BUCHER automation

Application Note 057



60886446_01

MCX-Achsen

Dieses Dokument wurde von der Bucher Automation AG mit der gebotenen Sorgfalt und basierend auf dem ihr bekannten Stand der Technik erstellt. Änderungen und technische Weiterentwicklungen an unseren Produkten werden nicht automatisch in einem überarbeiteten Dokument zur Verfügung gestellt.
Die Bucher Automation AG übernimmt keine Haftung und Verantwortung für inhaltliche oder formale Fehler, fehlende Aktualisierungen sowie daraus eventuell entstehende Schäden oder Nachteile.

Bucher Automation AG
Thomas-Alva-Edison-Ring 10
71672 Marbach am Neckar, Deutschland
T +49 7141 2550-0
info@bucherautomation.com

Technischer Support
T +49 7141 2550-444
support@bucherautomation.com

Vertrieb
T +49 7141 2550-663
sales@bucherautomation.com

www.bucherautomation.com

Originaldokument

Dokumentenversion	1.01
Ausgabedatum	22.06.2026

Inhaltsverzeichnis

1 Voraussetzungen 5

2 Arten von Achsen 6

2.1 Reale Achse 6

2.2 Simulationsachse 6

2.3 Virtuelle Achse 7

2.4 Externe Achse 7

2.5 Schattenachse/Mitlaufachse 8

3 Mechanische Ausführung 9

3.1 Linear - normal 9

3.2 Linear – modulo, Richtung: bidirektional 9

3.3 Linear – modulo, Richtung: unidirektional positiv 9

3.4 Linear – modulo, Richtung: unidirektional negativ 9

3.5 Rotatorisch - normal 9

3.6 Rotatorisch - modulo, Richtung: bidirektional 10

3.7 Rotatorisch - modulo, Richtung: unidirektional positiv 10

3.8 Rotatorisch - modulo, Richtung: unidirektional negativ 10

4 Mechanische Maßeinheiten 11

5 Anwenden der Achsen 12

5.1 Anlegen eines MCX-Projekts 12

5.2 Reale Achse 13

5.2.1 Hinzufügen eines Servoverstärkers 13

5.2.2 Konfigurieren einer realen Achse 14

5.2.3 Erweiterte Achsparameter 15

5.2.4 Mechanische Parameter 15

5.2.5 Achsinbetriebnahme 16

5.2.6 Manueller Betrieb 16

5.3 Simulationsachse 20

5.4 Virtuelle Achsen 21

5.5 Externe Achse 23

5.5.1 JC-440Ext mit Ethercat®-Drives 23

5.5.2 JC-365MC 24

5.5.3 Parametrierung 26

5.5.4 Einstellen des Gebertyps 27

5.6 Mitlaufachse 28

5.6.1 MotionSetup-Status 29

6 Programmieren von Achsen 30
6.1 Parametereinstellungen der Rampen 30
6.1.1 Rampentyp 30
6.1.2 Rampenunterbrechung 31
6.2 Freigeben/Sperren von Achsen 32
6.2.1 Power.Enable(), Power.Disable(); 32
6.2.2 Drive.IsReadyForOperation 32
6.2.3 Power.QuickStop() 32
6.3 Referenzieren 33
6.3.1 Referenz setzen 33
6.3.2 Referenzfahrt 33
6.4 Positionierung 36
6.4.1 Rampenstatus 36
6.4.2 Punkt-zu-Punkt (MovePtp) 38
6.4.3 PostioningMode - Positionierarten 40
6.4.4 Endlos-Positionierung (MoveVelocity) 42
6.4.5 Anhalten (MoveHalt) 44

1 Voraussetzungen

Verwendung

- JetSym 5.3.6
- In der Beschreibung wird eine Maus mit Rechtshänder-Konfiguration angenommen
  - Untermenü öffnen: Rechter Mausklick
  - Doppelklick: Zweimaliges schnelles Klicken mit der linken Maustaste
  - Auswahl: Einmaliger Klick auf die linke Maustaste

Benennung von Achsen und Verbunden

- Der Benutzer ist bei der Benennung der Achsen und Verbunden grundsätzlich frei
- In dieser Application Note werden zur Benennung die folgenden Präfixe verwendet

Parameter	Präfix
Reale Achsen	x
Virtuelle Achsen	xv
Schatten-/Mitlaufachsen	xs
Technologieverbund	tec
Bahnverbund	geo

Tab. 1: Verwendete Präfixe

- Präfixe haben keine logische Auswirkung, zeigen jedoch dem Leser eines Programms schnell den Typ des verwendeten Objekts an.

Sicherheitshinweise

⚠ WARNUNG



Mögliche Personen- oder Sachschäden durch Nichtbeachtung der Sicherheitshinweise

Bei der Anwendung, Inbetriebnahme, Programmierung bzw. dem Testen von Achsen können gefährliche Situationen, sei es durch bewegte Mechanik aber auch elektrische Spannungen an Motor und Servoverstärkern auftreten.

► Lesen und befolgen Sie die im Sicherheitskapitel aufgezeigten Sicherheitsvorkehrungen und Nutzungsbeschränkungen der Servoverstärker und Steuerungskomponenten.

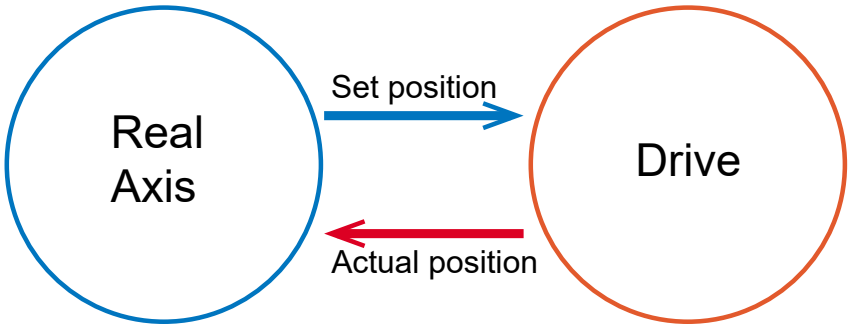
2 Arten von Achsen

Im MotionControl-Kernel können 5 Arten von Achsen definiert werden:

- Reale Achse
- Simulationsachse
- Virtuelle Achse
- Externe Achse
- Schattenachse/Mitlaufachse

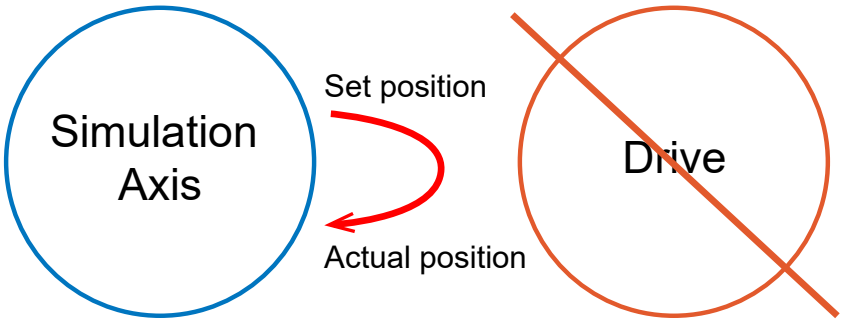
2.1 Reale Achse

- Eine reale Achse ist – wie der Name schon sagt – physikalisch vorhanden.
- Über einen Systembus sind Servoverstärker angeschlossen, an die die Sollwerte und Kommandos übertragen werden.
- Gleichzeitig werden Istwerte vom Servoverstärker an die MotionControl gesendet.
- Die Sollwerte werden in jedem Zyklus des MotionControl-Kernels berechnet.



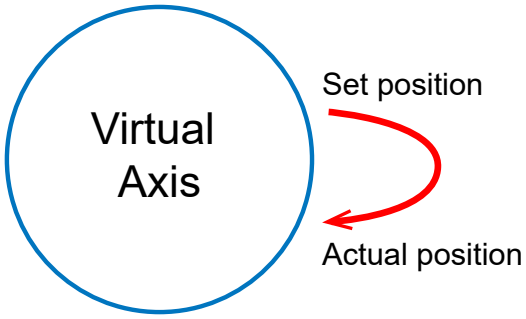
2.2 Simulationsachse

- Eine Simulationsachse ist die virtuelle Repräsentation einer realen Achse.
- Da bei der Anwendungsentwicklung nicht unbedingt die benötigte Hardware an die Steuerung angeschlossen ist, können reale Achsen zu Simulationsachsen umkonfiguriert werden.
- Damit können die Abläufe auch ohne Servoverstärker in Betrieb genommen werden.



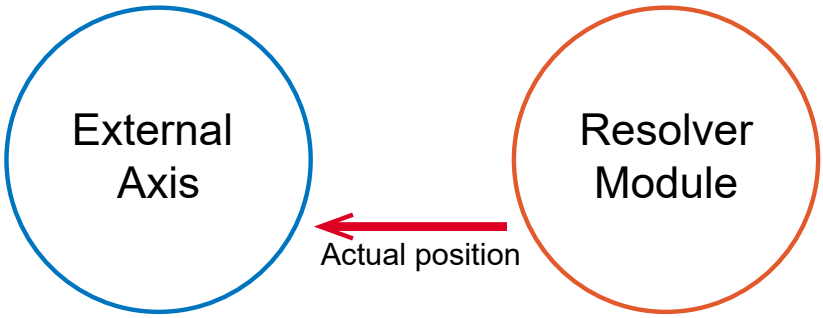
2.3 Virtuelle Achse

– Eine virtuelle Achse wird nur auf dem MotionControl-Kernel verwendet und hat keine Anbindung an einen Servoverstärker.
– Oft bildet sie eine Leitachse in einem Technologieverbund.



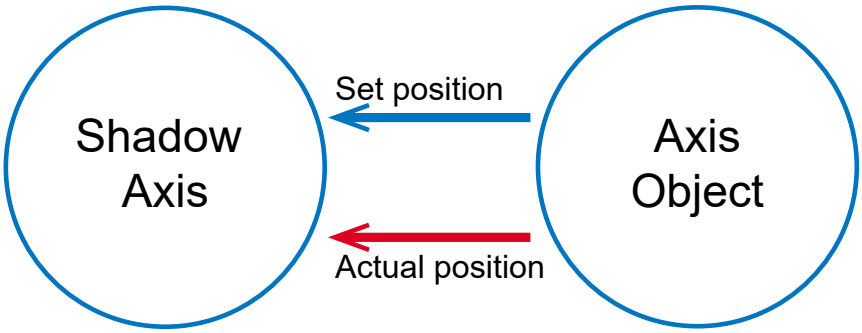
2.4 Externe Achse

– Für Anwendungen, bei denen z.B. die Position eines Schleifrads benötigt wird, verwendet man eine externe Achse.
– Diese verwendet meist ein Gebermodul oder einen zusätzlichen Gebereingang eines Servoverstärkers.
– Hierbei werden nur die vom Modul gesendeten Istwerte ausgewertet.
– Es werden keine Sollwerte für diese Achse generiert.
– Der Betriebszustand für diese Achse ist immer „inaktiv“!

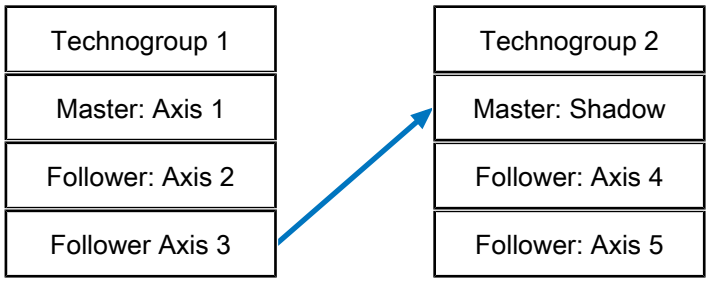


2.5 Schattenachse/Mitlaufachse

– Eine Mitlaufachse ist ein Mix aus externer Achse und interner Achse.
– Die Mitlaufachse wird mit einem Achsobjekt eines anderen Typs verknüpft.
– Somit erbt diese Achse die Soll- und Istwerte der verknüpften Achse.



– Eine Mitlaufachse ist immer eine Leitachse in einem Technologieverbund.
– Verwendet wird diese Achse, wenn die Quellachse nicht Leitachse eines Technologieverbundes sein kann, z.B. weil diese Achse selbst eine Folgeachse in einem anderen Verbund ist oder diese unabhängig von einem Technologieverbund verwendet werden soll.



– In diesem Beispiel ist Achse 3 Folgeachse in einem Technologieverbund.
– Da aber Achse 4 und 5 Achse 3 folgen sollen, wird eine Mitlaufachse (Achse 6) verwendet, die die Repräsentation der Achse 3 darstellt.
– Da eine Achse immer nur in einem Verbund aktiv sein kann (sie kann Mitglied in mehreren Verbunden sein, aber zu einer Zeit immer nur in einem Verbund aktiv sein), kann Achse 3 unabhängig von Verbund 2 bedient werden.

3 Mechanische Ausführung

Bei der Anwendung und Programmierung der Achsen wird hauptsächlich auf den Arbeitspunkt geschaut. Daher muss der MotionControl-Kernel wissen, wie die Mechanik aufgebaut ist.

3.1 Linear - normal



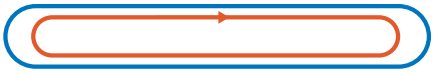
Eine lineare normale Mechanik hat eine geradlinige Bewegung und einen Anfang und Ende wie z.B. bei Linearachsen.

3.2 Linear – modulo, Richtung: bidirektional



Eine lineare Mechanik hat eine geradlinige Bewegung und kann endlos bewegt werden, wie z.B: bei einem Transportband.

3.3 Linear – modulo, Richtung: unidirektional positiv



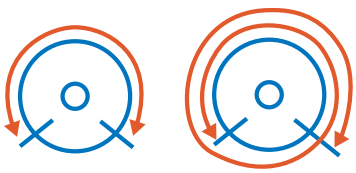
Die lineare Mechanik darf nur in positiver Richtung bewegt werden.

3.4 Linear – modulo, Richtung: unidirektional negativ



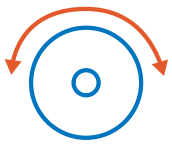
Die lineare Mechanik darf nur in negativer Richtung bewegt werden.

3.5 Rotatorisch - normal



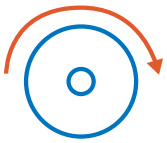
Eine sich im Kreis drehende Mechanik mit einem begrenzten Weg. Der Verfahrweg ist nicht auf den Modulobereich beschränkt.

3.6 Rotatorisch - modulo, Richtung: bidirektional



Eine sich im Kreis endlos drehende Mechanik, die in beide Richtungen bewegt werden darf.

3.7 Rotatorisch - modulo, Richtung: unidirektional positiv



Eine sich im Kreis endlos drehende Mechanik, die nur in positiver Richtung bewegt werden darf.

3.8 Rotatorisch - modulo, Richtung: unidirektional negativ



Eine sich im Kreis endlos drehende Mechanik, die nur in negativer Richtung bewegt werden darf.

4 Mechanische Maßeinheiten

Für jede Achse lässt sich separat die verwendete Maßeinheit angeben.

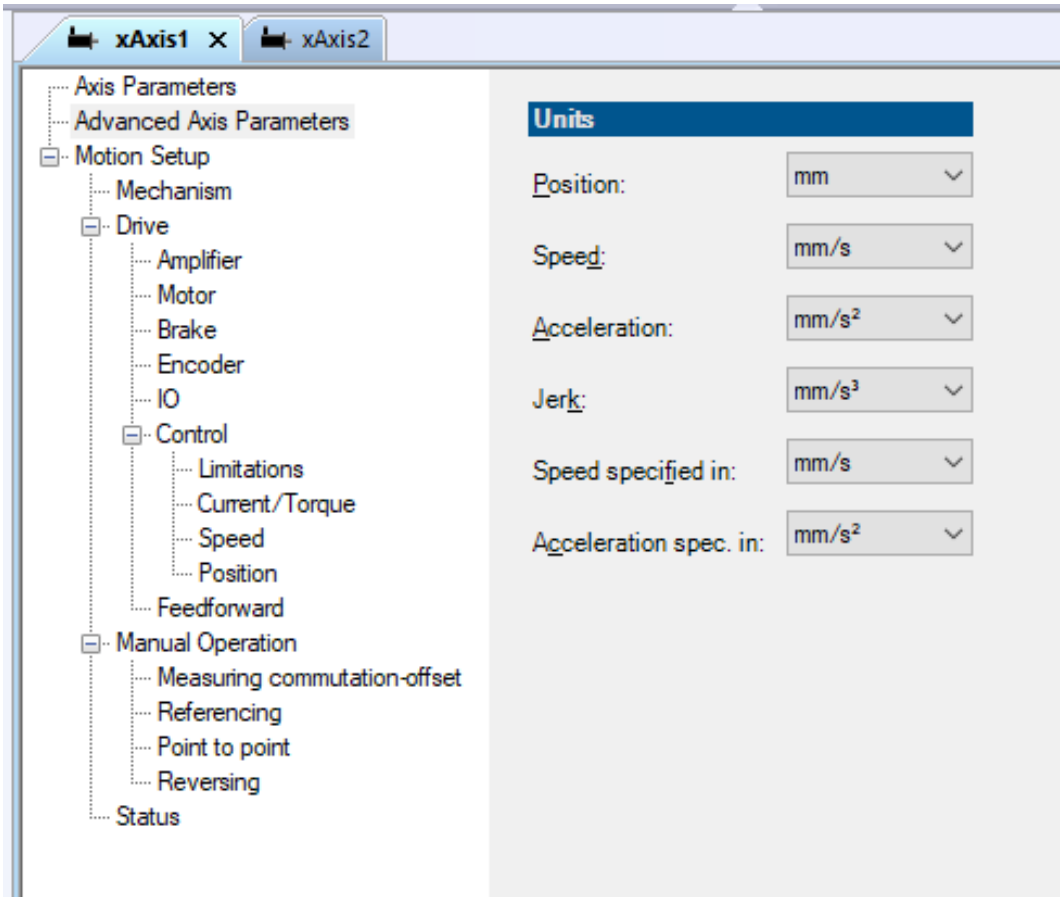


Abb. 1: Mech. Maßeinheiten einstellen

Diese Einheiten bilden die Grundlage, wie die entsprechenden Werte durch die MotionControl interpretiert werden.

