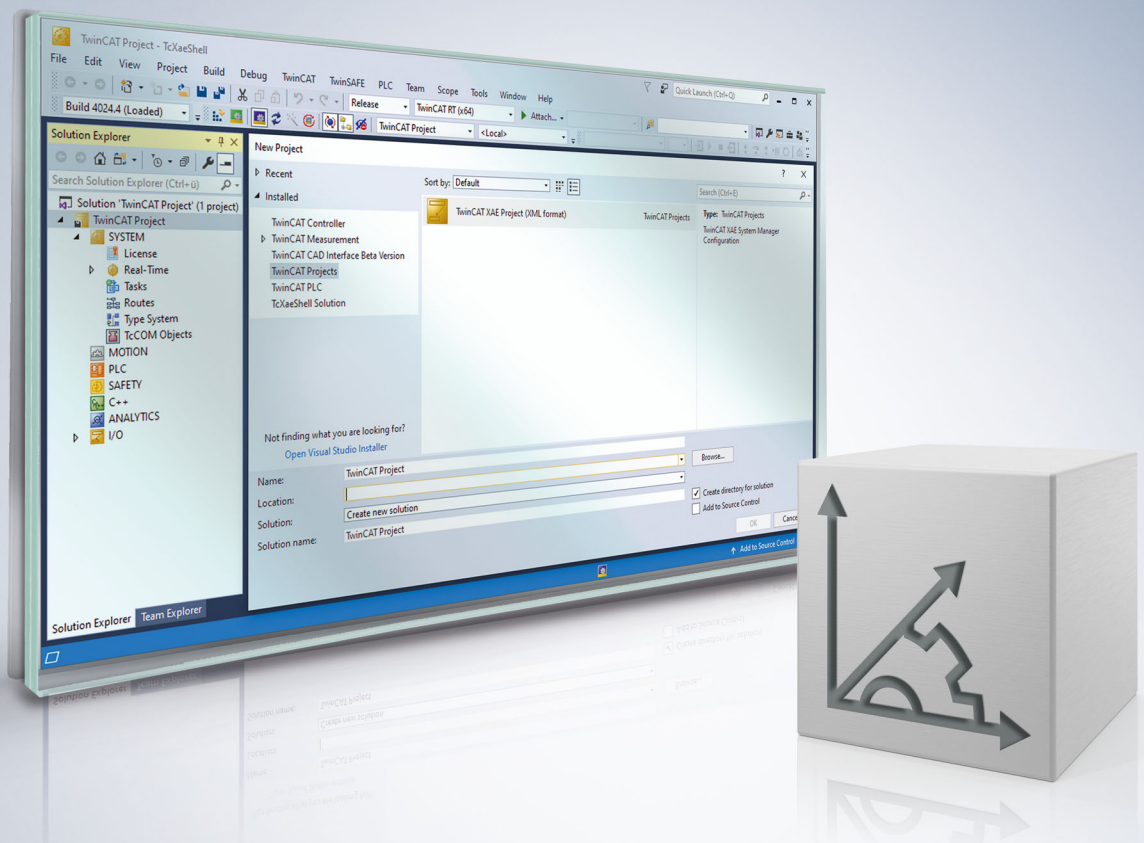
BECKHOFF New Automation Technology

Handbuch | DE

TF5100

TwinCAT 3 | NC I



12.01.2024 | Version: 2.11.0

Inhaltsverzeichnis

1 Vorwort

1.1 Hinweise zur Dokumentation

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

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

Disclaimer

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

Marken

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

Patente

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

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

Copyright

1.2 Zu Ihrer Sicherheit

Sicherheitsbestimmungen

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

Haftungsausschluss

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

Qualifikation des Personals

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

Signalwörter

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

Warnungen vor Personenschäden

⚠ GEFAHR

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

⚠ WARNUNG

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

⚠ VORSICHT

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

Warnung vor Umwelt- oder Sachschäden

HINWEIS

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

Information zum Umgang mit dem Produkt

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

1.3 Hinweise zur Informationssicherheit

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

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

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

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

2 Einleitung

TwinCAT NCI steht für 'numerical control interpolation' und ist das NC-System für interpolierende Bahnbewegungen.
TwinCAT NCI bietet eine 3D-Interpolation (Interpreter, Sollwertgenerierung, Lageregler), eine integrierte SPS mit NC-Schnittstelle und eine E/A-Anbindung für Achsen über den Feldbus.

Mit der NCI können 3 Bahnachsen und bis zu 5 Hilfsachsen pro Kanal verfahren werden. Zusätzlich können noch Master/Slave Kopplungen gebildet werden. In Kombination mit TwinCAT Kinematic Transformation (TF511x) lassen sich auch komplexe Kinematiken aus der NCI heraus ansteuern.

Die Programmierung erfolgt aus einem eigenen NC-Programm nach DIN 66025 mit eigenen Spracherweiterungen (vgl. Classic Dialect Reference Manual [▶ 128]) oder direkt aus der SPS mit der Bibliothek PLC Library: Tc2_PlcInterpolation [▶ 304].

Installationsvoraussetzung

TwinCAT NCI ist in die TwinCAT 3-Installation integriert.

Zielsystem

Windows 7, Windows 10, Windows CE (nur Classic Interpreter)

Minimum Plattform-Level: 40

Übersicht

Kapitel	Inhalt
XAE - Benutzerschnittstelle [▶ 11]	Beschreibung der Parameter und Funktionalitäten für den Interpreter in der TwinCAT Engineering Umgebung (XAE)
Interpreter [▶ 128]	Programmieranleitung des Interpreters
PLC NCI Libraries [▶ 203]	Beschreibung der speziellen NCI Bibliotheken
Beispiele [▶ 321]	Beispiele zur Verwendung von TwinCAT NCI mit SPS und Teileprogramm, sowie zur direkten Bewegungssteuerung aus der SPS mit der Tc2_PlcInterpolation-Bibliothek
Anhang [▶ 330]	Parametrierung, Zyklisches Kanalinterface

Weiterführende Informationen

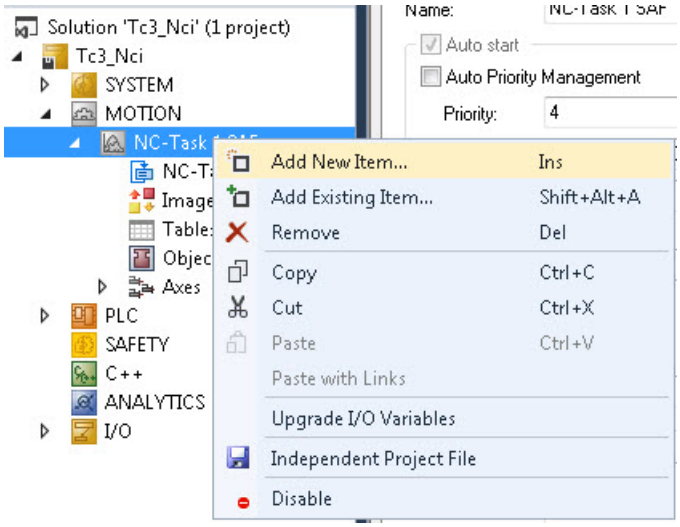
- ADS Return Codes
- ADS-Spezifikation für NC

3 Benutzerschnittstelle in der TwinCAT Engineering Umgebung

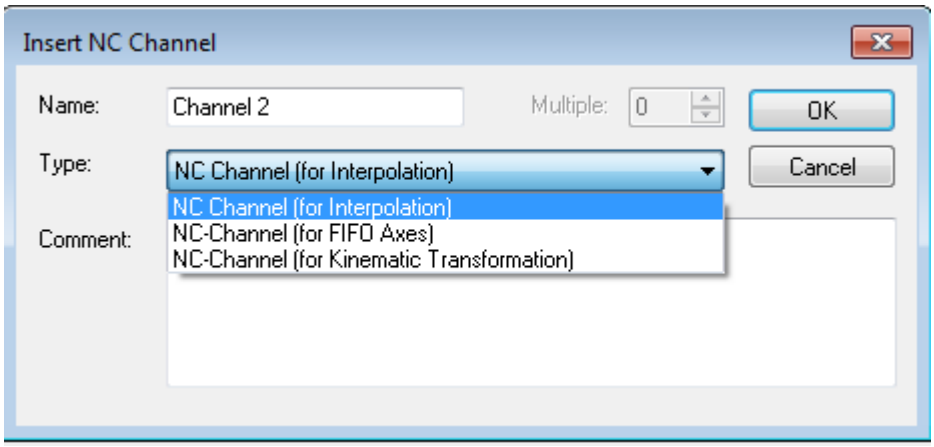
3.1 Überblick

Um die Interpolation nutzen zu können, fügen Sie einen Interpolationskanal im XAE ein. Dies gilt sowohl für die Verwendung des Interpreters als auch der PLC Library: Tc2_PlcInterpolation [▶ 304].

1. Legen Sie einen NC-Kanal an.



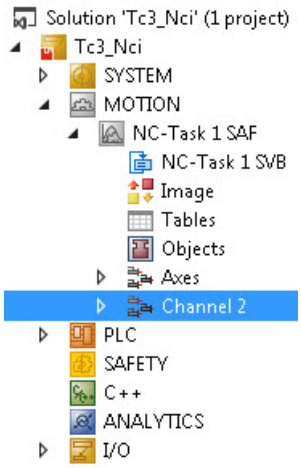
2. Wählen Sie in der Auswahlbox den NC Kanal für die Interpolation an.



3. Ordnen Sie diesem aus der SPS per Funktionsbaustein PTP-Achsen zu.

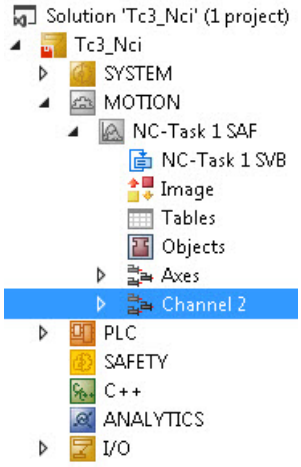
⇨ Der angelegte Kanal besteht aus folgenden Elementen:

Interpolationskanal [▶ 12]	Beschreibung der Eigenschaftsseiten, die sich hinter dem Element 'Interpolation' befinden
Interpreter Element [▶ 14]	Beschreibung der Eigenschaftsseiten, die sich hinter dem Element 'Interpreter' befinden
Gruppenelement [▶ 23]	Beschreibung der Eigenschaftsseiten, die sich hinter dem Element 'Gruppe' befinden



Hinweis Achsspezifische Parameter für die NCI befinden sich in der Achsparametrierung unter dem Unterpunkt 'NCI Parameter'.

3.2 Interpolationskanal

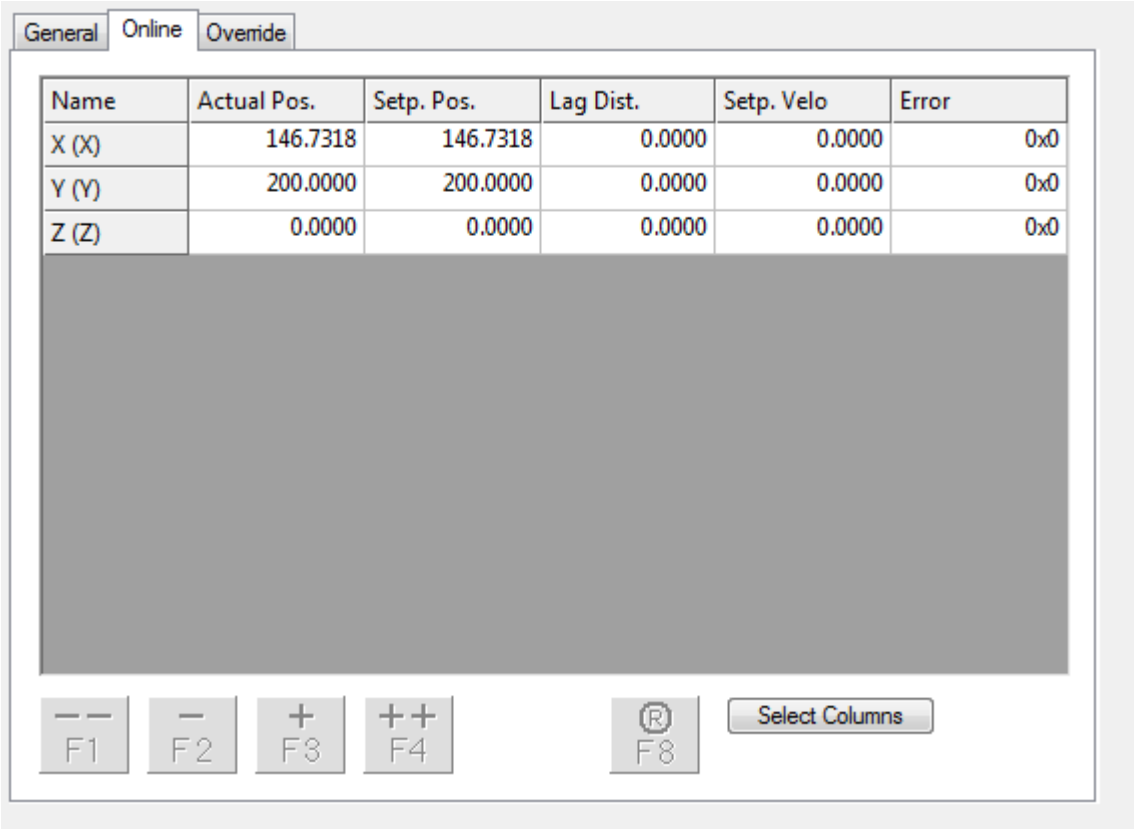


Mit einem Klick auf den Interpolationskanal werden folgende Dialoge sichtbar:

Karteireiter "Online"

Hier werden alle Achsen der aktuellen Interpolationsgruppe [▶ 23] aufgelistet. Dabei werden aktuell dargestellt:

- Istpositionen
- Sollpositionen
- Schleppabstände
- Sollgeschwindigkeiten und
- Fehlercodes

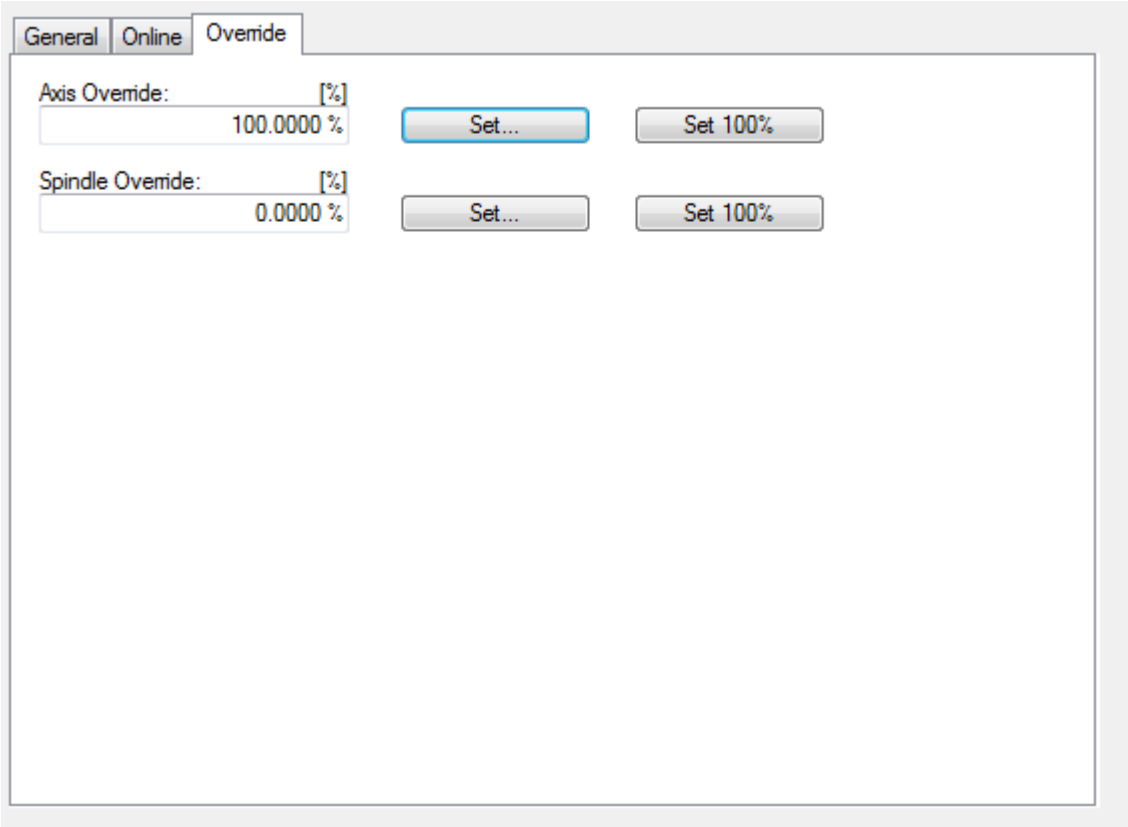


Karteireiter "Override"

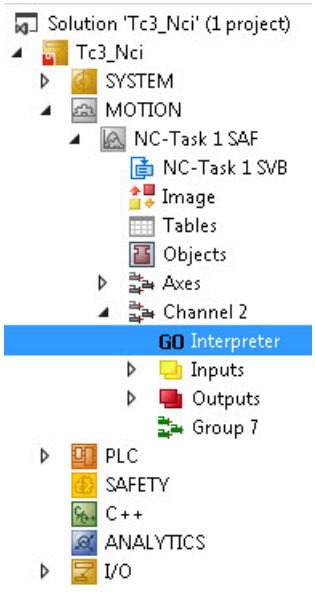
Auf der Seite 'Override' ist der Kanal-Override für die Achsen abzulesen und zu setzen. Falls die SPS läuft und dort das zyklische Kanal-Interface [▶ 335] beschrieben wird, dann wird der hier gesetzte Override durch die SPS wieder überschrieben.

Nähere Information zur Wirkungsweise des Overrides sind unter Bahnoverride (Interpreter-Overridetypen) [▶ 333] zu finden.

Der Spindel-Override wird zwar durch das zyklische Kanal-Interface beschrieben, aber derzeit nicht unterstützt.

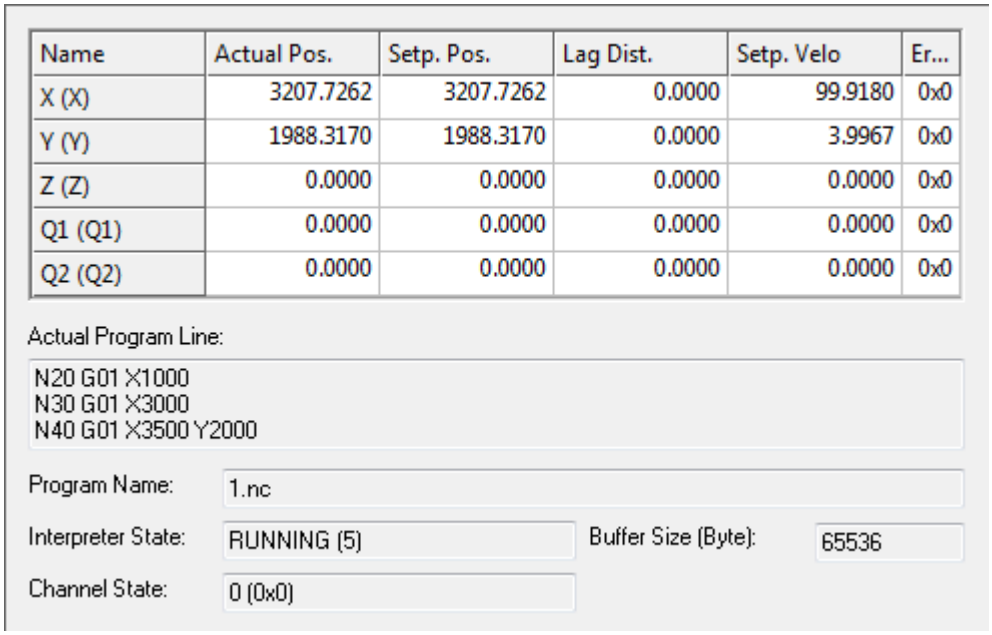


3.3 Interpreter-Element



Mit einem Klick auf „Interpreter“ werden folgende Eigenschaftsseiten und das Online Fenster sichtbar:

3.3.1 Interpreter Online-Fenster



Achsen

Wie auch auf der Eigenschaftsseite „Online“ im Interpolationskanal, werden in diesem Fenster alle Achsen, die sich aktuell in der Interpolationsgruppe befinden, aufgelistet. Dabei werden die Werte für

- Istpositionen
- Sollpositionen
- Schleppabstände
- Sollgeschwindigkeiten und
- aktuelle Fehlercodes

dargestellt.

Actual Program Line

Die Actual Program Line stellt den z. Zt. in der Satzausführung abzuarbeitenden NC Satz dar. Dabei ist die letzte Zeile in dem Fenster der aktuelle Satz.
Im Unterschied dazu ist bei GST der aktuelle Satz in der mittleren Zeile.

Wie nahezu alle Parameter, lässt sich auch die Programmanzeige via ADS auslesen. Dies können Sie z.B. dazu verwenden, in einer Visual Basic Applikation die aktuellen NC-Sätze anzuzeigen (vergl. ADS-Device-Dokumentation - ADS Interface NC).

Programmname

Zeigt den Namen des z.Zt. geladenen Programms. Dies muss nicht unbedingt das Programm sein, das im Editor dargestellt ist.

Interpreterstatus

Der Interpreterstatus gibt den aktuellen Status der Interpreter State Maschine wieder. Die vollständige Liste ist unten aufgeführt. Da für die Auswertung in der SPS nicht alle Zustände relevant sind, sollen nur die wichtigsten erläutert werden.

Status	Beschreibung
ITP_STATE_IDLE	Der Interpreter befindet sich im Idle-Zustand, wenn noch kein NC Programm geladen ist oder wenn ein Gruppen-Reset ausgeführt wird. Beim Stoppen eines laufenden Programms, geht der Interpreter ebenfalls in den

Status	Beschreibung
	Idle State. In diesem Fall ist ein Gruppen-Reset unbedingt erforderlich, da sonst Fehler 0x42C5 ausgegeben wird. Es empfiehlt sich daher nach einem Stopp aus der SPS direkt ein Gruppen-Reset durchzuführen.
ITP_STATE_READY	Nach dem erfolgreichen Laden eines NC-Programms befindet sich der Interpreter im Ready State. Wenn ein Programm erfolgreich abgearbeitet und beendet wird, befindet sich der Interpreter anschließend ebenfalls im Ready State. Zwischenzeitlich werden aber andere Zustände angenommen.
ITP_STATE_ABORTED	Tritt während der Abarbeitung des NC-Programms ein Laufzeitfehler auf, so geht der Interpreter in den Aborted State. Der eigentliche Fehlercode ist dem Kanalstatus zu entnehmen.
ITP_STATE_SINGLESTOP	Dieser Status wird **nur** im Einzelsatzbetrieb [▶ 135] angenommen. Sobald der Eintrag aus dem Interpreter an den NC-Kern gesendet wird, geht der Interpreter in diesen Zustand.

Abfrage des Interpreterstatus während der Ausführung des Programms

Da der Interpreterstatus während der Ausführung des Programms zwischen verschiedenen Zuständen wechseln kann, wird empfohlen, ihn mit einer negativen Logik abzufragen. Während der Abarbeitung des Programms ist der Interpreter nicht zwingend im State ITP_STATE_RUNNING. Wurde das Programm erfolgreich ausgeführt, ist der Interpreter anschließend immer im Ready State (siehe auch Beispiele [▶ 321]).

Programmende

Das Programmende wird durch eine M-Funktion gekennzeichnet. Hierfür wird entweder M2 oder M30 verwendet. Fehlt diese M-Funktion am Programmende kann der Interpreter-Status falsche Werte annehmen.

Rückgabewerte Interpreterstatus

```
0 ITP_STATE_INITFAILED
1 ITP_STATE_IDLE
2 ITP_STATE_READY
3 ITP_STATE_STARTED
4 ITP_STATE_SCANNING
5 ITP_STATE_RUNNING
6 ITP_STATE_STAY_RUNNING
7 ITP_STATE_WRITETABLE
8 ITP_STATE_SEARCHLINE
9 ITP_STATE_END
10 ITP_STATE_SINGLESTOP
11 ITP_STATE_ABORTING
12 ITP_STATE_ABORTED
13 ITP_STATE_FAULT
14 ITP_STATE_RESET
15 ITP_STATE_STOP
16 ITP_STATE_WAITFUNC
17 ITP_STATE_FLUSHBUFFERS
```

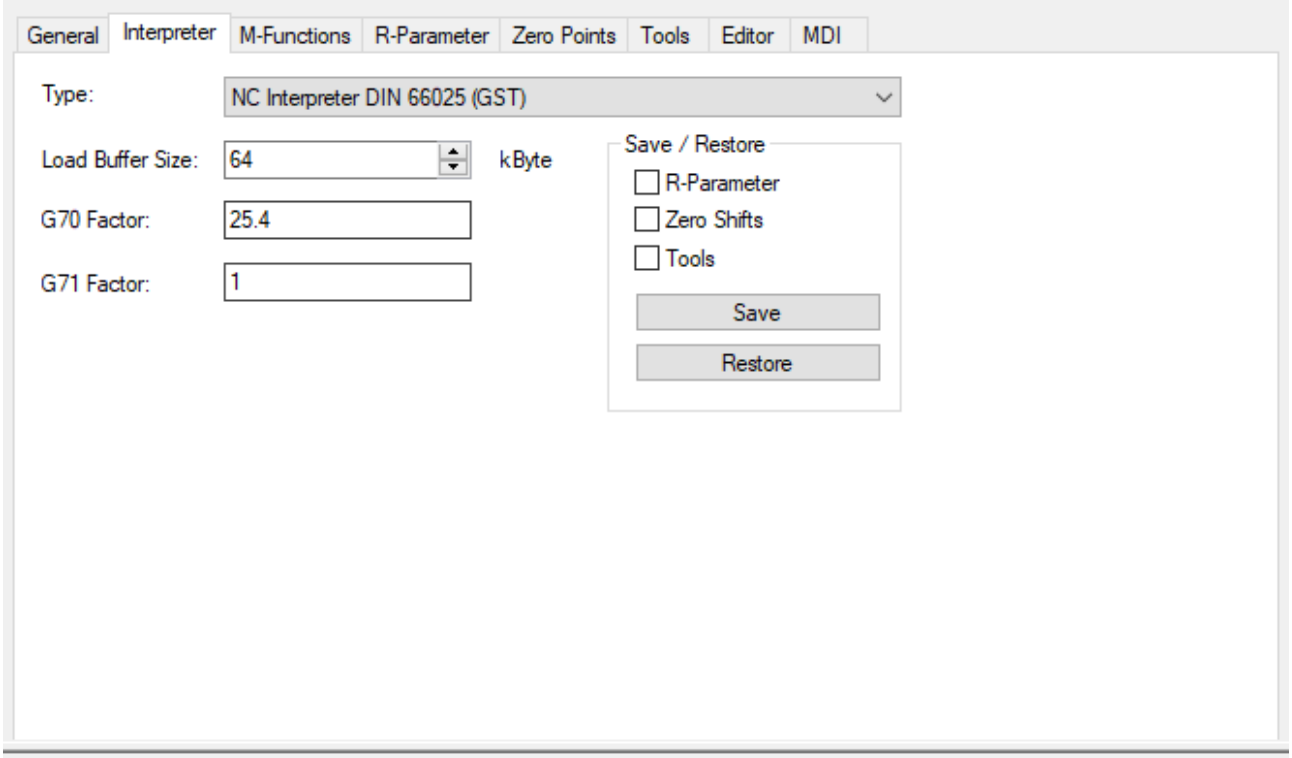
Kanalstatus

Der Kanalstatus gibt den aktuellen Fehlerstatus des Kanals wieder. D.h. tritt zur Lade- oder Laufzeit des NC-Programms ein Fehler auf, so wird hier der dazugehörige Fehlercode angezeigt. Geht z.B. eine Achse während der Bearbeitung in einen Schleppfehler, so wird das NC-Programm gestoppt und der Kanalstatus hat einen Wert ungleich 0. Deshalb sollte der Kanalstatus unbedingt in der SPS überprüft werden, damit auf Fehler reagiert werden kann. Im fehlerfreien Betrieb ist der Kanalstatus immer 0.

Ladepuffer

Hier wird die aktuelle Größe des Ladepuffers für den Interpreter angezeigt. Um den Wert zu verändern, muss der Karteireiter „Interpreter“ angewählt werden.

3.3.2 Karteireiter "Interpreter"



Typ

In der Auswahlbox „Typ“ kann der Interpreter-Typ ausgewählt werden. Zur Verfügung stehen

- der GST-Interpreter [▶ 31]. GST kombiniert nativen DIN 66025 basierten G-Code mit Programmiererweiterungen durch Strukturierten Text als eine höhere Sprache.
- Der NC-Interpreter [▶ 128] (Classic Dialect) basierend auf DIN 66025 mit Registerfunktionen-Erweiterungen als `@`-Kommandos.
- Die Auswahl keines Interpreters, wenn die PlcInterpolation-Bibliothek [▶ 304] verwendet wird.

Als Standardeinstellung ist der GST-Interpreter eingestellt. Um den NC-Interpreter mit Registerfunktionen-Erweiterungen zu benutzen, müssen Sie ihn explizit auswählen.

Ladepuffergröße

Hier kann die Ladepuffergröße für den Interpreter editiert werden. Dabei ist zu beachten, dass der benötigte Speicher im Interpreter wesentlich größer ist als die Größe der NC-Datei. Die maximal erlaubte Ladepuffergröße ist auf 64 MB begrenzt.

Änderung der Ladepuffergröße

Wenn die Ladepuffergröße verändert wird, ist unbedingt ein TwinCAT-Restart notwendig.

`G70`/`G71` Faktor

Wird im Teileprogramm von G71 [▶ 134] (Millimeter - default) auf `G70` umgeschaltet, so ist hier der Umrechnungsfaktor hinterlegt. Dieser Umrechnungsfaktor muss nur dann editiert werden, wenn das Basisbezugssystem nicht Millimeter ist.

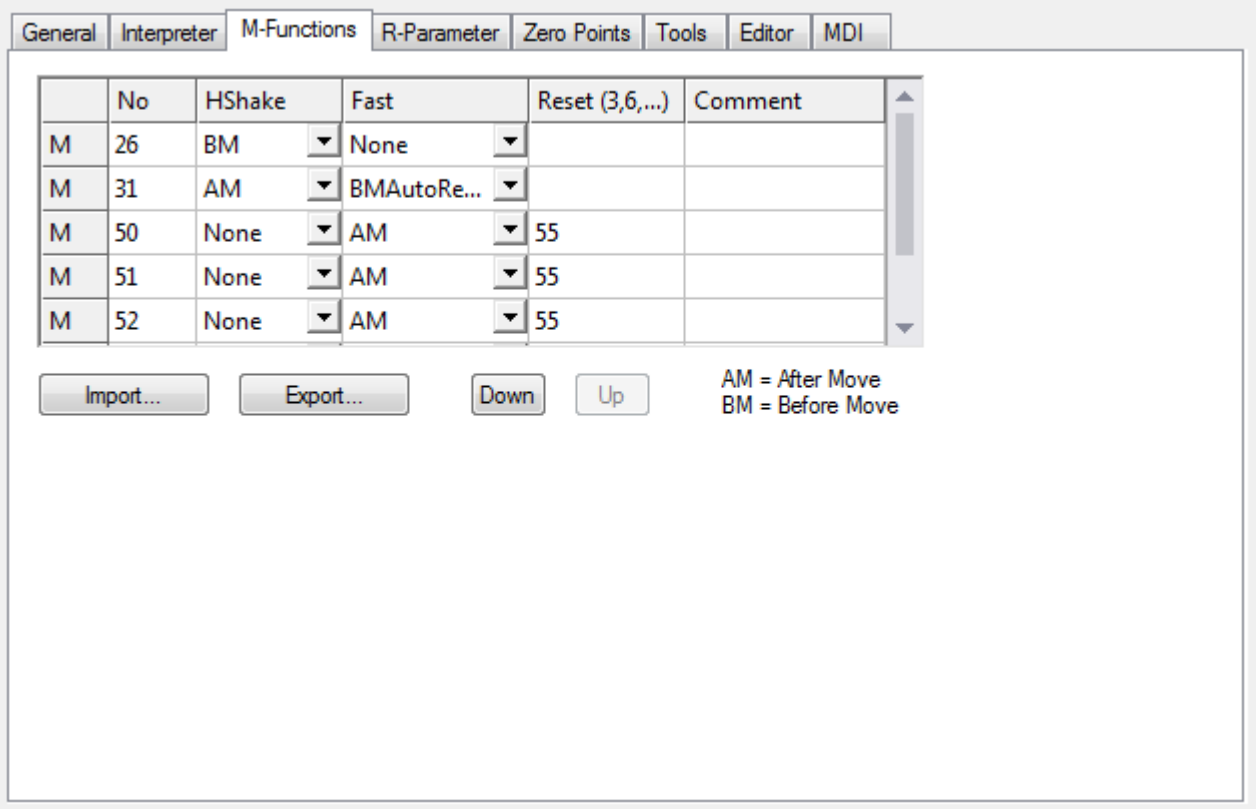
Ist die Maschine z. B. auf Inch eingemessen und im Teileprogramm wird `G70` aktiv geschaltet, dann muss der `G70`-Faktor gleich `1` gesetzt werden und muss der `G71`-Faktor gleich `1/25.4` gesetzt werden.

Save/Restore

Mit der Save-Funktion kann zur Laufzeit ein „Schnappschuss“ von den gerade aktuellen Parametern gesichert werden. Welche Parameter dabei berücksichtig werden sollen, wird mit den Checkboxen festgelegt. Die Save-Funktion generiert dabei die Datei ‚SnapShot.bin‘ im TwinCAT\CNC-Verzeichnis.

Die Restore-Funktion lädt die mit „Save“ beschriebene Datei. Diese Funktion ist ausschließlich für Debugzwecke vorgesehen.

3.3.3 Karteireiter "M-Functions"



Verwendung ausschließlich mit Interpreter

Dieser Karteireiter hat für den Betrieb mit der Library Tc2_PlcInterpolation keine Bedeutung.

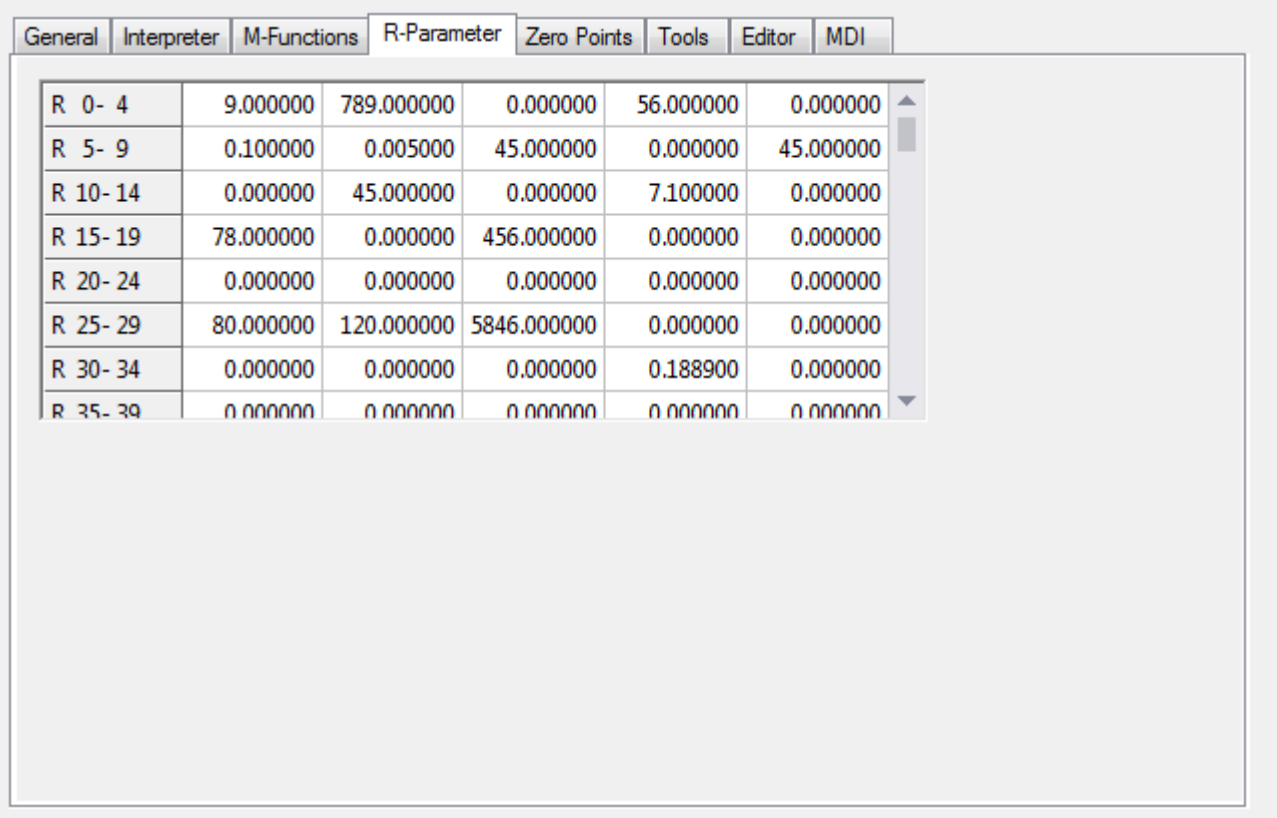
Hier werden die aktuell parametrierten M-Funktionen dargestellt. Auf dieser Seite können neue M-Funktionen hinzugefügt bzw. bestehende anders parametriert werden.

Eine nähere Beschreibung der möglichen Parameter ist der Interpreterbeschreibung unter M-Funktionen [▶ 167] zu entnehmen.

Parametrierung von M-Funktionen

Wenn M-Funktionen umparametriert werden, sind anschließend eine Aktivierung der Konfiguration und ein TwinCAT Neustart erforderlich.

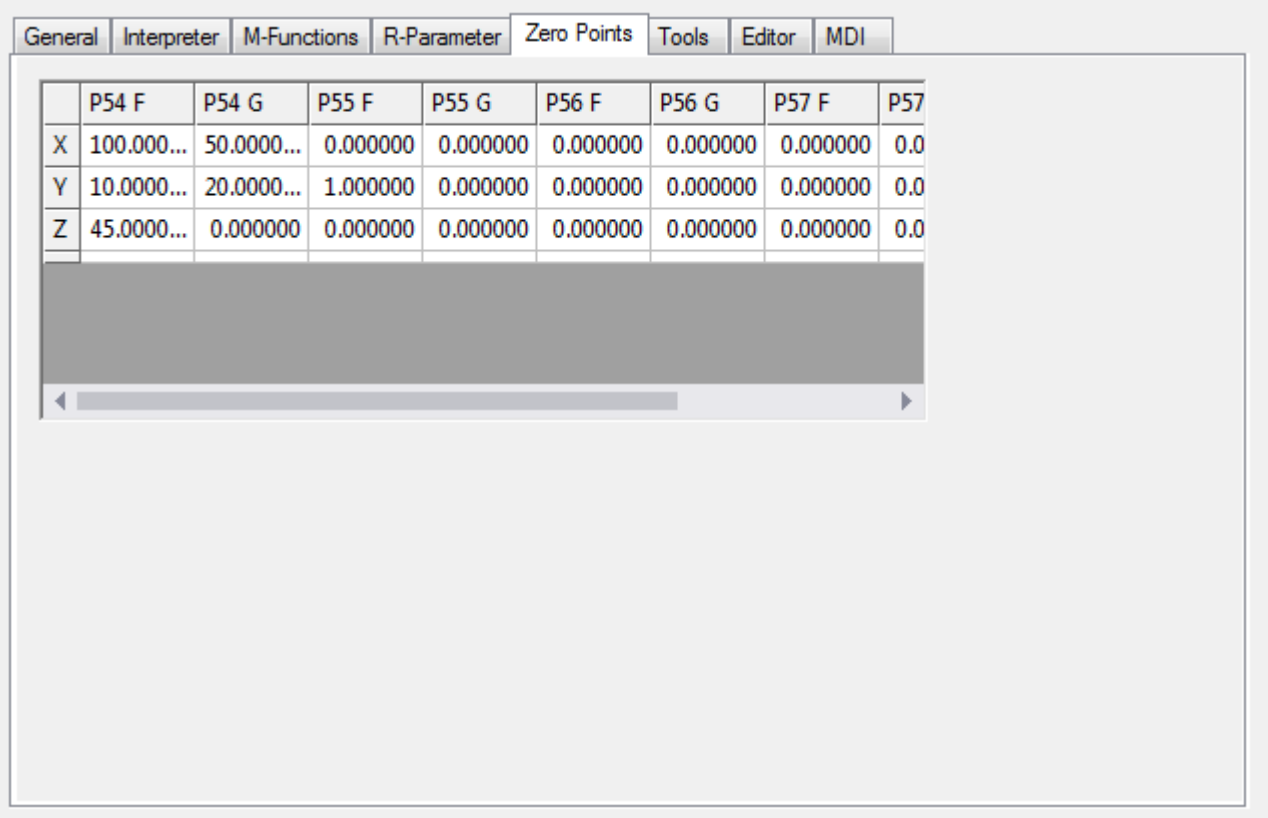
3.3.4 Karteireiter "R-Parameter"



Auf der Eigenschaftsseite „R-Parameter“ werden die zurzeit gültigen R-Parameter angezeigt. Während der Testphase ist es hier z.B. möglich, R-Parameter zu initialisieren oder zu verändern. Das Editieren der R-Parameter sollte allerdings im Regelfall aus dem NC-Programm oder ggf. aus der SPS erfolgen.

Näheres zur Verwendung von R-Parametern ist in der Interpreterbeschreibung unter R-Parameter [▶ 136] zu finden.

3.3.5 Karteireiter "Nullpunkte"

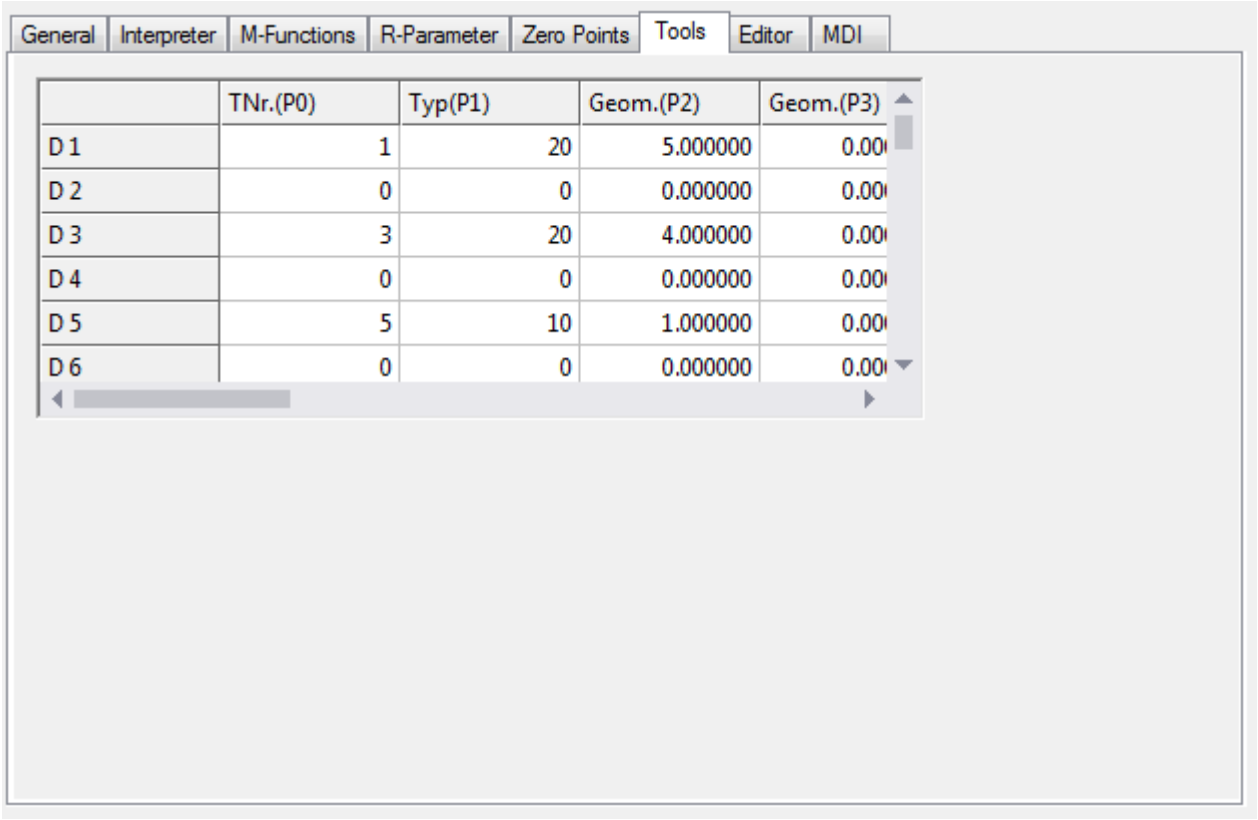


Hier werden die aktuellen Nullpunktverschiebungen für die Achsen, die sich in der Interpolationsgruppe befinden, dargestellt. Die Parameter P54..P59 stehen für den entsprechenden G Code. Wie auch bei den R-Parametern, lassen sich an dieser Stelle die Nullpunktverschiebungen editieren.

Hinweis Die Spalten F & G (z.B. P54 F & P54 G) sind historisch bedingt und werden jeweils für einen Parameter addiert.

Näheres zur Wirkungsweise ist in der Interpreterbeschreibung unter Nullpunktverschiebungen [▶ 146] zu finden.

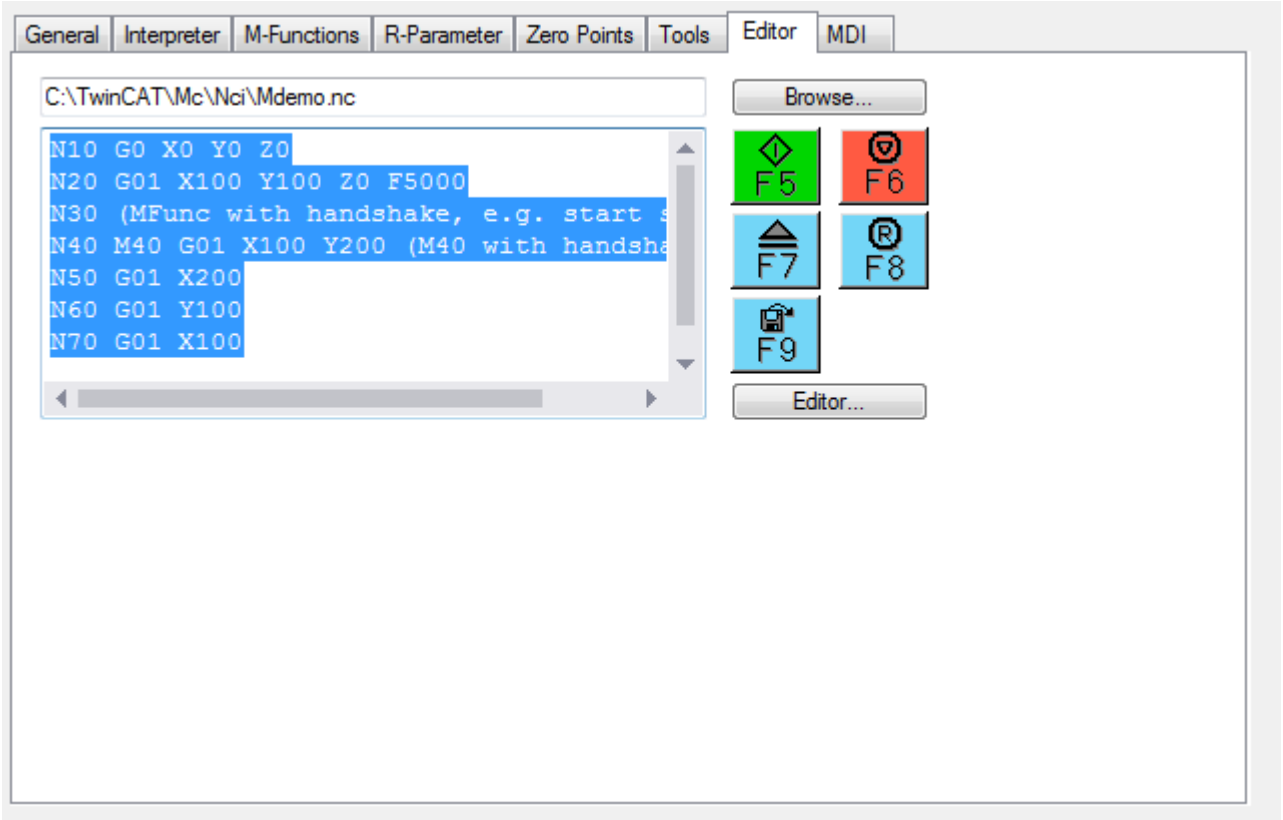
3.3.6 Karteireiter "Werkzeuge"



Auf der Eigenschaftsseite „Werkzeuge“ können Sie die Daten für die Werkzeugkorrektur editieren.

Eine nähere Beschreibung der Parameter ist in der Interpreterbeschreibung unter Werkzeugkorrekturen [▶ 184] zu finden.

3.3.7 Karteireiter "Editor“



Mit dem Editor können NC-Programme dargestellt und editiert werden.

- **Browse...**
  Öffnet einen Dialog, mit dem vorhandene NC-Programme ausgewählt und dargestellt werden können.

Remote-Verbindung: Lade NC-Datei vom Zielsystem

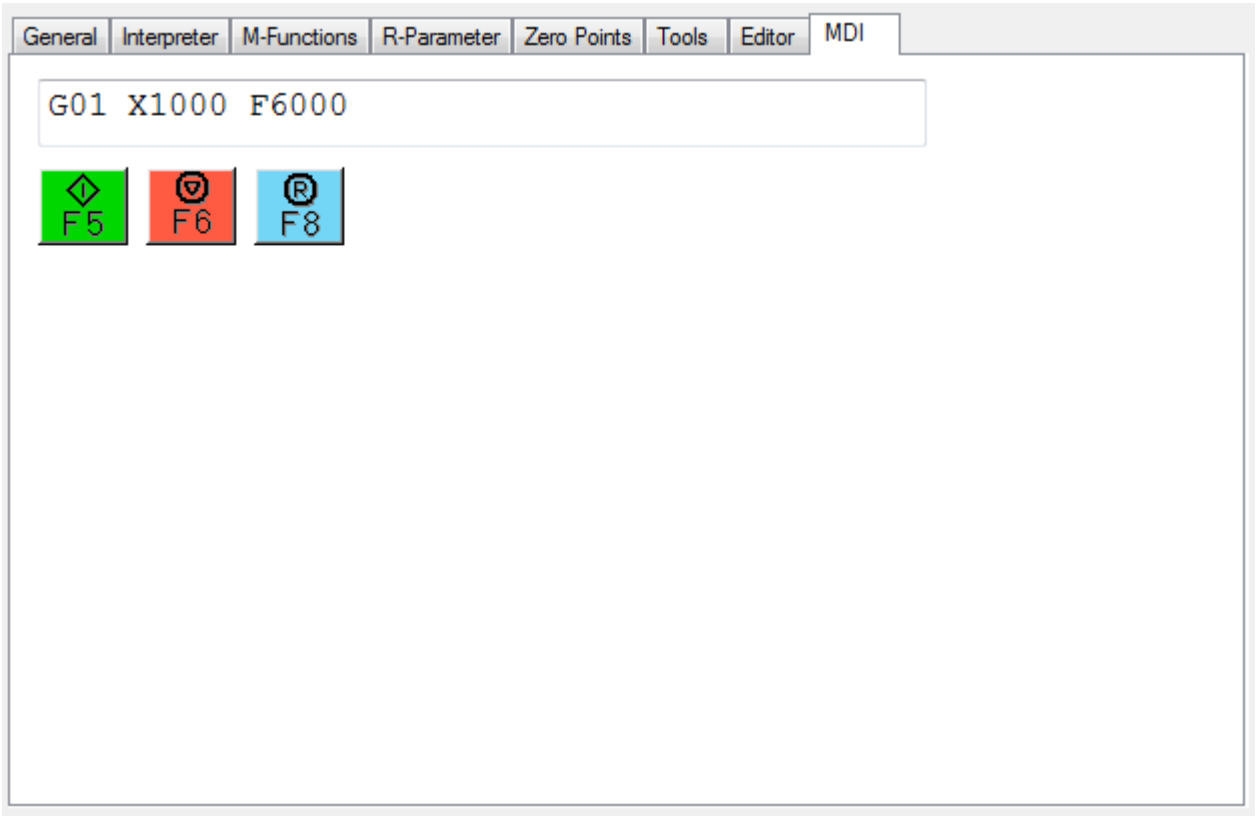
Wenn das Zielsystem über eine Remote Verbindung verbunden ist, dann muss die NC-Datei vom Zielsystem ausgewählt werden und kann nicht vom lokalen System geladen werden.

- **F5**
  Startet das aktuell geladene NC-Programm.

Das im Editor dargestellte NC-Programm muss nicht unbedingt das aktuell geladene Programm sein.

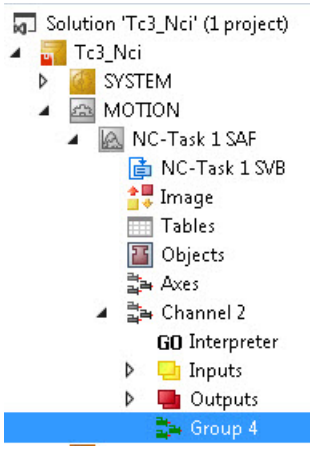
- **F6**
  Stoppt das aktuell laufende NC-Programm.
- **F7**
  Lädt das im Editor dargestellte NC-Programm.
- **F8**
  Führt einen Gruppen-Reset durch.
- **F9**
  Speichert das aktuell im Editor dargestellte NC-Programm unter dem gleichen Namen.
- **Editor...**
  Öffnet ein größeres Fenster, in dem das NC-Programm dargestellt wird.

3.3.8 Karteireiter "MDI"



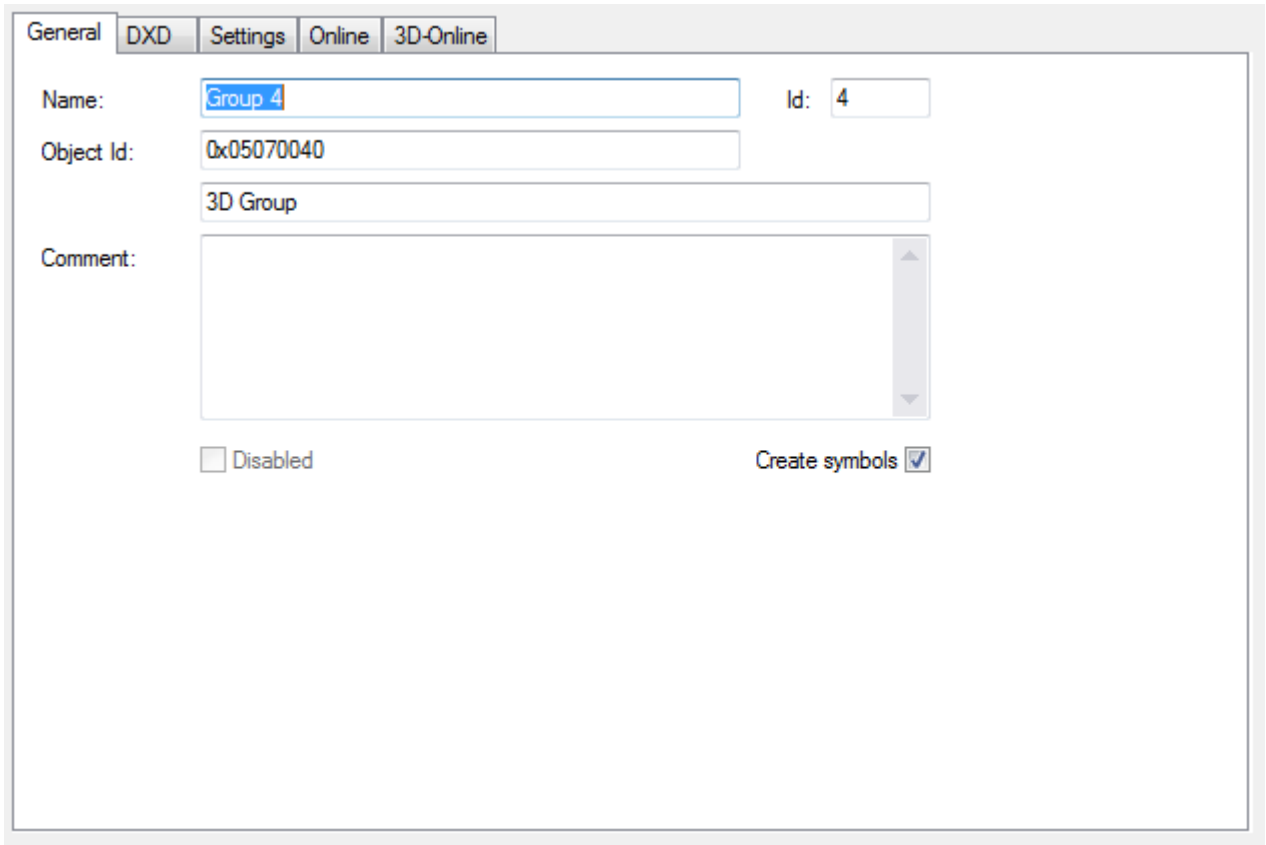
MDI steht für „Manual Data Interface“. Hiermit lassen sich aus der TwinCAT Engineering Umgebung (XAE) einzelne NC-Sätze direkt eingeben. Mit F5 und F6 wird die Abarbeitung gestartet bzw. gestoppt.

3.4 Gruppenelement



Allgemein [▶ 23]
DXD [▶ 24]
Einstellungen [▶ 28]
Online [▶ 29]
3D-Online [▶ 30]

3.4.1 Karteireiter "Allgemein"



Gruppen-ID

Auf der Seite „Allgemein“ kann die Gruppen-ID ermittelt werden. Diese wird für gruppenspezifische ADS Kommandos benötigt.

Symbole erzeugen

Um symbolisch auf Bahnvariablen etc. zugreifen zu können, muss an dieser Stelle die Generierung von Symbolen für die Gruppe angewählt werden.

3.4.2 Karteireiter "DXD"

General | DXD | Settings | Online | 3D-Online

	Parameter	Offline Value	Online Value	Type	Unit
	Curve Velocity Reduction Mode	'COULOMB'		E	
	Velocity Reduction Factor for C0-Transition	0.1		F	
	Velocity Reduction Factor for C1-Transition	1.0		F	
	Critical Angle for Segment Transition 'Low'	10.0		F	°
	Critical Angle for Segment Transition 'High'	75.0		F	°
	Minimum velocity at segment transitions	0.0		F	
	Global Soft Position Limits (for x,y,z-axes)	TRUE		B	
	Interpreter Override Type	Reduced		E	
	Enable calculation of the total remaining chord length	FALSE		B	
	Maximum number of transferred jobs per nc cycle [1 ... 20]	1		D	
	SAF cycle time divisor	1		D	
	User defined SAF table length [128 ... 1024]	128		D	

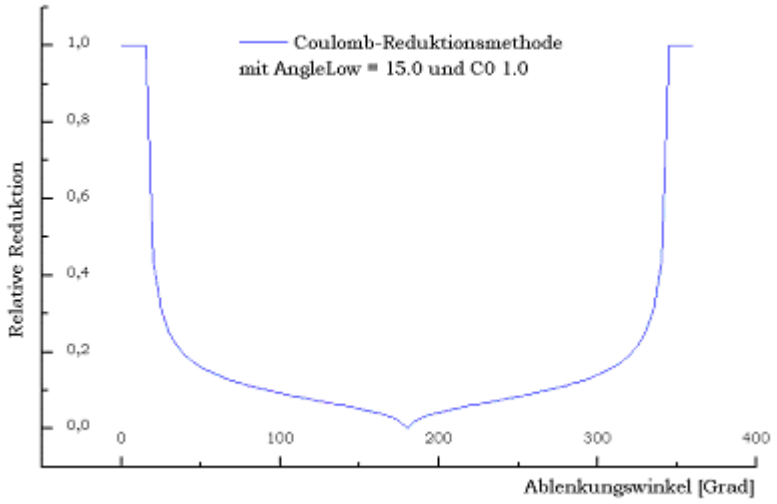
Auf der Eigenschaftsseite „DXD“ werden die NCI-Gruppenparameter beschrieben.

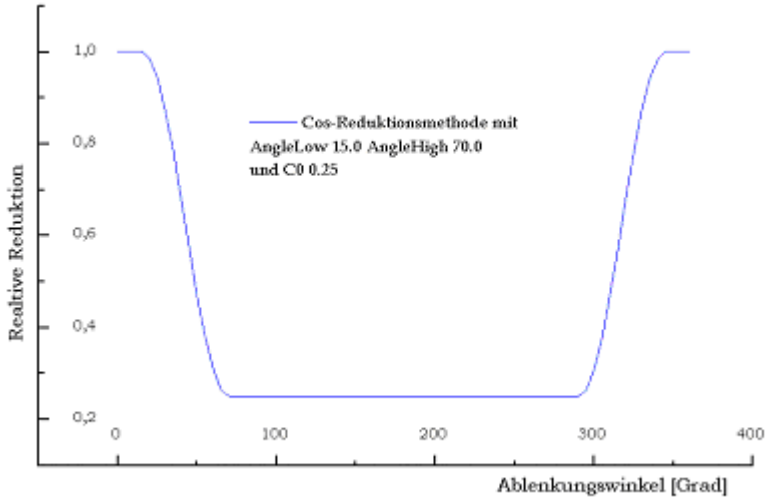
Kurvengeschwindigkeitsreduktionsmethode

Die Kurvengeschwindigkeitsreduktionsmethode wirkt ausschließlich für C0-Übergänge (Siehe Klassifikation der Segmentübergänge [▶ 330]).

Defines der Kurvengeschwindigkeitsreduktionsmethoden

```
0 Coulomb
1 Cosinus
2 VeloJump
3 DeviationAngle (not yet released)
```

Methode	Beschreibung
Coulomb	Die Coulomb-Reduktionsmethode ist ein dynamisches Verfahren in Analogie zur Coulombstreuung. Der Ablenkungswinkel φ im Übergangspunkt ist der Winkel zwischen der Tangente der Bahn am Ende des Segments S1 und der Tangente der Bahn am Anfang des Segments S2. Die Geschwindigkeit wird in Analogie zur Coulombstreuung proportional zur Geschwindigkeit im Unendlichen angesetzt $V_k \propto (\tan(0.5(\pi-\varphi)))^{1/2}$ und dann per C0-Faktor reduziert. $V_k \leftarrow C0\ V_k$. Bei Bewegungsumkehr (φ=180) wird damit auf jeden Fall auf V_k = C0 reduziert. Da die Reduktion jedoch bei kleinen Ablenkungswinkeln drastisch ist, gibt es einen Winkel $\varphi_{low} \in [0,180]$ ab dem die Reduktion voll einsetzt. Wollen Sie nicht reduzieren, dann setzen Sie φ_{low} = 180. Wollen Sie (bis auf φ = 0) vollständig reduzieren, dann setzen Sie C0 = 0.0 und φ_{low} = 0. 

Methode	Beschreibung
Cosinus	Die Cosinus-Reduktionsmethode ist ein rein geometrisches Verfahren. Hier gibt es • den C0-Faktor ∈ [0,1], • einen Winkel $\varphi_{low} \in [0,180]$, • einen Winkel $\varphi_{high} \in [0,180]$, sodas $\varphi_{low} < \varphi_{high}$ Reduktionsschema: • $\varphi < \varphi_{low}$: keine Reduktion: $V_k \leftarrow V_{k,}$ • $\varphi_{high} < \varphi$: Reduktion um den C0-Faktor: $V_k \leftarrow$ C0 V_k • $\varphi_{low} < \varphi < \varphi_{high}$: partielle Reduktion stetig interpolierend zwischen den Fällen 1 und 2, proportional zur cos-Funktion im Bereich [0,π/2]. Wollen Sie (bis auf φ = 0) vollständig reduzieren, dann setzen Sie C0 = 0.0, φ_{low}=0 und φ_{high} sehr klein aber ungleich 0 (z.B. 1.0E-10) 
VeloJump	Hierbei handelt es sich um ein geometrisches Verfahren zur Bestimmung der Segmentübergangsgeschwindigkeit bei einem C0-Übergang. Bei diesem Verfahren wird die Bahngeschwindigkeit ggf. so reduziert, dass der auftretende Geschwindigkeitssprung nicht grösser wird, als max. erlaubt. Dieser berechnet sich nach der Formel: VeloJump-Faktor * Zykluszeit * min(Beschleunigung; Verzögerung). Weitere Information [▶ 330]

Geschwindigkeitsreduktionsfaktor C0-Übergang

Reduktionsfaktor für C0-Übergänge. Die Wirkungsweise ist von der Reduktionsmethode abhängig.

C0 ∈ [0.0, 1]

Geschwindigkeitsreduktionsfaktor C1-Übergang

Zuerst wird *V_link* gleich dem Minimum der beiden Segmentsollgeschwindigkeiten gesetzt:
V_link = min(*V_in,V_out*).

In Abhängigkeit von den Geometrietypen *G_in* und *G_out* auf den zu verbindenden Segmenten und den Ebenenanwahlen auf *G_in* und *G_out* wird der geometrisch induzierte absolute Beschleunigungssprung *AccJump* im Segentübergang unter der Geschwindigkeit *V_link* berechnet.

Ist dieser größer als *C1* mal der für die Geometrien und Ebenen zulässigen Bahn-Beschleunigung / (absolute)Verzögerung *AccPathReduced*, dann wird die Geschwindigkeit *V_link* so reduziert, dass der sich ergebende Beschleunigungssprung gleich *AccPathReduced* ist.

Ist dieser Wert kleiner als *V_min*, dann hat *V_min* Priorität.

Hinweis Bei Wechsel der Dynamikparameter ändert sich automatisch die für die Geometrien und Ebenen zulässige Bahn-Beschleunigung und damit das Verhalten der Reduktion.

Reduktionsfaktor für *C1*-Übergänge: *C1 ≥ 0.0*

Kritischer Winkel Segementübergang 'Low'

Paramter für φ_{low} (Siehe: Kurvengeschwindigkeitsreduktionsmethode [▶ 24]).

Kritischer Winkel Segementübergang 'High'

Paramter für φ_{high} (Siehe: Kurvengeschwindigkeitsreduktionsmethode [▶ 24]).

Mindestgeschwindigkeit an Segmentübergängen

Jede NCI-Gruppe hat eine Mindestbahngeschwindigkeit *V_min≥ 0.0* die niemals unterschritten werden sollte. Vom Benutzer vorgegebene Ausnahmen sind: programmierter Halt am Segmentübergang, Bahnende und Override-Anforderungen die unter die Mindestgeschwindigkeit führen. Systembedingte Ausnahme ist eine Bewegungsumkehr.

Bei der Reduktionsmethode DEVIATIONANGLE gilt für den Ablenkungswinkel *φ ≥ φ_h*, dann wird die Mindestgeschwindigkeit nicht beachtet. *V_min* muss kleiner als die Bahnsollgeschwindigkeit (F-Wort) jedes Segments sein.

Die Mindestgeschwindigkeit kann jederzeit im NC-Programm auf einen neuen Wert *V_min≥ 0.0* in Einheiten *mm/sec* gesetzt werden.

Global Soft Position Limits (für x,y,z-Achsen)

Parameter zum Einschalten der Software-Endlagen der Bahn (Siehe: Parametrierung [▶ 332]).

Interpreter Override Typ

Parameter zur Auswahl des Overridetyps der Bahn (Siehe Bahnoverride (Interpreter-Overridetypen) [▶ 333]).

Enable calculation of the total remaining chord length

Aktiviert die Berechnung der verbleibenden Pfadlänge. Wenn die Berechnung der verbleibenden Pfadlänge aktiviert worden ist, dann kann sie anschließend per ADS ausgelesen werden. Siehe auch im Anhang: Anzeige der verbleibenden Bahnlänge [▶ 330].

Maximum number of transferred jobs per nc cycle [1 … 20]

Maximale Anzahl von zu übertragenden Kommandos pro NC Zyklus. Mit diesem Parameter ist es möglich, dass die SVB Task weiterhin langsamer läuft als die SAF Task und dennoch ausreichend viele Jobs übergeben werden, so dass die SAF-Tabelle mit Jobs nicht leer läuft.

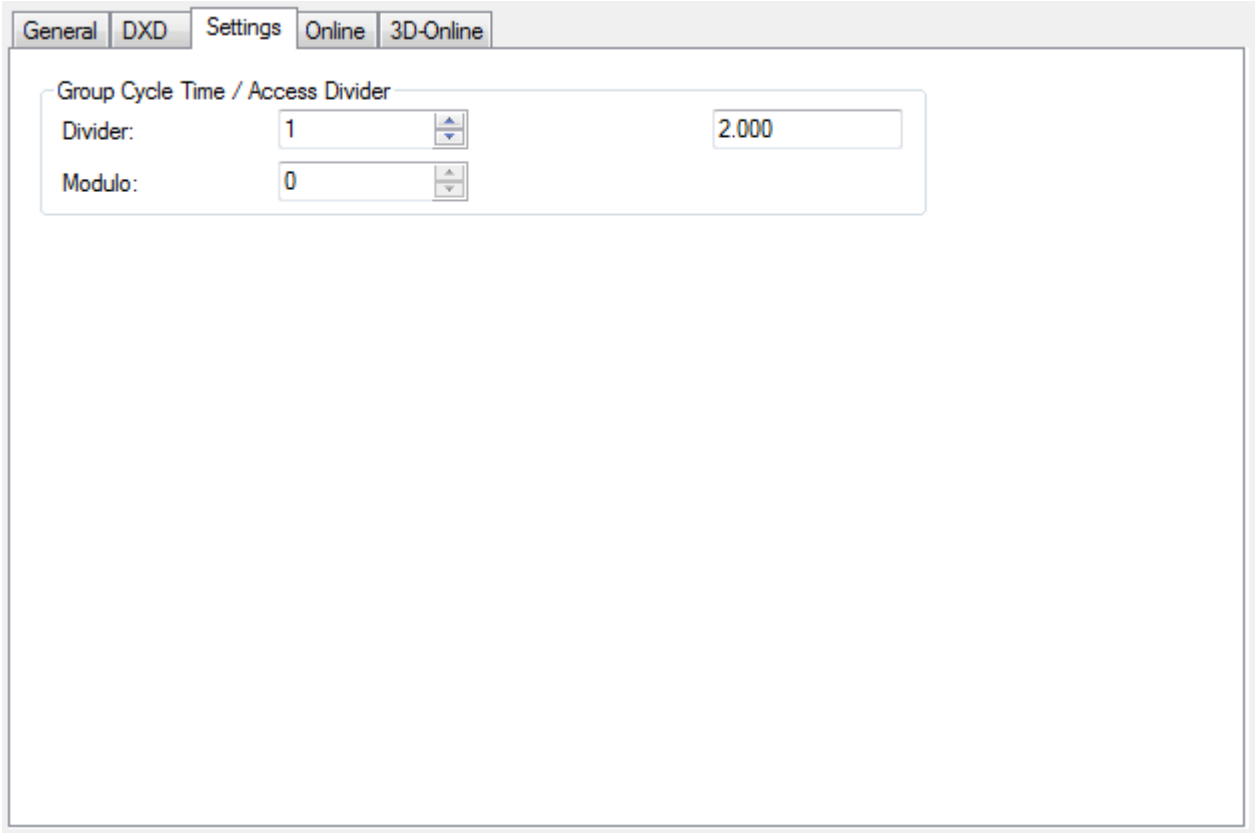
SAF cycle time divisor

Die Zykluszeituntersetzung sorgt dafür, dass die Sollwertberechnung in der SAF nicht mit der SAF-Zykluszeit, sondern mit einer durch den hier angegebenen Wert dividierten Zeit berechnet wird. Beim Verfahren mit sehr großer Dynamik kann es sinnvoll sein, den Parameter auf einen Wert größer 1 zu setzen um Diskretisierungsungenauigkeiten zu minimieren. Die Erhöhung des SAF cycle time divisor führt dazu, dass der Sollwertgenerator intern häufiger aufgerufen wird.

User defined SAF table length

Parameter, der die Größe der SAF-Tabelle und somit die maximale Anzahl der gepufferten SAF-Einträge (Look-Ahead) definiert. Werden in einem NC-Programm viele sehr kurze Segmente nacheinander verfahren, so kann durch Vergrößerung dieses Wertes eine ungewollte Geschwindigkeitsreduktion am Segmentübergang vermieden werden.

3.4.3 Karteireiter „Einstellungen“

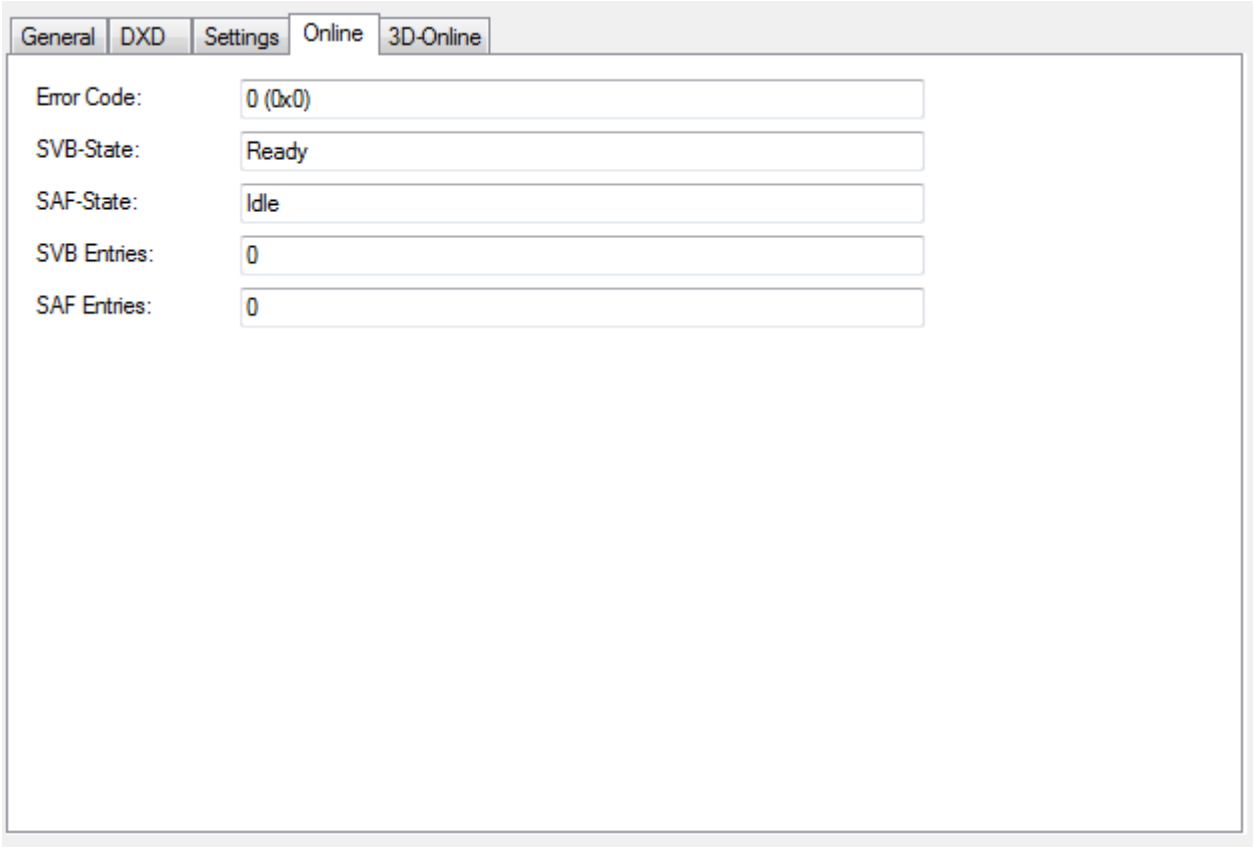


Unter dem Karteireiter „Einstellungen“ können Sie die Zykluszeit für die Interpolation einstellen. Die hier eingestellte Zykluszeit ist hierbei ein Vielfaches der Zykluszeit der SAF-Task.

Verwendung der Zykluszeit im Karteireiter „Einstellungen“

Die Einstellung der Zykluszeit können Sie nutzen, wenn Sie eine von der SAF-Task verschiedene Zykluszeit für die Interpolation wählen müssen. Im Allgemeinen sollte zum Einstellen der Zykluszeit die Zykluszeit der SAF-Task angepasst werden.

3.4.4 Karteireiter "Online"



Fehlercode

Hier wird der aktuelle Fehlercode des Kanals dargestellt. Der Wert ist der gleiche, wie er im Online-Fenster des Interpreters unter 'Kanalstatus [▶ 14]' angezeigt wird.

SVB-Status

Der SVB-Status stellt den aktuellen Zustand der SVB (**S**atz**v**or**b**ereitung) dar. Mögliche SVB-Zustände sind:

```
ERROR
IDLE
READY
START
DRIVEOUT
CALIBRATE
MFUNC
SYNCREC
DELAY
MFUNCWAIT
SPINDLEWAIT
```

Für gewöhnlich ist es nicht erforderlich, dass der SVB-Status von der SPS ausgewertet wird.

SAF-Status

Der SAF-Status stellt den aktuellen Zustand der SAF (**S**atz**a**us**f**ührung) dar. Mögliche SAF-Zustände sind:

```
ERROR
IDLE
CONTROL
RUN
RUN_DRIVEOUT
WAIT
```

Für gewöhnlich ist es nicht erforderlich, dass der SAF-Status von der SPS ausgewertet wird.

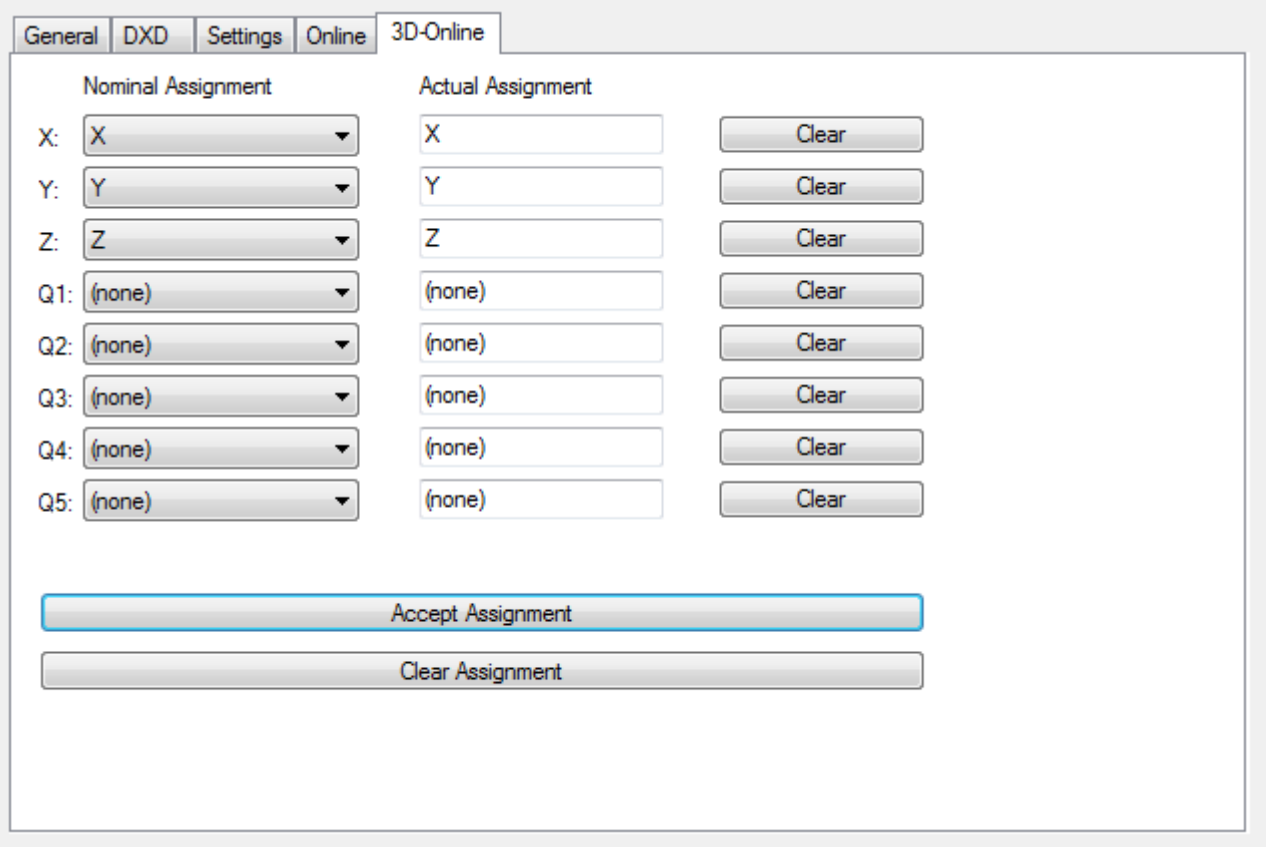
SVB-Einträge

Anzahl der momentanen SVB-Einträge

SAF-Einträge

Anzahl der momentanen SAF-Einträge

3.4.5 Karteireiter "3D-Online"



Soll-Belegung

An dieser Stelle wird die **Interpolationsgruppe** gebildet. D.h. die PTP Achsen, die den Bahnachsen X, Y und Z zugewiesen werden, können anschließend interpolierend verfahren werden.

Mit Hilfe der Auswahllisten können Sie für die Bahnachsen X, Y und Z beliebige PTP Achsen auswählen. Wenn Sie den Button 'Übernehmen' drücken, wird die 3D-Gruppe gebildet.

Ein vergleichbarer SPS Baustein steht in der PLC Library: Tc2_NCI [▶ 203] zur Verfügung. (Siehe CfgBuildExt3DGroup [▶ 204])

Ist-Belegung

Hier wird die aktuelle Belegung der Bahnachsen angezeigt. Wenn Sie einzelne Achsen aus der 3D-Gruppe entfernen möchten, können Sie dies über 'Löschen' machen.

Belegung komplett löschen

Löst die komplette 3D-Gruppe auf. Auch hierfür steht ein SPS Baustein in der PLC Library: Tc2_NCI [▶ 203] zur Verfügung. (Siehe CfgReconfigGroup [▶ 207])

4 GST-Referenzhandbuch

4.1 Allgemeine Hinweise

Alle GST-Beispiele in dieser Dokumentation gehen von den folgenden Annahmen aus:

- Zu Beginn befindet sich das Werkzeug unter X0, Y0, Z0.
- Alle Zustandsvariablen des Interpreters werden auf ihre Standardwerte gesetzt, mit der Ausnahme, dass die Geschwindigkeit auf einen Wert ungleich Null gesetzt wird.

4.2 Präprozessor

Include-Direktive

```
#include "<path>"
#include < <path> >
```

Die Direktive #include fügt den Inhalt einer anderen Datei ein. Die inkludierte Datei wird durch ihren Pfad referenziert. In der Regel wird er verwendet, um häufig verwendeten Code wie z. B. Bibliotheken zu "importieren". Das Verhalten ist dem des C-Präprozessors ähnlich.

Beispiel:

In dem folgenden Beispiel inkludiert die Datei a.nc die Datei b.nc ein. Bei der Ausführung von a.nc ersetzt der Interpreter intern die Include-Zeile durch den Text von b.nc. Daher hat die Ausführung des Programms a.nc die gleiche Wirkung wie die Ausführung des Programms c.nc.

FILE a.nc:

```
G01 X0 Y0 F6000
#include "b.nc"
G01 X0 Y100
```

FILE b.nc:

```
G01 Z-2
G01 X100
G01 Z2
```

FILE c.nc:

```
G01 X0 Y0 F6000
G01 Z-2
G01 X100
G01 Z2
G01 X0 Y100
```

- Wenn path absolut ist, wird dieser direkt verwendet, um die inkludierte Datei zu finden. Ein absoluter Pfad muss zwischen Anführungszeichen gesetzt werden.
- Wenn path relativ ist und zwischen Anführungszeichen steht, wird dieser an das Verzeichnis der inkludierenden Datei angehängt, um den Pfad der inkludierten Datei zu bilden.
- Wird path in spitze Klammern dargestellt, gilt dieser als relativ zu den Pfaden in der Liste searchpath. Der erste Eintrag in dieser Liste, der zu einer bestehenden Datei führt, wird für die Einbindung verwendet. Die Liste searchpath wird von der Interpreterumgebung des Interpreters geliefert.

Beispiel:

Das folgende Beispiel geht davon aus, dass die searchpath auf die Verzeichnisse c:\jjj und c:\kkk eingestellt ist. Die Datei aaa.nc besteht aus einer Reihe von #include-Direktiven, die im Folgenden erläutert werden.

- Die Datei `bbb.nc` wird über einen absoluten Pfad inkludiert. Daher ist ihr Ort unabhängig vom Ort von `aaa.nc`. Absolute Referenzierung ist nützlich für Dateien, die sich immer an einem festen Ort im Dateisystem befinden.
- Die Datei `ccc.nc` ist relativ referenziert. Sie muss sich im Verzeichnis von `aaa.nc` (der inkludierenden Datei) befinden, also `c:\mmm\`.
- Die Datei `ddd.nc` ist ebenfalls relativ referenziert. Sie wird voraussichtlich unter `c:\mmm\ooo\ddd.nc` zu finden sein.
- Die relative Referenz von `eee.nc` verwendet die Sequenz `'..'`, die sich auf das übergeordnete Verzeichnis bezieht. Daher wird die Datei `eee.nc` in `c:\ppp\qqq\eee.nc` erwartet.
- Der relative Pfad von `fff.nc` wird in spitzen Klammern dargestellt. Daher werden die Verzeichnisse im `searchpath` berücksichtigt und nicht das Verzeichnis von `aaa.nc`. Die Datei wird in `c:\jjj\fff.nc` oder `c:\kkk\fff.nc` erwartet. Der erste Pfad, der zu einer vorhandenen Datei führt, wird berücksichtigt. Wenn sich in keinem Verzeichnis vom `searchpath` eine Datei `fff.nc` befindet, wird ein Fehler ausgegeben.
- Schließlich wird die Datei `ggg.nc` in `c:\rrr\ggg.nc` erwartet. Beide Einträge im `searchpath` führen zu diesem Ort.

FILE c:\mmm\aaa.nc:

```
#include "c:\nnn\bbb.nc"
#include "ccc.nc"
#include "ooo\ddd.nc"
#include "..\ppp\qqq\eee.nc"
#include <fff.nc>
#include <../rrr/ggg.nc>
```

- Jede include-Direktive muss in einer eigenen Zeile angegeben werden. Dann wird diese gesamte Zeile durch den Inhalt der inkludierten Datei ersetzt. Ein zusätzliches "Newline"-Zeichen wird angehängt.
- Die Include-Direktive kann mehrfach an beliebigen Stellen der inkludierenden Datei verwendet werden.
- Wenn eine inkludierte Datei nicht existiert, wird ein Fehler ausgegeben.
- Wenn die Include-Direktive nicht an der ersten Position einer Zeile steht, wird ein Fehler ausgegeben.
- Include-Direktiven in inkludierten Dateien werden ebenfalls ersetzt.
- Eine Endlosschleife aufgrund rekursiver Inklusionen (z. B. `A` inkludiert `B`, `B` inkludiert `C` und `C` inkludiert `A`) wird erkannt und als Fehler gemeldet.
- Dieselbe Datei kann mehrfach inkludiert werden.

Die mehrfache Inklusion einer Datei ist in der Regel eine schlechte Praxis. Vor allem, wenn diese Funktion dazu missbraucht wird, Code auszuklammern. Stattdessen sollte eine Funktion bevorzugt werden, um Code zu definieren, der mehrfach wiederverwendet wird (siehe Abschnitt Benutzerdefinierte Funktionen [▶ 61]).

Beispiel:

In dem folgenden Beispiel inkludiert die Datei `a.nc` die Datei `b.nc` zweimal. Die zweite Inklusion wird immer, unabhängig von der einschließenden Bedingung, durch den Ausdruck `IF-THEN` erweitert. Die inkludierte Datei `b.nc` inkludiert selbst die Datei `c.nc`.

FILE a.nc:

```
G01 X100 F6000
#include "b.nc"
G01 Y100
! IF stVariable=47 THEN
#include "b.nc"
! END_IF;
```

FILE b.nc:

```
#include "c.nc"
G01 X0 Y0
```

FILE c.nc:

```
G01 Z0
```

Beispiel:

Die Datei `x.nc` demonstriert eine Reihe von ungültigen Include-Direktiven. Die ersten drei Zeilen verstoßen gegen die Regel, dass jede include-Direktive in einer eigenen Zeile stehen muss. In den Zeilen 4 und 5 ist der Dateiname nicht richtig in Anführungszeichen oder spitze Klammern eingeschlossen. In Zeile 6 ist eine nicht existierende Datei enthalten. Zeile 7 verstößt gegen die Regel, dass die include-Direktive immer an der ersten Stelle einer Zeile stehen muss. Zeile 8 inkludiert die Datei `y.nc`, die wiederum die Datei `x.nc` inkludiert. Diese Schleife wird als Fehler ausgegeben.

