für die Steuerung sind keinerlei besondere Einstellungen bezüglich Lose nötig, da der Lageregler, bezogen auf das am Werkstück montierte Gebersystem, eine stationäre Genauigkeit erzwingt.

Allgemeine Hinweise und Anmerkungen:

- Bezüglich der Implementierung in TwinCAT existiert keine Unterscheidung zwischen **positiver Lose** und **negativer Lose** (außer dass sich für diese beiden Fälle das Vorzeichen der parametrierten Lose unterscheidet). Eine positive Lose wird als positiver Wert und eine negative Lose als negativer Wert parametriert.

- Der Fall der **negativen Lose** ist aus regelungstechnischer Sicht sehr ungünstig, da eine Achse mit einer Lose im Gebersystem nur sehr schwer regelbar ist („stationäre zyklische Schwingung“). Typischerweise sind zur Lösung dieses Problems weitere Maßnahmen nötig.

- Für all die verschiedenen Varianten der Lose gilt, dass nicht weiter zwischen Positionsinterface (Lageregung im Abtrieb) und Geschwindigkeitsinterface (Lageregelung in TwinCAT) unterschieden werden muss, da sich letztlich die gleiche Wirkung einstellt.

- Im Fall der „neutralen Lose“ sind, obwohl eine mechanische Lose vorliegt, keine weiteren Maßnahmen der Losekompensation nötig, da sich das Gebersystem am Werkstücktisch (Werkstück) befindet und somit eine stationäre Genauigkeit durch die Lageregelung auf diesen ergibt.

- Falls eine Referenzierung (Homing) einer Achse benötigt wird, dann sollte diese bei deaktivierter Losekompensation bzw. deaktivierter Positionskorrektur durchgeführt werden. Die letzte Fahrtrichtung beim Referenzierablauf legt durch das Setzen einer Referenzposition fest, ob als Bezugspunkt die linke oder die rechte Flanke verwendet wird (siehe NC-Implementierung der TwinCAT-Losekompensation (Backlash Compensation) [▶ 170]).

11.1.2 NC Implementierung der TwinCAT-Positionskorrektur

Die Realisierung der **Losekompensation** basiert auf der TwinCAT-Funktion **Positionskorrektur**.

Die nachfolgende Tabelle liefert eine Beschreibung der **TwinCAT-Positionskorrektur** unterschieden nach Antrieben im zyklischen Positions- und Geschwindigkeitsinterface.

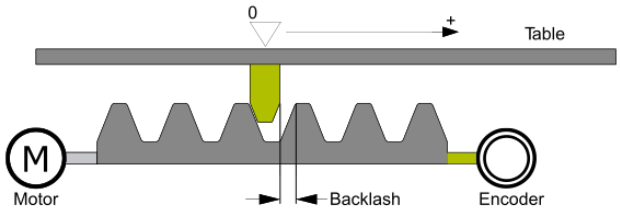
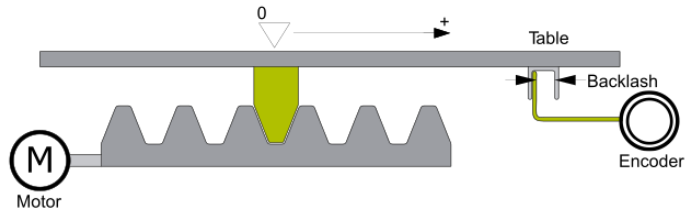
	Implementierung und Wirkung der Positionskorrektur bzw. der Losekompensation
Implementierung	1. Die zum Drive übertragene Drive-Sollposition wird um den Anteil der Positionskorrektur (Lose) subtrahiert. 2. Die vom Encoder bzw. Drive übertragene Ist-Position wird um den Anteil der Positionskorrektur (Lose) addiert.

	Implementierung und Wirkung der Positionskorrektur bzw. der Losekompensation
Wirkung im Positionsinterface (Lageregler im Drive)	Durch die Manipulation der zum Antrieb ausgegeben Drive-Sollposition wird der Bereich der Lose durchfahren (Subtraktion der Positionskorrektur, siehe Fall 1). Damit aber auch die angezeigte Istposition wieder der idealen Position entspricht, wird genau dieser Anteil bei der übertragenen Istposition abgezogen (Addition der Positionskorrektur, siehe Fall 2).
Wirkung im Geschwindigkeitsinterface (Lageregler in TwinCAT)	In der Betriebsart Geschwindigkeitsinterface wird keine Drive-Sollposition zum Antrieb übertragen, somit ist dieser Weg für die Positionskorrektur unwirksam. Durch die Manipulation der vom Encoder bzw. Drive übertragenen Ist-Position wird eine Lageregeldifferenz aufgebaut. Durch den Lageregler in TwinCAT führt diese Lageregeldifferenz („Schleppfehler“) zum Durchfahren der Lose (Addition der Positionskorrektur, siehe Fall 2).

11.1.3 NC-Implementierung der TwinCAT-Losekompensation (Backlash Compensation)

Tabellarische Übersicht der **Wirkung der Losekompensation** unterschieden nach Art der Lose und Wahl der Referenzposition (linke oder rechte Flanke):

- **Positive Lose**: Motorgeber bzw. externer Geber sind OHNE Lose. Losewert wird positiv vorgegeben.
- **Negative Lose:** Zusätzlicher externer Geber MIT Lose. Losewert wird negativ vorgegeben.

	Implementierung und Wirkung der Losekompensation
Referenzposition an der linken Flanke 	**Losekompensation findet in negativer Fahrtrichtung statt** **Positive Fahrtrichtung:** • Keinerlei Manipulation **Negative Fahrtrichtung:** • Manipulation der Ist-Position um +Lose • Manipulation der Soll-Position um - Lose
Referenzposition an der rechten Flanke 	**Losekompensation findet in positiver Fahrtrichtung statt** **Positive Fahrtrichtung:** • Manipulation der Ist-Position um - *Lose* • Manipulation der Soll-Position um +*Lose* **Negative Fahrtrichtung:** • Keinerlei Manipulation

Trademark statements

Beckhoff®, ATRO® , EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, MX-System®, Safety over EtherCAT®, TC/BSD®, TwinCAT®, TwinCAT/BSD®, TwinSAFE®, XFC®, XPlanar® and XTS® are registered and licensed trademarks of Beckhoff Automation GmbH.

Third-party trademark statements

BiSS is a trademark of IC Haus GmbH.

EnDat is a trademark of Dr. Johannes Heidenhain GmbH.

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