Beispiel

Eingestellte Einheit	Wert	Geschwindigkeit
mm/s	100	100 mm/s
m/s	100	100 000 mm/s

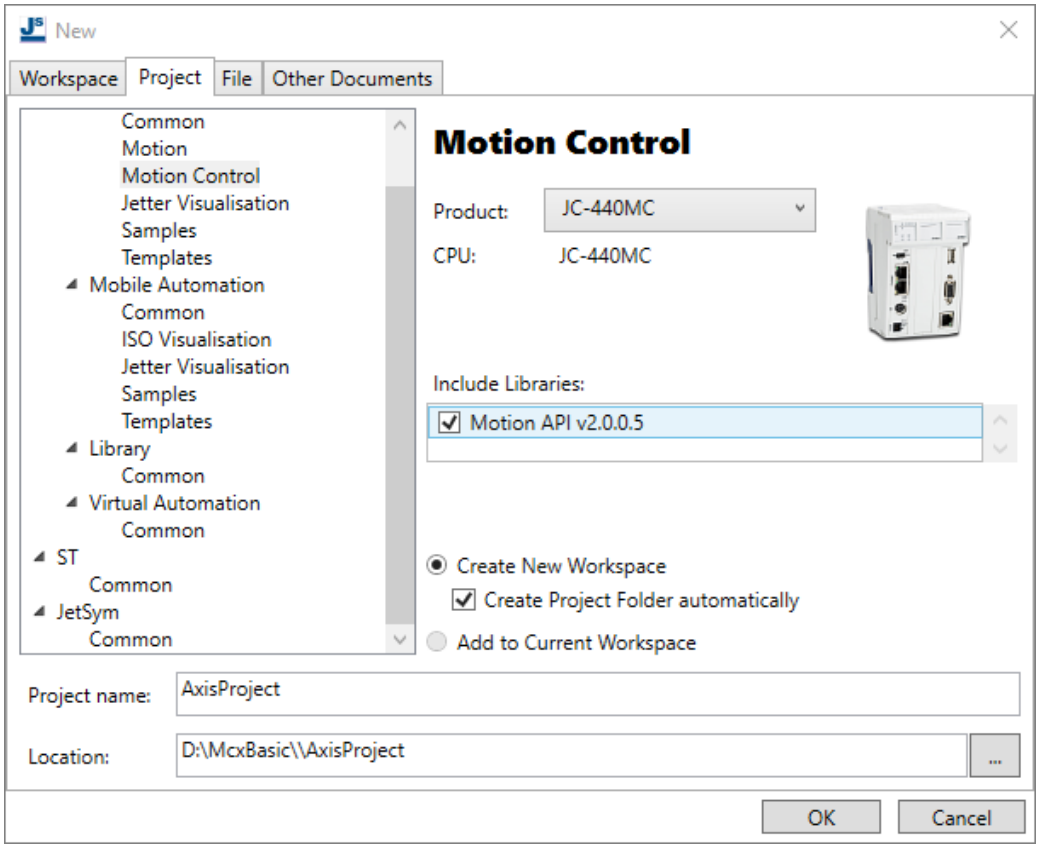
Es empfiehlt sich, die Standard-SI-Einheiten beizubehalten.

5 Anwenden der Achsen

5.1 Anlegen eines MCX-Projekts

Als Grundlage für die Darstellung der einzelnen Achstypen erstellen wir ein Projekt, dem wir nach und nach die unterschiedlichen Achsen hinzufügen bzw. diese konfigurieren.

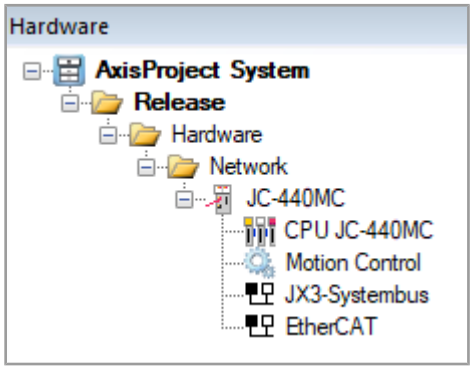
1. Öffnen Sie In JetSym über Datei->Neu den Projekt-Wizard.



2. Wählen Sie eine MCX-Steuerung aus und binden Sie die MotionAPI ein.
   Es wird immer die aktuell installierte und zur Steuerung passende Version vorgeschlagen.
3. Bestätigen Sie mit OK.

   ⇨ Der Projektbaum wird erzeugt.

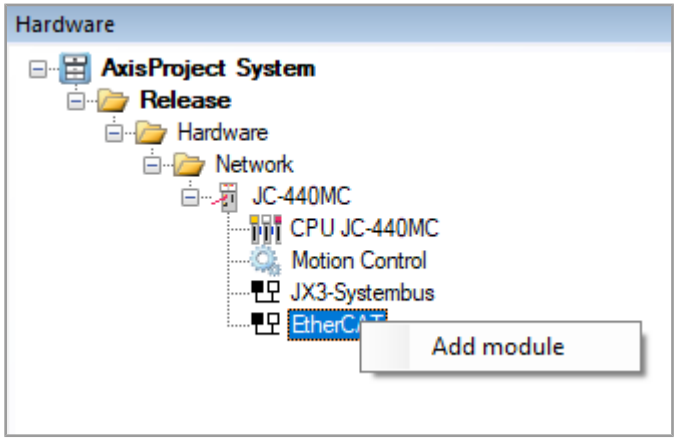
⇨ Im Hardwaremanager sind nun die Steuerung und die Basis-Knoten zur weiteren Konfiguration angelegt.



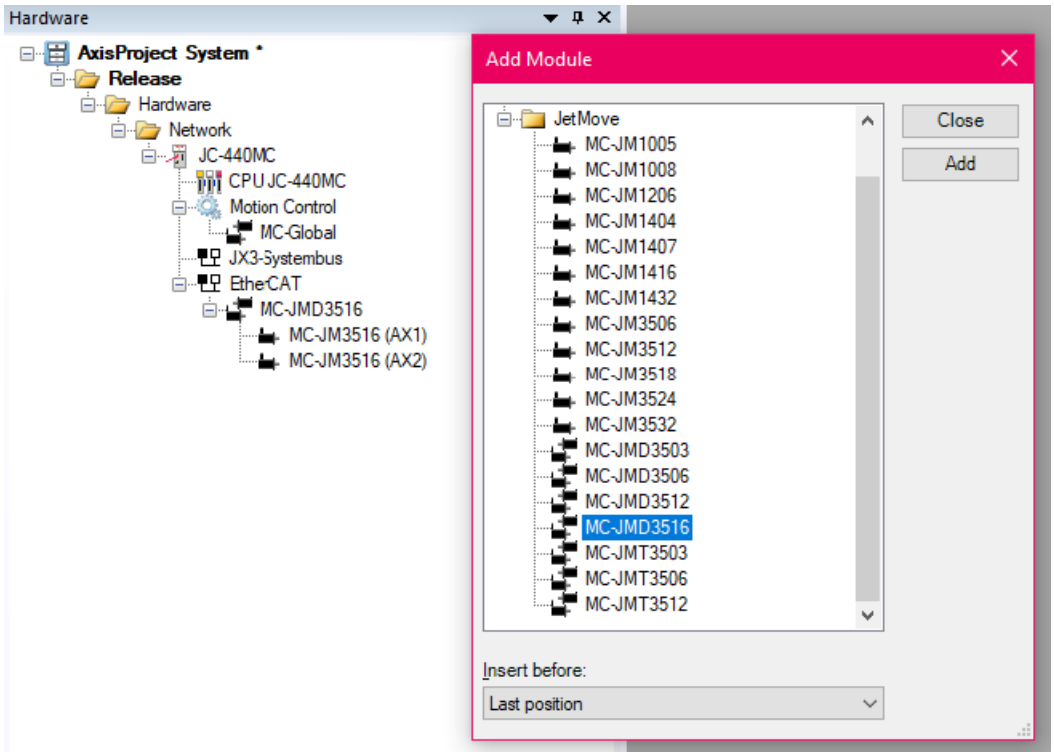
5.2 Reale Achse

5.2.1 Hinzufügen eines Servoverstärkers

Hier lassen sich nun Servoverstärker am EtherCAT anlegen.



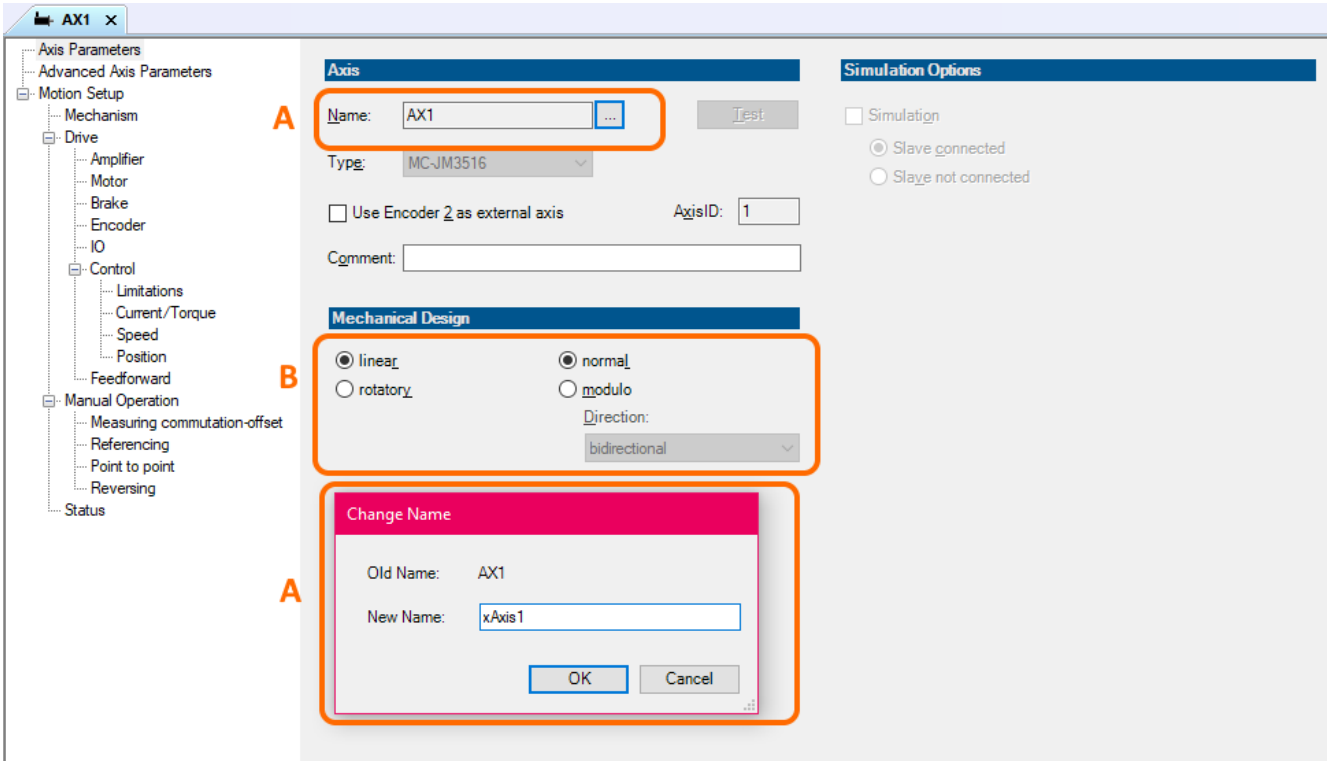
1. Öffnen Sie über das Untermenü am Knoten „EtherCAT“ das Dialogfeld zur Auswahl der Module.
2. Schließen Sie per Modul hinzufügen diese an den Knoten an.



3. Wählen Sie nun die benötigten Servoverstärker aus und fügen Sie diesen per Doppelklick oder Hinzufügen (Add) der Hardwarekonfiguration hinzu.

5.2.2 Konfigurieren einer realen Achse

Per Doppelklick auf ein Achsobjekt öffnet sich das MotionSetup – ein Konfigurations- und Inbetriebnahmedialog.



In diesem Beispiel ist das MotionSetup der Achse „AX1“ geöffnet.

Auf der Startseite des MotionSetups lassen sich die grundsätzlichen Einstellungen des Achsobjekts festlegen.

A

Es empfiehlt sich, statt des vorgeschlagenen Achsnamens „AX1“ einen für die Anwendung sinnvollen Namen zu vergeben. Über diesen Namen wird das jeweilige Achsobjekt auch im STX-Programm angesprochen. In diesem Dialog lässt sich der Achsname anpassen.

B

Im Dialog „Mechanische Ausführung“ wird festgelegt, ob es sich um eine rotative oder lineare Bewegung handelt und ob die Achse eine Modulo- oder normale Achse ist.

Normale Achse

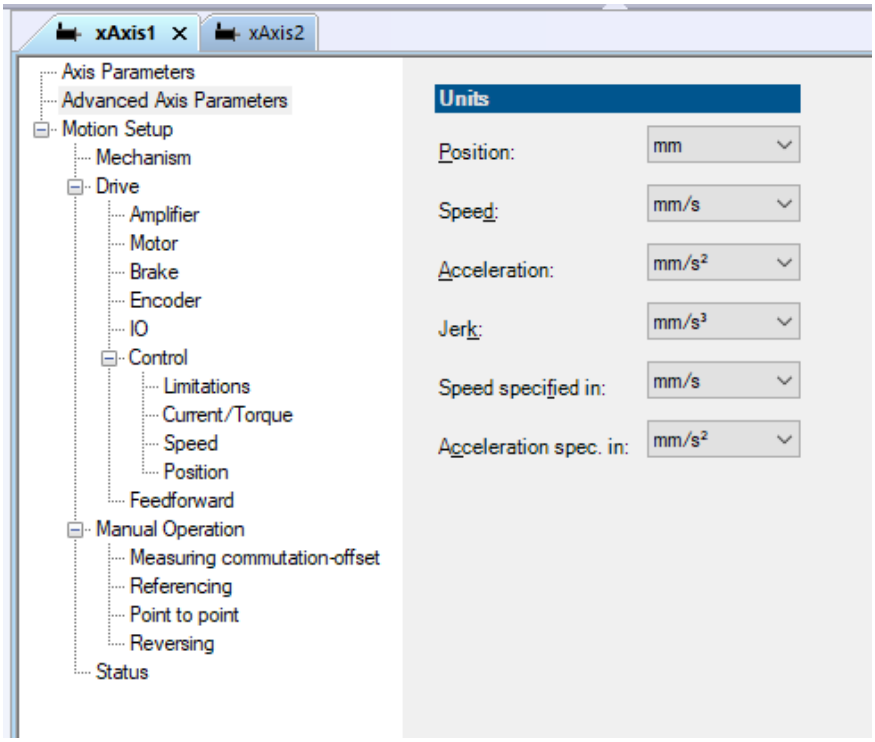
Die Achse hat einen festen Verfahrbereich, der nicht verlassen werden kann.

Modulo-Achse

Die Position der Achse springt bei Überschreiten des Modulobereichs wieder auf Anfang. Es wird hierbei meist die Bewegung am Arbeitspunkt berücksichtigt.

5.2.3 Erweiterte Achsparameter

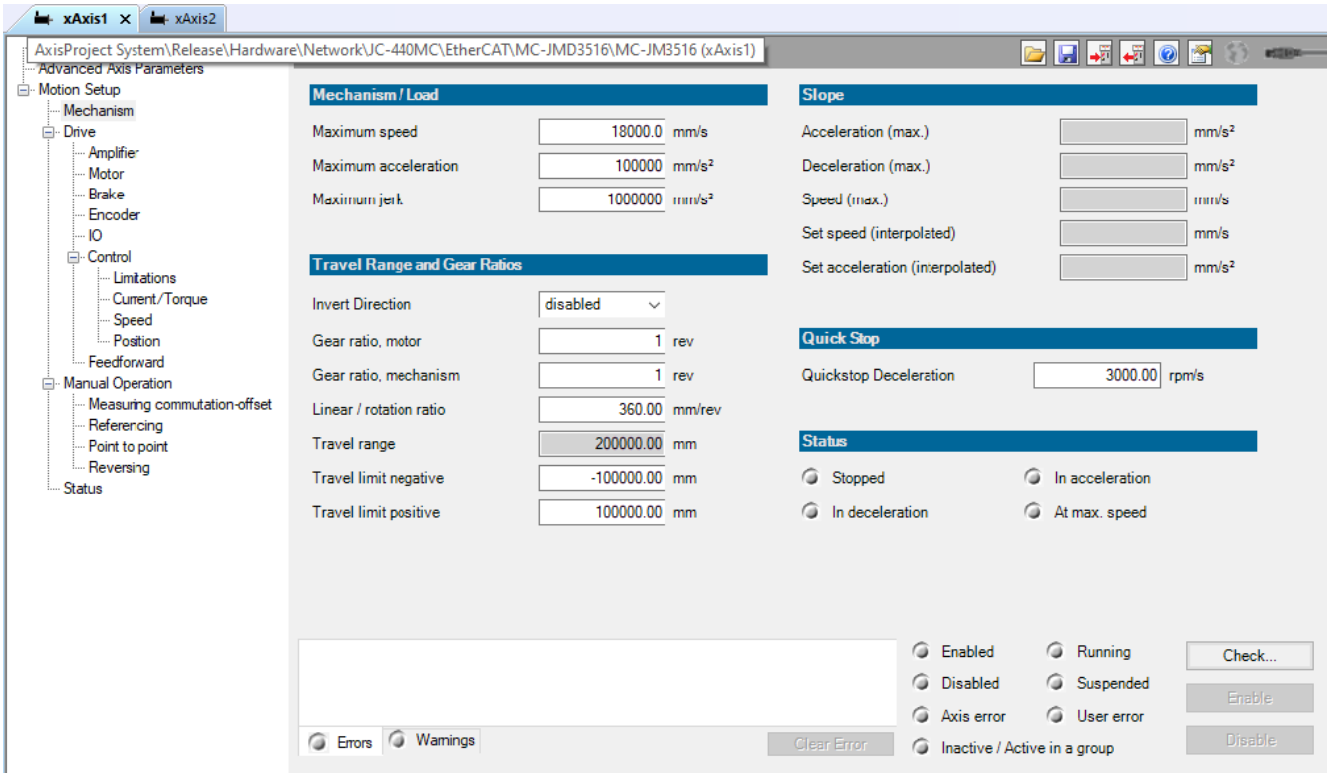
Hier lassen sich die Einheiten der Achse einstellen.



Siehe Mechanische Maßeinheiten [▶ 11].

5.2.4 Mechanische Parameter

Auf der MotionSetup-Seite Mechanik werden für die Sollwertberechnung wichtige Parameter voreingestellt.



Diese Parameter beschreiben die mechanischen Verhältnisse:

– Wie schnell kann/darf der Mechanismus bewegt werden?
– Welche Beschleunigungen sind zulässig?