FILE x.nc:

```
#include "a.nc" G01 X100
! #include "a.nc"
#include "a.nc"    #include "b.nc"
#include a.nc
#include "a.nc>
#include "non_existing_file.nc"
    #include "a.nc"
#include y.nc
```

FILE y.nc:

```
#include "x.nc"
```

4.3 Kombination von G-Code und ST

Ein GST-Programm

```
<g-code>
<g-code>
! <st-code>
<g-code>
<g-code>
{
<st-code>
<st-code>
! <g-code>
<st-code>
<st-code>
}
<g-code>
<g-code>
```

Eine GST-Datei besteht aus Sequenzen von `G`-Code und Sequenzen von `ST`-Code, die wie oben gezeigt verschachtelt werden können. Jedes Programm startet im Modus `G`-code. Der Modus kann mit einem Ausrufezeichen ('!') für eine Zeile auf `ST` umgeschaltet werden. Der `ST`-Modus endet automatisch am Ende der Zeile.

Alternativ kann ein Satz von `ST`-Code mit geschweiften Klammern ('{'...'}') definiert werden. Diese Notation ist praktischer, um eine lange Sequenz von `ST`-Code in einem GST-Programm zu definieren. Innerhalb des `ST`-Satzes kann der `G`-Codemodus für eine Zeile mit dem Ausrufezeichen aktiviert werden. Dabei endet der `G`-Codemodus automatisch am Ende der Zeile.

G-Code Satz

`<address><value> <address>=<G-Expression> <address>{<ST-Expression>}`

Eine Zeile von `G`-Code wird als **Satz** bezeichnet. Sie besteht aus einer Folge von **Wörtern**. Ein Wort ist eine Kombination aus einer **Adresse** (z.B. `G` oder `X`) und einem **Wert**. Ein Wert kann durch ein Literal (z. B. `2.54`), durch einen `G`-Ausdruck (z. B. `2*foo+1`) oder durch einen `ST`-Ausdruck (z. B. `sin(foo**2)-1`) definiert werden.

G-Code Ausdruck

```
<address>=a+b-c*d/e
```

Das Ergebnis des Ausdrucks wird als Wert des Wortes verwendet. Die vier Grundrechenarten ('+', '–', '*', '/') können in einem G-Ausdruck verwendet werden. Sie werden wie erwartet ausgewertet, d.h. alle Operationen sind links-assoziativ und '*', '/' haben einen höheren Vorrang als '+', '–'. Variablen, die in ST deklariert wurden, können auch in einem G-Ausdruck (in Bezug auf ihren Geltungsbereich) verwendet werden.

Alle Berechnungen werden mit dem Typ LReal (64-Bit-Fließkomma gemäß IEEE 754) durchgeführt. Der Wert einer ST-Variablen wird gemäß den Konvertierungsregeln von ST implizit in den Typ LReal konvertiert. Wenn ein Typ (z.B. STRING) nicht konvertiert werden kann, wird ein Fehler ausgegeben.

EINSCHRÄNKUNG:

ST-Variablen, die eine Zahl in ihrem Namen enthalten (z. B. x0), können nicht in einem G-Ausdruck verwendet werden, um Verwechslungen mit einem G-Code wie X0 zu vermeiden. Diese Einschränkung gilt nicht für ST-Ausdrücke.

EINSCHRÄNKUNG:

Array-Variablen, Struktur-Variablen und Objekte können nicht in einem G-Ausdruck verwendet werden. Diese Einschränkung gilt nicht für ST-Ausdrücke.

EINSCHRÄNKUNG:

Klammern sind in einem G-Ausdruck nicht erlaubt, da sie zur Kennzeichnung von Kommentaren in G-Code verwendet werden. Aus demselben Grund sind auch keine Funktionsaufrufe möglich. Diese Einschränkungen gelten nicht für ST-Ausdrücke.

Eingebetteter ST-Ausdruck

```
<address>{<ST-Expression>}
```

Das Ergebnis des ST-Ausdrucks wird als Wert des Wortes verwendet. Es muss in LReal umgewandelt werden können. Im Grunde ist ein ST-Ausdruck ST-Code, der auf der rechten Seite einer Zuweisung platziert werden kann. Ein anderer ST-Code (z.B. eine ST-Anweisung) ist nicht zulässig. Umfangreiche Berechnungen können jedoch in einer ST-Funktion gekapselt werden, die dann in dem ST-Ausdruck aufgerufen wird.

Ein ST-Ausdruck sollte keine Nebeneffekte haben, da die Auswertungsreihenfolge von ST-Ausdrücken im Allgemeinen nicht definiert ist und sich in Zukunft ändern kann. Außerdem ist diese Art der Programmierung, bei der Seiteneffekte eingesetzt werden, ein schlechter Programmierstil. Zum Beispiel sollte ein ST-Ausdruck keine Funktion aufrufen, die G-Code enthält.

Beispiel:

- Das folgende GST-Programm beginnt mit einer Zeile des G-Codes, der das Werkzeug im Eilgang zum Ursprung bewegt.
- Auf diese Zeile folgt eine Zeile mit ST-Code, in der die Variable 'i' deklariert wird. Der ST-Modus wird durch das vorangestellte Ausrufezeichen ('!') aktiviert. Nach dieser Zeile wird der G-Code-Modus wieder aufgenommen.
- Der G-Code in Zeile 3 bewegt das Werkzeug nach unten.
- Die Zeilen 4 bis 8 definieren einen Satz von ST-Code, der eine FOR-Schleife enthält. Der Code in diesem Satz wird als ST-Code interpretiert, mit Ausnahme der G-Codezeile in Zeile 6. Diese Zeile des G-Codes verwendet einen G-Ausdruck, um die X-Achse auf 10*i zu setzen. Der Wert der Y-Achse wird durch einen ST-Ausdruck definiert, der in geschweifte Klammern eingeschlossen ist. Dieser Ausdruck wird zu 0 ausgewertet, wenn 'i' gerade ist, und ansonsten zu 10.
- Die programmierte Bahn des Programms ist in Abbildung "Beispiel Ausdrücke" dargestellt.

```
G00 X0 Y0 Z0
! VAR   i : INT;   END_VAR
G01 Z-1 F6000
{
```

```
FOR i := 1 TO 5 DO
!G01 X=i*10 Y{ (i MOD 2) *10 }
END_FOR;
}
```

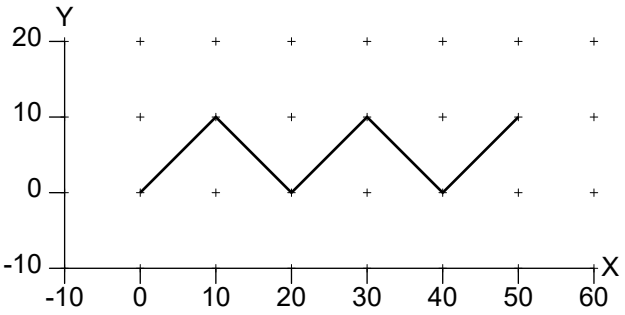


Abbildung "Beispiel Ausdrücke".

Unterdrückung von G-Code-Sätzen

/<n> <G-Code block>

Die Ausführung eines G-Code Satzes kann bedingt unterdrückt werden. Wenn '/<n>' vorangestellt ist und das n-te Bit in einer internen Disable-Maske gesetzt ist, wird der Satz unterdrückt (nicht ausgeführt). Die Disable-Maske kann von der SPS und über die ST-Funktion disableMaskSet eingestellt werden. Wenn n nicht angegeben wird, hat es standardmäßig den Wert 0. [Siehe Abschnitt Unterdrückung von G-Code-Sätzen [▶ 84].]

4.4 G-Code (DIN 66025)

4.4.1 Werkzeugradiuskorrektur (D, G40, G41, G42)

D

D<v>

Werkzeug v auswählen. Das neue Werkzeug gilt für seinen eigenen Satz und alle folgenden Sätze, bis ein neues Werkzeug ausgewählt wird. Das Werkzeug 0 ist etwas Besonderes. Durch seine Auswahl wird jede Werkzeugkorrektur deaktiviert. Das Werkzeug 0 kann als ein Werkzeug betrachtet werden, bei dem alle Werkzeugparameter auf Null gesetzt sind. Es ist standardmäßig ausgewählt.

Beispiel:

Im folgenden Beispiel wird das Werkzeug 1 mit einem Y-Offset von 10 und das Werkzeug 2 mit einem Y-Offset von 20 definiert. Satz N10 und Satz N50 verwenden das Werkzeug 0. Das Werkzeug 1 gilt für den Satz N20 und für den Satz N30. Im Satz N40 ist das Werkzeug 2 aktiv. Abbildung "BeispielD" zeigt die resultierende programmierte Bahn (gepunktete Linie) und die resultierende Bahn des Werkzeugmittelpunkts (durchgezogene Linie).

```
!toolSet(index:=1, nr:=1, offsetY:=10);
!toolSet(index:=2, nr:=2, offsetY:=20);
N10 G01 X10 Y0 F6000
N20 G01 X20 Y0 D1
N30 G01 X30 Y0
N40 G01 X40 Y0 D2
N50 G01 X50 Y0 D0
M02
```

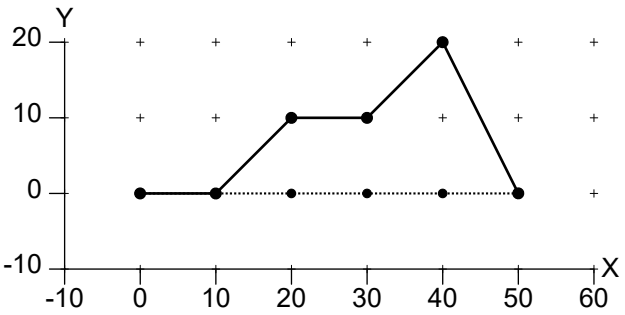


Abbildung "BeispielD".

G40

Befehl	G40 (Standardeinstellung)
Aufhebung	G41 oder G42

Deaktivieren Sie die Werkzeugradiuskorrektur (TRC).

G41

Befehl	G41
Aufhebung	G40 oder G42

Aktivieren Sie die Werkzeugradiuskorrektur (TRC). Nach der Aktivierung wird die programmierte Bahn um den Radius des aktuell gewählten Werkzeugs nach links verschoben. (Siehe D.)

Bei der Aktivierung muss ein Werkzeug mit einem Index ungleich Null ausgewählt werden.

Beispiel:

Das folgende Beispiel zeigt die Aktivierung und Deaktivierung der Werkzeugradiuskorrektur. Die programmierte Bahn (gepunktete Linie) und die kompensierte Bahn (durchgezogene/gestrichelte Linie) sind in der Abbildung "BeispielG40G41" dargestellt.

- Die erste Zeile des GST-Programms setzt den Parameter `offset` auf `5 mm`. Daher werden die benachbarten Segmente einer Lücke um `5 mm` erweitert. Die verbleibende Lücke wird durch einen Kreisbogen geschlossen.
- Die zweite Zeile definiert das An- und Abfahrverhalten, um einen Kreisbogen mit einem Radius von `5 mm` und einem Winkel von `90` Grad zu verwenden.
- Die dritte Zeile definiert das Werkzeug `1` mit einem Radius von `10`.
- Der Satz `N10` beschreibt eine lineare Bewegung auf `[10,0,0]`.
- Der nächste Satz `N20` wählt das Werkzeug `1` aus und aktiviert die Werkzeugradiuskorrektur, wobei `D1` vor der Bearbeitung von `G40` und `G40` vor der Bearbeitung von `X20` wirksam wird. Daher unterliegt das Ende des Segments `N20` der TRC (Werkzeugradiuskorrektur). Die lineare Bewegung vom Ende des Segments `N10` zum Ende des Segments `N20` in der programmierten Bahn wird durch ein Annäherungssegment (gestrichelte Linie) vom Ende von `N10` zum Ende von `N20'` in der kompensierten Bahn ersetzt.
- In den nächsten drei Zeilen wird eine lineare Bewegung entlang `N30`, `N40` und `N50` programmiert. Da das Segment `N40` zu einer Kollision führen würde, wird es aus der kompensierten Bahn entfernt.
- In der nächsten Zeile wird ein Kreisbogen entlang `N60` programmiert. Die Lücke zwischen dem Ende von `N50'` und dem Beginn von `N60'` wird wie oben beschrieben geschlossen.

- Die Linie entlang N70 ist das letzte Segment, das der TRC (Werkzeugradiuskorrektur) unterliegt, da seine Deaktivierung vor dem Ende von N80 aktiv wird. Die Linie entlang N80 wird durch das Abfahrtssegment N80' ersetzt, ähnlich wie das Anfahrtssegment.

```
!trcOffsetSet(offset:=5);
!trcApproachDepartSet(approachRadius:=5, approachAngle:=90, departRadius:=5, departAngle:=90);
!toolSet(index:=1, tooltype:=tooltypeMill, radius:=10);
N10 G01 X10 F6000
N20 X20 G41 D1
N30 X35
N40 X40
N50 Y20
N60 G02 X50 Y10 U10
N70 G01 X70
N80 X80 Y0 G40
N90 X90
M02
```

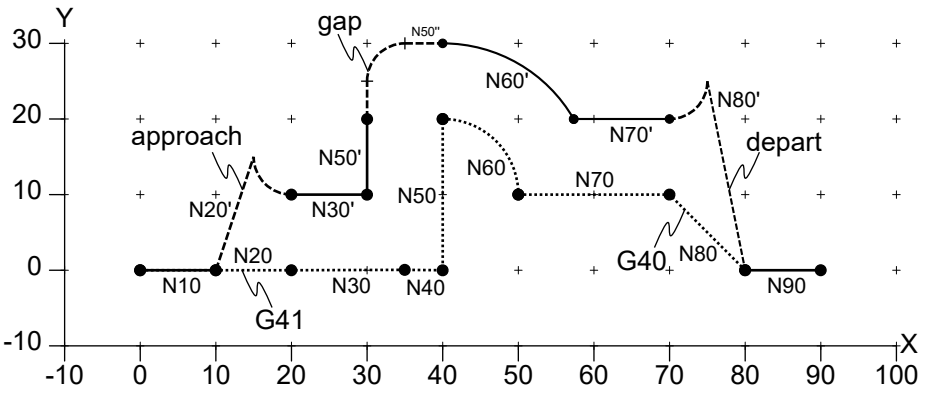


Abbildung "BeispielG40G41".

G42

Befehl	G42
Aufhebung	G40 oder G41

Diese Funktion ist die gleiche wie G41, mit dem Unterschied, dass die Bahn nach rechts verschoben wird. Siehe G41 für Einzelheiten.

4.4.2 Kommentare

DIN 66025 Kommentar

<g-code> (<comment>) <g-code>

Text, der in runde Klammern eingeschlossen ist, wird im G-Code (nach DIN 66025) als Kommentar behandelt. Der Kommentar darf keine weiteren Klammern enthalten. Ein Kommentar innerhalb runder Klammern kann sich über mehrere Sätze oder Zeilen erstrecken und kann daher auch einen Zeilenumbruch überspringen.

Beispiel:

Das folgende Beispiel zeigt die Notation von Kommentaren in G-Code.

```
N10 G01 X0 Y-10 F3000
N20 G01 (activate linear interpolation) X10 (set X-coordinate to
10) Y0 F6000
(the next block results in a semicircle with center point
X10 Y10)
N30 G02 (activate clockwise interpolation) Y20 U10 (radius is 10)
M02
```

Zeilenkommentar

```
<g-code> // <comment>
```

Text zwischen '//' und dem Ende der Zeile wird in G-Code als Kommentar behandelt.

Beispiel:

Das folgende Beispiel zeigt die Notation von Zeilenkommentaren in G-Code.

```
N10 G01 X10 F6000 // perform a linear movement to X10 Y0
// the next block results in a semicircle with center point X10 Y10
N20 G02 Y20 U10
M02
```

4.4.3 Ausführungsreihenfolge

Ein Satz (eine Zeile des G-Codes) besteht aus einer Folge von Wörtern. Die programmierte Reihenfolge der Wörter wird vom GST-Interpreter nicht berücksichtigt. Stattdessen wird die folgende Ausführungsreihenfolge befolgt, die aus 7 aufeinanderfolgenden und voneinander abhängigen Schritten besteht.

1. Referenzsystem	N*	Satznummer einstellen.
	G17..G19	Auswahl einer Arbeitsebene.
	G70, G71, G700, G710	Auswahl einer Einheit.
	G90, G91	Auswahl zwischen absoluter und inkrementeller Programmierung.
	D*, P*	Auswahl eines Werkzeugs und dessen Ausrichtung.
2. Konfiguration	G40..G42	(De-)Aktivierung der Werkzeugradiuskorrektur.
	G53..G59	Auswahl und Programmierung der Nullpunktverschiebung.
	F*	Sollgeschwindigkeit.
3. M-Funktion Pre	M*	M-Funktionen, die als "vorher" konfiguriert sind.
4. Parameter zur SPS	H*, S*, T*	
5. Bewegung	Q*, G00..G03	Bewegung bis zu einem gewissen Punkt.
	G09, G60	Aktivierung des Genauhalts.
6. Warten	G04	Eine bestimmte Dauer warten.

7. M-Funktion Post	M*	M-Funktionen, die als "nach" konfiguriert sind.

Im ersten Schritt wird das Referenzsystem eingerichtet. Im zweiten Schritt werden die folgenden Bewegungen konfiguriert. Beachten Sie, dass der zweite Schritt vom ersten abhängen kann. Z.B. berücksichtigt die programmierte Geschwindigkeit (F) eine Geschwindigkeitseinheit (G700), die im gleichen Satz programmiert ist. Schritt drei und die folgenden Schritte führen Aktionen wie eine Bewegung aus.

4.4.4 Gegenseitig exklusive G-Codes

Bestimmte Kombinationen von G-Codes dürfen nicht im selben Satz (Zeile von G-Code) programmiert werden. Solche widersprüchlichen G-Codes setzen typischerweise Zustandsvariablen auf widersprüchliche Werte (z. B. setzen Sie die Längeneinheit auf mm und auf inch). Es gibt auch Kombinationen, die dieselben Parameter verwenden und daher nicht im selben Satz programmiert werden dürfen (z. B. G58 und G59). Nachstehend finden Sie eine Liste der Gruppen von G-Codes. G-Codes, die zur gleichen Gruppe gehören, stehen im Konflikt.

- G00, G01, G02, G03, G04, G58, G59
  Interpolationen und programmierte Nullpunktverschiebung.
- G70, G71, G700, G710
  Einheit für Länge und Geschwindigkeit einstellen.
- G90, G91
  Absolute/relative Programmierung einstellen.
- G53, G54, G55, G56, G57
  Nullpunktverschiebung deaktivieren/auswählen.
- G40, G41, G42
  Werkzeugradiuskorrektur deaktivieren/aktivieren.
- G17, G18, G19
  Arbeitsebene auswählen.

4.4.5 Eilgang (G00)

Befehl	G0 bzw. G00
Aufhebung	G01 [▶ 40], G02 oder G03 [▶ 40]

Interpolationsmodus auf "schnell, linear" einstellen. Der Interpolationsmodus gilt für diesen Satz und allen folgenden Sätze, bis er durch G01, G02 oder G03 zurückgesetzt wird. G00 ist der Standard-Interpolationsmodus.

Wenn G00 aktiv ist, führt die Programmierung eines Punktes (siehe X) zu einem linearen Geometriesegment, das mit Maximalgeschwindigkeit bearbeitet wird. Die programmierte Geschwindigkeit wird nicht berücksichtigt. G00 wird normalerweise zur Positionierung des Werkzeugs verwendet. Für die Bearbeitung sollte G01 verwendet werden, das die programmierte Geschwindigkeit berücksichtigt.

G01, G02, G03, G04, G58 und G59 schließen sich gegenseitig aus. Sie dürfen nicht in einem gemeinsamen Satz programmiert werden.

Beispiel:

Die sich daraus ergebende Bahn des folgenden Beispiels ist in Abbildung "Beispiel G00" dargestellt. Der erste Satz N10 fährt das Werkzeug schnell auf die Positionen X20, Y10, Z30. Das resultierende Geometriesegment ist eine Linie im Raum. Die Ausrichtung bleibt unverändert. Der zweite Satz N20 führt eine schnelle Bewegung zu X50, Y10, Z30 durch. Es ist nicht notwendig, in dieser Zeile G00 anzugeben, da die Interpolation modal ist.

```
N10 G00 X20 Y10 Z30
N20 X50
M02
```

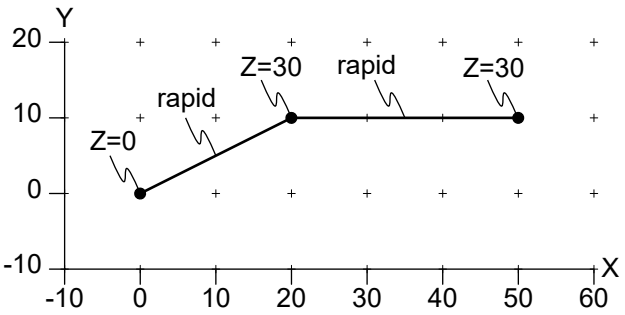


Abbildung "BeispielG00".

4.4.6 Linearinterpolation (G01)

Befehl	G1 bzw. G01
Aufhebung	G00, G02 oder G03

Setzen Sie den Interpolationsmodus auf "linear". Dieser Interpolationsmodus ist wie `G00`, mit dem Unterschied, dass die Bahn mit der programmierten Geschwindigkeit bearbeitet wird. (Siehe `F`.) Der Interpolationsmodus gilt für diesen Satz und alle folgenden Sätze, bis er durch `G00`, `G02` oder `G03` zurückgesetzt wird.

```
N20 G01 X100.1 Y200 F6000
N30 X150
M02
```

4.4.7 Kreisinterpolation (G02, G03, IJK, U)

G02 Kreisinterpolation im Uhrzeigersinn

Befehl	G2 bzw. G02
Aufhebung	G00 [▶ 39], G01 [▶ 40] oder G03

Stellen Sie den Interpolationsmodus auf "kreisförmig/helikal, im Uhrzeigersinn". Der Interpolationsmodus gilt für diesen Satz und allen folgenden Sätze, bis er durch `G00`, `G01` oder `G03` zurückgesetzt wird. Wenn `G02` aktiv ist, führt die Programmierung eines Punktes zu einem kreisförmigen (oder spiralförmigen) Bogen, der mit der aktuellen Geschwindigkeit bearbeitet wird. (Siehe Allgemeine Codes (F, N, Q, X, Y, Z, A, B, C) [▶ 51].) Im Folgenden wird ein Kreisbogen betrachtet. Der spiralförmige Bogen wird später behandelt.

Ein Kreisbogen beginnt am aktuellen Punkt und endet am programmierten Punkt. Er dreht sich um die Normale der Arbeitsebene (PCS, d.h. Programmkoordinatensystem) im Mittelpunkt. Der Mittelpunkt kann über die Mittelpunktprogrammierung oder über die Radiusprogrammierung definiert werden.

Mittelpunktprogrammierung I, J, K

Bei der Mittelpunktprogrammierung wird der Mittelpunkt mit Hilfe der Parameter `I,J,K` standardmäßig relativ zum Startpunkt definiert. Der Mittelpunkt ist dabei die Summe aus dem Startpunkt und dem Vektor `[I,J,K]`. Alternative kann der Mittelpunkt auch absolute angegeben werden. Hierfür ist vorab mit dem ST-Kommando circleCenterReferenceSet [▶ 98] der Referenztyp auf absolute zu setzen. Die Parameter `I,J,K` sind optional und haben standardmäßig den Wert `0`. Wenn der Anfangs- und der Endpunkt in Bezug auf die Arbeitsebene gleich sind, wird ein Vollkreis ausgesendet.

EINSCHRÄNKUNGEN:

- Der Radius am Startpunkt und am Endpunkt muss gleich sein. Geringe Abweichungen sind jedoch zulässig und werden automatisch korrigiert (siehe Mittelpunktskorrektur [▶ 79]).
- Der Mittelpunkt darf nicht mit dem Anfangs- oder Endpunkt übereinstimmen.

Radiusprogrammierung U

Bei der Radiusprogrammierung wird der Mittelpunkt aus dem Radius abgeleitet, der durch den Parameter `U` gegeben ist. In der Regel gibt es zwei Kreisbögen, die mit dem vorgegebenen Radius vom Startpunkt zum Endpunkt führen. Ist der Radius positiv, wird der Weg des kürzeren Kreisbogens gewählt, andernfalls der Längere. Ansonsten wird der Absolutwert des Radius vom Interpreter berücksichtigt.

EINSCHRÄNKUNGEN:

- Die Radiusprogrammierung kann nicht zur Programmierung eines Vollkreises verwendet werden. Hierfür ist die Mittelpunktprogrammierung zu verwenden.
- Der Radius darf nicht Null sein.
- Der Radius darf nicht kleiner sein als die Hälfte des Abstands zwischen Anfangs- und Endpunkt in Bezug auf die Arbeitsebene.

Helix

Liegen Start- und Endpunkt nicht in einer zur Arbeitsebene parallelen Ebene, wird eine *spiralförmige Bewegung* durchgeführt.

TIPP: moveCircle3D

Die `ST`-Funktion `moveCircle3D` ist eine leistungsfähigere Methode zur Definition eines Kreises oder einer Spirale. Sie umfasst 3D-Bögen und Multiturn-Kreise.

Beispiel:

Das folgende Beispiel ergibt die in der Abbildung "BeispielG00G02" dargestellten Bahn. Der Satz `N10` verwendet die Radiusprogrammierung, um einen Kreisbogen im Uhrzeigersinn von `X0 Y0` nach `X10 Y10` mit dem Radius `10` zu definieren. Da der Radius positiv ist, wird der Mittelpunkt `c1` des kürzeren Bogens gewählt. Im Satz `N30` wird der Mittelpunkt `c2` des längeren Bogens verwendet, da der Radius negativ ist. Der Satz `N50` verwendet Mittelpunktprogrammierung, wobei der Mittelpunkt `c3=[60,0,0]` die Summe der Startpunkte `[50,0,0]` und `[I,J,K]=[10,0,0]` ist. Der Satz `N70` definiert einen Vollkreis mit dem Mittelpunkt `C04`, da der Anfangs- und Endpunkt gleich sind. Der Satz `N90` definiert eine Helix mit Mittelpunkt `C05` und Höhe `30` (in `Z`-Richtung).

```
N01 G00 X0  Y0
N10 G02 X10 Y10 U10 F6000
N20 G00 X30 Y0
N30 G02 X40 Y10 U-10
N40 G00 X50 Y0
N50 G02 X60 Y10 I10
N60 G00 X80 Y0
N70 G02 J10
N80 G00 X110 Y0
N90 G02 J10 X120 Y10 Z30
M30
```

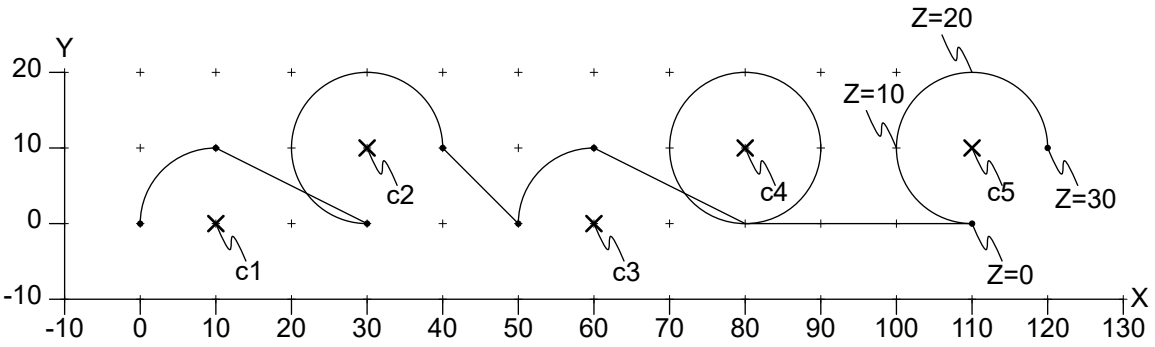


Abbildung "BeispielG00G02".

G03 Kreisinterpolation gegen den Uhrzeigersinn

Befehl	G3 bzw. G03
Aufhebung	G00 [▶ 39], G01 [▶ 40] oder G02

Stellen Sie den Interpolationsmodus auf "kreisförmig/spiralförmig, gegen den Uhrzeigersinn". Diese Interpolation verhält sich ähnlich wie `G02`. Der Interpolationsmodus gilt für diesen Satz und allen folgenden Sätze, bis er durch `G00`, `G01` oder `G02` zurückgesetzt wird.

Mittelpunktprogrammierung `I<vx> J<vy> K<vz>`

Legt den Mittelpunkt für kreisförmige Bewegungen fest. Siehe G2, G3 für Einzelheiten. Der Mittelpunkt ist definiert als `currentPoint + [vx,vy,vz]`. Die aktuelle Längeneinheit wird für `vx`, `vy`, `vz` verwendet. Die Parameter `I`, `J`, `K` sind optional und haben den Standardwert `0`.

Radiusprogrammierung `U<v>`

Im Kontext von `G2` oder `G3` wird der Radius auf `|v|` gesetzt. Die aktuelle Längeneinheit wird für `v` verwendet. Wenn `v` positiv ist, wird der kürzere Bogen zur Interpolation zwischen dem aktuellen und dem nächsten Punkt verwendet. Wenn `v` negativ ist, wird der längere Bogen verwendet. Siehe G2, G3 für Einzelheiten.

G303

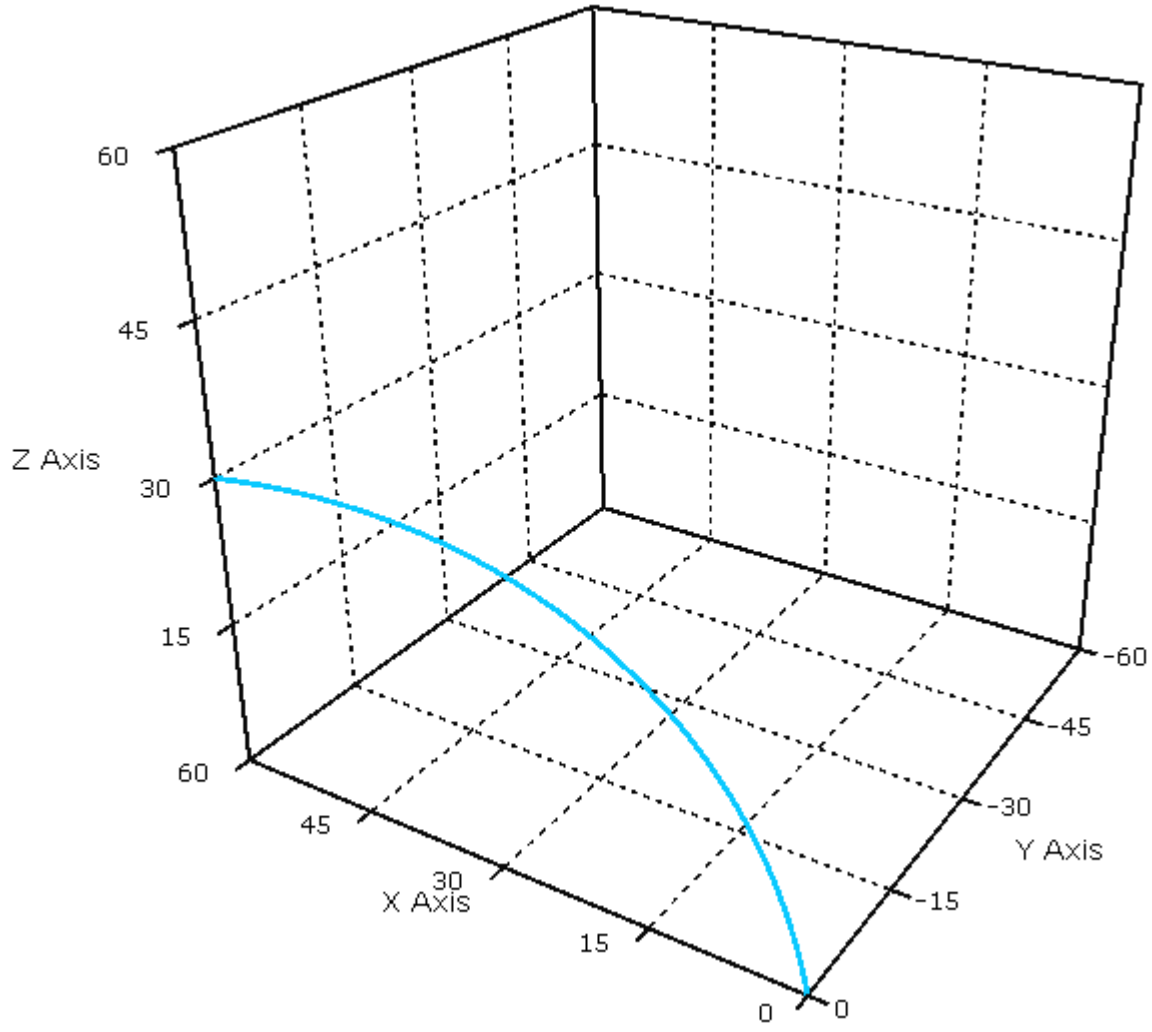
Mit G303 kann ein Kreisbogen (ein CIP-Kreis) programmiert werden, der frei im Raum platziert werden kann.

Der CIP-Kreis kann auch verwendet werden, um einen Kreis an einer beliebigen Stelle im Raum zu programmieren. Dazu ist es notwendig, nicht nur einen Endpunkt, sondern auch einen anderen Punkt auf der Bahn zu programmieren.

Damit der Kreis eindeutig beschrieben werden kann, dürfen alle 3 Punkte (Anfangspunkt ist implizit vorgegeben) nicht kollinear sein. Es lässt sich also auf diese Weise kein Vollkreis programmieren.

Als Parameter für den Bahnpunkt stehen I, J und K zur Verfügung, die standardmäßig relativ zum Kreisanfangspunkt beschrieben werden.

```
G90
N10 G01 F2000 X0 Y0 Z0              // P1 (start point): X, Y, Z,
N20 G303 I30 J-15 K15 X60 Y0 Z30    // P2 (path point): I, J, K
                                    // P3 (end point): X, Y, Z
N30 M02
```



Anforderungen G303

TwinCAT	GST
TwinCAT V3.1.4024.40	GST 3.1.8.62

4.4.8 Verweilzeit (G04)

Befehl	G4 bzw. G04
Aufhebung	Satzende
Parameter	F oder X

Unterbrechen Sie die Bearbeitung für eine bestimmte Dauer. Die Dauer wird entweder durch `X` oder `F` in der aktuellen Zeiteinheit definiert. (Siehe `unit` für Einzelheiten.)

Beispiel:

Im folgenden Beispiel wird davon ausgegangen, dass die aktuelle Zeiteinheit auf Sekunden eingestellt ist. Bei einer Ausführung des Programms fährt die Maschine zu `X10`, wartet für `1.5` Sekunden und fährt dann zu `X20`.

```
N10 G01 X10 F6000
N20 G04 F1.5
N30 G01 X20
M02
```

4.4.9 Genauhalt (G09, G60)

Die Genauhaltanweisung wird z.B. dann benutzt, wenn scharfe Konturecken hergestellt werden müssen. Dabei wird die Sollgeschwindigkeit der Bahn im Konturübergang bis auf null reduziert und anschließend wieder erhöht. Auf diese Weise wird sichergestellt, dass die programmierte Position genau angefahren wird.

G09 satzweiser *Genauhalt – Nichtmodal*

Befehl	G9 bzw. G09
Aufhebung	Satzende

G09 wirkt nur sollwertseitig.

G60 *Genauhalt - Modal*

Befehl	G60
Aufhebung	G00 [▶ 39]

4.4.10 Restweglöschen (G31)

Befehl	G31
Aufhebung	Satzende

G31 ("delete distance to go") wird aus dem NC-Programm satzweise aktiviert. Dieses Kommando ermöglicht es, aus der SPS mit dem Funktionsbaustein ItpDelDtgEx [▶ 213] den Restweg der aktuellen Geometrie zu löschen. D.h. trifft das Kommando während der Abarbeitung des Satzes ein, so wird mit den üblichen Verzögerungsrampen die Bewegung angehalten. Anschließend wird mit dem nächsten Satz im NC-Programm fortgefahren. Falls das SPS-Kommando nicht während der Ausführung eines Satzes mit angewähltem Restweglöschen eintrifft, wird dieses mit einer Fehlermeldung beantwortet.

G31 bewirkt immer einen impliziten Dekodierstopp, d.h. es erfolgt am Satzende immer ein Genauhalt.

Beispiel:

```
N10 G01 X0 Y0 F6000
N20 G31 G01 X2000
N30 G01 X0
N40 M02
```

Requirements

Development Environment	Target System
TwinCAT V3.1.4024.20	PC or CX (x86 or x64)

4.4.11 Nullpunktverschiebungen (G53, G54..G59)

G53 Unterdrückung der Nullpunktverschiebung

Befehl	G53 (Standardeinstellung)
Aufhebung	G54 bis G59

Deaktivieren Sie alle Nullpunktverschiebungen. Diese Einstellung ist die Standardeinstellung. Die Deaktivierung wird auch für den aktuellen Satz aktiv. Siehe Abschnitte Nullpunktverschiebung [▶ 85] und G58/ G59 für weitere Informationen.

G54..G57 einstellbare Nullpunktverschiebung

Befehl	G54 G55 G56 G57
Aufhebung	G53 oder Anwahl einer anderen einstellbaren Nullpunktverschiebung

Aktiviert die Translation, die mit dem angegebenen `G`-Code (`TZ54…TZ57`) verbunden ist. Aktiviert auch die Translationen von `G58` und `G59`. Die Translationen gelten für den aktuellen Satz und alle folgenden Sätze, bis sie geändert werden. Siehe Abschnitt Nullpunktverschiebung [▶ 85] für weitere Informationen.

G58, G59 programmierbare Nullpunktverschiebung

Befehl	G58 oder G59
Aufhebung	G53

Legt die Translation fest, die mit dem angegebenen `G`-Code verbunden ist. Der neue Translationswert wird durch die Parameter `X,Y,Z` angegeben, die obligatorisch sind. Standardmäßig sind die zugehörigen Translationen gleich Null. Siehe Abschnitt Nullpunktverschiebung [▶ 85] für weitere Informationen.

Beispiel:

Die resultierende MCS-Bahn (Maschinenkoordinatensystem) und die angewandten Translationen dieses Beispiels sind in Abbildung "BeispielG54G58G59" dargestellt.

- Die erste Zeile setzt die Translation, die mit `G54` verbunden ist, auf `[0,5,0]`.
- Die nächste Zeile setzt die programmierte Translation von `G58` auf `[0,10,0]`. Da Nullpunktverschiebungen weiterhin deaktiviert sind (Standard `G53`), stimmen das PCS (Programmkoordinatensystem) und das MCS (Maschinenkoordinatensystem) überein.
- Dementsprechend führt der Satz `N20` zu einer linearen Bewegung von der MCS-Koordinate (Maschinenkoordinatensystem) `[0,0,0]` zu `[20,0,0]`.
- Die nächste Zeile aktiviert `G54` und programmiert eine lineare Bewegung entlang `N30`, wobei `G54` vor der Bewegung aktiv wird. Die programmierte PCS-Koordinate (Programmkoordinatensystem) `[40,0,0]` wird auf die MCS-Koordinate (Maschinenkoordinatensystem) `[40,15,0]` abgebildet.
- Die nächste Zeile setzt die programmierte Transformation `G59` auf `[0,5,0]`. Dabei ändert sich die effektive Translation von `[0,15,0]` auf `[0,20,0]`. Da die aktuelle MCS-Koordinate (Maschinenkoordinatensystem) von dieser Änderung nicht betroffen sein darf, wird die aktuelle PCS-Koordinate (Programmkoordinatensystem) implizit auf `[40,-5,0]` gesetzt.
- Die nachfolgende `ST`-Funktion `frameGet` speichert diese Koordinaten in `[pcsX,pcsY,pcsZ]`.
- Die nächste Zeile programmiert lediglich die `X`-Koordinate des Endes des Segments `N50`. Daher lautet die PCS-Koordinate (Programmkoordinatensystem) des Endes des Segments `N50 [60,-5,0]` , die auf die MCS-Koordinate (Maschinenkoordinatensystem) `[60,15,0]` abgebildet wird. Mit anderen Worten: Die Translation `G59` ist aktiv, wird aber durch die Anpassung der aktuellen PCS-Koordinate (Programmkoordinatensystem) nicht sichtbar. (Weitere Informationen finden Sie im Abschnitt Transformationsanwendungen [▶ 101])
- Dies wird durch die letzte Zeile deutlich, die die PCS-Koordinate (Programmkoordinatensystem) des Endes des Segments `N60` auf `[80,0,0]` setzt. Diese Koordinate wird auf die Koordinate des MCS (Maschinenkoordinatensystem) `[80,20,0]` abgebildet.

```
!zeroOffsetShiftSet(g:=54, x:=0, y:=5, z:=0);
N10 G58 X0 Y10 Z0
N20 G01 X20 Y0 F6000
N30 G54 X40 Y0
N40 G59 X0 Y5 Z0
!VAR pcsX, pcsY, pcsZ : LREAL; END_VAR
!frameGet(x=>pcsX, y=>pcsY, z=>pcsZ);
N50 X60
```

```
N60 X80 Y0
M02
```

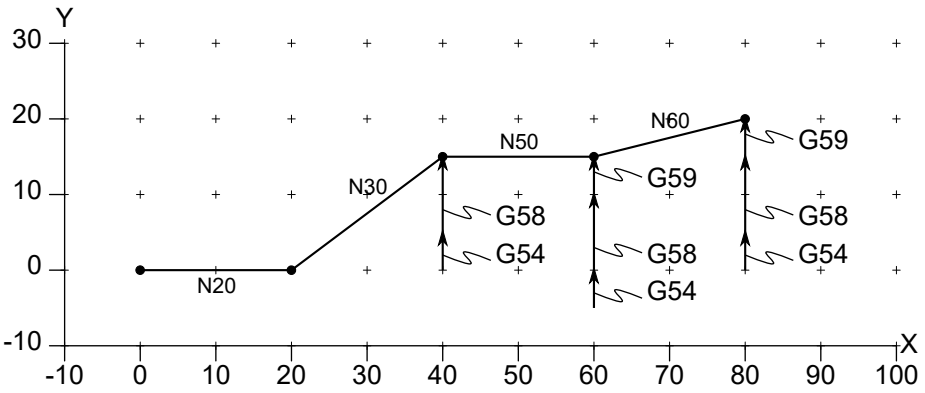


Abbildung "BeispielG54G58G59".

4.4.12 Arbeitsebene und Zustellrichtung (G17, G18, G19, P)

G17 Arbeitsebene XY

Befehl	G17 (Standardeinstellung)
Aufhebung	G18 oder G19

`XY`-Ebene als Arbeitsebene auswählen, d.h. die Normale der Arbeitsebene wird auf `[0,0,1]` gesetzt. Diese Arbeitsebene ist die Standard-Arbeitsebene.

G18 Arbeitsebene ZX

Befehl	G18
Aufhebung	G17 oder G19

`ZX`-Ebene als Arbeitsebene auswählen, d.h. die Normale der Arbeitsebene wird auf `[0,1,0]` gesetzt.

G19 Arbeitsebene YZ

Befehl	G19
Aufhebung	G17 oder G18

`YZ`-Ebene als Arbeitsebene auswählen, d.h. die Normale der Arbeitsebene wird auf `[1,0,0]` gesetzt.

P Festlegung der Zustellrichtung

`P<v>`

Werkzeugausrichtung umschalten. Der Wert von `v` muss `1` oder `-1`. Wenn `v` negativ ist, zeigt das Werkzeug in Richtung der Normale der Arbeitsebene. Andernfalls zeigt sie in die entgegengesetzte Richtung.

Beispiel:

Die sich ergebende MCS-Bahn(MCS: Maschinenkoordinatensystem) des folgenden Beispiels ist in Abbildung "BeispielP" dargestellt. In der ersten Zeile des Programms wird für das Werkzeug `1` eine Länge von `10` festgelegt. `G18` aktiviert die `XZ`-Arbeitsebene.

`N10`: Das Ende des Segments `N10` unterliegt keiner Werkzeugkorrektur, da `D0` aktiv ist.

`N20`: Für das Segment `N20` ist das Werkzeug `1` mit einer positiven Werkzeugausrichtung aktiv. Um die Werkzeuglänge zu kompensieren, wird die Translation `[0,10,0]` angewendet. (Siehe Abschnitt Transformationen [▶ 99] für Einzelheiten.) Dabei wird der PCS-Endpunkt (Programmkoordinatensystem) `[20,10,0]` von `N20` auf den MCS-Endpunkt (Maschinenkoordinatensystem) `[20,20,0]` abgebildet. Der MCS-Punkt (Maschinenkoordinatensystem) und die angewandte Transformation sind in Abbildung "BeispielP" dargestellt.

`N30`: Im Satz `N30` wird die Werkzeugausrichtung umgeschaltet, wodurch die Translation auf `[0,-10,0]` gesetzt wird. Diese Translation wird auf den PCS-Endpunkt (Programmkoordinatensystem) von `N30` angewandt und ergibt den MCS-Endpunkt (Maschinenkoordinatensystem) `[30,0,0]`.

`N20..N90`: Die Sätze `N60..N90` sind ähnlich wie `N20..N50`, nur dass die `Y`-Koordinate nicht programmiert ist. Daher wird die Werkzeuglängenkorrektur nicht sichtbar, obwohl sie aktiv ist. Dieses Verhalten tritt auf, weil der aktuelle PCS-Punkt (Programmkoordinatensystem) bei einer geänderten Transformation immer angepasst wird. (Weitere Informationen finden Sie im Abschnitt Transformationsanwendungen [▶ 101])

```
!toolSet(index:=1, tooltype:=tooltypeDrill, length:=10);
G18
N10 X10 Y10 D0 F6000
N20 X20 Y10 D1
N30 X30 Y10 P-1
N40 X40 Y10 P1
N50 X50 Y10 D0
N60 X60 D1
N70 X70 P-1
N80 X80 P1
N90 X90 D0
M02
```

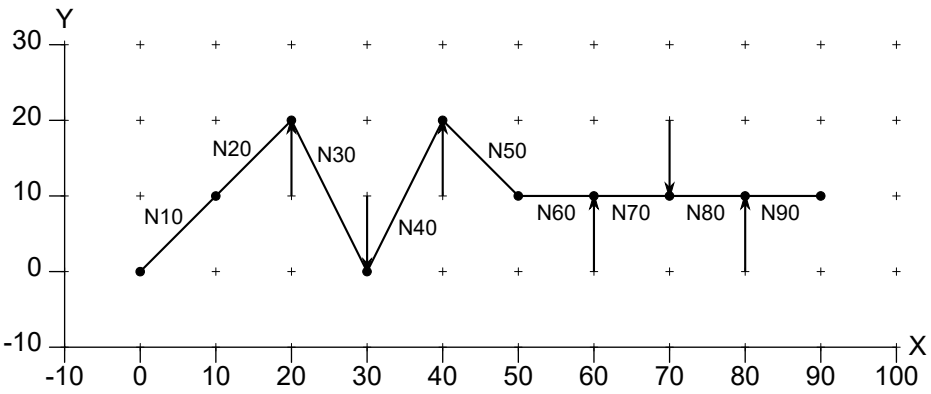


Abbildung "BeispielP".

4.4.13 Maßangaben Inch / metrisch (G70, G71, G700, G710)

G70 Maßangaben in Inch

Befehl	G70
Aufhebung	G71, G700 oder G710

Einheit für Längen auf *Inch* einstellen. Die neue Einheit gilt auch für den aktuellen Satz. `G70` ist gleichbedeutend mit dem Aufruf `unitLengthSet(unitLengthInch)`. Die Einheit für die Geschwindigkeit ist davon nicht betroffen. Siehe UnitLength [▶ 86] und G71 für Einzelheiten.

G71 Maßangaben in mm

Befehl	G71 (Standardeinstellung)
Aufhebung	G70, G700 oder G710

Einheit für Längen auf *Millimeter* einstellen. Die neue Einheit gilt auch für den aktuellen Satz. `G71` ist gleichbedeutend mit dem Aufruf `unitLengthSet(unitLengthMillimeter)`. Die Einheit für die Geschwindigkeit ist davon nicht betroffen. Siehe UnitLength [▶ 86] für weitere Einzelheiten.

Beispiel:

In Abbildung "BeispielG70G71" ist die Bahn des folgenden Beispiels dargestellt, das die Einheit *Millimeter* verwendet.

- In der ersten Zeile des Programms wird die Einheit für Längen auf *Inch* gesetzt. Diese Einheit wird in der gleichen Zeile verwendet, um `X2` in Inch zu interpretieren. Die Bahn `N10` endet also an der Position [`50.8` mm, `0` mm, `0` mm].
- Dementsprechend bewegt die nächste Zeile das Werkzeug entlang `N20` in Richtung [`50.8` mm, `25.4` mm, `0` mm].
- In der letzten Zeile wird die Einheit auf *Millimeter* gesetzt. Daher endet die Bahn `N30` an der Position [`80` mm, `25.4` mm, `0` mm]. Dementsprechend ist das Segment `N30` eine horizontale Linie.

```
N10 G01 X2 G70 F6000
N20 G01 Y1
N30 G01 X80 Y25.4 G71
M02
```

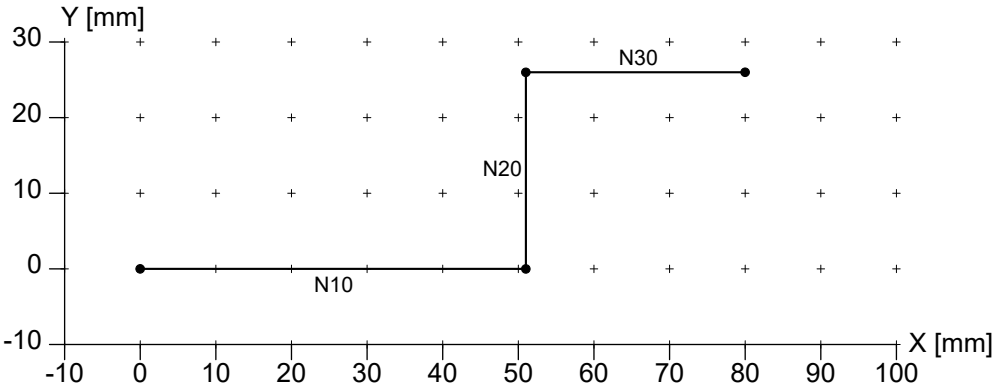


Abbildung "BeispielG70G71".

G700 Maßangabe in Inch mit Verrechnung des Vorschubs

Befehl	G700
Aufhebung	G70, G71 oder G710

Wie `G70`, gilt aber auch für die Interpretation der Geschwindigkeit. Die neue Einheit wird im aktuellen Satz wirksam. `G700` ist gleichbedeutend mit den Aufrufen `unitLengthSet(unitLengthInch)` und `unitVelocitySet(unitLengthInch,unitTimeMinute)`.

G710 Maßangabe in Millimetern mit Verrechnung des Vorschubs

Befehl	G710
Aufhebung	G70, G71 oder G700

Wie `G71`, gilt aber auch für die Interpretation der Geschwindigkeit. Die neue Einheit wird im aktuellen Satz wirksam. `G710` ist gleichbedeutend mit den Aufrufen `unitLengthSet(unitLengthMillimeter)` und `unitVelocitySet(unitLengthMillimeter,unitTimeMinute)`.

Beispiel:

Die Bahn des folgenden Beispiels ist in Abbildung "BeispielG700G710" dargestellt.

- Die erste Zeile definiert eine lineare Bewegung auf `[1 in, 1 in, 0 in]` mit einer Geschwindigkeit von `100 in/min`.
- Die zweite Zeile setzt die Längeneinheit auf `mm`, hat aber keinen Einfluss auf die Geschwindigkeitseinheit. Sie definiert eine Bewegung auf `[30 mm, 10 mm, 0 mm]` mit einer Geschwindigkeit von `50 in/min`.
- In der letzten Zeile wird auch die Geschwindigkeitseinheit auf `mm/min` gesetzt. Es findet also eine Bewegung nach `[40 mm, 20 mm, 0 mm]` mit einer Geschwindigkeit von `1000 mm/min` statt.

```
N10 G700 G01 X1 Y1 F100
N20 G71 G01 X50 Y10 F50
N30 G710 G01 X80 Y20 F1000
```

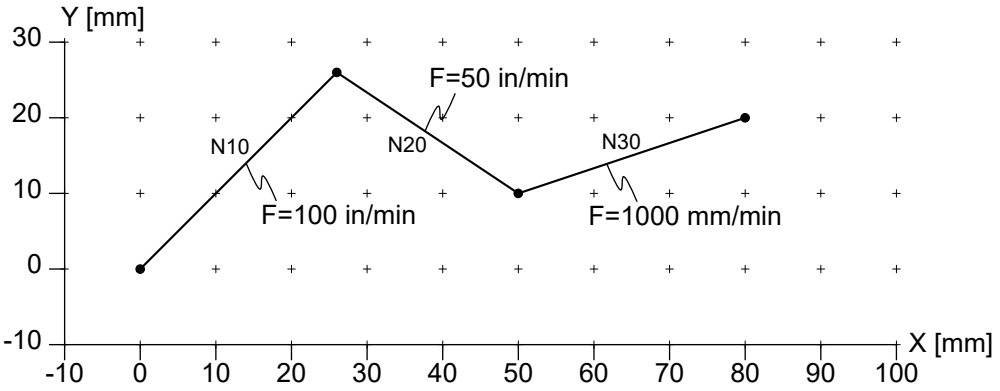


Abbildung "BeispielG700G710".

4.4.14 Maßangaben absolute / relative (G90, G91)

G90 Absolutmaßangabe

Befehl	G90
Aufhebung	G91

Schaltet auf absolute Koordinaten um. `X`, `Y`, `Z` werden als absolute PCS-Koordinaten (Programmkoordinatensystem) interpretiert. Diese Einstellung ist die Standardeinstellung. Die Umstellung wird im gleichen Satz aktiv.

G91 Kettenmaßangabe

Befehl	G91
Aufhebung	G90

Schaltet auf relative Koordinaten um. `X`, `Y`, `Z` und werden so interpretiert, dass sie sich auf den aktuellen Punkt beziehen, d. h. der nächste Punkt wird als Summe von `[X,Y,Z]` und dem aktuellen Punkt errechnet. Die Umstellung wird im gleichen Satz aktiv.

Manuelle Implementierung von Offsets

Durch die Verwendung von `G91` und die damit verbundene Umstellung auf relative Koordinaten werden die zuvor definierten Werkzeugoffsets und Nullpunktverschiebungen nicht innerhalb dieser Koordinaten ausgewertet und müssen daher im Rahmen des `G91`-Codes manuell definiert und umgesetzt werden.

Beispiel:

Die Bahn des folgenden Beispiels ist in der Abbildung "BeispielG90G91" dargestellt. Die Umstellung auf G90/ G91 wird sofort wirksam.

```
N10 G90 G01 X10 Y20 F6000
N20 X20 Y10
N30 G91 X10 Y10
N40 X10 Y-10
N50 G90 X50 Y20
M02
```

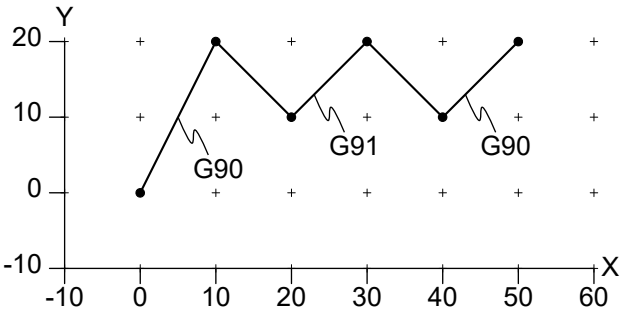


Abbildung "BeispielG90G91".

4.4.15 M-Funktionen (M)

M

M<v>

Löst die M-Funktion v aus. Das Timing und Verhalten hängt von der Definition von v in der Entwicklungsumgebung von TwinCAT ab.

M2 und M30 sind intern definiert. Beide Funktionen lösen eine Synchronisation mit dem NC-Kanal aus. (Siehe wait()-Funktion, Kapitel Synchronisation [▶ 81].) Beide Funktionen stoppen die Ausführung des GST-Programms. Aufgrund dieser Reihenfolge wartet der Interpreter auf die Beendigung des NC-Kanals, bevor er anhält.

Darüber hinaus setzt M30 auch alle schnellen M-Funktionen und H, S, T zurück.

Es darf nicht mehr als eine M-Funktion vom Typ Handshake in einem Satz geben.

Die M-Funktionen M2 und M30 müssen nicht vom Anwender in der Entwicklungsumgebung von TwinCAT definiert werden.

Beispiel:

Dieses Beispiel setzt die folgenden Definitionen der M-Funktionen voraus:

M10:	Schnell vor Bewegung.
M11:	Schnell nach Bewegung.
M12:	Schnell vor Bewegung, Auto-Reset, Reset M10, M11.
M20:	Handshake vor Bewegung.

`M21`: Handshake nach Bewegung.

`M02`: Programmende.

Die Abbildung "BeispielM10M11M12M20M21" veranschaulicht die programmierte Bahn und die Aktivierung der `M`-Funktionen. Die schnellen `M`-Funktionen `M10`, `M11` werden von `M12` zurückgesetzt, das seinerseits automatisch zurückgesetzt wird.

```
N10 G01 X10 F6000
N20 X30 M10 M20
N30 X50 M11 M21
N40 X70
N50 X90 M12
M02
```

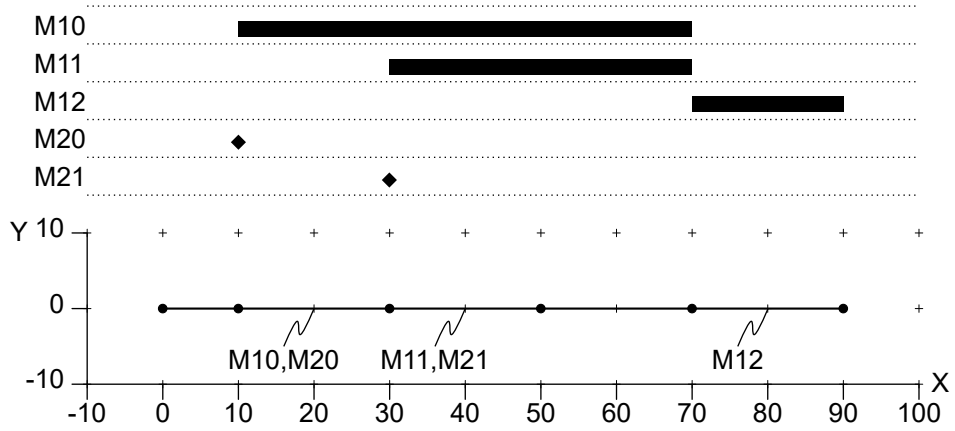


Abbildung "ExampleM10M11M12M20M21".

4.4.16 Allgemeine Codes (F, N, Q, X, Y, Z, A, B, C)

F

F<v>

Geschwindigkeit auf `v` einstellen. Gilt für den aktuellen Satz und alle folgenden Sätze, bis eine neue Geschwindigkeit programmiert wird. Es wird die aktuell gewählte Einheit für die Geschwindigkeit verwendet. (Siehe Abschnitt unitVelocitySet [▶ 86] für Einzelheiten.) Die Standardgeschwindigkeit ist `0`.

Die Geschwindigkeit muss auf einen Wert ungleich Null gesetzt werden, bevor eine Bewegung programmiert wird. Anderenfalls wird ein Fehler ausgegeben.

Beispiel:

Die ersten beiden Segmente `N10` und `N20` werden mit einer Geschwindigkeit von `6000` mm/min bearbeitet, und das letzte Segment `N30` wird mit einer Geschwindigkeit von `3000` mm/min bearbeitet.

```
N10 G01 X100 F6000
N20 G01 X200
N30 G01 X300 F3000
M02
```

N

N<v>

Satznummer auf v einstellen. Typischerweise wird die Satznummer verwendet, um den Fortschritt des NC-Programms zu überwachen.

Q

Q<i>=<v>

Wert der Achse Q<i> auf v einstellen, wobei i im Bereich 1 bis 5 liegen muss. Die Q-Achsen verwenden Linearinterpolation.

Die Adressbuchstaben Q und R werden aus historischen Gründen in besonderer Weise behandelt.

Die Adresse Q<i> muss von einem G-Ausdruck oder einem ST-Ausdruck gefolgt werden. Das G-Wort Q1100 ist ungültig. Verwenden Sie stattdessen Q1=100.

Beispiel:

Die Bahn des folgenden Beispiels ist in Abbildung "BeispielQ" dargestellt. Die Q-Achsen werden linear mit der Interpolation einer Bewegung interpoliert. Der letzte Satz (N40) führt zu einer Linearinterpolation einer Q-Achse ohne gleichzeitige Bewegung.

```
N10 G01 X30 Y0 Q1=100 F6000
N20 G02 X50 Y20 I20 Q2=200
N30 G01 X60 Q1=300 Q2=300
N40 Q1=0
M02
```

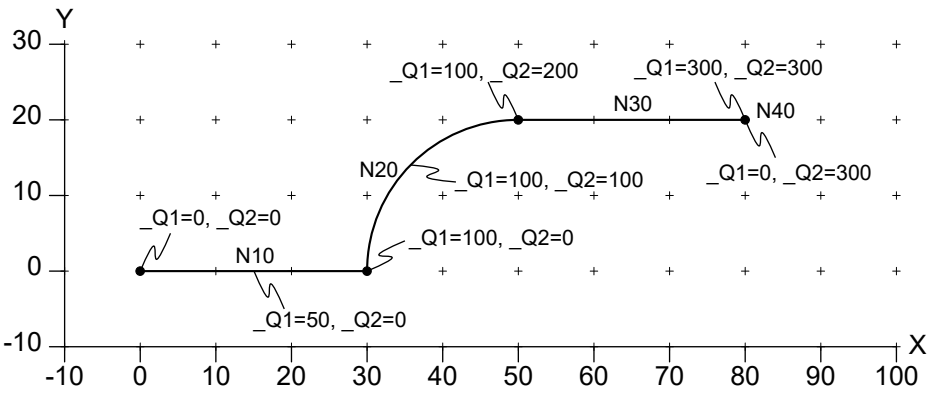


Abbildung "BeispielQ".

X

X<v>

Setzt die X-Koordinate des nächsten Punktes auf v. Die aktuelle Längeneinheit wird für v verwendet.

Y

Y<v>

Setzt die Y-Koordinate des nächsten Punktes auf v. Die aktuelle Längeneinheit wird für v verwendet.

Z

Z<v>

Setzt die Z-Koordinate des nächsten Punktes auf v. Die aktuelle Längeneinheit wird für v verwendet.

A

`A<v>`

Setzt den A-Winkel der nächsten Orientierung auf v. Die aktuelle Winkeleinheit wird für v verwendet.

B

`B<v>`

Setzt den B-Winkel der nächsten Orientierung auf v. Für v wird die aktuelle Winkeleinheit verwendet.

C

`C<v>`

Setzt den C-Winkel der nächsten Orientierung auf v. Für v wird die aktuelle Winkeleinheit verwendet.

4.5 ST - Strukturierter Text (IEC 61131-3)

4.5.1 Kommentare

Zeilenkommentar

```
<st-code> // <comment>
```

Text zwischen '//' und dem Ende der Zeile wird in ST-Code als Kommentar behandelt.

Beispiel:

```
{
VAR
    i : INT; // this variable is primarily used in FOR-loops for counting
END_VAR
}
```

/* */ Kommentar

```
<st-code> /* <comment>
<comment> */ <st-code>
```

Text zwischen '/*' und '*/' wird als Kommentar in ST behandelt. Diese Art von Kommentar kann bis zu einer Tiefe von 3 verschachtelt werden. Der '/*…*/'-Kommentar kann überall zwischen Literalen, Schlüsselwörtern, Bezeichnern und speziellen Symbolen erscheinen. Er kann auch G-Codezeilen enthalten.

Beispiel:

Das folgende Beispiel zeigt die Notation von Kommentaren in ST-Code. Der erste Kommentar befindet sich innerhalb einer Variablendeklaration. Der zweite Kommentar schließt eine ganze ST-Schleife ein. Der Kommentar enthält weitere Kommentare und eine G-Code Zeile, die wiederum einen G-Code Kommentar enthält.

```
{
VAR i /* used for counting */ : INT;   END_VAR

/* The following loop is commented out.
FOR i := 0 TO 10 DO
    /* zigzag pattern */
    ! G01 (linear interpolation) X=i Y{i MOD 2} F6000
    // end of loop
END_FOR;
*/
}
```

(* *) Kommentar

```
<st-code> (* <comment>
<comment> *) <st-code>
```

Text zwischen '(*' und '*)' wird als Kommentar in ST behandelt. Dieser Kommentartyp kann bis zu einer Tiefe von 3 verschachtelt werden. Er ähnelt dem /*…*/-Kommentar.

4.5.2 Literale

Ganzzahlige Literale

Dezimal	18
Binär	2#10010
Oktal	8#22
Hexadezimal	16#12

Der gleiche ganzzahlige Wert in dezimaler, binärer, oktaler und hexadezimaler Schreibweise.

Real-Literale

Notation der reellen Werte

```
1.0
```

```
1.602E-19
```

Boolesche Literale

Notation der booleschen Werte

```
0
```

```
1
```

```
TRUE
```

```
FALSE
```

Getypte Literale

```
<typename>#<literal>
```

Getypte Literale, bei denen Typname ein nativer Typ (z. B. Word oder LReal) oder ein Aufzählungstyp ist (um Mehrdeutigkeiten zu vermeiden).

Die Typisierung von Literalen ist in der Regel in GST nicht notwendig, da der Interpreter ein Typisierungssystem implementiert, das ungetypte Literale korrekt behandelt. Es gibt einige wenige Ausnahmen, in denen der Typ eines Literals für die Semantik von Bedeutung ist, wie im folgenden Beispiel.

Beispiel:

Die erste Zuweisung weist w den Wert 16#80 zu, während die zweite Zuweisung den Wert 16#8000 w zuweist.

```
{
VAR  w: word;  END_VAR
w :=  ror(BYTE#1,1);
```

```
w :=  ror(WORD#1,1);
}
```

Zeichenfolge-Literale

```
"abc"
```

```
'abc'
```

Notation einer 2-Byte- bzw. einer 1-Byte-Zeichenfolge Beachten Sie, dass es keine implizite Konvertierung zwischen beiden Typen gibt. Die folgenden Escape-Sequenzen können in beiden Arten von Literalen verwendet werden:

`$L`	Zeilenvorschub
`$N`	Zeilenumbruch
`$P`	Seitenvorschub
`$R`	Carriage return
`$t`	Tabulator
`$'` oder `$"`	Anführungszeichen
`$<2 or 4 hexadecimal digits>`	Zeichen des angegebenen Codes

.

Dauer-Literale

```
T#[+/-]<value><unit>[…]<value><unit>
```

```
TIME#[+/-] <value><unit>[…]<value><unit>
```

```
LT#[+/-]<value><unit>[…]<value><unit>
```

```
LTIME#[+/-]<value><unit>[…]<value><unit>
```

Zeitliterale vom Typ `TIME` oder `LTIME`. Das Literal besteht aus einem optionalen Zeichen (`+/-`) und einer Folge von `value`/ `unit` Paaren. `Value` muss eine ganze Zahl sein, mit Ausnahme der letzten Zahl, die auch eine Fließkommazahl sein kann. `Values` darf nicht negativ sein und kann beliebig groß sein. `Units` müssen in der folgenden Reihenfolge erscheinen.

`d`	Tag
`h`	Stunde
`m`	Minute
`s`	Sekunde
`ms`	Millisekunde
`us`	Mikrosekunde

`ns`	Nanosekunde

Eine beliebige Untermenge von `units` kann in einem Literal verwendet werden. So ist zum Beispiel das Literal `T#1d15ms1500.01us` gültig.

Datumsliterale

```
DATE#<yyyy>-<mm>-<dd>

D#<yyyy>-<mm>-<dd>

LDATE#<yyyy>-<mm>-<dd>

LD#<yyyy>-<mm>-<dd>
```

Datumsliteral vom Typ `DATE` oder `LDATE`. Das Literal wird als UTC interpretiert, d. h. Zeitzone, Sommerzeit und Schaltsekunden werden nicht berücksichtigt. Die Jahreszahl darf nicht kleiner als `1970` sein. Die Werte `yyyy`, `mm` und `dd` müssen ganzzahlige Werte sein, d.h. `D#1980-20-10` ist z.B. ein gültiges Datumsliteral.

Tageszeit-Literale

```
TIME_OF_DAY#<hh>:<mm>:<ss>

TOD#<hh>:<mm>:<ss>

LTIME_OF_DAY#<hh>:<mm>:<ss>

LTOD#<hh>:<mm>:<ss>
```

Tageszeit-Literal vom Typ `TOD` oder `LTOD`. Das Literal wird als UTC interpretiert, d. h. Zeitzone, Sommerzeit und Schaltsekunden werden nicht berücksichtigt. `hh` und `mm` müssen ganzzahlige Werte sein. `ss` kann eine Ganzzahl oder eine Fließkommazahl sein, d. h. `TOD#7:30:3.1415` ist beispielsweise ein gültiges Literal.

Datums- und Zeitformeln

```
DATE_AND_TIME#<yyyy>-<mm>-<dd>-<hh>:<mm>:<ss>

DT#<yyyy>-<mm>-<dd>-<hh>:<mm>:<ss>

LDATE_AND_TIME#<yyyy>-<mm>-<dd>-<hh>:<mm>:<ss>

LDT#<yyyy>-<mm>-<dd>-<hh>:<mm>:<ss>
```

Datums- und Zeitliteral vom Typ `DT` oder `LDT`. Das Literal wird als UTC interpretiert, d. h. Zeitzone, Sommerzeit und Schaltsekunden werden nicht berücksichtigt. Dieses Literal ist eine Kombination aus dem Datumsliteral und dem Tageszeitliteral. Die entsprechenden Regeln für diese beiden Teile gelten analog.

4.5.3 Native Datentypen

Bitstring-Typen

`BOOL`, `BYTE`, `WORD`, `DWORD`, `LWORD`

Bitstring-Typen von 1, 8, 16, 32 und 64 Bit. Implizite Umwandlung von links nach rechts unter Verwendung der Nullerweiterung.

Unsigned Integer Typen

`USINT`, `UINT`, `UDINT`, `ULINT`

Vorzeichenlose Integer-Typen von 8, 16, 32 und 64 Bit. Implizite Konvertierung von links nach rechts unter Beibehaltung des Wertes.

Signed Integer Typen

`SINT, INT, DINT, LINT`

Vorzeichenbehaftete Integer-Typen von 8, 16, 32 und 64 Bit. Implizite Konvertierung von links nach rechts unter Beibehaltung des Wertes. Ein vorzeichenloser Typ von `n` Bit wird ebenfalls implizit in einen vorzeichenbehafteten Typ von `m` Bit umgewandelt, wobei `m > n` gelten muss. Es gibt keine implizite Konvertierung zwischen Bitstring-Typen und Integer-Typen.

Fließkomma-Typen

`REAL, LREAL`

Fließkommadatentypen von 32 und 64 Bit. Implizite Konvertierung von links nach rechts unter Beibehaltung des Wertes.

String-Typen

`string[<length>]`

`wstring[<length>]`

1-Byte- und 2-Byte-Strings mit definierter `length`. Wenn `length` nicht angegeben wird, gilt `255` als Standardwert.

Character-Typen

`char`

`wchar`

Single 1-Byte und 2-Byte Character eines Strings. Kann implizit in einen String umgewandelt werden.

Time-Related-Typen

`TIME, LTIME`

`DATE, LDATE`

`TIME_OF_DAY, TOD, LTIME_OF_DAY, LTOD`

`DATE_AND_TIME, DT, LDATE_AND_TIME, LDT`

Datentypen für Dauer, Datum und Uhrzeit. Intern werden alle Werte dieser Typen mit einer Granularität von 1 Nanosekunde dargestellt. Die Werte der datumsbezogenen Typen stellen die Anzahl der Nanosekunden seit `1.1.1970` (UTC) dar. Schaltsekunden werden ignoriert. Eine implizite Konvertierung von einem nicht-`L`-Typ zu einem `L`-Typ ist zulässig, z. B. von `TIME` zu `LTIME`.

4.5.4 Benutzerdefinierte Typen

Abgeleitete Typen

```
TYPE
<typeName>: <typeName> := <defaultValue>;
END_TYPE
```

Definition eines neuen Typs als Alias für einen bestehenden Typ. Der Standardwert ist optional.

Aufzählungstypen

```
TYPE
<typeName> : (<enumValue>, …, <enumValue>) := <defaultValue>;
```

```
END_TYPE
```

Definition eines Aufzählungstyps. Der Standardwert ist optional.

Aufzählungstypen mit definierten Werten

```
TYPE

<typeName> : (<enumValue>:=<integer value>, …,

<enumValue>:=<integer value>) := <defaultValue>;

END_TYPE
```

Definition eines Aufzählungstyps mit benutzerdefinierten Werten für jedes Element. Der Standardwert ist optional.

Arraytypen

```
TYPE

<typeName>: ARRAY [<from>..<to>,<from>..<to>] OF <typeName> :=

[<defaultValue>, <repetition>(<defaultValue>), …];

END_TYPE
```

Definition eines Arraytyps. Das Array kann mehrdimensional sein. Der Indexbereich wird für jede Dimension definiert. Zur Laufzeit werden die Grenzen des Arrays überprüft. Eine Grenzüberschreitung führt zu einem Laufzeitfehler. Die Standardwerte werden in aufsteigender Reihenfolge definiert, beginnend mit der letzten Dimension. Ein Wert kann wiederholt werden, indem er in Klammern gesetzt wird, denen die Anzahl der Wiederholungen vorangestellt wird. Wenn die Anzahl der definierten Standardwerte nicht mit der Arraygröße übereinstimmt, wird die Initialisierung abgeschnitten oder mit dem Standardwert des Elementtyps aufgefüllt. In beiden Fällen wird eine Kompilierzeit-Warnung ausgegeben.

Structure-Typen

```
TYPE

<typeName>: STRUCT

    <memberName>: memberType;

    …

END_STRUCT := (<memberName> := <defaultValue>, …);

END_TYPE
```

Definiert einen Structure-Typ mit den angegebenen Mitgliedern. Derzeit wird der Standardwert nach der Typdefinition eingefügt. Dieser Positionsstil unterscheidet sich von der `ST`-Norm.

Pointer-Typen

```
TYPE

    <typeName>: REF_TO <basetypeName>;

END_TYPE
```

Definiert einen Pointer-Typ mit dem angegebenen Basistyp.

4.5.5 Kontrollstrukturen

IF-THEN-ELSIF-ELSE

```
IF <condition> THEN

    <statements>

ELSIF <condition> THEN

    <statements>

ELSE

    <statements>

END_IF;
```

Bedingte Anweisung. Der `ELSIF`-Zweig und `ELSE`-Zweig sind optional. `ELSIF` kann beliebig oft wiederholt werden.

CASE OF

```
CASE <expression> OF

    <value>, <value>, …, <value>: <statements>

ELSE

    <statements>

END_CASE;
```

Die Case-Liste besteht aus einer durch Komma getrennten Folge von Werten oder Bereichen. Nur der erste passende Fall wird ausgeführt. Die optionale `ELSE`-Zweig wird ausgeführt, wenn kein Fall zutrifft.

FOR

```
FOR <variable> := <expression> TO <expression> BY <expression> DO

    <statements>

END_FOR;
```

Iteriert über die angegebene `variable` im definierten Bereich (einschließlich) unter Verwendung der angegebenen Schrittweite. Wenn letzteres weggelassen wird, hat es den Standardwert `1`.

WHILE

```
WHILE <condition> DO

    <statements>

END_WHILE;
```

Kontrolle vor der Schleife

REPEAT

```
REPEAT

    <statement>

UNTIL <condition>

END_REPEAT;
```

Kontrolle am Ende der Schleife. Die Unterbrechungsbedingung wird nach Ausführung der <statements> ausgewertet, die die Schleife enthält.

EXIT

```
EXIT;
```

EXIT kann innerhalb von Schleifen verwendet werden, um die Schleife zu verlassen. Wenn Schleifen verschachtelt sind, bleibt nur die innerste Schleife übrig. Wenn das Schlüsselwort EXIT von keiner Schleife umgeben ist, wird ein Kompilierzeitfehler ausgegeben.

4.5.6 Sprunganweisung

```
LABEL <label_name>;

  <statements>

GOTO <label_name>;
```

Die LABEL- und GOTO-Anweisungen ermöglichen Sprünge im G-code.

```
LABEL <label_name>;
```

Mit LABEL wird ein Sprungziel mit dem angegebenen Namen an die angegebenen Position eingebunden. Dabei muss <label_name> ein gültiger Bezeichner sein, wie der Name einer Variable oder einer Funktion. Eine LABEL-Anweisung kann dort verwendet werden, wo eine Anweisung erwartet wird, außer im Zusammenhang mit einer CASE-Anweisung.
Ist eine LABEL-Anweisung in einer Funktion definiert, ist deren Gültigkeitsbereich auf die Funktion beschränkt.
Befindet sich die Label-Anweisung im globalen Code, so ist diese nur im globalen Bereich gültig.
Es ist ein Fehler, wenn zwei LABEL-Anweisungen mit dem gleichen Namen im gleichen Gültigkeitsbereich definiert sind.

```
GOTO <label_name>;
```

Sorgt für die Fortsetzung der Ausführung des g-Codes an der Stelle, an der das angegebene LABEL eingebunden ist.
Es ist ein Fehler, wenn der angegebene <label_name> nicht im Gültigkeitsbereich der GOTO-Anweisung definiert ist.
Zudem ist es nicht erlaubt, GOTO-Anweisungen innerhalb einer CASE-Anweisung zu verwenden.

Beispiel

```
N10 G00 X0 Y0 Z0
N20 G01 X10 F1000
!R1 := 0;

!LABEL jumpPos;
!R2 := R1;
!R1 := R2 + 1;
N30 G01 Y=10*R1

!IF R1 = 1 THEN
    !GOTO jumpPos;
!END_IF

N40 G01 Z =10*r1

!IF R1 = 2 THEN
    !goto jumpPos;
!END_IF

M30
```

Voraussetzungen

TwinCAT	GST Interpreter
TwinCAT 3.1.4024.47	GST 3.1.8.67

4.5.7 Benutzerdefinierte Funktionen

Function Definition

```
FUNCTION <name> : <returntype>
VAR_INPUT
    <variable declarations>
END_VAR
VAR_OUTPUT
    <variable declarations>
END_VAR
VAR_IN_OUT
    <variable declarations>
END_VAR
VAR
    <variable declarations>
END_VAR
VAR_EXTERNAL
    <variable declarations>
END_VAR
    <statements>
END_FUNCTION
```

Deklariert eine Funktion. Danach ist sie über ihren Namen aufrufbar. Die Deklaration des Rückgabetyps ist optional. Wird er angegeben, gibt die Funktion einen Wert des angegebenen Typs zurück. Der Rückgabewert wird innerhalb des Funktionskörpers durch eine Zuweisung an den Funktionsnamen definiert.

Die Funktion kann Eingangs-, Ausgangs- und In-Out-Parameter enthalten. Die Reihenfolge der Deklaration ist von Bedeutung. Sie wird für nicht-formale Aufrufe verwendet. Deklarierte Variablen werden nur innerhalb des Funktionskörpers verwendet. Externe Variablen werden aus dem globalen Bereich importiert. Variablen und Parameter sind nicht persistent, d.h. sie behalten ihren Wert zwischen zwei Aufrufen nicht bei.

Nicht-formaler Funktionsaufruf

```
<functionname>(<expression>, …, <expression>)
```

Nicht-formaler Funktionsaufruf. Die Reihenfolge der Ausdrücke muss mit der Anzahl und der Reihenfolge der deklarierten Parameter übereinstimmen.

Formaler Funktionsaufruf

```
<functionname>(
    <inputParamName>  := <expression>,
    <outputParamName> => <variableName>,
    <inputParamName>  := <variableName>)
```

Formaler Funktionsaufruf. Parameter werden durch ihren Namen identifiziert. Wenn ein deklarierter Parameter nicht aufgeführt ist, wird er implizit auf seinen Standardwert gesetzt.

Formales und Nicht-Formales nicht vermischen

Die Vermischung von formalen und nicht-formalen Funktionsaufrufen führt zu ungültiger `GST`-Syntax.

4.5.8 Standardfunktionen

4.5.8.1 Typkonvertierung

Typkonvertierung (*_TO_*)

```
<nativeType>_to_<nativeType>(x)
```

```
to_<nativeType>(x)
```

Explizite Konvertierung zwischen den angegebenen nativen Typen. Die zweite Alternative ist für jeden anwendbaren Typ überladen.

Bei der Umwandlung von Fließkommazahlen in Ganzzahlen wird `x` gerundet.

4.5.8.2 Arithmetik und Trigonometrie

ABS

```
ABS(x)
```

Liefert den Absolutwert von `x`.

Die Funktion ist für jeden Ganzzahl- und Fließkommatyp überladen. Der Typ von `x` wird als Rückgabetyp verwendet.

SQRT

```
SQRT(x)
```

Liefert die Quadratwurzel von `x`.

Die Funktion ist für jeden Fließkommatyp überladen. Der Typ von `x` wird als Rückgabetyp verwendet.

EINSCHRÄNKUNG:

Die Variable `x` darf nicht negativ sein.

LN

```
LN(x)
```

Liefert den natürlichen Logarithmus von `x`, d. h. den Logarithmus zur Basis `e`.

Die Funktion ist für jeden Fließkommatyp überladen. Der Typ von `x` wird als Rückgabetyp verwendet.

EINSCHRÄNKUNG:

Die Variable `x` muss größer sein als `0`.

LOG

```
LOG(x)
```

Liefert den Logarithmus von `x` zur Basis `10`.

Die Funktion ist für jeden Fließkommatyp überladen. Der Typ von `x` wird als Rückgabetyp verwendet.

EINSCHRÄNKUNG:
Die Variable `x` muss größer sein als `0`.

EXP

```
EXP(x)
```

Liefert `e` hoch `x`.

Die Funktion ist für jeden Fließkommatyp überladen. Der Typ von `x` wird als Rückgabetyp verwendet.

SIN

```
SIN(x)
```

Liefert den Sinus von `x`, wobei `x` in Radiant zu erwarten ist.

Die Funktion ist für jeden Fließkommatyp überladen. Der Typ von `x` wird als Rückgabetyp verwendet.

Siehe auch: Die gSin-Funktion (Kapitel Trigonometrie [▶ 87]).

COS

```
COS(x)
```

Liefert den Cosinus von `x`, wobei `x` in Radiant zu erwarten ist.

Die Funktion ist für jeden Fließkommatyp überladen. Der Typ von `x` wird als Rückgabetyp verwendet.

Siehe auch: Die Funktion gCos (Kapitel Trigonometrie [▶ 87]).

TAN

```
TAN(x)
```

Liefert den Tangens von `x`, wobei `x` in Radiant zu erwarten ist.

Die Funktion ist für jeden Fließkommatyp überladen. Der Typ von `x` wird als Rückgabetyp verwendet.

Siehe auch: Die Funktion gTan (Kapitel Trigonometrie [▶ 87]).

ASIN

```
ASIN(x)
```

Liefert den Arkussinus von `x` innerhalb des Intervalls `[-PI/2,PI/2]` Radianten.

Die Funktion ist für jeden Fließkommatyp überladen. Der Typ von `x` wird als Rückgabetyp verwendet.

Siehe auch: Die gASin-Funktion (Kapitel Trigonometrie [▶ 87]).

EINSCHRÄNKUNG:
Die Variable `x` muss innerhalb des Intervalls `[-1,1]` liegen.

ACOS

```
ACOS(x)
```

Liefert den Arkuscosinus von `x` innerhalb des Intervalls `[0,PI]` Radianten.

Die Funktion ist für jeden Fließkommatyp überladen. Der Typ von `x` wird als Rückgabetyp verwendet.

Siehe auch: Die gACos-Funktion (Kapitel Trigonometrie [▶ 87]).

EINSCHRÄNKUNG:
Die Variable `x` muss innerhalb des Intervalls `[-1,1]` liegen.

ATAN

```
ATAN(x)
```

Liefert den Arkustangens von `x` innerhalb des Intervalls `[-PI/2,PI/2]` Radianten.

Die Funktion ist für jeden Fließkommatyp überladen. Der Typ von `x` wird als Rückgabetyp verwendet.

Siehe auch: Die gATan-Funktion (Kapitel Trigonometrie [▶ 87])..

ATAN2

```
ATAN2(y,x)
```

Liefert den Arkustangens von `y/x` innerhalb des Intervalls `[-PI,PI]` Radianten.

Die Funktion ist für jeden Fließkommatyp überladen. Der kleinste gemeinsame Typ von `x` und `y` wird als Rückgabetyp verwendet.

Siehe auch: Die Funktion gATan2 (Kapitel Trigonometrie [▶ 87]).

ADD

```
ADD(x1, x2, …)
```

Liefert die Summe aller Parameter. Die `ADD`-Funktion kann eine beliebige Anzahl von Parametern haben, muss aber mindestens einen haben.

Die Funktion ist für jeden Ganzzahl- und Fließkommatyp überladen. Der kleinste gemeinsame Typ aller Parameter wird als Rückgabetyp verwendet.

MUL

```
MUL(x1, x2, …)
```

Liefert das Produkt aller Parameter. Die `MUL`-Funktion kann eine beliebige Anzahl von Parametern haben, muss aber mindestens einen haben. Alternativ kann der Infix-Operator '`*`' verwendet werden.

Die Funktion ist für jeden Ganzzahl- und Fließkommatyp überladen. Der kleinste gemeinsame Typ aller Parameter wird als Rückgabetyp verwendet.

SUB

```
SUB(x,y)
```

Liefert die Differenz `x-y`. Alternativ kann der Infix-Operator '`-`' verwendet werden.

Die Funktion ist für jeden Ganzzahl- und Fließkommatyp überladen. Der kleinste gemeinsame Typ von `x` und `y` wird als Rückgabetyp verwendet.

DIV

```
DIV(x,y)
```

Liefert den Quotienten `x/y`. Alternativ kann der Infix-Operator '`/`' verwendet werden.

Die Funktion ist für jeden Ganzzahl- und Fließkommatyp überladen. Der kleinste gemeinsame Typ von `x` und `y` wird als Rückgabetyp verwendet. Ist der Rückgabetyp ein Integer-Typ, wird das Ergebnis gegen Null abgeschnitten.

EINSCHRÄNKUNG:
Die Variable `y` darf nicht Null sein.

MOD

```
MOD(x,y)
```

Liefert den Rest der ganzzahligen Division `x/y`. Alternativ kann der Infix-Operator '`MOD`' verwendet werden.

Die Funktion ist für jeden Integer-Typ überladen. Der kleinste gemeinsame Typ von `x` und `y` wird als Rückgabetyp verwendet. Das Ergebnis kann auch negativ sein. Die Gleichung `x = MUL(DIV(x,y),y) + MOD(x,y)` gilt.

EINSCHRÄNKUNG:
Die Variable `y` darf nicht Null sein.

EXPT

```
EXPT(x,y)
```

Liefert `x` hoch `y`.

Die Funktion ist so überladen, dass `x` einen Fließkommatyp und `y` einen Fließkommatyp oder einen Integer-Typ hat. Der Typ von `x` wird als Rückgabetyp verwendet, d. h. zurückgegeben wird ein `Real` oder ein `LReal`-Fließkommatyp. Alternativ kann der Infix-Operator '`**`' verwendet werden.

EINSCHRÄNKUNG:
Wenn `x` negativ ist, dann muss `y` ein Integer sein.

EINSCHRÄNKUNG:
Wenn `x` gleich Null ist, muss `y` größer als Null sein.

4.5.8.3 Verschiebung und Rotation

SHL

```
SHL(x,y)
```

Liefert den um `y` Bit nach links verschobenen Bitstring `x`. Null-Bits werden auf der rechten Seite eingefügt. Es wird angenommen, dass das niederwertigste Bit ganz rechts ist.

Die Funktion ist überladen für jeden Bitstring-Typ für `x` und jeden Integer-Typ für `y`. Der Typ von `x` wird als Rückgabetyp verwendet.

EINSCHRÄNKUNG:
Die Variable `y` darf nicht negativ sein.

SHR

```
SHR(x,y)
```

Liefert den um `y` Bit nach rechts verschobenen Bitstring `x`. Null-Bits werden auf der linken Seite eingefügt. Es wird angenommen, dass das niederwertigste Bit ganz rechts ist.