– Muss die Drehrichtung geändert werden, weil bei der Bewegung des Motors in positiver Richtung die Mechanik in die logisch negative Richtung fährt?
– Wie sind die Übersetzungsverhältnisse?
– Welcher Verfahrbereich ist zulässig?

5.2.5 Achsinbetriebnahme

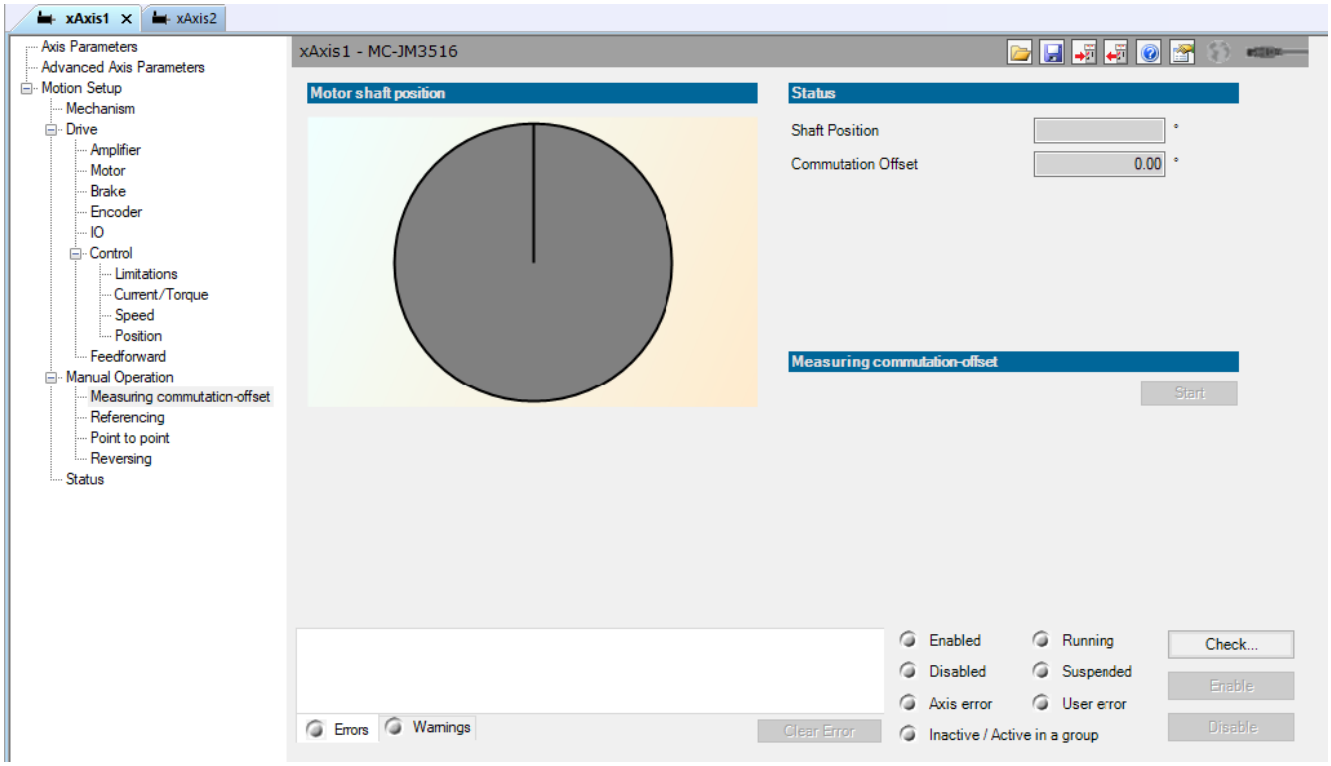
Zur Inbetriebnahme der Achse stehen weitere MotionSetup-Seiten zu Verfügung.

► Entnehmen Sie bitte den Handbüchern der Servoverstärker die Vorgehensweise zur Inbetriebnahme.

5.2.6 Manueller Betrieb

Messung Kommutierungsoffset

Hier kann eine Kommutierungsmessung durchgeführt werden, um den Kommutierungswinkel zu ermitteln.

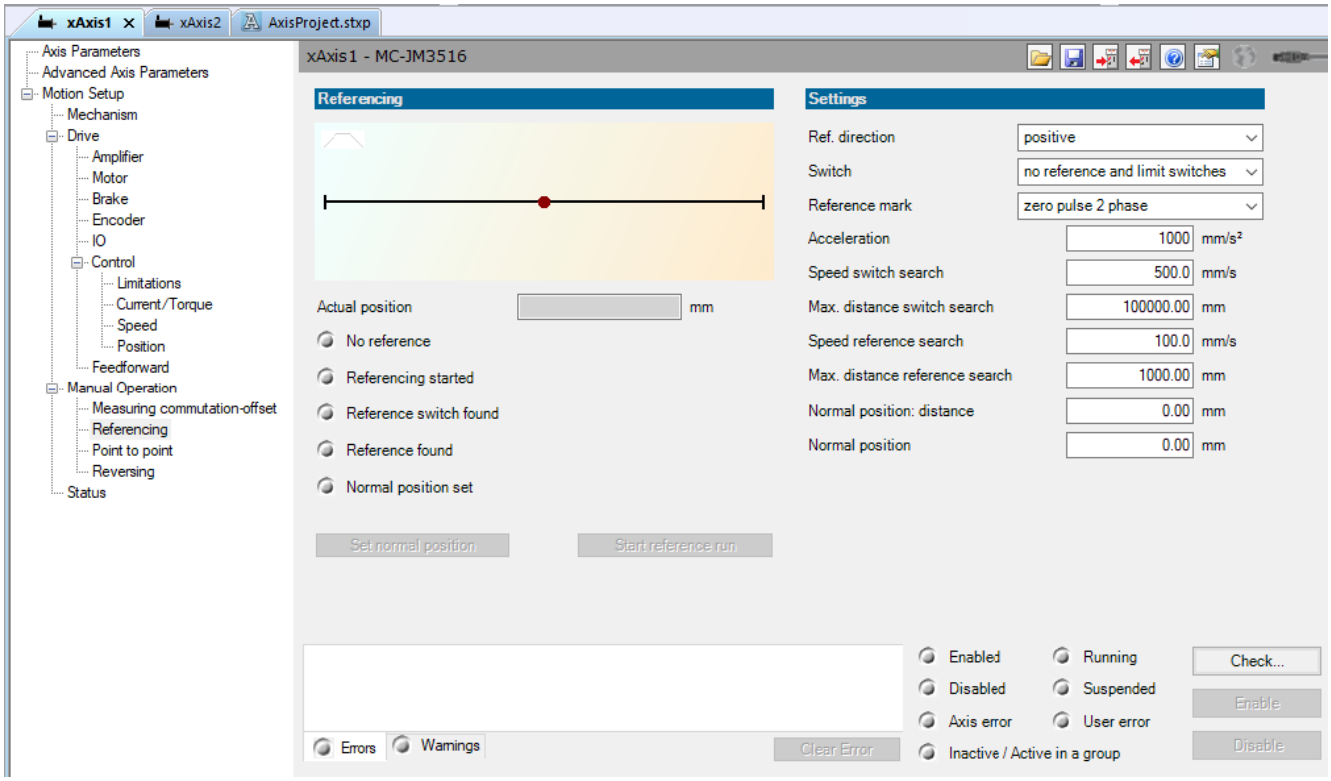


– Standardmäßig haben Servomotoren von Bucher Automation einen Kommutierungsoffset von 0°. Bei Motoren anderer Hersteller kann dieser Winkel aber auch abweichen. Die Kommutierung lässt sich mit dieser Messung herausfinden.
– Die ermittelte Kommutierung wird in der ini-Datei der Achse abgespeichert und beim Neustart der Steuerung geladen.
– Eine Kommutierungsmessung kann auch über die Applikation gestartet werden:

```
xAxis1.Drive.Motor.MeasureCommutationOffset(DriveCommOffsetMeasurementMo-
des.DynamicMode);
when xAxis1.State.IsDisabled continue;
```

Referenzierung

Auf dieser Seite lässt sich die Referenzfahrt konfigurieren und testen.



– Die eingestellten Referenzfahrtparameter werden in der ini-Datei der Achse abgespeichert und beim Neustart der Steuerung geladen.
– Wird in der Applikation die Referenzfahrt ohne Übergabeparameter gestartet, so werden die ein-gestellten Parameter aus der ini-Datei verwendet, sofern die Referenzparameter nicht durch die Applikation überschrieben wurden.
– **Achtung:**
Beim Starten der Referenzfahrt ohne Übergabeparameter werden die zuletzt eingestellten Parameter verwendet. Sind in der Applikation Referenzfahrtparameter verändert worden, so werden diese Werte verwendet.

```
xAxis1.MoveHome.Start();
```

– Die Referenzfahrtparameter lassen sich aber auch in der Applikation gesondert einstellen.

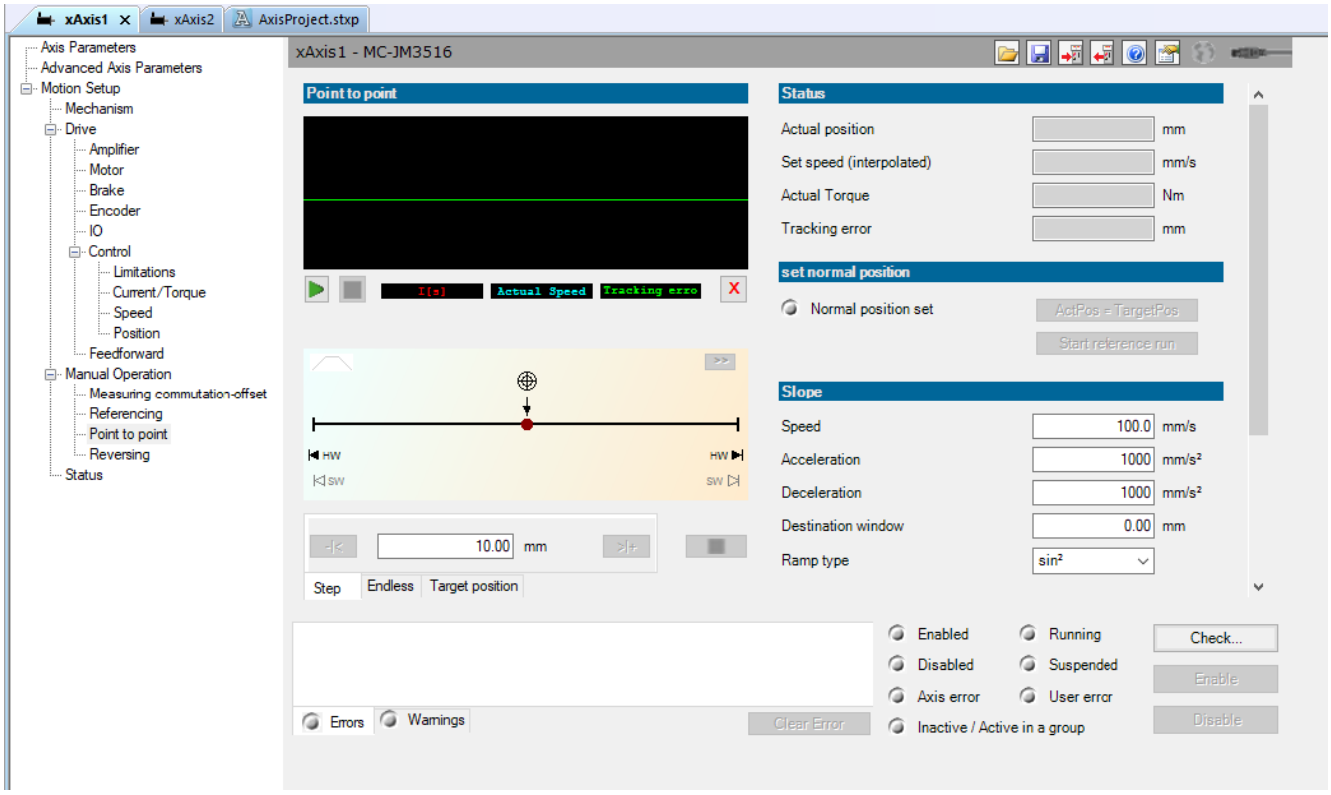
ⓘ Info

Weiterführende Informationen

Eine ausführliche Beschreibung der Referenzfahrtparameter und Möglichkeiten der Referenzfahrt entnehmen sie bitte den Handbüchern der Servoverstärker und der Jet-Sym-Hilfe.

Punkt-zu-Punkt

Auf dieser Seite lässt sich eine Punkt-zu-Punkt-Positionierung konfigurieren und testen.



– Die eingestellten Punkt-zu-Punkt-Parameter werden in der ini-Datei der Achse abgespeichert und beim Neustart der Steuerung geladen.

– Wird in der Applikation die Punkt-zu-Punkt-Positionierung ohne Übergabeparameter gestartet, so werden die eingestellten Parameter aus der ini-Datei verwendet, sofern die Punkt-zu-Punkt-Parameter nicht durch die Applikation überschrieben wurden.

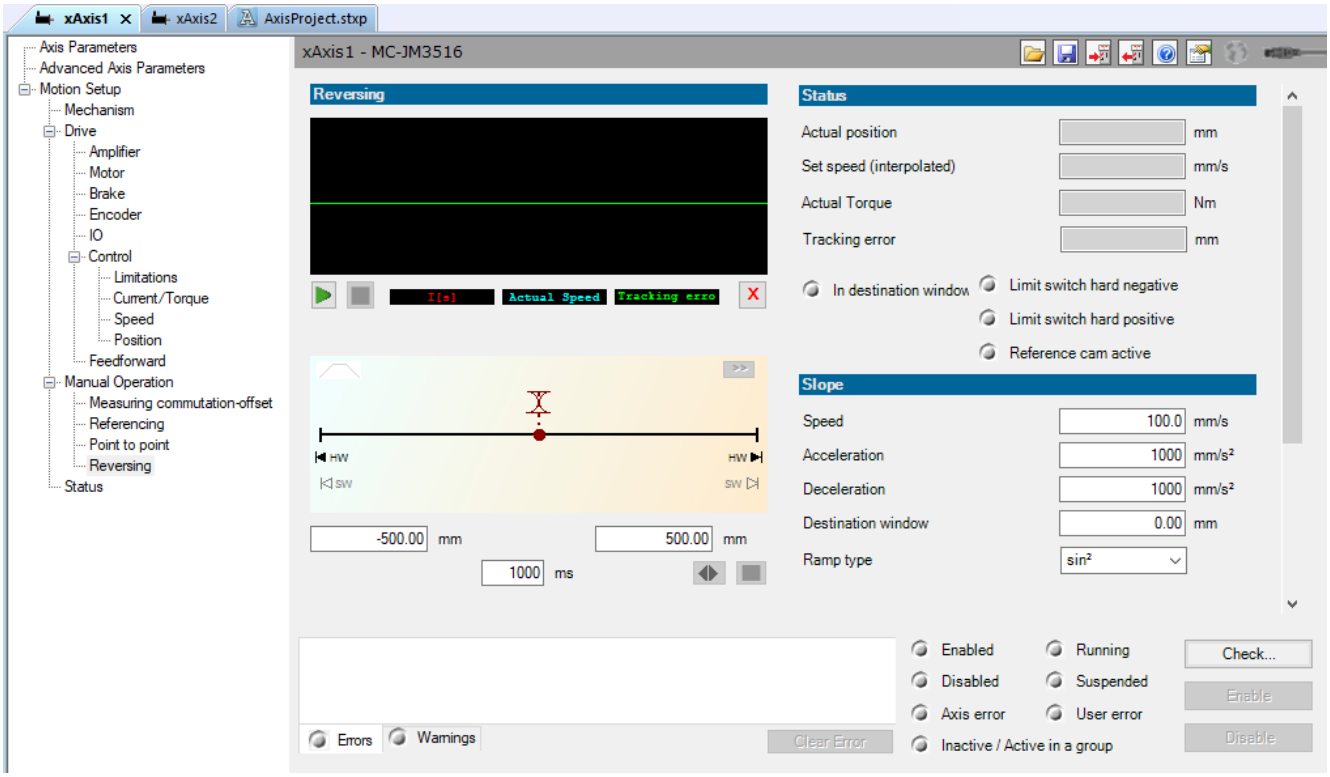
– **Achtung:**
Bei Starten der Punkt-zu-Punkt-Positionierung ohne Übergabeparameter werden die zuletzt eingestellten Parameter verwendet. Sind in der Applikation Punkt-zu-Punkt-Parameter verändert worden, so werden diese Werte verwendet.

```
xAxis1.MovePtp.Start();
```

– Die Punkt-zu-Punkt-Parameter lassen sich aber auch in der Applikation gesondert einstellen.

Reversieren

Auf dieser Seite lässt sich ein Reversierbetrieb einstellen. Dieser wird oft zum Abgleich der Regelparameter verwendet.



– Die eingestellten Parameter wie Geschwindigkeit, Beschleunigung, Verzögerung, Zielfenster und Rampentyp sind identisch mit den Punkt-zu-Punkt-Parametern. Diese werden in der ini-Datei der Achse abgespeichert und beim Neustart der Steuerung geladen.

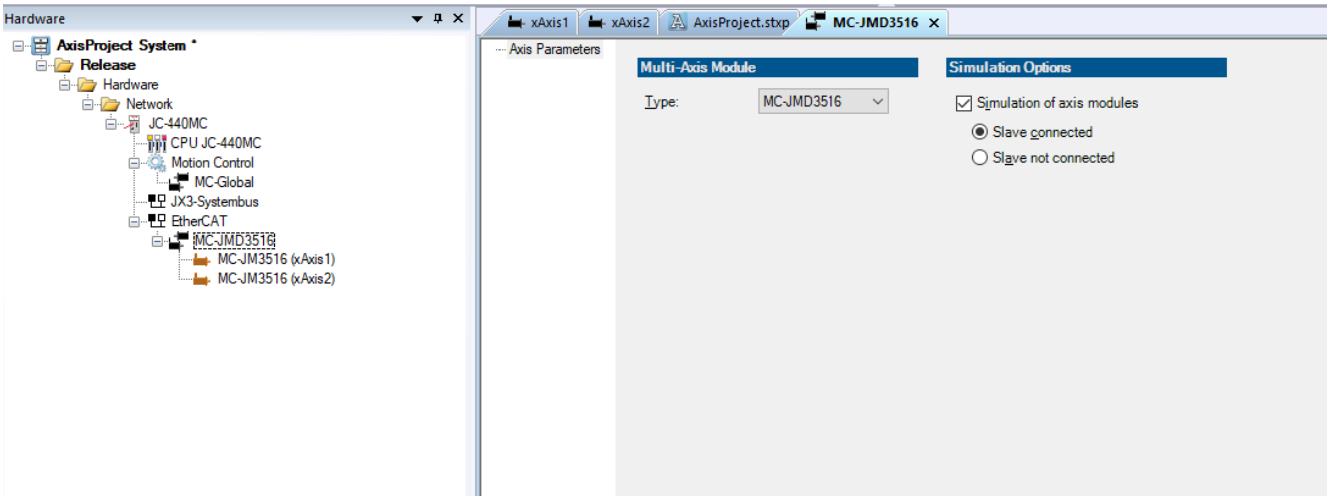
– Ein gesonderter Reversierbetrieb ist in der Applikation nicht vorhanden, kann aber entsprechend programmiert werden und sieht vereinfacht z.B. so aus:

```
loop
xAxis1.MovePtp.Start(,Target1);
when xAxis1.State.IsEnabled continue;
xAxis1.MovePtp.Start(,Target3);
when xAxis1.State.IsEnabled continue;
end_loop;
```

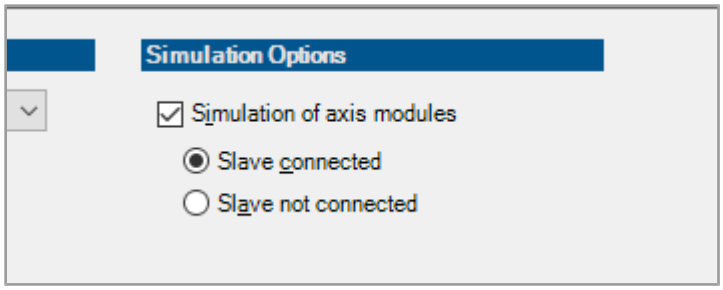
5.3 Simulationsachse

Die Verwendung von Simulationsachsen ermöglicht eine Entwicklung des Applikationsprogramms ohne oder mit nur teilweise angeschlossener Hardware.

Steht die Maschine nicht oder nur zum Teil zur Verfügung, können die fehlenden Antriebe simuliert werden. Der Test von in Entwicklung befindlichen oder zu überprüfenden Abläufen ist gefahrlos möglich. Hierzu wird bei den Achsparametern des jeweiligen Servoverstärkers die Option Simulation ausgewählt.

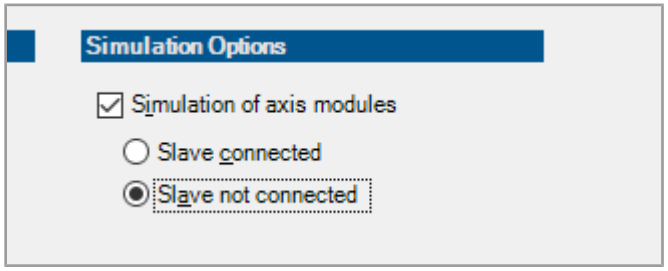


– **Achtung:**
  Es können immer nur ganze Module simuliert werden. Werden Mehrfach-Achsverstärker verwendet, wird die Simulation-Option im überlagerten Knoten gesetzt und gilt für alle Achsen dieses Moduls.
– Ist der Servoverstärker angeschlossen, so wird die Option Slave angeschlossen ausgewählt.



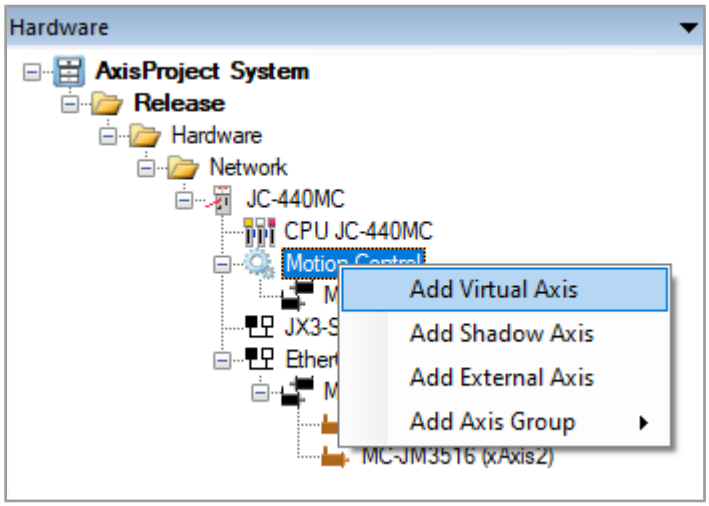
Diese Option ist dann sinnvoll, wenn die Hardware verbunden ist, aber z.B. keine Motoren angeschlossen sind oder die Mechanik während des Tests nicht bewegt werden soll.

Ist keine Hardware angeschlossen, so muss die Option Kein Slave angeschlossen ausgewählt werden.

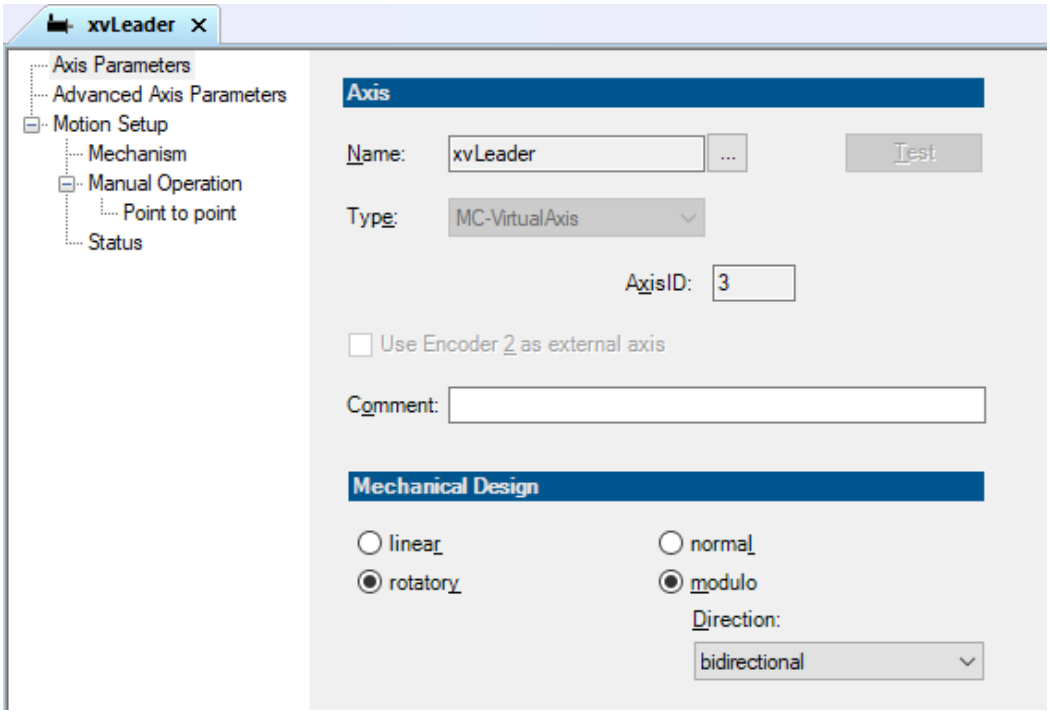


– Die Unterscheidung ist wichtig, da für die Initialisierung des Systembusses klar sein muss, welche Module angeschlossen sind.
– **Anwendung:**
  Simulationsachsen werden in der Applikation wie reale Achsen verwendet. Es ist jedoch zu berücksichtigen, dass nicht direkt auf die Achse zugegriffen werden kann. Z.B. würde ein Auslesen der Zwischenkreisspannung nicht ein erwartetes Ergebnis zurückliefern.
– Auch können externe Reaktionen ausbleiben. Wird an der realen Maschine z.B. während der Fahrt eine Lichtschranke betätigt, so wird dies in der Simulation nicht stattfinden.

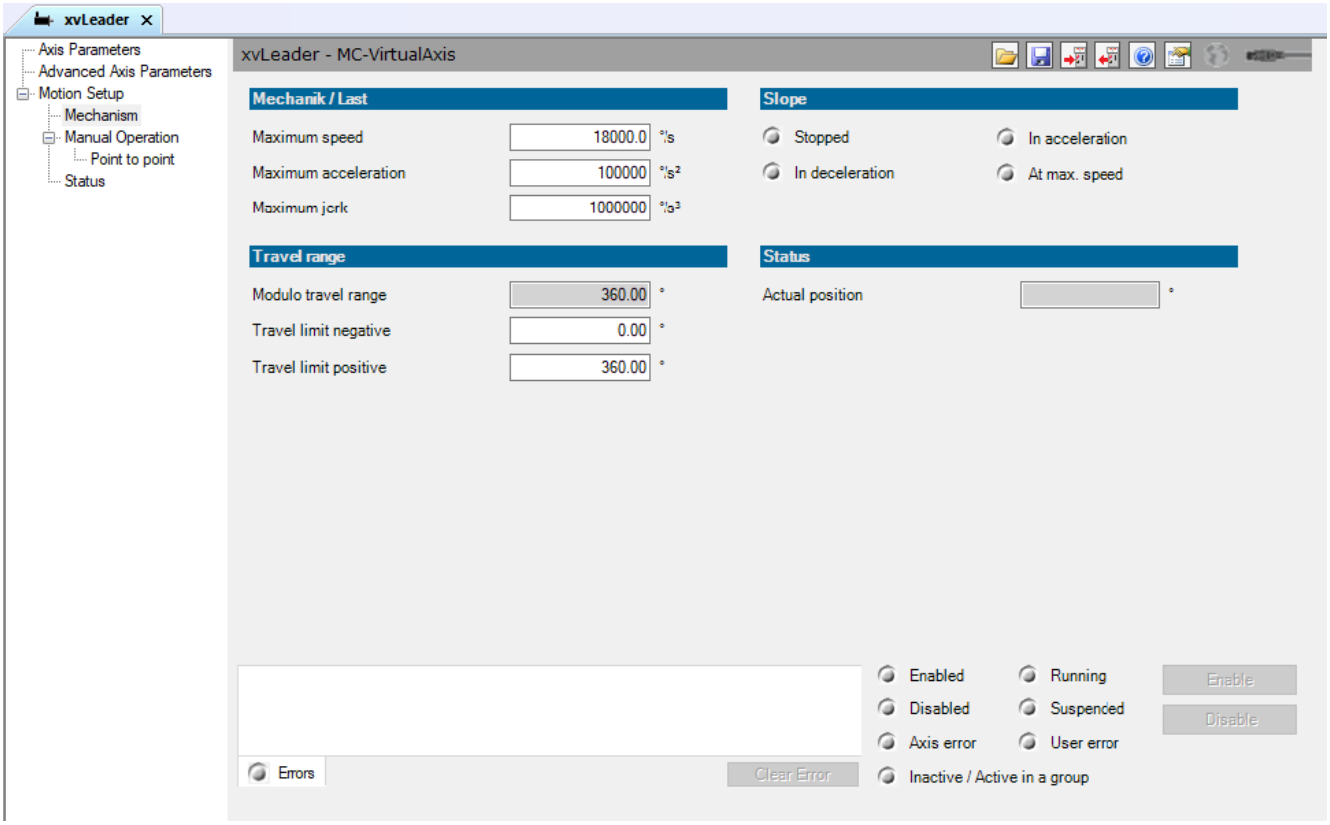
5.4 Virtuelle Achsen



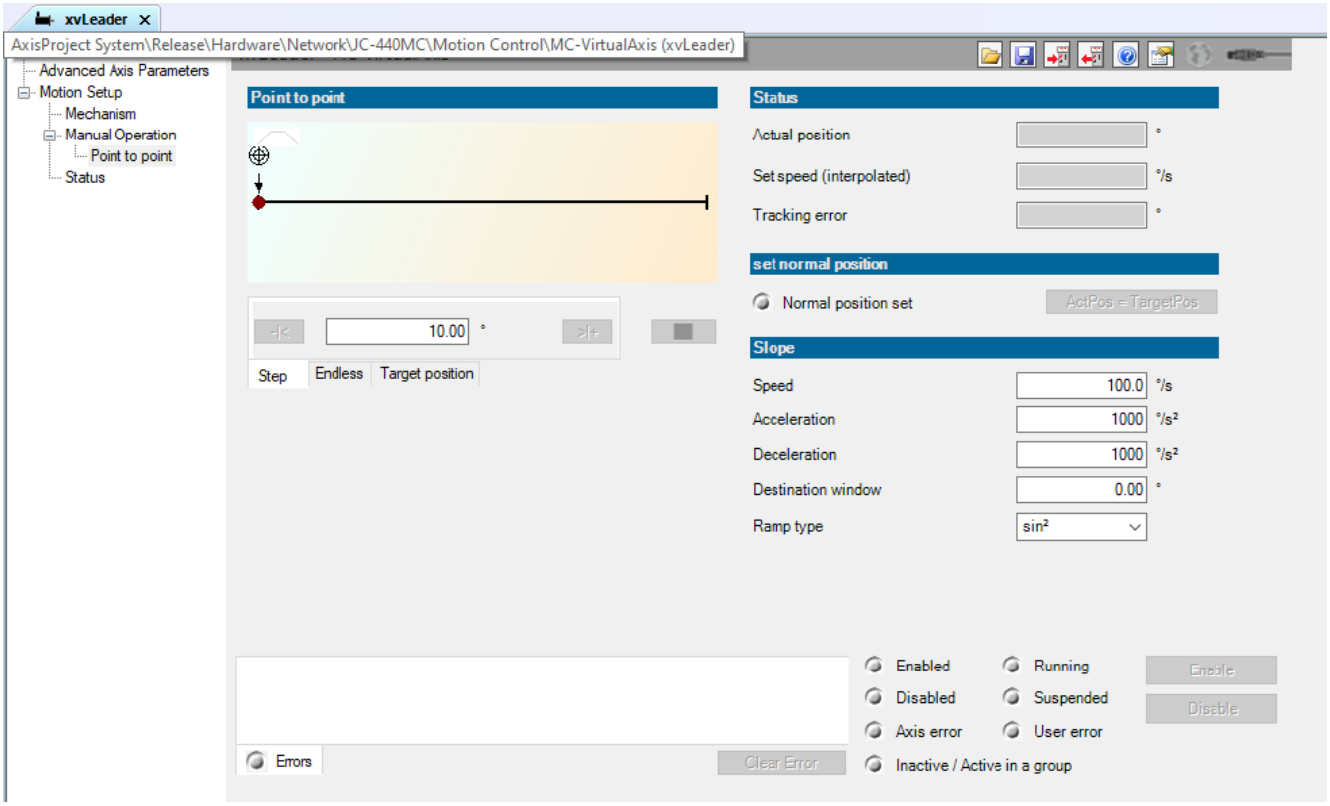
1. Öffnen Sie über das Untermenü am Knoten „Motion-Control“ das Dialogfeld zur Auswahl der Achsen.
2. Fügen Sie per Virtuelle Achse hinzufügen diese der Hardwarekonfiguration hinzu.
   Die Konfiguration einer virtuellen Achse erfolgt wie bei einer realen Achse, nur dass diese Achse keine Hardware besitzt und somit keine Motorparameter eingestellt werden müssen.



3. Wählen Sie bei der Konfiguration das benötigte mechanische Design aus und stellen Sie Verfahrbereich und Maximalwerte ein.



Zu Testzwecken lässt sich über das MotionSetup auch eine virtuelle Achse positionieren.



– Die eingestellten Punkt-zu-Punkt-Parameter werden in der ini-Datei der Achse abgespeichert und beim Neustart der Steuerung geladen.

– Wird in der Applikation die Punkt-zu-Punkt-Positionierung ohne Übergabeparameter gestartet, so werden die eingestellten Parameter aus der ini-Datei verwendet, sofern die Punkt-zu-Punkt-Parameter nicht durch die Applikation überschrieben wurden.

– **Achtung:**
Beim Starten der Punkt-zu-Punkt-Positionierung ohne Übergabeparameter werden die zuletzt ein-gestellten Parameter verwendet. Sind in der Applikation Punkt-zu-Punkt-Parameter verändert worden, so werden diese Werte verwendet.

```
xvLeader.MovePtp.Start();
```

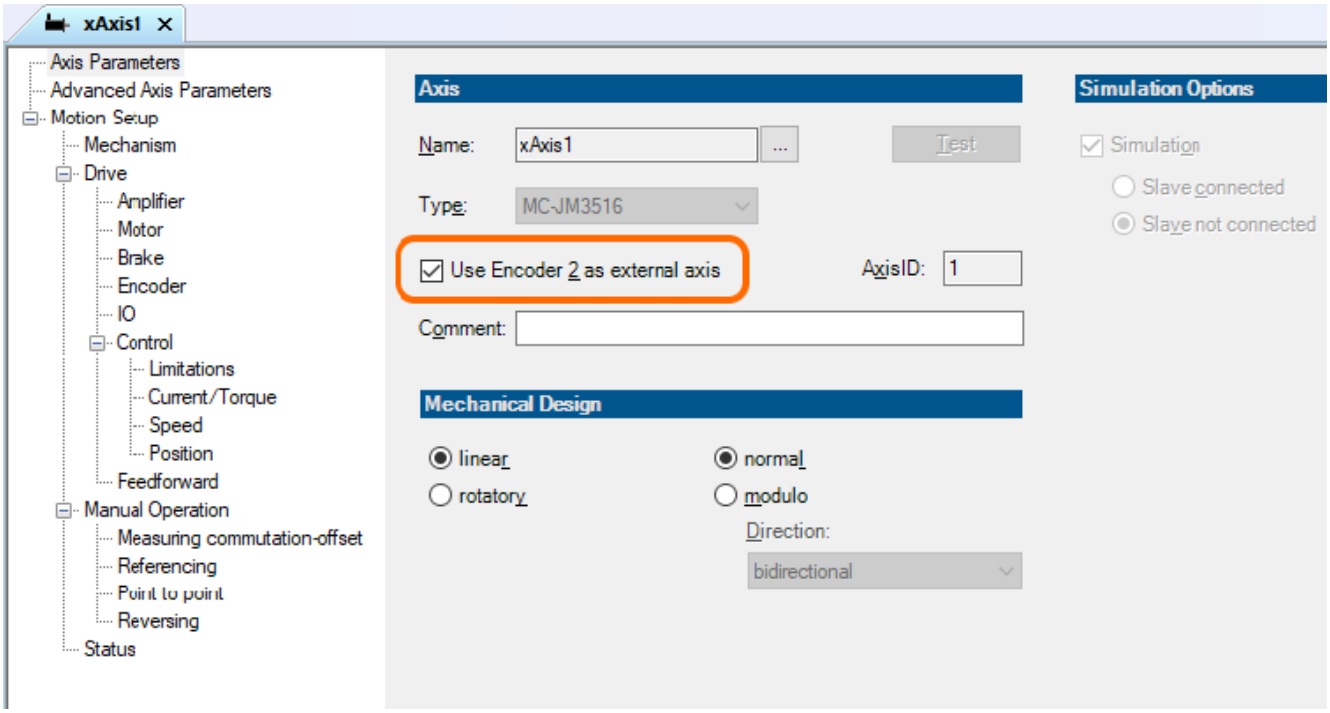
– Die Punkt-zu-Punkt-Parameter lassen sich aber auch in der Applikation gesondert einstellen.