Die Funktion ist überladen für jeden Bitstring-Typ für `x` und für jeden Integer-Typ für `y`. Der Typ von `x` wird als Rückgabetyp verwendet.

EINSCHRÄNKUNG:
Die Variable `y` darf nicht negativ sein.

ROL

```
ROL(x,y)
```

Liefert den um `y` Bit nach links gedrehten Bitstring `x`. Bits, die auf der linken Seite herausgeschoben werden, werden auf der rechten Seite eingefügt. Es wird angenommen, dass das niederwertigste Bit ganz rechts ist.

Die Funktion ist überladen für jeden Bitstring-Typ für `x` und für jeden Integer-Typ für `y`. Der Typ von `x` wird als Rückgabetyp verwendet.

EINSCHRÄNKUNG:
Die Variable `y` darf nicht negativ sein.

ROR

```
ROR(x,y)
```

Liefert den um `y` Bit nach rechts gedrehten Bitstring `x`. Bits, die auf der rechten Seite herausgeschoben werden, werden auf der linken Seite eingefügt. Es wird angenommen, dass das niederwertigste Bit ganz rechts ist.

Die Funktion ist überladen für jeden Bitstring-Typ für `x` und für jeden Integer-Typ für `y`. Der Typ von `x` wird als Rückgabetyp verwendet.

EINSCHRÄNKUNG:
Die Variable `y` darf nicht negativ sein.

4.5.8.4 Logische Operationen

AND

```
AND(x1, x2, …)
```

Liefert das bitweise logische Und aller Parameter. Das Bit `i` wird im Ergebnis gesetzt, wenn das Bit `i` in allen Parametern gesetzt ist. Die Funktion `AND` kann eine beliebige Anzahl von Parametern haben, muss aber mindestens einen haben.

Die Funktion ist für jeden Bitstring-Typ überladen. Als Rückgabetyp wird der kleinste gemeinsame Bitstring-Typ verwendet.

OR

```
OR(x1, x2, …)
```

Liefert das bitweise logische Oder aller Parameter. Das Bit `i` wird im Ergebnis gesetzt, wenn das Bit `i` in mindestens einem aller Parameter gesetzt ist. Die Funktion `OR` kann eine beliebige Anzahl von Parametern haben, muss aber mindestens einen haben.

Die Funktion ist für jeden Bitstring-Typ überladen. Als Rückgabetyp wird der kleinste gemeinsame Bitstring-Typ verwendet.

XOR

```
XOR(x1, x2, …)
```

Liefert das bitweise logische Exklusiv-Oder aller Parameter. Das Bit `i` wird im Ergebnis gesetzt, wenn das Bit `i` in einer ungeraden Anzahl von allen Parametern gesetzt ist. Die Funktion `XOR` kann eine beliebige Anzahl von Parametern haben, muss aber mindestens einen haben.

Die Funktion ist für jeden Bitstring-Typ überladen. Als Rückgabetyp wird der kleinste gemeinsame Bitstring-Typ verwendet.

NOT

```
NOT(x)
```

Liefert das bitweise Komplement von `x`. Das Bit `i` wird im Ergebnis gesetzt, wenn das Bit `i` in `x` nicht gesetzt ist.

Die Funktion ist für jeden Bitstring-Typ überladen. Der Typ von `x` wird als Rückgabetyp verwendet.

4.5.8.5 Auswahl (bedingte Ausdrücke)

SEL

```
SEL(cond,x1,x2)
```

Liefert `x1`, wenn `cond` falsch ist, und ansonsten `x2`.

MUX

```
MUX(select, x0, x1, …, xN)
```

Liefert `x<select>`. Wenn `select` gleich `0` ist, wird `x0` zurückgegeben. Wenn `select 1` ist, wird `x1` zurückgegeben und so weiter. Die Funktion `MUX` kann eine beliebige Anzahl von Parametern haben, muss aber mindestens zwei haben.

Die Funktion ist überladen für jeden Typ für `x<i>` und für jeden Integer für `select`. Der kleinste gemeinsame Typ von `x<i>` wird als Rückgabetyp verwendet.

EINSCHRÄNKUNG:

Die Variable `select` muss innerhalb des Intervalls `[0,N]` liegen. Andernfalls wird zur Laufzeit ein Out-of-Bounds-Fehler ausgegeben.

4.5.8.6 Min, Max und Grenzwert

MAX

```
MAX(x1, x2, …)
```

Liefert das Maximum aller Parameter.

Die Funktion ist für jeden Ganzzahl- und Fließkommatyp überladen. Der kleinste gemeinsame Typ aller Parameter wird als Rückgabetyp verwendet.

MIN

```
MIN(x1, x2, …)
```

Liefert das Minimum aller Parameter.

Die Funktion ist für jeden Ganzzahl- und Fließkommatyp überladen. Der kleinste gemeinsame Typ aller Parameter wird als Rückgabetyp verwendet.

LIMIT

```
LIMIT(min,in,max)
```

Liefert `in`, wenn es in dem Intervall `[min,max]` liegt. Andernfalls wird die verletzte Grenze (`min` oder `max`) zurückgegeben.

Die Funktion ist für jeden Ganzzahl- und Fließkommatyp überladen. Der kleinste gemeinsame Typ aller Parameter wird als Rückgabetyp verwendet.

EINSCHRÄNKUNG:
Die `min`-Grenze muss kleiner sein als die `max`-Grenze.

4.5.8.7 Vergleich

GT

```
GT(x,y)
```

Liefert `TRUE`, wenn `x` größer ist als `y`. Der kleinste gemeinsame Typ von `x` und `y` wird für den Vergleich verwendet.

Die Funktion ist für alle Ganzzahl- und Fließkommatypen überladen. Der Rückgabetyp ist `BOOL`.

GE

```
GE(x,y)
```

Liefert `TRUE`, wenn `x` nicht kleiner als `y` ist. Der kleinste gemeinsame Typ von `x` und `y` wird für den Vergleich verwendet.

Die Funktion ist für alle Ganzzahl- und Fließkommatypen überladen. Der Rückgabetyp ist `BOOL`.

EQ

```
EQ(x,y)
```

Liefert `TRUE`, wenn `x` und `y` gleich sind. Der kleinste gemeinsame Typ von `x` und `y` wird für den Vergleich verwendet.

Die Funktion ist für alle Ganzzahl- und Fließkommatypen überladen. Der Rückgabetyp ist `BOOL`.

LE

```
LE(x,y)
```

Liefert `TRUE`, wenn `x` nicht größer ist als `y`. Der kleinste gemeinsame Typ von `x` und `y` wird für den Vergleich verwendet.

Die Funktion ist für alle Ganzzahl- und Fließkommatypen überladen. Der Rückgabetyp ist `BOOL`.

LT

```
LT(x,y)
```

Liefert `TRUE`, wenn `x` kleiner als `y` ist. Der kleinste gemeinsame Typ von `x` und `y` wird für den Vergleich verwendet.

Die Funktion ist für alle Ganzzahl- und Fließkommatypen überladen. Der Rückgabetyp ist `BOOL`.

NE

NE(x,y)

Liefert TRUE, wenn x und y nicht gleich sind. Der kleinste gemeinsame Typ von x und y wird für den Vergleich verwendet.

Die Funktion ist für alle Ganzzahl- und Fließkommatypen überladen. Der Rückgabetyp ist BOOL.

4.5.9 R-Parameter

Rechenparameter

Bei den Rechenparametern (kurz R-Parameter) handelt es sich um Interpreter-Variablen, die mit einem Ausdruck der Form "R<n>" genannt werden. Da es sich bei 'n' um eine Ganzzahl im Wertebereich 0..999 handelt, stehen insgesamt 1000 R-Parameter zur Verfügung. Davon sind die ersten 900 R0..R899 Werte lokale Variablen des NC-Kanals. Sie sind nur durch den Interpreter des Kanals zugreifbar. Die R-Parameter R900..R999 sind global angelegt. Sie existieren nur einmal pro NC und die Zugriffe aller Kanäle erfolgen auf denselben Speicher. Dadurch ist ein Datenaustausch (z.B. für eine Teileverfolgung, Kollisionsvermeidung etc.) über die Kanalgrenze hinweg möglich.

Zuweisung eines Wertes zu einem R-Parameter

Die Zuweisung eines Wertes zu einem R-Parameter ist nur innerhalb von Strukturiertem Text möglich. Es gibt zwei Möglichkeiten, einem R-Parameter einen Wert zuzuweisen. Der Wert kann direkt zugewiesen werden oder es kann die Funktion rSet verwendet werden. Die Funktion rSet eignet sich, wenn der Index des zuzuweisenden R-Parameters erst zur Laufzeit ermittelt werden soll.

Strukturierter Text: Direkte Zuweisung eines R-Parameterwerts

```
R<n> := LReal;
```

Beispiel

```
!R1 := 7;
```

Strukturierter Text: Zuweisung eines R-Parameterwertes mit der Funktion "rSet"

```
rSet(index := LINT, value := LREAL)
```

Beispiel

```
!rSet(1, 7);
```

Lesen eines R-Parameterwerts

Es gibt zwei Möglichkeiten, einen R-Parameter zu lesen. Ein R-Parameter kann direkt in G-Code verwendet werden oder er kann mit der Funktion rGet innerhalb des strukturierten Textes extrahiert werden. Die Funktion rGet extrahiert einen R-Parameterwert entsprechend seinem Index.

Strukturierter Text: Lesen eines R-Parameterwertes mit der "rGet"-Funktion

```
rGet(index := LINT) : LREAL
```

G-Code Beispiel: Direktes Extrahieren eines R-Parameterwerts

```
!R1 := 7;
N10 G01 X=R1 F6000
```

G-Code Beispiel: Extrahieren eines R-Parameterwertes mit der "rGet"-Funktion

```
!R1 := 7;
N10 G01 X={rGet(1)} F6000
```

Beispiel: Zuordnen und Extrahieren

```
{
VAR
    valueR1 : LREAL;
END_VAR

rSet(1, 7);
valueR1 := rGet(1);
```

```
R2 := 10;
R3 := R1 + R2;

!N10 G01 X=R1 Y0 Z=R2 F6000
!N20 G01 X={rGet(3)}

MSG(toString('R1 = ', valueR1, ',R2 = ', rGet(2), ', R3 = ', R3));
}
M02
```

Output:

```
R1 = 7.000000, R2 = 10.000000, R3 = 17.000000
```

R-Parameter in Unterprogrammen (Funktionen)

Innerhalb einem Unterprogramm (Funktion) muss ein R-Parameter über eine VAR_EXTERNAL-Deklaration deklariert werden.
Beispiel:

```
{
FUNCTION myFunction : LREAL
VAR_EXTERNAL
    R45: LREAL;
END_VAR
}

N10 G01 X=R45 F6000

!END_FUNCTION
```

Voraussetzungen

Entwicklungsumgebung	Zielsystem
TwinCAT V3.1.4024.4 oder 4022.32	PC oder CX (x86 oder x64)

4.5.10 H-, S- und T-Parameter

Die H-, S- und T-Parameter werden verwendet, um bei der Satzausführung Parameter an die SPS zu übertragen.

Parameter	Datentyp	Verwendung	Beispiel
H-Parameter	DINT (32Bit signed)	Hilfsparameter	`N1 G1 X10 Y20 H=1020`
S-Parameter	WORD	Spindel	`N2 G1 X20 Y30 S=30`
T-Parameter	WORD	Werkzeug	`N3 G1 X30 Y40 T4`

- Im Gegensatz zum Classic-Interpreter wirkt beim GST-Interpreter auch der H-Parameter vor der Bewegung, siehe Ausführungsreihenfolge [▶ 38] eines Satzes.
- Einem T-Parameter kann kein R-Parameter zugewiesen werden.
- Beim T-Parameter erfolgt die Zuweisung ohne Zuweisungsoperator (‚=‘).

4.6 CNC Funktionen

4.6.1 Strings und Nachrichten

toString

```
toString(<arg0>, …, <argN>): STRING
```

Konvertiert und verkettet die angegebenen Argumente zu einem String. Dieser String ist auf 255 Zeichen begrenzt, was der Standardlänge eines Strings entspricht. Die `toString`-Funktion verhält sich wie die Druckfunktion, nur dass sie einen formatierten String ausgibt, anstatt zu drucken.

Die `toString`-Funktion ist besonders nützlich, um einen String für die `msg(…)`-Funktion zu formatieren.

msg

```
msg(str:= String[81])
```

Sendet die angegebene Nachricht an die Nachrichtenliste von TwinCAT. Die Nachricht wird vom NC-Kanal synchron verarbeitet. Sie erscheint in der Benutzeroberfläche, wenn alle vorangegangenen NC-Befehle abgeschlossen sind.

Um formatierte Strings zu senden, kann diese Funktion mit der `toString`-Funktion kombiniert werden.

Die Nachricht ist auf 81 Zeichen begrenzt. Längerer Text wird abgeschnitten.

Beispiel:

Die Bahn des folgenden Beispiels ist in der Abbildung "BeispielMsg" dargestellt. Er ist mit den ausgegebenen Nachrichten versehen.

```
{
VAR
    x,y,z: LREAL;
    start: LDT;
END_VAR

!N10 G00 X0 Y0 F300
start := currentLdt();
!N20 G01 X30
msg('N20 completed');
!N30 X60 Y10
frameGet(x=>x,y=>y,z=>z);
msg(toString('Current position: [',x,',',y,',',z,']'));
!N40 X90
sync();
msg(toString('Machining time: ', currentLdt()-start));
}
M02
```

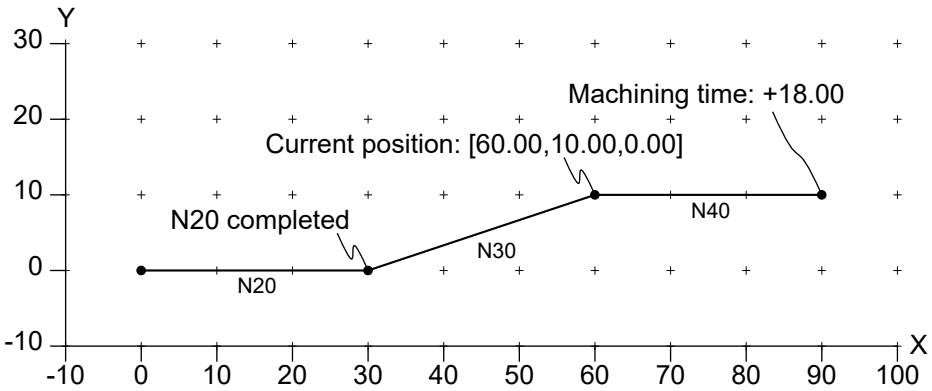


Abbildung "BeispielMsg".

4.6.2 Transformationen

transRotX/Y/Z

```
transRotX(angle := LREAL)

transRotY(angle := LREAL)

transRotZ(angle := LREAL)
```

Drehung um die jeweilige Achse um den angegebenen Winkel in der benutzerdefinierten Winkeleinheit. Die Drehung wird auf den Stapel der Transformationen geschoben. Der Winkelwert wird in der aktuellen Winkeleinheit interpretiert. Siehe Abschnitt Transformationen [▶ 99] für weitere Informationen.

Beispiel:

Die resultierende Bahn des folgenden Beispiels ist in der Abbildung "BeispielTransRotZ" dargestellt.

- `N10` wird so programmiert, dass das PCS (Programmkoordinatensystem) und das MCS (Maschinenkoordinatensystem) gleich sind.
- `N20` wird programmiert, nachdem eine `45`-Grad-Drehung um die `Z`-Achse in `[0,0,0]` auf den Stapel der Transformationen geschoben wurde. Eine weitere Drehung von `45` Grad wird auf den Transformationsstapel geschoben, so dass sich die Drehungen zu `90` Grad summieren.
- Die MCS-Koordinate (Maschinenkoordinatensystem) des Endes des Segments `N30` ist daher `[0,30,0]`.

```
N10 G01 X30 Y0 F6000
!transRotZ(45);
N20 G01 X30 Y0
!transRotZ(45);
N30 G01 X30 Y0
!transPop();
!transPop();
M02
```

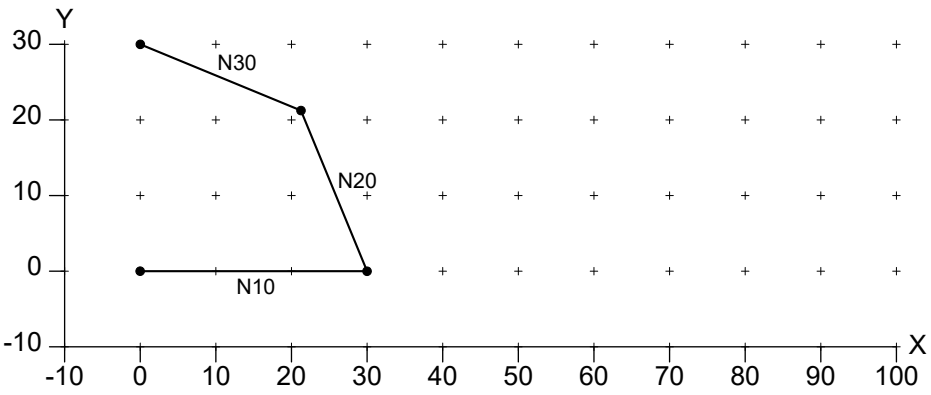


Abbildung "BeispielTransRotZ".

transRotA

```
transRotA(x:=LReal, y:=LReal, z:=LReal, angle:=LReal)
```

Drehen um den Vektor `[x,y,z]` mit dem angegebenen `angle`. Die Drehung wird auf den Stapel der Transformationen geschoben. Der Winkelwert wird in der aktuellen Winkeleinheit interpretiert. Siehe Abschnitt Transformationen [▶ 99] für weitere Informationen.

Der Vektor `[x,y,z]` darf nicht der Null-Vektor sein.

Beispiel:

Die resultierende Bahn des folgenden Beispiels ist in der Abbildung "BeispielTransRotA" dargestellt. Der erste Aufruf von `transRotA` dreht das PCS (Programm-Koordinatensystem) um die positive `Z`-Achse (Rechte-Hand-Regel) um `45` Grad. Der zweite Aufruf dreht sich um die negative `Z`-Achse um den gleichen Winkel, also in die entgegengesetzte Richtung. Die Kombination der beiden Drehungen ist die Identitätstransformation.

```
!transRotA(0,0,1,45);
N10 G01 X30 Y0 F6000
!transRotA(0,0,-1,45);
N20 G01 X30 Y0
!transPop();
!transPop();
M02
```

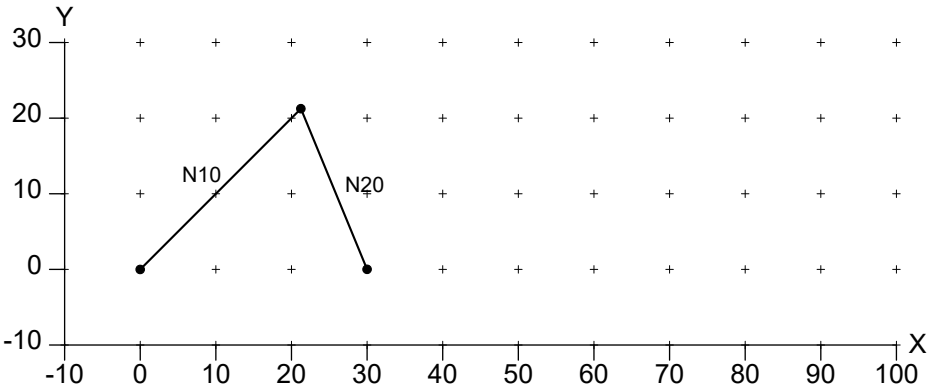


Abbildung "BeispielTransRotA".

transMirrorX/Y/Z

```
transMirrorX()

transMirrorY()

transMirrorZ()
```

Spiegeln in Bezug auf die `X`-Richtung, `Y`-Richtung oder `Z`-Richtung relativ zum Ursprung des aktuellen PCS (Programmkoordinatensystem). Die Transformation wird auf den Stapel der Transformationen geschoben.

Der Aufruf einer Spiegelfunktion schaltet die Ausrichtung des Koordinatensystems von rechtsdrehend nach linksdrehend oder umgekehrt um. Vor allem schaltet dieses Verhalten die Drehrichtung von Kreisen und die Kompensationsrichtung der Werkzeugradiuskorrektur um. Standardmäßig ist das Koordinatensystem rechtsdrehend.

Beispiel:

Die resultierende Bahn des folgenden Beispiels ist in der Abbildung "BeispielTransMirrorX" dargestellt. Das PCS (Programmkoordinatensystem) wird an der `X`-Dimension gespiegelt. Dadurch wird das Koordinatensystem zu einem linksdrehenden System, in dem die Drehrichtung von `G2` (absichtlich) vertauscht ist.

```
N10 G02 X20 Y20 U20 F6000
!transMirrorX();
N20 G02 X-40 Y0 U20
!transPop();
M02
```

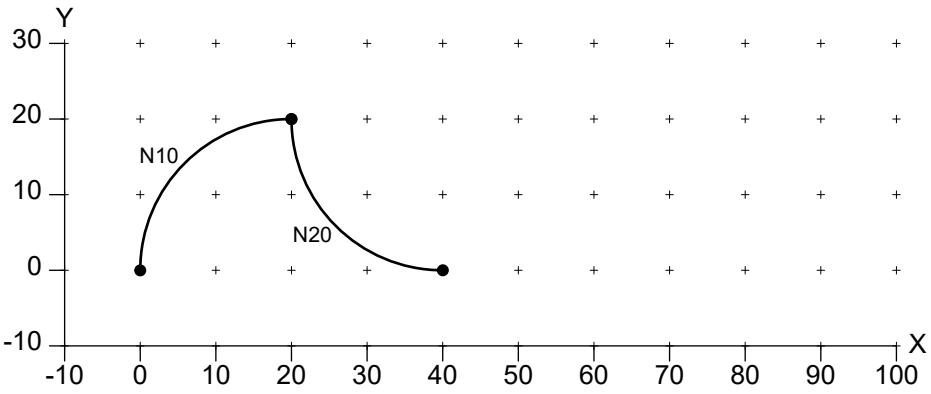


Abbildung "BeispielTransMirrorX".

transScale

```
transScale(factor:= LReal)
```

Skaliert das Koordinatensystem um den `factor` in der `X`-, `Y`- und `Z`-Dimension. Die Transformation wird auf den Stapel der Transformationen geschoben.

Der Faktor muss ungleich Null sein.

Ist der Faktor negativ, so wird das Koordinatensystem effektiv an der `X`-, `Y`- und `Z`-Dimension gespiegelt. Dadurch wird die Ausrichtung des Koordinatensystems vertauscht.

Beispiel:

Die sich ergebende Bahn des folgenden Beispiels ist in der Abbildung "BeispielTransScale" dargestellt. Nach einer Skalierung um den Faktor `2` wird der Endpunkt des Segments `N20` auf `[60,20,0]` abgebildet.

```
N10 G01 X30 Y10 F6000
!transScale(2);
N20 G01 X30 Y10
!transPop();
M02
```

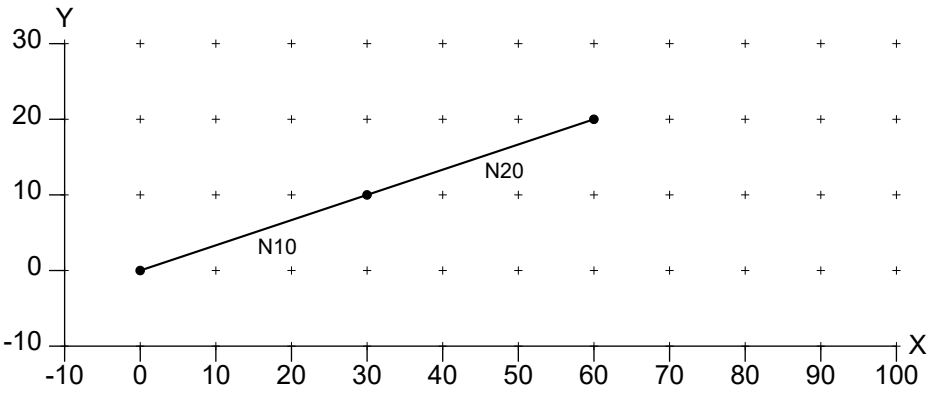


Abbildung "BeispielTransScale".

transScaleAxis

```
transScaleAxis(axisNo := axisIndex, factor := value);
```

Skaliert die ausgewählte Bahnachse (`axisNo`) um den Faktor. Die unterstützten Achsen und Indizes sind:

- X: 0
- Y: 1
- Z: 2

Q-Achsen werden nicht unterstützt.

Eine unterschiedliche Achsenskalierung ist nur für lineare Bewegungen zulässig, nicht für Kreisbewegungen.

Beispiel 1

```
N10 G01 X30 Y10 F6000
!transScaleAxis(axisNo:= 0, factor:=2.0);
!transScaleAxis(axisNo:= 1, factor:=2.0);
!transScaleAxis(axisNo:= 2, factor:=3.0);
N20 G01 X30 Y10
N30 G03 X40 Y10 I5 J0
M02
```

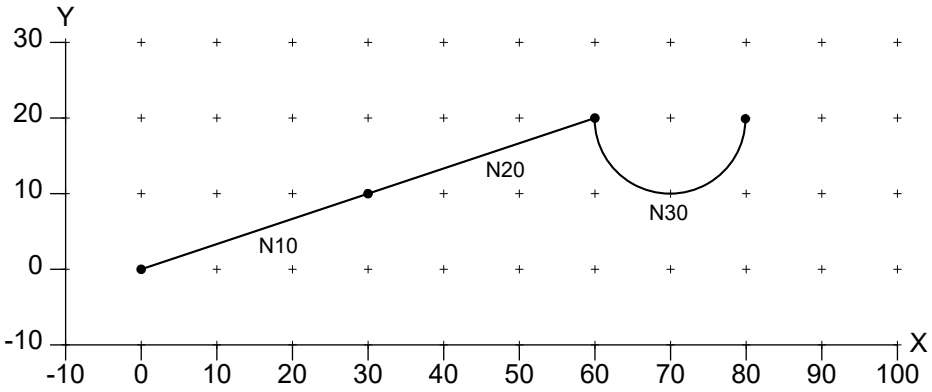


Abbildung "Beispiel 1 TransScaleAxis".

Beispiel 2

```
N10 G01 X20 Y5 F6000
!transScaleAxis(axisNo:= 0, factor:=2.0);
!transScaleAxis(axisNo:= 1, factor:=2.0);
!transScaleAxis(axisNo:= 2, factor:=3.0);
N20 G01 X20 Y5
!transScaleAxis(axisNo:= 0, factor:=2.0);
!transScaleAxis(axisNo:= 1, factor:=3.0);
N30 G01 X20 Y5
M02
```

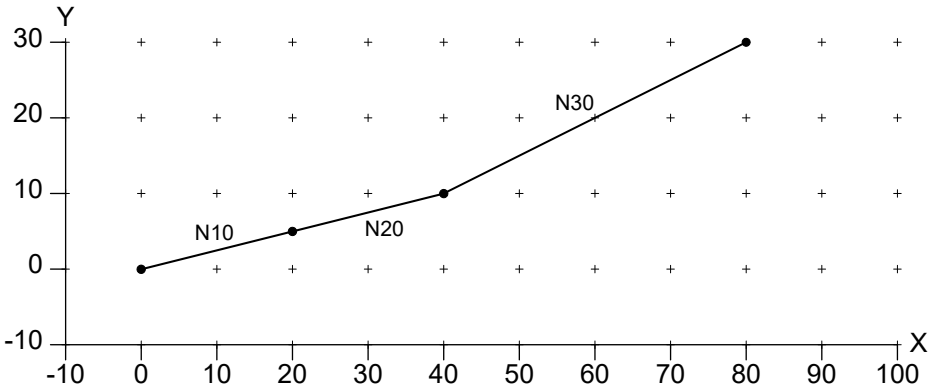


Abbildung "Beispiel 2 TransScaleAxis".

Requirements

Development Environment	Target System
TwinCAT V3.1.4024.20	PC or CX (x86 or x64)

transTranslate

`transTranslate(x:=LReal, y:=LReal, z:=LReal)`

Verschieben um Vektor `[x,y,z]`. Die Translation wird auf den Stapel der Transformationen geschoben.

Beispiel:

Die resultierende Bahn des folgenden Beispiels ist in der Abbildung "BeispielTransTranslate" dargestellt. Nach der Translation um `[40,20,0]` wird der Endpunkt des Segments `N20` auf `[80,20,0]` abgebildet.

```
N10 G01 X20 Y0 F6000
!transTranslate(40,20,0);
N20 G01 X40 Y0
!transPop();
M02
```

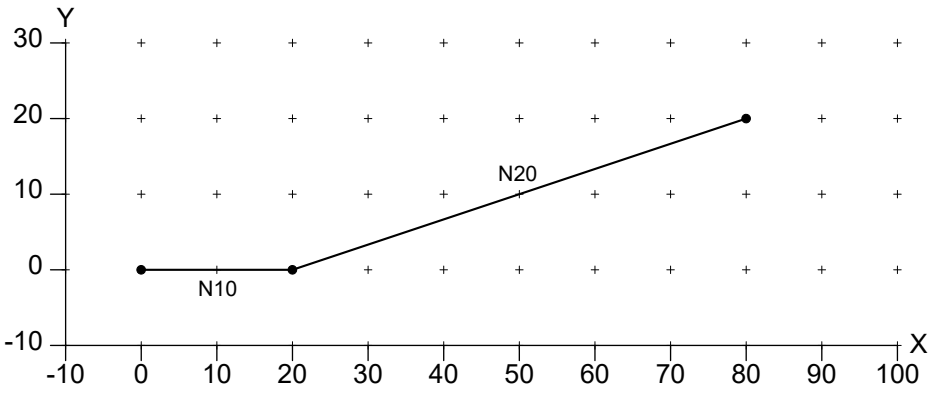


Abbildung "BeispielTransTranslate".

transPop

`transPop()`

Entnimmt eine Transformation aus dem Stapel der Transformationen.

Beispiel:

Die sich ergebende Bahn des folgenden Beispiels ist in der Abbildung "BeispielTransPop" dargestellt. In diesem Beispiel wird die Translation `[0,20,0]` auf den Stapel geschoben, gefolgt von der Translation `[0,10,0]`. Die effektive Translation für `N30` ist dann `[0,30,0]`. Mit dem Aufruf von `transPop` wird die Translation `[0,10,0]` vom Stapel entfernt. Der Endpunkt des Segments `N40` wird also um `[0,20,0]` verschoben. Nach dem Entfernen der letzten Translation vom Stapel wird der Endpunkt des Segments `N50` überhaupt nicht verschoben.

```
N10 G01 X10 Y0 F6000
!transTranslate(0,20,0);
N20 G01 X30 Y0
!transTranslate(0,10,0);
N30 G01 X50 Y0
!transPop();
N40 G01 X70 Y0
!transPop();
N50 G01 X90 Y0
M02
```

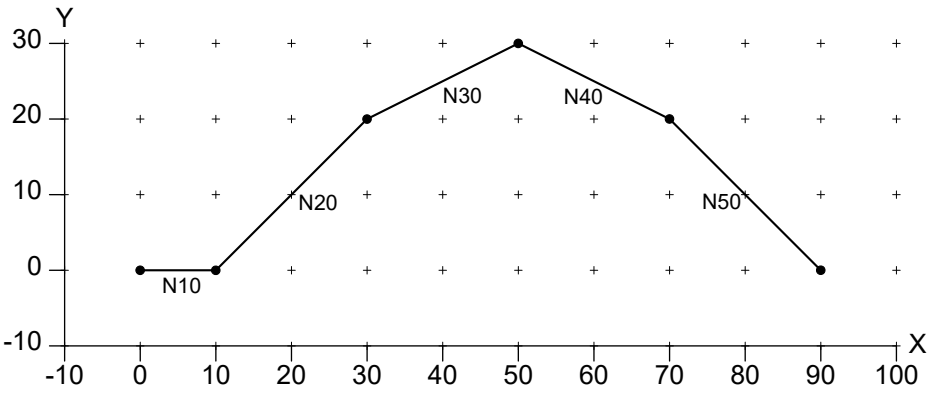


Abbildung "BeispielTransPop".

transDepth

```
transDepth(): UInt
```

Ergibt die Tiefe des Transformationsstapels, d. h. die Anzahl der aktiven Transformationen. Siehe transRestore(...), Kapitel Transformationen [▶ 99], für weitere Einzelheiten.

transRestore

```
transRestore(depth:= UInt)
```

Reduziert den Stapel der Transformationen auf die angegebene Tiefe. Dieser Befehl wird normalerweise in Verbindung mit `transDepth()` verwendet, um einen früheren Zustand des Stapels wiederherzustellen.

Die aktuelle Tiefe des Stapels darf nicht kleiner als die angegebene Tiefe sein.

Beispiel:

Die resultierende Bahn des folgenden Beispiels ist in der Abbildung "BeispielTransDepthTransRestore" dargestellt. Eine Translation nach `[40,10,0]` wird zunächst auf den Transformationsstapel geschoben. Die resultierende Tiefe wird in der Variablen `savedDepth` gespeichert. Der folgende Code führt wiederholt eine lineare Bewegung zu `X20 Y0` und eine Drehung um `45` Grad durch. Dieser resultierende Pfad ist eine Hälfte eines Achtecks, bestehend aus den Segmenten `N10` bis `N50`. Wenn `N50` verarbeitet wird, enthält der Transformationsstapel die anfängliche Translation und `4` Drehungen um `45` Grad. Der Aufruf von `transRestore(savedDepth)` stellt die Stapeltiefe von `1` wieder her, indem er alle Rotationen entfernt. Daher wird nur die Translation auf `N60` angewendet.

```
!VAR savedDepth : UINT; END_VAR
!transTranslate(40,10,0);
!savedDepth := transDepth();

N10 G01 X20 Y0 F6000
!transRotZ(45);
N20 G01 X20 Y0
!transRotZ(45);
N30 G01 X20 Y0
!transRotZ(45);
N40 G01 X20 Y0
!transRotZ(45);
N50 G01 X20 Y0
!transRestore(savedDepth);
N60 G01 X10 Y0
M02
```

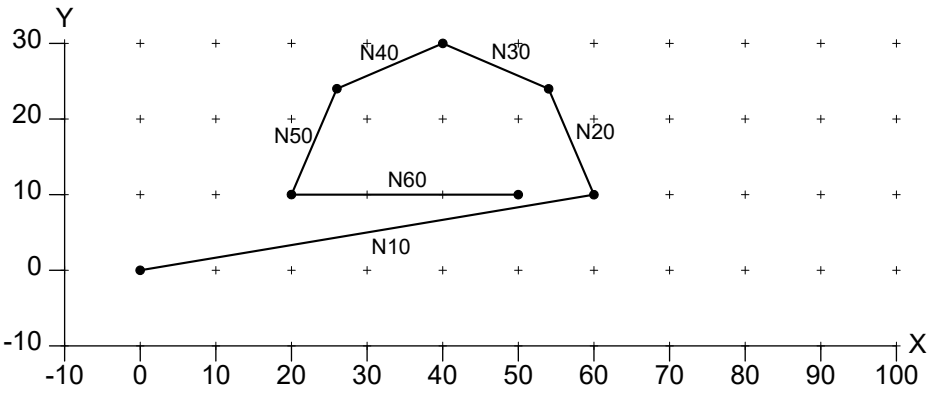


Abbildung "BeispielTransDepthTransRestore".

4.6.3 Kreisbewegung

moveCircle3d

```
moveCircle3d(cx:=LREAL, cy:=LREAL, cz:=LREAL, nx:=LREAL, ny:=LREAL, nz:=LREAL,
angle:=LREAL, height:=LREAL)
```

Kreisbewegung durch Rotation um den Mittelpunkt `cx,cy,cz` und den Normalenvektor `nx,ny,nz` um den angegebenen `angle`. Wenn `height` ungleich Null ist, wird eine Helix beschrieben. Wenn `angle` größer als ein Vollkreis ist, wird ein Multiturn-Kreis oder eine Multiturn-Helix beschrieben. Die Drehung wird gemäß der Rechte-Hand-Regel durchgeführt. Die Verwendung eines negativen Winkels oder das Umdrehen der Normalen kehrt die Drehrichtung um. Der Winkelwert wird in der aktuellen Winkeleinheit interpretiert. Die Parameter `x`, `y`, `z`, `cx`, `cy`, `cz` werden in der aktuellen Längeneinheit interpretiert.

Der Radius muss ungleich Null sein.

Beispiel:

Die resultierende Bahn des folgenden Beispiels ist in Abbildung "BeispielMoveCircle3D" dargestellt. Der Aufruf von `moveCircle3D` beschreibt eine spiralförmige Bewegung. Sie beginnt an dem aktuellen Punkt `[40,10,0]`. Die Mittelachse der Helix wird durch den Punkt `[30,10,0]` und die Richtung `[gSin(22.5),0,gCos(22.5)]` definiert. Gegenüber der Normalen der Arbeitsebene `[0,0,1]` ist die Achse um `22.5` Grad in `X`-Richtung geneigt. Der Winkel von `720+90` Grad beschreibt eine Multiturn-Helix. Sie weist eine Höhe von `30` in Bezug auf die Mittelachse auf. Der Endpunkt der Helix ist nicht explizit programmiert, um Redundanz zu vermeiden. Wenn der Benutzer diese Koordinaten benötigt, können sie, wie gezeigt, mit der Funktion `frameGet(…)` abgerufen werden. Die ungefähren Koordinaten sind in der Abbildung "BeispielMoveCircle3D" dargestellt.

```
{
VAR
    x,y,z: LREAL;
END_VAR

!N10 G01 X40 Y10 F6000
moveCircle3D(cx:=30, cy:=10, cz:=0, nx:=gSin(22.5), ny:=0, nz:=gCos(22.5), angle:=720+90, height:=30
);
frameGet(x=>x,y=>y,z=>z);
}
M02
```

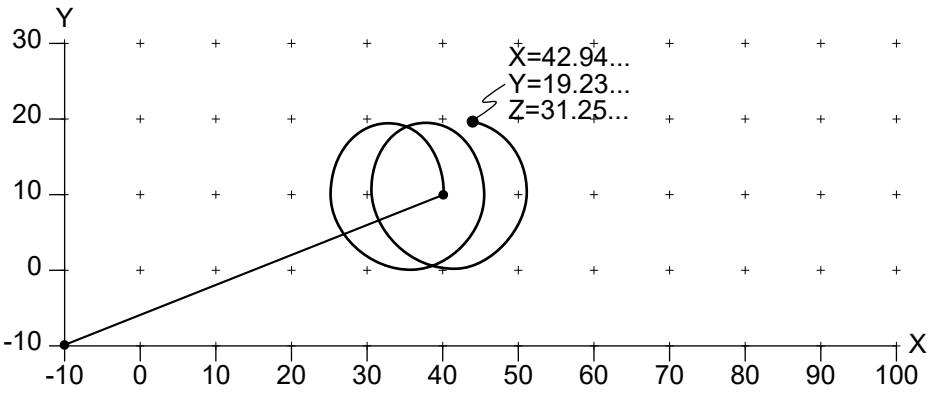


Abbildung "BeispielMoveCircle3D".

4.6.4 Mittelpunktskorrektur

centerpointCorrectionSet

```
centerpointCorrectionSet(on:= bool)
```

Aktiviert die Mittelpunktskorrektur für Kreise. Die Mittelpunktskorrektur wird für Kreise verwendet, die mit der Mittelpunktprogrammierung definiert wurden, (siehe G2 und G3 [▶ 40]). Aufgrund von Ungenauigkeiten (z.B. Rundungsfehler des CAD-Programms) kann der Radius von Anfangs- und Endpunkt in Bezug auf den Mittelpunkt abweichen. Wenn die Mittelpunktskorrektur aktiv ist, wird der Mittelpunkt so verschoben, dass der Start- und Endradius ihrem früheren Mittelwert entsprechen.

Eine Grenze für die Mittelpunktskorrektur kann mit `centerpointCorrectionLimitSet(…)` konfiguriert werden. Wird diese Grenze überschritten, wird ein Laufzeitfehler gemeldet.

centerpointCorrectionLimitSet

```
centerpointCorrectionLimitSet(limit:= LREAL)
```

Legt die Genauigkeitsgrenze für den Mittelpunkt von Kreisen fest. Wenn die angegebene Grenze überschritten wird, wird ein Laufzeitfehler gemeldet. Der Standardwert ist 0,1.

4.6.5 Werkzeuge

Verwendung von Werkzeugverschiebung und Rotation

Wird die kartesische Werkzeugverschiebung in Verbindung mit der Rotation [▶ 72] verwendet, so wird nur dann richtig kompensiert, wenn das Aggregat (Werkzeugträger) ebenfalls um den gleichen Winkel rotiert wird.

toolParamSet

```
toolParamSet(tidx:= USINT, col:= USINT, val:= LREAL)
```

Parameter des Werkzeugs `tidx` (1..255) auf `val` einstellen. Der Parameter ist durch `col` (0..15) gekennzeichnet.

COL	DESCRIPTION
0	Werkzeugnummer Um dem Werkzeug eine Nummer zu geben. Zu T-Parameter im zyklischen Kanalinterface geschrieben.
1	Werkzeugtyp (`10`: Bohrer, `20`: Fräser) Der Bohrer ist vom Typ `10`. Der Fräser ist vom Typ `20`.

2	Länge Beschreibt beispielsweise die Länge des Bohrers.
3	–
4	Radius
5	Länge (addiert zum Längenwert der Spalte 2) Beschreibt beispielsweise den Verschleiß des Bohrers. Dabei muss der Verschleiß negativ angegeben werden, da er zur Länge hinzuaddiert wird.
6	–
7	Radius (addiert zum Radiuswert der Spalte 4)
8	x-shift Kartesische Werkzeugverschiebung in x-Richtung
9	y-shift Kartesische Werkzeugverschiebung in y-Richtung
10	z-shift Kartesische Werkzeugverschiebung in z-Richtung
11	–
12	–
13	Typ 20 (Fräser): zur freien Verwendung durch den Benutzer
14	Typ 20 (Fräser): zur freien Verwendung durch den Benutzer
15	Typ 20 (Fräser): zur freien Verwendung durch den Benutzer

toolParam

```
toolParam(tidx:= USINT, col:= USINT): LREAL
```

Liefert den angegebenen Werkzeugparameter.

toolSet

```
toolSet(index:= USINT, nr:= INT, tooltype:= ToolType, length:= LREAL, radius:=
LREAL, lengthAdd:= LREAL, radiusAdd:= LREAL, offsetX:= LREAL, offsetY:= LREAL,
offsetZ:= LREAL)
```

Setzt alle Werkzeugparameter. Der Index wird in D-Wörtern [▶ 35] verwendet, um auf das Werkzeug zu verweisen. Er muss im Bereich von 1 bis 255 liegen. Der Parameter nr hat nur informativen Charakter. In der Regel handelt es sich dabei um eine unternehmensinterne Nummer zur Kennzeichnung eines bestimmten Werkzeugs. Der Parameter tooltype gibt die Art des Werkzeugs an, z. B. ein Bohrer. Die übrigen Parameter sind Abmessungen, die in der Abbildung "ToolSetDimensions" dargestellt sind. Wird die Werkzeugausrichtung in negativer Richtung geändert (siehe P-Wort [▶ 46]), wird der Wert length+lengthAdd implizit negiert. Die Parameter length, radius, lengthAdd, radiusAdd, offsetX, offsetY und offsetZ werden in der aktuellen Längeneinheit interpretiert.

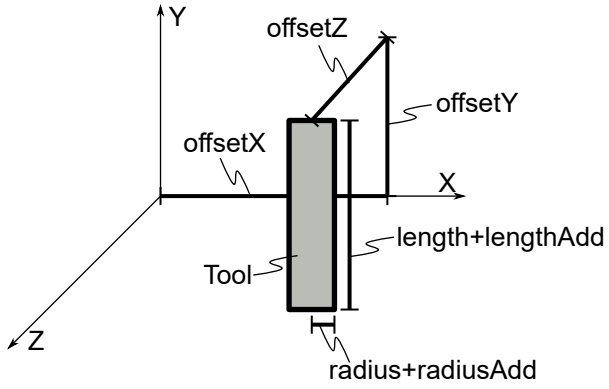


Abbildung "ToolSetDimensions".

Beispiel:

Das Beispiel definiert das Werkzeug `1` als Bohrer mit der Gesamtlänge `48.5` und das Werkzeug `2` als Fräser mit einer Länge von `30` und einem Durchmesser von `5`.

```
!toolSet(index:=1, nr:=4711, tooltype:=tooltypeDrill, length:=50, lengthAdd:=-1.5);
!toolSet(index:=2, nr:=10783, tooltype:=tooltypeMill, length:=30, radius:=2.5);
```

toolType

Aufzählung der Werkzeugtypen.

```
tooltypeDrill
tooltypeMill
```

tooltypeDrill: Wählt einen Bohrer als Werkzeug aus.
tooltypeMill: Wählt einen Fräser als Werkzeug aus.

4.6.6 Synchronisation

sync

`sync()`

Synchronisiert den Interpreter mit dem zugehörigen NC-Kanal. Der `sync()`-Befehl blockiert, bis alle anstehenden NC-Befehle abgearbeitet sind, d.h. bis die Job-Queue des NC-Kanals leer ist. Dieser Befehl ersetzt den früheren `@714`-Befehl. Häufig wird der `sync()`-Befehl mit einer vorangehenden `M`-Funktion vom Typ Handshake kombiniert. Dann blockiert der `sync()`-Befehl, bis die `M`-Funktion von der SPS quittiert wird.

wait

`wait()`

Wartet auf ein `GoAhead`-Signal von der SPS. Der `wait()`-Befehl blockiert, bis dieses Signal empfangen wird. Dieser Befehl ersetzt den früheren `@717`-Befehl. Im Vergleich zu einer Kombination aus einer `M`-Funktion und `sync()` führt diese Art der Synchronisation nicht zu einer leeren Job-Queue. Insbesondere zwingt eine leere Queue den Rechner zum Anhalten.

Das `GoAhead`-Signal kann von der SPS gesendet werden, bevor die zugehörige `wait()`-Funktion aufgerufen wird. In diesem Fall wird die `wait()`-Funktion nicht blockiert.

4.6.7 Abfrage der Achsen

queryAxes

`queryAxes()`

Setzen Sie die MCS-Koordinaten (Maschinenkoordinatensystem) des Interpreters auf die Ist-Koordinaten der physikalischen Achsen. Die MCS-Koordinaten (Maschinenkoordinatensystem) werden automatisch in PCS-Koordinaten (Bahnkoordinatensystem) übertragen, die dem Programmierer angezeigt werden. Sie können auch über `frameGet(…)` abgerufen werden. Eine Kombination aus `sync()` und `queryAxes()` ersetzt den früheren `@716`-Befehl.

- Der `queryAxes()`-Befehl berücksichtigt sowohl die Bahnachsen (`X`, `Y`, `Z`), als auch die Hilfsachsen (`Q1..Q5`).

Dem `queryAxes()`-Befehl sollte `sync()` vorangestellt werden, um unerwartetes Verhalten zu vermeiden.

Beispiel:

Die resultierende Bahn des folgenden Beispiels ist in der Abbildung "BeispielQueryAxes" dargestellt. Das Beispiel geht davon aus, dass `M20` eine `M`-Funktion vom Typ "handshake after" ist. Es wird angenommen, dass die SPS

- auf `M20` wartet,
- die `Y`-Achse auf `20` bewegt,
- den Abschluss der Bewegung abwartet,
- `M20` quittiert.

Der Interpreter sendet das Zeilensegment `N10` an den NC-Kanal, gefolgt von der `M`-Funktion `M20`. Dann wird der Aufruf von `sync()` blockiert. Der NC-Kanal meldet die `M`-Funktion an die SPS, nachdem das Zeilensegment `N10` abgearbeitet wurde. Dann fährt die SPS das Werkzeug vom Ende des Segments `N10` an den Anfang des Segments `N20` und quittiert `M20`. Der Interpreter nimmt den Betrieb wieder auf und ruft `queryAxes()` auf, der den internen "aktuellen Punkt" auf den Endpunkt des Segments `N10'` setzt. Daher sendet der letzte Satz das Zeilensegment `N20` an den NC-Kanal.

```
N00
N10 G01 X40 M20 F6000
!sync();
!queryAxes();
N20 G01 X80
M02
```

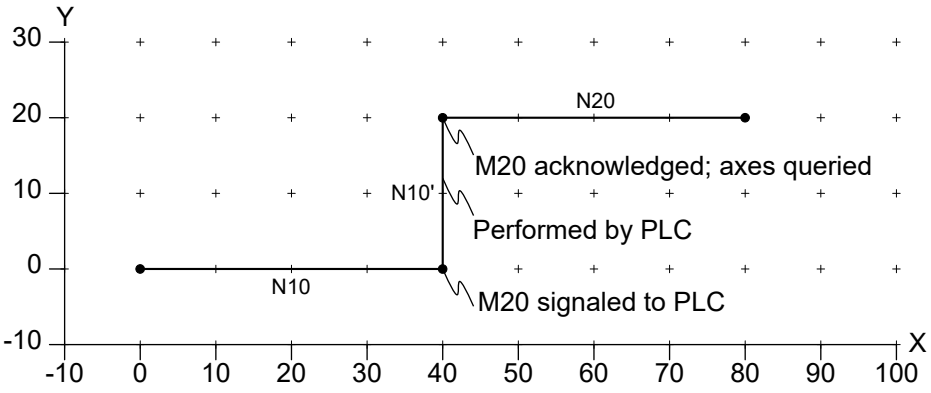


Abbildung "BeispielQueryAxes".

4.6.8 Aktueller Punkt

frameGet

```
frameGet(x:=LREAL, y:=LREAL, z:=LREAL, a:=LREAL, b:=LREAL, c:=LREAL)
```

Speichert den aktuellen Rahmen des PCS (Programmkoordinatensystem) in `x`, `y`, `z` und `a`, `b`, `c`.

Beispiel:

Die Ausgabe des folgenden Beispiels ist unten dargestellt. Der `G`-Code im Beispiel führt eine lineare Bewegung zum PCS-Punkt (Programmkoordinatensystem) `[10,20,30]` aus. Diese Koordinaten werden dann in `curX`, `curY`, `curZ` und `frameGet(…)` gespeichert. Die Translation `[1,2,3]`, die auf den Transformationsstapel geschoben wird, führt zu einer Anpassung des aktuellen PCS-Punktes (Programmkoordinatensystem), so dass der MCS-Punkt (Maschinenkoordinatensystem) `[10,20,30]` unverändert bleibt. Daher wird beim anschließenden Aufruf von `frameGet(…)` der PCS-Punkt (Programmkoordinatensystem) `[9,18,27]` abgerufen.

```
{
VAR
    curX, curY, curZ : LREAL;
END_VAR

!G01 X10 Y20 Z30 F65000

frameGet(x=>curX, y=>curY, z=>curZ);
MSG(toString(curX,' ',curY,' ',curZ,''));

transTranslate(1,2,3);
frameGet(x=>curX, y=>curY, z=>curZ);
MSG(toString(curX,' ',curY,' ',curZ,''));
}
M02
```

Output:

```
10.000000 20.000000 30.000000
9.000000 18.000000 27.000000
```

qAxisGet

```
qAxisGet(q1:=LREAL, q2:=LREAL, q3:=LREAL, q4:=LREAL, q5:=LREAL)
```

Speichert die aktuellen Werte der `Q`-Achsen in `q1` bis `q5`. Die `Q`-Achsen sind die Hilfsachsen.

4.6.9 Werkzeugradiuskorrektur

trcApproachDepartSet

```
trcApproachDepartSet(approachRadius:= LREAL, approachAngle:= LREAL, departRadius
:= LREAL, departAngle:= LREAL)
```

Konfiguriert das An- und Abfahrverhalten so, dass ein Kreisbogen mit einem bestimmten Radius und Winkel verwendet wird. Wenn das Produkt aus Radius und Winkel Null ist, wird kein An- oder Abfahrtssegment eingefügt.

Die resultierende Konfiguration wird von `G41`/`G42` verwendet.

trcOffsetSet

```
trcOffsetSet(offset:= LREAL)
```

Konfiguriert den Umfang der Segmenterweiterung, die zum Schließen von Lücken verwendet wird. Wenn `offset` gleich Null ist, wird keine Verlängerung durchgeführt.

Die resultierende Konfiguration wird von `G41`/`G42` verwendet.

trcLimitSet

```
trcLimitSet(limit:= ULINT)
```

Konfiguriert den Lookahead, der für die Kollisionsvermeidung verwendet wird.

Die resultierende Konfiguration wird von `G41`/`G42` verwendet.

trcParam

```
trcParam(): TrcParamType
```

Liefert die aktuelle Konfiguration als Strukturwert.

trcParamSet

```
trcParamSet(param:= TrcParamType)
```

Konfiguriert die Werkzeugradiuskorrektur. Diese Funktion ist eine Alternative, die `trcApproachDepartSet`, `trcOffsetSet` und `trcLimitSet` zusammenfasst. Es kann in Kombination mit `trcParam` verwendet werden, um verschiedene TRC-Konfigurationen (Werkzeugradiuskorrektur) effizient zu laden, zu speichern und wiederherzustellen.

trcParamType

```
trcParamType
```

Diese Struktur enthält alle Konfigurationsparameter der Werkzeugradiuskorrektur. Sie besteht aus den folgenden Parametern.

```
approachRadius: LREAL;
approachAngle: LREAL;
departRadius: LREAL;
departAngle: LREAL;
offset: LREAL;
limit: ULINT;
```

Siehe `trcApproachDepartSet`, `trcOffsetSet`, `trcLimitSet` für eine umfassende Beschreibung der aufgeführten Parameter.

collisionElimination

```
collisionElimination(nx:= LREAL, ny:= LREAL, nz:= LREAL, limit:= ULINT)
```

Aktiviert die Kollisionsbeseitigung in Bezug auf die Ebene des Normalenvektors `nx`, `ny`, `nz`. Kollisionen innerhalb der Projektion der Bahn auf die Ebene werden eliminiert. Durch die Angabe eines Null-Vektors wird die Kollisionsbeseitigung deaktiviert. Der Parameter `limit` kann verwendet werden, um die Eliminierung auf die letzten `n` Segmente zu beschränken. Standardmäßig ist die Eliminierung unbegrenzt.

collisionEliminationFlush

```
collisionEliminationFlush()
```

Diese Funktion kann während der aktiven Kollisionsbeseitigung aufgerufen werden, um etwaige Konflikte zwischen der Bahn vor dem Aufruf und der Bahn nach dem Aufruf zu ignorieren.

4.6.10 Unterdrückung von G-Code-Sätzen

disableMask

```
disableMask(): LWORD
```

Liefert den aktuellen Wert der Disable-Maske. Beachten Sie, dass die Maske auch von der SPS eingestellt werden kann.

disableMaskSet

```
disableMaskSet(mask:= LWORD)
```

Setzt die interne Disable-Maske auf den angegebenen Wert. Die Maske wird verwendet, um die Ausführung von G-Codesätzen zu unterdrücken. Die Disable-Maske hat den Standardwert 0, d.h. standardmäßig ist keine Unterdrückung aktiv. Die Maske besteht aus 64 Bit.

In einer binären Schreibweise wie 2#1101 werden die Bits von rechts nach links nummeriert, beginnend mit dem Bit 0. Bei dem Wert 2#1101 werden die Bits 0, 2 und 3 durch den Wert one gesetzt. Die übrigen Bits sind nicht gesetzt, wenn sie den Wert zero aufweisen.

Beispiel:

Die resultierende Bahn des folgenden Beispiels ist in der Abbildung "BeispielDisableMaskSet" dargestellt. Die Disable-Maske ist zunächst auf den binären Wert 2#1101 eingestellt, der dem dezimalen Wert 13 entspricht. Der erste G-Code, im Beispiel N10, wird immer ausgeführt, unabhängig von der aktuellen Disable-Maske, da in der N10-Zeile kein '/'-Operator vorhanden ist. N20 wird nur ausgeführt, wenn das Bit 0 nicht gesetzt ist. Wenn das Bit 0 gesetzt ist, wird N20 unterdrückt, was im gegebenen Beispiel zutrifft. Dasselbe gilt für N30, da "/" nur eine Kurzform von "/0" ist. N40 wird nicht unterdrückt, da das Bit 1 nicht gesetzt ist. Die G-Codes N50 und N60 nach disableMaskSet(0) werden ausgeführt, da kein Bit in der Disable-Maske gesetzt ist. Im Gegensatz dazu setzt der Aufruf disableMaskSet(-1) alle Bits der Maske. Folglich sind die nachfolgenden G-Codes, denen ein '/' vorangestellt ist, N80 und N90, deaktiviert.

```
!disableMaskSet(2#1101);
N10 G01 X10 Y10 F6000
/0 N20 G01 X20 Y0
/ N30 G01 X30 Y0
/1 N40 G01 X40 Y10
!disableMaskSet(0);
/ N50 G01 X50 Y0
/1 N60 G01 X60 Y10
!disableMaskSet(-1);
N70 G01 X70 Y0
/1 N80 G01 X80 Y10
/2 N90 G01 X90 Y20
M02
```

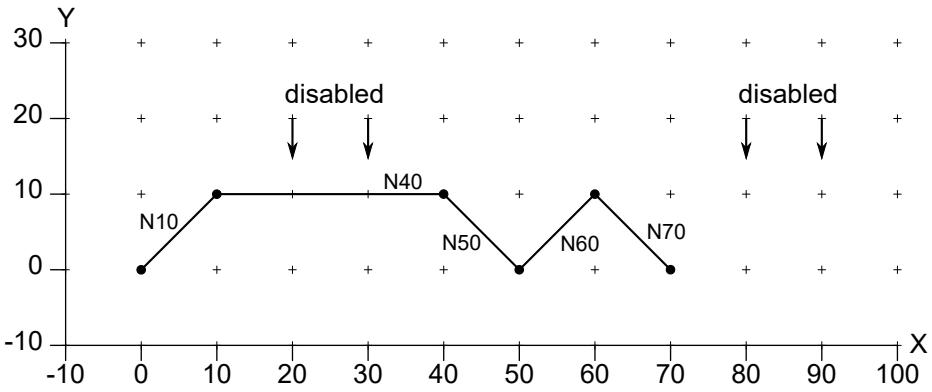


Abbildung "BeispielDisableMaskSet".

4.6.11 Nullpunktverschiebung

zeroOffsetShiftSet

```
zeroOffsetShiftSet(g:= USINT, x:= LREAL, y:= LREAL, z:= LREAL)
```

Setzt die Translation für G-Code g, wobei g eine der Zahlen 54, 55, 56 oder 57 sein muss. Alternativ kann die Nullpunktverschiebung auch mit dem SPS-Baustein ItpWriteZeroShiftEx [▶ 256] eingestellt werden.

Beispiel:

Die resultierende Bahn des folgenden Beispiels ist in der Abbildung "BeispielZeroOffsetShiftSet" dargestellt. Die Nullpunktverschiebung von `G54` wird zunächst auf die Translation `[0,10,0]` eingestellt. Sie wird für `N20` und alle späteren Segmentendpunkte aktiv, bis eine neue Translation angewendet wird. Der zweite Aufruf von `zeroOffsetShiftSet` hat eine unmittelbare Wirkung. Sie gilt für `N30` und alle späteren Segmentendpunkte, bis eine Novel Translation angewendet wird. Das Gleiche gilt für den letzten Aufruf. Der Satz `N40` programmiert jedoch nicht die `Y`-Koordinate. Daher wird die Änderung für `N40` nicht sichtbar. (Siehe Abschnitt Transformationen [▶ 99] für Einzelheiten.) Da der Satz `N50` die `Y`-Koordinate programmiert, wendet er die aktuelle `[0,30,0]`-Translation an.

```
!zeroOffsetShiftSet(g:=54, x:=0, y:=10, z:=0);
N10 G01 X20 Y0 F6000
N20 G01 G54 X40 Y0
!zeroOffsetShiftSet(g:=54, x:=0, y:=20, z:=0);
N30 G01 X60 Y0
!zeroOffsetShiftSet(g:=54, x:=0, y:=30, z:=0);
N40 G01 X80
N50 G01 X100 Y0
M02
```

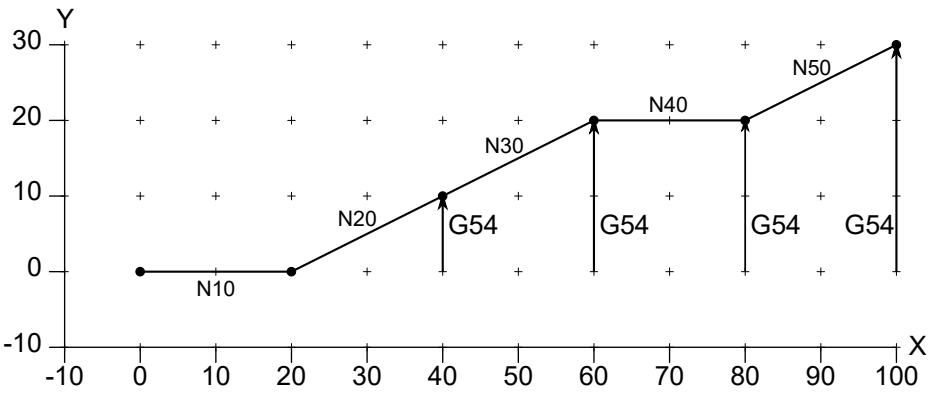


Abbildung "BeispielZeroOffsetShiftSet".

4.6.12 Einheiten

unitAngleSet

`unitAngleSet(unitAngle:= UnitAngle)`

Setzt die Einheit für Winkel auf `unitAngle`. Standardmäßig ist dies `unitAngleDegree`. Die Einheit für Winkel gilt für alle NC-bezogenen Funktionen wie `transRotX`. Sie gilt nicht für `ST`-Standardfunktionen wie `sin`. Aus diesem Grund bietet der Interpreter eine Reihe von NC-spezifischen Gegenstücken wie `gSin` an, die die Winkeleinheit berücksichtigen.

UnitAngle

Aufzählung der folgenden Werte:

```
unitAngleRadian: 0…2pi
unitAngleDegree: 0…360
unitAngleGon: 0…400
unitAngleTurn: 0…1
```

.

unitLengthSet

`unitLengthSet(unitLength:= UnitLength)`

Setzt die Einheit für Längen auf `unitLength`. Standardmäßig ist dies `unitLengthMillimeter`. Die Einheit für die Länge gilt für alle NC-bezogenen Funktionen wie `G01` oder `zeroOffsetShiftSet(…)`.

UnitLength

Aufzählung der folgenden Werte:

```
unitLengthMeter
unitLengthCentimeter
unitLengthMillimeter
unitLengthMicrometer
unitLengthNanometer
unitLengthInch
unitLengthFoot
```

.

unitTimeSet

```
unitTimeSet(unitTime:= UnitTime)
```

Setzt die Einheit für die Zeit auf `unitTime`. Standardmäßig ist dies `unitTimeSecond`. Die Einheit für Zeit gilt für alle NC-bezogenen Funktionen wie `G04`. Sie gilt nicht für ST-Standardfunktionen wie `currentLdt()`.

UnitTime

Aufzählung der folgenden Werte:

```
unitTimeSecond
unitTimeMillisecond
unitTimeMicrosecond
unitTimeMinute
unitTimeHour
```

.

unitVelocitySet

```
unitVelocitySet(unitLength:= UnitLength, unitTime:= UnitTime)
```

Setzt die Einheit für die Geschwindigkeit auf `unitLength/unitTime`. Standardmäßig ist dies `unitLengthMillimeter/unitTimeMinute`. Die Einheit für die Geschwindigkeit gilt für alle NC-bezogenen Funktionen. Sie wird zum Beispiel vom `F`-Parameter verwendet.

4.6.13 Trigonometrie (Einheiten beachten)

gSin

```
gSin(angle:= LREAL)
```

- Liefert den Sinus des angegebenen Winkels (`angle`) zurück, wobei die aktuelle Winkeleinheit [▶ 86] zur Interpretation des Winkels verwendet wird.
- Der Rückgabetyp ist `LREAL`. Diese Funktion ist nicht überladen.

gCos

```
gCos(angle:= LREAL)
```

- Liefert den Cosinus des angegebenen Winkels (`angle`) zurück, wobei die aktuelle Winkeleinheit [▶ 86] zur Interpretation des Winkels verwendet wird.
- Der Rückgabetyp ist `LREAL`. Diese Funktion ist nicht überladen.

gTan

```
gTan(angle:= LREAL)
```

- Liefert den Tangens des angegebenen Winkels (`angle`) zurück, wobei die aktuelle Winkeleinheit [▶ 86] zur Interpretation des Winkels verwendet wird.

- Der Rückgabetyp ist `LREAL`. Diese Funktion ist nicht überladen.

gASin

```
gASin(val:= LREAL)
```

- Liefert den Arkussinus von `val` in der aktuellen Winkeleinheit [▶ 86].
- Der Rückgabetyp ist `LREAL`. Diese Funktion ist nicht überladen.
- Das Ergebnis liegt innerhalb des Intervalls `[-c/4,c/4]`, wobei `c` der Winkel eines Vollkreises in der aktuellen Winkeleinheit ist.

EINSCHRÄNKUNG:
Die Variable `val` muss sich innerhalb des Intervalls `[-1,1]` befinden.

gACos

```
gACos(val:= LREAL)
```

- Liefert den Arkuscosinus von `val` in der aktuellen Winkeleinheit [▶ 86] zurück.
- Der Rückgabetyp ist `LREAL`. Diese Funktion ist nicht überladen.
- Das Ergebnis liegt innerhalb des Intervalls `[0,c/2]`, wobei `c` der Winkel eines Vollkreises in der aktuellen Winkeleinheit ist.

EINSCHRÄNKUNG:
Die Variable `val` muss sich innerhalb des Intervalls `[-1,1]` befinden.

gATan

```
gATan(val:= LREAL)
```

- Liefert den Arkustangens von `val` in der aktuellen Winkeleinheit [▶ 86] zurück.
- Der Rückgabetyp ist `LREAL`. Diese Funktion ist nicht überladen.
- Das Ergebnis liegt innerhalb des Intervalls `[-c/4,c/4]`, wobei `c` der Winkel eines Vollkreises in der aktuellen Winkeleinheit ist.

gATan2

```
gATan2(y:= LREAL, x:= LREAL)
```

- Liefert den Arkustangens von `y/x` in der aktuellen Winkeleinheit [▶ 86] zurück.
- Der Rückgabetyp ist `LREAL`. Diese Funktion ist nicht überladen.
- Das Ergebnis liegt innerhalb des Intervalls `[-c/2,c/2]`, wobei `c` der Winkel eines Vollkreises in der aktuellen Winkeleinheit ist.

4.6.14 Vorschubmodus

feedModeSet

```
feedModeSet(feedMode:= FeedModeType)
```

FeedModeType

Aufzählung der folgenden Werte:

```
fmContour
fmInternalRadius
fmToolCenterPoint
```

fmKontur: Hält die Vorschubgeschwindigkeit an der Kontur konstant.

fmInternalRadius: Reduziert den Vorschub bei Innenradien. Dadurch ergibt sich an der Kontur eine konstante Geschwindigkeit. Am Außenkreis wird die Geschwindigkeit nicht erhöht.

fmToolCenterPoint: Hält die Vorschubgeschwindigkeit des Werkzeugmittelpunkts konstant. D.h. an Innenkreisen wird die Geschwindigkeit an der Kontur erhöht und an Außenkreisen entsprechend verringert.

4.6.15 Vorschubinterpolation

feedInterpolationSet

```
feedInterpolationSet(feedInterpolation:= FeedInterpolationType)
```

FeedInterpolationType

```
FeedInterpolationType
```

Aufzählung der folgenden Werte:

```
fiConstant
fiLinear
```

fiConstant: Mit der konstanten Vorschubinterpolation (default) wird die programmierte Geschwindigkeit so schnell wie möglich angefahren.

fiLinear: Die lineare Vorschubinterpolation überführt die Geschwindigkeit von v_start nach v_end linear über den Bahnweg.

4.6.16 Streaming von großen G-Code-Dateien

runFile

```
runFile(path:= string)
```

Die Größe der Dateien, die mit dem GST-Interpreter ausgeführt werden können, ist begrenzt. Manchmal ist es jedoch erforderlich, große Dateien auszuführen, die z. B. von einem CAD-Programm erstellt wurden. Somit hat der Benutzer die Möglichkeit, Filestreams von nativem G-Code auszuführen.

Führt den G-Code aus, der in der G-Code-Datei enthalten ist, die durch `path` angegeben wird. Der Funktionsaufruf kehrt zurück, nachdem alle Zeilen der übergebenen Datei verarbeitet wurden. Die Funktion ist dafür gedacht, große G-Codedateien effizient an den NC-Kernel zu streamen.

> **Native G-Code: Strukturierter Text ist nicht erlaubt**
>
> Beachten Sie, dass die mitgelieferte G-Code-Datei keine ST-Elemente enthalten darf, sondern nur einfachen G-Code.

Ein G-Code-Filestream aus der Datei `'myNativeGCodeFile.nc'` kann aus einem GST-Programm mit folgender Syntax aufgerufen werden:

```
!runfile('myNativeGCodeFile.nc');
```

runFile mit R-Parametern und einfachen arithmetischen Ausdrücken

Ab der TwinCAT V3.1.4024.40 sind R-Parameter und einfache arithmetische Ausdrücke mit Runfile erlaubt.

callRunfileWithRParamsAndExpressions.nc

```
N0 G0 X0 Y0 Z0
N1 G1 X10 F5000
!R3:=3;
!R5:=5;
!R10:=15;
!r100:=-1.234;
```

```
!R888:=98.123;
N2 G1 y={r100}
!runfile(RunfileWithRParamsAndExpressions.nc');
M30
```

RunfileWithRParamsAndExpressions.nc

```
G1 X200
G1 Y150
G1 Z234
G1 z = R888 - r100 - r100 / R10
G1 y=100-r888
G1 x=-R10/r5-r3 q1=20-r3*R5
```

4.6.17 Vertex-Verschleifung

smoothingSet

```
smoothingSet(mainType:= SmoothingMainType, subType:= SmoothingSubType, value:=
LREAL)
```

Legt das Verhalten der Vertex-Verschleifung entsprechend den angegebenen Parametern fest.

SmoothingMainType

Aufzählung der folgenden Werte:

```
smoothingNone
smoothingParabola
smoothingBiquadratic
smoothingBezier3
smoothingBezier5
smoothingTwinBezier
```

smoothingNone: Keine Verschleifung.

smoothingParabola: Bei der Parabel-Verschleifung wird geometrisch eine Parabel im Segmentübergang eingefügt. Dadurch wird die Geschwindigkeit innerhalb des Toleranzradius stetig überführt.

smoothingBiquadratic: Bei der Bi-Quadratischen-Verschleifung tritt kein Beschleunigungssprung in den Achskomponenten auf. Bei gleichem Radius ist deshalb evtl. eine kleinere Eingangsgeschwindigkeit als bei der Parabel notwendig.

smoothingBezier3: Bei der Bezier-Verschleifung 3-ter Ordnung tritt mit dem Eintritt in die Toleranzkugel ein Beschleunigungssprung in den Achskomponenten auf. Die max. Größe wird durch die Beschleunigungen der Achskomponenten und den C1-Faktor begrenzt.

smoothingBezier5: Bei der Bezier-Verschleifung 5-ter Ordnung tritt mit dem Eintritt in die Toleranzkugel kein Beschleunigungssprung in den Achskomponenten auf. D. h. bei angewählter Verschleifung, ist der Beschleunigungsverlauf für die Bahnachsen immer stetig.

smoothingTwinBezier: Mit Hilfe der Glättung ist es möglich, zwischen zwei Geometrieeinträgen automatisch ein Bezier-Spline einzufügen. Hierfür muss lediglich der Radius der Toleranzkugel programmiert werden. Dieser beschreibt die maximal erlaubte Abweichung von der programmierten Kontur im Segmentübergang. Der Vorteil bei dieser Art der Glättung gegenüber der Verrundung mit Kreiselement ist, dass hier an den Segmentübergängen keine Beschleunigungssprünge entstehen.

Spitze Winkel am Segmentübergang

Die Bezier-Splines werden standardmäßig auch bei sehr spitzen Winkeln generiert. Damit die Dynamikwerte nicht überschritten werden, ist für diesen Fall eine erhebliche Geschwindigkeitsreduktion erforderlich. Da im Spline die Dynamik konstant gehalten wird, dauert es entsprechend lange, bis der Spline durchfahren wird. Für diesen Fall ist es häufig sinnvoll, den Segmentübergang mit einem Genauhalt anzufahren. Damit die Winkel nicht manuell berechnet werden müssen, gibt es den Befehl AutoAccurateStop [▶ 161].

SmoothingSubType

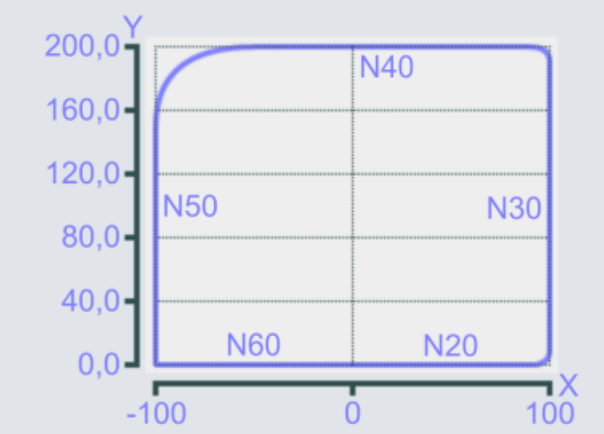
Aufzählung der folgenden Werte:

```
smoothingRadius
smoothingDistance
smoothingAdaptive
```

Beispiel

Das Beispiel veranschaulicht den Effekt der Verwendung einer Parabel-Verschleifung. Für die ersten beiden Ecken wurde der Glättungswert `10` und für die dritte Ecke der Glättungswert `50` verwendet. Die vierte Ecke schließlich weist den Glättungswert `0` auf.

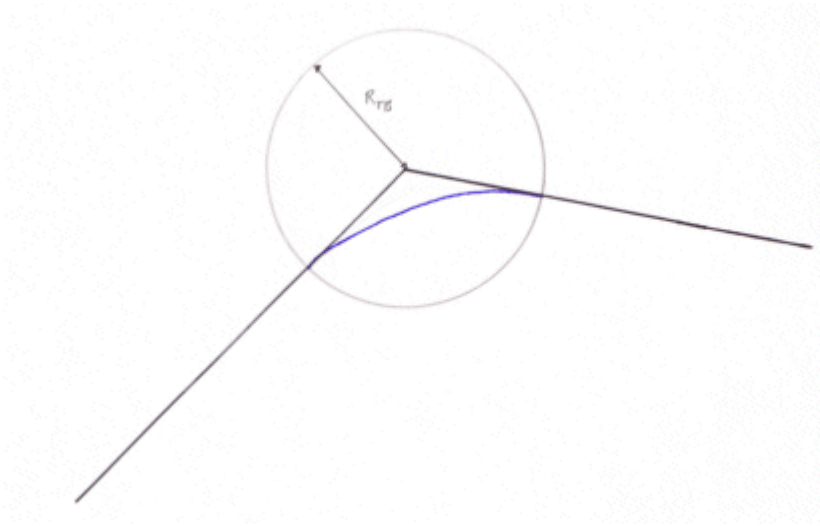
```
N10 G01 X0 Y0 F60000
!smoothingSet(mainType:=smoothingParabola, subType:=smoothingRadius, value:=10);
N20 G01 X100
N30 Y200
!smoothingSet(mainType:=smoothingParabola, subType:=smoothingRadius, value:=50);
N40 X-100
!smoothingSet(mainType:=smoothingParabola, subType:=smoothingRadius, value:=0);
N50 Y0
N60 X0
M02
```



4.6.17.1 Subtypen

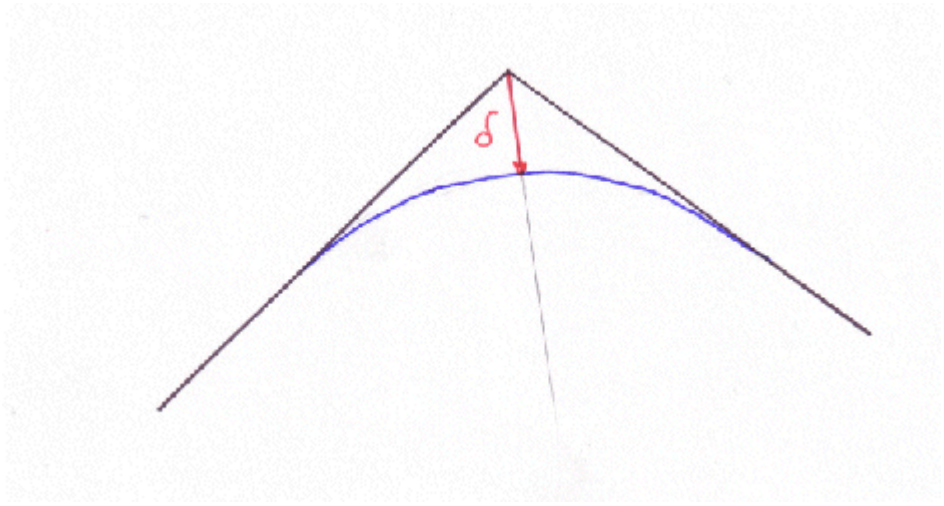
Konstanter Toleranzradius (SmoothingRadius – Subtyp 1)

Ist Subtyp 1 angewählt, so wird immer der maximale Toleranzradius (R_{TB}) für die Verschleifung verwendet. Eine Verkleinerung von R_{TB} erfolgt genau dann, wenn das Ein- bzw. Ausgangssegment kleiner als $3*R_{TB}$ ist.



Abstand vom Schnittpunkt zum Scheitelpunkt (SmoothingDistance – Subtyp 2)

Mit dem Subtyp 2 wird der Abstand vom programmierten Segmentübergang zum Scheitelpunkt der Parabel vorgegeben. Anhand dessen ergibt sich dann der Toleranzradius (R_{TB}). Ist ein Segment zu kurz, so wird der Abstand so verkürzt, dass der Toleranzradius max. 1/3 einnimmt.



Adaptiver Toleranzradius (SmoothingAdaptive – Subtyp 3)

Innerhalb des Toleranzradius (auch beim konstanten Toleranzradius) wird immer sichergestellt, dass die maximal zulässige Beschleunigung nicht überschritten wird. Je nach Ablenkungswinkel und Geschwindigkeit kann damit die maximal auftretende Achsbeschleunigung im Verrundungssegment unterschiedlich ausfallen. Ziel beim adaptiven Toleranzradius ist eine maximale Beschleunigung innerhalb der Verschleifung. Um dies zu erreichen, wird der Verrundungsradius unter Berücksichtigung der programmierten Geschwindigkeit und Dynamik entsprechend verkleinert. D.h. wird die programmierte Geschwindigkeit verändert, so kann sich auch der Toleranzradius ändern. Der Override hat keinen Einfluss auf den Radius.

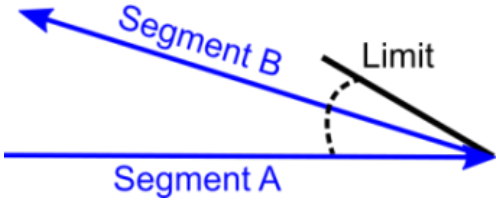
4.6.18 Automatischer Genauhalt

autoAccurateStopSet

```
autoAccurateStopSet(angle:= LREAL);
```

Das Kommando `autoAccurateStopSet` wird in Verbindung mit Verschleifungen verwendet (siehe smoothingSet) und erlaubt das Anfahren von spitzen Winkeln bei aktiver Verschleifung. Dazu wird im Kommando `autoAccurateStopSet` ein Grenzwinkel definiert, bis zu dem ein Genauhalt zwischen 2 Segmenten zu erfolgen hat.

```
!autoAccurateStopSet(angle:= 30.0);
```

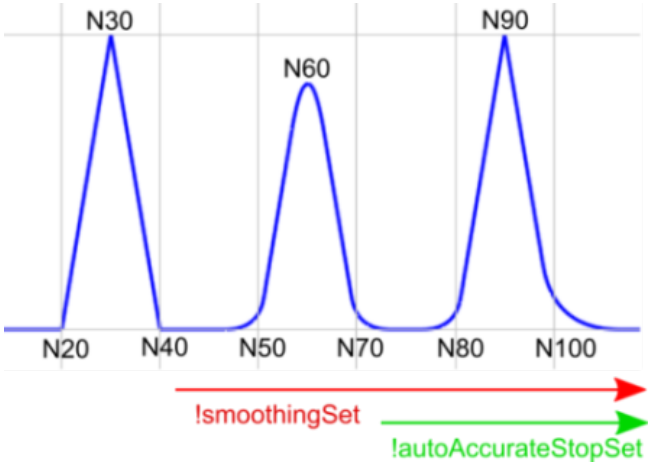


Bei Kreissegmenten errechnet sich der Winkel aus den Eingangs- bzw. Ausgangstangenten.

Beispiel

```
N10 G0 X0 Y0 Z0

N20 G01 X10 F20000
N30 G01 X15 Y30
N40 G01 X20 Y0

!smoothingSet(mainType:=smoothingParabola,
subType:=smoothingRadius, value:=50);
N50 G01 X30
N60 G01 X35 Y30
N70 G01 X40 Y0

!autoAccurateStopSet(angle:= 46.0);
N80 G01 X50
N90 G01 X55 Y30
N100 G01 X60 Y0
```

```
N110 G01 X80

N110 M30
```



Voraussetzungen

Entwicklungsumgebung	Zielsystem
TwinCAT V3.1.4024.15	PC oder CX (x86 oder x64)

4.6.19 Spline-Interpolation

transBSpline

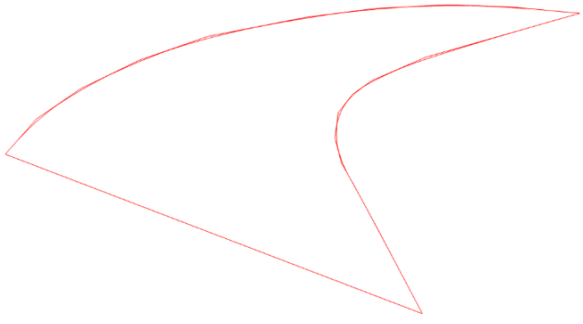
```
transBSpline(BreakAngle:=LREAL, BreakLength:=LREAL, MergeDiff:=LREAL,
LineBreakAngle:=LREAL,LineBreakLength:=LREAL, LineMergeDiff:=LREAL)
```

transBSpline erzeugt eine kontinuierliche Kurve aus einer stückweisen linearen Polylinie.
Die Kurve wird durch die Eingangspolylinie begrenzt, die Start- und Endpunkte werden interpoliert, innere Punkte sind die Kontrollpunkte (DeBoor-Punkte) der Kurve. Es sind mindestens drei Punkte erforderlich. Eine BSpline-Kurve weist eine lokale Kontrolle auf und ist daher für die Manipulation von Kontrollpunkten geeignet.

```
//Enable
transBSpline(BreakAngle := 70, BreakLength := 1000);
```

```
//Disable
transBSpline();
```

Beispiel:

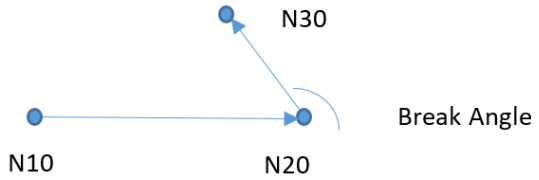


```
!//BSpline
N10 G00 X18.498 Y0
!transBSpline(BreakAngle:=70.0, BreakLength:=1000.0);
N20 G01 X18.498 Y0 Z0 F6000
N30 X16.572 Y6.543 Z1
N40 X15.616 Y9.715 Z2
N50 X15.121 Y11.275 Z3
N60 X14.838 Y13.196 Z4
N70 X14.982 Y15.085 Z5
N80 X15.595 Y16.485 Z6
```

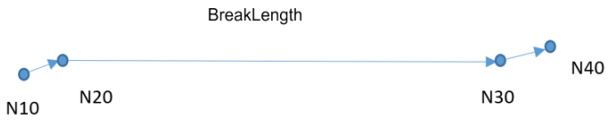
```
N90 X16.396 Y17.490 Z7
N100 X18.653 Y19.243 Z8
N110 X25.07 Y22.526 Z9
N120 X22.228 Y22.997 Z8
N130 X19.569 Y23.174 Z7
N140 X16.488 Y22.884 Z6
N150 X13.634 Y22.228 Z5
N160 X9.533 Y20.793 Z4
N170 X6.668 Y19.009 Z3
N180 X4.224 Y16.877 Z2
N190 X2.376 Y14.61 Z1
N200 X1.068 Y11.959 Z0
! transBSpline();
M02
```

Parameter

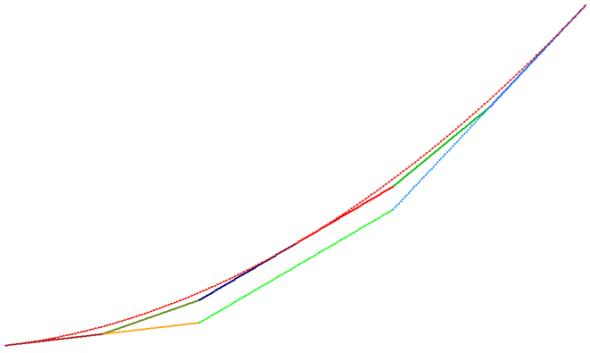
BreakAngle (obligatorisch): Ermöglicht die Beibehaltung von scharfkantigen Merkmalen in der Bahn. Der Spline bricht, wenn derdie Bahn um mehr als BreakAngle abweicht. Der Spline beendet und interpoliert den Punkt.



BreakLength (obligatorisch): Ermöglicht die Beibehaltung von langen Merkmalen in der Bahn. Der Spline bricht bei Segmenten, die länger als BreakLength sind. Der Spline beendet und interpoliert die Start- und Endpunkte des langen Segments.



MergeDiff (optional): Der BSpline setzt sich aus Bezier-Segmenten zusammen. Um die Verarbeitungsgeschwindigkeit zu erhöhen, kann der Spline durch Zusammenführen komprimiert werden. Benachbarte Segmente werden zusammengeführt, wenn die Differenz der Kontrollpunkte kleiner ist als MergeDiff. Unten werden benachbarte Segmente zu einem zusammengeführt.



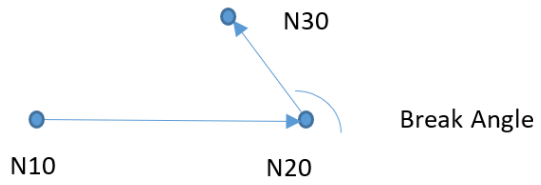
Übermäßige Krümmung

Ein zu aggressives Zusammenführen kann zu einer übermäßigen Verformung führen, und ein Segment mit übermäßiger Krümmung wird mit einem Laufzeitfehler zurückgewiesen.

Die zulässige Krümmung ergibt sich aus der Bahngeschwindigkeit und der Beschleunigung.

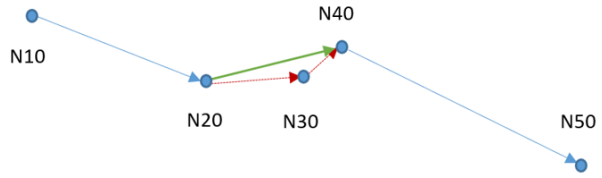
Die BSpline wird aus einer Kontrollpunkt-Polylinie konstruiert, die aus G01-Segmenten gebildet wird, z. B.: CADCAM. Um die Verarbeitungsgeschwindigkeit zu erhöhen, kann die Kontrollpunkt-Polylinie durch Zusammenführen benachbarter Segmente komprimiert oder vereinfacht werden.

LineBreakAngle (optional): Die Zusammenführung benachbarter Kontrollpunkte wird unterbrochen, wenn der Ablenkungswinkel LineBreakAngle überschreitet.



LineBreakLength (optional): Die Zusammenführung benachbarter Kontrollpunkte wird unterbrochen, wenn die Länge LineBreakLength überschritten wird.

LineMergeDiff (optional): Benachbarte Kontrollpunkte werden zusammengeführt, wenn die Differenz (senkrechter Abstand) kleiner als LineMergeDiff ist. In diesem Beispiel kann N30 weggelassen werden, was das Kontrollpolygon vereinfacht.



Sind die optionalen Parameter nicht parametrisiert oder sind sie 0, findet keine Zusammenführung statt.

Übermäßige Krümmung

Ein zu aggressives Zusammenführen kann zu einer übermäßigen Verformung führen, und ein Segment mit übermäßiger Krümmung wird mit einem Laufzeitfehler zurückgewiesen.

Die zulässige Krümmung ergibt sich aus der Bahngeschwindigkeit und der Beschleunigung.

Abarbeitungsreihenfolge:

Wenn BreakAngle oder BreakLength 0 sind, findet keine weitere Verarbeitung statt. LineBreakAngle, LineBreakLength und LineMergeDiff werden zunächst abgearbeitet, um die Kontrollpunkt-Polylinie zu vereinfachen. BreakAngle, BreakLength und MergeDiff werden schließlich abgearbeitet, um die BSpline-Kurve zu erzeugen.

Dekodierstopps und Handshake M-Funktionen:

Die BSpline sollte vor einem Dekodierstopp oder einer M-Funktion vom Typ Handshake mit `!transBSpline();` beendet werden.

```
!//BSpline
N10 G00 X18.498 Y0
!transBSpline(BreakAngle:=70.0, BreakLength:=1000.0);
N20 G01 X18.498 Y0 Z0 F6000
N30 X16.572 Y6.543 Z1
N40 X15.616 Y9.715 Z2
N50 X15.121 Y11.275 Z3
N60 X14.838 Y13.196 Z4
N70 X14.982 Y15.085 Z5
```

```
N80 X15.595 Y16.485 Z6
N90 X16.396 Y17.490 Z7
N100 X18.653 Y19.243 Z8
N110 X25.07 Y22.526 Z9
!transBSpline();
!sync();
!transBSpline(BreakAngle:=70.0, BreakLength:=1000.0);
N120 X22.228 Y22.997 Z8
N130 X19.569 Y23.174 Z7
N140 X16.488 Y22.884 Z6
N150 X13.634 Y22.228 Z5
N160 X9.533 Y20.793 Z4
N170 X6.668 Y19.009 Z3
N180 X4.224 Y16.877 Z2
N190 X2.376 Y14.61 Z1
N200 X1.068 Y11.959 Z0
! transBSpline();
M02
```

Kompatible G-Codes und Funktionen

Andere G-Codes als `G01` werden unterstützt.

`G00`

`G02, G03 (Circle and Helix):` Die BSpline wird vorher beendet und danach fortgesetzt.

`G04`

`G09, G60`

`G54` und andere Transformationen

```
disableMask()
runFile(path:= )
smoothingSet(mainType:=smoothingTwinBezier,subType:=smoothingRadius,value:= )
```

ToolRadiusCompensation wird nicht unterstützt.

Voraussetzungen

Entwicklungsumgebung	Zielsystem
TwinCAT V3.1.4024.4	PC oder CX (x86 oder x64)

4.6.20 Dynamischer Override

dynOverrideSet

`dynOverrideSet(value:= LREAL)`

Setzt den dynamischen Override der Achsen auf die angegebene `value`.

Mit der dynamischen Override-Funktion können prozentuale Änderungen an den dynamischen Achsparametern in der Achsgruppe während der Laufzeit des Programms `NC` vorgenommen und evoziert werden. Diese Änderungen führen also zu neuen Werten für die Bahndynamik. Die neuen Dynamikwerte werden ohne Stopp mit der Ausführung der Zeile gültig.

Wertebereich

Der Faktor `value` für `dynOverrideSet` muss sich innerhalb des Bereichs `0 < value ≤ 1.0` befinden.

Beispiel

In diesem Beispiel werden die neuen dynamischen Werte ohne Stopp gültig. Im Satz `N010` werden die zuvor eingestellten Werte für die Verzögerung verwendet, während die geänderten Werte im Satz `N020` für die Beschleunigung verwendet werden.

```
N010 G01 X100 Y200 F6000
!dynOverrideSet(value:= 0.4);
N020 G01 X500
M02
```

Beispiel

Der Befehl `dynOverrideSet` kann verwendet werden, um Beschleunigung und Ruck z. B. nur für eine Bewegung zu reduzieren. Im Beispiel werden Beschleunigung und Ruck um `50` Prozent nur im Satz `N020` reduziert.

```
N010 G01 X100 Y100 F6000
!dynOverrideSet(value:= 0.5);
N020 X0
!dynOverrideSet(value:= 1);
N030 X100
M02
```

4.6.21 Programmierreferenz

programmingReferenceSet

```
programmingReferenceSet(value:= ReferenceType)
```

Alternativ zu G90/G91 [▶ 49] kann mit `programmingReferenceSet` die Maßangabe der nachfolgenden Bewegungsbefehle auf absolute oder relative gesetzt werden.

programmingReferenceGet

```
ReferenceType := programmingReferenceGet()
```

Mit `programmingReferenceGet` kann die aktuell aktive Maßangabe für die Bewegungsbefehle ausgelesen werden.

ReferenceType

Aufzählung der folgenden Werte:

```
referenceAbsolute
referenceRelative
```

Beispiel

Die Verwendung von `programmingRefGet/Set` empfiehlt sich besonders bei Funktionen. Zu Beginn einer Funktion kann mit `programmingRefGet` die aktuell eingestellte Maßangabe ausgelesen werden, bevor diese mit `G90/G91` oder `programmingRefSet` auf die Maßangabe der Funktion gesetzt wird. Am Ende der Funktion wird mit `programmingRefSet` sichergestellt, dass die Maßangabe zurückgesetzt ist.

```
{
FUNCTION TestFunction
VAR
    programmingRef : ReferenceType;
END_VAR

    programmingRef := programmingReferenceGet();
    programmingReferenceSet(value:= referenceRelative);

    !N21 G01 X10 Y-10
    !N22 G01 X10 Y-10

    programmingReferenceSet(value:= programmingRef);

END_FUNCTION
}

N10 G00 X0 Y0 Z0
N20 G01 G90 X100 Y20 F6000

!TestFunction();

N30 G01 X100 Y30
M30
```

Voraussetzung

TwinCAT
TwinCAT V3.1.4024.54

4.6.22 Mittelpunkt-Referenz von Kreisen

circleCenterReferenceSet

```
circleCenterReferenceSet(value:= ReferenceType)
```

- Legt die Art der Mittelpunkt-Referenz für Kreise fest, die über `G02`/`G03` mit einem Mittelpunkt programmiert werden, dessen Definition die `i`,`j`,`k`-Parameter beinhaltet.
- Für `referenceAbsolute` wird der Mittelpunkt des Kreises durch den mitgelieferten `i`,`j`,`k`-Vektor definiert.
- Für `referenceRelative` wird der Mittelpunkt durch die Summe aus dem Kreisanfangspunkt und dem mitgelieferten `i`,`j`,`k`-Vektor definiert. Dies ist die Standardeinstellung und das übliche Verhalten von `G`-Code.

circleCenterReferenceGet

```
ReferenceType := circleCenterReferenceGet()
```

Gibt die aktuell aktive Mittelpunkt-Referenz für Kreise zurück.

Vorraussetzung circleCenterReferenceGet

TwinCAT
TwinCAT V3.1.4024.54

ReferenceType

Aufzählung der folgenden Werte:

```
referenceAbsolute
referenceRelative
```

.

4.6.23 Änderung der Achsdynamik

axisDynamicsSet

```
axisDynamicsSet(axisNo:= UDINT, acc:= LREAL, dec:= LREAL, jerk:= LREAL);
```

Mit `axisDynamicsSet` kann zur Laufzeit die Achsdynamik geändert werden.

Funktion	axisDynamicsSet
Parameter <axisNo>	Achse in der Interpolationsgruppe: X: 0 Y: 1 Z: 2 Q1: 3 ... Q5: 7
Parameter <acc>	Wert für die maximal erlaubte Beschleunigung in mm/s^2
Parameter <dec>	Wert für die maximal erlaubte Verzögerung in mm/s^2
Parameter <jerk>	Wert für den maximal erlaubten Ruck in mm/s^3.

Beispiel:

```
N10 G01 X100 Y200 F6000
!R4:=10000;
!axisDynamicsSet(axisNo:= 0, acc:= 2250, dec:= 2250, jerk:= R4);
N30 G01 X500
N40 M02
```

Voraussetzungen

Entwicklungsumgebung	Zielsystem
TwinCAT V3.1.4024.4	PC oder CX (x86 oder x64)

4.6.24 Änderung der Bahndynamik

pathDynamicsSet

```
pathDynamicsSet(acc:= LREAL, dec:= LREAL, jerk:= LREAL);
```

Mit `pathDynamicsSet` kann zur Laufzeit die Bahndynamik geändert werden.

Funktion	pathDynamicsSet
Parameter <acc>	Wert für die maximal erlaubte Beschleunigung in mm/s^2. Muss >= 1 gesetzt werden. Bei 0 wird der Standardwert verwendet.
Parameter <dec>	Wert für die maximal erlaubte Verzögerung in mm/s^2. Muss >= 1 gesetzt werden. Bei 0 wird der Standardwert verwendet.
Parameter <jerk>	Wert für den maximal erlaubten Ruck in mm/s^3. Muss >= 1 gesetzt werden. Bei 0 wird der Standardwert verwendet.

Beispiel:

```
N10 G01 X100 Y200 F60000
!R4:=10000;
//Set path dynamics
!pathDynamicsSet(acc:=200, dec := 200, jerk := R4);
N30 G01 X500 Y0
//Set path dynamics back to default values and jerk to 12000
!pathDynamicsSet(acc:=0, dec := 0, jerk := 12000);
N50 G01 X100 Y200
//Set path dynamics to default values
!pathDynamicsSet(acc:=0, dec := 0, jerk := 0);
N70 G01 X500 Y0
N80 M02
```

Requirements

Development Environment	Target System
TwinCAT V3.1.4024.12	PC or CX (x86 or x64)

4.7 Transformationen

Wenn wir von `GST`-Transformationen sprechen, sprechen wir z. B. von Rotationen oder Nullpunktverschiebungen.

Die Beziehung zwischen dem MCS (Maschinenkoordinatensystem) und dem PCS (Programmkoordinatensystem) wird durch die effektive Transformation T definiert. T ist die Verkettung der Transformationen T_Z, T_U und T_T ($T = T_Z * T_U * T_T$). Beachten Sie, dass die Reihenfolge der Verkettung von Bedeutung ist, da die Transformationen sich im Allgemeinen nicht vertauschen lassen. Die Transformation T_Z steht für eine (historische) Nullpunktverschiebung, die Transformation T_U für eine benutzerdefinierte Transformation und die Transformation T_T für eine Werkzeugtransformation. Sie werden später im Detail beschrieben.

Die Abbildung "TransformationenTzTuTt" visualisiert die Beziehung zwischen dem MCS (Maschinenkoordinatensystem) und dem PCS (Programmkoordinatensystem):

- T_Z ist definiert als eine Translation um `[20,20,0]`,
- T_U ist eine Kombination aus der Translation `[30,-10,0]`, gefolgt von einer Drehung um `45` Grad um die `Z`-Achse,
- T_T ist eine Translation um `[0,-10,0]`.

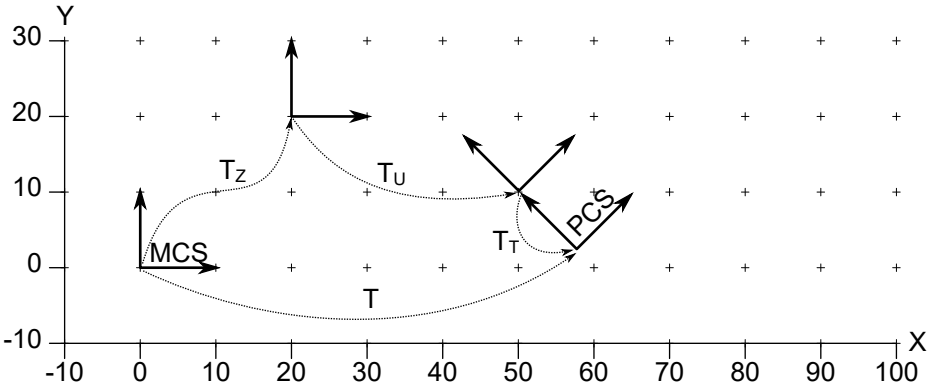


Abbildung "TransformationenTzTuTt".

4.7.1 Änderung der effektiven Transformation T und ihre Auswirkungen

Die meisten `G`-Codes definieren nur den Zielpunkt einer Bewegung. Daher behält der Interpreter die aktuelle Position des Werkzeugs bei. Dieser Punkt kann in MCS-Koordinaten (Maschinenkoordinatensystem) und PCS-Koordinaten (Programmkoordinatensystem) dargestellt werden, wobei die Gleichung `CurrentPointMCS = T * CurrentPointPCS` gilt. Im Gegensatz zur bisherigen Umsetzung gilt diese Transformationsgleichung auch nach einer Änderung von T. Dieses Verhalten wird durch die Anpassung von `CurrentPointPCS` erreicht. Der MCS-Punkt (Maschinenkoordinatensystem) wird nicht angepasst, da dies die Maschine beeinträchtigen würde. Dieses Verhalten lässt sich grob wie folgt zusammenfassen: Wenn die aktive Transformation geändert wird, wird der aktuelle PCS-Punkt (Programmkoordinatensystem) so angepasst, dass die Änderung keine Auswirkungen hat.

Beispiel:

Nach `N10` lauten die Koordinaten des aktuellen PCS- (Programmkoordinatensystem) und MCS-Punktes (Maschinenkoordinatensystem) `[20,10,80]`, da keine Transformation aktiv ist. Durch die Translation wird der aktuelle PCS-Punkt (Programmkoordinatensystem) auf `[28,7,84]` geändert. Die Anwendung der Translation auf diesen Punkt ergibt den unveränderten MCS-Punkt (Maschinenkoordinatensystem) `[20,10,80]`. Die Translation zeigt also keine Wirkung, obwohl sie aktiv ist. Der Satz `N20` programmiert eine Bewegung zum PCS-Punkt (Programmkoordinatensystem) `[25,7,10]`, der auf die MCS-Koordinate (Maschinenkoordinatensystem) `[17,10,6]` abgebildet wird. Nach dem Aufruf von `transPop()` wird der aktuelle PCS-Punkt (Programmkoordinatensystem) auf den aktuellen MCS-Punkt (Maschinenkoordinatensystem) gesetzt.

```
N10 G01 X20 Y10 Z80 F6000
!transTranslate(-8,3,-4);
N20 G01 X25 Z10
!transPop();
M02
```

Beispiel:

Wenn der Benutzer möchte, dass der PCS-Punkt (Programmkoordinatensystem) unverändert bleibt, muss er ihn abrufen und programmieren, wie im folgenden Code gezeigt. Der Wunsch nach einem unveränderten PCS-Punkt (Programmkoordinatensystem) deutet jedoch in der Regel auf einen schlechten Programmierstil hin. Eigentlich sollte der folgende Code nicht erforderlich sein.

```
{
VAR
    pcsX, pcsY, pcsZ : LREAL;
END_VAR
```

```
// … G-Code …

frameGet(x=>pcsX,y=>pcsY,z=>pcsZ);
// … modify transformations …
!G01 x=pcsX y=pcsY z=pcsZ F6000
}
```

4.7.2 Komponenten der effektiven Transformation T

Nullpunktverschiebung T_Z

Die T_Z-Transformation wird durch bestimmte `G`-Codes beeinflusst. Sie hat keine Wirkung, wenn `G53` aktiv ist. Andernfalls ist T_Z die Kombination aus den drei Translationen T_{Z58}, T_{Z59} und einer von {T_{Z54}, ..., T_{Z57}}. Die ersten beiden Translationen werden über die `G`-Codes `G58` und `G59` eingestellt. Die letztgenannte Translation wird durch die `G`-Codes `G54` bis `G57` ausgewählt. Jedem dieser 4 `G`-Codes ist eine Translation zugeordnet. Sie kann von der SPS oder über die `ST`-Funktion `zeroOffsetShiftSet` eingestellt werden.

Werkzeug-Transformation T_U

T_T wird durch das aktuell ausgewählte Werkzeug definiert. Sie hat keine Wirkung, wenn das Werkzeug `0` (`D0`) ausgewählt ist. Andernfalls handelt es sich um eine Translation um `[offsetX, offsetY, offsetZ] + (length+lengthAdd) * D`, wobei `D` die Normale der aktuellen Arbeitsebene ist.

Benutzerdefinierte Transformation T_U

T_U ist durch einen Stapel von Transformationen definiert. Der Stapel der Tiefe `N` enthält elementare Transformationen T_{U1}, T_{U2}, ..., $T_{U<N>}$ wobei $T_{U<N>}$ die oberste Transformation ist. Zu Beginn ist der Stapel leer. Die benutzerdefinierte Transformation ist die Verkettung dieser elementaren Transformationen $T_U = T_{U1} * T_{U2} * … * T_{U<N>}$. Beachten Sie, dass die Reihenfolge von Bedeutung ist, da sich die Transformationen im Allgemeinen nicht vertauschen lassen. Wenn der Stapel leer ist, ist T_U die Identitätstransformation, die keine Transformationswirkung hat.

4.7.3 Transformationsanwendungen

Eine Transformation wird mit den folgenden `ST`-Funktionen auf den Stapel geschoben. Die zuletzt geschobene Transformation ist die oberste Transformation auf dem Transformationsstapel. Wenn eine Transformation auf den Transformationsstapel geschoben wird, wird die Stapeltiefe um eins erhöht und T_U entsprechend angepasst.

```
transTranslate(x:= LREAL, y:= LREAL, z:= LREAL);
(* A rotation pushed onto the stack of transformations is interpreted around the respective
axis using the current angle-unit, e.g. degree or radian. *)
transRotX(angle:= LREAL);
transRotY(angle:= LREAL);
transRotZ(angle:= LREAL);
transMirrorX();
transMirrorY();
transMirrorZ();
transScale(factor:= LREAL);
```

4.7.4 Widerrufen von Transformationen

transPop

`transPop()`

Die Funktion `transPop()` entfernt die oberste Transformation aus dem Transformationsstapel. Wenn `transPop()` eine Transformation aus dem Transformationsstapel entfernt, wird die Stapeltiefe um eins verringert und T_U entsprechend angepasst. Üblicherweise wird die `transPop()`-Funktion verwendet, um eine temporäre Transformation zu widerrufen.

Beispiel:

Im folgenden Beispiel wird die Translation auf `N10`, `N20` und `N30` angewendet. Die Drehung wird nur auf `N20` angewendet, da sie von `transPop()` widerrufen wird. Die Abbildung "BeispielRevokingTransformations" zeigt die resultierende Bahn im Maschinenkoordinatensystem (MCS).
Beachten Sie, dass der Drehpunkt `[20,0,0]` im MCS (Maschinenkoordinatensystem) ihren Ursprung im Programmkoordinatensystem (PCS) nach der vorangegangenen Translation hat.

```
!transTranslate(20,0,0);
N10 X10 Y0 F6000
!transRotZ(90);
N20 X20 Y0
!transPop();
N30 X30 Y0
!transPop();
M02
```

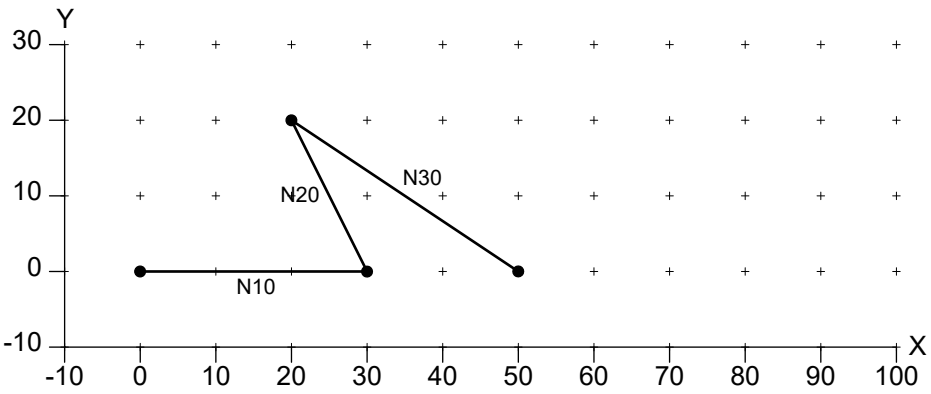


Abbildung "BeispielRevokingTransformations".

4.7.5 Wiederherstellung des Stapels

transRestore

`transRestore(depth)`

Die Funktion `transDepth()` gibt die aktuelle Tiefe des Stapels an. Die Funktion `transRestore(depth)` entfernt die Transformationen vom Stapel, bis die angegebene Tiefe erreicht ist. Normalerweise werden die beiden Funktionen kombiniert, um den Zustand des Transformationsstapels zu speichern und wiederherzustellen.

Es gehört zum guten Programmierstil, dieses Speichern und Wiederherstellen im Rahmen von benutzerdefinierten `ST`-Funktionen durchzuführen.

Beispiel:

In der folgenden Funktion wird zunächst die Tiefe des Stapels in der Variablen `depth` gespeichert. Am Ende der Funktion wird der Ausgangszustand durch `transRestore` wiederhergestellt. Beachten Sie, dass die Wiederherstellung nur richtig funktioniert, wenn die Stapeltiefe innerhalb der Funktion nicht unter `depth` fällt. Statt mit `transDepth()` und `transRestore()` könnte die Stapeltiefe auch mit `transPop()` wiederhergestellt werden. Es kann jedoch mühsam werden, das Schieben und Verschieben von Transformationen synchron zu halten, insbesondere wenn Transformationen unter bestimmten Bedingungen geschoben werden.

```
{
FUNCTION draw
VAR
    depth : UINT;
END_VAR
    depth := transDepth();
transTranslate(10,0,0);
    // … G-Code …
    transRotZ(45);
    // … G-Code …
    transMirrorX();
```

```
    // … G-Code …
    transRestore(depth);
END_FUNCTION
}
```

4.8 Error Reporting

Die effiziente Entwicklung von CNC-Programmen erfordert eine angemessene Unterstützung durch Entwicklungswerkzeuge. Diese Unterstützung umfasst die ordnungsgemäße Meldung von Programmierfehlern sowohl bei Kompilierzeit- als auch bei Laufzeitfehlern. Eine Fehlermeldung sollte direkt auf den Quellcode verweisen, aus dem der Fehler stammt, und eine genaue Beschreibung der Umstände enthalten, unter denen der Fehler aufgetreten ist (dynamische Daten). Solche individuellen Fehlermeldungen helfen einem Entwickler erheblich, Fehler in kurzer Zeit zu beheben. Der GST-Interpreter gibt solche Fehlermeldungen aus, wie in den folgenden Texten beschrieben.

4.8.1 Fehlermeldungen

Im Falle eines Fehlers gibt der Interpreter eine beschreibende Fehlermeldung aus. Eine Fehlermeldung besteht aus einer Quellcode-Koordinate und einer Beschreibung. Die Quellcode-Koordinate verbindet den Fehler mit seinem Ursprung im GST-Programm. Ein Bereich von Quellcode wird definiert, der vom ersten Zeichen des Codebereichs bis zum letzten Zeichen des Codebereichs reicht. Beide, das erste und das letzte Zeichen, sind durch ihre Datei, Zeile und Spalte definiert. Beachten Sie, dass das letzte Zeichen tatsächlich auf das erste Zeichen hinter dem Bereich zeigt, was eine übliche technische Konvention ist.

Beispiel:

Im folgenden Beispiel wird eine Integer-Variable `i` deklariert und initialisiert. Die Initialisierung verwendet ein Fließkomma-Literal. Da eine implizite Konvertierung von Fließkomma nach Integer in ST nicht erlaubt ist, erzeugt der Interpreter beim Laden des Programms die unten stehende beschreibende Fehlermeldung. Die Fehlermeldung meldet nicht nur, dass ein Typ-Fehler aufgetreten ist, sondern gibt auch die genaue Position an: Datei `aaa.nc`, Zeile `3`, Spalte `14` bis `17`. In diesem Codebereich wird das Literal '`1.5`' angezeigt. Darüber hinaus werden der programmierte Typ (`real`) und der erwartete Typ (`int`) gemeldet. Mit einer solchen detaillierten Fehlermeldung können Bugs vom Entwickler leicht behoben werden.

```
{
VAR
    i : int := 1.5;
END_VAR
}
M02
```

Fehlermeldung:

```
aaa.nc: 3.14-3.17: Invalid implicit conversion from type
'<real literal>' to 'int'.
```

4.8.2 Kompilierzeitfehler und Laufzeitfehler

Fehler können beim Laden des Programms (sog. Kompilierzeitfehler) oder bei der Programmausführung (sog. Laufzeitfehler) auftreten. Glücklicherweise werden die meisten Fehler bereits bei der Kompilierung erkannt. Diese Erkennung umfasst fehlende Dateien, Syntaxfehler, Typfehler und unerwartete Bezeichner. Der Entwickler erhält sofort eine Rückmeldung, wenn er versucht, das Programm zu laden. Auf diese Weise wird ein Teil der unerwarteten Ausfälle während der Bearbeitung vermieden.

Es gibt jedoch auch Fehler, die aufgrund ihrer Natur nicht zur Kompilierzeit erkannt werden können. Dazu gehört zum Beispiel eine Division durch Null, da der Divisor dynamisch berechnet werden kann. Wenn ein Laufzeitfehler auftritt, wird der Interpreter sicher gestoppt und eine Fehlermeldung ausgegeben. Eine Laufzeit-Fehlermeldung ist ähnlich wie eine Kompilierzeit-Fehlermeldung. Er enthält sogar einen Verweis auf den entsprechenden Quellcode.

Beispiel:

Im folgenden Beispiel enthält die FOR-Schleife eine Division von 10 durch die Schleifenvariable i. Da die Variable i von -3 nach 3 iteriert wird, führt dieses Programm bei der 4. Iteration zu einem Fehler, wenn i den Wert 0 hat. Dieser Fehler wird zur Laufzeit erkannt und hält den Interpreter an. Es wird die unten stehende Fehlermeldung angezeigt. Er zeigt genau auf den Code '10/i' im Beispiel.

FILE aaa.nc:

```
{
VAR
    i, j : int;
END_VAR

FOR i := -3 TO 3 DO
    j := j + 10/i;
END_FOR;
}
M02
```

Fehlermeldung:

```
aaa.nc: 7.12-7.16: Division by zero
```

Beispiel:

Zur Laufzeit führt der Interpreter auch eine Überprüfung der Array-Grenzen durch. Folglich führen ungültige Indizes nicht zu unvorhersehbaren und in der Regel fatalen Abstürzen. Die Laufzeit-Fehlermeldung definiert genau Ort und Ursprung des Fehlers unter 'idx', meldet den fehlerhaft gelieferten Index (20) und den gültigen Indexbereich (10..19).

FILE aaa.nc:

```
{
VAR
    idx : INT;
    a : ARRAY [10..19] OF INT;
END_VAR

FOR idx := 10 TO 20 DO
    a[idx] := i;
END_FOR;
}
M02
```

Fehlermeldung:

```
aaa.nc: 8.5-8.8: Out of bounds. 20 exceeds range 10..19.
```

4.8.3 Fehler im G-Code

Fehlermeldungen werden auch für Fehler im G-Code erstellt. Diese Meldungen umfassen Kompilier- und Laufzeitfehler. Zu den Laufzeitfehlern gehört die ungültige Verwendung von G-Code, z. B. eine falsche Definition für einen Kreis.

Beispiel:

Im folgenden Beispiel wird der Wert einer String-Variablen str dem Adressbuchstaben X des G-Codesatzes zugewiesen. Wie zuvor wird die Position des Fehlers unter '=str' im Code genau angegeben. Außerdem werden der programmierte Typ und der erwartete Typ angegeben.

FILE aaa.nc:

```
{
VAR
    str : string := 'Hello World';
END_VAR
}

G01 X=str F6000
```

```
G01 Y100
M02
```

Fehlermeldung:

```
7.5-7.9: Invalid implicit conversion from type 'string[255]'
to 'lreal'
```

Beispiel:

Im folgenden Beispiel wird eine Folge von Kreisbögen durch eine `FOR`-Schleife bearbeitet. Der Radius des Bogens ist `4`. Der Abstand zwischen dem Start- und dem Endpunkt des Bogens wird in jeder Iteration sukzessive vergrößert. Während der 9. Iteration überschreitet der Abstand den Kreisdurchmesser von `8`. Die angezeigte Fehlermeldung identifiziert den Ursprung `G2 X=i*10+i U4` und gibt Informationen über den Durchmesser und die Entfernung zwischen Start- und Endpunkt.

FILE aaa.nc:

```
{
VAR
    i : INT;
END_VAR

!G00 X0 Y0 Z0
FOR i := 1 TO 10 DO
    !G01 X=i*10 F6000
    !G02 X=i*10+i U4
END_FOR;
}
M02
```

Fehlermeldung:

```
aaa.nc: 9.4-10.1: Invalid definition of circle. Distance
between start-point and end-point (=9.000000) is larger than
diameter (=8.000000).
```

4.8.4 Vorverarbeitung

Bei der Vorverarbeitung werden `#include`-Direktiven durch den Inhalt der referenzierten Dateien ersetzt. Es wurde darauf geachtet, dass die Informationen über die Herkunft des Quellcodes ordnungsgemäß aufbewahrt werden. Daher bezieht sich ein Fehler, der durch Code in einer eingeschlossenen Datei verursacht wird, auf diese eingeschlossene Datei und nicht auf das Ergebnis der Vorverarbeitung, wie es eine einfache Implementierung tun würde.

Beispiel:

Im folgenden Beispiel inkludiert die Datei `aaa.nc` die Datei `bbb.nc`. In letzterer Datei werden die Variablen `i` und `j` in `G`-Codes verwendet. Die Variable `i` wird am Anfang von `aaa.nc` deklariert, die Variable `j` jedoch nicht. Daher wird die unten stehende Fehlermeldung ausgegeben. Wie Sie sehen können, verweist sie ordnungsgemäß auf die Verwendung der Variablen `j` in der Datei `bbb.nc`.

FILE aaa.nc:

```
{
VAR
    i : INT;
END_VAR
}

G00 X0 Y0 Z0

#include "bbb.nc"

G00 X100

M02
```

FILE bbb.nc:

```
G01 X=i F6000
G01 Y=j
G01 Z100
```

Fehlermeldung:

```
bbb.nc: 2.6-2.7: Undeclared variable or enumeration value
'j'
```

4.9 Allgemeine Befehlsübersicht

Präprozessor

Befehl	Beschreibung
#include [▶ 31]	Die #include-Direktive fügt den Inhalt einer anderen Datei ein. Die inkludierte Datei wird durch ihren Pfad referenziert. In der Regel wird er verwendet, um häufig verwendeten Code wie z. B. Bibliotheken zu "importieren". Das Verhalten ist dem des C-Präprozessors ähnlich.

Interpolationen

Befehl	Beschreibung	Modal oder Nichtmodal	Default
G00 [▶ 39]	Interpolationsmodus: Linear. Anwenden der Maximalgeschwindigkeit ohne Berücksichtigung der programmierten Geschwindigkeit. Zurücksetzen durch `G01`, `G02`, `G03`.	Modal.	Default.
G01 [▶ 40]	Interpolationsmodus: Linear. Anwenden der programmierten Geschwindigkeit. Zurücksetzen durch `G00`, `G02`, `G03`.	Modal.	Nein.
G02 [▶ 40]	Interpolationsmodus im Uhrzeigersinn: Kreisförmig oder spiralförmig. Anwendung der aktuellen Geschwindigkeit. Zurückgesetzt durch `G00`, `G01`, `G03`.	Modal.	Nein.
G03 [▶ 40]	Interpolationsmodus gegen den Uhrzeigersinn: Kreisförmig oder spiralförmig. Anwendung der aktuellen Geschwindigkeit. Zurückgesetzt durch `G00`, `G01`, `G02`.	Modal.	Nein.
G303 [▶ 42]	Mit G303 kann ein Kreisbogen (CIP-Kreis) programmiert werden, der frei im Raum platziert werden kann.		Nein
G04 [▶ 43]	Definiert eine Verweilzeit, d.h. unterbricht die Bearbeitung für eine bestimmte Dauer.	Nichtmodal.	Nein.

Auswahl der Arbeitsebene

Befehl	Beschreibung	Modal oder Nichtmodal	Default
G17 [▶ 46]	Wählt `XY`-Ebene als Arbeitsebene.	Modal.	Default.
G18 [▶ 46]	Wählt `ZX`-Ebene als Arbeitsebene.	Modal.	Nein.
G19 [▶ 46]	Wählt `YZ`-Ebene als Arbeitsebene.	Modal.	Nein.

Restweglöschen

Befehl	Beschreibung	Modal oder Nichtmodal	Default
G31 [▶ 44]	Restweglöschen.	Nichtmodal.	Nein.

Deaktivieren und Aktivieren der Werkzeugradiuskorrektur

Befehl	Beschreibung	Modal oder Nichtmodal	Default
G40 [▶ 35]	Deaktiviert die Werkzeugradiuskorrektur (TRC). Bei/nach einem G40-Befehl ist es zwingend erforderlich, mindestens ein Geometrieelement zu programmieren.	Modal.	Default.
G41 [▶ 35]	Aktiviert die Werkzeugradiuskorrektur (TRC). Links.	Modal.	Nein.
G42 [▶ 35]	Aktiviert die Werkzeugradiuskorrektur (TRC). Rechts.	Modal.	Nein.

Nullpunktverschiebungen einstellen, deaktivieren und aktivieren

Befehl	Beschreibung	Modal oder Nichtmodal	Default
G53 [▶ 44]	Deaktiviert jede Nullpunktverschiebung.	Modal.	Default.
G54..G57 [▶ 44]	Aktiviert die Nullpunktverschiebung, die mit dem angegebenen G-Code verbunden ist. Aktiviert die Translation G58 und G59.	Modal.	Nein.
G58, G59 [▶ 44]	Setzt die Translation, die mit dem angegebenen G-Code verbunden ist.	Modal.	Nein.

Befehl	Beschreibung
zeroOffsetShift Set(g:= USInt, [▶ 85] x:= LReal, [▶ 85] y:= LReal [▶ 85], z:= LReal) [▶ 85]	Setzt die Translation für G-Code g, wobei g eine der Zahlen 54, 55, 56 oder 57 sein muss.

Genauhalt

Befehl	Beschreibung	Modal oder Nichtmodal	Default
G09 [▶ 44]	Genauhalt.	Nichtmodal.	Nein.
G60 [▶ 44]	Genauhalt.	Modal.	Nein.

Einheit für Länge und Geschwindigkeit einstellen

Befehl	Beschreibung	Modal oder Nichtmodal	Default
G70 [▶ 47]	Setzt die Einheit für Längen auf Inch. Beeinflusst nicht die Einheit für Geschwindigkeit.	Modal.	Nein.
G71 [▶ 47]	Setzt die Einheit für Längen auf Millimeter. Beeinflusst nicht die Einheit für Geschwindigkeit.	Modal.	Default.
G700 [▶ 47]	Setzt die Einheit für Längen auf Inch. Gilt auch für die Interpretation von Geschwindigkeiten.	Modal.	Nein.
G710 [▶ 47]	Setzt die Einheit für Längen auf Millimeter. Gilt auch für die Interpretation von Geschwindigkeiten.	Modal.	Default.

Umschaltung zu absoluten oder relativen Koordinaten

Befehl	Beschreibung	Modal oder Nichtmodal	Default
G90 [▶ 49]	Schaltet auf absolute Programmierung um.	Modal.	Default.
G91 [▶ 49]	Schaltet auf inkrementelle Programmierung um.	Modal.	Nein.

IJK

Befehl	Beschreibung	Modal oder Nichtmodal	Default
I<vx> J<vy> K<vz> [▶ 40]	Der Mittelpunkt ist `currentPoint + [vx,vy,vz]`. Die aktuelle Längeneinheit wird für `vx,vy,vz` verwendet. `I` wird von `G4` verwendet und definiert eine Dauer.	Modal.	Default: Der Mittelpunkt ist `currentPoint + [0,0,0]`. Die `IJK`-Parameter sind optional.

M-Funktionen

Befehl	Beschreibung
M<v> [▶ 50]	Triggert die `M`-Funktion `v`. Das Zeitverhalten hängt von der Definition von `v` in der Entwicklungsumgebung von TwinCAT ab. Es darf nicht mehr als eine `M`-Funktion vom Typ Handshake in einem Satz geben.
M2	Vordefinierte `M`-Funktion. Signalisiert das Programmende.
M30	Vordefinierte `M`-Funktion. Signalisiert das Programmende.

Werkzeugausrichtung

Befehl	Beschreibung	Modal oder Nichtmodal	Default
P<v> [▶ 46]	Schaltet die Werkzeugausrichtung um.	Modal.	Nein.

Satznummer einstellen

Befehl	Beschreibung
N<v> [▶ 51]	Satznummer.

Radius einstellen

Befehl	Beschreibung	Modal oder Nichtmodal	Default
U<v> [▶ 40]	Setzt den Radius im Kontext von G02 oder G03 auf |v|.	Modal.	Nein.

Kartesische Koordinate einstellen

Befehl	Beschreibung
X<v> [▶ 51]	Setzt die X-Koordinate des nächsten Punktes auf v. Verwendet die aktuelle Längeneinheit für v.
Y<v> [▶ 51]	Setzt die Y-Koordinate des nächsten Punktes auf v. Verwendet die aktuelle Längeneinheit für v.
Z<v> [▶ 51]	Setzt die Z-Koordinate des nächsten Punktes auf v. Verwendet die aktuelle Längeneinheit für v.

Hilfsachsen

Befehl	Beschreibung	Modal oder Nichtmodal	Default
Q<i>=<v> [▶ 51]	Legt Bezeichner der Hilfsachse fest.	Modal.	Nein.

Ausrichtungswinkel einstellen

Befehl	Beschreibung	Modal oder Nichtmodal	Default
A<v> [▶ 51]	Setzt den A-Winkel der nächsten Ausrichtung auf v. Verwendet die aktuelle Längeneinheit für v.	Nichtmodal, kann aber nachfolgende Sätze beeinflussen.	Nein.
B<v> [▶ 51]	Setzt den B-Winkel der nächsten Ausrichtung auf v. Verwendet die aktuelle Längeneinheit für v.	Nichtmodal, kann aber nachfolgende Sätze beeinflussen.	Nein.
C<v> [▶ 51]	Setzt den C-Winkel der nächsten Ausrichtung auf v. Verwendet die aktuelle Längeneinheit für v.	Nichtmodal, kann aber nachfolgende Sätze beeinflussen.	Nein.

Kontrollstrukturen

Befehl	Beschreibung
IF-THEN-ELSIF-ELSE [▶ 59]	Bedingte Anweisung.
CASE OF [▶ 59]	Bedingte Anweisung.

Befehl	Beschreibung
FOR [▶ 59]	Zähler-gesteuerte Schleife.
WHILE [▶ 59]	Kopf-gesteuerte Schleife.
REPEAT [▶ 59]	Fuß-gesteuerte Schleife.
EXIT [▶ 59]	Verlassen einer Schleife.

Sprunganweisung

Befehl	Beschreibung
LABEL-GOTO [▶ 60]	Sprunganweisung.

Trigonometrisch

Befehl	Beschreibung
SIN(x) [▶ 62]	Liefert den Sinus von x; x in Radiant.
COS(x) [▶ 62]	Liefert den Cosinus von x; x in Radiant.
TAN(x) [▶ 62]	Liefert den Tangens von x; x in Radiant.
ASIN(x) [▶ 62]	Liefert den Arkussinus von x; x in Radiant.
ACOS(x) [▶ 62]	Liefert den Arkuscosinus von x; x in Radiant.
ATAN(x) [▶ 62]	Liefert den Arkustangens von x; x in Radiant.
ATAN2(y, x) [▶ 62]	Liefert den Arkustangens von y/x; y/x in Radiant.

Arithmetik

Befehl	Beschreibung
ABS(x) [▶ 62]	Liefert den Absolutwert von x.
SQRT(x) [▶ 62]	Liefert die Quadratwurzel von x.
LN(x) [▶ 62]	Liefert den natürlichen Logarithmus von x.
LOG(x) [▶ 62]	Liefert den dezimalen Logarithmus von x.
EXP(x) [▶ 62]	Liefert e hoch x.
ADD(x1, x2, ...) [▶ 62]	Liefert die Summe aller Parameter.
MUL(x1, x2, ...) [▶ 62]	Liefert das Produkt aller Parameter.
SUB(x, y) [▶ 62]	Liefert die Differenz x-y.
DIV(x, y) [▶ 62]	Liefert den Quotienten x/y.
MOD(x, y) [▶ 62]	Liefert den Rest der ganzzahligen Division x/y.
EXPT(x, y) [▶ 62]	Liefert x hoch y.

Rechenparameter

Befehl	Beschreibung
rSet(index := LINT, value := LREAL) [▶ 69]	Weist einen R-Parameterwert zu.
rGet(index := LINT) [▶ 69]	Extrahiert einen R-Parameterwert.

Bitverschiebung und Bitdrehung

Befehl	Beschreibung
SHL(x, y) [▶ 65]	Liefert den um y Bit nach links verschobenen Bitstring x.

Befehl	Beschreibung
SHR(x, y) [▶ 65]	Liefert den um `y` Bit nach rechts verschobenen Bitstring `x`.
ROL(x, y) [▶ 65]	Liefert den um `y` Bit nach links gedrehten Bitstring `x`.
ROR(x, y) [▶ 65]	Liefert den um `y` Bit nach rechts gedrehten Bitstring `x`.

Logische Operationen

Befehl	Beschreibung
AND(x1, x2, ...) [▶ 66]	Liefert das bitweise logische Und aller Parameter.
OR(x1, y2, ...) [▶ 66]	Liefert das bitweise logische Oder aller Parameter.
XOR(x1, x2, ...) [▶ 66]	Liefert das bitweise logische Exklusiv-Oder aller Parameter.
NOT(x) [▶ 66]	Liefert das bitweise Komplement von `x`.

Auswahl (Bedingte Ausdrücke)

Befehl	Beschreibung
SEL(cond, x1, x2) [▶ 67]	Liefert `x1` wenn `cond` = `FALSE`, ansonsten `x2`.
MUX(select, x0, x1, ... , xN) [▶ 67]	Liefert `x<select>`.

Min, Max und Grenzwert

Befehl	Beschreibung
MAX(x1, x2, ...) [▶ 67]	Liefert das Maximum aller Parameter.
MIN(x1, x2, ...) [▶ 67]	Liefert das Minimum aller Parameter.
LIMIT(min, in, max) [▶ 67]	Liefert `in`, wenn es in dem Intervall `[min,max]` liegt. Andernfalls wird die verletzte Grenze (`min` oder `max`) zurückgegeben.

Vergleich

Befehl	Beschreibung
GT(x, y) [▶ 68]	Liefert `TRUE`, wenn `x` größer ist als `y`.
GE(x, y) [▶ 68]	Liefert `TRUE`, wenn `x` nicht kleiner als `y` ist.
EQ(x, y) [▶ 68]	Liefert `TRUE`, wenn `x` und `y` gleich sind.
LE(x, y) [▶ 68]	Liefert `TRUE`, wenn `x` nicht größer ist als `y`.
LT(x, y) [▶ 68]	Liefert `TRUE`, wenn `x` kleiner als `y` ist.
NE(x, y) [▶ 68]	Liefert `TRUE`, wenn `x` und `y` nicht gleich sind.

Strings und Nachrichten

Befehl	Beschreibung
toString(<arg0>, ..., <argN>): STRING [▶ 70]	Konvertiert und verkettet die angegebenen Argumente zu einem String.
msg(str:= String[81]) [▶ 70]	Sendet die angegebene Nachricht an die Nachrichtenliste von TwinCAT.

Transformationen

Befehl	Beschreibung
transRotX(angle:= LReal) [▶ 72] transRotY(angle:= LReal) [▶ 72] transRotZ(angle:= LReal) [▶ 72]	Drehung um die jeweilige Achse um den angegebenen Winkel in der benutzerdefinierten Winkeleinheit.
transRotA(x:= LReal [▶ 72], y:= LReal, [▶ 72] z:= LReal [▶ 72], angle:= LReal) [▶ 72]	Drehen um den Vektor `[x,y,z]` mit dem angegebenen `angle`.
transMirrorX() [▶ 72] transMirrorY() [▶ 72] transMirrorZ() [▶ 72]	Spiegeln in Bezug auf die `X`-Richtung, `Y`-Richtung oder `Z`-Richtung relativ zum Ursprung des aktuellen PCS.
transScale(factor:= LReal) [▶ 72]	Skaliert das Koordinatensystem um den `factor` in der `X`-, `Y`- und `Z`-Dimension.
transScaleAxis(axisNo := axisIndex, factor := value) [▶ 72]	Skaliert die ausgewählte Bahnachse (axisNo) mit dem Faktor.
transTranslate(x:= LReal, [▶ 72] y:= LReal, [▶ 72] z:= LReal) [▶ 72]	Verschieben um Vektor `[x,y,z]`.
transPop() [▶ 72]	Entnimmt eine Transformation aus dem Stapel der Transformationen.
transDepth(): UInt [▶ 72]	Ergibt die Tiefe des Transformationsstapels, d. h. die Anzahl der aktiven Transformationen.
transRestore(depth:= UInt) [▶ 72]	Reduziert den Stapel der Transformationen auf die angegebene Tiefe.

Bewegung

Befehl	Beschreibung
moveCircle3d(cx:= LReal, [▶ 78] cy:= LReal, [▶ 78] cz:= LReal, [▶ 78] nx:= LReal [▶ 78], ny:= LReal [▶ 78], nz:= LReal [▶ 78], angle:= LReal [▶ 78], height:= LReal) [▶ 78]	Kreisbewegung durch Rotation um den Mittelpunkt `cx,cy,cz` und den Normalenvektor `nx,ny,nz` um den angegebenen `angle`. Wenn `height` ungleich Null ist, wird eine Helix beschrieben. Die Drehung wird gemäß der Rechte-Hand-Regel durchgeführt.

Mittelpunktskorrektur

Befehl	Beschreibung
centerpointCorrectionSet(on:= bool) [▶ 79]	Aktiviert die Mittelpunktskorrektur für Kreise. Wird für Kreise verwendet, die durch Mittelpunktprogrammierung definiert sind.
centerpointCorrectionLimitSet(li mit:= LReal) [▶ 79]	Legt die Genauigkeitsgrenze für den Mittelpunkt von Kreisen fest.

Werkzeuge

Befehl	Beschreibung
toolParamSet(tidx:= USInt, [▶ 79] col:= USInt [▶ 79], val:= LReal) [▶ 79]	Parameter des Werkzeugs `tidx` (1..255) auf `val` einstellen. Der Parameter ist durch `col` (0..15) gekennzeichnet.

Befehl	Beschreibung
toolParam(tidx:= USInt [▶ 79], col:= USInt): LReal [▶ 79]	Liefert den angegebenen Werkzeugparameter.
toolSet(index:= USInt [▶ 79], nr:= Int [▶ 79], tooltype:= ToolType, [▶ 79] length:= LReal [▶ 79], radius:= LReal, [▶ 79] lengthAdd:= LReal, [▶ 79] radiusAdd:= LReal [▶ 79], offsetX:= LReal [▶ 79], offsetY:= LReal [▶ 79], offsetZ:= LReal) [▶ 79]	Setzt alle Werkzeugparameter.
ToolType [▶ 79]	Aufzählung der Werkzeugtypen.

Werkzeugradiuskorrektur

Befehl	Beschreibung
trcApproachDepartSet(approachRadius:= LReal [▶ 83], approachAngle:= LReal [▶ 83], departRadius:= LReal, [▶ 83] departAngle:= LReal) [▶ 83]	Konfiguriert das An- und Abfahrverhalten so, dass ein Kreisbogen mit einem bestimmten Radius und Winkel verwendet wird.
trcOffsetSet(offset:= LReal) [▶ 83]	Konfiguriert den Umfang der Segmenterweiterung, die zum Schließen von Lücken verwendet wird.
trcLimitSet(offset:= LReal) [▶ 83]	Konfiguriert den Lookahead, der für die Kollisionsbeseitigung verwendet wird.
trcParam(): TrcParamType [▶ 83]	Liefert die aktuelle Konfiguration als Strukturwert.
trcParamSet(param:= TrcParamType) [▶ 83]	Konfiguriert die Werkzeugradiuskorrektur. Fasst `trcApproachDepartSet`, `trcOffsetSet` und `trcLimitSet` zusammen.
TrcParamType [▶ 83]	Struktur, die alle Konfigurationsparameter der Werkzeugradiuskorrektur enthält.
collisionElimination(nx:= LReal [▶ 83], ny:= LReal, [▶ 83] nz:= LReal [▶ 83], limit:= ULInt) [▶ 83]	Aktiviert die Kollisionsbeseitigung in Bezug auf die Ebene des Normalenvektors `nx`, `ny`, `nz`.
collisionEliminationFlush() [▶ 83]	Um Konflikte zwischen der Bahn, die dem Aufruf vorausgeht, und der Bahn, die dem Aufruf folgt, zu ignorieren.

Befehl	Beschreibung	Modal oder Nichtmodal	Default
G40 [▶ 35]	Deaktiviert die Werkzeugradiuskorrektur (TRC).	Modal.	Default.
G41 [▶ 35]	Aktiviert die Werkzeugradiuskorrektur (TRC). Links.	Modal.	Nein.
G42 [▶ 35]	Aktiviert die Werkzeugradiuskorrektur (TRC). Rechts.	Modal.	Nein.

Synchronisation

Befehl	Beschreibung
sync() [▶ 81]	Synchronisiert den Interpreter mit dem zugehörigen NC-Kanal.
wait() [▶ 81]	Wartet auf ein `GoAhead`-Signal von der SPS.

Abfrage der Achsen

Befehl	Beschreibung
queryAxes() [▶ 82]	Setzen Sie die MCS-Koordinaten des Interpreters auf die Ist-Koordinaten der physischen Achsen.

Aktueller Punkt

Befehl	Beschreibung
frameGet(x:= LReal [▶ 83], y:= LReal [▶ 83], z:= LReal [▶ 83], a:= LReal [▶ 83], b:= LReal [▶ 83], c:= LReal) [▶ 83]	Speichert den aktuellen Rahmen des PCS in `x`, `y`, `z` und `a`, `b`, `c`.
qAxisGet(q1:= LReal, [▶ 83] q2:= LReal [▶ 83], q3:= LReal, [▶ 83] q4:= LReal, [▶ 83] q5:= LReal) [▶ 83]	Speichert die aktuellen Werte der `Q`-Achsen `q1` bis `q5`.

Unterdrückung von G-Code-Sätzen

Befehl	Beschreibung
disableMask():= LWord [▶ 84]	Liefert den aktuellen Wert der Disable-Maske.
disableMaskSet(mask:= LWord) [▶ 84]	Setzt die interne Disable-Maske auf den angegebenen Wert.

Einheiten

Befehl	Beschreibung
unitAngleSet(unitAngle:= UnitAngle) [▶ 86]	Setzt die Einheit für Winkel auf `unitAngle`.
UnitAngle [▶ 86]	Aufzählung der Winkeleinheiten.
unitLengthSet(unitLength:= UnitLength) [▶ 86]	Setzt die Einheit für Längen auf `unitLength`.
UnitLength [▶ 86]	Aufzählung der Längeneinheiten.
unitTimeSet(unitTime:= UnitTime) [▶ 86]	Setzt die Einheit für die Zeit auf `unitTime`.
UnitTime [▶ 86]	Aufzählung der Zeiteinheiten.
unitVelocitySet(unitLength:= UnitLength, [▶ 86] unitTime:= UnitTime) [▶ 86]	Setzt die Einheit für die Geschwindigkeit auf `unitLength/unitTime`.

Trigonometrie (Unit Aware)

Befehl	Beschreibung
gSin(angle:= LReal) [▶ 87]	Liefert den Sinus des angegebenen `angle`, wobei die aktuelle Winkeleinheit zur Interpretation des Winkels verwendet wird.
gCos(angle:= LReal) [▶ 87]	Liefert den Cosinus des angegebenen `angle`, wobei die aktuelle Winkeleinheit zur Interpretation des Winkels verwendet wird.
gTan(angle:= LReal) [▶ 87]	Liefert den Tangens des angegebenen `angle`, wobei die aktuelle Winkeleinheit zur Interpretation des Winkels verwendet wird.
gASin(val:= LReal) [▶ 87]	Liefert den Arkussinus von `val` in der aktuellen Winkeleinheit.
gACos(val:= LReal) [▶ 87]	Liefert den Arkuscosinus von `val` in der aktuellen Winkeleinheit.
gATan(val:= LReal) [▶ 87]	Liefert den Arkustangens von `val` in der aktuellen Winkeleinheit.
gATan2(y:= LReal, x:= LReal) [▶ 87]	Liefert den Arkustangens von `y/x` in der aktuellen Winkeleinheit.

Vorschubmodus

Befehl	Beschreibung
feedModeSet(feedMode:= FeedModeType) [▶ 88]	Setzt den Vorschubmodus.
FeedModeType [▶ 88]	Aufzählung der Vorschubmodustypen.

Vorschubmodus

Befehl	Beschreibung
feedInterpolationSet(feedInterpolation:= FeedInterpolationType) [▶ 88]	Setzt die Vorschubinterpolation.
FeedInterpolationType [▶ 88]	Aufzählung der Vorschubinterpolationtypen.

Streaming von großen G-Code-Dateien

Befehl	Beschreibung
runFile(path:= string) [▶ 89]	Führt den `G`-Code aus, der in der `G`-Code-Datei enthalten ist, die durch `path` angegeben wird.

Vertex-Verschleifung

Befehl	Beschreibung
smoothingSet [▶ 90] (mainType:= SmoothingMainType [▶ 90], subType:= SmoothingSubType [▶ 90], value:= LReal) [▶ 90]	Legt das Verhalten der Vertex-Verschleifung fest.
SmoothingMainType [▶ 90]	Aufzählung der Haupttypen der Verschleifung.
SmoothingSubType [▶ 90]	Aufzählung der Untertypen der Verschleifung.
autoAccurateStopSet [▶ 92]	Automatischer Genauhalt.

Dynamischer Override

Befehl	Beschreibung
dynOverrideSet(value:= LReal) [▶ 96]	Setzt den dynamischen Override der Achsen auf die angegebene `value`.

Programmierreferenz

Befehl	Beschreibung
programmingReferenceSet [▶ 97]	Alternativ zu `G90/G91` kann mit `programmingReferenceSet` die Maßangabe der nachfolgenden Bewegungsbefehle auf absolute oder relative gesetzt werden.
programmingReferenceGet [▶ 97]	Mit `programmingReferenceGet` kann die aktuell aktive Maßangabe für die Bewegungsbefehle ausgelesen werden.
ReferenceType [▶ 97]	Aufzählung der Referenztypen.

Mittelpunkt-Referenz von Kreisen

Befehl	Beschreibung
circleCenterReferenceSet(value:= ReferenceType) [▶ 98]	Legt die Art der Mittelpunkt-Referenz für Kreise fest, die über `G02/G03` programmiert werden.
ReferenceType [▶ 98]	Aufzählung der Referenztypen.

Einstellung Dynamik

Befehl	Beschreibung
axisDynamicsSet [▶ 98]	Änderung der Achsdynamik.
pathDynamicsSet [▶ 99]	Änderung der Bahndynamik.

Spline-Interpolation

Befehl	Beschreibung
transBSpline [▶ 93]	Spline-Interpolation.

4.10 Vergleichende Befehlsübersicht

Befehl `Classic` Interpreter	Befehl `GST` Interpreter	Modal oder Nichtmodal	Beschreibung
ANG [▶ 150]		Nichtmodal.	Konturzugprogrammierung (Winkel).

Befehl `Classic` Interpreter	Befehl `GST` Interpreter	Modal oder Nichtmodal	Beschreibung
CalcInvRot [▶ 151]		Nichtmodal.	Berechnet die inverse Rotation eines Vektors.
CalcRot [▶ 151]		Nichtmodal.	Berechnet die Rotation eines Vektors.

Befehl Classic Interpreter	Befehl GST Interpreter	Modal oder Nichtmodal	Beschreibung
CDOF [▶ 196]	collisionElimination [▶ 83] Bereitstellung eines Null-Vektors.	Modal.	Flaschenhalserkennung aus.
CDON [▶ 196]	collisionElimination [▶ 83] Bereitstellung eines Vektors, der nicht Null ist.	Modal.	Flaschenhalserkennung ein.
	collisionEliminationFlush [▶ 83]		Diese Funktion kann während der aktiven Kollisionsbeseitigung aufgerufen werden, um etwaige Konflikte zwischen der Bahn vor dem Aufruf und der Bahn nach dem Aufruf zu ignorieren.

Befehl Classic Interpreter	Befehl GST Interpreter	Modal oder Nichtmodal	Beschreibung
CFC [▶ 195]	feedModeSet(feedMode:= FeedModeType) [▶ 88]	Modal.	Konstante Geschwindigkeit an der Kontur.
CFIN [▶ 195]	feedModeSet(feedMode:= FeedModeType) [▶ 88]	Modal.	Konstante Geschwindigkeit im Innenkreis.
CFTCP [▶ 195]	feedModeSet(feedMode:= FeedModeType) [▶ 88]	Modal.	Konstante Geschwindigkeit des Werkzeugmittelpunkts.

Befehl Classic Interpreter	Befehl GST Interpreter	Modal oder Nichtmodal	Beschreibung
CIP [▶ 141]	G303 [▶ 42]		Es kann ein Kreisbogen (CIP-Kreis) programmiert werden, der frei im Raum platziert werden kann.
	moveCircle3D [▶ 78]	Nichtmodal.	Kreisinterpolation. Kreisbewegung durch Rotation um den Mittelpunkt `cx,cy,cz` und den Normalenvektor `nx,ny,nz` um den angegebenen `angle`.
CPCOF [▶ 141]	centerpointCorrectionSet [▶ 78]	Modal.	Mittelpunktskorrektur aus.
CPCON [▶ 141]	centerpointCorrectionSet [▶ 78]	Modal.	Mittelpunktskorrektur ein.

Befehl Classic Interpreter	Befehl GST Interpreter	Modal oder Nichtmodal	Beschreibung
DelDTG [▶ 162]	G31 [▶ 44]	Nichtmodal.	Restweglöschen.

Befehl Classic Interpreter	Befehl GST Interpreter	Modal oder Nichtmodal	Beschreibung
DYNOVR [▶ 179]	dynOverrideSet [▶ 96]	Modal.	Dynamischer Override.

Befehl Classic Interpreter	Befehl GST Interpreter	Modal oder Nichtmodal	Beschreibung
FCONST [▶ 144]	feedInterpolationSet(feedInterpolation:= fiConstant) [▶ 89]	Modal.	Konstante Vorschubprogrammierung.
FLIN [▶ 144]	feedInterpolationSet(feedInterpolation:= fiLinear) [▶ 89]	Modal.	Lineare Vorschubprogrammierung.

Befehl Classic Interpreter	Befehl GST Interpreter	Modal oder Nichtmodal	Beschreibung
G00 [▶ 139]	G00 [▶ 39]	Modal.	Eilgang.
G01 [▶ 140]	G01 [▶ 40]	Modal.	Geradeninterpolation.
G02 [▶ 141]	G02 [▶ 40]	Modal.	Kreisinterpolation im Uhrzeigersinn.
G03 [▶ 141]	G03 [▶ 40]	Modal.	Kreisinterpolation gegen den Uhrzeigersinn.
G04 [▶ 143]	G04 [▶ 43]	Nichtmodal.	Verweilzeit.
G09 [▶ 144]	G09 [▶ 44]	Nichtmodal.	Genauhalt.

Befehl Classic Interpreter	Befehl GST Interpreter	Modal oder Nichtmodal	Beschreibung
G17 [▶ 132]	G17 [▶ 46]	Modal.	Ebenenanwahl XY.
G18 [▶ 132]	G18 [▶ 46]	Modal.	Ebenenanwahl ZX.
G19 [▶ 132]	G19 [▶ 46]	Modal.	Ebenenanwahl YZ.

Befehl Classic Interpreter	Befehl GST Interpreter	Modal oder Nichtmodal	Beschreibung
G40 [▶ 190]	G40 [▶ 35]	Modal.	Keine Fräser-/Schneidenradiuskorrektur.
G41 [▶ 190]	G41 [▶ 35]	Modal.	Fräser-/Schneidenradiuskorrektur links.
G42 [▶ 190]	G42 [▶ 35]	Modal.	Fräser-/Schneidenradiuskorrektur rechts.

Befehl Classic Interpreter	Befehl GST Interpreter	Modal oder Nichtmodal	Beschreibung
G53 [▶ 146]	G53 [▶ 44]	Modal.	Unterdrückung der Nullpunktverschiebung.
G54 [▶ 146]	G54 [▶ 44]	Modal.	1. einstellbare Nullpunktverschiebung.
G55 [▶ 146]	G55 [▶ 44]	Modal.	2. einstellbare Nullpunktverschiebung.
G56 [▶ 146]	G56 [▶ 44]	Modal.	3. einstellbare Nullpunktverschiebung.
G57 [▶ 146]	G57 [▶ 44]	Modal.	4. einstellbare Nullpunktverschiebung.
G58 [▶ 146]	G58 [▶ 44]	Modal.	1. programmierbare Nullpunktverschiebung.

Befehl Classic Interpreter	Befehl GST Interpreter	Modal oder Nichtmodal	Beschreibung
G59 [▶ 146]	G59 [▶ 44]	Modal.	2. programmierbare Nullpunktverschiebung.

Befehl Classic Interpreter	Befehl GST Interpreter	Modal oder Nichtmodal	Beschreibung
G60 [▶ 144]	G60 [▶ 44]	Modal.	Genauhalt.

Befehl Classic Interpreter	Befehl GST Interpreter	Modal oder Nichtmodal	Beschreibung
G70 [▶ 134]	G70 [▶ 47]	Modal.	Maßangabe in Inch.
G71 [▶ 134]	G71 [▶ 47]	Modal.	Maßangabe metrisch.
G74 [▶ 139]		Nichtmodal.	Referenzpunktanfahren per Programm.

Befehl Classic Interpreter	Befehl GST Interpreter	Modal oder Nichtmodal	Beschreibung
G90 [▶ 132]	G90 [▶ 49]	Modal.	Bezugsmaßangabe.
G91 [▶ 132]	G91 [▶ 49]	Modal.	Kettenmaßangabe.

Befehl Classic Interpreter	Befehl GST Interpreter	Modal oder Nichtmodal	Beschreibung
G700 [▶ 134]	G700 [▶ 47]	Modal.	Maßangabe in Inch mit Verrechnung des Vorschubs.
G710 [▶ 134]	G710 [▶ 47]	Modal.	Maßangabe metrisch mit Verrechnung des Vorschubs.

Befehl Classic Interpreter	Befehl GST Interpreter	Modal oder Nichtmodal	Beschreibung
MOD [▶ 162]		Nichtmodal.	Modulo-Bewegung.

Befehl Classic Interpreter	Befehl GST Interpreter	Modal oder Nichtmodal	Beschreibung
MSG [▶ 184]	msg [▶ 70]	Nichtmodal.	Nachricht aus dem NC-Programm.

Befehl Classic Interpreter	Befehl GST Interpreter	Modal oder Nichtmodal	Beschreibung
NORM [▶ 195]		Nichtmodal.	Orthogonales An- und Abfahren an der Kontur.

Befehl Classic Interpreter	Befehl GST Interpreter	Modal oder Nichtmodal	Beschreibung
P+ [▶ 132]	P<1> [▶ 51]	Modal.	Zustellrichtung positiv.
P- [▶ 132]	P<-1> [▶ 51]	Modal.	Zustellrichtung negativ.
paramAutoAccurateStop	autoAccurateStopSet [▶ 92]	Modal.	Automatischer Genauhalt.
paramAxisDynamics [▶ 179]	axisDynamicsSet [▶ 98]	Modal.	Parametrierung der Achsdynamik.

Befehl Classic Interpreter	Befehl GST Interpreter	Modal oder Nichtmodal	Beschreibung
paramC1ReductionFactor [▶ 180]		Modal.	C1 Reduktionsfaktor.
paramC2ReductionFactor [▶ 180]		Modal.	C2 Reduktionsfaktor.
paramCircularSmoothing [▶ 160]		Modal.	Verrundung.
paramDevAngle [▶ 180]		Modal.	C0 Reduktion - Ablenkungswinkel.
paramGroupVertex [▶ 160]		Modal.	Verrundung (alt).
paramGroupDynamic [▶ 179]		Modal.	Bahndynamik (alt).
paramPathDynamics [▶ 179]	pathDynamicsSet [▶ 99]	Modal.	Bahndynamik.
paramRadiusPrec [▶ 142]		Modal.	Kreisgenauigkeit.
paramSplineSmoothing [▶ 158]	smoothingSet [▶ 90]	Modal.	Vertex-Verschleifung. NC: Verschleifung mit Bezier-Splines.
paramVertexSmoothing [▶ 155]	smoothingSet [▶ 90]	Modal.	Glättung von Segmentübergängen.
	transBSpline [▶ 93]	Modal.	Spline-Interpolation.
paramVeloJump [▶ 180]		Modal.	C0 Reduktion - max. Geschwindigkeitssprung.
paramVeloMin [▶ 182]		Modal.	Minimale Geschwindigkeit.
paramZeroShift [▶ 146]	zeroOffsetShiftSet [▶ 146]	Modal.	Parametrierung der einstellbaren Nullpunktverschiebung.
PathAxesPos [▶ 182]	frameGet, [▶ 83] qAxisGet [▶ 83]	Nichtmodal.	Einlesen der Istposition.

Befehl Classic Interpreter	Befehl GST Interpreter	Modal oder Nichtmodal	Beschreibung
ZeroShiftIncOff [▶ 146]		Modal.	Nullpunktverschiebung wird unter G91 nicht herausgefahren.
ZeroShiftIncOn [▶ 146]		Modal.	Nullpunktverschiebung wird unter G91 herausgefahren.

Befehl Classic Interpreter	Befehl GST Interpreter	Modal oder Nichtmodal	Beschreibung
AROT [▶ 151]	transRotA [▶ 72]	Modal.	Rotation additiv.
ROT [▶ 151]	transRotX, transRotY, transRotZ [▶ 72]	Modal.	Rotation absolut.
RotExOff [▶ 151]		Modal.	Erweiterte Rotationsfunktion aus.
RotExOn [▶ 151]		Modal.	Erweiterte Rotationsfunktion ein.
RotVec [▶ 151]		Nichtmodal.	Berechnungsroutine zum Drehen eines Vektors.
RToDwordGetBit [▶ 136]		Modal.	Wandelt einen R-Parameter zum DWord und prüft, ob ein definiertes Bit gesetzt ist.

Befehl Classic Interpreter	Befehl GST Interpreter	Modal oder Nichtmodal	Beschreibung
Mirror [▶ 154]	transMirrorX, transMirrorY, transMirrorZ [▶ 72]	Modal.	Spiegeln des Koordinatensystems.
	transScale [▶ 72]	Modal.	Skaliert das Koordinatensystem um factor in der X-, Y- und Z-Dimension.
	transScaleAxis [▶ 72]	Modal.	Skaliert die ausgewählte Bahnachse (axisNo) mit dem Faktor.
	transDepth [▶ 72]	Nichtmodal.	Ergibt die Tiefe des Stapels von Transformationen.
	transRestore [▶ 72]	Modal.	Reduziert den Stapel der Transformationen auf die angegebene Tiefe.
	transPop [▶ 72]	Modal.	Entnimmt eine Transformation aus dem Stapel der Transformationen.

Befehl Classic Interpreter	Befehl GST Interpreter	Modal oder Nichtmodal	Beschreibung
rParam [▶ 136]	rSet(index := LINT, value := LREAL [▶ 69])	Nichtmodal.	Zuweisung eines Wertes an einen R-Parameter.
rParam [▶ 136]	rGet(index := LINT) [▶ 69]	Nichtmodal.	Lesen eines R-Parameterwerts.

Befehl Classic Interpreter	Befehl GST Interpreter	Modal oder Nichtmodal	Beschreibung
SEG [▶ 150]		Nichtmodal.	Konturzugprogrammierung (Segmentlänge).
skip VirtualMovements [▶ 183]		Modal.	Virtuelle Bewegungen überspringen.

Befehl Classic Interpreter	Befehl GST Interpreter	Modal oder Nichtmodal	Beschreibung
ToolOffsetIncOff [▶ 187]		Modal.	Kartesische Werkzeugverschiebung und Längenkorrektur wird unter G91 nicht herausgefahren.
ToolOffsetIncOn [▶ 187]		Modal.	Kartesische Werkzeugverschiebung und Längenkorrektur wird unter G91 herausgefahren.
ToolParam [▶ 184]	toolParamSet [▶ 79]	Modal.	Setzt ein Werkzeugparameter. NC: Schreiben und lesen von Werkzeugparametern.
ToolParam [▶ 184]	toolParam [▶ 79]	Modal.	Liefert den angegebenen Werkzeugparameter. NC: Schreiben und lesen von Werkzeugparametern.
ToolParam [▶ 184]	toolSet [▶ 79]	Modal.	Setzt alle Werkzeugparameter. NC: Schreiben und lesen von Werkzeugparametern.
TPM [▶ 148]		Nichtmodal.	Zielpositionsüberwachung.

Befehl Classic Interpreter	Befehl GST Interpreter	Beschreibung
	trcApproachDepartSet [▶ 83]	Konfiguriert das An- und Abfahrverhalten so, dass ein Kreisbogen mit einem bestimmten Radius und Winkel verwendet wird.
	trcOffsetSet [▶ 83]	Konfiguriert den Umfang der Segmenterweiterung, die zum Schließen von Lücken verwendet wird.
	trcLimitSet [▶ 83]	Konfiguriert den Lookahead, der für die Kollisionsvermeidung verwendet wird.
	trcParam [▶ 83]	Liefert die aktuelle Konfiguration als Strukturwert.
	trcParamSet [▶ 83]	Konfiguriert die Werkzeugradiuskorrektur.
	trcParamType [▶ 83]	Diese Struktur enthält alle Konfigurationsparameter der Werkzeugradiuskorrektur.

Befehl Classic Interpreter	Befehl GST Interpreter	Beschreibung
	queryAxes [▶ 82]	Setzen Sie die MCS-Koordinaten (Maschinenkoordinatensystem) des Interpreters auf die Ist-Koordinaten der physikalischen Achsen.

Befehl Classic Interpreter	Befehl GST Interpreter	Beschreibung
	disableMask [▶ 84]	Liefert den aktuellen Wert der Disable-Maske.
Block Skipping [▶ 129] /	disableMaskSet [▶ 84]	Setzt die interne Disable-Maske auf den angegebenen Wert.

Befehl Classic Interpreter	Befehl GST Interpreter	Beschreibung
L [▶ 177]	Benutzerdefinierte Funktionen [▶ 61]	Aufruf eines Unterprogramms.
	#include [▶ 31]	Direktive fügt den Inhalt einer anderen Datei ein. In der Regel wird er verwendet, um häufig verwendeten Code wie z. B. Bibliotheken zu "importieren".
	runFile [▶ 89]	Streaming von großen G-Code-Dateien.

Befehl Classic Interpreter	Varianten	Befehl GST Interpreter	Beschreibung
@40 [▶ 136]	`@40 Kn Rn Rm …`		Rette Register auf dem Stack.

Befehl Classic Interpreter	Varianten	Befehl GST Interpreter	Beschreibung
@41 [▶ 136]	@41 Rn Rm		Rette Register auf dem Stack.
@42 [▶ 136]	@42 Kn … Rm Rn		Restauriere Register vom Stack.
@43 [▶ 136]	@43 Rm Rn		Restauriere Register vom Stack.

Befehl Classic Interpreter	Varianten	Befehl GST Interpreter	Beschreibung
@100 [▶ 174]	@100 K±n @100 Rm	Sprunganweisung [▶ 60]	Unbedingter Sprung.

Befehl Classic Interpreter	Varianten	Befehl GST Interpreter	Beschreibung
@111 [▶ 174]	@111 Rn K/Rn Km …	CASE OF	Case-Satz.

Befehl Classic Interpreter	Varianten	Befehl GST Interpreter	Beschreibung
@121 [▶ 174]	@121 Rn K/Rn Kn	IF-THEN-ELSIF-ELSE;CASE OF [▶ 59]; GOTO [▶ 60]	Springe wenn ungleich.
@122 [▶ 174]	@122 Rn K/Rn Kn	IF-THEN-ELSIF-ELSE;CASE OF [▶ 59]; GOTO [▶ 60]	Springe wenn gleich.
@123 [▶ 174]	@123 Rn K/Rn Kn	IF-THEN-ELSIF-ELSE;CASE OF [▶ 59]; GOTO [▶ 60]	Springe wenn kleiner gleich.
@124 [▶ 174]	@124 Rn K/Rn Kn	IF-THEN-ELSIF-ELSE;CASE OF [▶ 59]; GOTO [▶ 60]	Springe wenn kleiner.
@125 [▶ 174]	@125 Rn K/Rn Kn	IF-THEN-ELSIF-ELSE;CASE OF [▶ 59]; GOTO [▶ 60]	Springe wenn größer gleich.
@126 [▶ 174]	@126 Rn K/Rn Kn	IF-THEN-ELSIF-ELSE;CASE OF [▶ 59]; GOTO [▶ 60]	Springe wenn größer.

Befehl Classic Interpreter	Varianten	Befehl GST Interpreter	Beschreibung
@131 [▶ 175]	@131 Rn K/Rn Kn	WHILE [▶ 59]	Schleife solange gleich.
@132 [▶ 175]	@132 Rn K/Rn Kn	WHILE [▶ 59]	Schleife solange ungleich.
@133 [▶ 175]	@133 Rn K/Rn Kn	WHILE [▶ 59]	Schleife solange größer.
@134 [▶ 175]	@134 Rn K/Rn Kn	WHILE [▶ 59]	Schleife solange größer oder gleich.
@135 [▶ 175]	@135 Rn K/Rn Kn	WHILE [▶ 59]	Schleife solange kleiner.
@136 [▶ 175]	@136 Rn K/Rn Kn	WHILE [▶ 59]	Schleife solange kleiner oder gleich.

Befehl Classic Interpreter	Varianten	Befehl GST Interpreter	Beschreibung
@141 [▶ 175]	@141 Rn K/Rn Kn	REPEAT [▶ 59]	Wiederhole bis gleich.
@142 [▶ 175]	@142 Rn K/Rn Kn	REPEAT [▶ 59]	Wiederhole bis ungleich.

Befehl Classic Interpreter	Varianten	Befehl GST Interpreter	Beschreibung
@143 [▶ 175]	@143 Rn K/Rn Kn	REPEAT [▶ 59]	Wiederhole bis größer.
@144 [▶ 175]	@144 Rn K/Rn Kn	REPEAT [▶ 59]	Wiederhole bis größer oder gleich.
@145 [▶ 175]	@145 Rn K/Rn Kn	REPEAT [▶ 59]	Wiederhole bis kleiner.
@146 [▶ 175]	@146 Rn K/Rn Kn	REPEAT [▶ 59]	Wiederhole bis kleiner oder gleich.

Befehl Classic Interpreter	Varianten	Befehl GST Interpreter	Beschreibung
@151 [▶ 175]	@151 Rn K/Rn Kn	FOR [▶ 59]	FOR_TO Schleife.

Befehl Classic Interpreter	Varianten	Befehl GST Interpreter	Beschreibung
@161 [▶ 175]	@161 Rn K/Rn Kn	FOR [▶ 59]	FOR_DOWNTO Schleife.

Befehl Classic Interpreter	Varianten	Befehl GST Interpreter	Beschreibung
@200	@200 Rn		Löschen einer Variablen.
@202	@202 Rn Rm		Vertauschen von zwei Variablen.

Befehl Classic Interpreter	Varianten	Befehl GST Interpreter	Beschreibung
@302	@302 K/R/Pn K/R/Pn R/Pn		Lese Maschinendatenbit.
@361 [▶ 182]	@361 Rn Km		Lese maschinenbezogenen Achsen-Istwert.
@372	@372 Rn		Extrahiere die NC-Channel-ID und in einer Variablen speichern.

Befehl Classic Interpreter	Varianten	Befehl GST Interpreter	Beschreibung
@402 [▶ 141]	@402 K/R/Pn K/R/Pn K/R/Pn	circleCenterReferenceSet [▶ 98]	Schreibe Maschinendatenbit.
		circleCenterReferenceGet [▶ 98]	Gibt die aktuelle Mittelpunktreferenz für Kreise zurück.

Befehl Classic Interpreter	Varianten	Befehl GST Interpreter	Beschreibung
@610	@610 Rn Rn	ABS [▶ 62]	Absolutwert einer Variablen ermitteln.
@613	@613 Rn Rn	SQRT [▶ 62]	Quadratwurzel einer Variablen ermitteln.
@614	@614 Rn Rm Rm	SQRT(a^2 + b^2)	Quadratwurzel der Summe der Quadrate von zwei Variablen ermitteln ! x := sqrt(a^2 + b^2);.

Befehl Classic Interpreter	Varianten	Befehl GST Interpreter	Beschreibung
@620 [▶ 175]	@620 Rn	!var := var+1;	Variable inkrementieren.
@621	@621 Rn	!var := var-1;	Variable dekrementieren.

Befehl Classic Interpreter	Varianten	Befehl GST Interpreter	Beschreibung
@622	@622 Rn		Ganzzahl einer Variablen ermitteln.

Befehl Classic Interpreter	Varianten	Befehl GST Interpreter	Beschreibung
@630 [▶ 136]	@630 Rn Rm	SIN [▶ 62]	Sinus einer Variablen ermitteln.
@630 [▶ 136]	@630 Rn Rm	gSin [▶ 87]	Sinus einer Variablen ermitteln.
@631 [▶ 136]	@631 Rn Rm	COS [▶ 62]	Cosinus einer Variablen ermitteln.
@631 [▶ 136]	@631 Rn Rm	gCos [▶ 87]	Cosinus einer Variablen ermitteln.
@632 [▶ 136]	@632 Rn Rm	TAN [▶ 62]	Tangens einer Variablen ermitteln.
@632 [▶ 136]	@632 Rn Rm	gTan [▶ 87]	Tangens einer Variablen ermitteln.
@633 [▶ 136]	@633 Rn Rm		Cotangens einer Variablen ermitteln.
@634 [▶ 136]	@634 Rn Rm	ASIN [▶ 62]	Arkussinus einer Variablen ermitteln.
@634 [▶ 136]	@634 Rn Rm	gASin [▶ 87]	Arkussinus einer Variablen ermitteln.
@635 [▶ 136]	@635 Rn Rm	ACOS [▶ 62]	Arkuscosinus einer Variablen ermitteln.
@635 [▶ 136]	@635 Rn Rm	gACos [▶ 87]	Arkuscosinus einer Variablen ermitteln.
@636 [▶ 136]	@636 Rn Rm	gATan [▶ 87]	Arkustangens einer Variablen ermitteln.
		gATan2 [▶ 87]	Liefert den Arkustangens von y/x.

Befehl Classic Interpreter	Varianten	Befehl GST Interpreter	Beschreibung
@714 [▶ 172]	@714	sync() [▶ 81]	Dekodierstopp.
@716 [▶ 172]	@716	Eine Kombination aus sync() [▶ 81]und queryAxes() [▶ 82] ersetzt den früheren @716-Befehl.	Dekodierstopp mit Rescan der Achspositionen.
@717 [▶ 172]	@717	wait() [▶ 81]	Dekodierstopp mit externem Triggerevent.

Befehl Classic Interpreter	Befehl GST Interpreter	Beschreibung
	LN(x) [▶ 62]	Liefert den natürlichen Logarithmus von x.
	LOG(x) [▶ 62]	Liefert den dezimalen Logarithmus von x.
	EXP(x) [▶ 62]	Liefert e hoch x.
	ADD(x1, x2, ...) [▶ 62]	Liefert die Summe aller Parameter.
	MUL(x1, x2, ...) [▶ 62]	Liefert das Produkt aller Parameter.
	SUB(x, y) [▶ 62]	Liefert die Differenz x-y.
	DIV(x, y) [▶ 62]	Liefert den Quotienten x/y.
	MOD(x, y) [▶ 62]	Liefert den Rest der ganzzahligen Division x/y.

Befehl Classic Interpreter	Befehl GST Interpreter	Beschreibung
	EXPT(x, y) [▶ 62]	Liefert x hoch y.

Befehl Classic Interpreter	Befehl GST Interpreter	Beschreibung
	GT(x, y) [▶ 68]	Liefert TRUE, wenn x größer ist als y.
	GE(x, y) [▶ 68]	Liefert TRUE, wenn x nicht kleiner als y ist.
	EQ(x, y) [▶ 68]	Liefert TRUE, wenn x und y gleich sind.
	LE(x, y) [▶ 68]	Liefert TRUE, wenn x nicht größer ist als y.
	LT(x, y) [▶ 68]	Liefert TRUE, wenn x kleiner als y ist.
	NE(x, y) [▶ 68]	Liefert TRUE, wenn x und y nicht gleich sind.

Befehl Classic Interpreter	Befehl GST Interpreter	Beschreibung
	<nativeType>_to_<nativeType>(x), [▶ 62] to_<nativeType>(x) [▶ 62]	Explizite Konvertierung zwischen den angegebenen nativen Typen.

Logische Operationen

Befehl Classic Interpreter	Befehl GST Interpreter	Beschreibung
	AND(x1, x2, ...) [▶ 66]	Liefert das bitweise logische Und aller Parameter.
	OR(x1, y2, ...) [▶ 66]	Liefert das bitweise logische Oder aller Parameter.
	XOR(x1, x2, ...) [▶ 66]	Liefert das bitweise logische Exklusiv-Oder aller Parameter.
	NOT(x) [▶ 66]	Liefert das bitweise Komplement von x.

Min, Max und Grenzwert

Befehl Classic Interpreter	Befehl GST Interpreter	Beschreibung
	MAX(x1, x2, ...) [▶ 67]	Liefert das Maximum aller Parameter.
	MIN(x1, x2, ...) [▶ 67]	Liefert das Minimum aller Parameter.
	LIMIT(min, in, max [▶ 67])	Liefert in, wenn es in dem Intervall [min,max] liegt. Andernfalls wird die verletzte Grenze (min oder max) zurückgegeben.

Drehung

Befehl Classic Interpreter	Befehl GST Interpreter	Beschreibung
	ROL(x, y) [▶ 65]	Liefert den um y Bit nach links gedrehten Bitstring x.
	ROR(x, y) [▶ 65]	Liefert den um y Bit nach rechts gedrehten Bitstring x.

Auswahl (Bedingte Ausdrücke)

Befehl Classic Interpreter	Befehl GST Interpreter	Beschreibung
	SEL(cond, x1, x2) [▶ 67]	Liefert x1 wenn cond = FALSE, ansonsten x2.

Befehl `Classic` Interpreter	Befehl `GST` Interpreter	Beschreibung
	MUX(select, x0, x1, ... , xN) [▶ 67]	Liefert `x<select>`.

Verschiebung

Befehl `Classic` Interpreter	Befehl `GST` Interpreter	Beschreibung
	SHL(x, y) [▶ 65]	Liefert den um `y` Bit nach links verschobenen Bitstring `x`.
	SHR(x, y) [▶ 65]	Liefert den um `y` Bit nach rechts verschobenen Bitstring `x`.

Einheiten

Befehl `Classic` Interpreter	Befehl `GST` Interpreter	Beschreibung
	unitAngleSet(unitAngle:= UnitAngle [▶ 86])	Setzt die Einheit für Winkel auf `unitAngle`.
	UnitAngle [▶ 86]	Aufzählung der Winkeleinheiten.
	unitLengthSet(unitLength:= UnitLength) [▶ 86]	Setzt die Einheit für Längen auf `unitLength`.
	UnitLength [▶ 86]	Aufzählung der Längeneinheiten.
	unitTimeSet(unitTime:= UnitTime) [▶ 86]	Setzt die Einheit für die Zeit auf `unitTime`.
	UnitTime [▶ 86]	Aufzählung der Zeiteinheiten.
	unitVelocitySet(unitLength:= UnitLength, unitTime:= UnitTime) [▶ 86]	Setzt die Einheit für die Geschwindigkeit auf `unitLength/unitTime`.

5 Classic Dialect Reference Manual

5.1 Grundlagen der NC-Programmierung

5.1.1 Aufbau eines NC-Programms

Ein NC-Programm ist ein Text, der normalerweise als Kette von ASCII-Codes in einer Datei auf der Festplatte abgelegt ist. Es besteht aus einer Reihe von NC-Sätzen, die jeweils durch eine Zeilenschaltung (Return) getrennt sind. In der Regel wird es bei der Abarbeitung Zeichen für Zeichen und Zeile für Zeile interpretiert und abgearbeitet.

Programmaufbau

Dabei ist das NC-Programm aus drei Teilen zusammengesetzt

- Programmanfang (optional)·
- Anzahl von Sätzen
- Programmende

Programmanfang

Am Anfang eines NC-Programms kann das Zeichen „%“ für den Programmanfang stehen. Hinter diesem Zeichen befindet sich dann der Name des Programms. Der Satz für den Programmanfang muss nicht zwingend programmiert werden.

Beispiel:

```
% Test1 (program start)
N10 G0 X100 Y100 Z0
M30 (program end)
```

NC-Satz

Jeder der NC-Sätze besteht aus keinem (Leerzeile), einem oder mehreren NC-Worten, die durch Leerzeichen oder Tabulator getrennt sind. Daher darf innerhalb eines Wortes kein Leerzeichen verwendet werden.

Beispiel:

```
N10 G0 X100 Y100 Z0
```

NC-Wort

Das erste Zeichen des NC-Wortes legt seine Bedeutung fest. Es handelt sich dabei um einen Buchstaben oder ein Sonderzeichen.

Groß- / Kleinschreibung ist in der Regel nicht von Bedeutung. Allerdings ist eine einheitliche Großschreibung wegen der besseren Lesbarkeit empfehlenswert. Die optional folgenden Zeichen spezifizieren die Bedeutung genauer oder liefern Parameter für die Ausführung.

Um mit dem begrenzten Vorrat an Zeichen auszukommen, ist nicht für jede Variante jeder Funktion ein eigener Ausdruck verfügbar. Vielmehr wird die Bedeutung und Wirkung vieler NC-Worte durch den Zusammenhang mitbestimmt. Das kann sich auf die im Satz vorausgehenden NC-Worte beziehen, aber auch auf die vorausgehenden NC-Sätze. In einigen Fällen wird die Wirkung von NC-Worten sogar durch Maschinendaten beeinflusst.

Programmende

Das Programmende wird durch eine M-Funktion gekennzeichnet. Hierfür wird entweder M2 oder M30 verwendet.

Wirkungsdauer von Wörtern

Befehle, wie z.B. G0 [▶ 139], G17 [▶ 132] die über das Satzende hinaus wirken, werden laut DIN 66025 als **modal** bezeichnet. Diese Befehle wirken solange, bis sie durch einen anderen Befehl aufgehoben oder geändert werden.

Kommentare

Sollen Teilbereiche eines NC-Satzes oder der ganze Satz nicht interpretiert werden, ist der Bereich in runde Klammern zu setzen.

Beispiel:

```
N10 G0 X100 (comment)
```

Hinweis Ein Kommentar endet mit der schließenden Klammer, spätestens jedoch am Satzende. Ein Kommentar kann sich also nicht über mehrere Zeilen erstrecken. Eine Schachtelung von Kommentaren ist ebenfalls nicht möglich.

Satznummer

Jeder Satz kann mit einer Satznummer gekennzeichnet werden. Die Satznummer wird mit einem "N" für Nebensätze und mit ":" für Hauptsätze gekennzeichnet.

Hinweis Die Satznummer ist nicht zwingend erforderlich. Ein nicht mit einer Satznummer gekennzeichneter Satz kann jedoch nicht als Ziel für Sprungbefehle dienen. Außerdem kann bei einer Fehlermeldung der Ort des Fehlers nur ungenau (letzte Satznummer) angegeben werden.

5.1.2 Satzunterdrückung

Oft ist es nützlich, nicht immer alle Sätze eines Programms auszuführen. Dadurch ist es möglich, ähnliche Bearbeitungen mit einem einzigen Programm zu verwirklichen.

In einem solchen Fall werden die Sätze, die zu einer Variante gehören mit einer Satzausblendungs-Kennung versehen. Diese muss an den Satzanfang geschrieben werden und besteht aus einem Schrägstrich "/".

Sind mehrere Varianten erforderlich, so wird der Schrägstrich um eine Zahlenangabe (0..15) ergänzt, also z.B. "/12". Die Zahlenangabe (dabei ist "/" gleichzusetzen mit "/0") wählt ein Bit aus einem Wort des Kanalinterface von der SPS zur NC aus. Ist dieses Bit **gesetzt**, wird der Satz nicht interpretiert.

In der NC wird hierfür die Variable *'mSkipLine'* ausgewertet, die sich unter den Eingängen im zyklischen Kanalinterface befindet. Das entsprechende Gegenstück in der SPS ist bei den Ausgängen unter 'nSkipLine' [▶ 335] *(früher: nSatzunterdrückung)* zu finden (vergl. TwinCAT PLC Library: NCI Interpreter).

Soll eine bestimmte von mehreren Varianten aktiv sein, sind alle anderen Unterdrückungen zu setzen. Dann sind nur noch Sätze aktiv, die keine oder die gewünschte Kennung tragen.

Wirkzeitpunkt der Satzunterdrückung

Der Interpreter läuft gegenüber der Ausführung eine variable Anzahl von Sätzen voraus. Die Satzunterdrückung kann nur dann korrekt wirken, wenn sie früh genug (eventuell schon vor Programm-Start) gesetzt wird oder der Interpreter an einer geeigneten Stelle des Programms mit der Ausführung synchronisiert wird (Dekodierstopp [▶ 172]).