5.5 Externe Achse

1. Öffnen Sie über das Untermenü am Knoten „Motion-Control“ das Dialogfeld zur Auswahl der Achsen.
2. Fügen Sie per Externe Achse hinzufügen diese der Hardwarekonfiguration hinzu.

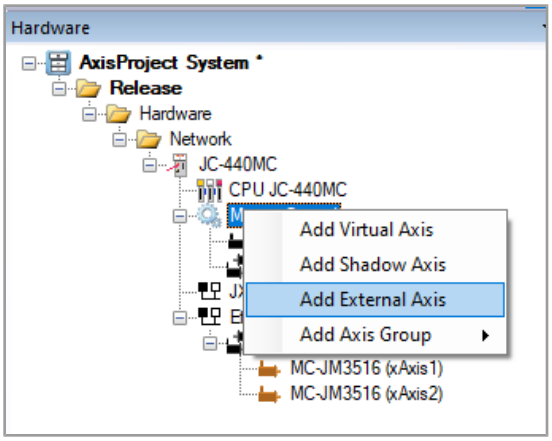
5.5.1 JC-440Ext mit Ethercat®-Drives

Es wird ein Ethercat®-Drive mit einem zusätzlichen Geber-Eingang benötigt.

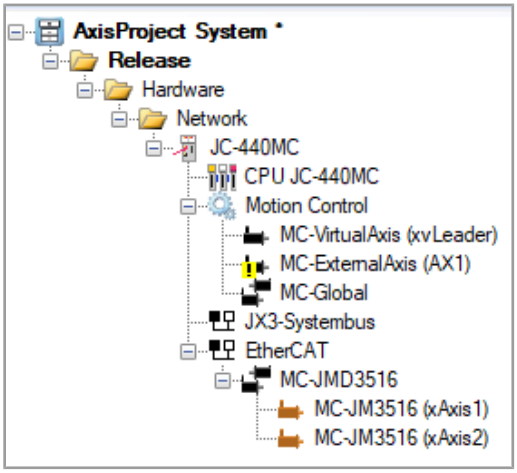


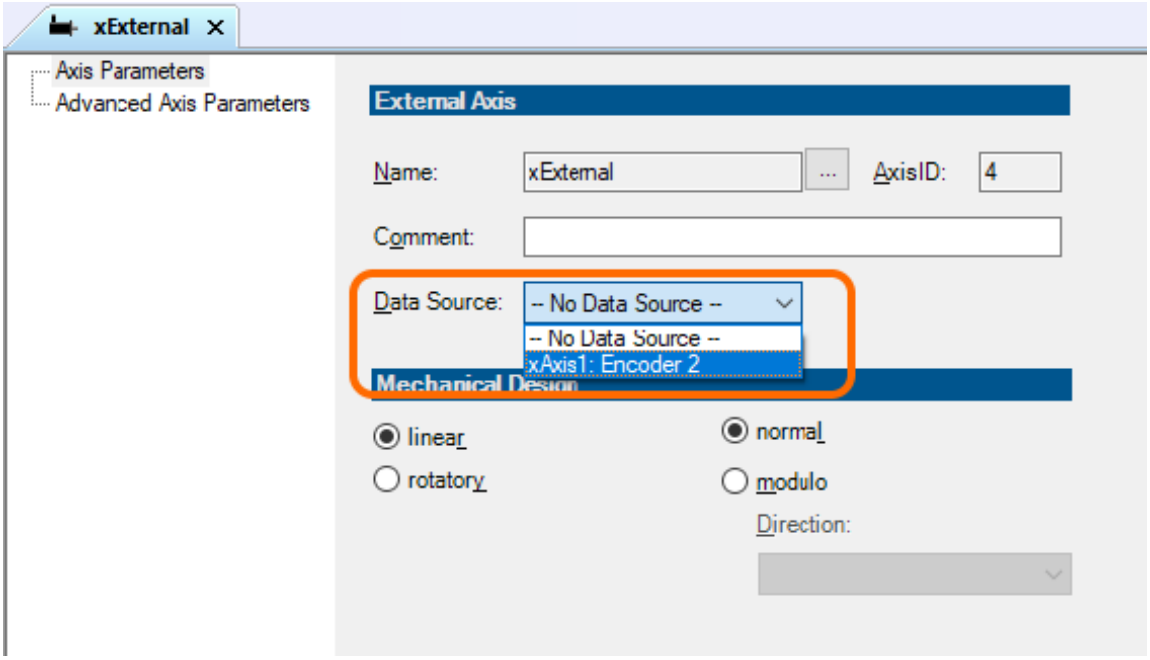
1. Wählen Sie aus, dass der zweite Gebereingang für eine Externe Achse verwendet werden soll.

   ⇨ Damit steht dieser Eingang nicht als zweiter Geber für die Regelung der realen Achse zu Verfügung.



2. Wählen Sie im Kontextmenü des Knoten „MotionControl“ Externe Achse hinzufügen aus.





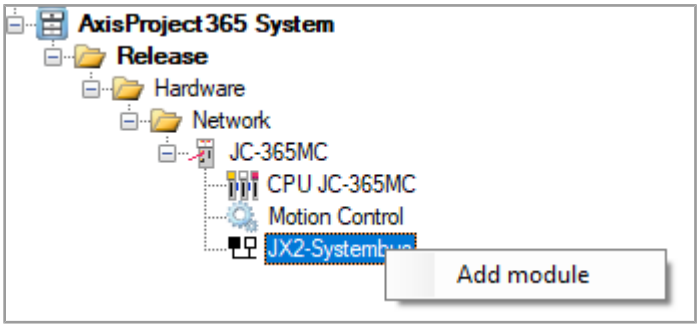
3. Wählen Sie hier den für die externe Achse freigegebenen Geber aus und weisen sie ihn diesem Achsobjekt zu.
4. Stellen Sie hier auch die „Mechanische Ausführung“ ein.

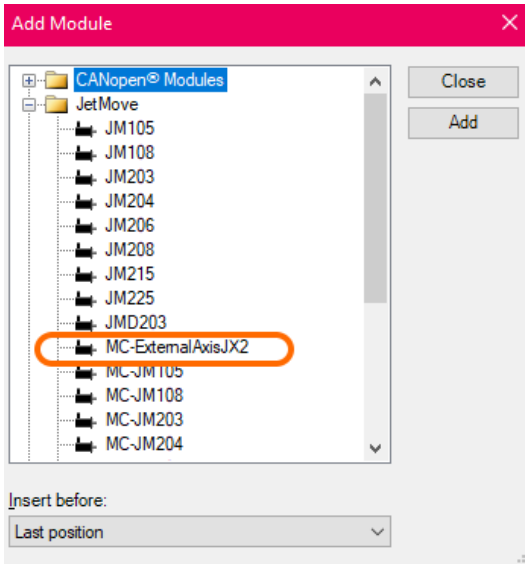
⇨ Damit legen Sie fest, ob Sie eine lineare oder rotatorische Achse haben und ob diese normal bzw. modulo ist.

Siehe dazu Kapitel Mechanische Ausführung [▶ 9].

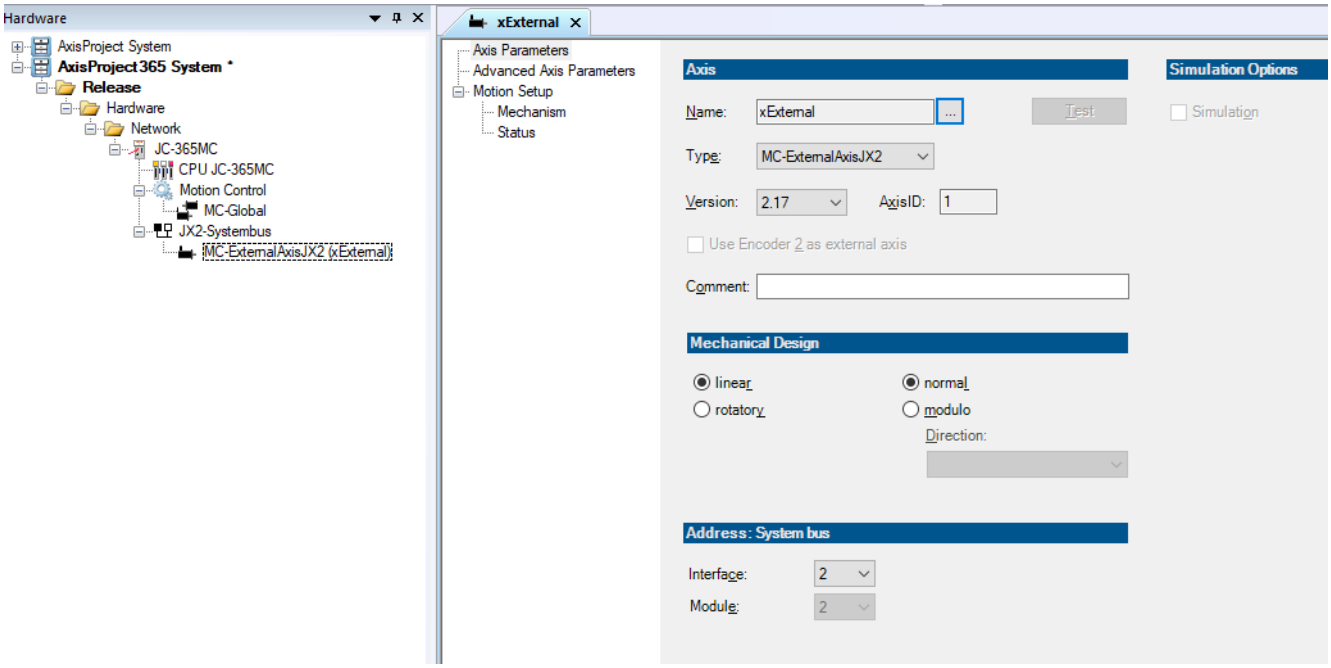
5.5.2 JC-365MC

– Bei JM-200 und JM-105/108 kann der zweite Gebereingang, soweit vorhanden, nicht als externe Achse verwendet werden.
– Für die Erfassung eines zusätzlichen Gebersignals für eine externe Achse ist ein JM-200 oder JM-105/JM108 am JX2-Systembus nötig.





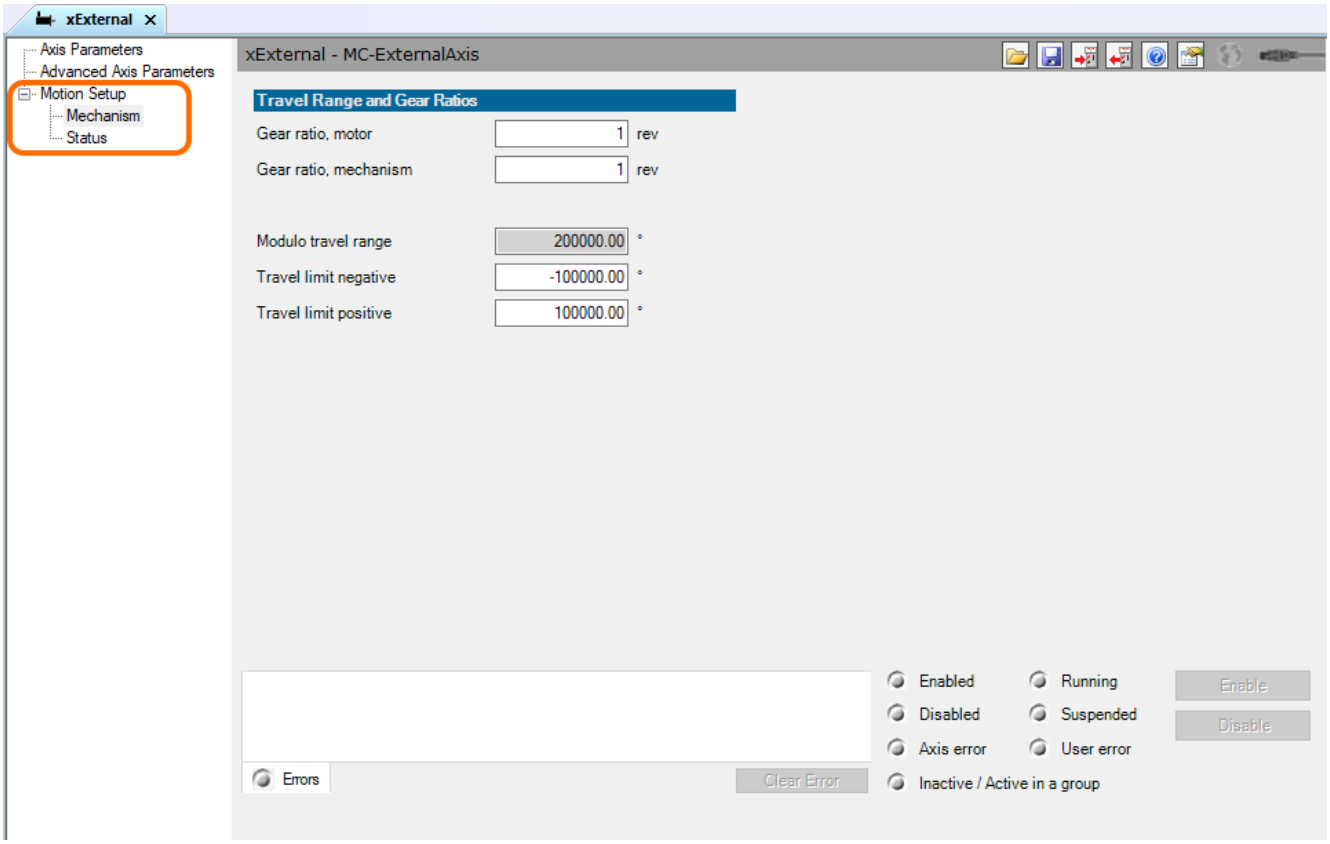
1. Hierfür fügen Sie das Modul „MC-ExternalAxisJX2“ dem Systembus hinzu.



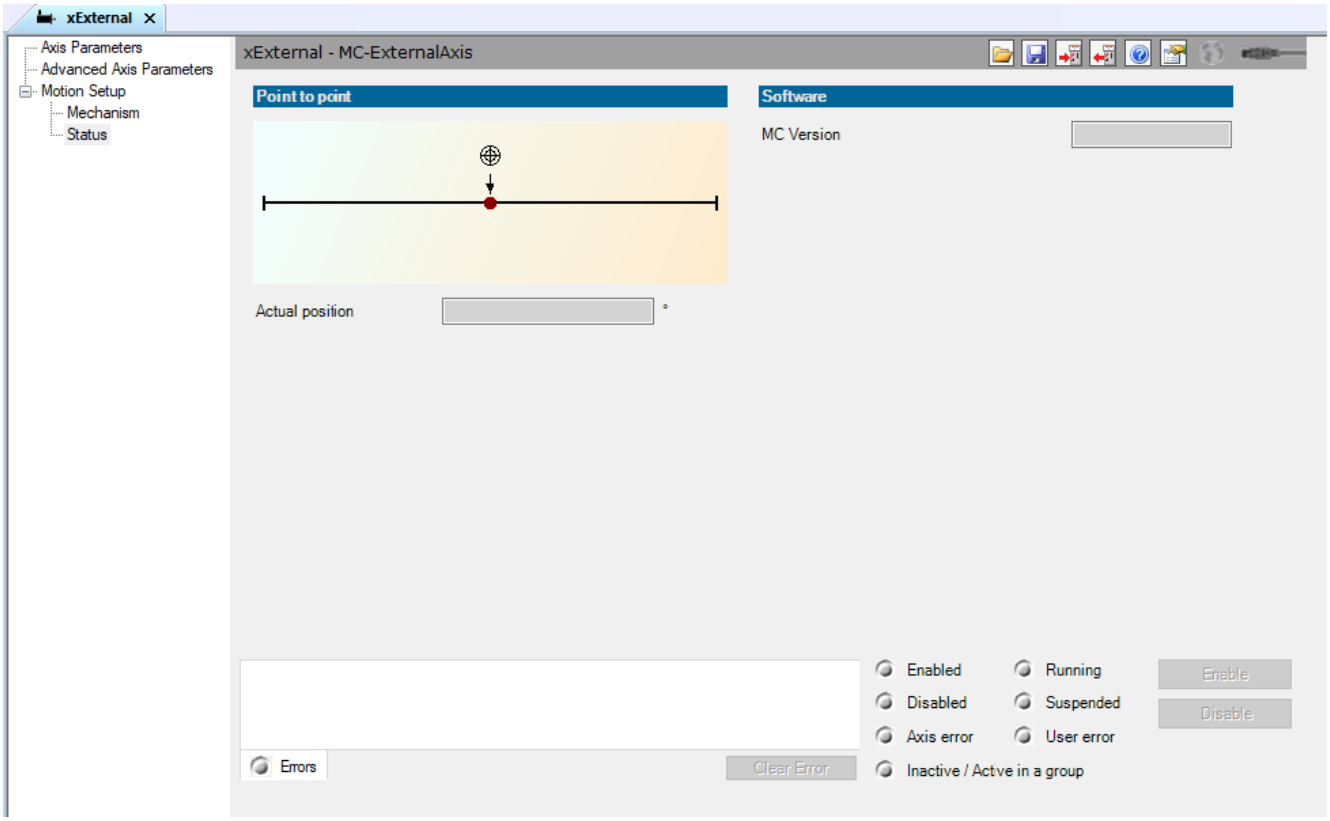
2. Sie können nun die gleichen Parameter einstellen wie beim Aufbau mit JC-440Ext und JM-3000.

5.5.3 Parametrierung

✓ Sind die Basis-Einstellungen gültig, erscheint im Menü die Auswahl des MotionSetups.



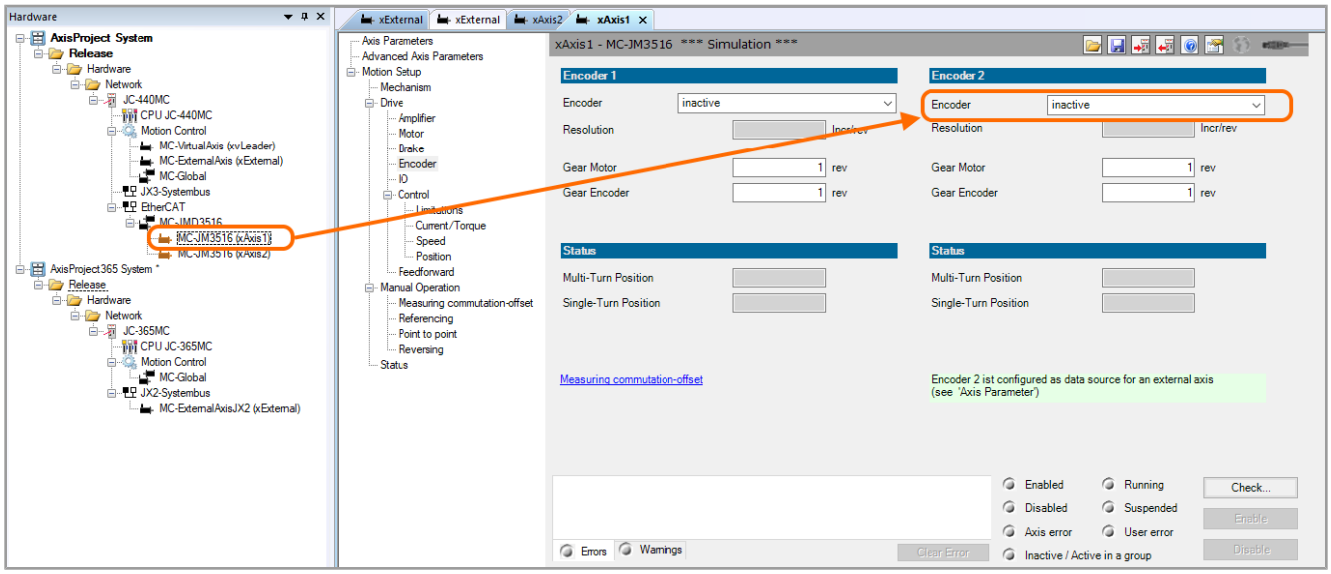
1. Stellen Sie auf der Mechanismus-Seite das Übersetzungsverhältniss und den Verfahrbereich ein.



2. Überprüfen Sie auf der Status-Seite über die Anzeige der aktuellen Position die Funktion des externen Gebers.

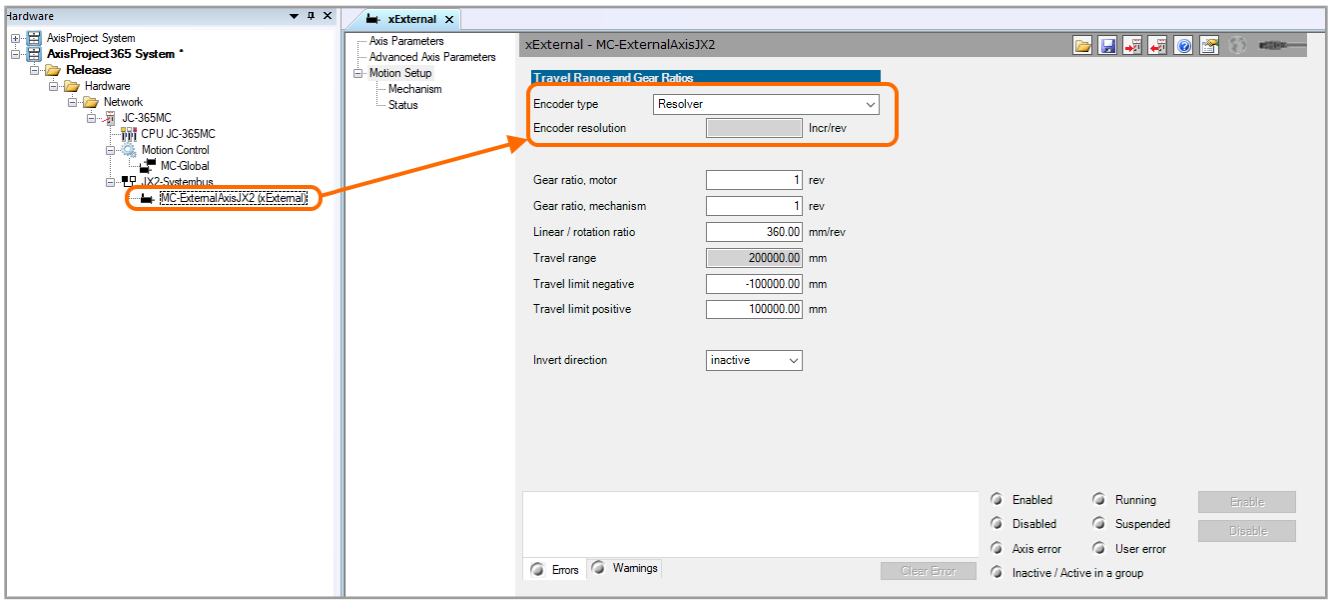
5.5.4 Einstellen des Gebertyps

JM-10xx, JM-2xxx, JM-3xxx



► Stellen Sie den Gebertyp über die Encoder-Seite der realen Achse ein, dessen zweiter Geber für die externe Achse verwendet wird.

JM-2xx/JM-1xx

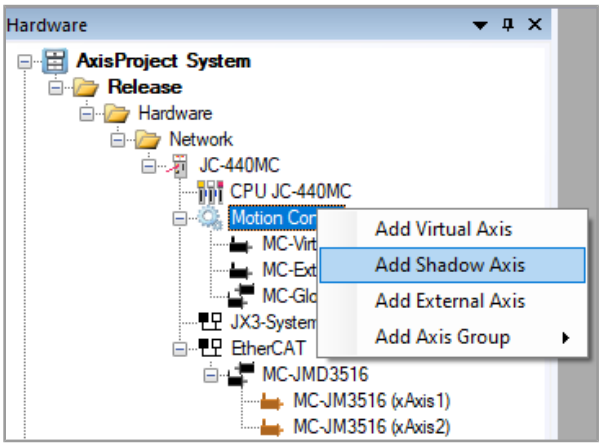


► Stellen Sie den Gebertyp auf der Mechanik-Seite der externen Achse ein.

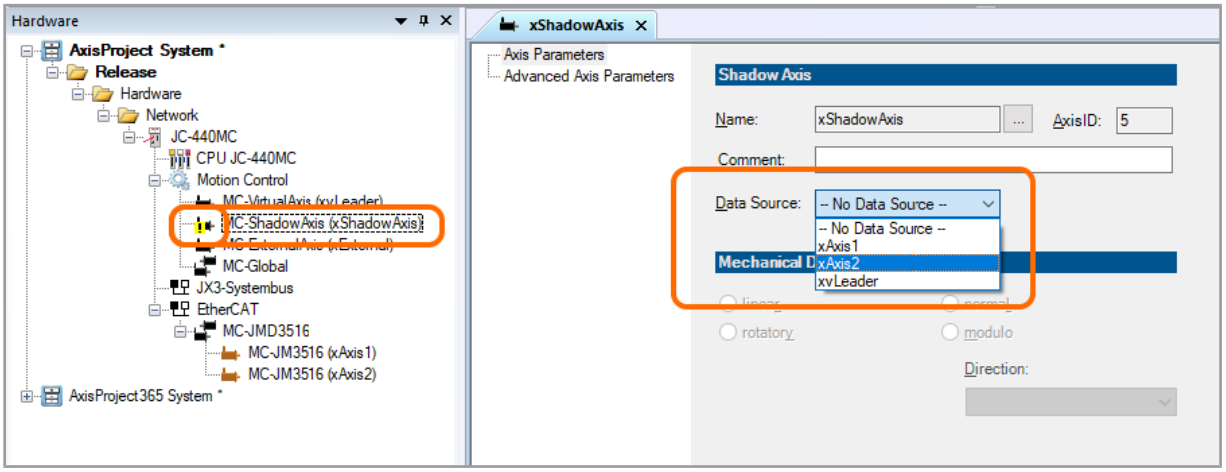
– Die externe Achse benötigt keine weiteren Einstellungen wie Regelparameter, Motortyp usw..

– Des Weiteren ist eine externe Achse immer im Betriebszustand „inaktiv“. Somit erübrigen sich weitere Ansichten im MotionSetup.

5.6 Mitlaufachse



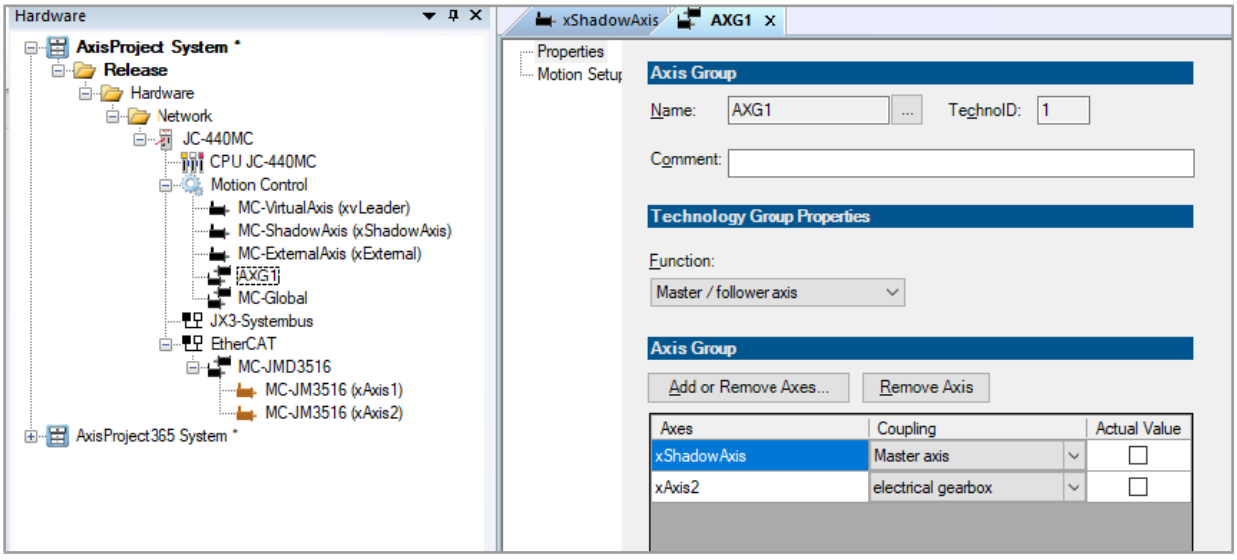
1. Öffnen Sie über das Untermenü am Knoten „Motion-Control“ das Dialogfeld zur Auswahl der Achsen.
2. Fügen Sie per Mitlaufachse hinzufügen diese der Hardwarekonfiguration hinzu.



   ⇨ Es wird eine Mitlaufachse angelegt. Diese besitzt standardmäßig ein Motorensymbol mit Ausrufezeichen, was darauf hindeutet, dass die Konfiguration noch unvollständig oder fehlerhaft ist.

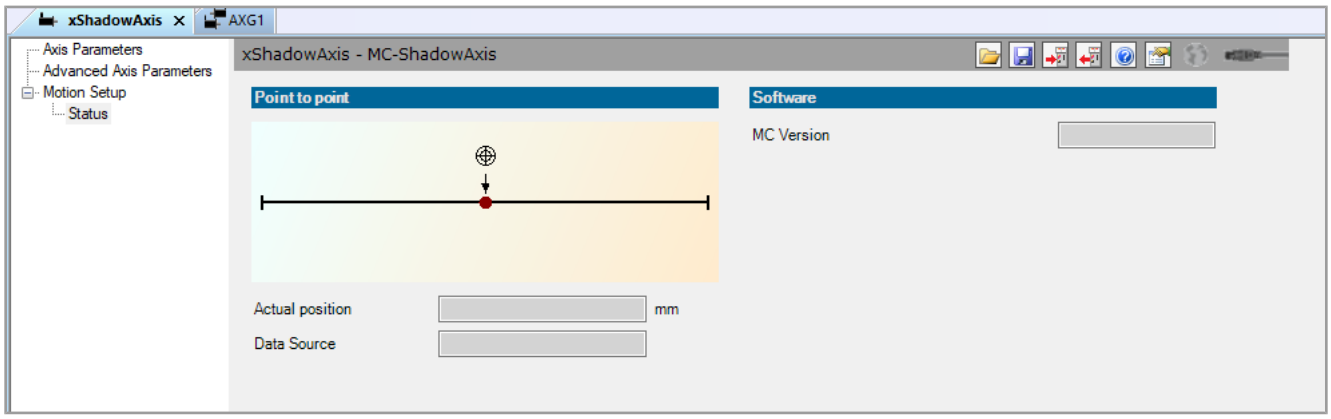
3. Legen Sie die Datenquelle der Mitlaufachse auf der Achsparameter-Seite fest.

Da eine Mitlaufachse nur als Leitachse in einem Technologieverbund eingesetzt werden kann, ist die Konfiguration erst dann vollständig, sobald diese Mitlaufachse als Leitachse in einem Technologieobjekt eingebunden ist.



5.6.1 MotionSetup-Status

Ist die Konfiguration der Mitlaufachse vollständig, lässt sich im MotionSetup die Ist-Position betrachten.



– Die Mitlaufachse erbt alle Einstellungen der Quellachse wie z.B. Verfahrbereich.
– Des Weiteren ist eine Mitlaufachse immer im Betriebszustand „inaktiv“.
– Somit erübrigen sich weitere Ansichten im MotionSetup.

6 Programmieren von Achsen

6.1 Parametereinstellungen der Rampen

- Real
- Simulation
- Virtuell
- Extern
- Mitlauf

6.1.1 Rampentyp

Hier lässt sich einstellen, welcher Rampentyp für eine Positionierung bei der Beschleunigung und Verzögerung verwendet werden soll.

```
xAxis1.Mechanism.Slope.SlopeType := AxisSlopeTypes.Sine;
xAxis1.Mechanism.Slope.SlopeType := AxisSlopeTypes.Linear
```

- Bei Rampentyp **AxisSlopeTypes.Sine** wird mit einer Sinus-Rampe beschleunigt und mit einer Sinus²-Rampe verzögert.
- Bei Rampentyp **AxisSlopeTypes.Linear** wird mit einer linearen Rampe beschleunigt und mit einer Sinus-Rampe verzögert.

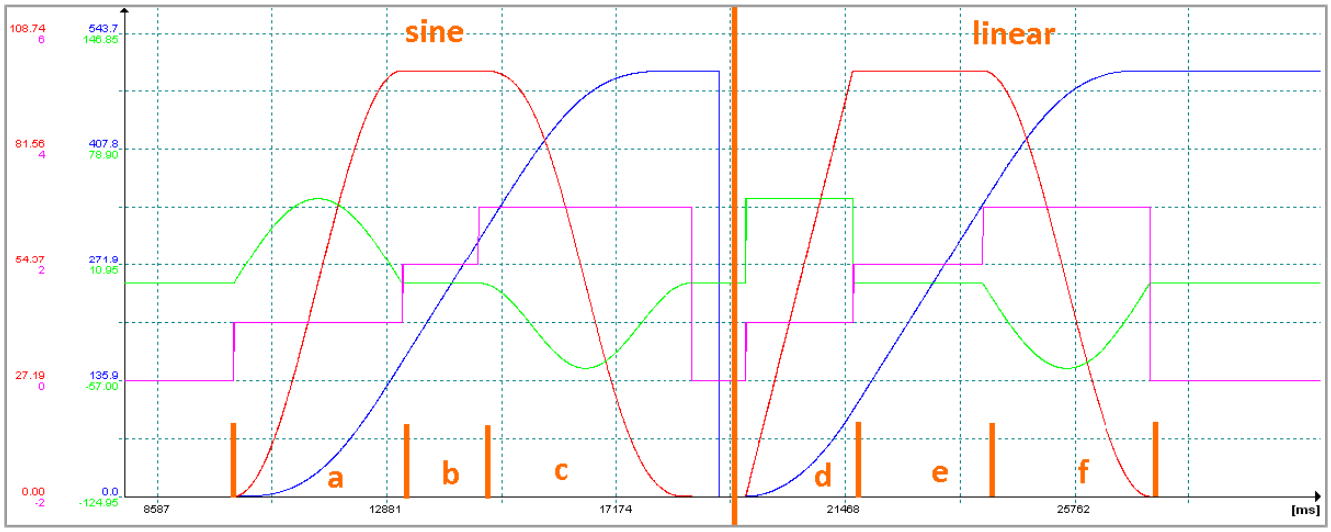


Abb. 2: Rampentyp

Legende

- Sollposition
- Sollgeschwindigkeit
- Sollbeschleunigung
- Rampenstatus

Positionierung *sine*		**Positionierung *linear***	
a	Sinus-Beschleunigung → sinus-förmiger Geschwindigkeitsanstieg	**d**	Lineare Beschleunigung → lineare Zunahme der Geschwindig-keit, konstante Beschleunigung
b	Konstantfahrphase	**e**	Konstantfahrphase
c	Sinus²-Verzögerung → ruckfreies Anhalten	**f**	Sinus-Verzögerung

Tab. 2: Positionierung Rampentyp

6.1.2 Rampenunterbrechung

Mithilfe der Eigenschaft **Config.RampBreakMode** verändern Sie das Verhalten der Sollwertgenerierung für den Fall, dass ein neuer MovePtp.Start-Befehl gegeben wird, während sich die Bewegung in der Verzögerungsrampe befindet.

Konfiguration der Rampe	Verfahrweg reicht noch zum Anhalten aus	Verhalten
None	Ja	Die erste Zielposition wird angefahren. Die Positionierung auf die zweite Zielposition wird direkt angeschlossen.
None	Nein	Die erste Zielposition wird angefahren. Die Positionierung auf die zweite Zielposition wird direkt angeschlossen.
Immediate	Ja	Die Verzögerungsrampe wird mit der neu übergebenen maximalen Verzögerung abgebrochen und die zweite Zielposition wird direkt angefahren.
Immediate	Nein	Die erste Zielposition wird angefahren. Die Positionierung auf die zweite Zielposition wird direkt angeschlossen.