5.1.3 Look-Ahead

Die tatsächliche Geschwindigkeit im Segmentübergang ist von vielen Parametern abhängig. Dazu zählen u.a. Bahnrestweg, Dynamikparameter des aktuellen Segments und indirekt der geometrische Winkel im Segmentübergang.

Der Dynamik-Look-Ahead (im Folgenden einfacher als Look-Ahead bezeichnet) sorgt dafür, dass die Geschwindigkeit an Segmentübergängen möglichst hoch bleiben kann. Dabei werden in der Standardkonfiguration 128 Geometrie-Einträge berücksichtigt.

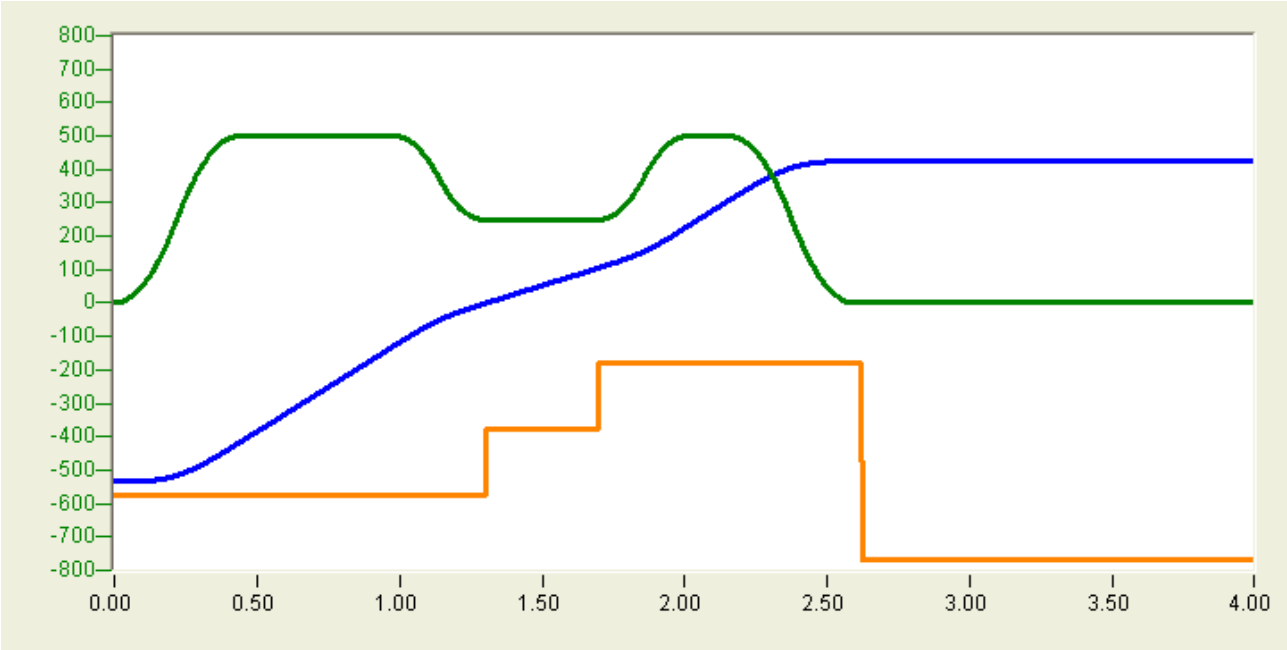
Ohne Look-Ahead wird die Geschwindigkeit an jedem Segmentübergang auf 0 reduziert (G60).

Die Anzahl der berücksichtigten Geometrie-Einträge kann in den DXD-Parametern [▶ 27] eingestellt werden.

Segmente mit unterschiedlichen Zielgeschwindigkeiten

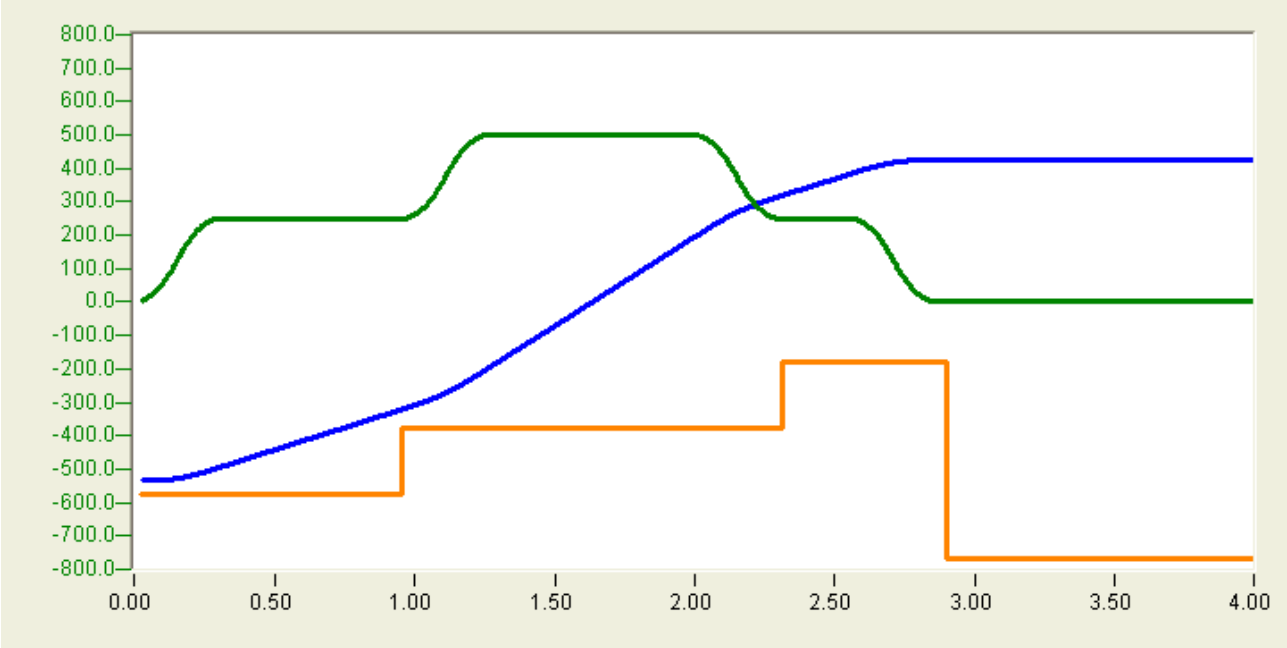
Ändert sich die Zielgeschwindigkeit von einem hohen Geschwindigkeitsniveau auf ein niedrigeres (N10 -> N20), so ist die niedrigere Geschwindigkeit zu Beginn des Segments bereits erreicht.

Ändert sich die Zielgeschwindigkeit von einem niedrigen Geschwindigkeitsniveau auf ein höheres (N20 -> N30), so wird die höhere Geschwindigkeit mit dem Segmentübergang eingeleitet. D.h. es ist immer sichergestellt, dass auch am Rande des Segments die aktuelle Geschwindigkeit nicht höher wird, als die programmierte.



grün: Bahngeschwindigkeit
blau: Position
orange: Satznummer

```
N10 G01 X600 F30000
N20 G01 X700 F15000
N30 G01 X900 F30000
M30
```



grün: Bahngeschwindigkeit
blau: Position
orange: Satznummer

```
N40 G01 X200 F15000
N50 G01 X800 F30000
N60 G01 X900 F15000
M30
```

5.1.4 Glättung von Segmentübergängen

Übersicht

Segmentübergänge die nicht zweimal stetig differenzierbar sind, führen zu Unstetigkeiten in der Dynamik, wenn die Bahngeschwindigkeit dort nicht auf *0* abgesenkt wird. Um mit endlicher Geschwindigkeit ohne dynamische Unstetigkeiten den Segmentübergang zu passieren, gibt es die Möglichkeit, die Segmentübergänge mittels Bezier-Splines so zu glätten, dass die Geometrie lokal verändert und damit die Gesamtbahn zweimal stetig differenzierbar wird.

Toleranzkugel

Zur Glättung wird um jeden Segmentübergang eine Toleranzkugel gelegt, innerhalb der die Bahn von ihrer vorgegebenen Geometrie abweichen darf, jedoch nur soweit, dass sie in der Toleranzkugel bleibt. Der Radius der Toleranzkugel (Parametrierung [▶ 330]) wird durch den Benutzer vorgegeben und gilt modal für alle Segmentübergänge die keinen Genauhalt oder Stopp im Segmentübergang implizieren. Die Radien der Toleranzkugeln werden automatisch adaptiv gesetzt indem verhindert wird, dass sich - bei kleinen Segmenten - Toleranzkugeln überlappen.

Dynamikparameter

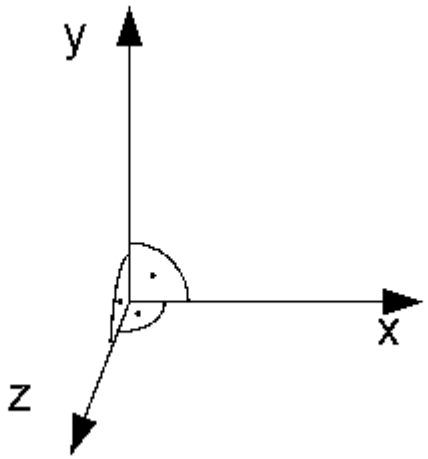
Die Glättung erlaubt eine schnellere Dynamik. Die vom System vorberechnete maximale Segmentübergangsgeschwindigkeit *VeloLink* kann vom Benutzer insofern beeinflusst werden, als der Systemparameter C2-Geschwindigkeitsreduktion *C2* (Parametrierung [▶ 330]) die Segmentübergangsgeschwindigkeit auf *C2x VeloLink* setzt. Der Faktor ist online änderbar.

Verhalten an Segmentübergängen

Bei Eintritt in die Toleranzkugel ist die Bahnbeschleunigung *0* und die Bahngeschwindigkeit gleich der Segmentüberganggeschwindigkeit. Das wird innerhalb der Toleranzkugel beibehalten. In der Toleranzkugel ist der Override nicht aktiv, d.h. die durch den Override bedingte Änderung des Geschwindigkeitsniveaus wird in der Toleranzkugel unterbrochen und nach dem Austritt aus der Toleranzkugel fortgesetzt.

5.1.5 Koordinatensystem

Die Bezeichnungen für die Achsen einer Werkzeugmaschine werden durch die DIN 66217 festgelegt. Dabei werden die Buchstaben X, Y und Z für Achsen vergeben. Diese bilden ein rechtsdrehendes und rechtwinkliges (kartesisches) Koordinatensystem. Bei vielen Maschinen sind nicht in jeder Einrichtung drei Achsen vorhanden. In diesen Fällen werden einzelne dieser Buchstaben sinngemäß vergeben und nicht vorhandene Achsen übersprungen.



5.1.6 Maßangaben

Maßangaben können sich wahlweise auf einen absoluten Punkt oder auf den aktuellen Sollwert beziehen.

Absolutmaßangabe

Befehl	G90 (Standardeinstellung)
Aufhebung	G91

Bei der Absolutmaßangabe beziehen sich alle Positionsangaben immer auf den gerade gültigen Nullpunkt.

Für die Werkzeugbewegung bedeutet das, dass mit der Absolutmaßangabe die Positionen beschrieben werden, die das Werkzeug anfahren soll.

Kettenmaß

Befehl	G91
Aufhebung	G90

Bei der Kettenmaßangabe bezieht sich eine Positionsangabe auf den jeweils vorherigen Punkt. Dabei werden neben den Bahnachsen auch die Hilfsachsen (Q1..Q5) berücksichtigt.

Für die Werkzeugbewegung bedeutet das, dass mit der Kettenmaßangabe beschrieben wird, um wie viel das Werkzeug verfahren werden soll.

Einheiten

In der folgenden Tabelle werden die Einheiten für Längen, Winkel etc. beschrieben:

	Einheit
Positionen und Längen	mm
Winkel	Grad
Zeiten	sec
Vorschub	mm/min

5.1.7 Arbeitsebene und Zustellrichtung

Für die Beschreibung von Kreisen (außer CIP [▶ 141]), sowie für die Fräserradius [▶ 190]- und Werkzeuglängenkorrektur [▶ 187] ist eine Festlegung der Arbeitsebene erforderlich.

Arbeitsebene XY

Befehl	G17 (Standardeinstellung)
Aufhebung	G18 oder G19

Die Funktion G17 legt die Arbeitsebene auf die XY-Ebene und die Zustellrichtung in Z-Richtung fest.

Die Funktion wirkt als:

- Ebene für Werkzeugradiuskorrektur [▶ 190]
- Zustellrichtung für Werkzeuglängenkorrektur [▶ 187] (Offset)
- Ebene für die Kreisinterpolation

Wechsel der Arbeitsebene

Bei aktiver Werkzeugkorrektur kann die Arbeitsebene nicht gewechselt werden.

Arbeitsebene ZX

Befehl	G18
Aufhebung	G17 oder G19

Die Funktion G18 legt die Arbeitsebene auf die ZX-Ebene und die Zustellrichtung in Y-Richtung fest.

Arbeitsebene YZ

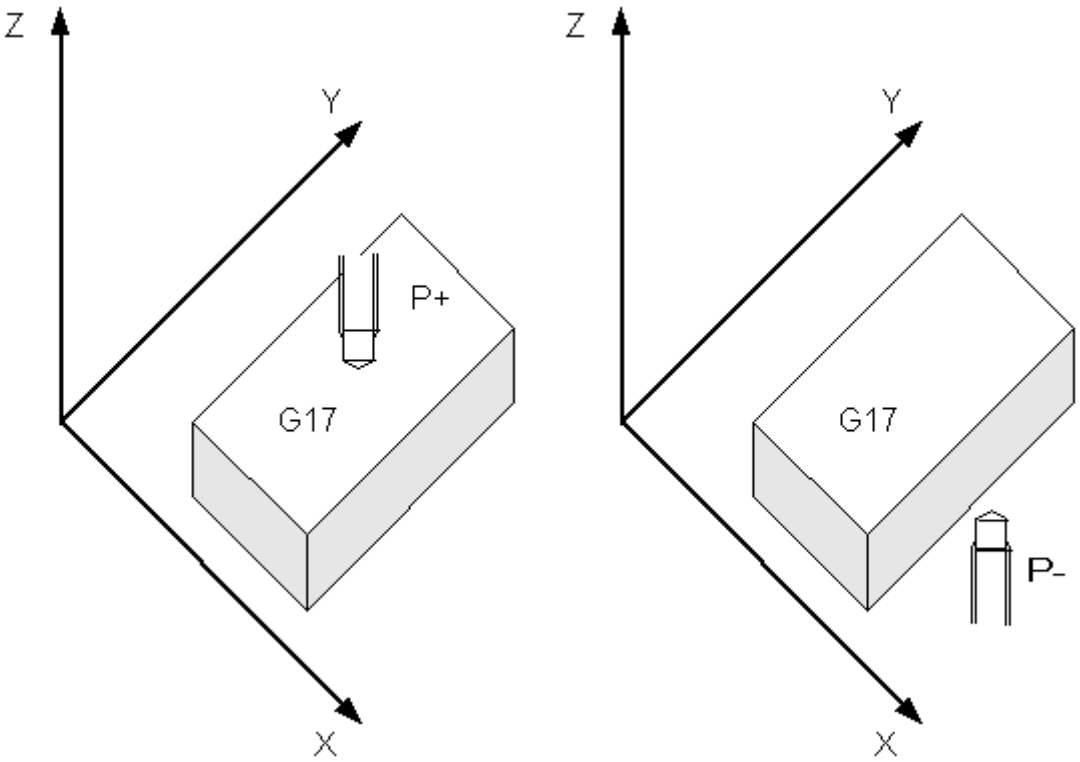
Befehl	G19
Aufhebung	G17 oder G18

Die Funktion G19 legt die Arbeitsebene auf die YZ-Ebene und die Zustellrichtung in X-Richtung fest.

Festlegung der Zustellrichtung

Befehl	P
Parameter	+Zustellrichtung positiv (Standardeinstellung) - Zustellrichtung negativ

Die Parametrierung der Zustellrichtung ist für die Werkzeuglängenkorrektur erforderlich. Damit wird festgelegt, ob das Werkzeug oberhalb oder unterhalb des Werkstücks arbeiten soll.



Beispiel:

```
N10 G0 X0 Y0 Z0 F6000
N20 D2 P- Z
N30 G01 X100
N40 D0 Z
N50 M30
```

In diesem Beispiel arbeitet die Längenkorrektur unterhalb des Werkstücks.

5.1.8 Maßangaben Inch / metrisch

G70	Maßangabe in Inch
G71	Maßangabe in Millimetern (Standardeinstellung)
G700	Maßangabe in Inch mit Verrechnung des Vorschubs
G710	Maßangabe in Millimetern mit Verrechnung des Vorschubs

Standardmäßig ist die Maßangabe in Millimetern (G71) aktiviert. Ob dafür die Koordinaten umgerechnet werden müssen, ist in den Maschinenparametern [▶ 14] (Karteireiter: Interpreter) hinterlegt. Auch hier ist standardmäßig das Basismaßsystem auf Millimeter eingestellt.

Auswirkungen der Umschaltung

Für den Fall, dass das Basismaßsystem ungleich dem aktuellen Maßsystem (mit G70 bzw. G71 eingestellten) ist, so müssen bestimmte Parameter und Koordinaten umgerechnet werden. Der hierfür benötigte Umrechnungsfaktor ist, wie das Basismaßsystem, in Maschinenparametern hinterlegt. Die Umschaltung hat auf folgende Parameter Auswirkungen:

- Weginformationen der Bahnachsen (X, Y & Z)
- Weginformationen der Hilfsachsen (Q1..Q5)
- Zwischenpunktkoordinaten (I, J, K)
- Kreisradius (B bzw. U)
- Programmierbare Nullpunktverschiebung

- Verrundungsradius (Kreis - und Spline-Smooting)

Des Weiteren gibt es Parameter, die grundsätzlich im **Basismaßsystem** verbleiben und nicht umgerechnet werden. Dazu zählen die

- Einstellbare Nullpunktverschiebung
- Werkzeugdaten
- Vorschübe (außer G700 bzw. G710)

Beispiel 1:

Basismaßsystem: Inch

```
...
N10 G71      (metric dimensions)
N20 G01 X100 (conversion is carried out)
N30 G70      (dimensions in inches)
N40 G01 Y100 (conversion is not necessary, because)
....         (the basic dimensions are also inches)
```

Beispiel 2:

Basismaßsystem: Millimeter

```
...
N10 G71 (metric dimensions)
N20 G01 X100 (conversion is not necessary, because)
    (the basic dimensions are also metric)
N30 G70 (dimensions in inches)
N40 G01 Y100 (conversion is carried out)
```

Nullpunktverschiebungen (NPV)

Einstellbare Nullpunktverschiebungen (G54-G57) verbleiben grundsätzlich im Basismaßsystem und werden nicht umgerechnet. Bei den programmierbaren Nullpunktverschiebungen (G58 & G59) ist die Wirkungsweise von dem aktuellen Maßsystem bei der Anwahl der Verschiebung abhängig.

Beispiel 3:

Basismaßsystem: Millimeter

```
...
N10 G71    (mm - default)
N20 G54    (activates adjustable zero offset shift)
N30 G58 X100 (programmable zero offset shift)
N40 G01 X0 F6000 (the axis travels to 100 in the machine co-ordinate system)
N50 G70    (inch)
N60 G01 X0   (zero offset shift is programmed under G71 => zero offset shift remains unchanged)
       (i.e. the axis does not move)
N70 G58 X100 (new programmable zero offset shift - now in inches)
N80 G01 X0   (axis moves out by zero offset shift - to 2540 in the machine co-ordinate system)
```

5.1.9 Einzelsatzbetrieb

Zum Testen eines neuen NC-Programms gibt es die Möglichkeit, die NC I mit dem Funktionsbaustein ItpSingleBlock [▶ 250] auf Einzelsatzbetrieb umzuschalten. Bei aktivem Einzelsatzbetrieb wird das NC-Programm nach jeder Zeile gestoppt. Die Ausführung der nächsten Zeile muss vom Anwender bestätigt werden. Dies können Sie tun, indem Sie im XAE unter dem Reiter Editor auf '**NC Start (F5)**' drücken oder im SPS-Funktionsbaustein ItpSingleBlock [▶ 250] den Eingang 'bTriggerNext' setzen.

Es werden zwei Modi unterschieden:

- Interpreter-Einzelsatzbetrieb
- NC-Kern-Einzelsatzbetrieb

Die Anwahl des SingleBlockModes wird nicht durch einen ItpResetEx2 [▶ 239] zurückgesetzt. Das heißt, wenn z. B. der NC-Kern-Einzelsatzbetrieb aktiv ist, bleibt dieser auch nach einem Reset aktiv.

Interpreter-Einzelsatzbetrieb

Bei aktivem Interpreter-Einzelsatzbetrieb wird das NC-Programm im **Interpreter** nach jeder Zeile gestoppt. Dies ist auch dann der Fall, wenn in der Zeile lediglich Berechnungen und kein Geometriesatz programmiert wurden.
Damit ist es zum Beispiel grundsätzlich möglich, R-Parameter neu zu beschreiben.

Die Aktivierung des Interpreter-Einzelsatzbetriebes sollten Sie vor dem Start des NC-Programms durchführen. Ist dies nicht möglich, so können Sie auch eine M-Funktion für die Aktivierung reservieren und diese mit einem Dekodierstopp kombinieren.
Wird der Interpreter-Einzelsatzbetrieb während der Abarbeitung des NC-Programms ohne M-Funktion und Dekodierstopp eingeschaltet, so kann nicht vorausgesagt werden, wann er aktiv sein wird. Theoretisch ist es möglich, dass die Speicher im NC-Kern (SVB & SAF) gefüllt sind und mehr als 100 Geometrie-Einträge beinhalten. Erst wenn diese Speicher komplett abgearbeitet sind, kann der Einzelsatz wirken.

NC-Kern-Einzelsatzbetrieb

Wie im Interpreter-Einzelsatzbetrieb werden im NC-Kern-Einzelsatzbetrieb die NC-Sätze einzeln ausgeführt. Allerdings mit dem Unterschied, dass im NC-Kern-Einzelsatzbetrieb bereits alle Einträge (z. B. Geometrie-Einträge) den Interpreter durchlaufen haben. Hier ist es also nicht möglich, z. B. R-Parameter nachträglich zu überschreiben.

Diese Betriebsart hat den Vorteil, dass der Einzelsatzbetrieb während der Bearbeitung des NC-Programms aktiviert werden kann. Wird während der Aktivierung ein Geometrie-Eintrag ausgeführt (d.h. die Achsen verfahren), so wird am nächstmöglichen Segmentende angehalten. Dies ist in der Regel das aktuelle Segment. Für die Aktivierung nach Programmstart ist hier keine M-Funktion mit Dekodierstopp erforderlich. Zum Programmende ist jedoch zwingend eine M-Funktion mit Programmende-Kennung (M02, M30) erforderlich.

Wird der NC-Kern-Einzelsatzbetrieb in Verbindung mit den Verschleifungen eingesetzt, so erfolgt die Satzweiterschaltung in der Verschleifungskugel. Die programmierte Verschleifung wird weiterhin ausgeführt (ab TwinCAT V2.10 Build 1301).

Alternativen zur Aktivierung

Es wird empfohlen, den Einzelsatzbetrieb mit ItpSingleBlock [▶ 250] zu aktivieren.

Aus Kompatibilitätsgründen zu früheren TwinCAT-Versionen kann der Einzelsatzbetrieb über das zyklische Kanalinterface aktiviert werden.

Den Einzelsatzbetrieb können Sie im zyklischen Kanalinterface der SPS an- bzw. abwählen. Dazu müssen Sie im Kanalinterface SPS/NC die Variable 'nItpMode' richtig maskieren.

Um den Interpreter-Einzelsatzbetrieb einzuschalten, muss Bit 14 (0x4000) gesetzt werden. Das Zurücksetzen schaltet den Einzelsatzbetrieb wieder aus.

Über dieses Interface ist es auch möglich, den Einzelsatz aus der SPS zu triggern. Dafür muss das Bit 15 gesetzt werden. Die Wirkungsweise ist dabei die gleiche, wie wenn NC-Start im XAE betätigt wird.

5.1.10 Rechenparameter

Bei den Rechenparametern (kurz R-Parameter) handelt es sich um Interpreter-Variablen, die mit einem Ausdruck der Form "R<n>" genannt werden. Da es sich bei 'n' um eine Ganzzahl im Wertebereich 0..999 handelt, stehen insgesamt 1000 R-Parameter zur Verfügung. Davon sind die ersten 900 (R0..R899) Werte lokale Variablen des NC-Kanals. Sie sind nur durch den Interpreter des Kanals zugreifbar. Die R-Parameter R900..R999 sind global angelegt. Sie existieren nur einmal pro NC und die Zugriffe aller Kanäle erfolgen auf denselben Speicher. Dadurch ist ein Datenaustausch (z.B. für eine Teileverfolgung, Kollisionsvermeidung etc.) über die Kanalgrenze hinweg möglich.

Mathematische Berechnungen

Die R-Parameter (wie auch die Achs-Koordinaten, Vorschübe etc.) sind als Variablen des Typs „double“ angelegt. Dadurch ist die Rechenfähigkeit des Rechners voll nutzbar. Die Zahl der Vor- und Nachkommastellen ist nicht durch eine Format-Vorschrift festgelegt. Allerdings ist die Auflösung und

Rechengenauigkeit begrenzt. Dies wird in der Praxis aber nur in besonders kritischen Fällen sichtbar. Beispiele dafür können Differenzen von fast gleichen sehr großen Zahlen oder trigonometrische Funktionen in bestimmten Winkelbereichen sein.

Zuweisung von R-Parametern

```
N100 R5=17.5
N110 R6=-4
N120 R7=2.5 R8=1
```

Wie die dritte Zeile zeigt, ist es ohne weiteres möglich, mehr als eine Zuweisung in einem Satz anzugeben. Dadurch wird die Interpretation ein wenig beschleunigt, aber ein Fehler in der Zeile kann schwieriger zu lokalisieren sein.

Rechenformeln

Eine Rechenformel ist eine Erweiterung der Zuweisung. Sie besteht aus einem Ziel-Parameter, einem Zuweisungszeichen und einer Kette von Werten (R-Parameter und Konstanten), die durch Rechenanweisungen getrennt sind.

```
N100 R1=R2+R3-17.5*R9/2.5
```

Diese Formel wird (entgegen der mathematischen Vorgehensweise) strikt von links nach rechts abgearbeitet.

Die gezeigte Formel wird wie folgt gerechnet:

1. Der Inhalt von R2 wird in das Rechenwerk geladen
2. Der Inhalt von R3 wird in das Rechenwerk geladen
3. Das Rechenwerk führt die Anweisung + aus
4. Der Wert 17.5 wird in das Rechenwerk geladen
5. Das Rechenwerk führt die Anweisung - aus
6. Der Inhalt von R9 wird in das Rechenwerk geladen
7. Das Rechenwerk führt die Anweisung * aus
8. Der Wert 2.5 wird in das Rechenwerk geladen
9. Das Rechenwerk führt die Anweisung / aus
10. Der Inhalt des Rechenwerks wird im R-Parameter R1 gespeichert

Mathematische Funktionen

Der Interpreter stellt Standard-Rechenfunktionen zur Verfügung. Die DIN 66025 legt in dieser Hinsicht keine Syntax fest. Der Aufruf der Rechenfunktionen erfolgt über @6xx (vergl. Anhang - @-Kommando Übersicht [▶ 200]).

Die trigonometrischen Funktionen werden dabei grundsätzlich in Grad berechnet.

Beispiel:

```
N10 R2=0 R3=45
N20 @630 R2 R3
```

In diesem Beispiel wird der Sinus von R3 in Grad berechnet. Das Ergebnis wird anschließend in R2 geschrieben.

R-Parameter Zugriff aus der SPS

Sie können die R-Parameter in die SPS einlesen, bzw. aus der SPS die R-Parameter beschreiben. Hierfür gibt es spezielle SPS-Bausteine, die dies ermöglichen

- ItpReadRParams [▶ 285]
- ItpWriteRParams [▶ 299]

Achten Sie beim Beschreiben der R-Parameter darauf, dass der Interpreter der Satzausführung vorauseilt. D. h. das Schreiben der R-Parameter aus der SPS sollte vor dem NC-Programmstart erfolgen oder mit einem Dekodierstopp [▶ 172] verbunden sein.

Zu Debug-Zwecken können Sie sämtliche R-Parameter zu beliebiger Zeit in eine Datei schreiben. Anstoßen können Sie diesen Prozess via ADS (vergl. ADS-Interface - Kanalfunktionen IndexOffset 0x24 & 0x25).

Sonstige Funktionen

RToDwordGetBit

Diese Funktion wandelt einen R-Parameter in ein DWord und überprüft dann, ob ein bestimmtes Bit gesetzt ist. Das Ergebnis wird wieder in einem R-Parameter hinterlegt.

Befehl	RToDwordGetBit[<dest>; <src>; <bit>]
Parameter <dest>	R-Parameter in dem das Ergebnis eingetragen wird
Parameter <src>	R-Parameter, der die Zahl enthält die gewandelt und überprüft werden soll
Parameter <bit>	Bit, das überprüft werden soll (0..31)

Beispiel:

```
N10 R1=7
N20 RToDwordGetBit[R2;R1;0]
R10=31
N30 RToDwordGetBit[R3;R1;R10]
```

In R2 wird hier eine 1 und in R3 eine 0 eingetragen.

Initialisierung von R-Parametern

Mit 'set RParam' wird einem zusammenhängenden Block von R-Parametern ein Wert zugewiesen.

Befehl	#set RParam(<start index>; <count>; <value>)#
Parameter <start index>	Beschreibt den ersten zu beschreibenden R-Parameter
Parameter <count>	Anzahl der R-Parameter, die beschrieben werden sollen
Parameter <value>	Wert der zugewiesen wird

Beispiel:

```
N10 G01 X100 Y200 F6000
N15 R2=3000
N20 #set RParam( 1; 2; 0.0 )# (R2 is overwritten again here)
N30 G01 X500
```

Retten von R-Parametern

Wenn Sie den Inhalt von R-Parametern [▶ 136] für eine spätere Verwendung noch brauchen, die R-Parameter aber zwischenzeitlich für einen anderen Zweck benutzt werden sollen, kann er zeitweise im Werte-Stapel des Rechenwerks hinterlegt werden.

Hierfür existieren zwei Möglichkeiten:

- Aufzählung der R-Parameter
- Bereichsangabe der R-Parameter

Retten der Werte:

Befehl	@40 <Anzahl> R<n> R<m>... @41 <1. R-Parameter> <letzter R-Parameter>

Restauration der Werte:

Befehl	@42 <Anzahl> R<n> R<m> @43 <letzter R-Parameter> <1. R-Parameter>

Beim Restaurieren der Werte nennen Sie die Parameter in umgekehrter Reihenfolge.

Beispiel 1:

```
(saving the data)
N100 @40 K4 R800 R810 R823 R4

N110 R800=4711
N120 ...

(restoring the data)
N200 @42 K4 R4 R823 R810 R800
```

Beispiel 2:

```
(saving the data)
N100 @41 R800 R805

N110 R800=4711
N120 ...

(restoring the data)
N200 @43 R805 R800
```

Größe des Stacks

Der Wertestapel des Rechenwerks hat eine begrenzte Kapazität. Läuft er über, so wird das NC-Programm mit einer Fehlermeldung abgebrochen. Das kann beim Retten von Werten auftreten, kann aber auch bei einer daran anschließenden Rechenformel vorkommen.

5.2 Programmierung von Bewegungssätzen

5.2.1 Referenzierung

Referenzieren (Homing) Sie die Achsen standardmäßig vor dem Bilden der 3D-Gruppe aus dem PTP-Kanal. Sie können dies aber auch noch aus dem NC-Programm heraus machen.

Wenn Sie die Achsen im PTP-Mode referenzieren, können Sie dies für mehrere Achsen gleichzeitig durchführen. Aus dem NC-Programm können Sie nur eine Achse zeitgleich referenzieren.

Befehl	G74
Aufhebung	Satzende

Beispiel:

```
N10 G74 X
N20 G74 Y
```

Referenzierung im eigenen Satz

Die Referenzierung muss in einem eigenen Satz erfolgen. Dabei dürfen Sie mit G74 nur eine Achse nennen. Dieses Kommando ist nur auf die Hauptachsen (X,Y,Z) anwendbar.

5.2.2 Eilgang

Befehl	G0
Aufhebung	G1 [▶ 140], G2 [▶ 141] oder G3 [▶ 141]

Der Eilgang wird zum schnellen Positionieren des Werkzeugs eingesetzt und ist nicht für die Bearbeitung des Werkstücks vorgesehen. G0 verfährt die Achsen mit einer linearen Interpolation so schnell wie möglich. Dabei wird die Geschwindigkeit aus MIN (Rapid Traverse Velocity (G0), Reference Velocity, Maximum Velocity) berechnet.

Wenn mehrere Achsen im Eilgang verfahren werden sollen, dann wird die Geschwindigkeit von der Achse bestimmt, die für ihren Bahnweg die längste Zeit benötigt.

Mit G0 wird ein Genauhalt (G60 [▶ 144]) aufgehoben.

Die Geschwindigkeit für den Eilgang wird für jede Achse individuell eingestellt. Editieren können Sie dies in den Parametern der Achse im XAE unter dem Punkt NCI-Parameter.

General | Settings | Parameter | Dynamics | Online | Functions | Coupling | Compensation

	Parameter	Offline Value	Online Value	Type	Unit
+	Maximum Dynamics:				
+	Default Dynamics:				
+	Manual Motion and Homing:				
+	Fast Axis Stop:				
+	Limit Switches:				
+	Monitoring:				
+	Setpoint Generator:				
-	NCI Parameter:				
	Rapid Traverse Velocity (G0)	2000.0		F	mm/s
	Velo Jump Factor	0.0		F	
	Tolerance ball auxiliary axis	0.0		F	
	Max. position deviation, aux. axis	0.0		F	
+	Other Settings:				

5.2.3 Linearinterpolation

Befehl	G1 bzw. G01 (Standardeinstellung)
Aufhebung	G0 [▶ 139], G2 [▶ 141] oder G3 [▶ 141]

Bei der Linearinterpolation verfährt das Werkzeug mit dem Vorschub F eine Gerade, die frei im Raum liegen kann. Dabei wird die Bewegung der betroffenen Achsen zeitgleich abgeschlossen.

Mit dem Vorschub F wird die Bahngeschwindigkeit in Millimeter pro Minute beschrieben. Dieser Wert ist modal wirksam, d.h. er muss nicht neu programmiert werden, wenn später für andere Geometrieeinträge der gleiche Vorschub verwendet werden soll.

Beispiel:

```
N10 G90
N20 G01 X100.1 Y200 F6000
```

In diesem Beispiel werden die Achsen linear auf die beschriebene Position gefahren. Die Z-Achse wird in diesem Programm nicht erwähnt und bleibt deshalb auf ihrer alten Position.

5.2.4 Kreisinterpolation

Die Programmierung von Kreisen kann auf unterschiedliche Weise erfolgen. Dabei muss zwischen zwei Arten unterschieden werden. Dies ist zum einen ein Kreis in der Arbeitsebene [▶ 132] (z.B. XY-Ebene) und zum anderen ein Kreis, der frei im Raum liegen kann (CIP-Kreis).

Kreisinterpolation im Uhrzeigersinn

Befehl	G2 bzw. G02
Aufhebung	G0 [▶ 139], G1 [▶ 140] oder G3 [▶ 142]

Mit der Funktion G2 wird eine Kreisbahn im Uhrzeigersinn beschrieben. Hierfür ist es allerdings erforderlich, dass die Arbeitsebene [▶ 132] zuvor bestimmt wird (standardmäßig G17 [▶ 132]).

Um den Kreis eindeutig zu beschreiben, sind neben dem Endpunkt noch weitere Parameter notwendig. Dabei kann zwischen einer Mittelpunktprogrammierung und einer Radiusprogrammierung gewählt werden.

Radiusprogrammierung

Bei der Radiusprogrammierung wird neben dem Endpunkt noch der Radius des Kreises programmiert. Für den Radius können wahlweise die Buchstaben *'B'* bzw. *'U'* verwendet werden.

Da mit G2 die Richtung vorgegeben ist, ist auch der Kreis eindeutig bestimmt. Die Anfangskoordinaten ergeben sich aus der vorangegangenen Geometrie.

Beispiel 1:

```
N10 G01 G17 X100 Y100 F6000
N20 G02 X200 B200
```

Winkelprogrammierung für Winkel >180°

Wenn ein Winkel größer als 180° gefahren werden soll, so muss der Radius negativ angegeben werden.

Vollkreisprogrammierung

Start- und Endpunkt müssen sich voneinander unterscheiden, damit der Mittelpunkt berechenbar ist. Somit ist mit der Radiusprogrammierung kein Vollkreis programmierbar. Hierfür kann die Mittelpunktprogrammierung verwendet werden.

Mittelpunktprogrammierung

Die Mittelpunktprogrammierung stellt eine Alternative zu der gerade beschriebenen Methode dar. Der Vorteil der Mittelpunktprogrammierung ist, dass hier auch Vollkreise beschrieben werden können.

In der Standardeinstellung wird der Mittelpunkt immer relativ zum Anfangspunkt des Kreises angegeben. Hierfür werden die Parameter I, J und K verwendet. Dabei steht

- I für den X-Anteil
- J für den Y-Anteil und
- K für den Z-Anteil.

Mindestens einer dieser Parameter ist 0 und es ist deshalb nicht erforderlich diese mit zu programmieren.

Beispiel 2:

```
N10 G01 G17 X100 Y100 F6000
N20 G02 I50 J0 (J is optional) X200
N30 M30 (program end)
```

Beispiel 3:

```
N10 G01 G18 X100 Y100 Z100 F6000
N20 G02 I0 K50 X150 Z150 (quarter circle in ZX plane)
N30 M30
```

Über die Programmierung eines Maschinendatums ist es aber auch möglich, den Mittelpunkt absolut einzugeben. Um auf ein Maschinendatenbit schreibend zuzugreifen, wird der Befehl @402 benötigt.

Im folgenden Beispiel wird der Kreis aus dem 1. Beispiel mit dem absoluten Kreismittelpunkt programmiert.

Beispiel 4:

```
N10 G01 G17 X100 Y100 F6000
N20 @402 K5003 K5 K1 (center point programming absolute)
N30 G02 I150 J100 X200
N40 @402 K5003 K5 K0 (center point programming relative)
N50 M30
```

Kreisinterpolation gegen Uhrzeigersinn

Befehl	G3 bzw. G03
Aufhebung	G0 [▶ 139], G1 [▶ 140] oder G2 [▶ 141]

Mit der Funktion G3 wird eine Kreisbahn gegen den Uhrzeigersinn gefahren. Die Parameter und auch die Eingabemöglichkeiten sind die gleichen, wie unter G2.

Kreisgenauigkeit

Befehl	#set paramRadiusPrec(<param>)#
Parameter	param: maximal erlaubte Radiustoleranz 0.001 < param < 1.0 (default 0.1)

Mit der Funktion 'set paramRadiusPrec' wird die erforderliche Kreisgenauigkeit parametriert. Dieser Parameter wirkt auf Kreise, die mit G02 bzw. G03 programmiert werden.

Ist bei der Mittelpunkprogrammierung die Differenz der Radienlänge grösser als <param>, so wird ein Fehler generiert.

Mittelpunktskorrektur

Befehl	CPCON (Standardeinstellung)
Aufhebung	CPCOF

Bei der Mittelpunktsprogrammierung ist der Kreis überbestimmt. Damit die Daten konsistent sind, wird im Standardfall der Mittelpunkt korrigiert. In der Regel ist dafür nur eine marginale Änderung des Mittelpunkts notwendig. Nach der Mittelpunktskorrektur ist der Betrag des Eingangsradius gleich dem Ausgangsradius.

Liegen Start- und Endpunkt sehr dicht beieinander, so kann der Mittelpunkt massiv verschoben werden. Dies kann bei automatisch generierten G-Code (Postprozessor) zu Problemen führen. Für manuell geschriebenen G-Code wird die Einstellung CPCON (center point correction on) empfohlen.

CIP-Kreis

Befehl	CIP
Aufhebung	Satzende

Die bislang besprochenen Kreise können nur in den Hauptebenen verfahren. Mit dem CIP-Kreis ist es auch möglich, einen Kreis frei im Raum zu programmieren. Hierfür muss neben dem Endpunkt auch noch ein Punkt auf der Bahn programmiert werden.

Damit der Kreis eindeutig beschrieben werden kann, dürfen alle 3 Punkte (Anfangspunkt ist implizit vorgegeben) nicht kollinear sein. Es lässt sich also auf diese Weise kein Vollkreis programmieren.

Als Parameter für den Bahnpunkt stehen I, J und K zur Verfügung, die standardmäßig relativ zum Kreisanfangspunkt beschrieben werden.

Beispiel 5:

```
N10 G01 X100 Y100 F6000
N20 CIP X200 Y200 I50 J50 K50
```

Hinweis **Um den CIP-Kreis verfahren zu können, darf die** Fräserradiuskorrektur [▶ 190] **nicht aktiv sein.**

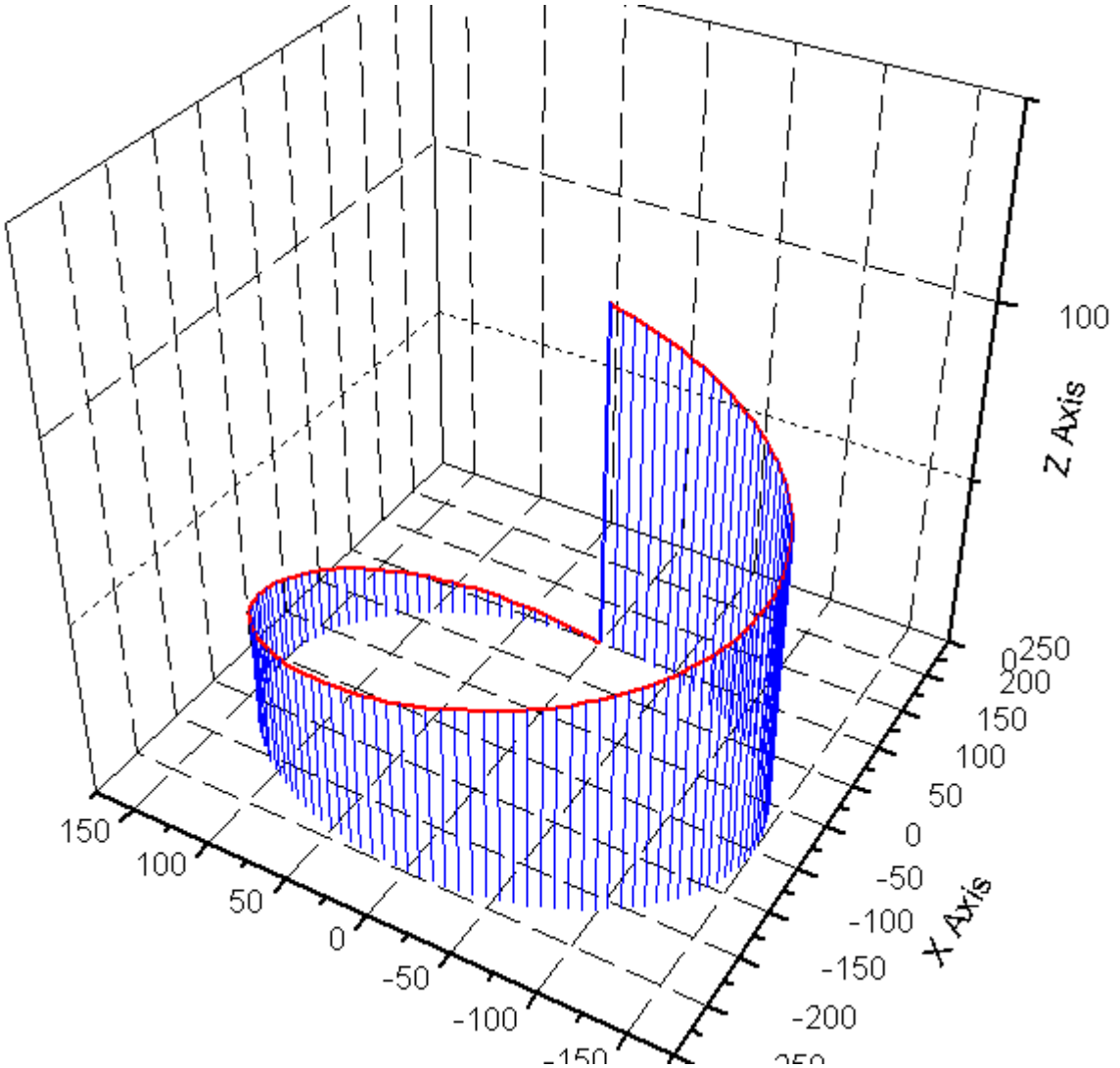
5.2.5 Helix

Wird einer Kreisbewegung eine senkrechte Linearbewegung überlagert, so erhält man eine Helix. Die Programmierung einer Helix ist nur in den Hauptebenen möglich. Es werden dabei die gleichen Parameter wie bei der Kreisbewegung in den Hauptebenen verwendet. Zusätzlich wird noch die Achse, die senkrecht zur Ebene steht, verfahren.

Die Helix kann zusammen mit der Fräserradiuskorrektur [▶ 190] verwendet werden.

Beispiel:

```
N10 G01 G17 X100 Y0 Z0 F6000
N20 G03 I-50 Z100
M30
```



5.2.6 Verweilzeit

Befehl	G4 bzw. G04
Aufhebung	Satzende
Parameter	F oder X

Mit G4 wird die Verweilzeit eingeschaltet. Sie dient dazu, zwischen zwei NC-Sätzen für eine programmierte Zeit in Sekunden, die Werkstückbearbeitung zu unterbrechen.

Beispiel:

```
N10 G01 X100 F6000
N20 G04 X0.5 (pause in sec)
N30 G02 X300
...
```

Hinweis Die Programmierung der Verweilzeit muss in einem eigenen Satz erfolgen und die Parameter (X bzw. F) müssen nach dem G04 programmiert werden.

5.2.7 Genauhalt

satzweise

Befehl	G9 bzw G09 (Standardeinstellung)
Aufhebung	Satzende

Die Genauhaltanweisung wird z.B. dann benutzt, wenn scharfe Konturecken hergestellt werden müssen. Dabei wird die Sollgeschwindigkeit der Bahn im Konturübergang bis auf null reduziert und anschließend wieder erhöht. Auf diese Weise wird sichergestellt, dass die programmierte Position genau angefahren wird.

Hinweis G09 wirkt nur sollwertseitig. Eine Überprüfung der Istwerte kann z.B. mit TPM (Zielpositionsüberwachung) vorgenommen werden.

modal

Befehl	G60
Aufhebung	G0 [▶ 139]

Beschreibung:

s.o.

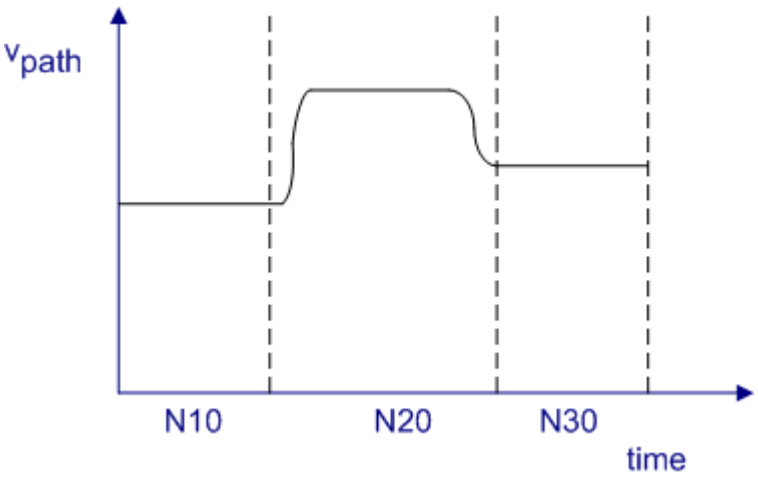
vergl. auch Zielpositionsüberwachung [▶ 148] (TPM)

5.2.8 Vorschubinterpolation

Konstante Vorschubinterpolation

Befehl	FCONST (Standardeinstellung)
Aufhebung	FLIN

Mit der konstanten Vorschubinterpolation (default) wird die programmierte Geschwindigkeit so schnell wie möglich angefahren.



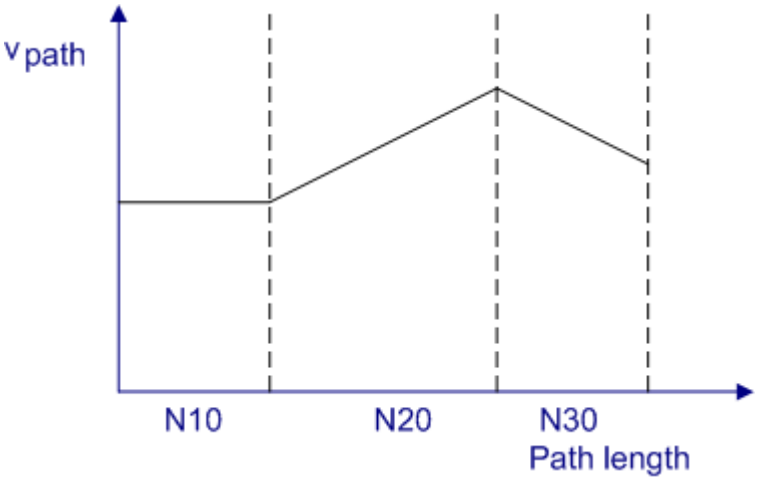
Beispiel 1:

```
N05 FCONST
N10 G01 X1000 F50000
N20 G01 X2500 F80000
N30 G01 X3500 F60000
...
```

Lineare Vorschubinterpolation

Befehl	FLIN
Aufhebung	FCONST

Die lineare Vorschubinterpolation überführt die Geschwindigkeit von v_start nach v_end linear über den Bahnweg.



Beispiel 1:

```
N05 FCONST
N10 G01 X1000 F50000
N15 FLIN
N20 G01 X2500 F80000
N30 G01 X3500 F60000
...
```

Hinweis Wenn aufgrund der Geometrie oder z.B. einer M-Funktion die Geschwindigkeit am Segmentübergang stärker reduziert werden muss, als die programmierte Segmentendgeschwindigkeit, so wird der lineare Geschwindigkeitsverlauf so lange wie möglich eingehalten. Erst wenn es dynamisch erforderlich ist, wird auf die reduzierte Segmentendgeschwindigkeit verzögert.

5.2.9 Nullpunktverschiebungen

In TwinCAT NC I stehen eine Reihe von Nullpunktverschiebungen zur Verfügung. Hiermit wird der Abstand zwischen dem Werkstück- und dem Maschinennullpunkt beschrieben.

Unterdrückung der Nullpunktverschiebung

Befehl	G53 (Standardeinstellung)
Aufhebung	G54 [▶ 146] bis G59 [▶ 147]

Mit G53 wird die Nullpunktverschiebung modal unterdrückt. Dabei wirkt sich die Unterdrückung sowohl auf die einstellbare, wie auch auf die programmierbare Nullpunktverschiebung aus.

Einstellbare Nullpunktverschiebung

Befehl	G54 G55 G56 G57
Aufhebung	G53 [▶ 146] oder Anwahl einer anderen einstellbaren Nullpunktverschiebung

Im NC-Programm kann über die Befehle G54 bis G57 zwischen den Nullpunktverschiebungen hin- und hergeschaltet werden.

Parametrierung

Die einstellbare Nullpunktverschiebung kann auf unterschiedliche Weise parametriert werden

1. SPS-Funktionsbaustein ItpWriteZeroShiftEx [▶ 256] (empfohlener Standard)
2. XAE Interpreter-Element [▶ 14]
3. aus dem DIN-Programm

Die Parameter werden für jeden Interpolationskanal einzeln gesichert. D.h. die einstellbaren Nullpunktverschiebungen sind kanalabhängig.

Hinweis Die Anwahl der einstellbaren Nullpunktverschiebung muss in einem eigenen Satz erfolgen. Damit die Verschiebung auch wirklich herausgefahren wird, müssen in einem folgenden Geometriesatz wenigstens die betroffenen Achsen genannt werden.

Beispiel 1:

```
N10 G01 X100 Y0 Z0 F6000
N20 G54 (activates adjustable zero offset shift (NPV))
N30 G01 X Y Z
N40 M30
```

In Beispiel 1 werden in Zeile 30 alle beteiligten Achsen genannt. Dies bewirkt, dass die Nullpunktverschiebungen für alle Achsen herausgefahren werden.

Beispiel 2:

```
N10 G01 X100 Y0 Z0 F6000
N20 G54 (activates adjustable zero offset shift (NPV))
N30 G01 X200 Y
```

In Beispiel 2 wird in Zeile 30 die X-Achse an die Position 200 + Verschiebung in X-Richtung verfahren. Bei der Y-Achse wird lediglich die Verschiebung herausgefahren und die Z-Achse wird nicht bewegt.

Parametrierung aus dem DIN-Programm

Befehl	#set paramZeroShift(G<n>; <value x>; <value y>; <value z>)#

Parameter G<n>	Nullpunktverschiebung die parametriert werden soll (G54..G59)
Parameter <value>	Koordinaten der Nullpunktverschiebung

Mit '*#set paramZeroShift(..)#*' wird die NPV parametriert aber noch nicht aktiviert. Dafür muss der G-Code noch explizit programmiert werden.

Beispiel 3:

```
N10 G01 X100 Y0 Z0 F6000
N20 R12=200
N30 #set paramZeroShift( G54; 100.0; R12; -20)#
N40 G54 (activates adjustable zero offset shift (NPV))
N50 G01 X200 Y Z
```

Programmierbare Nullpunktverschiebung

Befehl	G58 oder G59
Aufhebung	G53 [▶ 146]

Neben der einstellbaren existieren auch noch programmierbare Nullpunktverschiebungen. Diese Art der Nullpunktverschiebung wird direkt aus dem NC-Programm beschrieben.

Addition der Nullpunktverschiebungen

Die programmierbare Nullpunktverschiebung wirkt nur bei aktiver einstellbarer Nullpunktverschiebung. D.h. die gesamte Verschiebung ist die Summe aus

- eingestellter Nullpunktverschiebung(G54, G55, G56 oder G57)
- erster programmierbarer Nullpunktverschiebung (G58)
- zweiter programmierbarer Nullpunktverschiebung (G59)

Beispiel 4:

```
N10 G01 X100 Y0 Z0 F6000
N20 G54 (activates adjustable zero offset shift (NPV))
N30 G58 X0.5 Y0.5 Z0.5 (1st prg. zero offset shift)
N50 X Y Z (movements for the zero offset shift)
...
M30
```

Verhalten bei Kettenmaßangabe

Standardverhalten

Eine Veränderung des Nullpunktes wirkt sich auch im Kettenmaß aus.

Beispiel 5:

```
N10 G01 X100 Y0 Z0 F6000
N20 G54 (activates adjustable zero offset shift (NPV))
N25 G58 X10 Y10 Z0
N30 G91 (Incr. dimensions)
N40 G01 X200 Y0
N50 ...
```

In N40 verfährt Y im Basis-Koordinatensystem auf 10. Mit der Verschiebung des Nullpunktes, verschiebt sich auch der Bezugspunkt für die Kettenmaßprogrammierung, womit sich für Y dann ein Verfahrweg ergibt.

Auf diese Weise kann eine Kontur, die komplett im Kettenmaß programmiert ist, durch eine Nullpunktverschiebung an beliebiger Stelle abgefahren werden.

Das Verhalten von G91 ist parametrierbar.

Befehl	Beschreibung
ZeroShiftIncOn	Auch unter G91 werden die Nullpunktverschiebungen herausgefahren, wenn die Achse genannt wird. (Standardeinstellung)
ZeroShiftIncOff	Unter G91 wird die Nullpunktverschiebung nicht herausgefahren.

Beispiel 6:

```
N10 G01 X100 Y0 Z0 F6000
N15 ZeroShiftIncOff
N20 G54 (activates adjustable zero offset shift (NPV))
N25 G58 X10 Y10 Z0
N30 G91 (Incr. dimensions)
N40 G01 X200 Y
N50 ...
```

Da im Beispiel 6 'ZeroShiftInc**Off**' eingestellt ist, wird die X-Achse in N40 um 200mm unabhängig von der neuen Nullpunktverschiebung verfahren. Für die Y-Achse wurde keine Zielkoordinate programmiert, die Achse bleibt also stehen.

Vergl. auch ToolOffsetIncOn/Off [▶ 187]

5.2.10 Zielpositionsüberwachung

Befehl	TPM
Aufhebung	Satzende

Mit dem Befehl 'TPM' wird die Zielpositionsüberwachung aus dem NC-Programm angestoßen. Dies führt am Geometrieende immer zu einem sollwertseitigen Genauhalt und einer anschließenden Kontrolle des Zielpositionsfensters. Wenn für alle Achsen der Gruppe die Überwachungsbedingungen erfüllt sind, erfolgt die Satzweiterschaltung.

Wie bei der PTP wird diese Funktion für jede Achse einzeln aktiviert und parametriert. Somit können z.B. für Hilfsachsen andere Grenzwerte als für die Bahnachsen gewählt werden.

Beispiel 1:

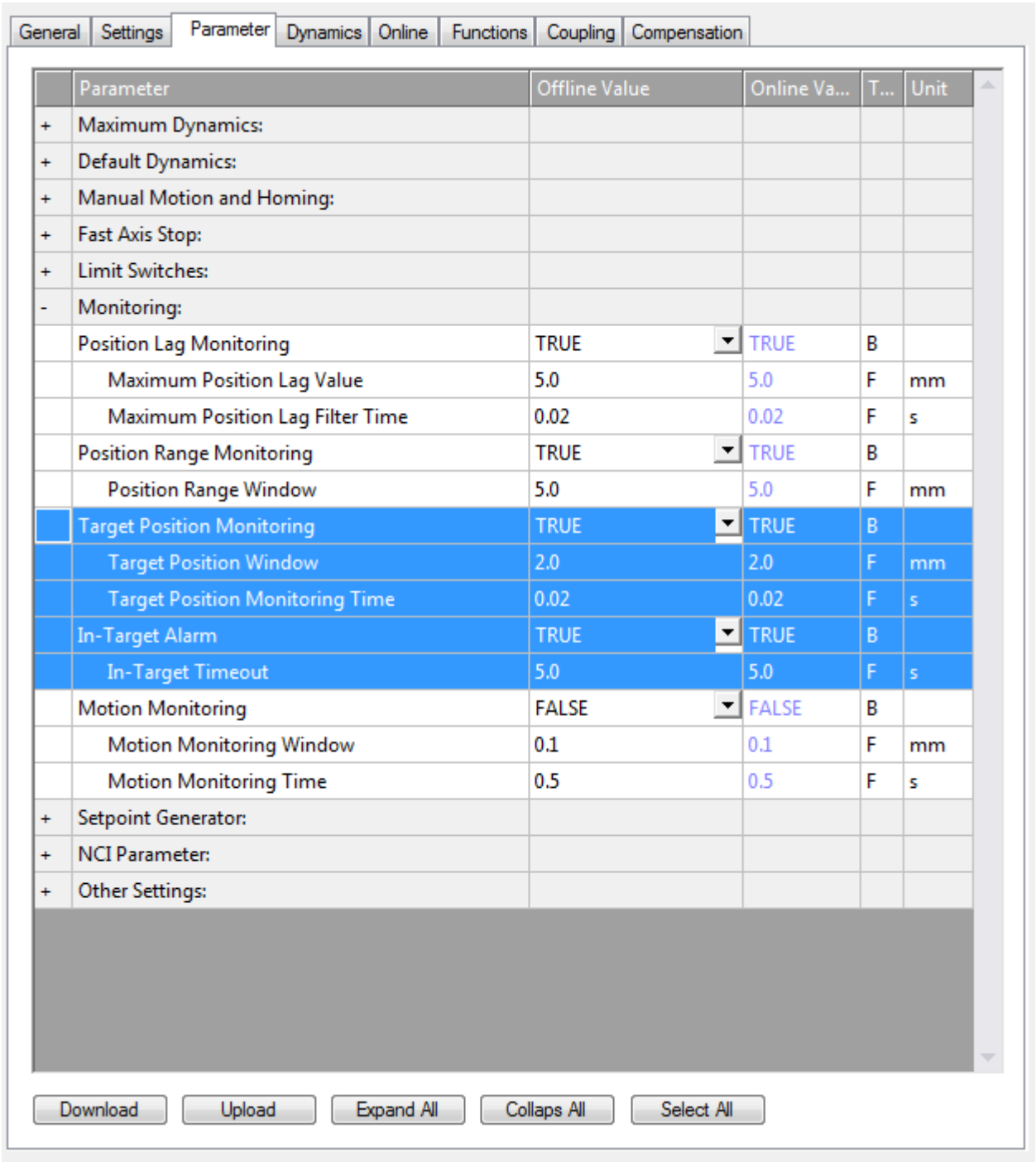
```
N10 G01 X100 Y100 F6000
N20 G01 X300 Y100 TPM
...
```

Am Ende der Bewegung von N20 wird sowohl für die X-Achse, als auch für Y die Zielpositionsüberwachung durchgeführt (vorausgesetzt, beide Achsen haben die Zielpositionsüberwachung aktiviert).

Beispiel 2:

```
N10 G01 X100 Y100 F6000
N20 G01 X300 Y100
N30 M61 (Type Handshake)
N40 TPM
...
```

TPM kann auch in einem eigenen Satz programmiert werden. Dabei wird dann die letzte Positionierung überprüft (hier von N20).



Hinweis **Wenn die Zielpositionsüberwachung für eine Achse aktiviert ist, sollte auch der Zielpositionsalarm (PEH) aktiv sein. Die Zeitüberwachung bewirkt, dass spätestens nach dem Timeout ein Kanalfehler generiert wird, falls sich die Achse noch nicht im Zielpositionsfenster befindet. Damit keine unnötigen Kanalfehler generiert werden, sollte der Timeout-Wert genügend groß gewählt werden (z.B. 5 - 10s). Für den Fall, dass keine PEH-Zeitüberwachung aktiv ist und sich die Achse dauerhaft außerhalb des Positionsfensters befindet, erfolgt keine Satzweiterschaltung und die NC bleibt von außen betrachtet stehen. Dabei befindet sich die SAF im Waiting-Zustand (nicht zu verwechseln mit dem Interpreterstatus).**

Vergl. auch Genauhalt [▶ 144] (G09).

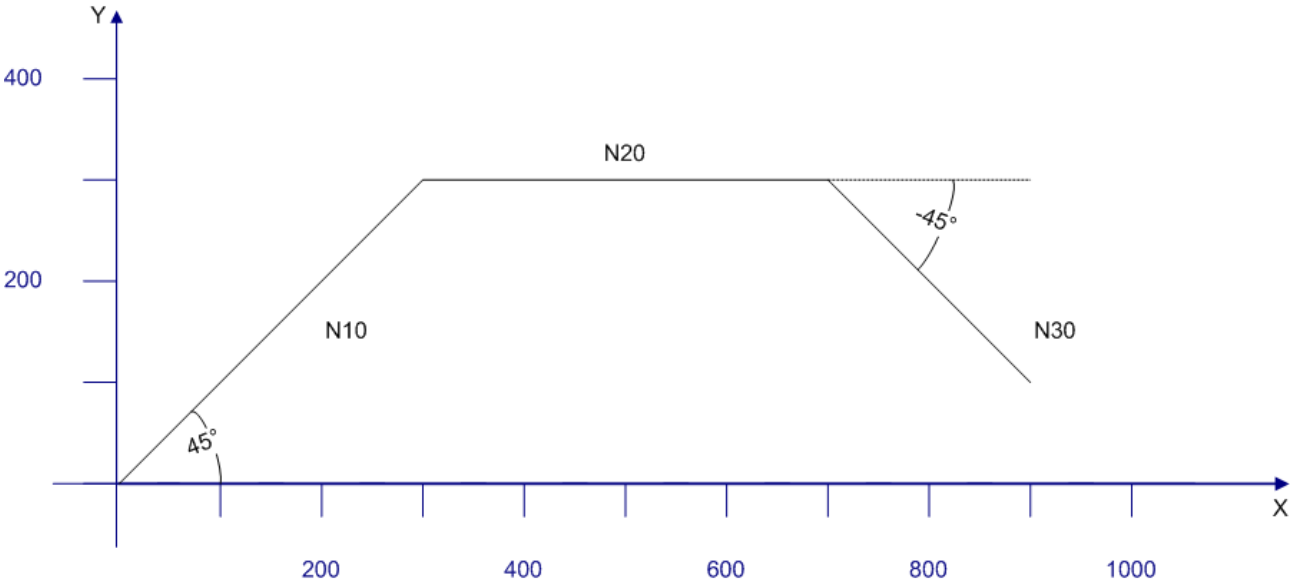
5.2.11 Konturzüge

Winkel und Segmentlänge

Bei dieser Art der Programmierung werden ähnlich wie bei Polarkoordinaten immer der Winkel und der Betrag (Segmentlänge) angegeben.

Parameter	Beschreibung
ANG	Winkel in Grad relativ zur Abszisse (-360 ≤ ang ≤ 360)
SEG	Betrag der Segmentlänge

Beispiel 1:




```
N10 G01 ANG=45  SEG=424.264 F60000
N20 G01 ANG=0   SEG=400
N30 G01 ANG=-45 SEG=282.843
```

oder

```
N10 G01 ANG=45  SEG=424.264 F60000
N20 G01 X700 Y300
N30 G01 ANG=-45 SEG=282.843
```

Einschränkungen:

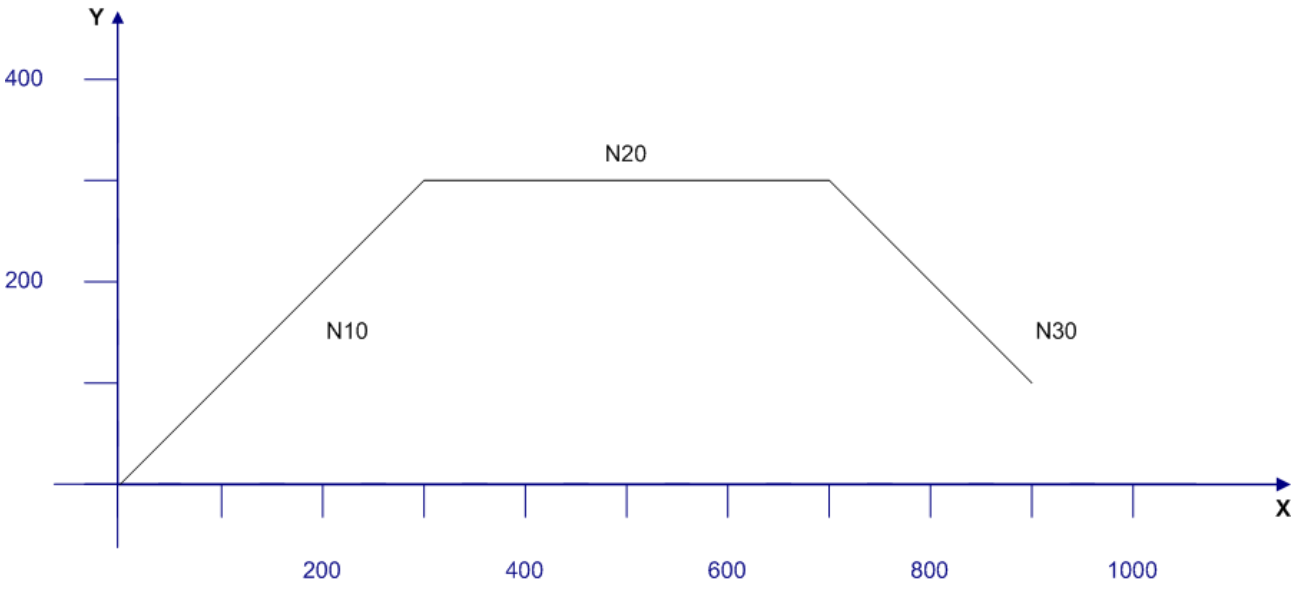
- Die Programmierung darf nur in der angewählten Hauptebene erfolgen.
- Die Segmentlänge muss echt größer Null sein und bezieht sich auf die Projektion der Hauptebene.

Hinweis Eine Verrundung bzw. Fase kann zusätzlich noch programmiert werden. Die Parameter ANG und SEG müssen in jedem Satz programmiert werden. Bei der Zuweisung dürfen R-Parameter, aber keine Formel programmiert werden.

Winkel und eine Komponente in der Ebene

Wie oben wird ein Winkel programmiert, aber die Länge des Segments wird nicht mehr direkt vorgegeben. Sie wird aus einer Komponente der angewählten Hauptebene berechnet.

Beispiel 2:




```
N10 G01 ANG=45 X300

N20 G01 ANG=0 Z700
R10=100
N30 G01 ANG=315
X=R10
```

Laufzeitfehler

Werden zwei Komponenten der Ebene oder keine angegeben, so führt dieses zu einem Laufzeitfehler. Des Weiteren wird ein Laufzeitfehler generiert, wenn die Bewegung parallel zur Abszisse bzw. Ordinate programmiert wird und es keinen Schnittpunkt damit gibt.

5.2.12 Rotation

Neben der Nullpunktverschiebung [▶ 146] ist es auch möglich, eine Rotation (Drehung) zu programmieren. Dabei wird zwischen einer absoluten und einer additiven Rotation unterschieden.

Mit der Rotation können im Werkstückkoordinatensystem die Koordinatenachsen (X, Y und Z) gedreht werden.

Damit ist es möglich, schräg liegende Flächen (in der Ebene oder auch im Raum) zu bearbeiten.

Absolute Rotation

Befehl	ROT X<Wert(x)> Y<Wert(y)> Z<Wert(z)>
Aufhebung	ROT (ohne Parameter)

Die Anweisungen zur Rotation müssen in einem eigenen Satz programmiert werden. Dabei sind die Winkelangaben grundsätzlich in Grad vorzunehmen.

Drehrichtung

Ein positiver Winkel beschreibt die Drehung in Richtung der positiven Koordinatenachse und Drehung gegen den Uhrzeigersinn.

Durchführung der Drehung

Bei der Drehung eines Koordinatensystems ist die Reihenfolge der Drehung von entscheidender Bedeutung. In TwinCAT NC I wird die Rotation immer in folgender Reihenfolge um das globale Koordinatensystem durchgeführt:

1. Drehung um die Z-Achse,
2. Drehung um die Y-Achse,

3. Drehung um die X-Achse.

Diese Reihenfolge wird auch dann eingehalten, wenn die Parameter in einer anderen Reihenfolge programmiert werden.

Als Drehpunkt wird immer der Ursprung des Werkstückkoordinatensystems verwendet. D. h. die gerade aktive gesamte Nullpunktverschiebung beschreibt den Drehpunkt.

Additive Rotation

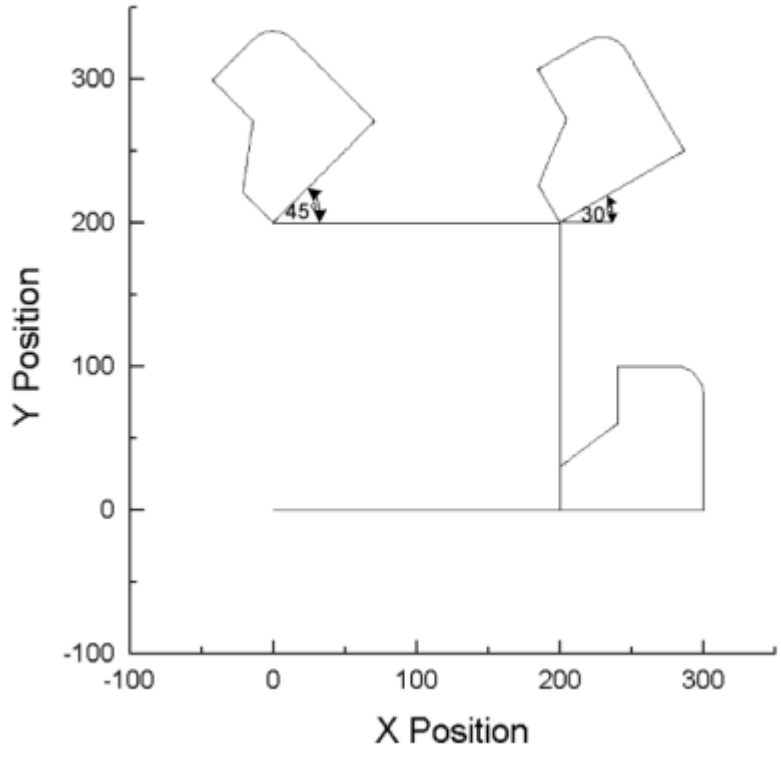
Neben der absoluten Programmierung der Rotation, ist es auch möglich diese additiv durchzuführen. Dabei gelten die gleichen Bedingungen wie bei der absoluten Rotation.

Befehl	AROT X <Wert(x)> Y<Wert(y)> Z<Wert(z)>
Aufhebung	ROT (ohne Parameter)

Beispiel:

```
N10 G01 G17 X0 Y0 Z0 F60000
N20 G55
N30 G58 X200 Y0
N50 L47
N60 G58 X200 Y200
N65 ROT Z30
N70 L47
N80 G58 X0 Y200
N90 AROT Z15
N100 L47
N110 M30
```

```
L47
N47000 G01 X0 Y0 Z0 (movements for zero shift & rotation)
N47010 G91 (incremental dimensions)
N47020 G01 X100
N47030 G01 Y80
N47040 G03 X-20 Y20 I-20 J0
N47050 G01 X-40
N47060 G01 Y-40
N47070 G01 X-40 Y-30
N47080 G01 Y-30
N47090 G90
N47100 M17
```



In diesem Beispiel wird die gleiche Kontur unter verschiedenen Drehwinkeln verfahren. Da die Kontur (L47) im Kettenmaß programmiert ist und der Anfangspunkt über die programmierte Nullpunktverschiebung beschrieben wird, ist die Drehung recht anschaulich.

Hinweis:

Nach der Programmierung des ROT- bzw. AROT-Kommandos muss immer der komplette Bahnvektor (X, Y & Z) zugewiesen werden.

Erweiterungen der Rotation

In der Default-Konfiguration muss nach einem ROT-Kommando immer der komplette Bahnvektor programmiert werden. Da dies in einigen Applikation schwer zu realisieren ist, kann optional diese Berechnung automatisch im Interpreter erfolgen. Möchte man diese Option nutzen, so sollte zu Beginn des NC-Programms 'RotExOn' programmiert werden.

Befehl	RotExOn
Aufhebung	RotExOff

Beispiel:

```
N10 RotExOn
...
N100 G54 (activate zero point & point of rotation)
N110 ROT X90
N120 G0 Z3 (preposition the tool)
N130 G01 Z-10 F6000 (lower to cutting depth)
N140 G01 X100
N150 G01 Z3 (raise to preposition)
...
N1000 RotExOff
N1010 M30
```

Berechne Rotation

Befehl	CalcRot[R<s>; R<t>; R<u>]
	CalcInvRot[R<s>; R<t>; R<u>]
Parameter	Die 3 R-Parameter beschreiben den zu berechnenden Vektor. Mit der Berechnung wird das Ergebnis in diese R-Parameter eingetragen und der ursprüngliche Wert damit überschrieben

Die Funktion **CalcRot** rotiert einen dreidimensionalen Vektor um die aktuellen Rotationswinkel. Dabei wurden die Rotationswinkel zuvor mit ROT bzw. AROT beschrieben. Die Reihenfolge der Berechnung ist die gleiche wie bei der eigentlichen Rotation, also Z, Y und X.

Die Funktion **CalcInvRot** verhält sich genau umgekehrt. Dabei werden die Vorzeichen der aktuell gültigen Rotationswinkel invertiert und die Reihenfolge der Berechnung ist X, Y und Z. Der Vektor wird also quasi zurückgedreht.

CalcRot und CalcInvRot generieren keine Geometrie, sondern führen lediglich die Berechnung des Vektors aus.

Beispiel:

```
N10 G01 X40 Y10 Z0 F6000 (the axes are moved
without rotation)
N20 R1=40 R2=10 R3=0

N30 ROT Z45

(What is the position to which X, Y, must be taken so that no
movement is executed?)
N40 CalcInvRot[R1; R2; R3]
N50 G01 X=R1 Y=R2 Z=R3 (R1=35.35 R2=-21.21 R3=0)
N60 ...
```

Befehl	RotVec[R<x>; R<y>; R<z>; R<α>; R<β>; R<γ>]

Parameter	Die 3 R-Parameter (x..z) beschreiben den zu drehenden Vektor. Mit der Berechnung wird das Ergebnis in diese R-Parameter eingetragen und der ursprüngliche Wert damit überschrieben Die letzten 3 R-Parameter beschreiben die Winkel.

Die Funktion **RotVec** rotiert einen dreidimensionalen Vektor um die mitgegebenen Winkel. Dabei ist die Reihenfolge der Drehung wie beim ROT Z, Y und X. RotVec ist eine reine Berechnungsroutine zum Drehen eines Vektors und hat keine Auswirkungen auf ROT bzw. AROT.

5.2.13 Spiegeln

Die Spiegeln-Funktionalität (Mirror) ändert das Vorzeichen benannter Achsen. Auf diese Art können Subroutinen nochmal verwendet werden.

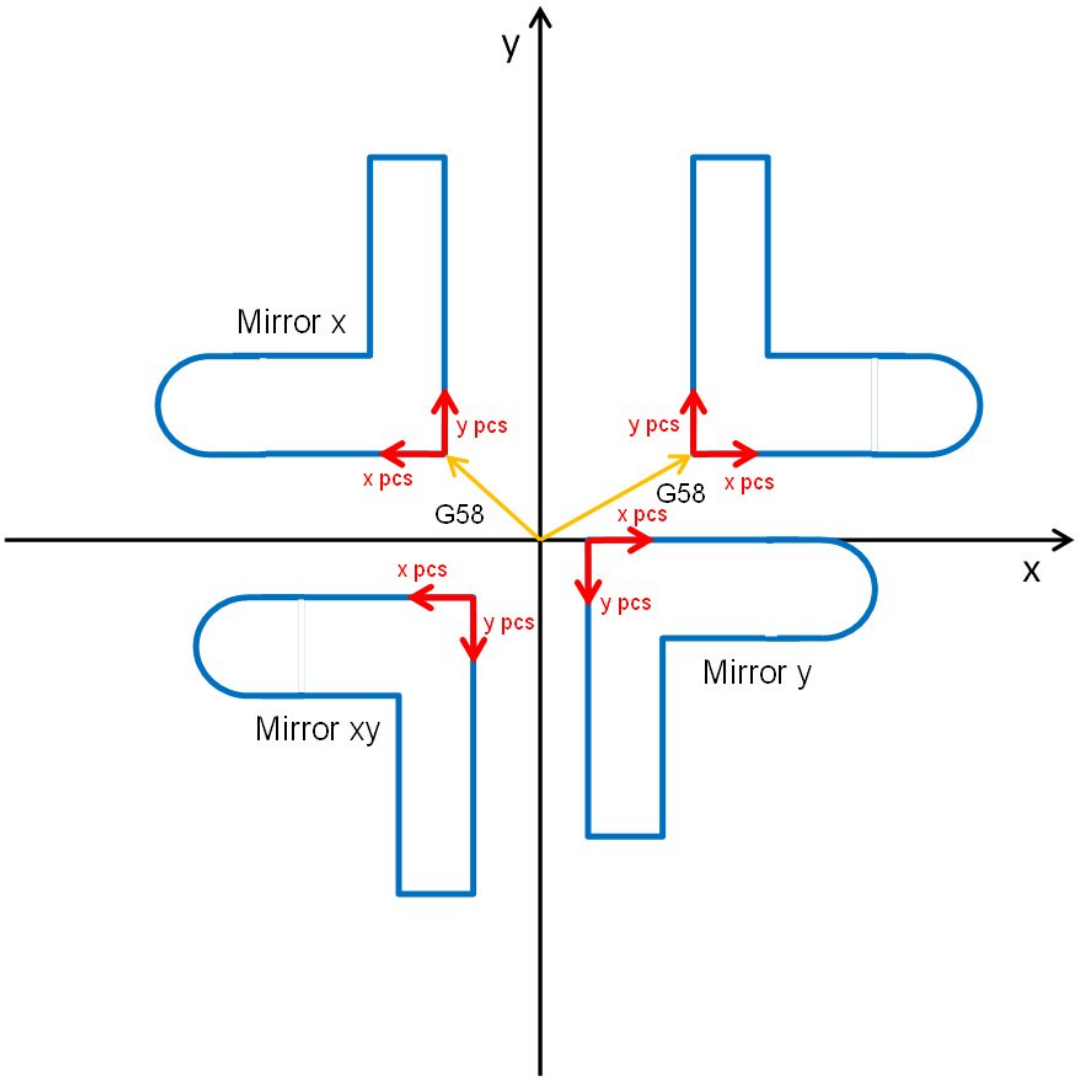
Spiegeln

Command	Mirror <opt. X> <opt. Y> <opt. Z>
Cancellation	Mirror (without parameters)

Die Spiegeln-Anweisungen müssen in ihrem eigenen Block programmiert werden. Gespiegelte Achsen müssen ohne weitere Parameter benannt werden.

Beispiel:

```
N20 G54
N30 G58 X100 Y100
N40 L100

N50 G58 X-100 Y100
N60 Mirror X
N70 L100

N80 G58 X-50 Y-50
N90 Mirror X Y
N100 L100

N110 G58 X10 Y-10
N120 Mirror Y
N130 L100

N140 Mirror (turn off mirror)
N150 G0 X0 y0
M02

L100
N1000 G0 X200 Y0 Z10 F60000 (move to start pos)
N1020 G01 Z0
N1030 G03 X200 Y100 J50
N1040 G01 X50
N1050 G01 Y400
N1060 G01 X0
N1070 G01 Y0
N1080 G01 X200
N1090 G01 Z10
M17
```



Falls eine Nullpunktverschiebung vorliegt (G54...G59) ist die Spiegeln-Funktionalität abhängig vom derzeitig programmierten Koordinatensystem.

5.2.14 Verschleifung von Segmentübergängen

5.2.14.1 Übersicht

Übersicht

Im Allgemeinen enthalten Polygon-Funktionen (`G01`-Sätze) an Segmentübergängen Knicke innerhalb ihrer Kontur. An diesen Übergängen sind Polygon-Funktionen in Bezug auf ihre Ortskoordinate nicht stetig differenzierbar, was auf diese Weise zu dynamischen Unstetigkeiten führt, wenn an diesen Übergängen die Pfadgeschwindigkeit nicht auf den Wert Null abgesenkt wird. Um es zu vermeiden, die Pfadgeschwindigkeit auf den Wert Null absenken zu müssen, können Segmentübergänge von Polygon-Funktionen durch Verschleifung an diesen Übergängen geglättet werden.

	Ausführung	Unterstützte Segment-übergänge	Beschleunigung der Achskomponenten	Max. Toleranzgröße	Adaptiver Toleranzradius	Befehl
Kreisverrundung [▶ 160]	Interpreter	Gerade-Gerade	Beschleunigungssprung (Höhe mit dem C1-Faktor parametrierbar)	1/2 des Ein- bzw. Ausgangssegments	Nein	paramCircularSmoothing(…)
Parabel-Verschleifung [▶ 157] <type>: **2**	NC-Kern	Gerade-Gerade	Beschleunigungssprung auf ein konstantes Niveau (Höhe mit dem C1-Faktor parametrierbar)	1/3 des Ein- bzw. Ausgangssegments	Kann angewählt werden	paramVertexSmoothing(…)
Bi-Quadratische - Verschleifung [▶ 157] <type>: **3**	NC-Kern	Gerade-Gerade	Stetige Beschleunigung - beim Ein- und Austritt ist die Beschleunigung 0 - es ist kein Zwischenpunkt erforderlich	1/3 des Ein- bzw. Ausgangssegments	Kann angewählt werden	paramVertexSmoothing(…)
Bézier-Verschleifung 3-ter Ordnung [▶ 157] <type>: **4**	NC-Kern	Alle	Beschleunigungssprung auf ein lineares Niveau (Höhe mit dem C1-Faktor parametrierbar)	1/3 des Ein- bzw. Ausgangssegments	Kann angewählt werden, wirkt bei Gerade-Geraden-Übergängen	paramVertexSmoothing(…)
Bézier-Verschleifung 5-ter Ordnung [▶ 158] <type>: **5**	NC-Kern	Alle	Stetige Beschleunigung - beim Ein- und Austritt ist die Beschleunigung 0 - es ist kein Zwischenpunkt erforderlich	1/3 des Ein- bzw. Ausgangssegments	Kann angewählt werden, wirkt bei Gerade-Geraden-Übergängen	paramVertexSmoothing(…)
‚Alte' Bézier-Verschleifung [▶ 158] <type>: **1**	NC-Kern	Alle	Stetige Beschleunigung - beim Ein- und Austritt, sowie im symmetrischen Zwischenpunkt ist die Beschleunigung 0	1/4 des Ein- bzw. Ausgangssegments	Nein	paramSplineSmoothing(…) paramVertexSmoothing(…)

Eine Verschleifung wirkt ab dem Übergang zwischen den nachfolgenden zwei Segmenten.

Wirkungsweise der Verschleifung

Der Radius der Toleranzkugel ist jederzeit im NC-Programm veränderbar und kann mit dem Radius 0 wieder ausgeschaltet werden. Eine Verschleifung bleibt bis zum nächsten Reset des Interpreters oder TwinCAT-Runtime-Restart aktiv.

5.2.14.2 Parabel-Verschleifung

Parabel-Verschleifung

Befehl	#set paramVertexSmoothing(<type>; <subtype>; <radius>)#
Parameter <type>	für die Parabel-Verschleifung: **2**
Parameter <subtype>	1: konstanter Toleranzradius [▶ 160] 2: Abstand Schnittpunkt zum Scheitelpunkt [▶ 160] 3: adaptiver Toleranzradius [▶ 160]
Parameter <radius>	max. Radius der Toleranzkugel

Bei der Parabel-Verschleifung wird geometrisch eine Parabel im Segmentübergang eingefügt. Dadurch wird die Geschwindigkeit innerhalb des Toleranzradius stetig überführt.

Die Parabel wird lediglich bei Geraden-Geraden-Übergängen eingefügt.

5.2.14.3 Bi-Quadratische-Verschleifung

Bi-Quadratische-Verschleifung

Befehl	#set paramVertexSmoothing(<type>; <subtype>; <radius>)#
Parameter <type>	für die Bi-Quadratische-Verschleifung: **3**
Parameter <subtype>	1: konstanter Toleranzradius [▶ 160] 2: Abstand Schnittpunkt zum Scheitelpunkt [▶ 160] 3: adaptiver Toleranzradius [▶ 160]
Parameter <radius>	max. Radius der Toleranzkugel

Bei der Bi-Quadratischen-Verschleifung tritt kein Beschleunigungssprung in den Achskomponenten auf. Bei gleichem Radius ist deshalb evtl. eine kleinere Eingangsgeschwindigkeit als bei der Parabel notwendig.

Die Wirkungsweise der Subtypen ist mit den Subtypen der Parabel identisch.

5.2.14.4 Bezier-Verschleifung 3-ter Ordnung

Bezier-Verschleifung 3-ter Ordnung

Befehl	#set paramVertexSmoothing(<type>; <subtype>; <radius>)#
Parameter <type>	für die Bezier-Verschleifung 3-ter Ordnung: **4**
Parameter <subtype>	1: konstanter Toleranzradius [▶ 160] 2: Abstand Schnittpunkt zum Scheitelpunkt [▶ 160] 3: adaptiver Toleranzradius [▶ 160]
Parameter <radius>	max. Radius der Toleranzkugel

Bei der Bezier-Verschleifung 3-ter Ordnung tritt mit dem Eintritt in die Toleranzkugel ein Beschleunigungssprung in den Achskomponenten auf. Die max. Größe wird durch die Beschleunigungen der Achskomponenten und den C1-Faktor begrenzt.

Diese Verschleifung ist für alle Segmentübergänge einsetzbar. Die Subtypen 2 und 3 wirken nur für Geraden-Geraden-Übergänge.

Spitze Winkel am Segmentübergang

Die Bezier-Splines werden standardmäßig auch bei sehr spitzen Winkeln generiert. Damit die Dynamikwerte nicht überschritten werden, ist für diesen Fall eine erhebliche Geschwindigkeitsreduktion erforderlich. Da im Spline die Dynamik konstant gehalten wird, dauert es entsprechend lange, bis der Spline durchfahren wird. Für diesen Fall ist es häufig sinnvoll, den Segmentübergang mit einem Genauhalt anzufahren. Damit die Winkel nicht manuell berechnet werden müssen, gibt es den Befehl AutoAccurateStop [▶ 161].