Tab. 3: Konfiguration Rampenunterbrechung

```
xAxis1.Mechanism.Slope.Config.RampBreakMode := MCAxisRampBreakModes.None;
xAxis1.Mechanism.Slope.Config.RampBreakMode := MCAxisRampBreakModes.Immediate;
```

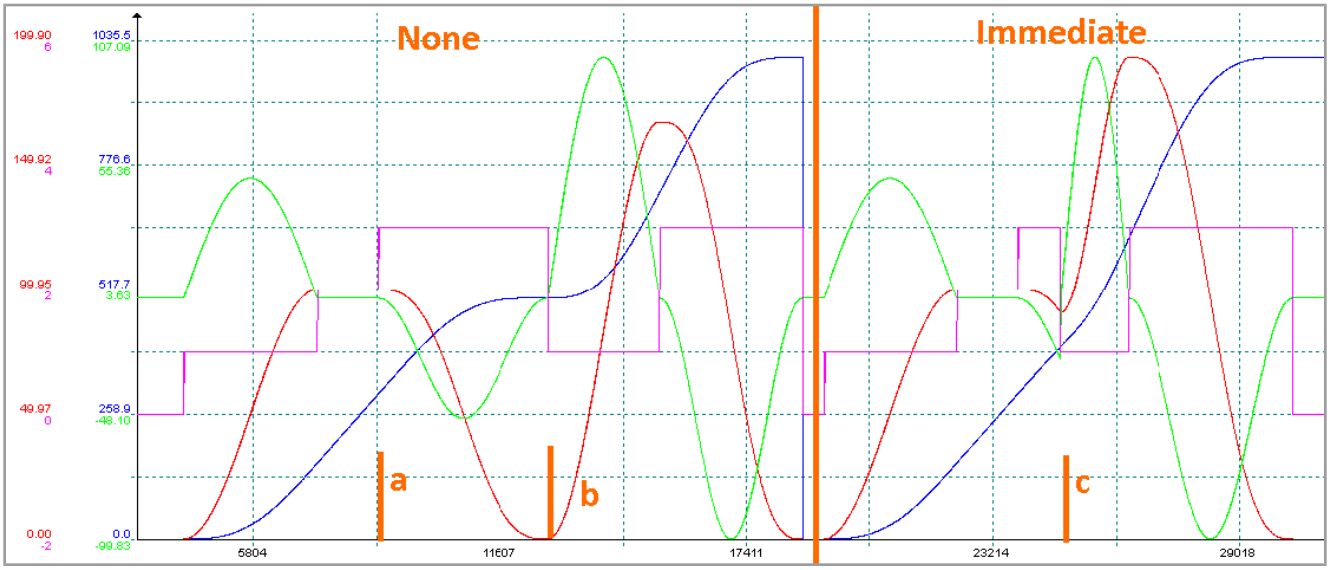


Abb. 3: Konfiguration Rampenunterbrechung

Legende

- Sollposition
- Sollgeschwindigkeit
- Sollbeschleunigung
- Rampenstatus

Positionierung *None*		Positionierung *Immediate*	
a	Zeitpunkt, an dem im Code die neue Positionierung gestartet wird.	**c**	Zeitpunkt, an dem im Code und real die neue Positionierung gestartet wird. Die aktuelle Verzögerung wird abgebrochen und in die Beschleunigung der neuen Positionierung gewechselt.
b	Zeitpunkt, an dem real die neue Positionierung gestartet wird.		

Tab. 4: Positionierung Rampenunterbrechung

6.2 Freigeben/Sperren von Achsen

- Real
- Simulation
- Virtuell
- Extern
- Mitlauf

Um Achsen freizugeben und zu sperren, wird das Power-Objekt der MotionAPI verwendet.

6.2.1 Power.Enable(), Power.Disable();

Anschließend kann der Betriebszustand abgefragt werden, um sicherzustellen, dass die Aktion erfolgreich war.

```
xAxis1.Power.Enable();
    when xAxis1.State.IsEnabled continue;
xAxis1.Power.Disable(MCPowerDisableModes.Forced);
    when xAxis1.State.IsDisabled continue;
```

6.2.2 Drive.IsReadyForOperation

- Bei realen Achsen empfiehlt es sich, vor der Freigabe zu prüfen, ob der Servoverstärker betriebsbereit ist.
- Es kann im Vorfeld dann genauer analysiert und für eine Fehlerausgabe aufbereitet werden, warum der Servoverstärker nicht betriebsbereit ist.
- Es ist zwar auch möglich, ein Enable-Kommando an einen nicht betriebsbereiten Servoverstärker zu senden, allerdings wird dies mit einem Achsfehler quittiert.
- Gerade in Applikationen mit mehreren Achsen sollten alle Achsen erst dann freigegeben werden, wenn diese auch freigegeben werden können.

```
    if xAxis1.Drive.IsReadyForOperation then
xAxis1.Power.Enable();
    when xAxis1.State.IsEnabled continue;
    else
    // inform user that axis is not operational
    end_if;
```

6.2.3 Power.QuickStop()

Als Alternative zum direkten Sperren steht auch ein Schnellhalt zur Verfügung.
Hierbei wird mit der Notstopprampe im Drehzahlregelkreis gestoppt und anschließend gesperrt.

```
    xAxis1.Power.QuickStop();
    when xAxis1.State.IsDisabled continue;
```

6.3 Referenzieren

6.3.1 Referenz setzen

- Real
- Simulation
- Virtuell
- Extern
- Mitlauf

Mit <Axis>.MoveHome.SetReference() wird eine Achse auf die angegebene Position gesetzt.

```
xAxis1.MoveHome.SetReference(0.0);
when xAxis1.MoveHome.IsReferenceSet continue;
```

Nach dem Setzen der Position wechselt der Referenz-Status auf „IsReferenceSet“.

6.3.2 Referenzfahrt

- Real
- Simulation
- Virtuell
- Extern
- Mitlauf

Mit **<Axis>.MoveHome.Start** wird eine Referenzfahrt gestartet. Beispielhaft ist hier ein Aufruf zu sehen.

```
xAxis1.MoveHome.Start(
Directions.Positive, // Direction
ReferencingSwitchConfigs.RefAndLimitSwitch, // SwitchConfig
ReferencingModes.SwitchingEdge_2Phase, // Mode
100.0, // Acceleration
50.0, // SpeedSwitchSearch
10000.0, // MaxDistanceSwitchSearch
10.0, // SpeedReferenceSearch
100.0, // MaxDistanceReferenceSearch
200.0, // ReferenceOffset
0.0 // NormalPosition
);
when xAxis1.MoveHome.IsReferenceSet continue;
```

- Alle Parameter des Aufrufs sind optional.
- Wurden bei der Inbetriebnahme über das MotionSetup auch die Referenzfahrt getestet und die nötigen Parameter eingestellt, so werden diese auch in der Initialisierungsdatei abgespeichert.
- Nach dem Übertragen der Datei auf die Steuerung werden die ermittelten Referenzparameter gesetzt.

Es reicht dann folgender Aufruf:

```
xAxis1.MoveHome.Start();
when xAxis1.MoveHome.IsReferenceSet continue;
```

Sollen nur Teile der Referenzparameter verändert werden, der Rest aber bleiben soll wie bei der Inbetriebnahme oder wie es zuvor konfiguriert war, dann werden nur die geänderten Parameter angegeben.

In diesem Beispiel soll der Abstand zum Referenzschalter und die Referenzposition geändert werden:

```
xAxis1.MoveHome.Start(
    , // Direction
    , // SwitchConfig
    , // Mode
    , // Acceleration
    , // SpeedSwitchSearch
    , // MaxDistanceSwitchSearch
    , // SpeedReferenceSearch
    , // MaxDistanceReferenceSearch
    m_nNewReferenceOffset, // ReferenceOffset
    m_nNewNormalPosition // NormalPosition
    );
    when xAxis1.MoveHome.IsReferenceSet continue;
```

ⓘ Info **Weiterführende Informationen**

Weiterführende Informationen zum Thema finden Sie in der JetSym-Online-Hilfe.

Referenzfahrtparameter

Der Referenzpunkt kann zum einen die Position des Nullimpulses sein oder die Position der Schalterflanke, wenn eine Referenzfahrt ohne Nullimpuls durchgeführt wird.

Die Referenzfahrtparameter lassen sich nicht nur im Befehl **<Axis>.MoveHome.Start()** konfigurieren, sondern sind auch einzeln einstellbar. Die jeweiligen Variablen sind in Klammern angegeben.

Direction (<Axis>MoveHome.Direction)

- Directions.Positive
- Directions. Negative

Legt fest in welcher Richtung der Referenzschalter gesucht werden soll.

SwitchConfigs (<Axis>MoveHome.SwitchConfig)

- ReferencingSwitchConfigs.NoSwitch
  - Ist auszuwählen, falls ausschließlich auf den K0 referenziert wird.
- ReferencingSwitchConfigs.RefAndLimitSwitch
  - Referenzfahrt mit Referenzschalter und Endschalter. Wird bei der Referenzschaltersuche ein Endschalter betätigt, dann wird gestoppt und in Gegenrichtung der Referenzschalter gesucht.
- ReferencingSwitchConfigs.RefSwitch
  - Bei der Referenzfahrt wird nur auf den Referenzschalter geschaut. Dies ist oft der Fall bei rotatorischen Achsen oder fehlenden Endschaltern.
- ReferencingSwitchConfigs.LimitSwitch
  - Es wird auf den Endschalter referenziert. Welcher Endschalter verwendet wird, hängt von der Richtung der Referenzfahrt ab. Wird die Referenzfahrt in positive Richtung gestartet, so wird auf den positiven Endschalter referenziert.

Mode (<Axis>MoveHome.Mode)

- ReferencingModes.ZeroPulse_2Phase

Es gibt 2 Referenzphasen:

1. Grobe Suchphase: meist schneller konfiguriert (SpeedSwitchSearch), um zügig den Nullimpuls K0 zu finden

2. Feine Suchphase: meist langsamer konfiguriert (SpeedReferenceSearch), um genauer die Flanke des Nullimpulses K0 zu finden

– ReferencingModes.SwitchingEdge_2Phase

Es gibt 2 Referenzphasen:

1. Grobe Suchphase: meist schneller konfiguriert (SpeedSwitchSearch), um zügig den Referenzschalter zu finden
2. Feine Suchphase: meist langsamer konfiguriert (SpeedReferenceSearch), um genauer die Flanke des Referenzschalters zu finden

– ReferencingModes.ZeroPulse_1Phase
  – Eine Suchphase: meist langsam konfiguriert (SpeedReferenceSearch), um genauer die Flanke des Nullimpuls K0 zu finden

– ReferencingModes.SwitchingEdge_1Phase
  – Eine Suchphase: meist langsam konfiguriert (SpeedReferenceSearch), um genauer die Flanke des Referenzschalters zu finden

Acceleration (<Axis>MoveHome.Acceleration)

Beschleunigung und Verzögerung während der Referenzfahrt.

SpeedSwitchSearch (<Axis>MoveHome.SpeedSwitchSearch)

Geschwindigkeit in der „groben“ Suchphase. Diese wird meist höher als die SpeedReferenceSearch gewählt, um zügig den Referenzpunkt zu finden.

MaxDistanceSwitchSearch (<Axis>MoveHome.MaxDistanceSwitchSearch)

Maximale Strecke bis der Referenzpunkt in der „groben“ Suchphase gefunden werden soll. Wird diese Distanz während der Referenzfahrt erreicht, wird ein Fehler generiert.

SpeedReferenceSearch (<Axis>MoveHome.SpeedReferenceSearch)

Geschwindigkeit in der „feinen“ Suchphase. Diese wird meist niedriger als die SpeedSwitchSearch gewählt, um die Flanke des Referenzpunkts zu bestimmen.

MaxDistanceReferenceSearch (<Axis>MoveHome.MaxDistanceReferenceSearch)

Maximale Strecke bis der Referenzpunkt in der „feinen“ Suchphase gefunden werden soll. Wird diese Distanz während der Referenzfahrt erreicht, wird ein Fehler generiert.

ReferenceOffset (<Axis>MoveHome.ReferenceOffset)

Abstand vom gefundenen Referenzpunkt.
Wurde während der Referenzfahrt die Flanke des Referenzpunktes bestimmt, so wird die Achse um diesen ReferenceOffset positioniert. Ist z.B. der ReferenceOffset = 0 mm, so wird auf die Flanke positioniert.

NormalPosition (<Axis>MoveHome.NormalPosition)

Position, die die Achse nach Erreichen des ReferenceOffset haben soll.

Beispiel:
ReferenceOffset = 100 mm, NormalPosition = 20 mm:
Bei der Referenzsuche wird die Schalterflanke gefunden und 100 mm weiterpositioniert. Dort wird nun die NomalPosition gesetzt, so dass am Ende der Referenzfahrt die Achse nun auf 20 mm steht.

Überwachen der Referenzfahrt

Die unterschiedlichen Phasen der Referenzfahrt werden über einen Zustand abgebildet und können dementsprechend abgefragt werden.

<Axis>MoveHome.IsReferenceRunStarted

Die Referenzfahrt wird begonnen und die Flanke des Referenzpunktes wird gesucht.

<Axis>MoveHome.IsSwitchFound

Die Schalterflanke des Referenzpunktes wurde in der „groben“ Suchphase gefunden. Die Achse sucht nun in der „feinen“ Suchphase erneut nach der Flanke.

<Axis>MoveHome.IsReferenceFound

Die Schalterflanke des Referenzpunktes wurde in der „feinen“ Suchphase gefunden. Die Achse fährt nun zu der Position, die sich aufgrund des Referenzoffsets ergibt.

<Axis>MoveHome.IsReferenceSet

Die Referenzfahrt ist beendet und die Grundstellung (NormalPosition) ist gesetzt.

6.4 Positionierung

- Real
- Simulation
- Virtuell
- Extern
- Mitlauf

Eine Achse lässt sich entweder mit Punkt-zu-Punkt (MovePtp) oder endlos (MoveVelocity) positionieren.

6.4.1 Rampenstatus

Eine Positionierung gliedert sich in 4 Phasen:

- Angehalten
- Beschleunigungsphase
- Konstantfahrphase
- Verzögerungsphase

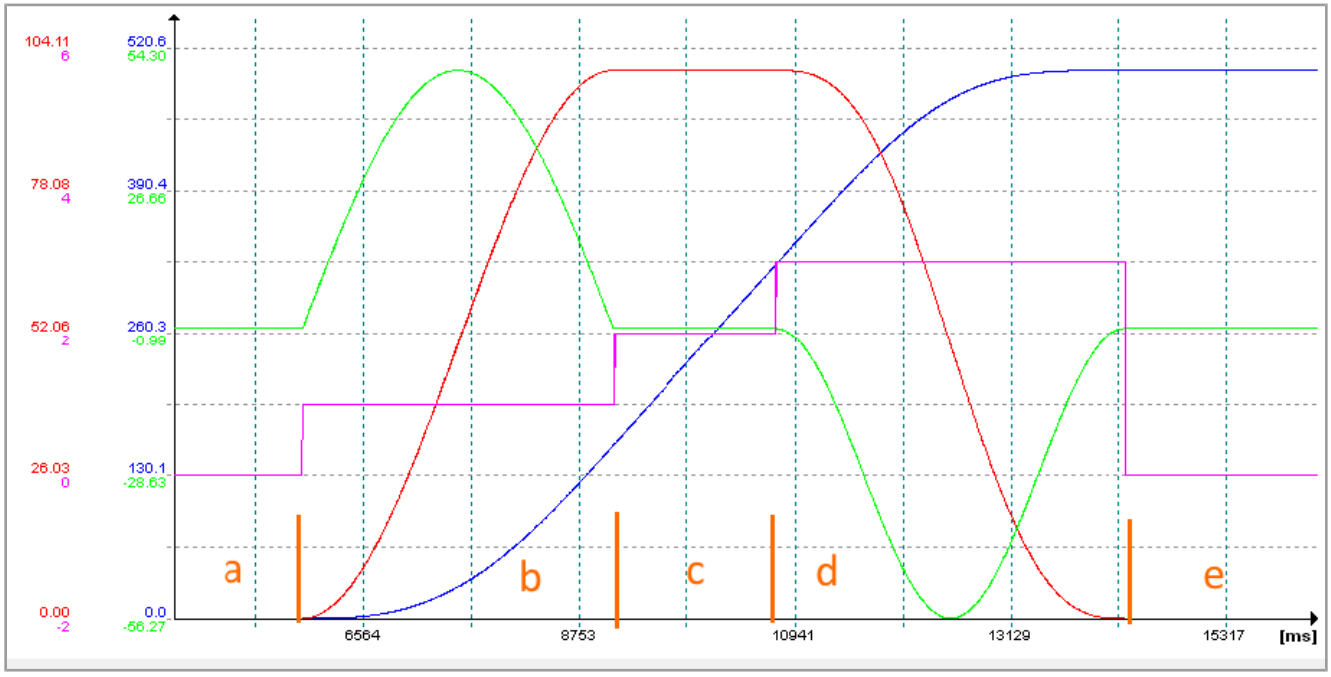


Abb. 4: Rampenstatus

Legende

- Sollposition
- Sollgeschwindigkeit
- Sollbeschleunigung
- Rampenstatus

a
Angehalten

– Achse ist angehalten
– Soll-Position verändert sich nicht
– Soll-Geschwindigkeit und Soll-Beschleunigung gleich null
– Rampenstatus ist angehalten (0: IsStopped)

b
Beschleunigungsphase

– Achse beschleunigt
– Soll-Position verändert sich
– Soll-Geschwindigkeit steigt an
– Soll-Beschleunigung steigt an und fällt bis zum Erreichen der Zielgeschwindigkeit wieder ab
– Rampenstatus ist „beschleunigend“ (1: IsAccelerating)

c
Konstantfahrphase

– Achse fährt mit konstanter Geschwindigkeit
– Soll-Position steigt nun linear an
– Soll-Geschwindigkeit bleibt konstant
– Soll-Beschleunigung ist 0
– Rampenstatus ist „konstant fahrend“ (2: IsAtConstantSpeed)

d
Verzögerungsphase

– Achse verzögert
– Soll-Position verändert sich
– Soll-Geschwindigkeit nimmt ab
– Soll-Beschleunigung steigt an und fällt bis zum Erreichen des Ziels wieder ab
– Rampenstatus ist „verzögernd“ (3: IsDecelerating)

e
Siehe a.

Je nach Einstellung der Positionierparameter kann auch die Konstantfahrt entfallen, wenn direkt von der Beschleunigung in die Verzögerung übergegangen wird.