5.2.14.5 Bezier-Verschleifung 5-ter Ordnung

Bezier-Verschleifung 5-ter Ordnung

Befehl	#set paramVertexSmoothing(<type>; <subtype>; <radius>)#
Parameter <type>	für die Bezier-Verschleifung 5-ter Ordnung: **5**
Parameter <subtype>	1: konstanter Toleranzradius [▶ 160] 2: Abstand Schnittpunkt zum Scheitelpunkt [▶ 160] 3: adaptiver Toleranzradius [▶ 160]
Parameter <radius>	max. Radius der Toleranzkugel

Bei der Bezier-Verschleifung 5-ter Ordnung tritt mit dem Eintritt in die Toleranzkugel **kein** Beschleunigungssprung in den Achskomponenten auf. D. h. bei angewählter Verschleifung, ist der Beschleunigungsverlauf für die Bahnachsen immer stetig.

Diese Verschleifung ist für alle Segmentübergänge einsetzbar. Die Subtypen 2 und 3 wirken nur für Geraden-Geraden-Übergänge.

Spitze Winkel am Segmentübergang

Die Bezier-Splines werden standardmäßig auch bei sehr spitzen Winkeln generiert. Damit die Dynamikwerte nicht überschritten werden, ist für diesen Fall eine erhebliche Geschwindigkeitsreduktion erforderlich. Da im Spline die Dynamik konstant gehalten wird, dauert es entsprechend lange, bis der Spline durchfahren wird. Für diesen Fall ist es häufig sinnvoll, den Segmentübergang mit einem Genauhalt anzufahren. Damit die Winkel nicht manuell berechnet werden müssen, gibt es den Befehl AutoAccurateStop [▶ 161].

5.2.14.6 Alte Bezier-Verschleifungen

Funktionen zur Kompatibilität mit bestehenden Projekten

Diese Funktionen werden aus Kompatibilitätsgründen noch zur Verfügung gestellt. Für neue Projekte sollte die Bezier-Verschleifung 3-ter Ordnung [▶ 157] oder die Bezier-Verschleifung 5-ter Ordnung [▶ 158] verwendet werden.

Alte Bezier-Verschleifung mit paramVertexSmoothing

Befehl	#set paramVertexSmoothing(<type>; <subtype>; <radius>)#
Parameter <type>	für die Bezier-Spline-Verschleifung: **1**
Parameter <subtype>	für die Bezier-Spline-Verschleifung: **1**
Parameter <radius>	Radius der Toleranzkugel

Beispiel 1:

```
N10 R57=100
#set paramVertexSmoothing(1; 1;R57)#
```

Alte Bezier-Verschleifung mit paramSplineSmoothing

Mit Hilfe der Glättung ist es möglich, zwischen zwei Geometrieeinträgen automatisch ein Bezier-Spline einzufügen. Hierfür muss lediglich der Radius der Toleranzkugel programmiert werden. Dieser beschreibt die maximal erlaubte Abweichung von der programmierten Kontur im Segmentübergang. Der Vorteil bei dieser Art der Glättung gegenüber der Verrundung mit Kreiselement ist, dass hier an den Segmentübergängen keine Beschleunigungssprünge entstehen.

Der Radius der Toleranzkugel ist jederzeit im NC-Programm veränderbar und kann mit dem Radius 0 wieder ausgeschaltet werden. Wird der Radius nicht auf 0 zurückgesetzt, bleibt er bis zum nächsten Reset des Interpreters oder TwinCAT-Restart aktiv.

Befehl	#set paramSplineSmoothing(<radius>)#
Parameter <radius>	Radius der Toleranzkungel

oder Alternativ

#set paramVertexSmoothing(...)

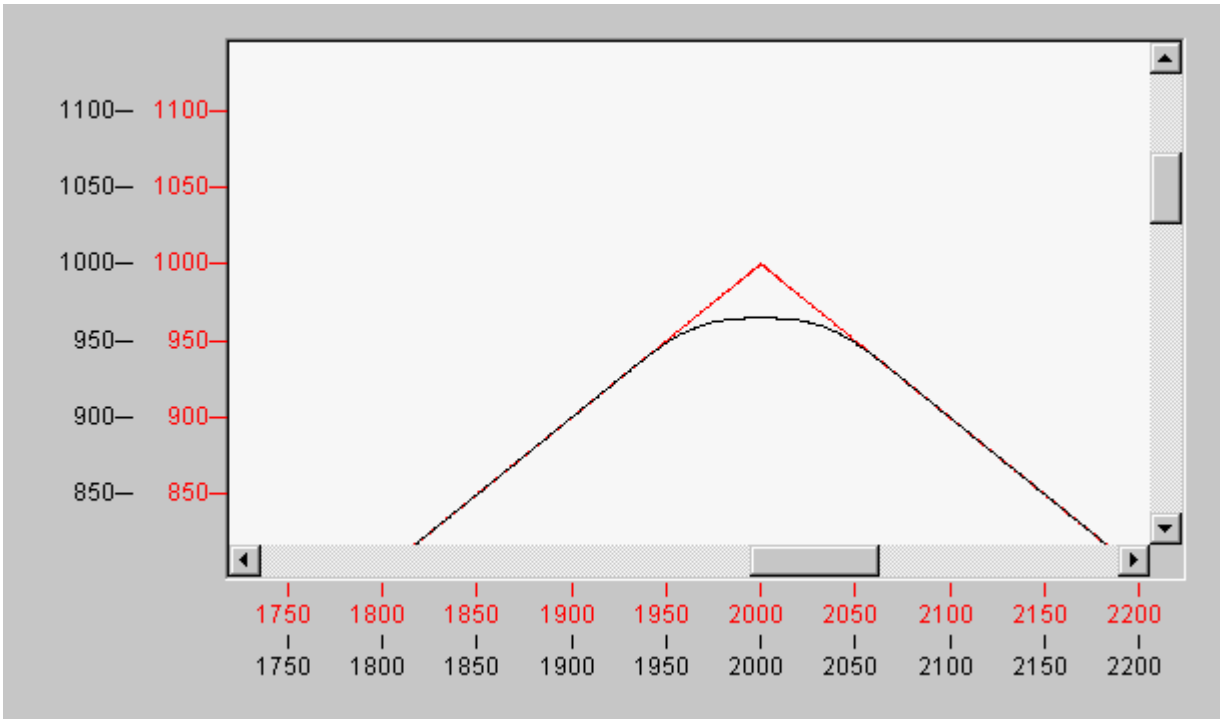
Beispiel 1:

```
N10 R57=100
#set paramSplineSmoothing(R57)#
```

Beispiel 2:

```
N10 G01 X0 Y0 F6000
N20 X1000
#set paramSplineSmoothing(100)#
N30 X2000 Y1000
N40 X3000 Y0
M30
```

Der neue Parameter ist ab dem Übergang zwischen den nachfolgenden zwei Segmenten gültig. D.h. in Beispiel 2 ist der neue Wert für die Toleranzkugel im Segmentübergang von N30 zu N40 gültig. Im folgenden Bild sehen Sie eine Kontur mit und ohne Spline im Segmentübergang.

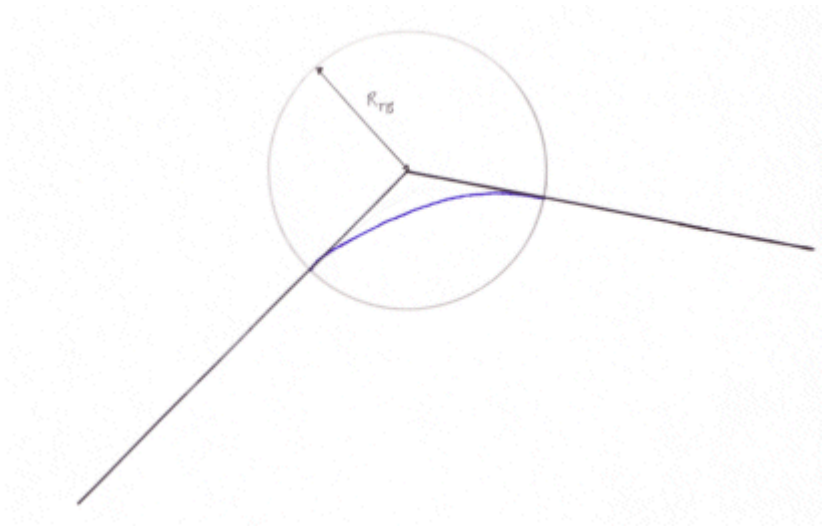


Die Splines werden standardmäßig auch bei sehr spitzen Winkeln generiert. Damit die Dynamikwerte nicht überschritten werden, ist für diesen Fall eine erhebliche Geschwindigkeitsreduktion erforderlich. Da aber im Spline die Dynamik konstant gehalten wird, dauert es entsprechend lange, bis der Spline durchfahren wird. Für diesen Fall ist es häufig sinnvoll, den Segmentübergang mit einem Genauhalt anzufahren. Damit die Winkel nicht manuell berechnet werden müssen, gibt es den Befehl 'AutoAccurateStop [▶ 161]', der ebenfalls aus dem NC-Programm aufgerufen werden kann.

5.2.14.7 Subtypen

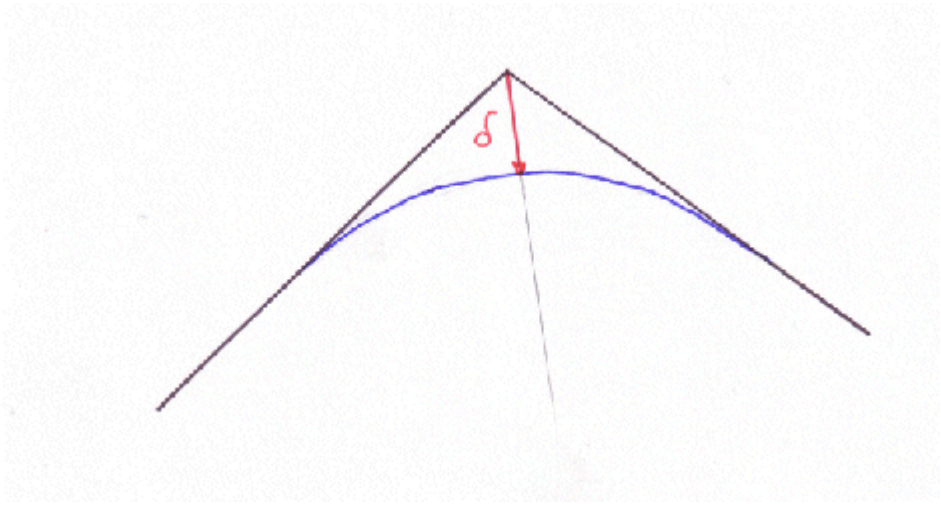
Konstanter Toleranzradius (Subtyp 1)

Ist Subtyp 1 angewählt, so wird immer der maximale Toleranzradius (R_{TB}) für die Verschleifung verwendet. Eine Verkleinerung von R_{TB} erfolgt genau dann, wenn das Ein- bzw. Ausgangssegment kleiner als $3*R_{TB}$ ist.



Abstand vom Schnittpunkt zum Scheitelpunkt (Subtyp 2)

Mit dem Subtyp 2 wird der Abstand vom programmierten Segmentübergang zum Scheitelpunkt der Parabel vorgegeben. Anhand dessen ergibt sich dann der Toleranzradius (R_{TB}). Ist ein Segment zu kurz, so wird der Abstand so verkürzt, dass der Toleranzradius max. 1/3 einnimmt.



Adaptiver Toleranzradius (Subtyp 3)

Innerhalb des Toleranzradius (auch beim konstanten Toleranzradius) wird immer sichergestellt, dass die maximal zulässige Beschleunigung nicht überschritten wird. Je nach Ablenkungswinkel und Geschwindigkeit kann damit die maximal auftretende Achsbeschleunigung im Verrundungssegment unterschiedlich ausfallen. Ziel beim adaptiven Toleranzradius ist eine maximale Beschleunigung innerhalb der Verschleifung. Um dies zu erreichen, wird der Verrundungsradius unter Berücksichtigung der programmierten Geschwindigkeit und Dynamik entsprechend verkleinert. D.h. wird die programmierte Geschwindigkeit verändert, so kann sich auch der Toleranzradius ändern. Der Override hat keinen Einfluss auf den Radius.

5.2.15 Verrundung mit Kreissegmenten

Mit Hilfe der Verrundung ist es möglich, zwischen zwei Geradenelementen automatisch ein Kreiselement einzufügen. Hierfür muss lediglich der Radius des Kreiselements programmiert werden.

Der Radius der Verrundung ist jederzeit im NC-Programm veränderbar und kann mit dem Radius 0 wieder ausgeschaltet werden. Vor dem Programmende oder einem Dekodierstopp [▶ 172] ist es erforderlich die Verrundung auszuschalten.

Befehl	#set paramCircularSmoothing(<radius>)#
Parameter <radius>	Radius des Kreiselements der Verrundung

Beispiel:

```
N10 R57=4.5
#set paramCircularSmoothing(R57)#
…
#set paramCircularSmoothing(0)#
N1000 M02
```

Hinweis In Kombination mit der Fräserradiuskorrektur ist zu beachten, dass zuerst die Radiuskorrektur berechnet und anschließend die Kreisverrundung eingefügt wird. Somit bezieht sich der Verrundungsradius auf den TCP.

Hinweis Das alte Kommando paramGroupVertex wird weiterhin unterstützt. Es ist hier allerdings nicht möglich R-Parameter zu übergeben.

Syntax:

```
#set paramGroupVertex(<grp>,<radius>)#
```

Der erste Parameter beschreibt die Gruppe, auf den sich die Verrundung bezieht. Dieser Wert ist zurzeit immer 1. Mit dem zweiten Parameter wird der Radius der Verrundung festgelegt.

5.2.16 Automatischer Genauhalt

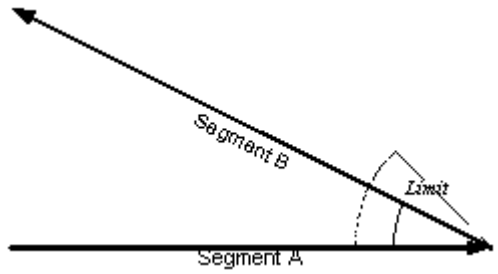
Befehl	#set paramAutoAccurateStop(<winkel>)#
Parameter <winkel>	Grenzwinkel (in Grad) ab dem ein Genauhalt eingefügt wird
Abwahl	#set paramAutoAccurateStop(**0**)#

Mit dem Kommando 'AutoAccurateStop' wird ab einem definierten Grenzwinkel ein Genauhalt zwischen 2 Segmenten eingefügt.

Bei Kreissegmenten errechnet sich der Winkel aus den Eingangs- bzw. Ausgangstangenten.

Beispiel:

```
#set paramAutoAccurateStop(45)# (angle in
degrees)
N10 G01 X1000 Y0 Z0 F60000 (start position: X0 Y0 Z0)
N20 X0 Y500
...
```



Für dieses Beispiel wird zwischen den Segmenten A und B ein Genauhalt eingefügt.

Anwendungsgebiet:

Dieses Kommando sollte in Verbindung mit Bezier-Verschleifungen verwendet werden, wenn im NC-Programm spitze Winkel programmiert werden.

Siehe auch:

- Bezier-Verschleifung 3-ter Ordnung [▶ 157]
- Bezier-Verschleifung 5-ter Ordnung [▶ 158]
- 'Alte' Bezier-Verschleifung [▶ 158]

Hinweis **Für Segmentübergänge mit einer Helix ist diese Funktion noch nicht implementiert.**

5.2.17 Restweglöschen

Befehl	DelDTG
Aufhebung	Satzende

DelDTG (**del**ete **d**istance **t**o **g**o) wird aus dem NC-Programm satzweise aktiviert. Dieses Kommando ermöglicht es, aus der SPS mit dem Funktionsbaustein ItpDelDtgEx [▶ 213] den Restweg der aktuellen Geometrie zu löschen. D.h. trifft das Kommando während der Abarbeitung des Satzes ein, so wird mit den üblichen Verzögerungsrampen die Bewegung angehalten. Anschließend wird mit dem nächsten Satz im NC-Programm fortgefahren. Falls das SPS-Kommando nicht während der Ausführung eines Satzes mit angewähltem Restweglöschen eintrifft, wird dieses mit einer Fehlermeldung beantwortet.

Das Restweglöschen bewirkt immer einen impliziten Dekodierstop, d.h. es erfolgt am Satzende immer ein Genauhalt.

Beispiel:

```
N10 G01 X0 Y0 F6000
N20 DelDTG G01 X2000
N30 G01 X0
```

Hinweis **DelDTG darf nicht bei aktiver Fräserradiuskorrektur aktiv sein.**

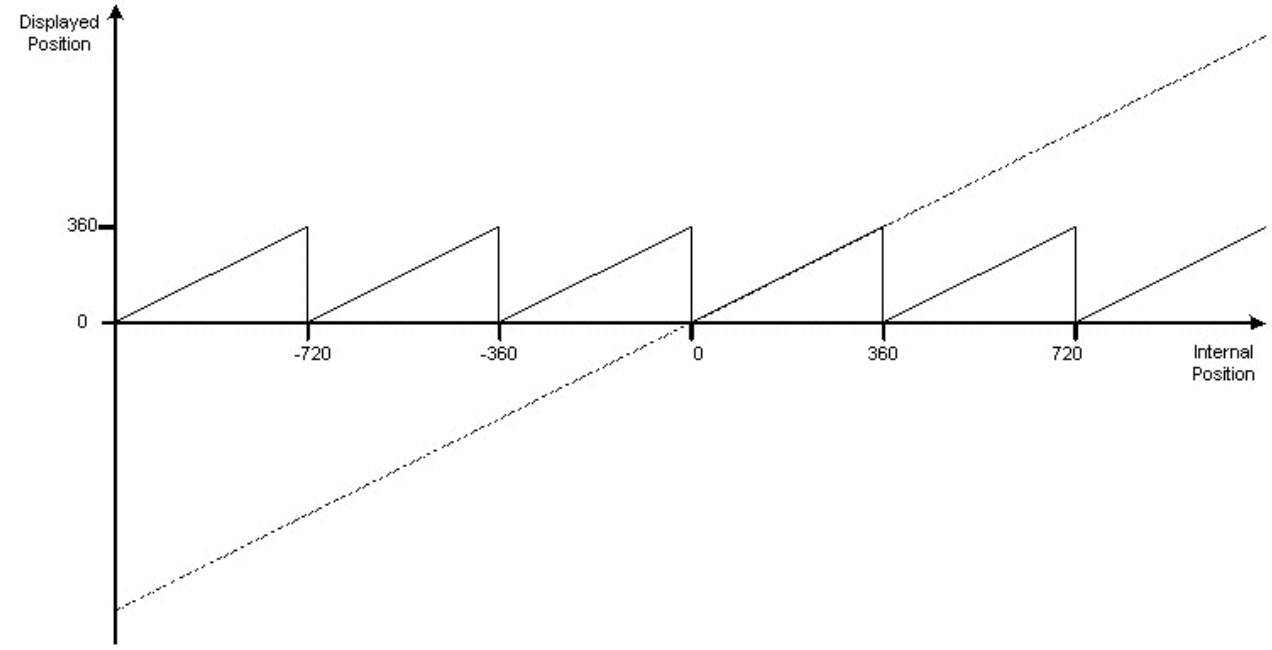
5.2.18 Modulo Bewegungen

Befehl	MOD[<axis and target modulo position>]
Aufhebung	Satzende
Parameter 1	Achse die modulo verfahren werden soll
Parameter 2	Vorzeichen für die Drehrichtung (optional)
Parameter 3	Modulo Position

Die Programmierung der Moduloposition erfolgt auf gleiche Weise, wie bei einer gewöhnlichen Positionierung.

Das MOD-Kommando gilt satzweise und muss für jede Achse, die Modulo verfahren werden soll, explizit programmiert werden. Dabei bestimmt das Vorzeichen der Moduloposition die Drehrichtung.

- Positives Vorzeichen: Die Achse fährt in Richtung 'größer'
- Negatives Vorzeichen: Die Achse fährt in Richtung 'kleiner'
- Ausnahme: Modulo -0 kann nicht angefahren werden, da 0 vorzeichenlos ist



Beispiel 1:

```
N10 G90
N20 G01 MOD[X200] Y30 F600
N30 G01 X200
```

In N20 wird in positiver Richtung für X die Moduloposition 200 angefahren. Y verfährt auf die absolute Position 30. Im Satz N30 wird X absolut auf Position 200 verfahren, also **nicht** modulo.

Modulo Bewegung anwendbar auf Q-Achsen

Die Q-Achsen sind die Hilfsachsen. Das MOD-Kommando kann auf Q-Achsen angewendet werden. Z. B. `N20 G01 MOD[Q4=200] Y30 F600.`

Modulobewegungen mit mehr als 360 Grad

Mit dem MOD-Kommando lassen sich auch Bewegungen mit mehr als 360 Grad verfahren.

Moduloposition = Anzahl der erforderlichen Umdrehungen * 360 + Moduloposition

Beispiel 2:

```
N10 G90
N20 G01 X3610 F6000
N30 R1=360
N40 G01 MOD[X=R1+20]
```

In diesem Beispiel verfährt die X Achse um 360 Grad auf die Moduloposition 20.

Einschränkungen und Hinweise bei Modulobewegungen:

- Es darf keine Radiuskorrektur für die Moduloachse aktiv sein.
- Es darf keine Nullpunktverschiebung für die Moduloachse aktiv sein.
- Bei Relativprogrammierung (G91 [▶ 132]) wird das Modulokommando nicht ausgewertet, d.h. die in den eckigen Klammern aufgerufene Achse wird so behandelt, als ob das MOD-Kommando nicht eingegeben wurde.

Modulofaktor

Der Modulofaktor ist konstant und beträgt 360.

5.2.19 Hilfsachsen

Hilfsachsen (auch Auxiliary oder Q-Achsen genannt) können zusätzlich zu den eigentlichen Bahnachsen (X, Y & Z) einer Interpolationsgruppe angefügt werden. Dabei muss man sich die Hilfsachse als eine Art Slave zur Bahn vorstellen. D.h. sie haben keinen direkten Einfluss auf die Bahngeschwindigkeit. Insgesamt können neben den 3 Bahnachsen noch 5 Hilfsachsen pro Kanal mit interpoliert werden.

Das Anfügen in die Interpolationsgruppe aus der SPS kann z.B. mit dem Funktionsbaustein CfgBuildExt3DGroup [▶ 204] aus der Bibliothek Tc2_NCI erfolgen.

Syntax

Aus dem Teileprogramm werden die Hilfsachsen mit Q1..Q5 angesprochen. Es kann dabei entweder direkt der Zahlenwert oder ein R-Parameter zugewiesen werden.

Beispiel 1:

```
(start position X=Y=Z=Q1=0)
N10 G01 X100 Q1=47.11 F6000
...
```

Wird ein NC-Satz mit Bahnachse(n) und einer Hilfsachse programmiert, so starten beide Achsen **gleichzeitig** und kommen auch **gemeinsam** ins Ziel.

Schwenken der Hilfsachsen

Von einem Schwenken der Hilfsachsen wird dann gesprochen, wenn der Bahnweg in einem Verfahrsatz null ist. Dies ist häufig beim 'Schwenken' eines Werkzeugs der Fall, wo der Zustellwinkel zur Kontur geändert wird.

Da es hier keinen Bezug mehr zur Bahn gibt, da der Bahnweg null ist, werden die Bewegungen der Hilfsachsen mit einer virtuellen Bahn berechnet. Dies hat aber keinen Einfluss auf die reale Bahn von X, Y und Z, sondern bewirkt auch hier, dass alle Hilfsachsen gleichzeitig gestartet werden und ebenfalls zeitgleich im Ziel ankommen.

Die Geschwindigkeit wird auch hier mit dem F-Parameter vorgegeben und bezieht sich jetzt auf die Hilfsachse mit dem größten Verfahrweg.

Beispiel 1:

```
(start position X=Y=Z=Q1=Q2=0)
N10 G01 X100 F6000
N20 Q1=100 Q2=200 F3000
...
```

In N20 ist die Geschwindigkeit von Q2 nun 3000 und von Q1 1500, da der Verfahrweg von Q1=Q2/2 ist.

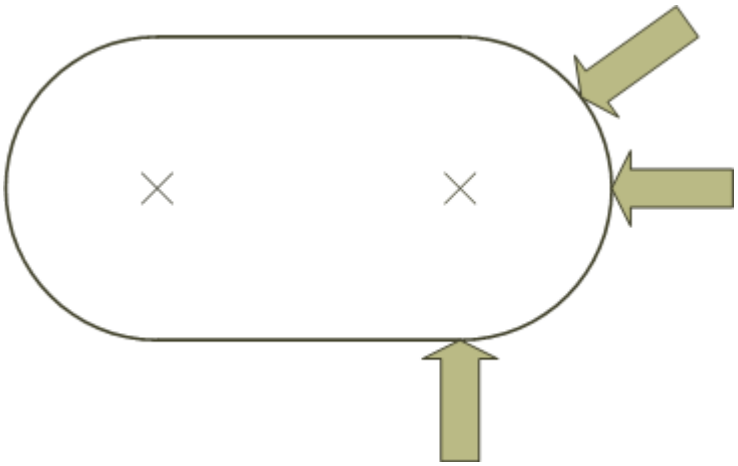
5.2.19.1 Berechnung der Geschwindigkeit

Für die Berechnung der Bahngeschwindigkeit werden zunächst nur die Bahnachsen (X, Y und Z) berücksichtigt.
Anhand des Bahnweges und der zu verfahrenden Distanz der einzelnen Hilfsachsen ergibt sich ein festes Kopplungsverhältnis für jede Hilfsachse in einem Segment. Damit ist auch die Sollgeschwindigkeit der Hilfsachse bekannt. Falls diese Geschwindigkeit größer ist, als die zulässige maximale Geschwindigkeit dieser Hilfsachse, so wird die Bahngeschwindigkeit soweit reduziert, bis die obere Geschwindigkeitsgrenze eingehalten wird. D.h. wenn die Geschwindigkeitsgrenzwerte der Hilfsachsen überschritten werden, hat das auch indirekt einen Einfluss auf die Bahngeschwindigkeit.

5.2.19.2 Bahngeschwindigkeit an Segmentübergängen

Im Folgenden soll die Reduzierung der Bahngeschwindigkeit anhand eines Beispiels näher erläutert werden. Dazu eignet sich besonders gut die Kontur eines Stadions. Ziel ist es, den Zustellwinkel eines Werkzeugs zur Bahntangente konstant zu halten.

Befindet man sich auf der Geraden des Stadions, so bleibt die Orientierung des Werkzeugs konstant, d.h. das Werkzeug wird nicht gedreht. Im Kreis dagegen muss die Orientierung in Bezug auf das Basiskoordinatensystem ständig geändert werden. Angenommen die Bahngeschwindigkeit wird im Geraden-Kreis-Übergang nicht bis auf null reduziert, so entsteht zwangsläufig ein Geschwindigkeitssprung für die Drehachse (nicht aber für die Bahnachsen!).



Dieser Geschwindigkeitssprung der Hilfsachse ist frei parametrierbar und von der Maschine abhängig. In den beiden Extremfällen wird einmal die Bahngeschwindigkeit an derartigen Segmentübergängen bis auf null reduziert oder im anderen Fall die Geschwindigkeit gar nicht reduziert.

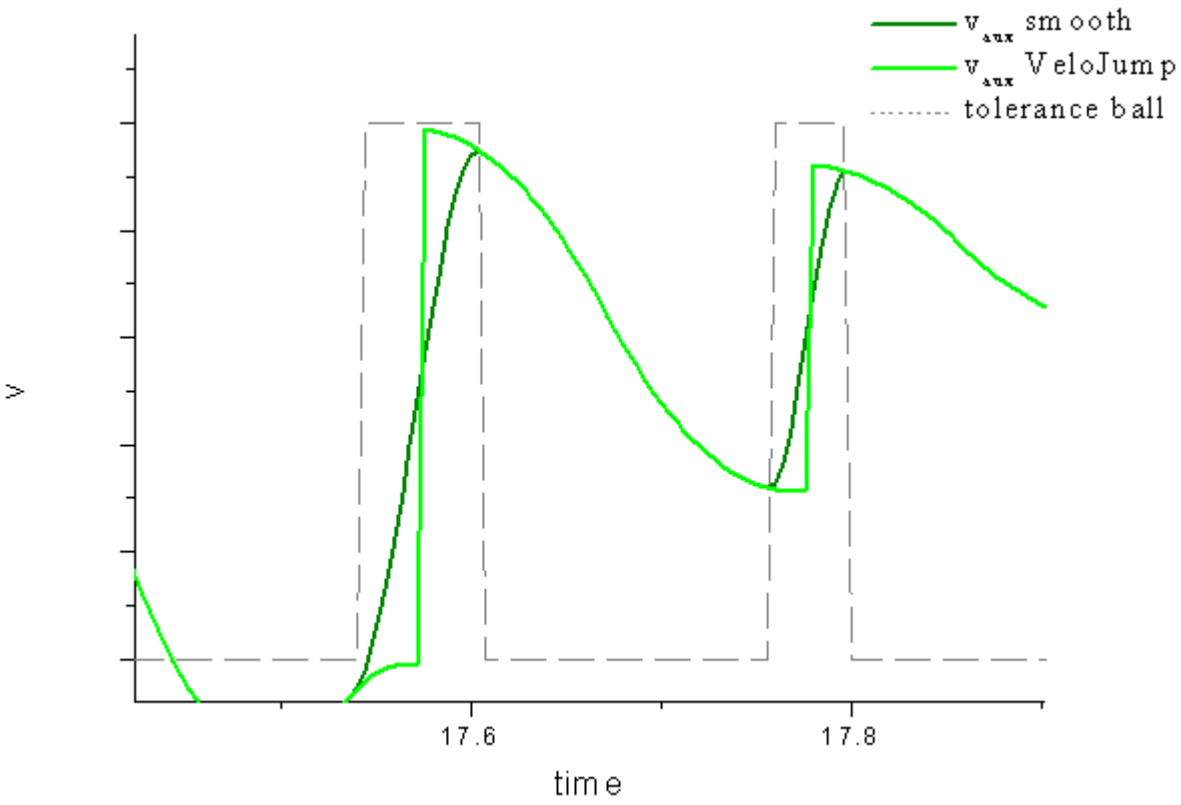
Die Parametrierung erfolgt mit dem globalen Achsparameter 'VeloJumpFactor', der für jede Achse individuell eingestellt werden kann. Die daraus resultierende Geschwindigkeit und die Berechnung ist im TwinCAT NCI Anhang auf der Seite Parametrierung [▶ 330] näher beschrieben.

Glättung der Geschwindigkeit an Segmentübergängen

Wie oben beschrieben, können an den Segmentübergängen Geschwindigkeitssprünge auftreten. Die Größe dieses Sprunges kann mit dem VeloJump-Parameter beeinflusst werden.

Zusätzlich kann für jede Hilfsachse eine Toleranzkugel bestimmt werden, die im Segmentübergang symmetrisch mit dem Bahnweg ist. Mit dem Eintritt in diese Kugel wird dann die Geschwindigkeit der Hilfsachse stetig in die neue Sollgeschwindigkeit am Kugelausgang überführt. D.h. die Geschwindigkeitssprünge werden damit eliminiert. Dabei wird innerhalb der Kugel eine Positionsabweichung für die Hilfsachse in Kauf genommen. Mit dem Eintritt in die Kugel wird schon damit begonnen die Achse auf die neue Zielgeschwindigkeit zu bringen. Damit wird ein Überschwingen der Position verhindert und an den Kugelgrenzen ist die Position dann wieder genau.

Für den Fall, dass die vorgegebene Kugel größer als 1/3 des Bahnweges ist, so wird der Radius automatisch auf diesen Wert begrenzt.



An- und Abwahl

Die Toleranzkugel der Hilfsachse ist ein Achsparameter (IO: 0x108). Sie kann im TwinCAT XAE Achsinterface via ADS eingestellt werden.

Parametrierung der Achsparameter

Die hier beschriebenen Parameter wirken nur für Achsen, die sich als Hilfsachsen (Q1..Q5) in der Interpolationsgruppe befinden. Für Bahnachsen (x,y,z) haben die Parameter 'Geschw. Sprung Faktor' ('Velo Jump Factor'),'Toleranzkugel Hilfsachse' ('Tolerance ball auxiliary axis') und 'Max. Positionsabweichung, Hilfsachse' ('Max. position deviation, aux. axis') keinen Einfluss.

Diagnose

Zu Diagnosezwecken kann für jede Hilfsachse die Toleranzkugel und der sich ergebene Positionsfehler der Hilfsachse aufgezeichnet werden. Auf die Variablen kann ebenfalls via ADS zugegriffen werden und befinden sich im Gruppenzustand (IO: 0x54n und 0x56n).

Rückwirkung auf VeloJump, falls die Toleranzkugel verkleinert wird

Wenn aufgrund der gegebenen Geometrie die Toleranzkugel verkleinert werden muss, so wird nun auch für diesen Segmentübergang der VeloJump-Parameter automatisch angepasst. D.h. falls notwendig wird die Bahngeschwindigkeit im Übergang stärker reduziert. Damit wird erreicht, dass bei kleineren Toleranzkugel die Dynamik der Hilfsachse nicht überschritten wird.

Positionsabweichung der Hilfsachse, falls die Toleranzkugel verkleinert werden müsste

Der Parameter 'maximal erlaubte Positionsabweichung der Hilfsachse' wirkt **nur dann**, wenn die Toleranzkugel aufgrund der Geometrie verkleinert werden müsste.

Ziel ist es, dass die Bahngeschwindigkeit trotz der verkleinerten Toleranzkugel hoch gehalten werden kann, wenn der daraus resultierende Positionsfehler einen Schwellwert nicht überschreitet. Dafür wird die Geschwindigkeit der Hilfsachse konstant gehalten und der Positionsfehler berechnet. Ist dieser kleiner als die maximale Positionsabweichung, dann wird für diesen Segmentübergang die Geschwindigkeit beibehalten und der entstandene Positionsfehler im nächsten Segment wieder ausgefahren (die Toleranzkugel wird dann für diesen Segmentübergang überflüssig).

Für den Fall, dass der Positionsfehler die max. Abweichung überschreiten würde, greift die verkleinerte Toleranzkugel mit der Rückwirkung auf den VeloJump-Faktor. Dabei wird dann ggf. die Bahngeschwindigkeit reduziert.

Beispiel 1:

Anfangsbedinungen:

- eingestellte Toleranzkugel: 5
- max. Positionsabweichung: 1
- aufgrund der Geometrie ergibt sich z.B. eine effektive Toleranzkugel von 0.2
- die potenzielle Positionsabweichung beträgt 0.3

Resultierendes Verhalten:

- Bahngeschwindigkeit bleibt auf konstant hohen Niveau
- Geschwindigkeit der Hilfsachse wird konstant gehalten
- für diesen Übergang ist keine Toleranzkugel erforderlich
- die entstandene Positionsabweichung wird im darauffolgenden Segment ausgeglichen

Beispiel2:

Anfangsbedingungen:

- eingestellte Toleranzkugel: 5
- max. Positionsabweichung: 1
- aufgrund der Geometrie ergibt sich z.B. eine effektive Toleranzkugel von 1.2
- die potenzielle Positionsabweichung beträgt 1.1

Resultierendes Verhalten:

- Toleranzkugel wird angepasst
- VeloJump-Parameter wird angepasst
- die Bahngeschwindigkeit wird im Segmentübergang reduziert
- es entsteht **keine** Positionsabweichung, die ausgeglichen werden muss

Parametrierung

Die Parametrierung der maximal erlaubten Positionsabweichung ist ein Achsparameter. Standardmäßig ist diese Eigenschaft ausgeschaltet (Abweichung = 0.0)

5.3 Zusatzfunktionen

5.3.1 M-Funktionen

Aufgabe: Signalaustausch zwischen NC und PLC

Eine Reihe von Einrichtungen, wie z.B. Spannzangen, Bohrantrieben, Transporteinrichtungen etc. werden vorteilhaft nicht von der NC direkt, sondern indirekt unter Nutzung der SPS als Anpass- und Verknüpfungssteuerung kontrolliert. Dadurch ist es leicht möglich, Rückmeldungen oder Sicherheitsbedingungen zu berücksichtigen, ohne das NC-Programm oder sogar das NC-System anpassen zu müssen. Bei den M-Funktionen der NC handelt es sich um einen Signalaustausch mit digitalem Charakter: Es werden Funktionen ein- oder ausgeschaltet, aktiviert oder deaktiviert. Die Übergabe von Zahlenwerten als Arbeitsparameter ist dabei nicht vorgesehen, lässt sich aber auf anderem Wege (H-Funktion [▶ 172], T-Nummer [▶ 172] etc.) verwirklichen.

5.3.1.1 Verfügbare M-Funktionen

Anzahl M-Funktionen

Es stehen insgesamt 160 M-Funktionen pro Kanal zur Verfügung

M-Funktion	Bedeutung
0..159	Frei definierbare M-Funktionen (außer 2, 17, 30)
2	Programmende
17	Unterprogrammende
30	Programmende mit ablöschen von allen schnellen M-Funktionen

Abgesehen von den 3 fest vorbelegten M-Funktionen (M2, M17, M30) kann über die Bedeutung der restlichen M-Funktionen frei verfügt werden. D.h. je nach Maschinentyp kann z.B. mit M8 ein Kühlmittel eingeschaltet werden, es lässt sich aber auch eine andere Funktionalität damit ausführen. Dies ist soweit vom Maschinenbauer frei wählbar.

Die Regeln für die reservierten M-Funktionen werden wie alle anderen beim TwinCAT Start eingelesen. Zusätzlich wird im Interpreter bei diesen Funktionen noch ein interner Code generiert, der für das beschriebene Verhalten sorgt. Somit müssen diese 3 M-Funktionen nicht in der Tabelle beschrieben werden. Wenn M-Funktionen verwendet werden, ist es dennoch sinnvoll, dass M2 und M30 parametriert werden.

Vorrang von M-Funktionen in der TwinCAT-3-Entwicklungsumgebung

Wenn `M`-Funktionen sowohl in der Datei `m_defs.t<xx>` als auch in der Entwicklungsumgebung definiert sind, dann wirken nur diejenigen `M`-Funktionen, die in der Entwicklungsumgebung definiert sind.

Typen von M-Funktionen

Grundsätzlich stehen zwei Varianten des Signalaustausches zur Verfügung: Schnelle Signalbits oder per Handshake gesicherte Übermittlung.

Gesicherter Handshake

M-Funktionen mit Rückmeldungsbedarf müssen mit einem bidirektionalen Signalaustausch zwischen NC und SPS abgearbeitet werden. Wird eine M-Funktion vom Typ Handshake programmiert, so wird die Geschwindigkeit an dieser Stelle auf 0 reduziert. In der PLC wird mit der Funktion ItpIsHskMFunc [▶ 233] geprüft, ob eine M-Funktion mit Handshake anliegt. Die M-Funktionsnummer wird dann mit ItpGetHskMFunc [▶ 226] ermittelt. Solange keine Bestätigung der M-Funktion aus der PLC erfolgt, ist die NC im Wartezustand und arbeitet keine weiteren NC-Kommandos ab. Erst mit der Quittierung aus der PLC (ItpConfirmHsk [▶ 212]) wird die Abarbeitung des NC-Programms fortgeführt.

Diese Arbeitsweise ermöglicht es, die Arbeit der NC-kontrollierten und der SPS-kontrollierten Einrichtungen in der Maschine sicher zu koordinieren. So wird sinnvollerweise die M-Funktion zum Einschalten der Spindel (z.B. M3) dann quittiert, wenn eine Mindestdrehzahl erreicht ist.

Da es sich bei dieser Art der M-Funktionen um synchrone Funktionen handelt, kann immer nur eine M-Funktion mit Handshake im NC-Programm aktiv sein.

Schnelle Signal-Bits

Wenn keine Rückmeldung von der PLC erforderlich ist, so können schnelle Signal-Bits zum Aktivieren von M-Funktionen eingesetzt werden. Da die NC für diese M-Funktionen nicht auf die SPS warten muss, kann der Look-Ahead [▶ 129] die Segmente miteinander verbinden. So ist es möglich, ohne Geschwindigkeitsreduzierung eine M-Funktion zu schalten.

Über ItpIsFastMFunc kann eine schnelle M-Funktion in der SPS erkannt werden. Dies ermöglicht es, aus der SPS eine beliebige Aktion während einer Bewegung zu starten (Laser an/aus, Fräser an/aus, …). Anschließend ist die M-Funktion mit ItpResetFastMFuncEx zurückzusetzen. So ist es möglich, die M-Funktion mehrfach zu verwenden.

Eine Kombination von schnellen Signalbits und Handshake ist ebenfalls möglich. Da mit dem Handshake immer auf die PLC gewartet wird, muss für diesen Fall auch die Geschwindigkeit auf 0 reduziert werden.

5.3.1.2 Zurücksetzen von M-Funktionen

Schnelle Signal-Bits zurücksetzen

Die Signal-Bits liegen solange an, bis sie entweder explizit zurückgesetzt, ein M30 (Programmende) oder ein Kanal-Reset durchgeführt wird.

Zurücksetzen mit Reset-Liste

Jede M-Funktion kann bis zu 10 schnelle M-Funktionen zurücksetzen. Wird z.B. mit M8 das Kühlmittel eingeschaltet, so kann mit M9 das Kühlmittel wieder ausgeschaltet werden. Dazu muss lediglich M8 in die Reset-Liste von M9 eingetragen werden.

Zurücksetzen einer schnellen M-Funktion erfordert eine schnelle M-Funktion

Eine schnelle M-Funktion kann nur mit einer schnellen M-Funktion zurückgesetzt werden. Es gibt keine alternative Möglichkeit an dieser Stelle, eine schnelle M-Funktion mit einer nicht-schnellen Handshake M-Funktion zurückzusetzen.

Automatisches Zurücksetzen

Bei der Parametrierung der M-Funktion kann ein 'Auto-Reset-Flag' gesetzt werden. Damit wird die M-Funktion am Ende des Satzes automatisch wieder zurückgesetzt.

Damit die PLC die Möglichkeit hat, das Signal zu sehen, muss der Verfahrsatz zeitlich lang genug sein oder diese M-Funktion wird mit einem Handshake kombiniert. Dabei kann der Handshake von der gleichen oder einer anderen M-Funktionsnummer sein.

Zurücksetzen aus der PLC

Möchte man die schnellen M-Funktionen aus der PLC zurücksetzen, so ist das mit dem Funktionsbaustein ItpResetFastMFunc [▶ 290] möglich. Aus Gründen der Übersichtlichkeit, sollte ein Mischbetrieb zwischen Zurücksetzen aus der SPS und NC vermieden werden.

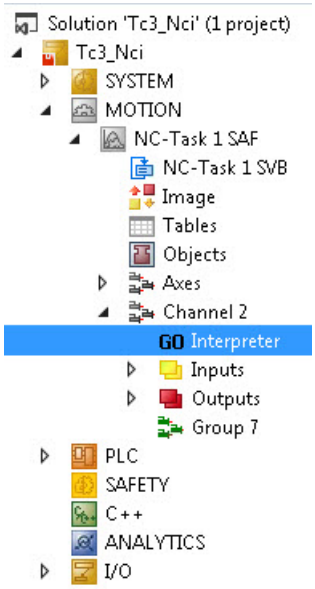
Löschen aller anstehenden M-Funktionen

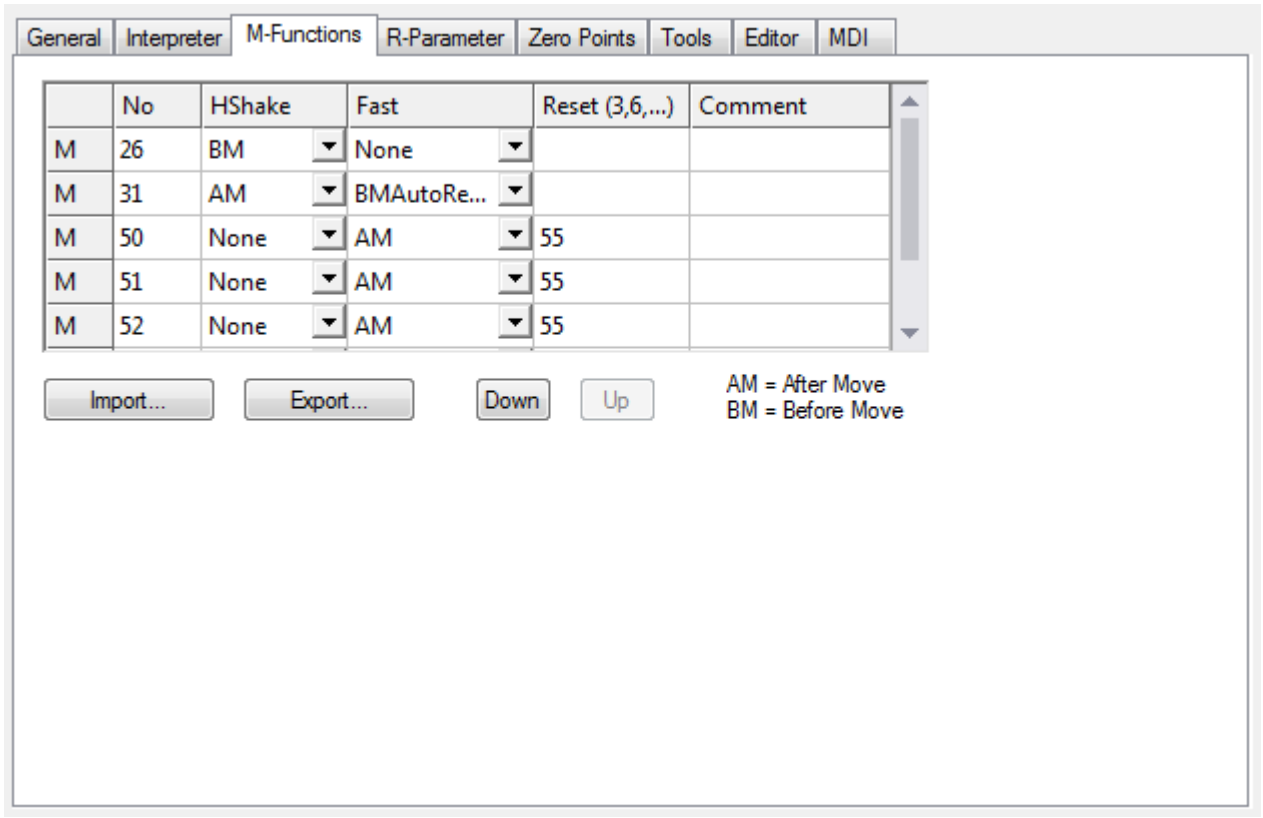
Mit einem Kanalstopp und einem Kanalreset werden alle anstehenden M-Funktionen zurückgesetzt. Das gilt für die M-Funktionen vom Typ 'Handshake' und auch für die schnellen Signal Bits. Wird das NC-Programm ordnungsgemäß mit M30 beendet sind ebenfalls alle M-Funktionen abgelöscht.

5.3.1.3 Parametrierung von M-Funktionen

Die Parametrierung der M-Funktionen erfolgt im TwinCAT XAE. Dabei wird für jeden Interpolationskanal eine eigene M-Funktionstabelle beschrieben.

Um eine Konfiguration von M-Funktionen wirksam zu schalten, ist ein Aktivieren der TwinCAT-Konfiguration erforderlich.





No

Nummer der zu parametrierenden M-Funktion. Der Wert muss zwischen 0 und 159 liegen

HShake

Ist hier ein Wert ungleich 'None' eingetragen, so ist die M-Funktion vom Type 'Handshake'

- *None*: es wird kein Handshake ausgeführt
- *BM* (Before Move)**:** Falls im gleichen Satz eine Bewegung programmiert ist, erfolgt der Handshake **vor** der Bewegung
- *AM* (After Move): Falls im gleichen Satz eine Bewegung programmiert ist, erfolgt der Handshake **nach** der Bewegung

Fast

Wird hier ein Wert ungleich 'None' eingetragen, so wird eine M-Funktion vom Type 'Schnelle-Signal-Bit' ausgeführt

- *None*: es wird keine schnelle M-Funktion ausgeführt
- *BM* (Before Move): Falls im gleichen Satz eine Bewegung programmiert ist, erfolgt die Ausgabe **vor** der Bewegung.
- *AM* (After Move): Falls im gleichen Satz eine Bewegung programmiert ist, erfolgt die Ausgabe **nach** der Bewegung.
- *BMAutoReset* (Before Move & automatisches Zurücksetzen): Falls im gleichen Satz eine Bewegung programmiert ist, erfolgt die Ausgabe **vor** der Bewegung. Zusätzlich wird die M-Funktion am Ende des Satzes automatisch wieder gelöscht. D.h. die M-Funktion ist nur satzweise wirksam. Um sicherzustellen, dass die PLC die M-Funktion auf jeden Fall erkennt, muss der der damit programmierte Verfahrsatz zeitlich lang genug sein (min. 2 PLC-Zyklen) oder zusätzlich eine M-Funktion mit Handshake programmiert werden.
- *AMAutoReset* (After Move & automatisches Zurücksetzen): Diese Parametrierung ist nur dann sinnvoll, wenn entweder eine M-Funktion vom Typ Handshake mit programmiert (bzw. parametriert) wird, oder die M-Funktion lediglich zum Zurücksetzen von anderen M-Funktionen genutzt wird. Ohne einen zusätzlichen Handshake, wird diese M-Funktion in der Regel nicht von der PLC erkannt.
- alle anderen Kombinationen sind aus Kompatibilitätsgründen anwählbar.

Reset

Hier können bis zu 10 M-Funktionen eingetragen werden, die mit dem Aufruf gelöscht werden.

Hinweis Für den Fall, dass kein Reset-Signal-Bit gesetzt wird, werden die abzulöschenden Bits unmittelbar vor dem Setzen der neuen Signal-Bits zurückgesetzt.

Import/Export

Die M-Funktionen werden für jeden Kanal einzeln parametriert. Mit der Im- und Exportfunktion kann die Parametrierung auf andere Kanäle übertragen werden.

5.3.1.4 Kombination von M-Funktionen

- Es darf pro Zeile nur **eine** M-Funktion vom Typ 'Handshake' programmiert werden!
- Es dürfen bis zu 10 M-Funktionen vom Typ 'Signal-Bit' in einer Zeile programmiert werden
- Eine Kombination von Punkt 1 & 2 ist erlaubt

Beispiel:

```
N10 G01 X1000 F60000
N20 M10 M11 M12 X2000 (M10 & M11 are signal bits)
(M12 is of type handshake)
M30
```

Beispiele für sinnvolle und praktisch einsetzbare Regelkombinationen:

- Eine M-Funktion soll für die Dauer einer Bewegung anstehen und dann automatisch gelöscht werden. Dazu ist in der Spalte HShake 'None' und in der Spalte Fast 'BMAutoReset' anzuwählen. Mit dem erzeugten Signal-Bit kann z.B. ein Beleimungsventil gesteuert werden.
- Eine M-Funktion startet einen Bohrantrieb und die folgenden Bewegungen dürfen erst nach einer Anlaufzeit und nur bei Betriebsbereitschaft ausgelöst werden. Hier muss in der Spalte HShake 'BM' ausgewählt werden. Die SPS quittiert die Anforderung erst nach einer Verzögerungszeit und nur, wenn der Frequenzumrichter betriebsbereit ist.
- Ein Bohrantrieb wird mit einer M-Funktion gestartet. Um nicht auf die Hochlaufzeit des Antriebs warten zu müssen, soll die M-Funktion bereits im Satz vor der Bohrbewegung programmiert werden. In der nächsten Bewegung (der eigentlichen Bohrbewegung) muss trotzdem sichergestellt sein, dass der Antrieb seine Drehzahl erreicht hat. Für diese Variante müssen entweder zwei verschiedene M-Funktionen verwendet werden (Vorlaufsignal als Signal-Bit, Sicherheitsabfrage als Handshake) oder aber es wird eine M-Funktion Fast 'BMAutoReset' und HShake 'AM' verwendet.

5.3.1.5 Verhalten im Fehlerfall

Tritt während der Abarbeitung eines NC-Programms ein Laufzeitfehler (z.B. Schleppabstandsüberwachung schlägt zu) auf, so wird das NC-Programm abgebrochen. Für diesen Fall bleiben die M-Funktionen, falls sie gesetzt sind, anliegen. D.h. das PLC-Programm hat für diesen Fall dafür Sorge zu tragen, dass ggf. M-Funktionen nicht bearbeitet werden.

5.3.2 H-, T- und S-Parameter

H-, T- und S-Parameter werden verwendet, um Parameter aus dem NC-Interpreter zur SPS zu übertragen.

Der H-Parameter steht in diesem Zusammenhang für Hilfsparameter und ist vom Type DINT (32 bit signed).

Die T- und S-Parameter sind vom Typ WORD und stehen für Tool (Werkzeug) und Spindel.

Beispiel:

```
H=4711
R1=23
S=R1
T4711
```

Hinweis Bei dem T-Parameter kann kein R-Parameter zugewiesen werden. Des Weiteren erfolgt die Zuweisung ohne Zuweisungsoperator ('=').

T- und S-Parameter wirken jeweils am Satzanfang, H-Parameter wirken am Ende des programmierten Satzes.

5.3.3 Dekodierstopp

Code	Funktion
@714 [▶ 172]	Dekodierstopp
@716 [▶ 173]	Dekodierstopp mit Rescan der Achspositionen
@717 [▶ 173]	Dekodierstopp mit Triggerevent, bedingter Dekodierstopp

5.3.3.1 Dekodierstopp (@714)

Der Interpreter bietet die Möglichkeit einen Dekodierstopp im NC-Programm auszuführen. D.h. wenn der Interpreter auf diesen Befehl läuft, wartet er solange, bis von außen ein bestimmtes Ereignis eintrifft. Erst wenn dieses Ereignis eingetroffen ist, wird mit der Bearbeitung des NC-Programms fortgefahren.

Ein Dekodierstopp kann z.B. dafür verwendet werden die Satzunterdrückung [▶ 129] aus der SPS ein- bzw. auszuschalten oder aber auch um R-Parameter [▶ 136] neu zuzuweisen.

Es existieren zwei Ereignisse um mit der Bearbeitung fortzufahren:

- Quittierung einer M-Funktion [▶ 167]
- SAF-Task ist leer

Quittierung einer M-Funktion

Die Dekodierung des NC-Programms wird solange unterbrochen, bis die M-Funktion [▶ 167], die unmittelbar vor dem Dekodierstopp programmiert ist, bestätigt wird. D.h. die M-Funktion muss vom Typ "Handshake" sein.

Beispiel 1:

```
N10...
N20 M43 (M-function with handshake)
N30 @714 (decoder stop)
N40 ...
```

SAF-Task ist leer

Der Dekodierstopp muss nicht unbedingt in Verbindung mit einer M-Funktion programmiert werden. Wenn die SAF-Task leer läuft, d.h. es existieren keine Fahraufträge mehr, wird ebenfalls ein Event an den Interpreter geschickt. Dieses Event bewirkt, dass der Interpreter wieder anläuft.

Hinweis Der Dekodierstopp darf nicht bei aktiver Werkzeugkorrektur oder Kreisverrundung programmiert werden, da diese dann nicht mehr funktionieren.

5.3.3.2 Dekodierstopp mit Rescan der Achspositionen (@716)

Neben dem gewöhnlichen Dekodierstopp (siehe Dekodierstopp (@714) [▶ 172]) existiert auch ein Dekodierstopp, bei dem die Achspositionen des Interpolationskanals neu eingelesen werden. Dieser Stopp wird dann benötigt, wenn z.B. während eines Werkzeugwechsels Achsen via PTP verfahren und anschließend die Achsen nicht auf die alte Position zurückgefahren werden. Ein weiterer Anwendungsfall besteht dann, wenn in einer M-Funktion (mit Handshake) die Achskonfiguration geändert wird.

Wird ein Dekodierstopp mit Rescan programmiert, ist es unbedingt erforderlich unmittelbar davor eine M-Funktion mit Handshake zu programmieren.

Beispiel 2:

```
N10...
N20 M43 (M function with handshake carries out a tool change, for
example)
N30 @716 (Decoder stop with rescan)
N40 ...
```

Hinweis Der Dekodierstopp darf nicht bei aktiver Werkzeugkorrektur oder Kreisverrundung programmiert werden, da diese dann nicht mehr funktionieren.

5.3.3.3 Dekodierstopp mit externem Triggerevent (@717)

In manchen Fällen kann es z.B. von Ereignissen in der SPS abhängen, ob das NC-Teileprogramm warten muss oder fortgesetzt werden kann. Mit den 2 Typen von M-Funktionen [▶ 167] gibt es dabei folgende Probleme:

- Handshake: Aufgrund des Handshakes der M-Funktion muss die Bahngeschwindigkeit an der Stelle, wo die M-Funktion programmiert wurde immer auf 0 gezogen werden und anschließend wird auf die Bestätigung aus der SPS gewartet.
- On The Fly (oder auch schnelle M-Funktion genannt): Da keine Bestätigung von der SPS erwartet wird, gibt es auch keine Möglichkeit, dass das Teileprogramm auf die SPS wartet.
- Auch eine Kombination der beiden M-Funktionstypen hilft hier erst mal nicht weiter.

Beispiel:

Aus dem NC-Teileprogramm wird während einer Positionierung mit einer fliegenden M-Funktion ein Prozess A angestoßen. Dabei wird vorausgesetzt, dass der Verfahrsatz im NC-Programm für gewöhnlich lang genug ist, so dass der Prozess A in der SPS fertig gestellt werden kann. Ist A fertig, so soll das NC-Teileprogramm mit Lookahead das nächste Segment abfahren. Für den Fall, dass A allerdings nicht fertig geworden ist, soll die NC am Segmentende anhalten und warten, bis der Prozess A beendet ist. Genau dieses Szenario kann mit dem Befehl @717 realisiert werden. Dabei schickt die SPS das sogenannte 'GoAhead [▶ 230]'-Kommando, wenn Prozess A beendet ist.

```
N10 ...
N20 G0 X0 Y0 Z0
N30 G01 X500 F6000
N40 M70 (flying M-function that triggers process A)
N50 G01 X700
N60 @717 (decoder stop with external trigger event)
N70 G01 X1000
N80 ...
```

Trifft das Go-Ahead Signal früh genug aus der SPS ein, so werden die Sätze N50 & N70 vom Lookahead verbunden und die Bahngeschwindigkeit wird dann nicht reduziert. Falls das Signal in der Verzögerungsphase von N50 eintrifft, so wird die Geschwindigkeit dann wieder aufgebaut. Ansonsten wird gewartet, bis das Signal aus der SPS eintrifft.

Hinweis Der Dekodierstopp darf nicht bei aktiver Werkzeugkorrektur oder Kreisverrundung programmiert werden, da diese dann nicht mehr funktionieren.

Der Funktionsbaustein 'ItpGoAheadEx' liefert den Fehlercode 0x410A, wenn zum Zeitpunkt des Aufrufs kein @717 im Interpreter anliegt.

5.3.4 Sprünge

Code	Funktion
@100 [▶ 174]	Unbedingter Sprung
@121 [▶ 174]	Springe wenn ungleich
@122 [▶ 174]	Springe wenn gleich
@123 [▶ 174]	Springe wenn kleiner gleich
@124 [▶ 175]	Springe wenn kleiner
@125 [▶ 175]	Springe wenn größer gleich
@126 [▶ 175]	Springe wenn größer
@111 [▶ 175]	Case-Anweisung

Unbedingter Sprung

Befehl	@100
Parameter	K oder R

Der Parameter beschreibt das Sprungziel. Dieses muss mit einer Richtungsangabe ('+' oder '-') versehen sein.

Beispiel 1:

```
N10 ..
...
N120 @100 K-10
```

In diesem Beispiel wird nach dem interpretieren von Zeile 110 wieder mit Zeile 10 fortgefahren. Dabei gibt das Vorzeichen die Richtung an, in der die zu suchende Zeile zu finden ist.

Springe wenn ungleich

Befehl	@121	
Parameter 1	R<n>	Wert mit dem verglichen wird
Parameter 2	K oder R<m>	Vergleichswert
Parameter 3	K	Sprungziel mit Richtungsangabe

Beispiel 2:

```
N10 ..
...
R1=14
N120 @121 R1 K9 K-10
N130 ...
```

Springe wenn gleich

vergl. Springe wenn ungleich [▶ 174]

Springe wenn kleiner gleich

vergl. Springe wenn ungleich [▶ 174]

Springe wenn kleiner

vergl. Springe wenn ungleich [▶ 174]

Springe wenn größer gleich

vergl. Springe wenn ungleich [▶ 174]

Springe wenn größer

vergl. Springe wenn ungleich [▶ 174]

Case-Anweisung

Befehl	@111	
Parameter 1	R<n>	Wert mit dem verglichen wird
Parameter 2	K oder R<m>	1. Vergleichswert
Parameter 3	K	1. Sprungziel
Parameter 4	K oder R<m>	2. Vergleichswert
...		

Beispiel 3:

```
N100 R2=12 (R2=13) (R2=14)
N200 @111 R2 K12 K300
K13 K400
K14 K500

N300 R0=300
N310 @100 K5000

N400 R0=400
N410 @100 K5000

N500 R0=500
N510 @100 K5000

N5000 M30
```

In der Zeile 200 wird eine Case-Anweisung aufgerufen. Für den Fall, dass R2 = 12 ist, wird zu der Zeile 300 gesprungen.

Ist R2 = 13, wird zu der Zeile 400 gesprungen. Wenn R2 = 14 ist, wird folglich zu der Zeile 500 gesprungen.

Für den Fall, dass keine der Bedingungen erfüllt ist, wird einfach mit der nächsten Zeile (hier 300) fortgefahren.

5.3.5 Schleifen

Im Folgenden werden die unterschiedlichen Schleifentypen beschrieben.

Code	Schleifentyp	Abbruchbedingung
@131	While-Schleife [▶ 176]	Schleife solange gleich
@132	While-Schleife [▶ 176]	Schleife solange ungleich
@133	While-Schleife [▶ 176]	Schleife solange grösser
@134	While-Schleife [▶ 176]	Schleife solange grösser oder gleich
@135	While-Schleife [▶ 176]	Schleife solange kleiner
@136	While-Schleife [▶ 176]	Schleife solange kleiner oder gleich
@141	Repeat-Schleife [▶ 176]	Wiederhole bis gleich
@142	Repeat-Schleife [▶ 176]	Wiederhole bis ungleich

Code	Schleifentyp	Abbruchbedingung
@143	Repeat-Schleife [▶ 176]	Wiederhole bis grösser
@144	Repeat-Schleife [▶ 176]	Wiederhole bis grösser oder gleich
@145	Repeat-Schleife [▶ 176]	Wiederhole bis kleiner
@146	Repeat-Schleife [▶ 176]	Wiederhole bis kleiner oder gleich
@151	For-To-Schleife [▶ 177]	
@161	For-DownTo-Schleife [▶ 177]	

Eine Verschachtelung der Schleifen ist grundsätzlich möglich.

While-Schleife

Befehl	@13<n>	mit 1 <= n <= 6
Parameter 1	R<m>	Wert mit dem verglichen wird
Parameter 2	K oder R<k>	Vergleichswert
Parameter 3	K	Sprungziel für den Fall, dass die Bedingung nicht erfüllt ist

Die While-Schleife wird solange ausgeführt, wie die Bedingung erfüllt ist. Dabei findet die Überprüfung am Schleifenanfang statt. Wenn die Bedingung nicht oder nicht mehr erfüllt ist, wird zu der angegebenen Zeile gesprungen (Parameter 3).

Am Ende der While-Schleife muss ein unbedingter Sprung (@100 [▶ 174]) programmiert werden. Dabei ist als Sprungziel die Zeilennummer der While-Schleife anzugeben.

Mit <n> wird die Abbruchbedingung der Schleife spezifiziert.

Beispiel 1:

```
N100 R6=4
N200 @131 R6 K4 K600 (K600 is the target of the jump, when the condition is no longer satisfied)
N210 ...
N220 @100 K-200

N600 ...

N5000 M30
```

Solange R6 = 4 ist, wird die Schleife (Zeile 200 bis 220) wiederholt. Wenn die Bedingung nicht mehr erfüllt ist, wird zu der Zeile 600 gesprungen.

Repeat-Schleife

Befehl	@14<n>	mit 1 <= n <= 6
Parameter 1	R<m>	Wert mit dem verglichen wird
Parameter 2	K oder R<k>	Vergleichswert
Parameter 3	K	Sprungziel zum Schleifenanfang

Bei der Repeat-Schleife erfolgt die Abfrage am Ende der Schleife. D.h. die Schleife wird mindestens einmal durchlaufen. Erst wenn die Bedingung erfüllt ist, wird die Schleife beendet und mit dem Programm fortgefahren.

Beispiel 2:

```
N200 ...
N210 ...

N300 @141 R6 K25 K200
```

Die Schleife wird solange wiederholt, bis R6 = 25 ist. Die zweite Konstante in Zeile 300 gibt das Sprungziel (Schleifenanfang) an.

For-To-Schleife

Befehl	@151 <Variable> <Wert> <Konstante>

Die For-To-Schleife ist eine Zählschleife, die solange ausgeführt wird, bis *Variable* gleich *Wert* ist. Dabei findet die Überprüfung am Anfang der Schleife statt. Falls die Bedingung erfüllt ist, wird zu der unter *Konstante* angegebenen Zeile gesprungen.

Am Ende der Schleife muss die Variable inkrementiert (@620) und mit einem unbedingten Sprung an den Schleifenanfang gesprungen werden.

Beispiel 3:

```
N190 R6=0
N200 @151 R6 K20 K400
N210 ...
N290 @620 R6 (increment R6)
N300 @100 K-200
```

For-Downto-Schleife

Befehl	@161 <Variable> <Wert> <Konstante>

Die For-Downto-Schleife ist eine Zählschleife. Die Verhaltensweise ist vergleichbar mit der For-To-Schleife. Es besteht lediglich der Unterschied, dass die Variable am Schleifenende um 1 dekrementiert (@621) wird.

5.3.6 Unterprogrammtechnik

Auch in der NC-Programmierung ist es sinnvoll, häufig benutzte Befehlsfolgen als Unterprogramm zu gestalten. Auf diese Weise ist es möglich, aus verschiedenen Werkstückprogrammen auf eine vorgefertigte und ausgetestete Funktion zurückzugreifen.

Innerhalb eines Programms werden Unterprogramme über eine Nummer identifiziert. Diese muss eindeutig sein: Es kann nur ein Unterprogramm mit einer bestimmten Nummer (1..>2.000.000.000) geben.

Bei der Interpretation wird das aufrufende Programm unterbrochen. Der Text des Unterprogramms wird (bei Bedarf mehrmals) abgearbeitet. Anschließend wird die Abarbeitung im aufrufenden Programm hinter der Aufrufstelle fortgesetzt.

Selbstverständlich kann auch aus einem Unterprogramm ein anderes Unterprogramm aufgerufen werden. In diesem Fall wird sinngemäß verfahren. Dabei bildet sich ein Stapel von Rücksprunginformationen. Zurzeit ist diese Unterprogramm-Schachtelung aus technischen Gründen auf 20 Ebenen begrenzt.

Definition eines Unterprogramms

Der Code eines Unterprogramms kann in derselben Datei geschrieben werden, in der sich auch das aufrufende Programm befindet. In diesem Fall wird das Unterprogramm direkt gebunden: Es wird beim Laden der Datei automatisch mitgeladen. Soll es allgemein verfügbar sein, muss es in eine eigene Datei geschrieben werden, die sich im CNC-Verzeichnis befinden muss.

Der Name der Datei beginnt mit dem Buchstaben 'L' und es folgt eine Ziffernkette. Diese Ziffernkette muss die Unterprogramm-Nummer ohne führende '0' wiedergeben.

Der Startpunkt des Unterprogramms ist im Code mit einem Unterprogramm-Label zu markieren. Dieses besteht wie der Dateiname aus dem Buchstaben 'L' und der beschriebenen Ziffernfolge.

Unmittelbar hinter diesem Label setzt der Interpreter auf.

Syntax Unterprogramm:

```
(Datei L2000.NC)
L2000
N100...
N110...
...
N5000 M17 (return command)
```

Aufruf eines Unterprogramms

Um in einem beliebigen Satz des NC-Programms ein Unterprogramm aufzurufen, ist folgende Syntax zu benutzen. Wichtig ist, dass der Ausdruck "L2000" nicht am Zeilenanfang steht, um eine Verwechslung mit einem Unterprogramm-Label zu vermeiden.

```
(syntax of the subroutine call)
N100 L2000
```

Im folgenden Beispiel wird mit dem Ausdruck "P5" eine 5-fache Wiederholung des Unterprogramms veranlasst.

```
(n-fold subroutine call (here: 5- fold))
N100 L2000 P5
```

Dynamischer Unterprogrammaufruf

Manchmal steht erst zur Laufzeit fest, welches Unterprogramm aufgerufen werden soll. Um sich für diesen Fall die CASE-Anweisung zu sparen, ist es auch möglich, das Unterprogramm mit einem R-Parameter aufzurufen. Dabei muss der Wert für R aber in einer eigenen Zeile zugewiesen bzw. berechnet werden.

```
(Dynamic call of a subroutine)
N099 R47=R45+1
N100 L=R47
```

Parameterübergabe

Die Übergabe von Parametern an Unterprogramme ist mit Hilfe der R-Parameter [▶ 136] zu verwirklichen. Dabei ist zu beachten, dass R-Parameter nicht automatisch gesichert werden (vergl. Retten von R-Parametern [▶ 136]) .

Parameterverwendung

In einem NC-Unterprogramm können R-Parameter im Prinzip frei verwendet werden. Das hat eine Reihe von Konsequenzen, die zu Fehlern führen können, wenn sie nicht beachtet werden. Auf der anderen Seite ist ein gezielter Einsatz möglich, der dem NC-Programmierer eine Reihe von Arbeitstechniken zur Verfügung stellt.

Ergebnisse von Unterprogrammen

Wird ein R-Parameter verändert, ohne dass sein Inhalt gerettet und restauriert wurde, ist die Veränderung nach einem Unterprogrammrücksprung wirksam. Wenn dies unbeabsichtigt war, kann es in der Folge zu einem nicht geplanten Verhalten der Maschine kommen.

Dies kann jedoch auch bewusst dazu ausgenutzt werden, den weiteren Verlauf der Bearbeitung von den Ergebnissen eines Unterprogramms abhängig zu machen. Dabei ist keinerlei Einschränkung außer denen der R-Parameter zu beachten.

Beispiel:

```
N100 L2000
N110 R2=R3+R4
...
N999 M30
```

```
L2000
N10 R3=17.5
N20 R4=1
N99 M17
```

Hier werden in einem Unterprogramm Werte festgelegt. Die Werte werden im aufrufenden Programm weiter verwendet.

Beenden eines Unterprogramms

Ein Unterprogramm wird mit M17 beendet.

5.3.7 Dynamischer Override

Befehl	DynOvr=<value> oder DynOvr = R<n>
Aufhebung	DynOvr=1

Beispiel:

```
N10 G01 X100 Y200 F6000
N20 DynOvr=0.4
N30 G01 X500
```

Mit 'DynOvr' können zur Laufzeit des NC Programms die Dynamikparameter der Achsen in der Gruppe prozentual verändert werden. Daraus ergeben sich ebenfalls neue Werte für die Bahndynamik. Die neuen Dynamikwerte werden ohne Stopp mit der Ausführung der Zeile gültig. Für das oben dargestellte Beispiel bedeutet das, dass in Satz 10 noch mit den alten Werten verzögert und in Satz 20 mit den neuen Werten beschleunigt wird.

Definitionsbereich

0 < DynOvr ≤ 1

Vergl. auch Änderung der Bahndynamik [▶ 179].

5.3.8 Änderung der Bahndynamik

Befehl	#set paramPathDynamics
Parameter <acc>	Wert für die maximal erlaubte Bahnbeschleunigung in mm/s^2
Parameter <dec>	Wert für die maximal erlaubte Verzögerung in mm/s^2
Parameter <jerk>	Wert für den maximal erlaubten Ruck in mm/s^3.

Beispiel:

```
N10 G01 X100 Y200 F6000
N15 R4=3000
N20 #set paramPathDynamics( 700; 700; R4 )#
N30 G01 X500
```

Mit 'paramPathDynamics' kann zur Laufzeit des NC Programms die Bahndynamik verändert werden. Die neuen Dynamikwerte werden mit der programmierten Zeile wirksam. Für das oben dargestellte Beispiel bedeutet das, dass Satz 10 auch am Satzende mit den Default-Werten behandelt wird. Für Satz 30 werden am Segmentanfang die neuen Parameter verwendet.

Dieses Kommando limitiert alle Bahnachsen auf die parametrierten Dynamikwerte. Die Bahn selber kann aber in Abhängigkeit ihrer Orientierung eine höhere Dynamik aufweisen. Die Dynamik von Hilfsachsen wird nicht verändert.

Vergl. auch Dynamischer Override [▶ 179].

Hinweis **Die aus dem NC Programm geänderten Dynamikwerte bleiben bis zum nächsten Reset des Interpreters bzw. Restart von TwinCAT aktiv.**

Hinweis **Das alte Kommando 'paramGroupDynamics' bleibt weiterhin wirksam. Es ist dabei allerdings nicht möglich, R-Parameter zu übergeben.**

Befehl	#set paramGroupDynamics(<grp>,<acc>,<dec>,<jerk>)#
Parameter <grp>	Gruppe für die die Dynamikänderung wirksam sein soll. Z. Zt. immer 1.
Parameter <acc>	Wert für die maximal erlaubte Bahnbeschleunigung in mm/s^2

Parameter <dec>	Wert für die maximal erlaubte Verzögerung in mm/s^2
Parameter <jerk>	Wert für den maximal erlaubten Ruck in mm/s^3.

Beispiel:

```
N10 G01 X100 Y200 F6000
N20 #set paramGroupDynamics( 1, 700, 700, 3000 )#
N30 G01 X500
```

Änderung der Achsdynamik

Befehl	#set paramAxisDynamics
Parameter <axis>	Achse in der Interpolationsgruppe: X: 0 Y: 1 Z: 2 Q1: 3 ... Q5: 7
Parameter <acc>	Wert für die maximal erlaubte Beschleunigung in mm/s^2
Parameter <dec>	Wert für die maximal erlaubte Verzögerung in mm/s^2
Parameter <jerk>	Wert für den maximal erlaubten Ruck in mm/s^3.

Beispiel:

```
N10 G01 X100 Y200 F6000
N15 R4=30000
N20 #set paramAxisDynamics( 0; 1500; 1400; R4 )#
N30 G01 X500
```

Mit 'paramAxisDynamics' kann zur Laufzeit die Dynamik einer Achse geändert werden. Grundsätzlich ist die Verhaltensweise die gleiche wie bei 'paramPathDynamics', allerdings mit dem Unterschied, dass hiermit für jede Achse individuell die Dynamik vorgegeben werden kann.

5.3.9 Änderung der Reduktionsparameter

C0 Reduktion [▶ 180]
C1 Reduktion [▶ 181]
C2 Reduktion [▶ 182]

C0 Reduktion

Bei manchen Maschinentypen ist es nicht unbedingt erforderlich, an Knicken die Bahngeschwindigkeit bis auf 0 zu reduzieren. Dazu werden 2 Reduktionsmethoden angeboten

- VeloJump
- DeviationAngle

VeloJump

Befehl	#set paramVeloJump(<C0X>; <C0Y>; <C0Z>)#
Parameter <C0X>	Reduktionsfaktor für *C0*-Übergänge: *X*-Achse: *C0X* ≥ *0.0*
Parameter <C0Y>	Reduktionsfaktor für *C0*-Übergänge: *Y*-Achse: *C0Y* ≥ *0.0*
Parameter <C0Z>	Reduktionsfaktor für *C0*-Übergänge: *Z*-Achse: *C0Z* ≥ *0.0*

Mit 'paramVeloJump' können zur Laufzeit des NC-Programms die Faktoren für die Geschwindigkeitssprünge verändert werden. Die neuen Werte werden aus der Satzausführung in der programmierten Zeile wirksam. Näheres zu der Wirkungsweise ist im Anhang unter Parametrierung [▶ 330] zu finden.

Beispiel:

```
N10 G01 X100 Y200 F6000
N20 R2=4.5
N30 #set paramVeloJump( 1.45; R2; R2 )#
N40 G01 X500
```

Zurücksetzen der Parameter

Die aus dem NC Programm geänderten VeloJump Parameter bleiben bis zum nächsten Reset des Interpreters bzw. Restart von TwinCAT aktiv.

DeviationAngle (noch nicht freigegeben)

Befehl	#set paramDevAngle(<C0Factor>; <AngleLow>; <AngleHeigh>)#
Parameter <C0Factor>	Bahn-Reduktionsfaktor für *C0*-Übergänge: 1.0 ≥ *C0* ≥ *0.0*
Parameter <AngleLow>	Winkel in Grad ab dem die Reduktion einsetzt: $0 \leq \varphi\_l < \varphi\_h \leq \pi$
Parameter <AngleHeigh>	Winkel in Grad ab dem auf *v_link = 0.0* reduziert wird: $0 \leq \varphi\_l < \varphi\_h \leq \pi$

Mit 'paramDevAngle' werden die Parameter für die C0 Reduktion beschrieben. Im Gegensatz zu der Reduktionsmethode VeloJump, wo der Geschwindigkeitssprung direkt beeinflusst wird, ergibt sich bei der Methode DeviationAngle, der Geschwindigkeitssprung in Abhängigkeit vom Winkel. Näheres zu der Wirkungsweise ist im Anhang unter Parametrierung [▶ 330] zu finden.

Beispiel:

```
N10 G01 X100 Y200 F6000
N20 #set paramDevAngle(0.15; 5; 160 )#
N30 G01 X500
```

Zurücksetzen der Parameter

Die aus dem NC-Programm geänderten DeviationAngle Parameter bleiben bis zum nächsten Reset des Interpreters bzw. Restart von TwinCAT aktiv.

C1 Reduktionsfaktor

Befehl	#set paramC1ReductionFactor(<C1Factor>)#
Parameter <C1Factor>	C1 Reduktionsfaktor

Mit 'paramC1ReductionFactor' kann zur Laufzeit des NC Programms der C1 Reduktionsfaktor verändert werden.

Der neue Parameter ist in dem Segmentübergang, in dem der Reduktionsfaktor programmiert wird, gültig. D.h. in dem angefügten Beispiel ist der neue Wert für die C1 Reduktion im Segmentübergang von N10 zu N30 bereits gültig.

Als Parameter kann ein Floating-Point-Wert oder ein R-Parameter übergeben werden.

Näheres zu der Wirkungsweise ist im Anhang unter Parametrierung [▶ 330] zu finden.

Beispiel:

```
N10 G01 X100 Y200 F6000
N20 #set paramC1ReductionFactor( 0.45 )#
N30 G01 X500
```

Zurücksetzen der Parameter

Der aus dem NC-Programm geänderte C1 Reduktionsfaktor bleibt bis zum nächsten Reset des Interpreters bzw. Restart von TwinCAT aktiv.

C2 Reduktionsfaktor

Befehl	#set paramC2ReductionFactor(<C2Factor>)#
Parameter <C2Factor>	C2 Reduktionsfaktor

Mit 'paramC2ReductionFactor' kann zur Laufzeit des NC-Programms der C2 Reduktionsfaktor verändert werden.

Der Befehl wirkt in dem Segmentübergang, in dem der Reduktionsfaktor programmiert wird. D.h. in dem angefügten Beispiel ist der neue Wert für die C2 Reduktion im Segmentübergang von N10 zu N30 bereits gültig.

Als Parameter kann ein Floating-Point-Wert oder ein R-Parameter übergeben werden.

Beispiel:

```
N10 G01 X100 Y200 F6000
N20 #set paramC2ReductionFactor( 1.45 )#
N30 G01 X500
```

Zurücksetzen der Parameter

Der aus dem NC-Programm geänderte C2 Reduktionsfaktor bleibt bis zum nächsten Reset des Interpreters bzw. Restart von TwinCAT aktiv.

5.3.10 Änderung der Mindestgeschwindigkeit

Befehl	#set paramVeloMin(<VeloMin>)#
Parameter <VeloMin>	Minimale Bahngeschwindigkeit

Mit 'paramVeloMin' kann zur Laufzeit des NC-Programms die Mindestgeschwindigkeit an der Bahn verändert werden. Die neue Geschwindigkeit wird aus der Satzausführung in der programmierten Zeile wirksam.

Als Parameter kann ein Floating-Point-Wert oder ein R-Parameter übergeben werden.

Beispiel:

```
N10 G01 X100 Y200 F6000
N20 #set paramVeloMin( 2.45 )#
N30 G01 X500
```

Zurücksetzen der Parameter

Die aus dem NC-Programm geänderte Mindestgeschwindigkeit bleibt bis zum nächsten Reset des Interpreters bzw. Restart von TwinCAT aktiv.

Programmierung der Geschwindigkeit

Die Einheit der Geschwindigkeit ist mm/sec und damit äquivalent mit den üblichen Einheiten im XAE.

5.3.11 Lese Achsen-Istwert

Befehl	@361	
Parameter 1	R<n>	R Parameter, dem der Achsen-Istwert zugewiesen wird
Parameter 2	K<m>	Konstante für die zu lesende Achskoordinate 0: X-Achse 1: Y-Achse 2: Z-Achse 3: Q1-Achse 4: Q2-Achse . . . 7: Q5-Achse

Beispiel 1:

```
N10 G0 X0 Y0 Z0 F24000
N30 G01 X1000
N40 @361 R1 K0 (read position of x axis)
N50 R0=X
N60 G01 X=R0+R1
N70 M30
```

Mit dem Befehl @361 wird implizit ein Dekodierstopp ausgeführt. Damit wird sichergestellt, dass in diesem Beispiel die Position gelesen wird, wenn Satz N30 abgearbeitet ist.

Eine mögliche Anwendung ist z.B. die Kombination mit dem Restweglöschen.

Lese Achsen-Istwert ohne Dekodierstopp

Befehl	#get PathAxesPos(R<a>; R<b>; R<c>)#	
Parameter 1	R<a>	R Parameter, dem der Achsen-Istwert der X-Achse zugewiesen wird
Parameter 2	R<b>	R Parameter, dem der Achsen-Istwert der Y-Achse zugewiesen wird
Parameter 3	R<c>	R Parameter, dem der Achsen-Istwert der Z-Achse zugewiesen wird

Der Befehl #get PathAxesPos( )# liest die aktuellen Ist-Positionen der Bahnachsen (X, Y &Z) aus. Er verhält sich ähnlich wie @361 mit dem Unterschied, dass mit diesem Kommando kein impliziter Dekodierstopp ausgelöst wird. D.h. der Programmierer muss selber sicherstellen, dass sich beim Abarbeiten des Kommandos im Interpreter die Achsen noch nicht bewegt haben oder im Satz vor dem Befehl muss ein Dekodierstopp (@714) programmiert werden.

#get PathAxesPos( )# ist eine Alternative zu @361, die aber an bestimmte Bedingungen geknüpft ist.

Beispiel 2:

```
@714(optional)
N27 #get PathAxesPos( R0; R1; R20 )#
```

Hinweis Wenn eine Bahnachse nicht zugewiesen ist (z.B. Z hat keine zugewiesene Achse), wird dem dazugehörigen R-Parameter der Wert 0 übergeben.

5.3.12 Überspringe virtuelle Bewegung

Befehl	#skip VirtualMovements(<parameter>)#
Parameter	0 (default): virtuelle Bewegungen werden „ausgefahren“ 1: virtuelle Bewegungen werden übersprungen

Bewegungen von nicht vorhandenen aber dennoch programmierten Hauptachsen (X, Y & Z) können mit dem Befehl „skip VirtualMovements“ übersprungen werden.

Beispiel:

Die Interpolationsgruppe (CfgBuildGroup) beinhaltet nur Zuweisungen für die X- und Y-Achse. Die Z-Achse ist **nicht** zugewiesen, aber im Teileprogramm programmiert.

```
(Startposition X0 Y0 Z0)
N10 #skip VirtualMovements(1)#
N20 G01 X100 Y200 F6000
N30 G01 Z1000 (virtual movement, because z is not assigned)
N40 G01 X500
```

Die Ausführung dieses Programms überspringt das Segment N30.

5.3.13 Meldungen aus dem NC-Programm

Befehl	#MSG (<message level>; <mask>; "<text>")#
<message level>	• ITP Die Meldung wird aus dem Interpreter abgesetzt. Damit erscheint die Meldung in der Regel deutlich vor der Abarbeitung im NC-Kern. • NCK Die Meldung wird mit der Ausführung des NC-Satzes aus dem NC-Kern abgesetzt. Damit erscheint sie synchron mit der Satzausführung (SAF)
<mask>	STRING
<text>	der darzustellende Text

```
N10 G0 X0 Y0
N20 G01 X100 Y0 F6000

N30 #MSG( NCK; STRING; "this is a text")#

N40 G01 X200 Y-100
```

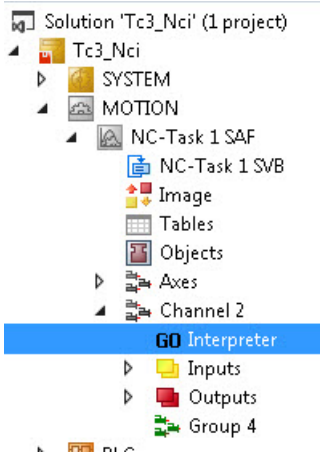
Mit dem Text können **keine** weiteren Parameter (z.B. R-Parameter) übergeben werden.

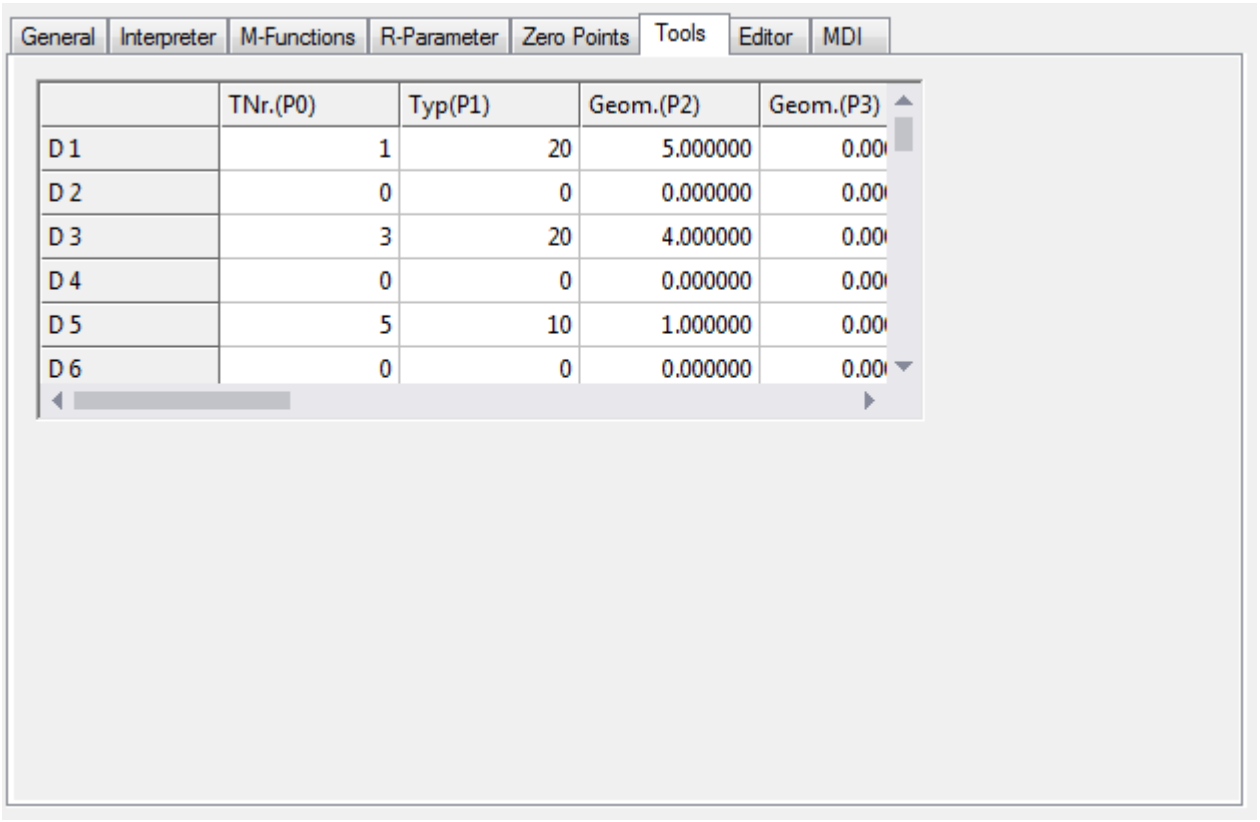
Die Meldung wird intern wie ein Hinweis gehandhabt.

5.4 Werkzeugkorrekturen

5.4.1 Werkzeugdaten

Für die Werkzeugdaten stehen in der NC 255 Speicherplätze (D1..D255) pro Kanal zur Verfügung. Die Parameter für die Werkzeugdaten können direkt im XAE beschrieben werden. Die Sicherung der Daten erfolgt als ASCII-Datei (<Kanal ID>.wz) und wird im TwinCAT\CNC Verzeichnis hinterlegt. Beim Start von TwinCAT werden diese Daten automatisch geladen.





Zurzeit werden zwei Werkzeugtypen unterstützt:

- Bohrer
- Schaftfräser

Im Folgenden werden die relevanten Spalten (Parameter) für diesen Werkzeugtyp beschrieben.

Bohrer

Parameter	Bedeutung
0	**Tool-Nummer** Mit dem Aufruf dieses D-Worts kann gleichzeitig eine Tool-Nummer angegeben werden, die hier festgelegt wird.
1	**Werkzeugtyp** Beim Bohrer handelt es sich um **Typ 10**
2	**Geometrie: Länge** Beschreibt die Länge des Bohrers
5	**Verschleiß: Länge** Beschreibt den Verschleiß des Bohrers. Dabei muss der Verschleiß negativ angegeben werden, da er zur Länge hinzuaddiert wird.
8	kartesische Werkzeugverschiebung [▶ 187] in X-Richtung
9	kartesische Werkzeugverschiebung in Y-Richtung
10	kartesische Werkzeugverschiebung in Z-Richtung

Schaftfräser

Parameter	Bedeutung
0	**Tool-Nummer** Mit dem Aufruf dieses D-Worts kann gleichzeitig eine Tool-Nummer angegeben werden, die hier festgelegt wird.

Parameter	Bedeutung
1	**Werkzeugtyp** Beim Schaftfräser handelt es sich um **Typ 20**
2	**Geometrie: Länge** Länge des Schaftfräsers
4	**Geometrie: Radius**
5	**Verschleiß: Länge**
7	**Verschleiß: Radius**
8	kartesische Werkzeugverschiebung [▶ 187] in X-Richtung
9	kartesische Werkzeugverschiebung in Y-Richtung
10	kartesische Werkzeugverschiebung in Z-Richtung

Schreiben der Werkzeugdaten

Werkzeugdaten mit dem XAE editieren

Wie bereits erwähnt ist es möglich, die Werkzeugdaten direkt aus dem XAE zu beschreiben. Editieren Sie dazu das oben abgebildete Fenster.

Werkzeugdaten mit der SPS parametrieren

Des Weiteren können Sie Werkzeugdaten mit dem Funktionsbaustein ItpWriteToolDescEx [▶ 254] aus der SPS lesen und schreiben.

Werkzeugdaten aus dem Teileprogramm schreiben

Bei manchen Applikationen ist es komfortabler, wenn man die Werkzeugdaten direkt aus dem Teileprogramm beschreiben kann.

Der zu beschreibende Werkzeugsatz darf nicht aktiv sein, wenn er beschrieben wird. D.h. wenn z.B. die Werkzeugradiuskorrektur mit dem Parametersatz D10 aktiv ist, darf dieser nicht überschrieben werden, solange D10 noch angewählt ist.

Befehl	#set ToolParam(<Zeile>; <Spalte>;<Wert>)#
Parameter <Zeile>	Beschreibt die Zeile des Werkzeugparameters (1..255) Dies entspricht der D-Nummer
Parameter <Spalte>	Spalte die beschrieben werden soll (0..15)
Parameter <Wert>	zu übertragender Parameterwert

Beispiel:

```
N10 G0 X0 Y0 Z0
N20 G01 X100 F60000
N30 R1=10 R2=4 R3=20.3
N40 #set ToolParam(10; 0; 5)# #set ToolParam(10;1;20)#
N50 #set ToolParam(R1; R2; R3)#
N60 G41 X200 Y D10
...
```

Hinweis Als Parameter dürfen keine Formeln übergeben werden. Für das Beschreiben der Werkzeugdaten ist kein Dekodierstopp erforderlich.

Werzeugdaten aus dem Teileprogramm lesen

Mit diesem Kommando können Werkzeugdaten einem R-Parameter zugewiesen werden.

Befehl	#get ToolParam(<Zeile>; <Spalte>;<R-Param>)#
Parameter <Zeile>	Beschreibt die Zeile des Werkzeugparameters (1..255), dies entspricht der D-Nummer
Parameter <Spalte>	Spalte die beschrieben werden soll (0..15)

Parameter <R-Param>	R-Parameter, in dem das Datum eingetragen wird

Beispiel:

```
N10 G0 X0 Y0 Z0
N20 G01 X100 F60000
N30 R1=10 R2=4
N40 #get ToolParam(10; 0; R5)# #getToolParam(10;1;R20)#
N50 #get ToolParam(R1; R2; R3)#
N60 G41 X200 Y D10
...
```

Hinweise:

Hinweis Als Parameter dürfen keine Formeln übergeben werden. Für das Lesen der Werkzeugdaten ist kein Dekodierstopp erforderlich.

5.4.2 An- und Abwahl der Längenkorrektur

Die Anwahl der Längenkorrektur kann nur bei wirksamen G0 [▶ 139] bzw. G1 [▶ 140] erfolgen. Dabei muss die Arbeitsebene [▶ 132] angewählt sein, auf der die Längenkorrektur senkrecht steht.

Die Zustellrichtung wird mit P festgelegt (vergl. Arbeitsebene und Zustellrichtung [▶ 132]).

Damit die Längenkorrektur herausgefahren wird, muss die betroffene Achse wenigstens genannt werden.

Beispiel:

```
N10 G17 G01 X0 Y0 Z0 F6000
N20 D1 X10 Y10 Z
N30 ...
N90 M30
```

Hinweis Mit der Anwahl der Fräserradiuskorrektur [▶ 190] wird die Längenkorrektur automatisch angewählt.Um die Längenkorrektur wieder abzuwählen, muss D0 programmiert werden. Auch hier ist es wieder erforderlich, die betroffenen Achsen zu nennen, um die neue Position anzufahren.

5.4.3 Kartesische Werkzeugverschiebung

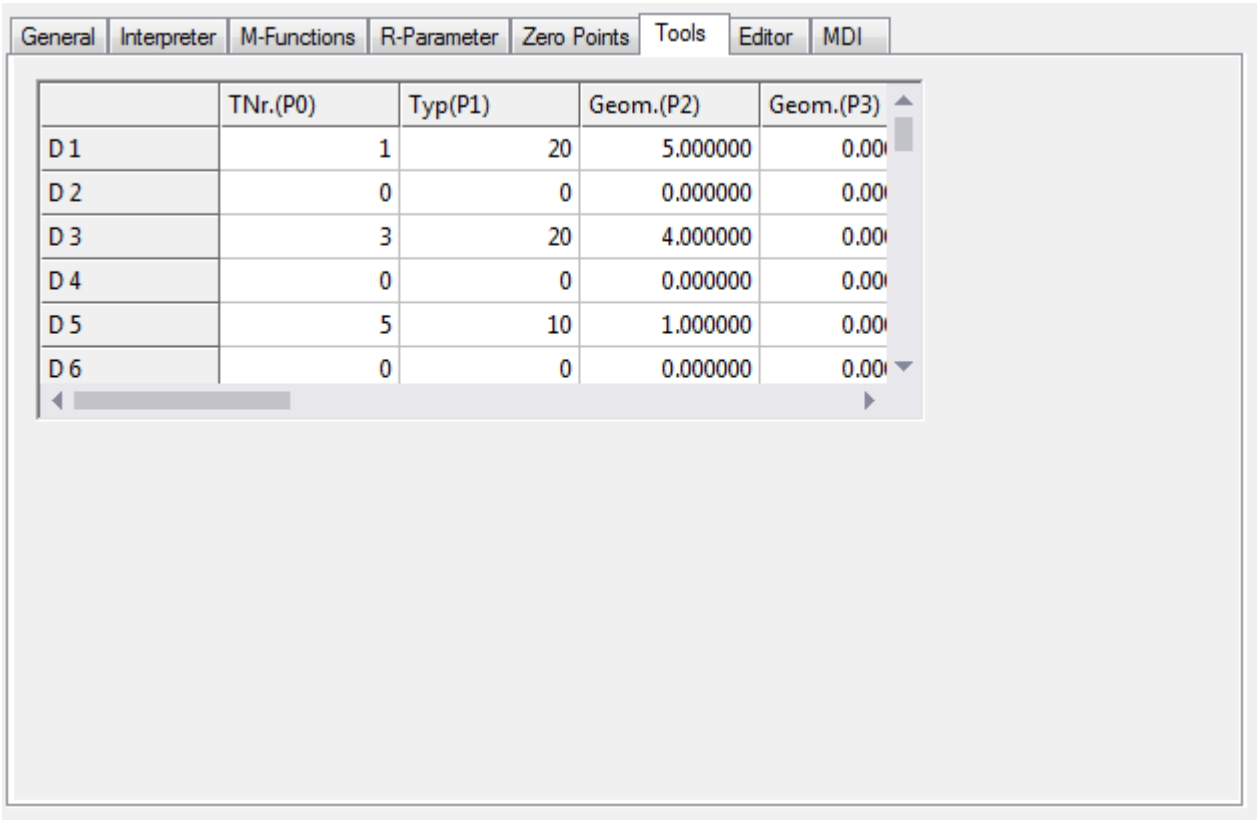
Mit der kartesischen Werkzeugverschiebung wird der Offset zwischen dem Bezugspunkt des Werkzeugträgers und dem Bezugspunkt des eigentlichen Werkzeugs beschrieben. In vielen Fällen liegen diese Bezugspunkte übereinander, sodass für die kartesische Werkzeugverschiebung 0 eingetragen werden kann.

Parameter

Die Parameter für die Verschiebung werden wie die Werkzeuglänge etc. bei den Werkzeugdaten [▶ 184] eingetragen. Hierfür stehen die Parameter 8 bis 10 zur Verfügung. Dabei beschreibt

- P8 immer die X-Komponente
- P9 immer die Y-Komponente
- P10 immer die Z-Komponente

unabhängig von der Ebenenanwahl.



An- und Abwahl der kartesischen Werkzeugverschiebung

Wie auch die Längenkorrektur, wird die Werkzeugverschiebung mit D<n> (n>0) eingeschaltet. Um die Verschiebung herauszufahren, müssen die Achsen wenigstens genannt werden. D.h. die Verschiebung wird dann herausgefahren, wenn die Achse das erste Mal aufgerufen wird. Zusätzlich kann eine neue Endposition für die Achse eingegeben werden.

Ausgeschaltet wird die Funktion mit D0. Auch hier müssen die Achsen wenigstens genannt werden, damit die Achsen auf die neuen Koordinaten fahren.

Beispiel 1:

```
N10 G17 G01 X0 Y0 Z0 F6000
N20 D1 X10 Y10 Z (Z-Axis is repositioned)
N30 ...
N90 M30
```

Beispiel 2:

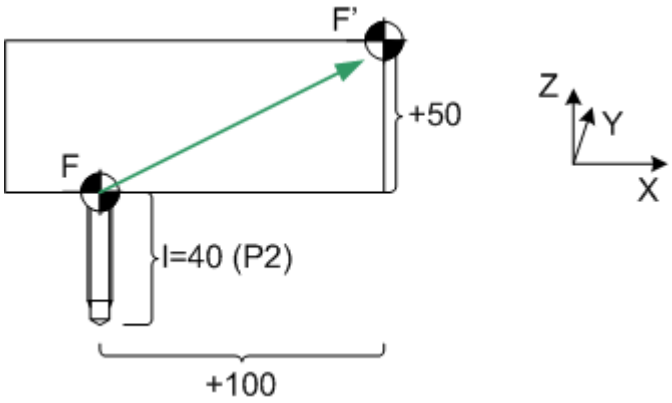
```
N10 G17 G01 X0 Y0 Z0 F6000
N20 D1 X10 Y10 (Z-Axis is not moved)
N30 ...
N90 M30
```

Verwendung von Werkzeugverschiebung und Rotation

Wird die kartesische Werkzeugverschiebung in Verbindung mit der Rotation [▶ 151] verwendet, so wird nur dann richtig kompensiert, wenn das Aggregat (Werkzeugträger) ebenfalls um den gleichen Winkel rotiert wird.

Anwendungsbeispiel

Bei Bearbeitungsmaschinen kommt es häufig vor, dass sich an einem Werkzeugträger mehrere Werkzeuge befinden. Je nachdem welche Bearbeitung anliegt, wird das jeweilige Werkzeug pneumatisch zugeschaltet. Da sich die Werkzeuge nun einmal an unterschiedlichen Orten befinden, wird eine kartesische Werkzeugverschiebung benötigt.



Werkzeugparameter

Parameter	Wert
0	0..65535
1	10
2	40
5	0
8	100.0
9	0.0
10	50

Verhalten bei Kettenmaßangabe

Default-Verhalten

Wird im Kettenmaß (G91) ein neuer Werkzeugversatz (und auch Längenkorrektur) angewählt, so wird mit dem Nennen der Achse der neue Korrekturwert herausgefahren.

Beispiel 3:

```
(Tooloffset D1: X10 Y20 Z30)
N10 G01 D1 X100 Y0 Z0 F6000
N20 G91 (incremental dimension)
N30 D2 (Tooloffset D2: X100
Y200 Z300)
N30 Z10
N40 ...
```

Befehl	Beschreibung
ToolOffsetIncOn	Auch unter G91 werden die Werkzeugverschiebungen und Längenkorrektur herausgefahren, wenn die Achse genannt wird. (Standardeinstellung)
ToolOffsetIncOff	Unter G91 werden die Werkzeugverschiebung und Längenkorrektur **nicht** herausgefahren.

Beispiel 4:

```
(Tooloffset D1: X10 Y20 Z30)
N05 ToolOffsetIncOff
N10 G01 D1 X100 Y0 Z0 F6000
N20 G91 (incremental dimension)
N30 D2 (Tooloffset D2: X100
Y200 Z300)
N30 Z10
N40 ...
```

In N10 wird der Tooloffset für alle 3 Achsen herausgefahren. D.h. die Achsen fahren im Maschinen-Koordinaten-System (MCS) auf X110 Y10 Z30.

In N30 wird der neue Tooloffset der Z-Achse **nicht** herausgefahren. Dadurch ergibt sich im MCS X110 Y10 **Z40.**

Vergl. auch ZeroShiftIncOn/Off [▶ 146]

5.4.4 Fräserradiuskorrektur

5.4.4.1 Fräser-/Schneidenradiuskorrektur aus

Fräser-/Schneidenradiuskorrektur aus

Befehl	G40 (Standardeinstellung)
Aufhebung	G41 [▶ 190] oder G42 [▶ 191]

Die Funktion G40 schaltet die Fräser-/Schneidenradiuskorrektur aus. Dabei bleibt die Längenkorrektur [▶ 187] noch solange aktiv, bis sie mit D0 ausgeschaltet wird. Zwischen G40 und Programmende muss mindestens ein Geometrieelement programmiert werden

5.4.4.2 Fräser-/Schneidenradiuskorrektur links

Fräser-/Schneidenradiuskorrektur links

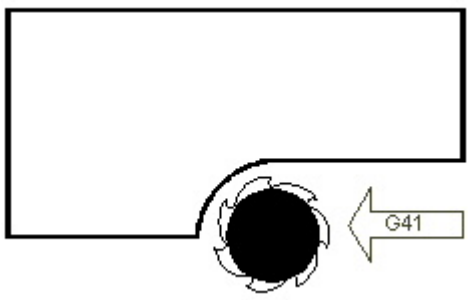
Befehl	G41
Aufhebung	G40 [▶ 190]

Die Funktion G41 schaltet die Fräser-/Schneidenradiuskorrektur ein. Dabei befindet sich das Werkzeug in Bewegungsrichtung **links** vom Werkstück.

Wie schon bei der Längenkorrektur [▶ 187] kann die Fräserradiuskorrektur nur bei wirksamen G0 [▶ 139] oder G1 [▶ 140] aktiviert werden. Mit der Anwahl der Fräserradiuskorrektur müssen die Achsen der Ebene verfahren werden.

Beispiel:

```
N10 G17 G01 X0 Y0 Z0 F6000
N20 G41 X10 Y20 Z D1
N30 X30
N40 G40 X10 Y10 Z
N50 M30
```



Fräserradiuskorrektur gilt nicht für Vollkreise

Es werden keine Vollkreise bei der Fräserradiuskorrektur unterstützt, Sie müssen diese z.B. in Halbkreise aufteilen.

Hinweise:

- Damit das NC-Programm ordnungsgemäß beendet werden kann, ist vor dem Programmende die Fräserradiuskorrektur zu deaktivieren. Zwischen G40 und Programmende muss mindestens ein Geometrieelement programmiert werden.
- Wird ein Dekodierstopp [▶ 172] programmiert, muss die Fräserradiuskorrektur zuvor deaktiviert werden.

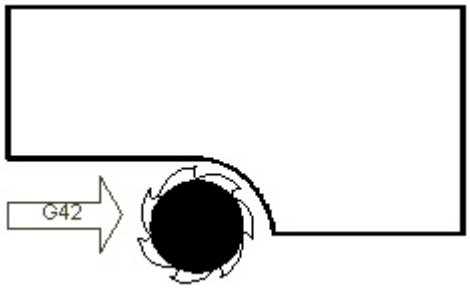
- Durch die Radiuskorrektur kann sich bei Kreisen die Bahngeschwindigkeit an der Kontur verändern, vergl. Sie hierzu 'Bahngeschwindigkeit bei Kreisen [▶ 195]'.
- vergl. Orthogonales An- bzw. Abfahren an der Kontur [▶ 195].

5.4.4.3 Fräser-/Schneidenradiuskorrektur rechts

Fräser-/Schneidenradiuskorrektur rechts

Befehl	G42
Aufhebung	G40 [▶ 190]

Die Funktion G42 schaltet die Fräser-/Schneidenradiuskorrektur ein. Dabei befindet sich das Werkzeug in Bewegungsrichtung **rechts** vom Werkstück.



Fräserradiuskorrektur gilt nicht für Vollkreise

Es werden keine Vollkreise bei der Fräserradiuskorrektur unterstützt, Sie müssen diese z.B. in Halbkreise aufteilen.

Hinweis Für den Fall, dass von G41 nach G42 geschaltet werden soll, so ist zwischen den Geometrien ein G40 zu programmieren.

5.4.4.4 An- und Abfahrverhalten der Fräserradiuskorrektur

Dieses Kapitel beschreibt das An- und Abfahrverhalten beim Einschalten bzw. Ausschalten der Fräserradiuskorrektur. Dieses Verhalten ist abhängig von der Startposition und kann ansonsten nicht beeinflusst werden.

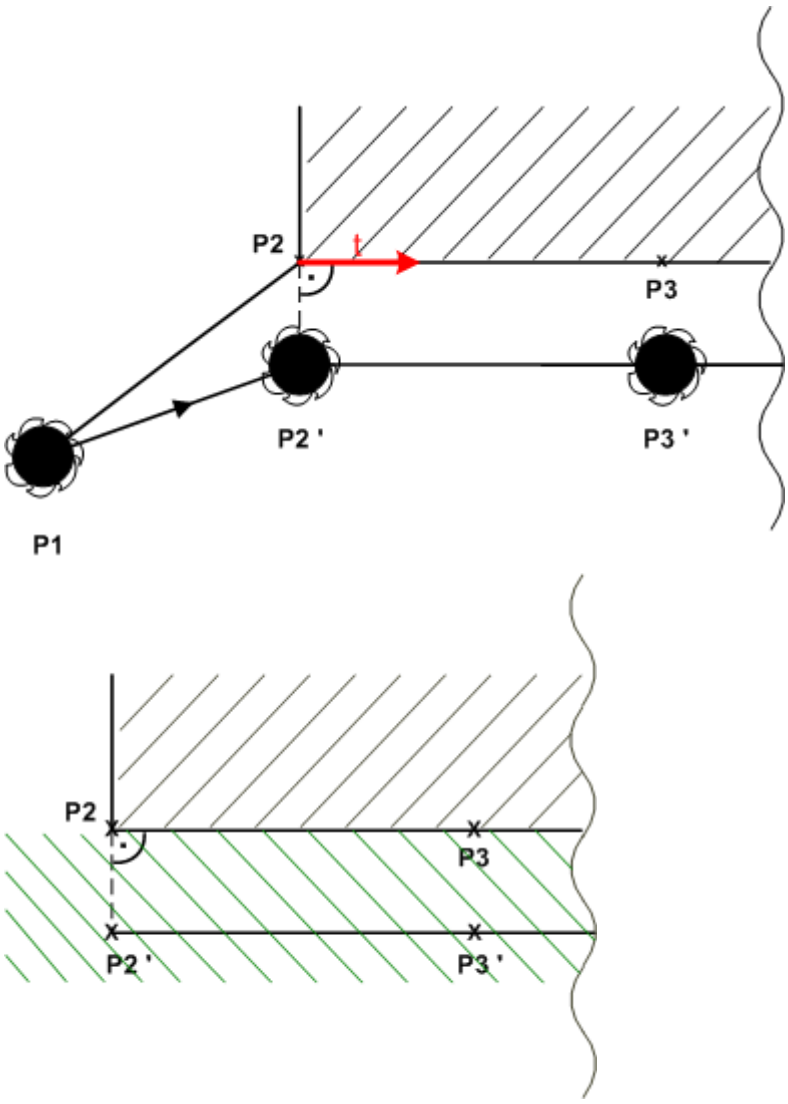
Nach dem Einschalten der Radiuskorrektur muss diese noch herausgefahren werden. D.h. der Fräser steht an einem Punkt P1 (ohne Radiuskorrektur) und verfährt zu einem Punkt P2', wobei der Fräserradius im Punkt P2' kompensiert wird.

Der Punkt P2' ist dabei von der Startposition P1 in der Ebene abhängig, wobei grundsätzlich 3 Fälle unterschieden werden. Diese Fälle werden im Folgenden exemplarisch beim Herausfahren der Radiuskorrektur mit einem programmierten G42 (Korrektur rechts) gezeigt.

Beim Deaktivieren der Korrektur gelten ähnliche Regeln, mit dem Unterschied, dass die Tangente t am Ende des Bahnsegments ermittelt wird und daraus werden die gleichen Bedingungen hergeleitet.

Fall 1: P1 rechts von der Bahntangente t

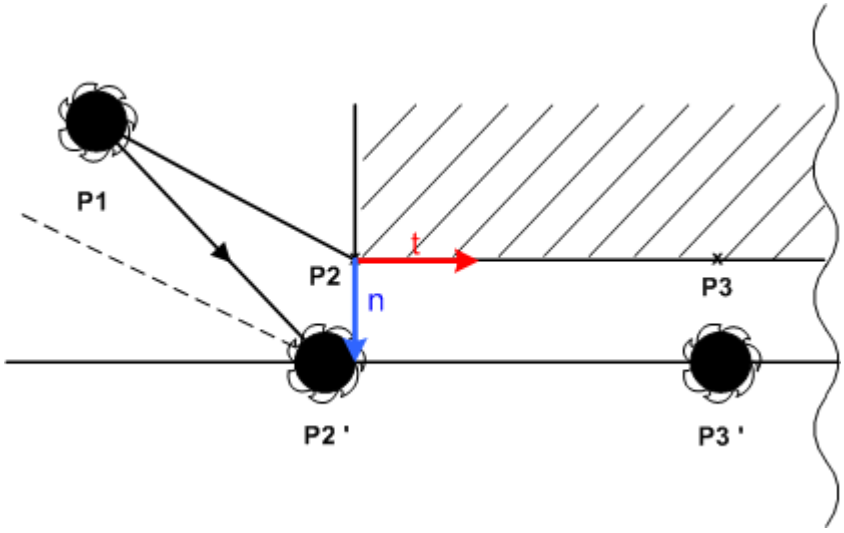
Befindet sich der Startpunkt P1 rechts von der Bahntangente t, so ist P2' orthogonal zur Tangente. Dieses Anfahrverhalten trifft auf den grün schraffierten Bereich zu.

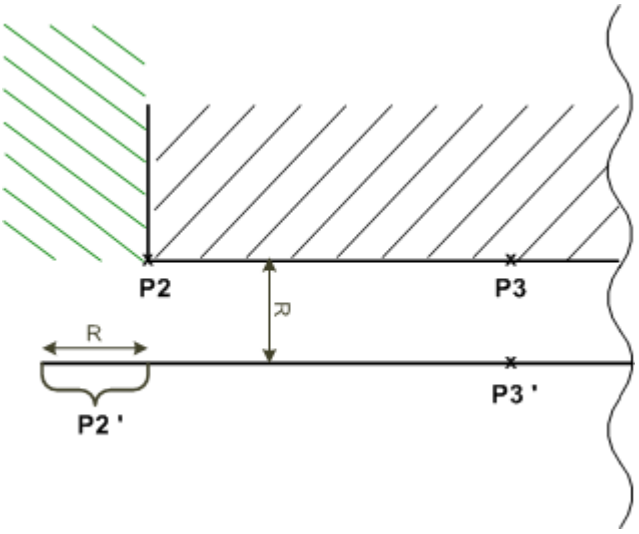


Fall 2: P1 rechts von der Normalen n und links von der Bahntangente t

Für den Fall, dass sich die Startposition P1 rechts von der Normalen n und links von der Bahntangente t befindet, so wird P2' verschoben. Dabei ergibt sich P2' aus dem Schnittpunkt der Parallelen von P1P2 und der verschobenen Strecke P2P3. Beide Geraden werden um den Radius R verschoben.

Diese Verhaltensweise gilt für den grün schraffierten Bereich.

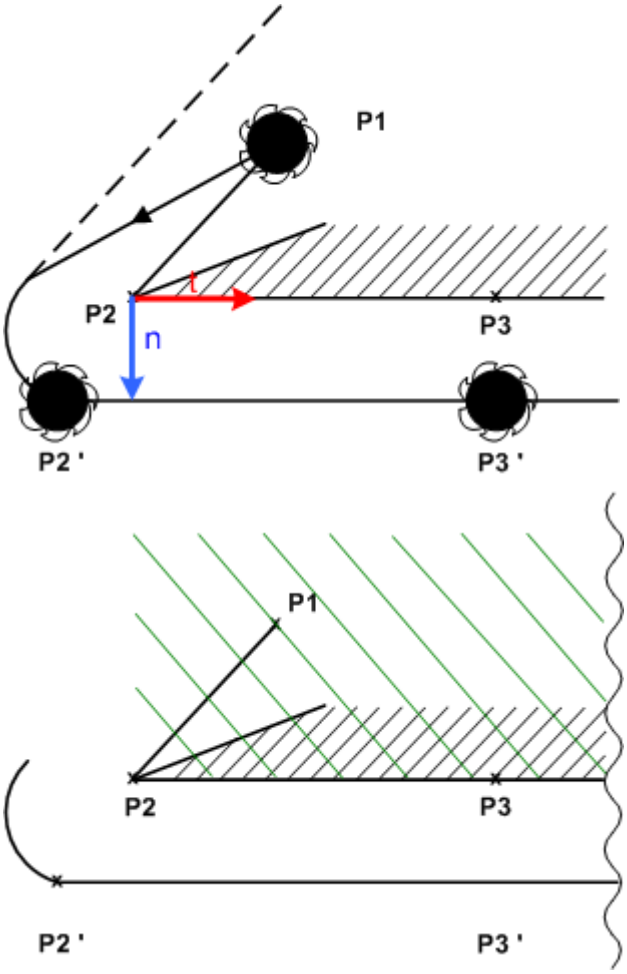




Fall 3: P1 links von der Normalen n und links von der Bahntangente t

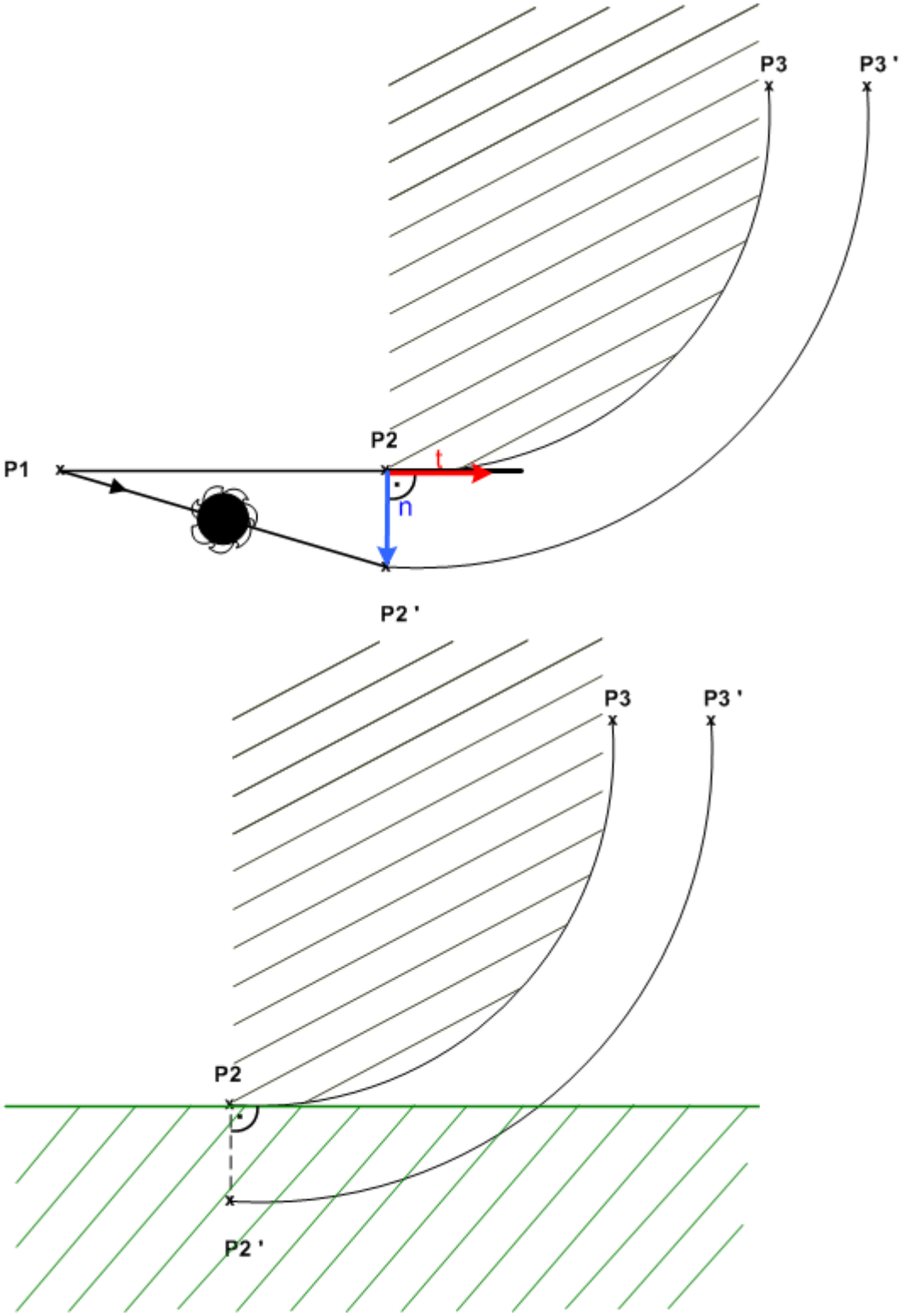
Befindet sich die Startposition P1 links von der Normalen n und auch links von der Bahntangente t, so wird beim Anfahren von P2' ein zusätzliches Kreissegment eingefügt. Damit in P2 kein Freischneiden durchgeführt wird, befindet sich P2' nicht orthogonal zur Anfangstangente der Strecke P2P3.

Das zusätzliche Kreissegment wird für alle Startpositionen im grün schraffierten Bereich eingefügt.



Nach dem Herausfahren folgt ein Kreissegment

Die Radiuskorrektur wird immer mit einer Geraden herausgefahren. (Dies muss im Teileprogramm sichergestellt werden, da sonst ein Laufzeitfehler generiert wird). Danach kann die Kontur mit einem Kreis beginnen. Die Regeln für das An- und Abfahren sind dabei die gleichen wie zuvor. D.h. auch hier wird die Bahntangente der Kontur zu P2 ermittelt und dann die 3 beschriebenen Fälle unterschieden.



Falls P2' immer unabhängig vom Startpunkt orthogonal zur Bahntangente von P2 angefahren werden soll, ist das mit einem zusätzlichen Kommando zu realisieren (vergl. Orthogonales An- bzw. Abfahren an der Kontur [▶ 195]).

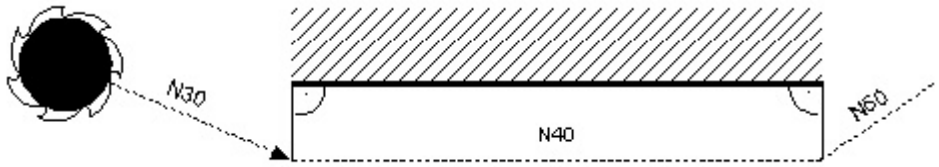
5.4.5 Orthogonales An- bzw. Abfahren der Kontur

Befehl	NORM
Aufhebung	Satzende
Programmierbar mit	G40 [▶ 190] G41 [▶ 190] G42 [▶ 190]

Der Befehl 'NORM' bewirkt, dass beim Einschalten der Fräserradiuskorrektur orthogonal zur Kontur angefahren wird. Dabei spielt die aktuelle Position des Fräsers keine Rolle. Bei der Abwahl wird das letzte Segment mit aktiver Korrektur ebenfalls orthogonal verlassen.

Beispiel:

```
N10 G17
N20 G01 X0 Y0 Z0 F6000
N30 G42 NORM X100 Y0 D5
N40 X200
N50 G40 NORM X220 Y0
N60 M30
```



Hinweis **Der Norm-Befehl ist bislang nur für Geraden-Geraden-Übergänge implementiert.**

5.4.6 Bahngeschwindigkeit bei Kreisen

Bei eingeschalteter Fräserradiuskorrektur [▶ 190] ändert sich bei Kreisen der programmierte Kreisradius. Damit verändert sich ebenfalls die Vorschubgeschwindigkeit. Mit den folgenden Befehlen wird festgelegt, ob sich die Vorschubangabe auf die Kontur bezieht oder auf den Werkzeugmittelpunkt.

Konstanter Vorschub an der Kontur

Befehl	CFC (Standardeinstellung)
Aufhebung	CFIN oder CFTCP

Mit CFC (constant feed contour) wird der Vorschub an der Kontur konstant gehalten.

Konstanter Vorschub an Innenkreisen

Befehl	CFIN
Aufhebung	CFC oder CFTCP

Mit CFIN (constant feed internal radius) wird der Vorschub an Innenkreisen reduziert. Dadurch ergibt sich an der Kontur eine konstante Geschwindigkeit. Am Außenkreis wird die Geschwindigkeit nicht erhöht.

Konstanter Vorschub des Werkzeugmittelpunkts

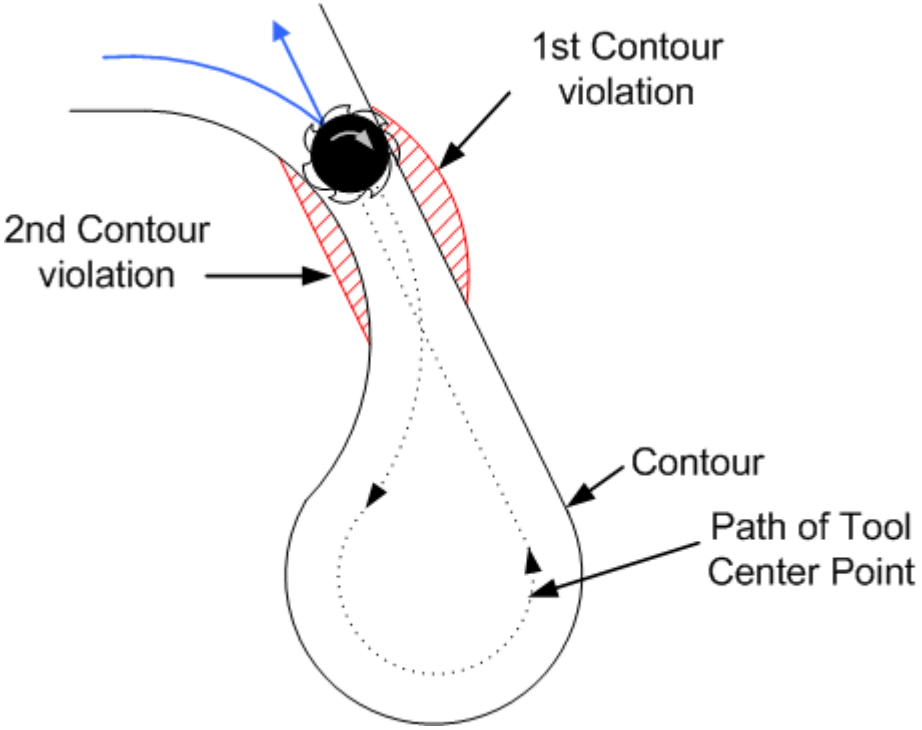
Befehl	CFTCP
Aufhebung	CFC oder CFIN

Mit CFTCP (constant feed tool center point) wird der Vorschub des Werkzeugmittelpunkts konstant gehalten. D.h. an Innenkreisen wird die Geschwindigkeit an der Kontur erhöht und an Außenkreisen entsprechend verringert.

5.4.7 Flaschenhalserkennung

Wird bei der Erstellung von Teileprogrammen der Radius des Fräsers nicht berücksichtigt, so kann es vorkommen, dass der Fräser z.B. ungewollt die gegenüberliegende Seite des Werkstücks bearbeitet. D.h. es resultiert eine Konturkollision mit dem Werkstück oder anders ausgedrückt, es wurde ein Flaschenhals programmiert.

Befehl	CDON
Aufhebung	CDOF



Dieses Verhalten kann in dieser Form nur in Verbindung mit der Fräserradiuskorrektur (G41/G42) auftreten. Um derartige Konturkollisionen zu verhindern, kann aus dem Teileprogramm mit **CDON** die Überwachung eingeschaltet werden. Damit sie auch wirklich aktiv ist, muss auch die Fräserradiuskorrektur angewählt sein.

Die Reaktion der NCI, wenn ein Flaschenhals erkannt wird, kann mit Hilfe der SPS parametriert werden. Dabei werden 3 Fälle unterschieden:

- Fehler und Abbruch
  Wird ein Flaschenhals erkannt, so generiert die NC einen Laufzeitfehler und bricht die Bearbeitung des Programms ab.
- Hinweis und Modifizierung der Kontur
  Wenn ein Flaschenhals erkannt wird, dann wird die Kontur so modifiziert, dass keine Konturkollision auftritt (vergl. Bild 1: blaue Linie). D.h. aber auch, dass je nach Programm Segmente ausgelassen werden. Außerdem wird ein Hinweis im Applikationsviewer eingetragen, dass ein Flaschenhals erkannt wurde.
- Hinweis und Konturkollision
  Wird ein Flaschenhals erkannt, so wird bei dieser Einstellung weder die Kontur verändert noch ein Fehler generiert. Es wird lediglich eine Meldung in den Applikation Viewer eingetragen.

Für die Konturkollisionsüberwachung wird eine gewisse Rechner-Performance benötigt. Deshalb sollte sie nur dann angewählt werden, wenn auch wirklich Bedarf besteht. Des Weiteren sollte noch die Größe des Look-Aheads für die Flaschenhalserkennung festgelegt werden. Dabei wird die Anzahl der Segmente bestimmt, die relativ vom n-ten Segment in die Zukunft geschaut wird, um sie auf Flaschenhälse zu überprüfen. Hier sollte die Anzahl der Segmente nicht unnötig groß gewählt werden, da ansonsten das System belastet wird. Der Wert für den Look-Ahead wird ebenfalls aus der SPS parametriert.

Funktionsbausteine zum Parametrieren der Flaschenhalserkennung:

- ItpSetBottleNeckModeEx [▶ 242]

- ItpGetBottleNeckModeEx [▶ 218]
- ItpSetBottleNeckLookAheadEx [▶ 241]
- ItpGetBottleNeckLookAheadEx [▶ 217]

Beispiel:

```
N10 G0 X0 Y0 Z0
N20 CDON
N30 G01 G41 D3 X100 F6000 (cutter radius 30mm)
...
N40 G01 X200
N50 G02 X220 Y-74.641 I0 J-40
N60 G01 X300 Y-104
N70 G01 X230 Y120
N80 G40 D0 Y200
N90 CDOF
...
M30
```

5.5 Befehlsübersicht

5.5.1 Allgemeine Kommandoübersicht

Kommando	Beschreibung	satzweise / modal	Default
ANG [▶ 150]	Konturzugprogrammierung (Winkel)	s	
AROT [▶ 151]	Rotation additiv	m	
CalcInvRot [▶ 151]	Berechnet die inverse Rotation eines Vektors	s	
CalcRot [▶ 151]	Berechnet die Rotation eines Vektors	s	
CDOF [▶ 196]	Flaschenhalserkennung aus	m	Default
CDON [▶ 196]	Flaschenhalserkennung ein	m	
CFC [▶ 195]	Konstante Geschwindigkeit an der Kontur	m	Default
CFIN [▶ 195]	Konstante Geschwindigkeit im Innenkreis	m	
CFTCP [▶ 195]	Konstante Geschwindigkeit des Werkzeugmittelpunkts	m	
CIP [▶ 141]	Kreisinterpolation	s	
CPCOF [▶ 141]	Mittelpunktskorrektur aus	m	
CPCON [▶ 141]	Mittelpunktskorrektur ein	m	Default
DelDTG [▶ 162]	Restweglöschen	s	
DYNOVR [▶ 179]	Dynamischer Override	m	
FCONST [▶ 144]	Konstante Vorschubprogrammierung	m	Default
FLIN [▶ 144]	Lineare Vorschubprogrammierung	m	
G00 [▶ 139]	Eilgang	m	
G01 [▶ 140]	Geradeninterpolation	m	Default

Kommando	Beschreibung	satzweise / modal	Default
G02 [▶ 141]	Kreisinterpolation im Uhrzeigersinn	m	
G03 [▶ 141]	Kreisinterpolation im Gegenuhrzeigersinn	m	
G04 [▶ 143]	Verweilzeit	s	
G09 [▶ 144]	Genauhalt	s	
G17 [▶ 132]	Ebenenanwahl XY	m	Default
G18 [▶ 132]	Ebenenanwahl ZX	m	
G19 [▶ 132]	Ebenenanwahl YZ	m	
G40 [▶ 190]	Keine Fräser-/ Schneidenradiuskorrektur	m	Default
G41 [▶ 190]	Fräser-/ Schneidenradiuskorrektur links	m	
G42 [▶ 190]	Fräser-/ Schneidenradiuskorrektur rechts	m	
G53 [▶ 146]	Unterdrückung der Nullpunktverschiebung	m	Default
G54 [▶ 146]	1. einstellbare Nullpunktverschiebung	m	
G55 [▶ 146]	2. einstellbare Nullpunktverschiebung	m	
G56 [▶ 146]	3. einstellbare Nullpunktverschiebung	m	
G57 [▶ 146]	4. einstellbare Nullpunktverschiebung	m	
G58 [▶ 146]	1. programmierbare Nullpunktverschiebung	m	
G59 [▶ 146]	2. programmierbare Nullpunktverschiebung	m	
G60 [▶ 144]	Genauhalt	m	
G70 [▶ 134]	Maßangaben inch	m	
G71 [▶ 134]	Maßangabe metrisch	m	Default
G74 [▶ 139]	Referenzpunktanfahren per Programm	s	
G90 [▶ 132]	Bezugsmaßangabe	m	Default
G91 [▶ 132]	Kettenmaßangabe	m	
G700 [▶ 134]	Maßangabe inch mit Verrechnung des Vorschubs	m	
G710 [▶ 134]	Maßangabe metrisch mit Verrechnung des Vorschubs	m	
Mirror [▶ 154]	Koordinatensystem spiegeln	m	
MOD [▶ 162]	Modulobewegung	s	
MSG [▶ 184]	Meldungen aus dem NC-Programm	s	
NORM [▶ 195]	orthogonales An-Abfahren an der Kontur	s	

Kommando	Beschreibung	satzweise / modal	Default
P+ [▶ 132]	Zustellrichtung positiv	m	Default
P- [▶ 132]	Zustellrichtung negativ	m	
paramAutoAccurateStop [▶ 161]	Automatischer Genauhalt	m	
paramAxisDynamics [▶ 179]	Parametrierung der Achsdynamik	m	
paramC1ReductionFactor [▶ 180]	C1 Reduktionsfaktor	m	
paramC2ReductionFactor [▶ 180]	C2 Reduktionsfaktor	m	
paramCircularSmoothing [▶ 160]	Verrundung	m	
paramDevAngle [▶ 180]	C0 Reduktion - Ablenkungswinkel	m	
paramGroupVertex [▶ 160]	Verrundung (alt)	m	
paramGroupDynamic [▶ 179]	Bahndynamik (alt)	m	
paramPathDynamics [▶ 179]	Bahndynamik	m	
paramRadiusPrec [▶ 142]	Kreisgenauigkeit	m	
paramSplineSmoothing [▶ 158]	Glättung mit Bezier-Splines	m	
paramVertexSmoothing [▶ 155]	Glättung von Segmentübergängen	m	
paramVeloJump [▶ 180]	C0 Reduktion - max. Geschwindigkeitssprung	m	
paramVeloMin [▶ 182]	Mindestgeschwindigkeit	m	
paramZeroShift [▶ 146]	Parametrierung der einstellbaren Nullpunktverschiebung	m	
PathAxesPos [▶ 183]	Liest aktuelle Ist-Position	s	
ROT [▶ 151]	Rotation absolut	m	
RotExOff [▶ 151]	Erweiterte Rotationsfunktion aus	m	Default
RotExOn [▶ 151]	Erweiterte Rotationsfunktion ein	m	
RotVec [▶ 151]	Berechnungsroutine zum Drehen eines Vektors	s	
RParam [▶ 136]	Initialisierung von R-Parametern	s	
RToDwordGetBit [▶ 136]	Wandelt einen R-Parameter zum DWord und prüft, ob ein definiertes Bit gesetzt ist	m	
SEG [▶ 150]	Konturzugprogrammierung (Segmentlänge)	s	
skip VirtualMovements [▶ 183]	Virtuelle Bewegungen überspringen	m	

Kommando	Beschreibung	satzweise / modal	Default
ToolOffsetIncOff [▶ 187]	Kartesische Werkzeugverschiebung und Längenkorrektur wird unter G91 nicht herausgefahren	m	
ToolOffsetIncOn [▶ 187]	Kartesische Werkzeugverschiebung und Längenkorrektur wird unter G91 herausgefahren	m	Default
ToolParam [▶ 184]	Schreiben und lesen von Werkzeugparametern	m	
TPM [▶ 148]	Zielpositionsüberwachung	s	
ZeroShiftIncOff [▶ 146]	Nullpunktverschiebung wird unter G91 nicht herausgefahren	m	
ZeroShiftIncOn [▶ 146]	Nullpunktverschiebung wird unter G91 herausgefahren	m	Default

Adresse	Beschreibung
Q<n> [▶ 164]	Achsbezeichner der Hilfsachse (1 <= n <= 5)

5.5.2 @-Kommando Übersicht

Bei diesen Befehlen sind oft mehrere Varianten möglich, die dadurch entstehen, dass für einen Parameter mit K eine Konstante, mit R ein R-Parameter und mit P ein als Pointer verwendeter R-Parameter angegeben werden kann. Beispielsweise ist die Schreibweise K/R/Pn zu verstehen als "entweder eine Zahl oder ein R-Parameter oder ein Pointer".

Die folgenden @-Befehle stehen zur Verfügung:

Kommando	Varianten	Funktion
@40 [▶ 136]	@40 Kn Rn Rm	Rette Register auf dem Stack
@41 [▶ 136]	@41 Rn Rm	Rette Register auf dem Stack
@42 [▶ 136]	@42 Kn .... Rm Rn	Restauriere Register vom Stack
@43 [▶ 136]	@43 Rm Rn	Restauriere Register vom Stack
@100 [▶ 174]	@100 K±n @100 Rm	Unbedingter Sprung
@111 [▶ 174]	@111 Rn K/Rn Km ...	Case-Anweisung
@121 [▶ 174]	@121 Rn K/Rn Kn	Springe wenn ungleich
@122 [▶ 174]	@122 Rn K/Rn Kn	Springe wenn gleich
@123 [▶ 174]	@123 Rn K/Rn Kn	Springe wenn kleiner gleich
@124 [▶ 174]	@124 Rn K/Rn Kn	Springe wenn kleiner
@125 [▶ 174]	@125 Rn K/Rn Kn	Springe wenn größer gleich
@126 [▶ 174]	@126 Rn K/Rn Kn	Springe wenn größer
@131 [▶ 175]	@131 Rn K/Rn Kn	Schleife solange gleich
@132 [▶ 175]	@132 Rn K/Rn Kn	Schleife solange ungleich
@133 [▶ 175]	@133 Rn K/Rn Kn	Schleife solange grösser
@134 [▶ 175]	@134 Rn K/Rn Kn	Schleife solange grösser oder gleich
@135 [▶ 175]	@135 Rn K/Rn Kn	Schleife solange kleiner

Kommando	Varianten	Funktion
@136 [▶ 175]	@136 Rn K/Rn Kn	Schleife solange kleiner oder gleich
@141 [▶ 175]	@141 Rn K/Rn Kn	Wiederhole bis gleich
@142 [▶ 175]	@142 Rn K/Rn Kn	Wiederhole bis ungleich
@143 [▶ 175]	@143 Rn K/Rn Kn	Wiederhole bis grösser
@144 [▶ 175]	@144 Rn K/Rn Kn	Wiederhole bis grösser oder gleich
@145 [▶ 175]	@145 Rn K/Rn Kn	Wiederhole bis kleiner
@146 [▶ 175]	@146 Rn K/Rn Kn	Wiederhole bis kleiner oder gleich
@151 [▶ 175]	@151 Rn K/Rn Kn	FOR_TO-Schleife
@161 [▶ 175]	@161 Rn K/Rn Kn	FOR_DOWNTO-Schleife
@200	@200 Rn	Löschen einer Variablen
@202	@202 Rn Rm	Vertauschen von zwei Variablen
@302	@302 K/R/Pn K/R/Pn R/Pn	Lese Maschinendatenbit
@361 [▶ 182]	@361 Rn Km	Lese maschinenbezogenen Achsen-Istwert
@372	@372 Rn	Auslesen der NC-Kanal-ID und Speichern in einer Variablen
@402 [▶ 141]	@402 K/R/Pn K/R/Pn K/R/Pn	Schreibe Maschinendatenbit
@610	@610 Rn Rn	Absolutwert einer Variablen ermitteln
@613	@613 Rn Rn	Quadratwurzel einer Variablen ermitteln
@614	@614 Rn Rm Rm	Quadratwurzel der Summe der Quadrate von zwei Variablen ermitteln x = sqrt(a^2 + b^2)
@620 [▶ 175]	@620 Rn	Variable inkrementieren
@621	@621 Rn	Variable dekrementieren
@622	@622 Rn	Ganzzahl einer Variablen ermitteln
@630 [▶ 137]	@630 Rn Rm	Sinus einer Variablen ermitteln
@631 [▶ 137]	@631 Rn Rm	Cosinus einer Variablen ermitteln
@632 [▶ 137]	@632 Rn Rm	Tangens einer Variablen ermitteln
@633 [▶ 137]	@633 Rn Rm	Cotangens einer Variablen ermitteln
@634 [▶ 137]	@634 Rn Rm	Arcus Sinus einer Variablen ermitteln
@635 [▶ 137]	@635 Rn Rm	Arcus Cosinus einer Variablen ermitteln
@636 [▶ 137]	@636 Rn Rm	Arcus Tangens einer Variablen ermitteln
@714 [▶ 172]	@714	Dekodier-Stop
@716 [▶ 172]	@716	Dekodier-Stop mit Rescan der Achspositionen
@717 [▶ 172]	@717	Dekodier-Stop mit externen Triggerevent

Maschinendaten

Zugriffe auf folgende Maschinendaten werden unterstützt:

Byte	Bit	Wirkungsweise
5003 [▶ 141]	5	0: IJK-Worte legen den Abstand des Kreismittelpunktes vom Anfangspunkt fest. 1: IJK sind Absolutangaben des Kreismittelpunktes.

6 PLC NCI Libraries

Voraussetzungen

Übersicht der PLC NCI Libraries	Beschreibung
PLC Library: Tc2_NCI [▶ 203]	Bausteine für die Konfiguration der Interpolationsgruppe (z.B. Bilden der 3D Gruppe) und für die Bedienung des Interpreters (G-Codes (DIN 66025)) wie z.B. Laden und Starten des NC-Programms.
PLC Library: Tc2_PlcInterpolation [▶ 304]	Programmierung von mehrdimensionalen Bewegungen aus der SPS (Alternative zur Benutzung des G-Codes (DIN 66025))

6.1 PLC Library: Tc2_NCI

6.1.1 Konfiguration

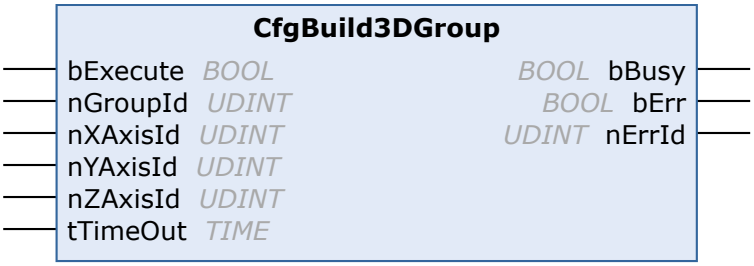
Die Bibliothek Tc2_NCI stellt Funktionsbausteine zur allgemeinen NC Achskonfiguration bereit. Damit ist es möglich, direkt aus der SPS heraus Achsen auf einfache Art und Weise zu konfigurieren bzw. umzukonfigurieren.

Funktionsbaustein	Beschreibung
CfgBuild3DGroup [▶ 203]	Gruppiert bis zu 3 PTP-Achsen in einer 3D-Gruppe
CfgBuildExt3DGroup [▶ 204]	Gruppiert bis zu 3 PTP-Achsen und 5 Hilfsachsen in einer 3D-Gruppe
CfgAddAxisToGroup [▶ 206]	Konfiguriert eine einzelne Achse an einen bestimmten Platz innerhalb einer Gruppe (PTP, 3D, FIFO)
CfgReconfigGroup [▶ 207]	Auflösen einer 3D- (FIFO-) Achsbelegung und Rückführung der Achsen in ihre persönliche PTP-Gruppe
CfgReconfigAxis [▶ 207]	Rückführung einer einzelnen Achse z.B. aus einer 3D-Gruppe in ihre persönliche PTP-Gruppe
CfgRead3DAxisIds [▶ 208]	Lesen der Achs-IDs (Achsbelegung) einer 3D-Gruppe
CfgReadExt3DAxisIds [▶ 209]	Lesen der Achs-IDs (Achsbelegung) einer 3D-Gruppe mit Hilfsachsen

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.1.1 CfgBuild3DGroup



Dieser Baustein konfiguriert eine 3D-Gruppe mit bis zu 3 PTP-Achsen (X, Y und Z).

VAR_INPUT

```
VAR_INPUT
    bExecute       : BOOL;
    nGroupId       : UDINT;
    nXAxisId       : UDINT;
    nYAxisId       : UDINT;
    nZAxisId       : UDINT;
    tTimeOut       : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt

nGroupId: ID der 3D-Gruppe

nXAxisId: ID der PTP-Achsen

nYAxisId: ID der PTP-Achsen

nZAxisId: ID der PTP-Achsen

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy     : BOOL;
    bErr      : BOOL;
    nErrId    : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

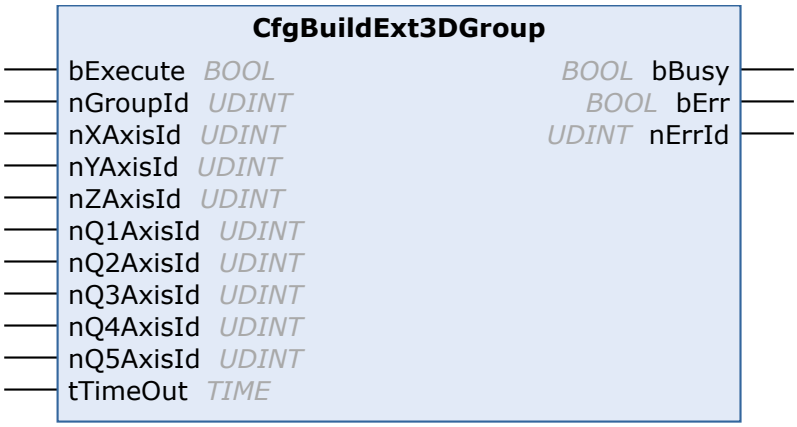
bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.1.2 CfgBuildExt3DGroup



Dieser Baustein konfiguriert eine 3D-Gruppe mit bis zu 3 Bahnachsen (X, Y und Z). Zusätzlich können bis zu 5 Hilfsachsen (Q1..Q5) konfiguriert werden.

An den Eingängen **nXAxisId** bis **nQ5AxisId** werden die Achs-IDs der PTP-Achsen angelegt, die in der Interpolationsgruppe aufgenommen werden sollen.

Hinweis Bei der Zuweisung der Hilfsachsen muss mit nQ1AxisId begonnen werden und es dürfen keine Plätze zwischen Hilfsachsen freigelassen werden. D.h. wenn z.B. nQ3AxisId belegt werden soll, dann muss auch nQ2AxisId eine gültige Achs-ID zugewiesen werden.

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    nGroupId        : UDINT;
    nXAxisId        : UDINT;
    nYAxisId        : UDINT;
    nZAxisId        : UDINT;
    nQ1AxisId       : UDINT;
    nQ2AxisId       : UDINT;
    nQ3AxisId       : UDINT;
    nQ4AxisId       : UDINT;
    nQ5AxisId       : UDINT;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Bei einer positiven Flanke wird der Befehl ausgeführt.

nGroupId: ID der 3D-Gruppe

nXAxisId: Achs-IDs der PTP-Achsen, die in der Interpolationsgruppe aufgenommen werden sollen

nYAxisId: Achs-IDs der PTP-Achsen, die in der Interpolationsgruppe aufgenommen werden sollen

nZAxisId: Achs-IDs der PTP-Achsen, die in der Interpolationsgruppe aufgenommen werden sollen

nQ1AxisId: Achs-IDs der PTP-Achsen, die in der Interpolationsgruppe aufgenommen werden sollen

nQ2AxisId: Achs-IDs der PTP-Achsen, die in der Interpolationsgruppe aufgenommen werden sollen

nQ3AxisId: Achs-IDs der PTP-Achsen, die in der Interpolationsgruppe aufgenommen werden sollen

nQ4AxisId: Achs-IDs der PTP-Achsen, die in der Interpolationsgruppe aufgenommen werden sollen

nQ5AxisId: Achs-IDs der PTP-Achsen, die in der Interpolationsgruppe aufgenommen werden sollen

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bErr        : BOOL;
    nErrId      : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.1.3 CfgAddAxisToGroup

Der Baustein CfgAddAxisToGroup konfiguriert eine einzelne Achse an einen bestimmten Platz innerhalb einer bestehenden Gruppe (PTP, 3D, FIFO).

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    nGroupId        : UDINT;
    nAxisId         : UDINT;
    nIndex          : UDINT;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt

nGroupId: ID der Ziel-Gruppe

nAxisId: ID der zu konfigurierenden Achse

nIndex: Platz der Achse innerhalb der Gruppe, kann Werte von 0...n-1 annehmen. Dabei hat n je nach Gruppentyp folgende Wertigkeit: PTP: n = 1, 3D: n = 3, FIFO: n = 8

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bErr        : BOOL;
    nErrId      : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.1.4 CfgReconfigGroup

Der Baustein CfgReconfigGroup löst die Achsbelegung einer bestehenden Gruppe (NCI oder FIFO) auf und führt die Achsen in ihre persönliche PTP-Gruppe zurück.

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    nGroupId        : UDINT;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt

nGroupId: ID der aufzulösenden Gruppe

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy      : BOOL;
    bErr       : BOOL;
    nErrId     : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

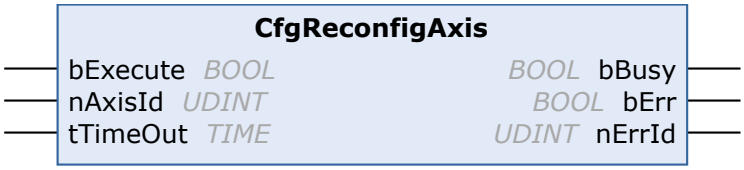
bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.1.5 CfgReconfigAxis



Der Baustein CfgReconfigAxis führt eine einzelne Achse aus z.B. einer 3D-Gruppe in ihre persönliche PTP-Gruppe zurück.

Interface

```
VAR_INPUT
    bExecute        : BOOL;
    nAxisId         : UDINT;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt

nAxisId: ID der rückzuführenden Achse

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy      : BOOL;
    bErr       : BOOL;
    nErrId     : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.1.6 CfgRead3DAxisIds

Der Baustein CfgRead3DAxisIds liest die Achskonfiguration einer 3D-Gruppe.

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    nGroupId        : UDINT;
    pAddr           : PVOID;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt

nGroupId: ID der 3D-Gruppe

pAddr: Adresse der Variablen, in die der Baustein die Achs-Ids der Gruppenbelegung schreibt (Array mit drei Elementen vom Typ UDINT)

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy     : BOOL;
    bErr      : BOOL;
    nErrId    : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Beispiel:

```
VAR
    (* instance *)
    ReadAxIds : CfgRead3DAxisIds;
    AxIds : ARRAY[1..3] OF UDINT;
END_VAR
```

```
ReadAxIds( bExecute := TRUE,
    nGroupId := 4,
    pAddr := ADR( AxIds ),
    tTimeOut := T#1s );
```

AxIds enthält nun die drei Achs-IDs der 3D-Gruppe mit der Gruppen-ID 4.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.1.7 CfgReadExt3DAxisIds

Der Baustein CfgReadExt3DAxisIds liest die Achskonfiguration der erweiterten 3D-Gruppe.

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    nGroupId        : UDINT;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt

nGroupId: ID der 3D-Gruppe

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
    stExt3dGroup    : NCI_EXT3DGROUP;
END_VAR
```

stExt3dGroup: Instanz der Struktur NCI_EXT3DGROUP (hier die Achs-IDs der aktuellen Interpolationsgruppe eintragen)

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy      : BOOL;
    bErr       : BOOL;
    nErrId     : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

```
TYPE NCI_EXT3DGROUP :
STRUCT
    nXAxisId        : UDINT;
    nYAxisId        : UDINT;
    nZAxisId        : UDINT;
    nQ1AxisId       : UDINT;
    nQ2AxisId       : UDINT;
    nQ3AxisId       : UDINT;
    nQ4AxisId       : UDINT;
    nQ5AxisId       : UDINT;
END_STRUCT
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2 NCI POUs

Die TwinCAT-Bibliothek Tc2_NCI enthält Funktionsbausteine zur Bedienung des NC-Interpreters aus der SPS heraus.

Die folgenden Funktionsbausteine sind in der Bibliothek Tc2_NCI enthalten.

Funktionsbaustein	Beschreibung
ItpConfirmHsk [▶ 212]	Bestätigt eine M-Funktion vom Typ Handshake
ItpDelDtgEx [▶ 213]	Triggert das Restweglöschen in der NC
ItpEnableDefaultGCode [▶ 214]	Führt vor dem Start jedes NC-Programms einen vom Anwender definierten Standard G-Code aus
ItpEStopEx [▶ 215]	Triggert den NCI EStop
ItpGetBlockNumber [▶ 216]	Liefert die Blocknummer des NC Programms des zyklischen Interfaces
ItpGetBottleNeckLookAheadEx [▶ 217]	Liefert die Größe des Look-Aheads für die Flaschenhalserkennung

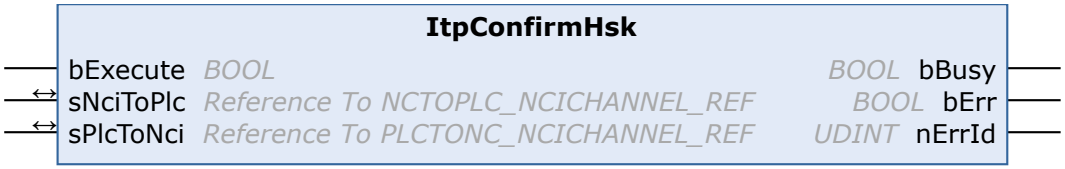
Funktionsbaustein	Beschreibung
ItpGetBottleNeckModeEx [▶ 218]	Liefert den Reaktionsmode für die Flaschenhalserkennung
ItpGetChannelId [▶ 219]	Liefert die Kanal ID
ItpGetChannelType [▶ 219]	Liefert den Kanaltyp des zyklischen Interfaces
ItpGetCyclicLRealOffsets [▶ 220]	Liefert die Index Offsets der im zyklischen Kanalinterface verwendeten LREAL Variablen
ItpGetCyclicUdintOffsets [▶ 221]	Liefert die Index Offsets der im zyklischen Kanalinterface verwendeten UDINT Variablen
ItpGetError [▶ 222]	Liefert die Fehlernummer
ItpGetGeoInfoAndHParamEx [▶ 223]	Liest Informationen über das gerade aktive Segment, vergangene und zukünftige Segmente aus.
ItpGetGroupAxisIds [▶ 224]	Liefert die Achs-IDs die für die Gruppe konfiguriert worden sind
ItpGetGroupId [▶ 225]	Liefert die Gruppen ID
ItpGetHParam [▶ 225]	Liefert den aktuellen H-Parameter aus der NC
ItpGetHskMFunc [▶ 226]	Liefert die aktuell anliegende M-Funktionsnummer vom Typ Handshake
ItpGetItfVersion [▶ 227]	Liefert die aktuelle Version des zyklischen Interfaces
ItpGetOverridePercent [▶ 227]	Liefert den Kanaloverride in Prozent
ItpGetSetPathVelocity [▶ 228]	Liefert die momentane Bahnsollgeschwindigkeit
ItpGetSParam [▶ 228]	Liefert den aktuellen S-Parameter aus der NC
ItpGetStateInterpreter [▶ 229]	Liefert den aktuellen Status des Interpreters
ItpGetTParam [▶ 229]	Liefert den aktuellen T-Parameter aus der NC
ItpGoAheadEx [▶ 230]	Triggert die GoAhead Funktion (Dekodierstopp mit externem Triggerevent)
ItpHasError [▶ 231]	Ermittelt, ob ein Fehler anliegt
ItpIsFastMFunc [▶ 231]	Ermittelt, ob die mitgegebene M-Funktionsnummer als schnelle M-Funktion anliegt
ItpIsEStopEx [▶ 232]	Ermittelt, ob ein EStop ausgeführt wird bzw. ob ein EStop anliegt
ItpIsHskMFunc [▶ 233]	Ermittelt, ob eine M-Funktion vom Typ Handshake anliegt
ItpLoadProgEx [▶ 233]	Lädt ein NC-Programm via Programmnamen
ItpReadCyclicLRealParam1 [▶ 235]	Liest den ersten LReal Parameter aus dem zyklischen Kanalinterface
ItpReadCyclicUdintParam1 [▶ 235]	Liest den ersten UDint Parameter aus dem zyklischen Kanalinterface
ItpReadRParamsEx [▶ 236]	Liest Rechenparameter
ItpReadToolDescEx [▶ 237]	Liest die Werkzeugbeschreibung aus der NC
ItpReadZeroShiftEx [▶ 238]	Liest die Nullpunktverschiebung aus der NC
ItpResetEx2 [▶ 239]	Führt einen Reset des Interpreters, bzw. des NC-Kanals aus
ItpResetFastMFuncEx [▶ 240]	Setzt ein schnelles Signal-Bit zurück
ItpSetBottleNeckLookAheadEx [▶ 241]	Setzt die Größe des Look-Ahead für die Flaschenhalserkennung
ItpSetBottleNeckModeEx [▶ 242]	Setzt den Reaktionsmode bei eingeschalteter Flaschenhalserkennung
ItpSetCyclicLRealOffsets [▶ 243]	Setzt die Index Offsets der im zyklischen Kanalinterface verwendeten LREAL Variablen

Funktionsbaustein	Beschreibung
ItpSetCyclicUdintOffsets [▶ 245]	Setzt die Index Offsets der im zyklischen Kanalinterface verwendeten UDINT Variablen
ItpSetOverridePercent [▶ 246]	Setzt den Kanaloverride in Prozent
ItpSetSubroutinePathEx [▶ 247]	Setzt optional den Suchpfad für Unterprogramme
ItpSetToolDescNullEx [▶ 248]	Setzt alle Toolparameter (inkl. Nummer und Type) auf Null
ItpSetZeroShiftNullEx [▶ 249]	Setzt alle Nullpunktverschiebungen auf Null
ItpSingleBlock [▶ 250]	Aktiviert bzw. deaktiviert die Einzelsatzausführung in der NCI.
ItpStartStopEx [▶ 251]	Startet bzw. stoppt den Interpreter (NC-Kanal)
ItpStepOnAfterEStopEx [▶ 252]	Ermöglicht die Weiterbearbeitung des Teileprogramms nach einem NCI EStop
ItpWriteRParamsEx [▶ 253]	Schreibt Rechenparameter
ItpWriteToolDescEx [▶ 254]	Schreibt die Werkzeugbeschreibung in die NC
ItpWriteZeroShiftEx [▶ 256]	Schreibt die Nullpunktverschiebung in die NC
Blocksearch (Funktionalitätsbeschreibung siehe Blocksearch [▶ 257])	
ItpBlocksearch [▶ 257]	Setzt den Interpreter auf eine vom Anwender definierte Stelle, sodass das NC-Programm ab dieser Stelle weiter abgearbeitet wird.
ItpGetBlocksearchData [▶ 260]	Liest nach der Unterbrechung eines NC-Programms den aktuellen Zustand aus.
ItpStepOnAfterBlocksearch [▶ 261]	Startet die Bewegung nachdem ein Blocksearch ausgeführt wurde.
Rückwärtsfahren	
ItpEnableFeederBackup [▶ 262]	Aktiviert die Backup-Liste für das Rückwärtsfahren
ItpIsFeederBackupEnabled [▶ 263]	Liest aus, ob die Backup-Liste zum Rückwärtsfahren aktiv ist
ItpIsFirstSegmentReached [▶ 265]	Liest aus, ob beim Rückwärtsfahren die Startposition erreicht ist
ItpIsFeedFromBackupList [▶ 264]	Liest aus, ob Feeder-Einträge aus der Backupliste gesendet werden
ItpIsMovingBackwards [▶ 265]	Liest aus, ob rückwärts auf der aktuellen Bahn verfahren wird
ItpRetraceMoveBackward [▶ 266]	Führt eine Rückwärtsbewegung auf der Bahn aus
ItpRetraceMoveForward [▶ 267]	Führt eine Vorwärtsbewegung auf der Bahn aus, wird aufgerufen um das Rückwärtsfahren abzubrechen.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.1 ItpConfirmHsk



Der Funktionsbaustein ItpConfirmHsk bestätigt die aktuell anliegende M-Funktion.

Wenn der Kanaloverride auf 0 gesetzt oder ein E-Stop aktiv ist, werden für diese Zeit keine M-Funktionen bestätigt. Somit bleibt das Busy-Signal von ItpConfirmHsk anliegen und muss weiterhin aufgerufen werden.

VAR_INPUT

```
VAR_INPUT
    bExecute  : BOOL;
END_VAR
```

bExecute: durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc : NCTOPLC_NCICHANNEL_REF;
    sPlcToNci : PLCTONC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

sPlcToNci: Struktur des zyklischen Kanalinterfaces von der SPS zur NCI. (Typ: PLCTONC_NCICHANNEL_REF [▶ 337])

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy     : BOOL;
    bErr      : BOOL;
    nErrId    : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.2 ItpDelDtgEx

Der Baustein ItpDelDtgEx triggert das Restweglöschen. Eine ausführlichere Beschreibung ist in der Interpreter [▶ 162] Dokumentation zu finden.

VAR_INPUT

```
VAR_INPUT
    bExecute  : BOOL;
    tTimeOut  : TIME;
END_VAR
```

bExecute: durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy     : BOOL;
    bErr      : BOOL;
    nErrId    : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.3 ItpEnableDefaultGCode

Der Funktionsbaustein ItpEnableDefaultGCode ermöglicht es, aus der SPS einen vom Anwender definierten G-Code vor dem Start jedes NC-Programms auszuführen. Das Defaultprogramm wird beim Start des eigentlichen NC-Programms vor dem geladenen Programm ausgeführt.

Mit diesem Funktionsbaustein ist es zum Beispiel möglich, das Koordinatensystem für alle auszuführenden NC-Programme zu drehen.

Der Standard-G-Code muss als „DefaultGCode<Channel-Number>.def“ im TwinCAT\Mc\Nci Verzeichnis gespeichert werden.

VAR_INPUT

```
VAR_INPUT
    bExecute            : BOOL;
    bUseDefaultGCode    : BOOL;
    tTimeOut            : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt

bUseDefaultGCode: Ist diese Variable TRUE, wird durch eine steigende Flanke an bExecute der Default G-Code aktiviert. Ist die Variable FALSE, wird der Default G-Code deaktiviert.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy     : BOOL;
    bErr      : BOOL;
    nErrId    : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Nicht verfügbar für GST

Dieser Funktionsbaustein ist nicht verfügbar, wenn der GST-Interpreter verwendet wird.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.4 ItpEStopEx

Der Baustein ItpEStopEx triggert den NCI EStop und ermöglicht so ein kontrolliertes Anhalten auf der Bahn. Dabei werden die Grenzwerte für die Verzögerung und den Ruck als Parameter übertragen. Falls diese kleiner sein sollten, als die z.Zt. wirkenden Dynamikparameter, so werden die übertragenen Parameter verworfen.

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    fDec            : LREAL;
```

```
    fJerk           : LREAL;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt

fDec: max. Verzögerung mit der angehalten werden soll. Ist fDec kleiner als die z.Zt. aktive Verzögerung, so wird fDec nicht übernommen. Es wird so also sichergestellt, dass mindestens mit der Standard-Rampe verzögert wird.

fJerk: max. Ruck mit dem angehalten werden soll. Ist fJerk kleiner als der z.Zt. akive Ruck, so wird fJerk nicht übernommen.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy     : BOOL;
    bErr      : BOOL;
    nErrId    : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Siehe auch ItpStepOnAfterEStopEx [▶ 252].

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.5 ItpGetBlockNumber

ItpGetBlockNumber ist eine Funktion, die die Blocknummer des NC-Programms des zyklischen Interfaces zurückliefert.

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

Rückgabewert

ItpGetBlockNumber: Blocknummer des aktiven Geometriesegments

Beispiel

```
VAR
    nBlockNumber        : UDINT;
    sNciToPlc AT%I*     : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

```
nBlockNumber := ItpGetBlockNumber(sNciToPlc);
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.6 ItpGetBottleNeckLookAheadEx

Der Baustein ItpGetBottleNeckLookAheadEx ermittelt die maximale verwendete Größe des LookAheads für die Falschenhalserkennung (Kontur-Kollisions-Überwachung).

Eine weitere Beschreibung ist in der Interpreter [▶ 196] Dokumentation zu finden.

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy           : BOOL;
    bErr            : BOOL;
    nErrId          : UDINT;
    nLookAhead      : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der, an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wenn der Baustein einen Timeout-Fehler hat, so ist 'Error' = TRUE und 'nErrId' = 1861 (Hexadezimal 0x745). Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

nLookAhead: Größe des Look-Aheads für die Flaschenhalserkennung

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.7 ItpGetBottleNeckModeEx

Der Baustein ItpGetBottleNeckModeEx liest die Verhaltensweise bei einer auftretenden Kontur-Kollision (Flaschenhals) aus.

Eine weitere Beschreibung ist in der Interpreter [▶ 196] Dokumentation zu finden.

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy           : BOOL;
    bErr            : BOOL;
    nErrId          : UDINT;
    eBottleNeckMode : E_ItpBnMode
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der, an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

eBottleNeckMode: Enum für die Verhaltensweise bei einer auftretenden Kontur-Kollision

```
TYPE E_ItpBnMode:
(
    ItpBnm_Abort := 0,
    ItpBnm_Adjust := 1,
    ItpBnm_Leave := 2
);
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.8 ItpGetChannelId

ItpGetChannelId ist eine Funktion, die aus dem zyklischen Interface die Kanal ID ermittelt.

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

Rückgabewert

ItpGetChannelId: Kanal ID (Typ: UDINT)

Beispiel

```
VAR
    nChnId         : UINT;
    sNciToPlc AT%I*: NCTOPLC_NCICHANNEL_REF;
END_VAR
```

```
nChnId := ItpGetChannelId( sNciToPlc );
```

siehe auch: ItpGetGroupId [▶ 225]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.9 ItpGetChannelType

ItpGetChannelType ist eine Function, die den Kanaltyp des zyklischen Interfaces zurückliefert.

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

Rückgabewert

ItpGetChannelType: Kanaltyp (Typ: E_ItpChannelType)

```
TYPE E_ItpChannelType :
(
    ItpChannelTypeNone,
    ItpChannelTypeInterpreter,
    ItpChannelTypeKinematic,
    ItpChannelType_InvalidItfVer := 16#4B14 (*ErrTcNciItp_ItfVersion the cyclic channel interface do
es not match to the requested function/fb *)
);
END_TYPE
```

Beispiel

```
VAR
    nChannelType    : E_ItpChannelType;
    sNciToPlc AT%I* : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

```
nChannelType := ItpGetChannelType( sNciToPlc );
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.10 ItpGetCyclicLrealOffsets

Mit dem Funktionsbaustein ItpGetCyclicLRealOffsets wird die aktuelle Konfiguration des zyklischen Kanalinterface für LREAL Variablen ausgelesen.

VAR_INPUT

```
VAR_INPUT
    bExecute            : BOOL;
    tTimeOut            : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy              : BOOL;
    bErr               : BOOL;
    nErrId             : UDINT;
    nIndexOffsetParam1 : UDINT;
    nIndexOffsetParam2 : UDINT;
    nIndexOffsetParam3 : UDINT;
    nIndexOffsetParam4 : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der, an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

nIndexOffsetParam1: Gruppenzustand (Index Offset) für Parameter 1

nIndexOffsetParam2: Gruppenzustand (Index Offset) für Parameter 2

nIndexOffsetParam3: Gruppenzustand (Index Offset) für Parameter 3

nIndexOffsetParam4: Gruppenzustand (Index Offset) für Parameter 4

Siehe auch:

- ItpReadCyclicLRealParam1 [▶ 235]
- ItpSetCyclicLRealOffsets [▶ 243]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.11 ItpGetCyclicUDintOffsets

Mit dem Funktionsbaustein ItpGetCyclicUDintOffsets wird die aktuelle Konfiguration des zyklischen Kanalinterface für UDINT Variablen ausgelesen.

VAR_INPUT

```
VAR_INPUT
    bExecute                : BOOL;
    tTimeOut                : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy               : BOOL;
    bErr                : BOOL;
    nErrId              : UDINT;
    nIndexOffsetParam1  : UDINT;
    nIndexOffsetParam2  : UDINT;
    nIndexOffsetParam3  : UDINT;
    nIndexOffsetParam4  : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der, an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

nIndexOffsetParam1: Gruppenzustand (Index Offset) für Parameter 1

nIndexOffsetParam2: Gruppenzustand (Index Offset) für Parameter 2

nIndexOffsetParam3: Gruppenzustand (Index Offset) für Parameter 3

nIndexOffsetParam4: Gruppenzustand (Index Offset) für Parameter 4

Siehe auch:

- ItpReadCyclicUDintParam1 [▶ 235]
- ItpSetCyclicUdintOffsets [▶ 245]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.12 ItpGetError

ItpGetError ist eine Funktion, die die Fehlernummer zurückliefert. Eine Beschreibung der NC Fehlercodes ist hier zu finden.

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

Rückgabewert

ItpGetError: Fehlernummer

ItpGetError wertet aus dem zyklischen Interface die Variable 'nItpErrCode' aus.

Beispiel

```
VAR
    bItpError       : BOOL;
    nErrId          : UDINT;
    sNciToPlc AT%I*: NCTOPLC_NCICHANNEL_REF;
END_VAR
```

```
bItpError := ItpHasError( sNciToPlc );
IF bItpError THEN
    nErrId := ItpGetError( sNciToPlc );
    …
END_IF
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.13 ItpGetGeoInfoAndHParamEx

Der Funktionsbaustein `ItpGetGeoInfoAndHParamEx` liest Informationen über das gerade aktive Segment und vergangene und zukünftige Segmente aus. Hierzu gehören Blocknummer, H-Parameter und Bahnrestweg auf dem Segment.

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

VAR_OUTPUT

```
VAR_OUTPUT
    stTab               : ST_ItpPreViewTabEx;
    nErrId              : UDINT;
END_VAR
```

stTab: Struktur, die die Segmentdaten enthält. Siehe ST_ItpPreViewTabEx [▶ 224].

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in `nErrId` können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

```
TYPE ST_ItpPreViewTabEx :
STRUCT
    nDcTime               : UDINT := 0
    nReserved             : UDINT := 0;
    arrLines              : ARRAY[1..NCI_MAX_PREVIEWTABLINES] OF ST_ItpPreViewTabLine;
END_STRUCT
END_TYPE
```

nDcTime: Aktueller Zeitstempel in ns. Dieser Zeitstempel kann z. B. im Zusammenspiel mit der `Tc2_NciXFC` Bibliothek verwendet werden.

arrLines: Array von segmentbezogenen Informationen (Größe 20). Der Eintrag an Stelle 11 des Arrays entspricht dem gerade aktiven Segment. An Stelle 1-10 des Arrays werden bereits bearbeitete Segmente angezeigt, an Stelle 12-20 zukünftige Segmente. Siehe ST_ItpPreViewTabLine [▶ 224].

```
TYPE ST_ItpPreViewTabLine :
STRUCT
    fLength               : LREAL := 0.0;
    nBlockNo              : UDINT := 0;
    nHParam               : UDINT := 0;
    nEntryID              : UDINT := 0;
    nReserved             : UDINT := 0;
END_STRUCT
END_TYPE
```

fLength: Verbleibende Segmentlänge. Für Segmente, die noch nicht aktiv sind, entspricht dies der Gesamtsegmentlänge. Für vergangene Segmente wird die verfahrene Distanz seit dem Segmentende angegeben.

nBlockNo: vom Anwender programmierte Blocknummer

nHParam: Wert des H-Parameters [▶ 172], der ab Beginn des nächsten Segments aktiv ist

nEntryID: vom System generierte Kommando-ID

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.14 ItpGetGroupAxisIds

ItpGetGroupAxisIds ist eine Funktion, die ein Array von Achsen-IDs liefert, die für die Gruppe konfiguriert worden sind.

VAR_IN_OUT

```
FUNCTION ItpGetGroupAxisIds

VAR_IN_OUT
    sNciToPlc AT%I*  : NCTOPLC_NCICHANNEL_REF;
    nNciAxisIds      : ARRAY[1..8] OF DWORD;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ NCTOPLC_NCICHANNEL_REF [▶ 335])

sNciAxisIds: Array der Achsen-IDs

Rückgabewert

ItpGetGroupAxisIds: Fehlernummer

> ItpGetGroupAxisIds wertet die Informationen der Variable 'nAcsAxisIDs[8]' aus dem zyklischen Interface aus.

Beispiel

```
VAR
    nNciAxisIds     : ARRAY[1..8] OF DWORD;
    sNciToPlc AT%I* : NCTOPLC_NCICHANNEL_REF;
    nVersionErr     : DWORD;
END_VAR
```

```
nVersionErr := ItpGetGroupAxisIds(nNciAxisIds, sNciToPlc );
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.15 ItpGetGroupId

ItpGetGroupId ist eine Funktion, die aus dem zyklischen Interface die Group ID ermittelt.

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

Rückgabewert

ItpGetGroupId: Group ID

Beispiel

```
VAR
    nGrpId         : UINT;
    sNciToPlc AT%I*: NCTOPLC_NCICHANNEL_REF;
END_VAR
```

```
nGrpId := ItpGetGroupId( sNciToPlc );
```

Siehe auch: ItpGetChannelId [▶ 219]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.16 ItpGetHParam

ItpGetHParam ist eine Funktion, die den aktuellen H-Parameter zurückliefert.

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

Rückgabewert

ItpGetHParam: H-Parameter

> ItpGetHParam wertet aus dem zyklischen Interface die Variable 'nHFuncValue' aus.

Beispiel

```
VAR
    nHParam        : DINT;
    sNciToPlc AT%I*: NCTOPLC_NCICHANNEL_REF;
END_VAR
nHParam := ItpGetHParam( sNciToPlc );
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.17 ItpGetHskMFunc

ItpGetHskMFunc liefert die Nummer der M-Funkion vom Typ Handshake.

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

Rückgabewert

ItpGetHskMFunc: Nummer der M-Funktion

> ItpGetHskMFunc wertet aus dem zyklischen Interface die Variable 'nHskMFuncNo' aus.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.18 ItpGetItfVersion

ItpGetItfVersion ist eine Funktion, die die Versionsnummer des zyklischen Interfaces ermittelt.

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

Rückgabewert

ItpGetItfVersion: Versionsnummer des zyklischen Interfaces

Beispiel

```
VAR
    nItfVer        : UINT;
    sNciToPlc AT%I*: NCTOPLC_NCICHANNEL_REF;
END_VAR
```

```
nItfVer := ItpGetItfVersion( sNciToPlc );
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.19 ItpGetOverridePercent

Die Funktion ItpGetOverridePercent liefert den Achsen-Kanal-Override in Prozent. Dabei muss unbedingt beachtet werden, dass es sich hierbei nicht um einen Wert von der NC handelt. Es wird der Wert ausgewertet, der sollwertseitig an die NC weitergegeben wird.

VAR_IN_OUT

```
VAR_IN_OUT
    sPlcToNci        : PLCTONC_NCICHANNEL_REF;
END_VAR
```

sPlcToNci: Struktur des zyklischen Kanalinterfaces von der SPS zur NCI (Typ: PLCTONC_NCICHANNEL_REF [▶ 337])

Rückgabewert

ItpGetOverridePercent: Override in Prozent

Beispiel

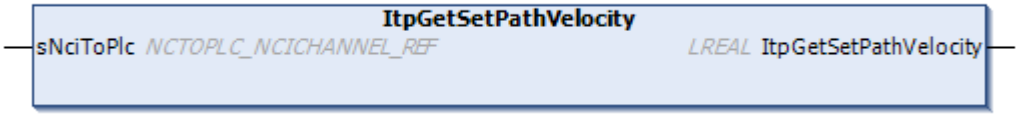
```
VAR
    sPlcToNci AT%Q*: PLCTONC_NCICHANNEL_REF;
    fOverride      : LREAL;
END_VAR
```

```
fOverride := ItpGetOverridePercent( sPlcToNci );
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.20 ItpGetSetPathVelocity



ItpGetSetPathVelocity ist eine Funktion, die aus dem zyklischen Interface die aktuelle Bahnsollgeschwindigkeit ausliest.

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

Rückgabewert

ItpGetSetPathVelocity: Momentane Bahnsollgeschwindigkeit

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.21 ItpGetSParam

ItpGetSParam ist eine Funktion, die den aktuellen S-Parameter zurückliefert.

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

Rückgabewert

ItpGetSParam: S-Parameter

ItpGetSParam wertet aus dem zyklischen Interface die Variable 'nSpindleRpm' aus.

Beispiel

```
VAR
    nSParam         : UINT;
    sNciToPlc AT%I*: NCTOPLC_NCICHANNEL_REF;
END_VAR
```

```
nSParam := ItpGetSParam( sNciToPlc );
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.22 ItpGetStateInterpreter

ItpGetStateInterpreter ist eine Funktion, die den Interpreter Status zurückliefert.

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

Rückgabewert

ItpGetStateInterpreter: Aktueller Status des Interpreters [▶ 15]

> ItpGetStateInterrpreter wertet aus dem zyklischen Interface die Variable 'nItpState' aus.

Beispiel

```
VAR
    nItpState      : UDINT;
    sNciToPlc AT%I*: NCTOPLC_NCICHANNEL_REF;
END_VAR
```

```
nItpState := ItpGetStateInterpreter( sNciToPlc );
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.23 ItpGetTParam

ItpGetTParam ist eine Funktion, die den aktuellen T-Parameter zurückliefert.

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

Rückgabewert

ItpGetTParam: T-Parameter

ItpGetTParam wertet aus dem zyklischen Interface die Variable 'nTool' aus.

Beispiel

```
VAR
    nTParam        : UINT;
    sNciToPlc AT%I*: NCTOPLC_NCICHANNEL_REF;
END_VAR
```

```
nTParam := ItpGetTParam( sNciToPlc );
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.24 ItpGoAheadEx

Der Baustein ItpGoAheadEx darf nur in Verbindung mit dem Dekodierstopp '@717' [▶ 172] verwendet werden. Eine ausführlichere Beschreibung dieses Dekodierstopps ist in der Interpreter-Dokumentation [▶ 128] zu finden.

VAR_INPUT

```
VAR_INPUT
    bExecute       : BOOL;
    tTimeOut       : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy     : BOOL;
    bErr      : BOOL;
    nErrId    : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.25 ItpHasError

ItpHasError ist eine Funktion, die ermittelt, ob der Interpreter im Fehlerzustand ist.