Abfrage in STX

```
if xAxis1.Mechanism.Slope.IsAccelerating then
elseif xAxis1.Mechanism.Slope.IsAtConstantSpeed then
elseif xAxis1.Mechanism.Slope.IsDecelerating then
elseif xAxis1.Mechanism.Slope.IsStopped then
elseif xAxis1.Mechanism.Slope.SlopeState == AxisSlopeStates.Stopped then
end_if;
```

6.4.2 Punkt-zu-Punkt (MovePtp)

– Eine Punkt-zu-Punkt-Positionierung erfolgt mit **<Axis>.MovePtp.Start** oder **<Axis>.MovePtp.StartNewTargetPosition**.
  Hierbei fährt die Achse entsprechend der Rampenkonfiguration und den eingegebenen Parametern zu dem angegebenen Ziel.
– Bei **<Axis>.MovePtp.Start** besteht die Möglichkeit, Modus, Ziel, Geschwindigkeit, Beschleunigung, Verzögerung und Zielfenster einzugeben.
  Sollen Geschwindigkeit, Beschleunigung und Verzögerung nicht verändert werden, kann alternativ auch **<Axis>.MovePtp.StartNewTargetPosition** verwendet werden.

Beispiel <Axis>.MovePtp.Start

```
xAxis1.MovePtp.Start(
    AxisPositioningModes.AbsNormal, // PostioningMode
    500.0, // TargetPosition
    100.0, // Speed
    50.0, // Acceleration
    50.0, // Deceleration
    1.0 // TargetWindow
);
    when xAxis1.State.IsEnabled and xAxis1.IsInTargetWindow continue;
```

Beispiel <Axis>.MovePtp.StartNewTargetPosition

```
xAxis1.MovePtp.StartNewTargetPosition (
    AxisPositioningModes.AbsNormal, // PostioningMode
    500.0, // TargetPosition
    1.0 // TargetWindow
);
    when xAxis1.State.IsEnabled and xAxis1.IsInTargetWindow continue;
```

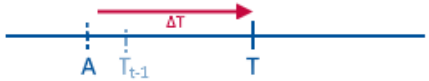
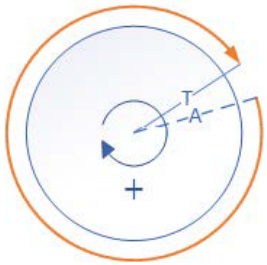
Da die Parameter beider Funktionen optional sind, kann auch über folgendes Beispiel eine neue Positionierung gestartet werden, bei der Geschwindigkeit, Beschleunigung und Verzögerung unverändert bleiben.

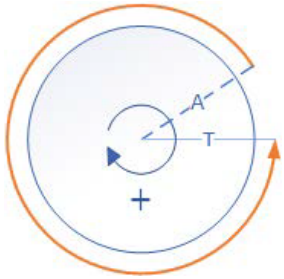
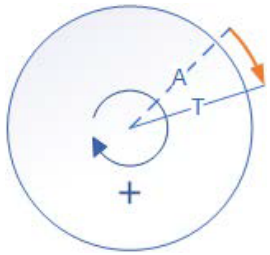
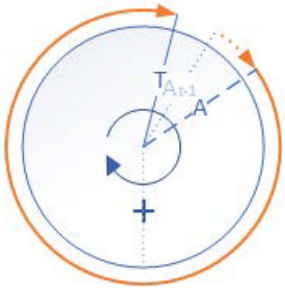
```
xAxis1.MovePtp.Start(
    AxisPositioningModes.AbsNormal, // PostioningMode
    500.0, // TargetPosition
    , // Speed
    , // Acceleration
    , // Deceleration
    1.0 // TargetWindow
);
    when xAxis1.State.IsEnabled and xAxis1.IsInTargetWindow continue;
```

Wann nehme ich MovePtp.Start und wann nehme ich MovePtp.StartNewTargetPosition?

Prinzipiell sind die Funktionen gerade im vorgenannten Beispiel redundant und gleichwertig anzusehen. Es liegt somit an der Präferenz des Programmentwicklers, welche Funktion er einsetzen möchte. Der Ersteller dieser Application Note verwendet ausschließlich MovePtp.Start.

6.4.3 PostioningMode - Positionierarten

Positionierart	Beschreibung	Grafik	Beispiel
AxisPositioningModes.AbsNormal	– Zielposition ist absolut – Nur bei normalen Achsen (nicht modulo)		
AxisPositioningModes.RelTarget	– Die neue Zielposition ist relativ zur letzten Zielposition – Wurde z.B. eine Punkt-zu-Punkt-Positionierung angehalten, bevor sie am Ziel angekommen ist, dann kann über „Start(AxisPositioning-Modes.RelTarget, 0.0….)“ zur letzten Zielposition gefahren		
AxisPositioningModes.RelActual	– Die neue Zielposition ist relativ zur aktuel-len Position – Entspricht der zur fahrenden Strecke		
AxisPositioningModes.AbsModuloPos	– Bei Modulo-Achsen – Fahrtrichtung ist positiv		Achse steht bei 50.0°, Ziel ist 40.0°, dann fährt die Achse 350° in die positive Richtung.

Positionierart	Beschreibung	Grafik	Beispiel
AxisPositioningModes.AbsModuloNeg	– Bei Modulo-Achsen – Fahrtrichtung ist negativ		Achse steht bei 40.0°, Ziel ist 50.0°, dann fährt die Achse 350° in die negative Richtung.
AxisPositioningModes.AbsModuloAuto	– Bei Modulo-Achsen – Fahrtrichtung ist der kürzeste Weg		Achse steht bei 40.0°, Ziel ist 50.0°, dann fährt die Achse 10° in die positive Richtung.
AxisPositioningModes.AbsModulo-KeepDirection	– Bei Modulo-Achsen – Fahrtrichtung entspricht der Richtung der letzten Positionierung		Achse steht bei 40.0° und positioniert auf 50°, nun soll sie mit AbsModulo-KeepDirection auf 20° fahren: Sie fährt 330° in positive Richtung.

Tab. 5: Positionierarten

Ändern der Geschwindigkeit (<Axis>.MovePtp.NewSpeed)

– Um während einer laufenden Bewegung die Geschwindigkeit der Achse zu verändern, wird über den Befehl **<Axis>.MovePtp.NewSpeed()** eine neue Geschwindigkeit vorgegeben.
– Es wird mit der eingestellten Beschleunigung auf die neue Geschwindigkeit beschleunigt oder verzögert.

```
xAxis1.MovePtp.NewSpeed(
    100.0 // Speed
);
```

Über den Parameter *Speed* wird die neue Geschwindigkeit angegeben.

Ändern der Beschleunigung (<Axis>.MovePtp.NewAcceleration)

– Mit dem Befehl **<Axis>.MovePtp.NewAcceleration ()** wird eine neue maximale Beschleunigung einer Punkt-zu-Punkt-Positionierung vorgegeben.
– Diese wird bei der nächsten Positionierung verwendet.
– Eine gerade aktive Beschleunigungsrampe wird nicht verändert.

```
xAxis1.MovePtp.NewAcceleration(
    100.0 // Acceleration
);
```

Ändern der Verzögerung (<Axis>.MovePtp.NewDeceleration)

Um während einer laufenden Verzögerung die Verzögerung der Achse zu ändern, wird über den Befehl **<Axis>.MovePtp.NewDeceleration ()** eine neue Verzögerung vorgegeben.

```
xAxis1.MovePtp.NewDeceleration(
    100.0 // Deceleration
);
```

6.4.4 Endlos-Positionierung (MoveVelocity)

Eine Endlos-Positionierung erfolgt mit **<Axis>.MoveVelocity.Start()**. Hierbei beschleunigt die Achse mit der eingestellten Beschleunigung auf die Zielgeschwindigkeit.

```
xAxis1.MoveVelocity.Start(
    Directions.Positive, // Direction
    100.0, // Speed
    100.0, // Acceleration
    150.0 // Deceleration
);
```

Direction

Richtung der Endlospositionierung.

Speed

Zielgeschwindigkeit der Endlospositionierung. Wert muss immer > 0 sein.

Acceleration

Mit diesem Wert wird auf die Zielgeschwindigkeit beschleunigt. Wert muss immer > 0 sein.

Deceleration

Beim Anhalten der Endlospositionierung wird diese Verzögerung verwendet, solange sie durch das Anhaltekommando nicht überschrieben wird. Wert muss immer > 0 sein.

Ändern der Richtung (<Axis>.MoveVelocity.Reverse)

Um die Bewegungsrichtung umzukehren, kann der Befehl **<Axis>.MoveVelocity.Reverse()** verwendet werden. Es wird dann mit der eingestellten Beschleunigung auf die gleiche Geschwindigkeit in Gegenrichtung gewechselt.

```
xAxis1.MoveVelocity.Reverse();
```

Alternativ zum vorigen Befehl, kann auch über ein erneutes **<Axis>.MoveVelocity.Start()** gegeben werden, nur diesmal mit der Gegenrichtung als Richtungsvorgabe.

```
xAxis1.MoveVelocity.Start(
    Directions.Negative, // Direction
    100.0, // Speed
    100.0, // Acceleration
    150.0 // Deceleration
);
```

Ändern der Geschwindigkeit (<Axis>.MoveVelocity.NewSpeed)

Um während einer laufenden Bewegung die Geschwindigkeit der Achse zu ändern, wird über den Befehl **<Axis>.MoveVelocity.NewSpeed()** eine neue Geschwindigkeit vorgegeben. Es wird mit der eingestellten Beschleunigung auf die neue Geschwindigkeit beschleunigt oder verzögert.

```
xAxis1.MoveVelocity.NewSpeed(
    100.0 // Speed
);
```

Über den Parameter *Speed* wird die neue Geschwindigkeit angegeben. Da diese immer größer 0 sein muss, kann zwar die Geschwindigkeit geändert werden, nicht aber die Bewegungsrichtung!

Ändern der Beschleunigung (<Axis>.MoveVelocity.NewAcceleration)

Um während einer laufenden Beschleunigung die Beschleunigung der Achse zu ändern, wird über den Befehl **<Axis>.MoveVelocity.NewAcceleration ()** eine neue Beschleunigung vorgegeben.

```
xAxis1.MoveVelocity.NewAcceleration(
    100.0 // Acceleration
);
```

Ändern der Verzögerung (<Axis>.MoveVelocity.NewDeceleration)

Um während einer laufenden Verzögerung die Verzögerung der Achse zu ändern, wird über den Befehl **<Axis>.MoveVelocity.NewDeceleration ()** eine neue Verzögerung vorgegeben.

```
xAxis1.MoveVelocity.NewDeceleration(
    100.0 // Deceleration
);
```

6.4.5 Anhalten (MoveHalt)

– Eine laufende Bewegung kann über **<Axis>.MoveHalt.Start** angehalten werden.
– Die Parameter sind optional. Sind diese nicht angegeben, so werden die zu diesem Zeitpunkt gültigen Parameter verwendet.
– Es wird mit der eingegebenen Verzögerung angehalten. Ist dieser Parameter frei, so wird z.B. die Verzögerung verwendet, welche beim vorhergehenden Fahrbefehl verwendet wurde.

```
xAxis1.MoveHalt.Start(
    MCAxisHaltModes.Normal, // Mode
    85.0 // Deceleration
);
when xAxis1.State.IsEnabled continue;
```

Über die Abfrage **<Axis>.State.IsEnabled** kann das Ende des Anhaltens abgefragt werden.

```
when xAxis1.Mechanism.Slope.IsStopped continue;
```

Alternativ kann auch **<Axis>.Mechanism.Slope.IsStopped** verwendet werden.

Der Unterschied liegt darin:

– Bei **<Axis>.State.IsEnabled** steht die Achse und ist freigegeben.
– Bei **<Axis>.Mechanism.Slope.IsStopped** steht die Achse zwar auch, aber der Betriebszustand ist nicht eindeutig. Dieser kann z.B. IsEnabled, aber auch IsDisabled oder ein anderer sein.

Mode

MCAxisHaltModes.Normal

Die aktuelle Bewegung wird ab diesem Zeitpunkt mit der aktiven Verzögerung angehalten.

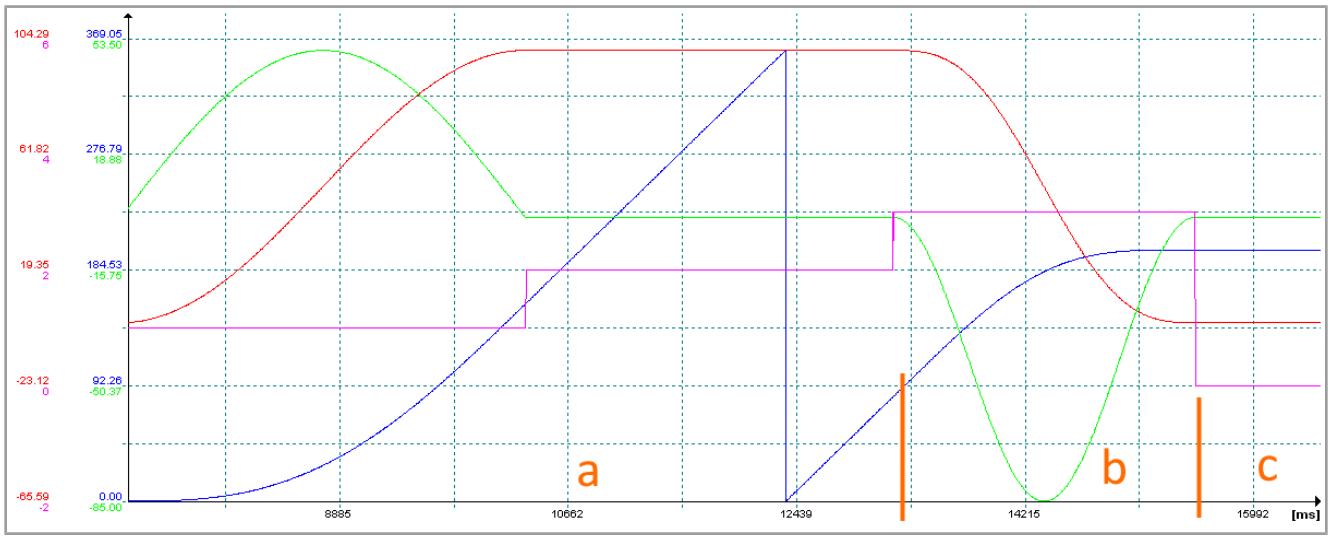


Abb. 5: Mode normal

Legende

- Sollposition
- Sollgeschwindigkeit
- Sollbeschleunigung
- Rampenstatus

a

Starten einer Endlospositionierung.

b

– Achse wird mit MCAxisHaltModes.Normal angehalten.
– Die Achse verzögert bis zum Stillstand.
– Der Rampenstatus wechselt auf „IsDecelerating".

c

– Die Achse ist nun gestoppt.
– Der Rampenstatus ist nun „IsStopped“.

MCAxisHaltModes.AtActualPosition

Die aktuelle Bewegung wird angehalten und es wird zu dem Punkt zurückgefahren, an dem das Kommando erteilt wurde.

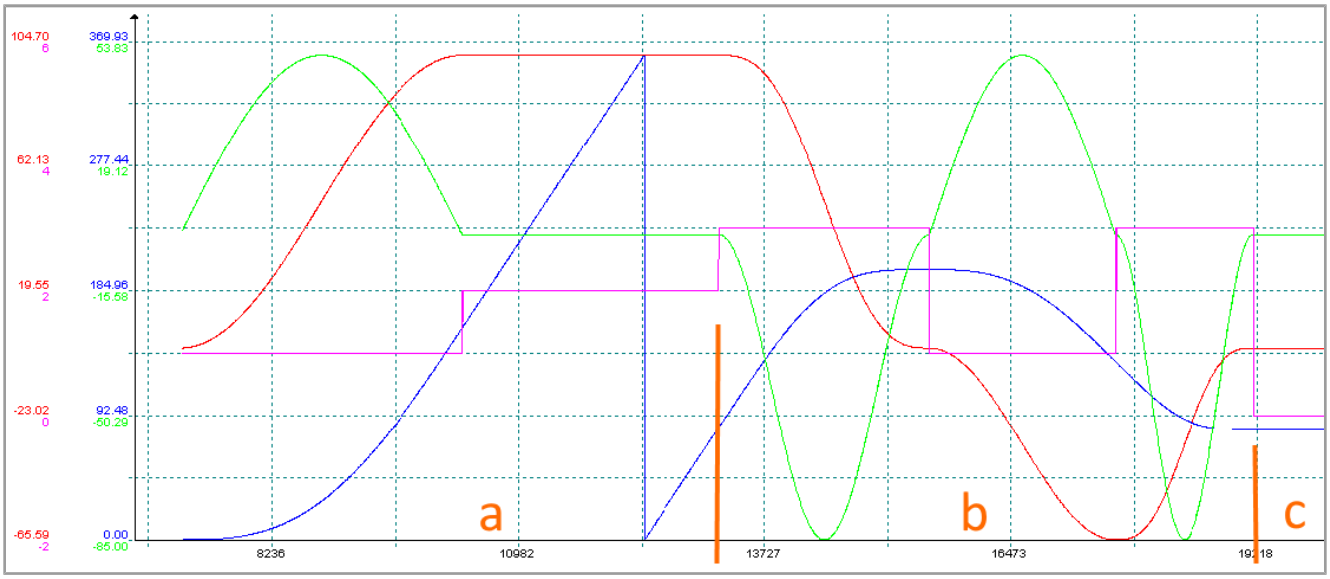


Abb. 6: Mode aktuelle Position

Legende

- Sollposition
- Sollgeschwindigkeit
- Sollbeschleunigung
- Rampenstatus

a

Starten einer Endlospositionierung.

b

– Die Achse wird mit MCAxisHaltModes.AtActualPosition angehalten.
– Zuerst verzögert die Achse bis zum Stillstand. Anschließend erfolgt eine Positionierung auf die Position zu Beginn von **b**.
– Der Rampenstatus wechselt auf „IsDecelerating“, gefolgt von „IsAccelerating“ und „IsDecelerating“ während der Rückpositionierung.

c

– Die Achse ist nun gestoppt an der Position zu Beginn von **b**.
– Der Rampenstatus ist nun „IsStopped“.

Abbildungsverzeichnis

Abb. 1 Mech. Maßeinheiten einstellen 11
Abb. 2 Rampentyp 30
Abb. 3 Konfiguration Rampenunterbrechung 31
Abb. 4 Rampenstatus 37
Abb. 5 Mode normal 44
Abb. 6 Mode aktuelle Position 45

Tabellenverzeichnis

Tab. 1 Verwendete Präfixe 5
Tab. 2 Positionierung Rampentyp 30
Tab. 3 Konfiguration Rampenunterbrechung 31
Tab. 4 Positionierung Rampenunterbrechung 32
Tab. 5 Positionierarten 40

Bucher Automation AG
Thomas-Alva-Edison-Ring 10
71672 Marbach am Neckar, Deutschland
T +49 7141 2550-0
info@bucherautomation.com

www.bucherautomation.com