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

Rückgabewert

Die Funktion liefert im Fehlerfall ein TRUE.

> ItpHasError wertet aus dem zyklischen Interface die Variable 'nItpErrCode' aus. Ist dieser Wert ungleich 0, so wird TRUE zurückgegeben.

Beispiel

```
VAR
    bItpError      : BOOL;
    sNciToPlc AT%I*: NCTOPLC_NCICHANNEL_REF;
END_VAR
```

```
bItpError := ItpHasError( sNciToPlc );
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.26 ItpIsFastMFunc

ItpIsFastMFunc ist eine Funktion, die ermittelt, ob bei der mitgegebenen M-Funktionsnummer die schnelle M-Funktion gesetzt ist.

VAR_IN

```
FUNCTION ItpIsFastMFunc

VAR_IN
    nFastMFuncNo    : INT;
END_VAR
```

nFastMFuncNo: Nummer der M-Funktion, die überprüft werden soll.

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

Rückgabewert

Die Funktion liefert TRUE, wenn das schnelle Bit der M-Funktion gesetzt ist.

ItpIsFastMFunc wertet aus dem zyklischen Interface die Variable 'nFastMFuncMask' aus.

Beispiel

```
(*this enum is defined by the user *)

TYPE FastMFuncs:
(
    M10_CoolingFluidOn := 10, (*fast M-Funktion M10*)
    M11_CoolingFluidOff := 11,
    M12_FanOn := 12,
    M13_FanOff := 13
);
END_TYPE
```

```
VAR
    sNciToPlc AT%I*: NCTOPLC_NCICHANNEL_REF
    enFastMFuncs   : FastMFuncs;
     bTurnFanOn    : BOOL;
END_VAR
```

```
bTurnFanOn := ItpIsFastMFunc( M12_FanOn,sNciToPlc );
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.27 ItpIsEStopEx

Die Funktion ItpIsEStopEx liefert die Information, ob ein EStop-Kommando angestoßen wurde.

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

Rückgabewert

Ist der Rückgabewert TRUE, dann wurde zuvor ein EStop ausgeführt (z.B. ItpEStopEx). Dabei liefert das Flag **keine** Information darüber, ob die Achsen bereits stehen oder sich noch auf der Bremsrampe befinden.

Nach der Ausführung von ItpStepOnAfterEStopEx, liefert ItpIsEStopEx wieder ein FALSE zurück.

ItpIsEStopEx wertet das zyklische Interface aus.

siehe auch:

ItpEStopEx [▶ 215]

ItpStepOnAfterEStopEx [▶ 252]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.28 ItpIsHskMFunc

ItpIsHskMFunc ermittelt, ob eine M-Funktion vom Typ Handshake anliegt.

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

Rückgabewert

Die Funktion liefert TRUE, wenn eine M-Funktion vom Typ Handshake ansteht.

ItpIsHskFunc wertet aus dem zyklischen Interface die Variable 'nHskMFuncReq' aus.

Beispiel

```
VAR
    bMFuncRequest   : BOOL;
    sNciToPlc AT%I* : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

```
bMFuncRequest := ItpIsHskMFunc( sNciToPlc );
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.29 ItpLoadProgEx

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    sPrg            : STRING(255);
    nLength         : UDINT;
    tTimeOut        : TIME;
END_VAR
```

bExecute: bei einer steigenden Flanke lädt der Baustein das NC-Programm

sPrg: Name des NC-Programms das geladen wird

nLength: Stringlänge des Programmnamens

tTimeOut: ADS Timeout-Delay

Das NC-Programm wird im Verzeichnis "TwinCAT\Mc\Nci " gesucht, wenn keine weiteren Angaben gemacht werden. Es kann jedoch auch ein absoluter Pfad angegeben werden.

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy     : BOOL;
    bErr      : BOOL;
    nErrId    : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Beispiel

```
VAR
    in_stItpToPlc AT %I*          : NCTOPLC_NCICHANNEL_REF;
    fbLoadProg                    : ItpLoadProgEx;
    sProgramPath                  : STRING (255):= 'TestIt.nc';
END_VAR
```

```
fbLoadProg(
    bExecute := TRUE,
    sPrg := sProgramPath,
    nLength := LEN(sProgramPath),
    tTimeOut := t#200ms,
    sNciToPlc := in_stItpToPlc
);
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.30 ItpReadCyclicLRealParam1

Diese Funktion liest aus dem zyklischen Kanalinterface den ersten LREAL Parameter aus. Dieser Parameter wird zuvor mit ItpSetCyclicLRealOffsets [▶ 243] konfiguriert.

Die Parameter 2 bis 4 werden über den gleichen Mechanismus ausgelesen (z.B. ItpReadCyclicLRealParam2).

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

Rückgabewert

Parameter 1 vom Type LREAL.

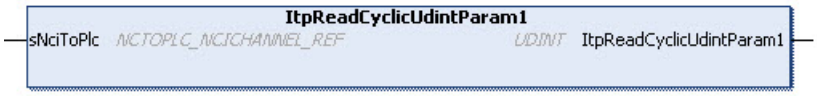
Siehe auch:

- ItpReadCyclicUdintParam1 [▶ 235]
- ItpSetCyclicLRealOffsets [▶ 243]
- ItpGetCyclicLRealOffsets [▶ 220]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.31 ItpReadCyclicUdintParam1



Diese Funktion liest aus dem zyklischen Kanalinterface den ersten UDINT Parameter aus. Dieser Parameter wird zuvor mit ItpSetCyclicUdintOffsets [▶ 245] konfiguriert.

Die Parameter 2 bis 4 werden über den gleichen Mechanismus ausgelesen (z.B. ItpReadCyclicUdintParam2).

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

Rückgabewert

Parameter 1 vom Type UDINT

Siehe auch:

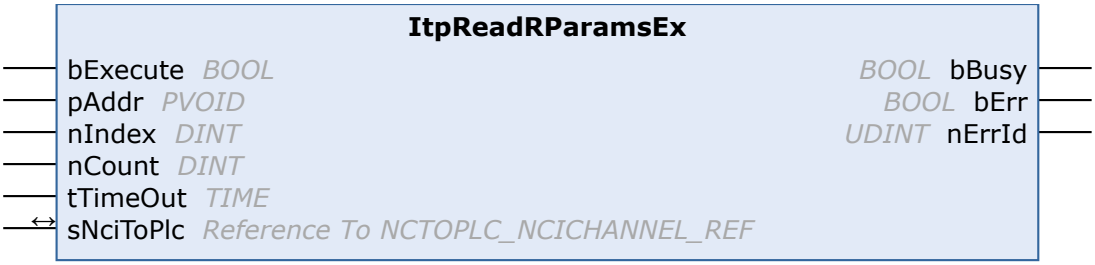
- ItpReadCyclicLRealParam1 [▶ 235]
- ItpSetCyclicUdintOffsets [▶ 245]

- ItpGetCyclicUdintOffsets [▶ 221]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.32 ItpReadRParamsEx



Der Baustein ItpReadRParamsEx liest Rechenparameter, kurz R-Parameter, der NC. Eine genaue Beschreibung der Rechenparameter ist hier [▶ 136] zu finden. Insgesamt stehen 1000 R-Parameter zur Verfügung, wovon die ersten 900 (0..899) lokal, d.h. nur im aktuellen NC-Kanal, sichtbar sind. Die letzten 100 (900..999) R-Parameter sind global und somit NC-weit sichtbar.

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    pAddr           : PVOID;
    nIndex          : DINT;
    nCount          : DINT;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Mit einer steigenden Flanke wird der Lesevorgang gestartet.

pAddr: Adresse der Zielvariablen der zu lesenden Daten. Dabei werden die Daten direkt ab der angegebenen Adresse beschrieben. D.h. nIndex ist nicht als Offset zu pAddr zu sehen. Die Daten werden gewöhnlich in einem Array vom Typ LREAL abgelegt, das vom Anwender definiert werden muss.

nIndex: Beschreibt den Index des R-Parameters der aus NC Sicht gelesen werden soll.

nCount: Anzahl der zu lesenden R-Parameter

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bErr        : BOOL;
    nErrId      : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.33 ItpReadToolDescEx

Der Baustein ItpReadToolDescEx liest für das mitgegebene D-Wort die Werkzeugparameter.

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    nDNo            : UDINT;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt.

nDNo: D-Wort für das die Werkzeugparameter ausgelesen werden sollen. nDNo kann Werte von 1 bis einschließlich 255 annehmen.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc       : NCTOPLC_NCICHANNEL_REF;
    sToolDesc       : ToolDesc;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen.

sToolDesc: Struktur, in der die Werkzeugparameter von nDNo geschrieben werden. Die Bedeutung der Parameter ist vom Werkzeugtyp abhängig und kann den Werkzeugdaten [▶ 184] entnommen werden. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bErr        : BOOL;
    nErrId      : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

```
TYPE ToolDesc:
STRUCT
    nToolNumber     : UDINT; (*valid range from 0 .. 65535*)
    nToolType       : UDINT;
    fParam          : ARRAY [2..15] OF LREAL;
END_STRUCT
END_TYPE
```

Siehe auch:

ItpWriteToolDescEx [▶ 254]

ItpSetToolDescNullEx [▶ 248]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.34 ItpReadZeroShiftEx

Der Baustein ItpReadZeroShiftEx liest für die angegebene Nullpunktverschiebung die Verschiebungskomponenten X, Y und Z.

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    nZsNo           : UDINT;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt.

nZsNo: Nummer der Nullpunktverschiebung. NC-seitig sind G54 bis G59 Nullpunktverschiebungen. Der gültige Wertebereich für 'nZsNo' ist deshalb von 54 bis 59.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc       : NCITOPLC_NCICHANNEL_REF;
    sZeroShiftDesc  : ZeroShiftDesc;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Type: NCTOPLC_NCICHANNEL_REF [▶ 335])

sZeroShiftDesc: Struktur mit den Komponenten der Nullpunktverschiebung.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy     : BOOL;
    bErr      : BOOL;
    nErrId    : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

```
TYPE ZeroShiftDesc:
STRUCT
    fShiftX         : LREAL;
    fShiftY         : LREAL;
    fShiftZ         : LREAL;
END_STRUCT
END_TYPE
```

Aus Kompatibilitätsgründen gibt es pro Nullpunktverschiebung (z.B. G54) für jede Achse zwei Einträge (grob & fein), die addiert werden. Dieser Funktionsbaustein wertet beide Einträge aus und addiert sie automatisch.

Siehe auch:

ItpWriteZeroShiftEx [▶ 256]

ItpSetZeroShiftNullEx [▶ 249]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.35 ItpResetEx2

Der Baustein 'ItpResetEx2' führt einen Kanal-Reset aus und löscht damit alle vorhandenen Tabellen des NC-Kanals. Im Gegensatz zum veralteten Baustein ItpReset wird ein aktiver Kanal erst gestoppt, bevor der Reset ausgeführt wird. Damit wird die Programmierung in der SPS vereinfacht, da nicht explizit überprüft werden muss, ob die Achsen noch in Bewegung sind.

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt.

tTimeOut: ADS Timeout-Delay (das bBusy-Signal kann länger anliegen als tTimeOut)

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy     : BOOL;
    bErr      : BOOL;
    nErrId    : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.36 ItpResetFastMFuncEx

Mit einer steigenden Flanke am Eingang bExecute wird die schnelle M-Funktion [▶ 167] nMFuncNo zurückgesetzt. Für den Fall, dass die M-Funktion nicht anliegt, wird **kein** Fehler zurückgegeben.

Dieser Funktionsbaustein stellt eine Alternative zum Autoreset bzw. dem Zurücksetzen mit einer anderen M-Funktion dar (Resetliste bei der Parametrierung der M-Funktion). Aus Gründen der Übersichtlichkeit sollte ein Mischbetrieb zwischen dem Zurücksetzen mit einer M-Funktion und diesem Funktionsbaustein vermieden werden.

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    nMFuncNo        : UINT;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt.

nMFuncNo: Fliegende M-Funktion, die zurückgesetzt werden soll

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy     : BOOL;
    bErr      : BOOL;
    nErrId    : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

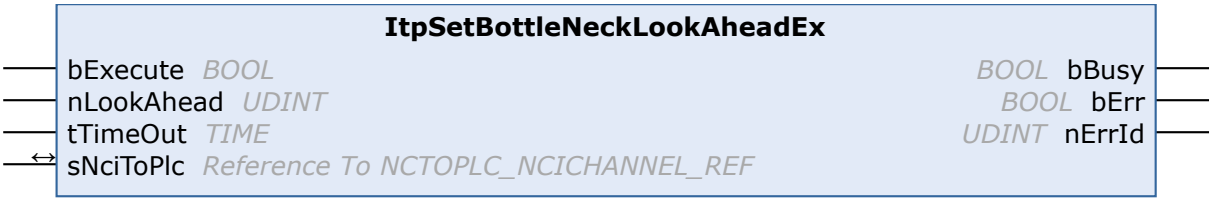
bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.37 ItpSetBottleNeckLookAheadEx



Der Baustein ItpSetBottleNeckLookAheadEx legt fest, wie viel Segmente maximal für die Flaschenhalserkennung (Kontur-Kollisions-Überwachung) in die Zukunft geschaut darf. Dabei ist zu beachten, dass Segmente, die aufgrund der Radiuskorrektur eingefügt wurden (z.B. Zusatzsegmente an spitzen Winkeln) mit berücksichtigt werden.

Eine weitere Beschreibung ist in der Interpreter [▶ 196]-Dokumentation zu finden.

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    nLookAhead      : UDINT;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt.

nLookAhead: Legt die Größe des Look-Aheads fest

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy     : BOOL;
    bErr      : BOOL;
    nErrId    : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

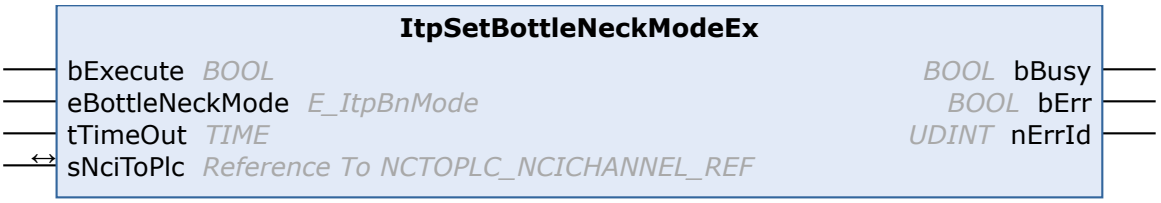
Nicht verfügbar für GST

Dieser Funktionsbaustein ist nicht verfügbar, wenn der GST-Interpreter verwendet wird.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.38 ItpSetBottleNeckModeEx



Der Baustein ItpSetBottleNeckModeEx legt die Verhaltensweise bei einer auftretenden Kontur-Kollision (Flaschenhals) fest.

Eine weitere Beschreibung ist in der Interpreter [▶ 196]-Dokumentation zu finden.

VAR_INPUT

```
VAR_INPUT
    bExecute       : BOOL;
    eBottleNeckMode: E_ItpBnMode
    tTimeOut       : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt.

eBottleNeckMode: Enum für die Verhaltensweise bei einer auftretenden Kontur-Kollision

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy     : BOOL;
    bErr      : BOOL;
    nErrId    : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

```
TYPE E_ItpBnMode:
(
    ItpBnm_Abort    := 0,
    ItpBnm_Adjust   := 1,
    ItpBnm_Leave    := 2
);
END_TYPE
```

Nicht verfügbar für GST

Dieser Funktionsbaustein ist nicht verfügbar, wenn der GST-Interpreter verwendet wird.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.39 ItpSetCyclicLrealOffsets

Mit dem Funktionsbaustein ItpSetCyclicLrealOffsets wird das zyklische Kanalinterface für die 4 frei konfigurierbaren LREAL Variablen beschrieben. Dabei können Variablen (Index Offsets) aus dem Gruppenzustand ausgewählt werden.

Die Funktionalität ist nur dann aktiv, wenn nIndexOffsetParam1 ungleich 0 ist.

VAR_INPUT

```
VAR_INPUT
    bExecute           : BOOL;
    tTimeOut           : TIME;
    nIndexOffsetParam1 : UDINT;
    nIndexOffsetParam2 : UDINT;
    nIndexOffsetParam3 : UDINT;
    nIndexOffsetParam4 : UDINT;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt.

tTimeOut: ADS Timeout-Delay

nIndexOffsetParam1: Gruppenzustand (Index Offset) für Parameter 1

nIndexOffsetParam2: Gruppenzustand (Index Offset) für Parameter 2

nIndexOffsetParam3: Gruppenzustand (Index Offset) für Parameter 3

nIndexOffsetParam4: Gruppenzustand (Index Offset) für Parameter 4

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc          : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy     : BOOL;
    bErr      : BOOL;
    nErrId    : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

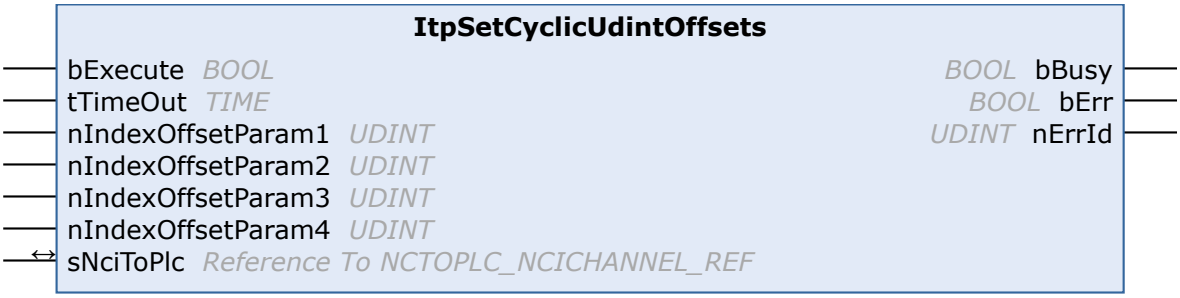
Siehe auch:

- ItpReadCyclicLRealParam1 [▶ 235]
- ItpGetCyclicLRealOffsets [▶ 220]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.40 ItpSetCyclicUDintOffsets



Mit dem Funktionsbaustein ItpSetCyclicUDintOffsets wird das zyklische Kanalinterface für die 4 frei konfigurierbaren UDINT Variablen beschrieben. Dabei können Variablen (Index Offsets) aus dem Gruppenzustand ausgewählt werden.

Die Funktionalität ist nur dann aktiv, wenn nIndexOffsetParam1 ungleich 0 ist.

VAR_INPUT

```
VAR_INPUT
    bExecute           : BOOL;
    tTimeOut           : TIME;
    nIndexOffsetParam1 : UDINT;
    nIndexOffsetParam2 : UDINT;
    nIndexOffsetParam3 : UDINT;
    nIndexOffsetParam4 : UDINT;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt.

tTimeOut: ADS Timeout-Delay

nIndexOffsetParam1: Gruppenzustand (Index Offset) für Parameter 1

nIndexOffsetParam2: Gruppenzustand (Index Offset) für Parameter 2

nIndexOffsetParam3: Gruppenzustand (Index Offset) für Parameter 3

nIndexOffsetParam4: Gruppenzustand (Index Offset) für Parameter 4

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc          : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy      : BOOL;
    bErr       : BOOL;
    nErrId     : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Siehe auch:

- ItpReadCyclicUDintParam1 [▶ 235]
- ItpGetCyclicUdintOffsets [▶ 221]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.41 ItpSetOverridePercent

Die Funktion ItpSetOverridePercent schreibt den Achsen-Kanal-Override in das zyklische Interface zur NCI. Dabei wird der Override in Prozent übergeben.

VAR_INPUT

```
FUNCTION ItpSetOverridePercent

VAR_INPUT
    fOverridePercent : LREAL;
END_VAR
```

fOverridePercent: Achsen-Kanal-Override in Prozent

VAR_IN_OUT

```
VAR_IN_OUT
    sPlcToNci        : PLCTONC_NCICHANNEL_REF;
END_VAR
```

sPlcToNci: Struktur des zyklischen Kanalinterfaces von der SPS zur NCI (Typ: PLCTONC_NCICHANNEL_REF [▶ 337])

Rückgabewert

ItpSetOverridePercent: immer True

Beispiel

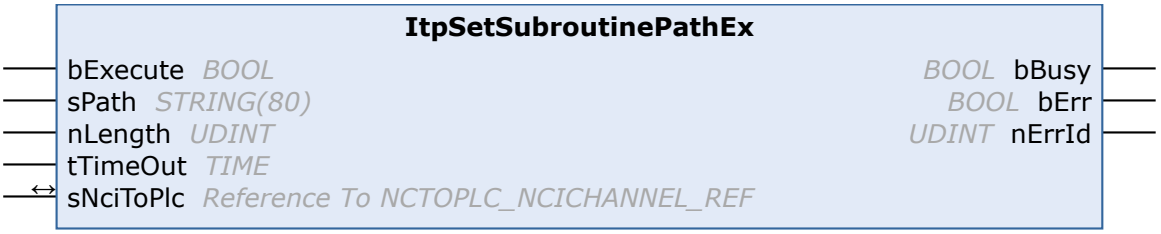
```
VAR
    sPlcToNci AT%Q*: PLCTONC_NCICHANNEL_REF;
    fOverride      : LREAL;
END_VAR
```

```
fOverride := 47.11;
ItpSetOverridePercent( fOverride, sPlcToNci );
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.42 ItpSetSubroutinePathEx



Mit dem Baustein ItpSetSubroutinePathEx wird optional der Suchpfad für Unterprogramme gesetzt.

Wenn ein Unterprogramm noch eingebunden werden muss, wird in folgender Reihenfolge nach der Datei gesucht:

1. optionaler Suchpfad (ItpSetSubroutinePath)
2. Pfad aus dem das Hauptprogramm geladen wurde
3. TwinCAT\Mc\Nci -Verzeichnis

Es kann immer nur ein optionaler Pfad wirken und dieser bleibt bestehen, bis er mit einem anderen Pfad oder mit einem Leerstring überschrieben wird.

Nach einem TwinCAT-Restart muss der Pfad neu zugewiesen werden.

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    sPath           : STRING;
    nLength         : UDINT;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt.

sPath: optionaler Pfad für Unterprogramme. Wird mit einem Leerstring deaktiviert

nLength: Stringlänge

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy     : BOOL;
    bErr      : BOOL;
    nErrId    : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Nicht verfügbar für GST

Dieser Funktionsbaustein ist nicht verfügbar, wenn der GST-Interpreter verwendet wird.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.43 ItpSetToolDescNullEx

Der FB ItpSetToolDescNullEx überschreibt alle Werkzeugparameter (inkl. Nummer & Type) des Kanals mit Null.

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Mit einer steigenden Flanke werden alle Werkzeugparameter des NC-Kanals mit Null überschrieben.

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy      : BOOL;
    bErr       : BOOL;
    nErrId     : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Siehe auch:

ItpWriteToolDescEx [▶ 254]

ItpReadToolDescEx [▶ 237]

Nicht verfügbar für GST

Dieser Funktionsbaustein ist nicht verfügbar, wenn der GST-Interpreter verwendet wird.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.44 ItpSetZeroShiftNullEx

Der Funktionsbaustein ItpSetZeroShiftNullEx überschreibt alle Nullpunktverschiebungen des Kanals mit Null.

VAR_INPUT

```
VAR_INPUT
    bExecute       : BOOL;
    tTimeOut       : TIME;
END_VAR
```

bExecute: Mit einer steigenden Flanke werden alle Nullpunktverschiebungen des NC-Kanals mit Null überschrieben.

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy     : BOOL;
    bErr      : BOOL;
    nErrId    : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Siehe auch:

- ItpWriteZeroShiftEx [▶ 256],
- ItpReadZeroShiftEx [▶ 238].

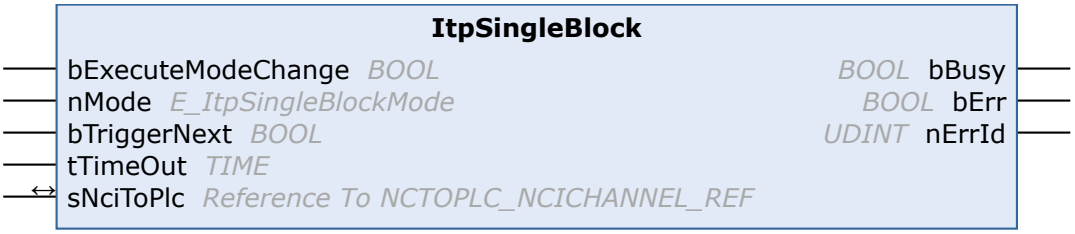
Nicht verfügbar für GST

Dieser Funktionsbaustein ist nicht verfügbar, wenn der GST-Interpreter verwendet wird.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.45 ItpSingleBlock



Der Baustein ItpSingleBlock aktiviert bzw. deaktiviert die Einzelsatzausführung in der NCI. Die Satzweiterschaltung kann direkt aus der SPS mit dem Eingang 'bTriggerNext' angestoßen werden, alternativ kann im XAE der Startbutton des Interpreters (F5) genutzt werden.

Eine weiterführende Beschreibung ist in der Interpreter-Dokumentation [▶ 135] zu finden.

VAR_INPUT

```
VAR_INPUT
    bExecuteModeChange  : BOOL;
    nMode               : E_ItpSingleBlockMode;
    bTriggerNext        : BOOL
    tTimeOut            : TIME;
END_VAR
```

bExecuteModeChange: Durch eine steigende Flanke an diesem Eingang wird der Single Block Mode, der an nMode anliegt, aktiviert.

nMode: Betriebsart für den Einzelsatz (vgl. Einzelsatzbetrieb):

- ItpSingleBlockOff: Einzelsatz aus
- ItpSingleBlockNck: Einzelsatz im NC-Kern
- ItpSingleBlockIntp: Einzelsatz im Interpreter

ItpSingleBlockIntp ist nicht verfügbar, wenn der GST-Interpreter verwendet wird.

bTriggerNext: Durch eine steigende Flanke an diesem Eingang wird die Satzweiterschaltung getriggert.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

```
TYPE E_ItpSingleBlockMode:
(
    ItpSingleBlockOff   := 0,
    ItpSingleBlockNck   := 1,
```

```
    ItpSingleBlockIntp  := 16#4000
);
END_TYPE
```

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy     : BOOL;
    bErr      : BOOL;
    nErrId    : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.46 ItpStartStopEx

Der Baustein ItpStartStopEx startet bzw. stoppt den NC-Kanal.

VAR_INPUT

```
VAR_INPUT
    bStart          : BOOL;
    bStop           : BOOL;
    tTimeOut        : TIME;
END_VAR
```

bStart: eine positive Flanke startet den NC-Kanal

bStop: eine positive Flanke stoppt den NC-Kanal. Bei einem Stop-Befehl werden alle Tabellen in der NC gelöscht und die Achsen geregelt angehalten.

Der Eingang bStop hat eine höhere Priorität als der Eingang bStart, d.h. wenn beide Eingänge eine positive Flanke haben, wird ein Kanal-Stop ausgeführt.

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy     : BOOL;
    bErr      : BOOL;
    nErrId    : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.47 ItpStepOnAfterEStopEx

Der Baustein ItpStepOnAfterEStopEx ermöglicht die weitere Abarbeitung des Teileprogramms nach einem programmierten EStopEx.

VAR_INPUT

```
VAR_INPUT
    bExecute       : BOOL;
    tTimeOut       : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy     : BOOL;
    bErr      : BOOL;
    nErrId    : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Siehe auch:

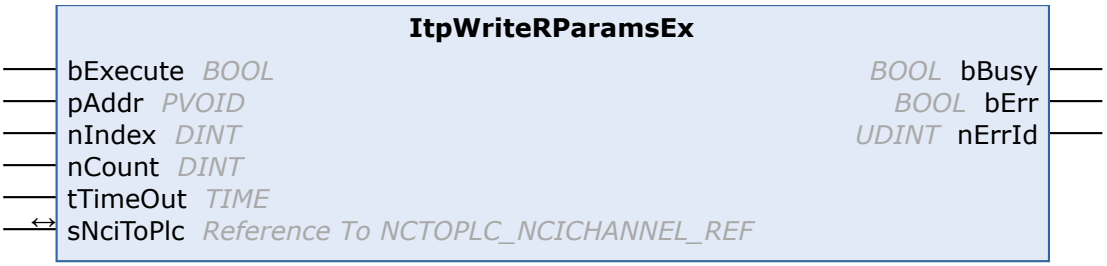
ItpEStopEx [▶ 215]

ItpIsEStopEx [▶ 232]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.48 ItpWriteRParamsEx



Der Baustein ItpWriteRParamsEx schreibt R-Parameter in die NC.

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    pAddr           : DWORD;
    nIndex          : DINT;
    nCount          : DINT;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Mit einer steigenden Flanke wird der Schreibvorgang gestartet.

pAddr: Adresse der Variablen, die die zu schreibenden Daten enthält. Dabei werden die Daten direkt ab der angegebenen Adresse verwendet. D.h. nIndex ist nicht als Offset zu pAddr zu sehen. Die Daten werden gewöhnlich aus einem Array vom Typ LREAL gelesen, das vom Anwender definiert werden muss.

nIndex: Beschreibt den Index des R-Parameters der aus NC Sicht beschrieben werden soll.

nCount: Anzahl der zu beschreibenden R-Parameter

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy     : BOOL;
    bErr      : BOOL;
    nErrId    : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Beispiel

```
VAR
    arrfRParam90to99 : ARRAY[0..9] OF LREAL;
    fbWriteRParam    : ItpWriteRParamsEx;
    n                : INT := 0;
    bWriteParam      : BOOL := FALSE;
    sNciToPlc AT%I*  : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

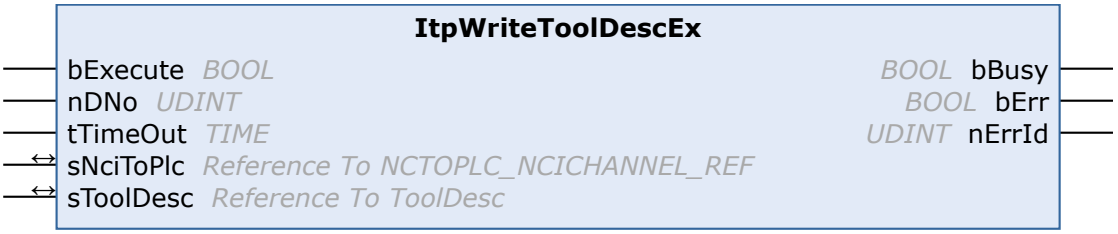
```
FOR n:=0 TO 9 DO
    arrfRParam90to99[n] := 90 + n;
END_FOR

fbWriteRParam(
 bExecute := bWriteParam,
 pAddr := ADR( arrfRParam90to99[0] ),
 nIndex := 90,
 nCount := 10,
 tTimeOut := T#200ms,
 sNciToPlc := sNciToPlc );
```

In diesem Beispiel werden aus NC Sicht die Parameter R90 bis R99 beschrieben.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.49 ItpWriteToolDescEx



Der Baustein **ItpWriteToolDescEx** schreibt einen Block von Werkzeugparametern.

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    nDNo            : UDINT;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt.

nDNo: D-Wort für das die Werkzeugparameter ausgelesen werden sollen. nDNo kann Werte von 1 bis einschließlich 255 annehmen.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc       : NCTOPLC_NCICHANNEL_REF;
    sToolDesc       : ToolDesc;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

sToolDesc: Struktur, die die neuen Werkzeugparameter enthält. Auf diese Struktur wird nur lesend zugegriffen. Die Bedeutung der Parameter ist vom Werkzeugtyp abhängig und kann den Werkzeugdaten [▶ 184] entnommen werden.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bErr        : BOOL;
    nErrId      : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

```
TYPE ToolDesc:
STRUCT
    nToolNumber     : UDINT; (*valid range from 0 .. 65535*)
    nToolType       : UDINT;
    fParam          : ARRAY [2..15] OF LREAL;
END_STRUCT
END_TYPE
```

Siehe auch:

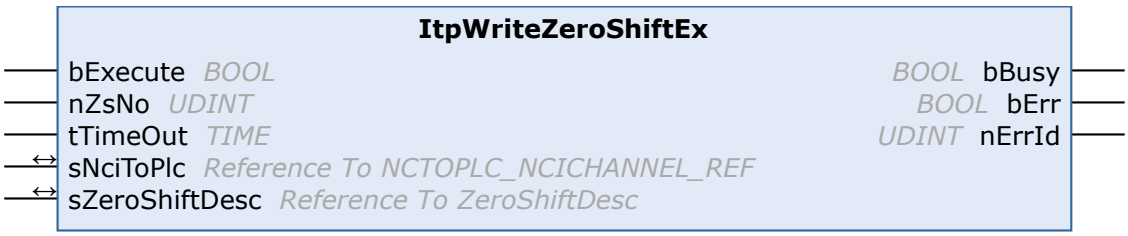
ItpReadToolDescEx [▶ 237]

ItpSetToolDescNullEx [▶ 248]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.50 ItpWriteZeroShiftEx



Der Baustein **ItpWriteZeroShiftEx** schreibt für die angegebene Nullpunktverschiebung die Verschiebungskomponenten X, Y und Z.

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    nZsNo           : UDINT;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt.

nZsNo: Nummer der Nullpunktverschiebung

NC-seitig sind G54 bis G59 Nullpunktverschiebungen. Wobei G58 und G59 nur aus dem NC-Programm editiert werden können. Der gültige Wertebereich für 'nZsNo' ist deshalb von 54 bis 57.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc AT%I*: NCTOPLC_NCICHANNEL_REF;
    sZeroShiftDesc : ZeroShiftDesc;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

sZeroShiftDesc: Struktur mit den Komponenten der Nullpunktverschiebung. Auf diese Struktur wird nur lesend zugegriffen.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy      : BOOL;
    bErr       : BOOL;
    nErrId     : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

```
TYPE ZeroShiftDesc:
STRUCT
    fShiftX         : LREAL;
    fShiftY         : LREAL;
```

```
    fShiftZ         : LREAL;
END_STRUCT
END_TYPE
```

Aus Kompatibilitätsgründen gibt es für jede einstellbare Nullpunktverschiebung pro Achse zwei Parameter (grob & fein). Beim Schreiben einer neuen Nullpunktverschiebung mit diesem Baustein, wird der neue Wert in den 'Fein-Parameter' geschrieben. In den 'Grob-Parameter' wird eine 0.0 eingetragen.

Auf diese Weise ist es möglich, dass z.B. mit dem ItpReadZeroShiftEx [▶ 238] Baustein die Nullpunktverschiebung gelesen, modifiziert und erneut der NC übergeben wird.

Siehe auch:

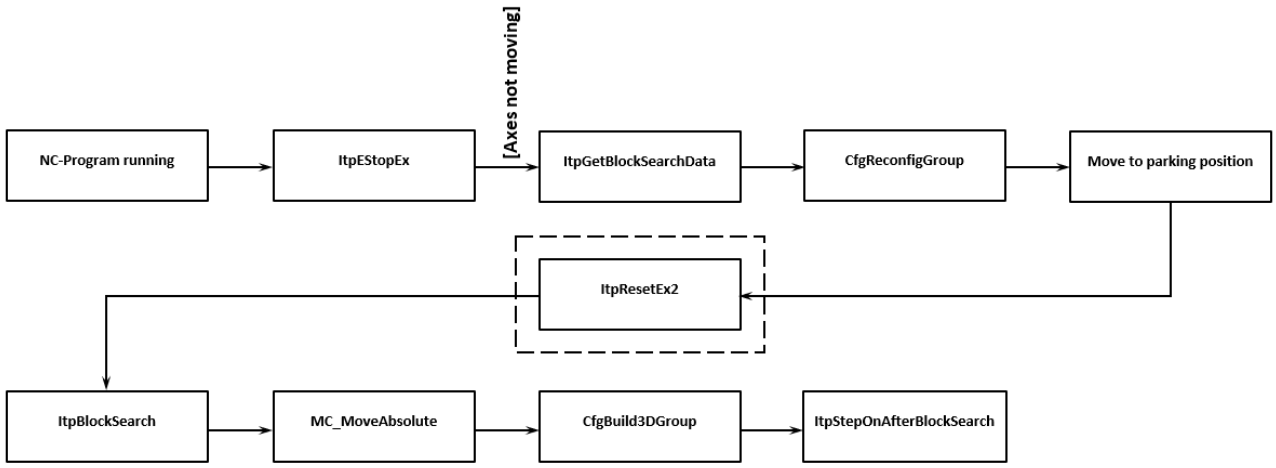
- ItpReadZeroShiftEx [▶ 238]
- ItpSetZeroShiftNullEx [▶ 249]

Voraussetzungen

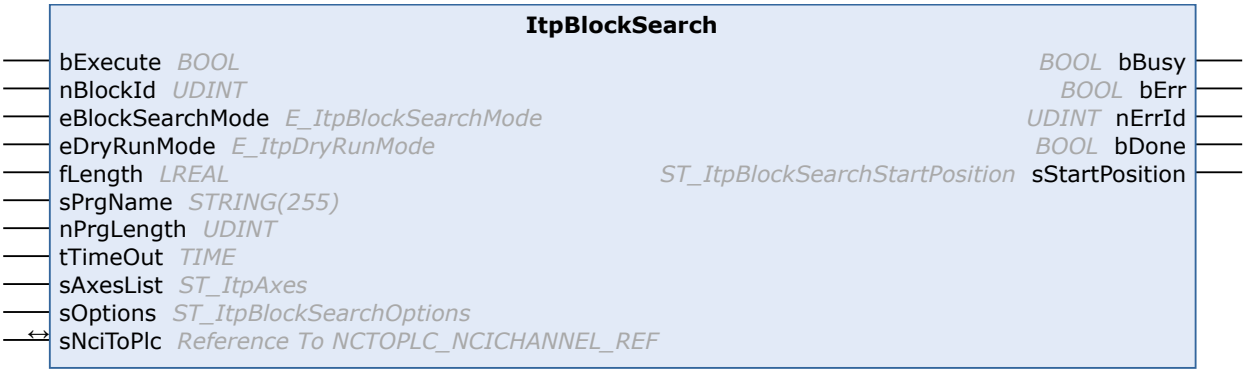
Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.51 Blocksearch

Blocksearch kann dazu benutzt werden, ein Programm für einen Werkzeugwechsel oder am Ende einer Schicht zu unterbrechen. Nach der Unterbrechung kann wieder auf der vorherigen Position aufgesetzt werden.
Das Schema stellt dar, wie der Blocksearch verwendet wird.



6.1.2.51.1 ItpBlocksearch



Der Funktionsbaustein ItpBlocksearch setzt den Interpreter auf die an den Eingängen definierte Stelle. Wenn Blocksearch während des ersten Segments, das eine Bewegung enthält, ausgeführt wird, kann der Ausgang sStartPosition des Bausteins ItpBlocksearch falsche Werte zurückgeben. Aus diesem Grund sollte Blocksearch erst ab dem zweiten Segment genutzt werden.

Die Eingangswerte können vom Funktionsbaustein ItpGetBlocksearchData [▶ 260] übernommen oder manuell gesetzt werden. Nachdem der Interpreter mit ItpBlocksearch auf die definierte Stelle gesetzt worden ist, kann die Bewegung an der Position, die am Ausgang sStartPosition ausgegeben wird, mit ItpStepOnAfterBlocksearch [▶ 261] fortgesetzt werden.

VAR_INPUT

```
VAR_INPUT
    bExecute          : BOOL;
    nBlockId          : UDINT;
    eBlockSearchMode  : E_ItpBlockSearchMode;
    eDryRunMode       : E_ItpDryRunMode;
    fLength           : LREAL;
    sPrgName          : STRING(255);
    nPrgLength        : UDINT;
    tTimeOut          : TIME;
    sAxesList         : ST_ItpAxes;
    sOptions          : ST_ItpBlockSearchOptions;
END_VAR
```

bExecute: Der Befehl wird durch eine steigende Flanke an diesem Eingang ausgelöst.

nBlockId: Blocknummer oder EntryCounter des Segments im NC-Programm, ab der gestartet werden soll.

eBlockSearchMode: Definiert, ob die angegebene nBlockId eine Blocknummer (z.B. N4711) oder ein fortlaufender EntryCounter ist. Voraussetzung für die Verwendung der Blocknummer ist, dass diese eindeutig ist. Vgl. E_ItpBlockSearchMode [▶ 259].

eDryRunMode: Definiert, welche Zeilen des Programms ausgeführt und welche übersprungen werden. Vgl. E_ItpDryRunMode [▶ 259].

fLength: Aufsetzpunkt innerhalb des mit nBlockId ausgewählten Segments in Prozent.

sPrgName: Name bzw. Dateipfad des Programms, das ausgeführt werden soll.

nPrgLength: Gibt die Länge des Strings *sPrgName* an.

tTimeOut: ADS-Timeout-Delay

sAxesList: Definition der Achsen, die sich in der NCI-Gruppe befinden. Vgl. ST_ItpAxes [▶ 259].

sOptions: Gibt Informationen zum Retrace (Rückwärtsfahren) an.

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy             : BOOL;
    bErr              : BOOL;
    nErrId            : UDINT;
    bDone             : BOOL;
    sStartPosition    : ST_ItpBlockSearchStartPosition;
END_VAR
```

bBusy: Bleibt TRUE, bis der Baustein eine Befehlsanforderung ausgeführt hat, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen.

bErr: Wird TRUE, wenn bei der Ausführung des Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS-Fehlerdokumentation oder in der NC-Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

bDone: Der Ausgang wird TRUE, wenn der Befehl erfolgreich ausgeführt wurde.

sStartPosition: Gibt die Startposition aus, von der das NC-Programm weiterläuft. Die einzelnen Achsen müssen vor dem Ausführen von ItpStepOnAfterBlocksearch [▶ 261] an diese Positionen verfahren werden. Vgl. ItpBlocksearch [▶ 260]

E_ItpBlockSearchMode

Über E_ItpBlockSearchMode wird definiert, auf welche Art die Blocksuche beim Blocksearch ausgeführt wird.

```
TYPE E_ItpBlockSearchMode :
(
    ItpBlockSearchMode_Disable        := 0,
    ItpBlockSearchMode_BlockNo        := 1,
    ItpBlockSearchMode_EntryCounter   := 2
);
END_TYPE
```

ItpBlockSearchMode_Disable: Blocksearch deaktiviert (Initialwert).

ItpBlockSearchMode_BlockNo: Der Blocksearch wird über die vom Anwender im NC-Programm programmierte Blocknummer (z.B. N4711) ausgeführt. Voraussetzung ist, dass die Blocknummer vom Anwender eindeutig vergeben wurde.

ItpBlockSearchMode_EntryCounter: Der Blocksearch wird über einen eindeutigen EntryCounter ausgeführt. Dieser EntryCounter ist implizit eindeutig, kann jedoch vom Anwender im NC-Programm nicht eingesehen werden.

E_ItpDryRunMode

Die Enumeration `E_ItpDryRunMode` zählt diejenigen Vorgehensweisen auf, wie mit den programmierten Sätzen vom Anfang des Programms bis zur gesuchten Stelle umgegangen werden soll.

```
TYPE E_ItpDryRunMode :
(
    ItpDryRunMode_Disable             := 0,
    ItpDryRunMode_SkipAll             := 1,
    ItpDryRunMode_SkipMotionOnly      := 2,
    ItpDryRunMode_SkipDwellAndMotion  := 3
);
END_TYPE
```

ItpDryRunMode_Disable: DryRun deaktiviert (Initialwert).

ItpDryRunMode_SkipAll: Alle vorherigen Sätze werden übersprungen. R-Parameter werden geschrieben.

ItpDryRunMode_SkipMotionOnly: Nur Bewegungssätze werden übersprungen. R-Parameter werden geschrieben und Verweilzeiten und M-Funktionen ausgeführt.

ItpDryRunMode_SkipDwellAndMotion: Bewegungssätze und Verweilzeiten werden übersprungen. R-Parameter werden geschrieben und M-Funktionen ausgeführt.

ST_ItpAxes

Die Struktur ST_ItpAxes beinhaltet die Achsen, die sich bei der Abarbeitung des Programms in der NCI-Gruppe befunden haben. Die Interpolationsgruppe sollte zum Zeitpunkt der Ausführung von Blocksearch nicht gebildet sein. Um dennoch eine Referenz auf die Gruppenachsen zu haben, ist die Struktur ST_ItpAxes mit den Gruppenachsen zu füllen.

```
TYPE ST_ItpAxes :
STRUCT
    nAxisIds          : ARRAY[1..8] OF UDINT;
END_STRUCT
END_TYPE
```

nAxisIds: Array der Achen, die sich in der NCI-Gruppe befunden haben. Dabei ist die Reihenfolge nAxisIds[1]=X, nAxisIds[2]=Y, nAxisIds[3]=Z, nAxisIds[4]=Q1, nAxisIds[5]=Q2… Die Achs-Id kann aus dem zyklischen Achsinterface ausgelesen werden.

St_ItpBlockSearchOptions

Die Struktur beinhaltet zusätzliche Blocksearch-Optionen.

```
TYPE ST_ItpBlockSearchOptions :
STRUCT
    bIsRetrace        : BOOL:= FALSE;
    bRetraceBackward  : BOOL:= FALSE;
    bScanStartPos     : BOOL:= FALSE;
END_STRUCT
END_TYPE
```

bIsRetrace: Zeigt an, ob die Retrace-Funktionalität aktiv ist.

bRetraceBackward: Zeigt an, ob rückwärts auf der Bahn verfahren wurde.

bScanStartPos: Gibt an, ob die aktuellen Achspositionen beim Programmstart eingelesen werden sollen oder nicht. In Verbindung mit ST_ItpAxesList ist dieser Eingang auf TRUE zu setzen. Diesen Eingang auf FALSE zu setzen, ist nur aus Kompatibilitätsgründen für alte Projekte sinnvoll.

ST_ItpBlockSearchStartPosition

Die Struktur gibt die Position an, an der das NC-Programm nach einem Blocksearch fortgesetzt wird. Der Anwender ist dafür verantwortlich, die Achsen an die entsprechenden Positionen zu verfahren.

```
TYPE ST_ItpBlockSearchStartPosition :
STRUCT
    sStartPosition    : ARRAY[1..8] OF LREAL;
END_STRUCT
END_TYPE
```

sStartPosition: Array der Achspositionen, an der das NC-Programm fortgesetzt wird.

Dabei ist die Reihenfolge sStartPosition[1]=X, sStartPosition [2]=Y, sStartPosition [3]=Z, sStartPosition [4]=Q1, sStartPosition [5]=Q2…

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
Classic Dialect Interpreter: TwinCAT V3.1.0 GST Interpreter: TwinCAT V3.1.4024.20	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.51.2 ItpGetBlocksearchData

Der Funktionsbaustein ItpGetBlocksearchData liest die aktuelle Position auf der Bahn aus. Üblicherweise wird dieser Befehl im Stillstand aufgerufen. Anschließend kann mit ItpBlockSearch [▶ 257] der Interpreter wieder an die in sBlockSearchData gespeicherte Position aufgesetzt werden.

VAR_INPUT

```
VAR_INPUT
    bExecute          : BOOL;
    tTimeOut          : TIME;
END_VAR
```

bExecute: Der Befehl wird durch eine steigende Flanke an diesem Eingang ausgelöst.

bTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy             : BOOL;
    bErr              : BOOL;
    nErrId            : UDINT;
    sBlockSearchData  : ST_ItpBlockSearchData;
END_VAR
```

bBusy: Bleibt TRUE, bis der Baustein eine Befehlsanforderung ausgeführt hat, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen.

bErr: Wird TRUE, wenn bei der Ausführung des Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS-Fehlerdokumentation oder in der NC-Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

sBlockSearchData: Enthält Informationen zur aktuellen Position auf der Bahn.

```
TYPE ST_ItpBlockSearchData :
STRUCT
    fLength           : LREAL;(* remaining distance of actual movement block in percent*)
    nBlockNo          : UDINT;(* number of the actual block *)
    nBlockCounter     : UDINT;(* counter value of the actual block *)
    bIsRetrace        : BOOL;(* indicates whether Retrace is active*)
    bRetraceBackward  : BOOL;(* indicates whether backward movement took place on the path*)
END_STRUCT
END_TYPE
```

6.1.2.51.3 ItpStepOnAfterBlocksearch

Startet die Bewegung, nachdem ein Blocksearch ausgeführt wurde.

Die Achsen müssen zuvor an die am Ausgang von ItpBlocksearch [▶ 257] ausgegebenen Positionen verfahren worden sein.

VAR_INPUT

```
VAR_INPUT
    bExecute          : BOOL;
    tTimeOut          : TIME;
END_VAR
```

bExecute: Der Befehl wird durch eine steigende Flanke an diesem Eingang ausgelöst.

bTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy     : BOOL;
    bErr      : BOOL;
    nErrId    : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

6.1.2.52 Rückwärtsfahren

6.1.2.52.1 ItpEnableFeederBackup

Der Funktionsbaustein ItpEnableFeederBackup aktiviert die Speicherung der abgefahrenen Bahn für das Rückwärtsfahren. Er muss einmal aktiviert werden, bevor das NC-Programm (G-Code) gestartet ist. Falls die Funktionalität Blocksearch [▶ 257] verwendet wird, muss ItpEnableFeederBackup vor dem Aufruf von ItpBlocksearch [▶ 257] aktiviert werden. Das Feeder-Backup wird solange ausgeführt, bis ein TwinCAT Neustart oder bEnable = FALSE mit steigender Flanke auf bExecute ausgelöst wird.

Wenn das Feeder-Backup nicht aktiviert ist, funktioniert das Rückwärtsfahren nicht. Dies lässt sich über ItpIsFeederBackupEnabled [▶ 263] überprüfen.

VAR_INPUT

```
VAR_INPUT
    bEnable         : BOOL;
    bExecute        : BOOL;
    tTimeOut        : TIME;
END_VAR
```

bEnable: TRUE: aktiviert Feeder-Backup, FALSE: deaktiviert Feeder-Backup

bExecute: Der Befehl wird durch eine steigende Flanke an diesem Eingang ausgelöst.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy     : BOOL;
    bErr      : BOOL;
    nErrId    : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

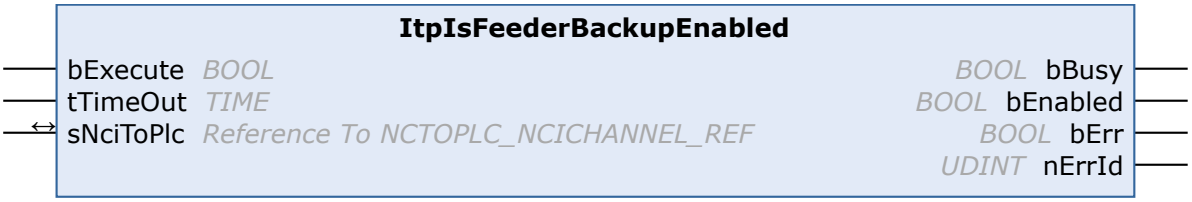
bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0 Classic Interpreter TwinCAT V3.1.4024.40 GST Interpreter	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.52.2 ItpIsFeederBackupEnabled



Der Funktionsbaustein ItpIsFeederBackupEnabled gibt an, ob das Feeder-Backup aktiviert ist. Um rückwärts zu fahren, muss das Feeder-Backup zuvor aktiviert werden. Hierdurch wird die abgefahrene Bahn gespeichert.

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Der Befehl wird durch eine steigende Flanke an diesem Eingang ausgelöst.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy           : BOOL;
    bEnabled        : BOOL;
    bErr            : BOOL;
    nErrId          : UDINT;
END_VAR
```

bBusy: Der Ausgang bBusy bleibt solange auf TRUE, bis der Baustein einen Befehl ausgeführt hat, längstens aber für die Dauer der an dem ‚Timeout'-Eingang angelegten Zeit. Während bBusy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bEnabled: TRUE: Backupliste für Rückverfolgung ist aktiviert, FALSE: Backupliste für Rückverfolgung ist deaktiviert

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in ‚nErrId' enthalten. Wenn der Baustein einen Timeout-Fehler hat, ist ‚Error' = TRUE und ‚nErrId' = 1861 (Hexadezimal 0x745). Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0 Classic Interpreter TwinCAT V3.1.4024.40 GST Interpreter	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.52.3 ItpIsFeedFromBackupList

Die Funktion ItpIsFeedFromBackupList wird TRUE, wenn die Zufuhr-Einträge (SAF & SVB) von der Backupliste gesendet werden. Bei der Rückwärtsbewegung werden alle Einträge von der Backupliste gesendet. Wird das Programm vorwärts ausgeführt, dann stammen die ersten Einträge in der Regel auch von der Backupliste. Dies ist abhängig von der Anzahl zurückverfolgter Einträge sowie der Anzahl Einträge in der SVB- und SAF-Tabelle zu dem Zeitpunkt, als die Rückverfolgung aufgerufen wurde. Alle weiteren Befehle stammen vom ‚Original'-Code.

Während die NCI an der Backupliste arbeitet, sind nicht alle Funktionen verfügbar oder sinnvoll. Hier einige Beispiele:

- Dekodierstopps wie @714 werden nicht ausgewertet
- R-Parameter die verändert werden, haben keinen Einfluss solange auf der Backup-Bahn verfahren wird (vorwärts und rückwärts). Sobald die Bahndaten nicht mehr aus der Backup-Liste kommen, wirken auch R-Parameter-Änderungen wieder.

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0 Classic Interpreter TwinCAT V3.1.4024.40 GST Interpreter	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.52.4 ItpIsFirstSegmentReached

ItpIsFirstSegmentReached ist eine Funktion, die aus dem zyklischen Kanalinterface ermittelt, ob beim Rückwärtsfahren die Startposition des Programms erreicht ist.

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

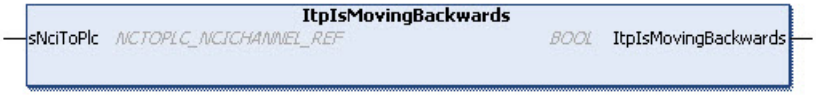
Rückgabewert

Die Funktion liefert TRUE, wenn die Startposition des G-Code-Programms erreicht ist. Ist die Version des zyklischen Kanalinterface kleiner 6, ist der Rückgabewert immer FALSE.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0 Classic Interpreter TwinCAT V3.1.4024.40 GST Interpreter	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.52.5 ItpIsMovingBackwards



ItpIsMovingBackwards ist eine Funktion, die aus dem zyklischen Kanalinterface ermittelt, ob rückwärts auf der Bahn des aktuellen G-Code Programms verfahren wird.

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

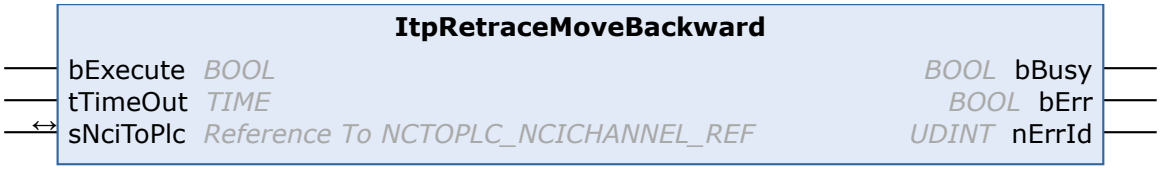
Rückgabewert

Die Funktion liefert TRUE, wenn rückwärts auf der Bahn verfahren wird. Ist die Version des zyklischen Kanalinterface kleiner 6, ist der Rückgabewert immer FALSE.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0 Classic Interpreter TwinCAT V3.1.4024.40 GST Interpreter	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.52.6 ItpRetraceMoveBackward



Der Funktionsbaustein ItpRetraceMoveBackward führt die geometrischen Einträge von der Istposition zum Beginn des Teilprogramms (G-Code).

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Der Befehl wird durch eine steigende Flanke an diesem Eingang ausgelöst.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bErr        : BOOL;
    nErrId      : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Vorgehensweise

1. Feeder-Backupliste aktivieren (siehe ItpEnableFeederBackup [▶ 262])
   ⇨ Das NC-Programm wird mit ItpEStopEx [▶ 215] gestoppt
2. Warten und sichergehen, dass alle Achsen in der Gruppe still stehen
3. ItpRetraceMoveBackward aufrufen
4. Rückwärtsbewegung mit ItpEStop stoppen, ansonsten kehrt das Programm wieder zum Anfang zurück
5. ItpRetraceMoveForward [▶ 267] aufrufen, um wieder nach vorne zu gehen
6. Gegebenenfalls ItpEStopEx und ItpRetraceMoveBackward aufrufen usw.

Hinweis **Nicht in Verbindung mit Vertex-Verschleifung verwenden. M-Funktionen werden bei Rückwärtsbewegung unterdrückt.**

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0 Classic Interpreter TwinCAT V3.1.4024.40 GST Interpreter	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.2.52.7 ItpRetraceMoveForward

Der Funktionsbaustein ItpRetraceMoveForward führt alle Einträge vom Istblock (z.B. Position) in Vorwärtsrichtung zum NC-Kernel. Er wird dazu aufgerufen, die Richtung wieder umzukehren, nachdem ItpRetraceMoveBackward [▶ 266] aufgerufen wurde.

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Der Befehl wird durch eine steigende Flanke an diesem Eingang ausgelöst.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bErr        : BOOL;
    nErrId      : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

siehe auch: ItpRetraceMoveBackward [▶ 266]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0 Classic Interpreter TwinCAT V3.1.4024.40 GST Interpreter	PC oder CX (x86 oder x64)	Tc2_NCI

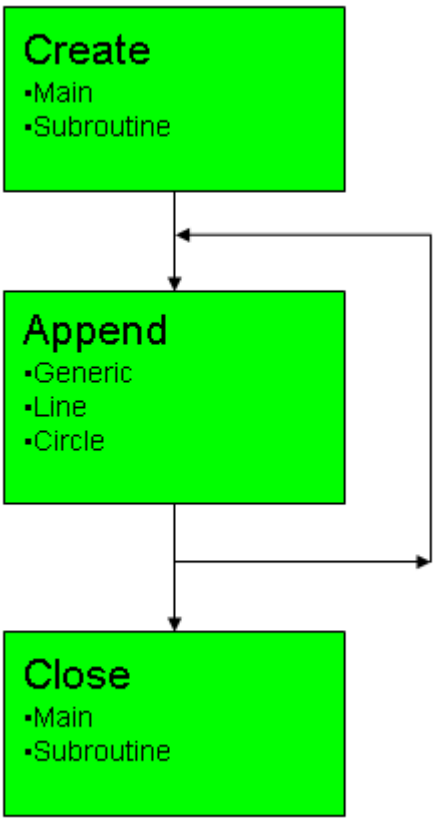
6.1.3 Teileprogramm-Generator

Die Bausteine ItpPpg* stellen eine Möglichkeit dar, um aus der SPS ein Teileprogramm (G-Code-File) zu erstellen. Dabei wird beim Generieren grundsätzlich zwischen einem Hauptprogramm (ItpPpgCreateMain [▶ 274]) und einem Unterprogramm (ItpPpgCreateSubroutine [▶ 275]) unterschieden.

Anschließend können mit ItpPpgAppend* verschiedene NC-Zeilen hinzugefügt werden. Dabei stehen folgende FBs zur Verfügung:

- ItpPpgAppendGeoLine [▶ 272] fügt eine Linearbewegung hinzu.
- ItpPpgAppendGeoCircleByRadius [▶ 270] fügt einen Kreis mit Radiusangabe hinzu.
- ItpPpgAppendGenericBlock [▶ 269] fügt eine selbstdefinierte Zeile, wie z.B. Einschalten der Verrundung oder M-Funktionen hinzu.

Ist das Teileprogramm soweit fertig gestellt, wird es mit den Routinen ItpPpgCloseMain [▶ 273] bzw. ItpPpgCloseSubroutine [▶ 274] geschlossen.



Folgende Funktionsblöcke können verwendet werden:

Funktionsbaustein	Beschreibung
ItpPpgAppendGenericBlock [▶ 269]	Hängt eine generische NC-Zeile an ein spezifiziertes Teileprogramm
ItpPpgAppendGeoCircleByRadius [▶ 270]	Fügt einen Kreis an ein spezifiziertes Teileprogramm an
ItpPpgAppendGeoLine [▶ 272]	Fügt eine Linearbewegung an ein spezifiziertes Teileprogramm an
ItpPpgCloseMain [▶ 273]	Schließt ein zuvor geöffnetes Teileprogramm
ItpPpgCloseSubroutine [▶ 274]	Schließt ein zuvor geöffnetes Unterprogramm
ItpPpgCreateMain [▶ 274]	Öffnet bzw. generiert ein Teileprogramm
ItpPpgCreateSubroutine [▶ 275]	Öffnet bzw. generiert ein Unterprogramm

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.3.1 ItpPpgAppendGenericBlock

Der Baustein ItpPpgAppendGenericBlock fügt eine generische Zeile ans Teileprogramm an. Er kann z.B. dazu verwendet werden, um eine M-Funktion oder die Verrundung einzuschalten.

Vor dem Aufruf rufen Sie ItpPpgCreateMain [▶ 274] oder ItpPpgCreateSubroutine [▶ 275] auf.

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    sPathName       : STRING;
    sBlock          : STRING;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt

sPathName: Name des Teileprogramms mit Pfadangabe

sBlock: Generische Zeile des Teileprogramms, die hinzugefügt werden soll

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy      : BOOL;
    bErr       : BOOL;
    nErrId     : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

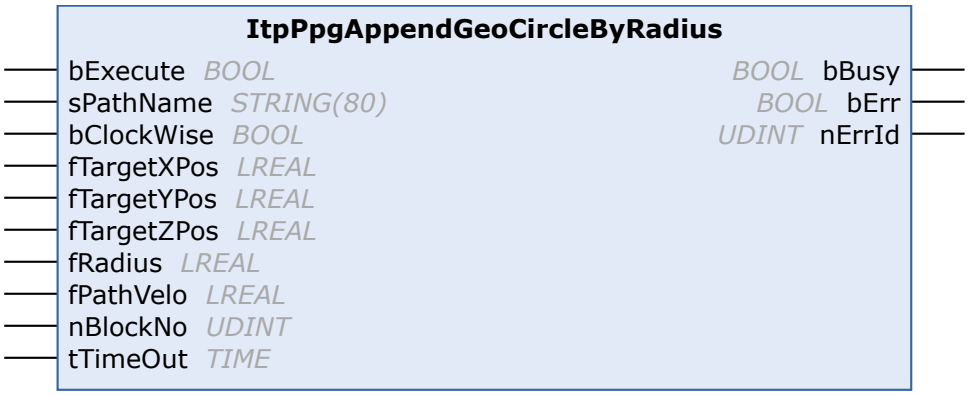
bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.3.2 ItpPpgAppendGeoCircleByRadius



Der Baustein ItpPpgAppendGeoCircleByRadius fügt eine Kreisbewegung ans Teileprogramm an. Dabei wird der Kreis mit dem Radius parametriert.

Vor dem Aufruf rufen Sie ItpPpgCreateMain [▶ 274] oder ItpPpgCreateSubroutine [▶ 275] auf.

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    sPathName       : STRING;
    bClockWise      : BOOL;
    fTargetXPos     : LREAL;
    fTargetYPos     : LREAL;
    fTargetZPos     : LREAL;
    fRadius         : LREAL;
    fPathVelo       : LREAL;
    nBlockNo        : UDINT;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt

sPathName: Name des Teileprogramms mit Pfadangabe

bClockwise: Wenn TRUE, wird der Kreis im Uhrzeigersinn abgefahren, andernfalls gegen den Uhrzeigersinn

fTargetXPos: Zielposition der X-Achse

fTargetYPos: Zielposition der Y-Achse

fTargetZPos: Zielposition der Z-Achse

fRadius: Kreisradius

fPathVelo: Bahngeschwindigkeit

nBlockNo: Zeilennummer im Teileprogramm

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy      : BOOL;
    bErr       : BOOL;
    nErrId     : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

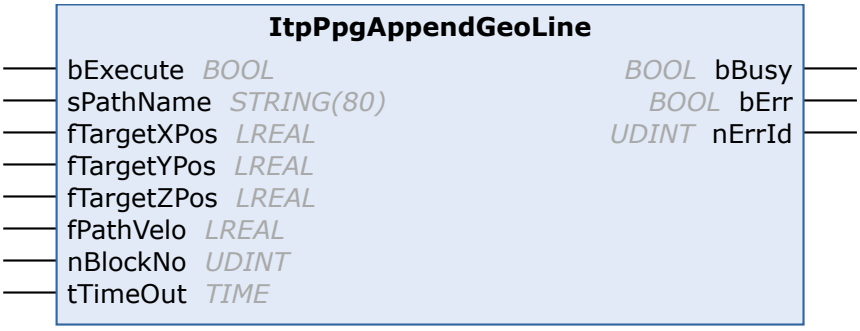
bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.3.3 ItpPpgAppendGeoLine



Der Baustein ItpPpgAppendGeoLine fügt eine Linearbewegung ans Teileprogramm an. Neben der eigentlichen Zielposition werden die vorgesehene Bahngeschwindigkeit und die Zeilennummer übergeben.

Vor dem Aufruf rufen Sie ItpPpgCreateMain [▶ 274] oder ItpPpgCreateSubroutine [▶ 275] auf.

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    sPathName       : STRING;
    fTargetXPos     : LREAL;
    fTargetYPos     : LREAL;
    fTargetZPos     : LREAL;
    fPathVelo       : LREAL;
    nBlockNo        : UDINT;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt.

sPathName: Name des Teileprogramms mit Pfadangabe

fTargetXPos: Zielposition der X-Achse

fTargetYPos: Zielposition der Y-Achse

fTargetZPos: Zielposition der Z-Achse

fPathVelo: Bahngeschwindigkeit

nBlockNo: Zeilennummer im Teileprogramm

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy      : BOOL;
    bErr       : BOOL;
    nErrId     : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.3.4 ItpPpgCloseMain

Der Baustein ItpPpgCloseMain schließt das Hauptprogramm mit dem entsprechenden Code für den Interpreter (M02) ab.

Vor dem Aufruf rufen Sie ItpPpgCreateMain [▶ 274] auf.

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    sPathName       : STRING;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt.

sPathName: Name des Teileprogramms mit Pfadangabe

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy      : BOOL;
    bErr       : BOOL;
    nErrId     : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

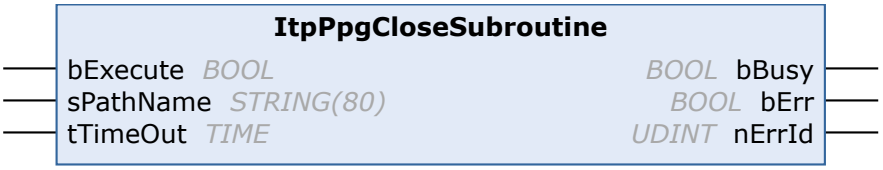
bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.3.5 ItpPpgCloseSubroutine



Der Baustein ItpPpgCloseSubroutine schließt das Unterprogramm mit dem entsprechenden Code für den Interpreter (M17) ab.

Vor dem Aufruf rufen Sie ItpPpgCreateSubroutine [▶ 275] auf.

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    sPathName       : STRING;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt.

sPathName: Name des Teileprogramms mit Pfadangabe

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy     : BOOL;
    bErr      : BOOL;
    nErrId    : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.3.6 ItpPpgCreateMain

Der Baustein ItpPpgCreateMain generiert eine neue Datei, die später als Hauptprogramm abgearbeitet werden kann. Falls die Datei noch nicht existiert, wird sie erzeugt, andernfalls überschrieben.

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    sPathName       : STRING;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt.

sPathName: Name des Teileprogramms mit Pfadangabe

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy      : BOOL;
    bErr       : BOOL;
    nErrId     : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

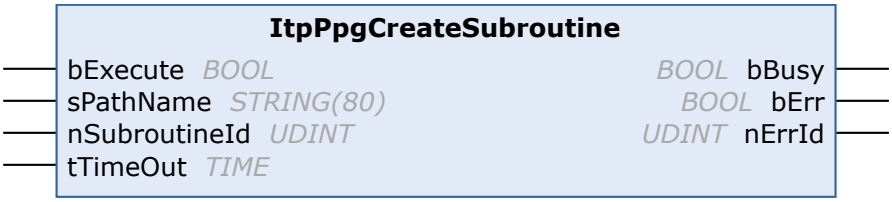
bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.3.7 ItpPpgCreateSubroutine



Der Baustein ItpPpgCreateSubroutine generiert eine neue Datei, die später als Unterprogramm abgearbeitet werden kann. Falls die Datei noch nicht existiert, wird sie erzeugt, andernfalls überschrieben.

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    sPathName       : STRING;
    nSubroutineId   : UDINT;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt.

sPathName: Name des Unterprogramms mit Pfadangabe

nSubroutineId: Nummer des Unterprogramms

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy     : BOOL;
    bErr      : BOOL;
    nErrId    : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.4 Bausteine zur Kompatibilität mit bestehenden Programmen

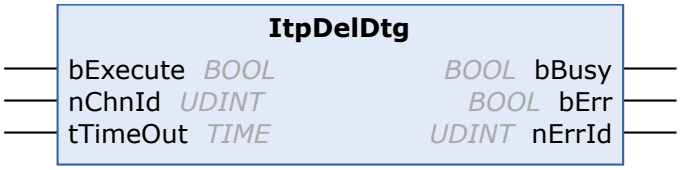
Funktionsbausteine zur Kompatibilität

Die unten aufgeführten Funktionsbausteine existieren zur Kompatibilität mit bestehenden Projekten. Für neue Projekte wird empfohlen, diese Bausteine **nicht** zu verwenden und stattdessen die äquivalenten Bausteine in der oberen Tabelle zu benutzen.

Funktionsbaustein	Beschreibung
ItpDelDtg [▶ 277]	Triggert das Restweglöschen in der NC
ItpEStop [▶ 278]	Triggert den NCI EStop
ItpGetBottleNeckLookAhead [▶ 279]	Liefert die Größe des Look-Aheads für die Flaschenhalserkennung
ItpGetBottleNeckMode [▶ 280]	Liefert den Reaktionsmode für die Flaschenhalserkennung
ItpGetGeoInfoAndHParam [▶ 281]	Liest Informationen über das gerade aktive Segment, vergangene und zukünftige Segmente aus
ItpGoAhead [▶ 282]	Triggert die GoAhead Funktion
ItpIsEStop [▶ 283]	Ermittelt, ob ein EStop ausgeführt wird bzw. ob ein EStop anliegt
ItpLoadProg [▶ 284]	Lädt ein NC-Programm via Programmnamen
ItpReadRParams [▶ 285]	Liest Rechenparameter
ItpReadToolDesc [▶ 286]	Liest die Werkzeugbeschreibung aus der NC
ItpReadZeroShift [▶ 287]	Liest die Nullpunktverschiebung aus der NC
ItpReset [▶ 288]	Führt einen Reset des Interpreters, bzw. des NC-Kanals aus
ItpResetEx [▶ 289]	Führt einen Reset des Interpreters, bzw. des NC-Kanals aus.
ItpResetFastMFunc [▶ 290]	Setzt ein schnelles Signal-Bit zurück
ItpSetBottleNeckLookAhead [▶ 291]	Setzt die Größe des Look-Ahead für die Flaschenhalserkennung

Funktionsbaustein	Beschreibung
ItpSetBottleNeckMode [▶ 292]	Setzt den Reaktionsmode bei eingeschalteter Flaschenhalserkennung
ItpSetSubroutinePath [▶ 293]	Setzt optional den Suchpfad für Unterprogramme
ItpSetToolDescNull [▶ 295]	Setzt alle Toolparameter (inkl. Nummer und Type) auf Null
ItpSetZeroShiftNull [▶ 296]	Setzt alle Nullpunkte auf Null
ItpStartStop [▶ 297]	Startet bzw. Stoppt den Interpreter (NC-Kanal)
ItpStepOnAfterEStop [▶ 298]	Ermöglicht die Weiterbearbeitung des Teileprogramms nach einem NCI EStop
ItpWriteRParams [▶ 299]	Schreibt Rechenparameter
ItpWriteToolDesc [▶ 300]	Schreibt die Werkzeugbeschreibung in die NC
ItpWriteZeroShift [▶ 301]	Schreibt die Nullpunktverschiebung in die NC

6.1.4.1 ItpDelDtg



Der Baustein ItpDelDtg triggert das Restweglöschen. Eine ausführlichere Beschreibung ist in der Interpreter [▶ 162]-Dokumentation zu finden.

Veraltete Version

Dieser Funktionsbaustein existiert ausschließlich zur Gewährleistung der Kompatibilität mit bestehenden Projekten. Für neue Projekte verwenden Sie bitte den Baustein ItpDelDtgEx [▶ 213].

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    nChnId          : UDINT;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt

nChnId: Kanal-ID

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy      : BOOL;
    bErr       : BOOL;
    nErrId     : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

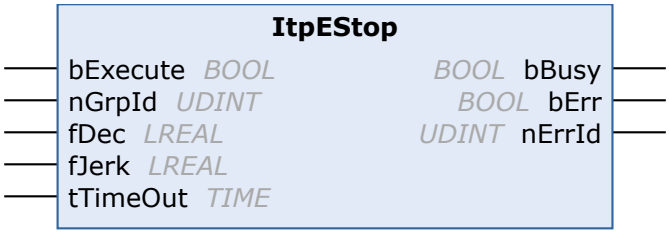
bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.4.2 ItpEStop



Der Baustein ItpEStop triggert den NCI EStop und ermöglicht so ein kontrolliertes Anhalten auf der Bahn. Dabei werden die Grenzwerte für die Verzögerung und den Ruck als Parameter übertragen. Falls diese kleiner sein sollten, als die zurzeit wirkenden Dynamikparameter, so werden die übertragenen Parameter verworfen.

Veraltete Version

Dieser Funktionsbaustein existiert ausschließlich zur Gewährleistung der Kompatibilität mit bestehenden Projekten. Für neue Projekte verwenden Sie bitte den Baustein ItpEStopEx [▶ 215].

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    nGrpId          : UDINT;
    fDec            : LREAL;
    fJerk           : LREAL;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt

nGrpId: Gruppen-ID

fDec: max. Verzögerung mit der angehalten werden soll. Ist fDec kleiner als die zurzeit aktive Verzögerung, so wird fDec nicht übernommen. Es wird so also sichergestellt, dass mindestens mit der Standard-Rampe verzögert wird.

fJerk: max. Ruck mit dem angehalten werden soll. Ist fJerk kleiner als der zurzeit aktive Ruck, so wird fJerk nicht übernommen.

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy      : BOOL;
    bErr       : BOOL;
    nErrId     : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

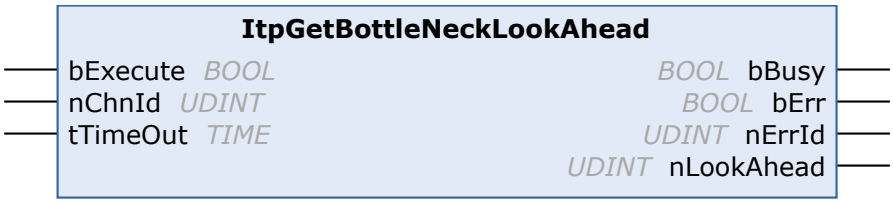
Siehe auch:

ItpStepOnAfterEStop [▶ 298]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.4.3 ItpGetBottleNeckLookAhead



Der Baustein ItpGetBottleNeckLookAhead ermittelt die maximale verwendete Größe des LookAheads für die Falschenhalserkennung (Kontur-Kollisions-Überwachung).

Eine weitere Beschreibung ist in der Interpreter [▶ 196]-Dokumentation zu finden.

Veraltete Version

Dieser Funktionsbaustein existiert ausschließlich zur Gewährleistung der Kompatibilität mit bestehenden Projekten. Für neue Projekte verwenden Sie bitte den Baustein ItpGetBottleNeckLookAheadEx [▶ 217].

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    nChnId          : UDINT;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt

nChnId: Kanal-ID

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy           : BOOL;
    bErr            : BOOL;
    nErrId          : UDINT;
    nLookAhead      : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der, an dem 'Timeout'-Eingang angelegten, Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wenn der Baustein ein Timeout-Fehler hat, so ist 'Error' = TRUE und 'nErrId' = 1861 (Hexadezimal 0x745). Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

nLookAhead: Größe des Look-Aheads für die Flaschenhalserkennung

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.4.4 ItpGetBottleNeckMode

Der Baustein ItpGetBottleNeckMode liest die Verhaltensweise bei einer auftretenden Kontur-Kollision (Flaschenhals) aus.

Eine weitere Beschreibung ist in der Interpreter [▶ 196]-Dokumentation zu finden.

Veraltete Version

Dieser Funktionsbaustein existiert ausschließlich zur Gewährleistung der Kompatibilität mit bestehenden Projekten. Für neue Projekte verwenden Sie bitte den Baustein ItpGetBottleNeckModeEx [▶ 218].

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    nChnId          : UDINT;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt

nChnId: Kanal-ID

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy           : BOOL;
    bErr            : BOOL;
    nErrId          : UDINT;
    eBottleNeckMode: E_ItpBnMode
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der, an dem 'Timeout'-Eingang angelegten, Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wenn der Baustein ein Timeout-Fehler hat, so ist 'Error' = TRUE und 'nErrId' = 1861 (Hexadezimal 0x745). Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

eBottleNeckMode: Enum für die Verhaltensweise bei einer auftretenden Kontur-Kollision

```
TYPE E_ItpBnMode:
(
    ItpBnm_Abort  := 0,
    ItpBnm_Adjust := 1,
    ItpBnm_Leave  := 2
);
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.4.5 ItpGetGeoInfoAndHParam

Der Funktionsbaustein ItpGetGeoInfoAndHParam liest Informationen über das gerade aktive Segment, vergangene und zukünftige Segmente aus. Hierzu gehören Blocknummer, H-Parameter und Bahnrestweg auf dem Segment.

Veraltete Version

Dieser Funktionsbaustein existiert ausschließlich zur Gewährleistung der Kompatibilität mit bestehenden Projekten. Für neue Projekte verwenden Sie bitte den Baustein ItpGetGeoInfoAndHParamEx [▶ 223].

VAR_IN_OUT

```
VAR_IN_OUT
    sNciToPlc           : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

sNciToPlc: Struktur des zyklischen Kanalinterfaces von der NCI zur SPS. Auf diese Struktur wird nur lesend zugegriffen. (Typ: NCTOPLC_NCICHANNEL_REF [▶ 335])

VAR_OUTPUT

```
VAR_OUTPUT
    stTab               : ST_ItpPreViewTabEx;
    nErrId              : UDINT;
END_VAR
```

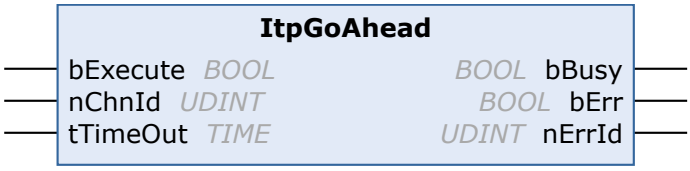
stTab: Struktur, die die Segmentdaten enthält.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.4.6 ItpGoAhead



Der Baustein ItpGoAhead darf nur in Verbindung mit dem Dekodierstopp '@717' [▶ 172] verwendet werden. Eine ausführlichere Beschreibung dieses Dekodierstopps ist in der Interpreter-Dokumentation [▶ 128] zu finden.

Veraltete Version

Dieser Funktionsbaustein existiert ausschließlich zur Gewährleistung der Kompatibilität mit bestehenden Projekten. Für neue Projekte verwenden Sie bitte den Baustein ItpGoAheadEx [▶ 230].

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    nChnId          : UDINT;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt

nChnId: Kanal-ID

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy      : BOOL;
    bErr       : BOOL;
    nErrId     : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.4.7 ItpIsEStop

Der Funktionsbaustein ItpIsEStop liefert mit bEStop die Information, ob ein EStop-Kommando angestoßen wurde. Ist bEStop TRUE, dann wurde zuvor ein EStop ausgeführt (z.B. ItpEStop). Dabei liefert das Flag **keine** Information darüber, ob die Achsen bereits stehen oder sich noch auf der Bremsrampe befinden.

Nach der Ausführung von ItpStepOnAfterEStop, liefert ItpIsEStop wieder ein FALSE zurück.

Veraltete Version

Dieser Funktionsbaustein existiert ausschließlich zur Gewährleistung der Kompatibilität mit bestehenden Projekten. Für neue Projekte verwenden Sie bitte den Baustein ItpIsEStopEx [▶ 232].

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    nGrpId          : UDINT
    tTimeOut        : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt

nGrpId: Gruppen-ID

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy           : BOOL;
    bEStop          : BOOL;
    bErr            : BOOL;
    nErrId          : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der, an dem 'Timeout'-Eingang angelegten, Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bEStop: TRUE: EStop-Kommando wurde ausgeführt, FALSE: Es liegt kein EStop an

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Siehe auch:

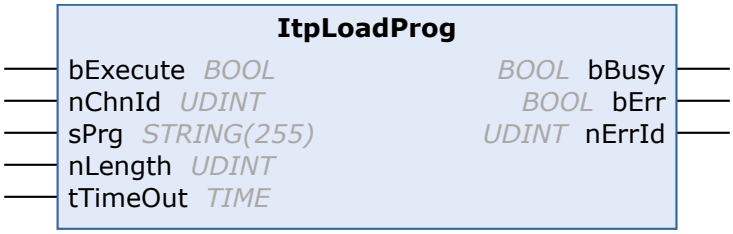
ItpEStop [▶ 278]

ItpStepOnAfterEStop [▶ 298]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.4.8 ItpLoadProg



Veraltete Version

Dieser Funktionsbaustein existiert ausschließlich zur Gewährleistung der Kompatibilität mit bestehenden Projekten. Für neue Projekte verwenden Sie bitte den Baustein ItpLoadProgEx [▶ 233].

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    nChnId          : UDINT;
    sPrg            : STRING;
    nLength         : UDINT;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird das NC-Programm ausgeführt

nChnId: Kanal-ID

sPrg: Name des NC-Programms, das ausgeführt wird

nLength: Stringlänge des Programmnamens

tTimeOut: ADS Timeout-Delay

Hinweis Das NC-Programm wird im Verzeichnis "TwinCAT\Mc\Nci " gesucht, wenn keine weiteren Angaben gemacht werden. Es kann jedoch auch ein absoluter Pfad angegeben werden.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy      : BOOL;
    bErr       : BOOL;
    nErrId     : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

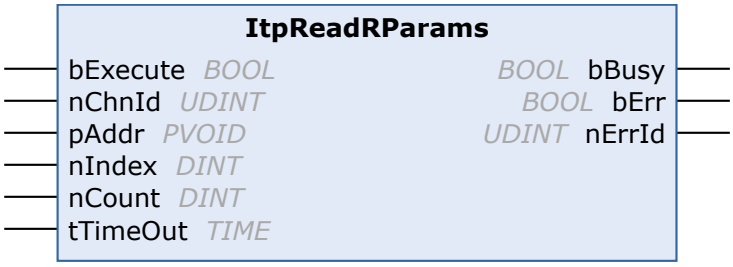
bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.4.9 ItpReadRParams



Veraltete Version

Dieser Funktionsbaustein existiert ausschließlich zur Gewährleistung der Kompatibilität mit bestehenden Projekten. Für neue Projekte verwenden Sie bitte den Baustein ItpReadRParamsEx [▶ 236].

Der Baustein ItpReadRParams liest Rechenparameter, kurz R-Parameter, der NC. Eine genaue Beschreibung der Rechenparameter ist hier [▶ 136] zu finden. Insgesamt stehen 1000 R-Parameter zur Verfügung, wovon die ersten 900 (0..899) lokal, d.h. nur im aktuellen NC-Kanal, sichtbar sind. Die letzten 100 (900..999) R-Parameter sind global und somit NC-weit sichtbar.

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    nChnId          : UDINT;
    pAddr           : PVOID;
    nIndex          : DINT;
    nCount          : DINT;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Mit einer steigenden Flanke wird der Lesevorgang gestartet

nChnId: ID des NC-Kanals dessen R-Parameter gelesen werden sollen

pAddr: Adresse der Zielvariablen der zu lesenden Daten. Dabei werden die Daten direkt ab der angegebenen Adresse von der NC beschrieben. D.h. nIndex ist nicht als Offset zu pAddr zu sehen. Die Daten befinden sich für gewöhnlich in einem Array vom Typ LREAL, das vom Anwender definiert werden muss.

nIndex: Beschreibt den Index des R-Parameters der aus NC Sicht gelesen werden soll.

nCount: Anzahl der zu lesenden R-Parameter

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bErr        : BOOL;
    nErrId      : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

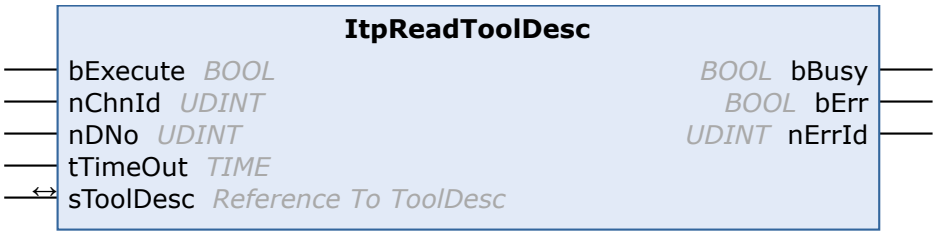
Siehe auch:

ItpWriteRParams [▶ 299]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.4.10 ItpReadToolDesc



Der Baustein ItpReadToolDesc liest für das mitgegebene D-Wort die Werkzeugparameter.

Veraltete Version

Dieser Funktionsbaustein existiert ausschließlich zur Gewährleistung der Kompatibilität mit bestehenden Projekten. Für neue Projekte verwenden Sie bitte den Baustein ItpReadToolDescEx [▶ 237].

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    nChnId          : UDINT;
    nDNo            : UDINT;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt

nChnId: Kanal-ID

nDNo: D-Wort für das die Werkzeugparameter ausgelesen werden sollen. nDoNo kann Werte von 1 bis einschließlich 255 annehmen.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
    sToolDesc       : ToolDesc;
END_VAR
```

sToolDesc: Struktur, in der die Werkzeugparameter von nDNo geschrieben werden. Die Bedeutung der Parameter ist vom Werkzeugtyp abhängig und kann den Werkzeugdaten [▶ 184] entnommen werden.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy     : BOOL;
    bErr      : BOOL;
    nErrId    : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

```
TYPE ToolDesc:
STRUCT
    nToolNumber     : UDINT; (*valid range from 0 .. 65535*)
    nToolType       : UDINT;
    fParam          : ARRAY [2..15] OF LREAL;
END_STRUCT
END_TYPE
```

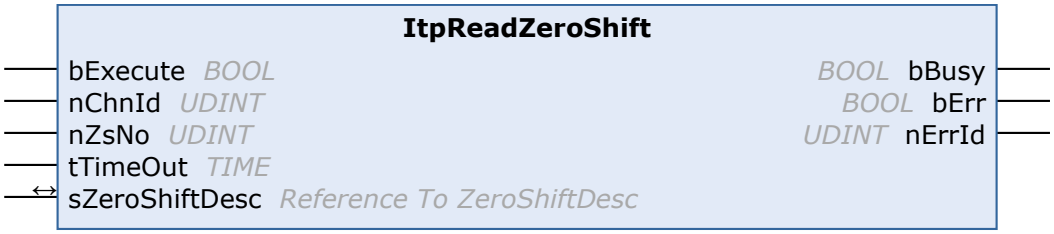
siehe auch:

ItpWriteToolDesc [▶ 300]; ItpSetToolDescNull [▶ 295]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.4.11 ItpReadZeroShift



Der Baustein **ItpReadZeroShift** liest für die angegebene Nullpunktverschiebung die Verschiebungskomponenten X, Y und Z.

Hinweis Aus Kompatibilitätsgründen gibt es pro Nullpunktverschiebung (z.B. G54) für jede Achse zwei Einträge (grob & fein), die addiert werden. Dieser Funktionsbaustein wertet beide Einträge aus und addiert sie automatisch.

Veraltete Version

Dieser Funktionsbaustein existiert ausschließlich zur Gewährleistung der Kompatibilität mit bestehenden Projekten. Für neue Projekte verwenden Sie bitte den Baustein ItpReadZeroShiftEx [▶ 238].

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    nChnId          : UDINT;
```

```
    nZsNo           : UDINT;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt

nChnId: Kanal-ID

nZsNo: Nummer der Nullpunktverschiebung, NC-seitig sind G54 bis G59 Nullpunktverschiebungen. Der gültige Wertebereich für 'nZsNo' ist deshalb von 54 bis 59.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
    sZeroShiftDesc : ZeroShiftDesc;
END_VAR
```

sZeroShiftDesc: Struktur mit den Komponenten der Nullpunktverschiebung.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy      : BOOL;
    bErr       : BOOL;
    nErrId     : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

```
TYPE ZeroShiftDesc:
STRUCT
    fShiftX         : LREAL;
    fShiftY         : LREAL;
    fShiftZ         : LREAL;
END_STRUCT
END_TYPE
```

siehe auch:

ItpWriteZeroShift [▶ 301]; ItpSetZeroShiftNull [▶ 296]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.4.12 ItpReset

Veraltete Version

Dieser Funktionsbaustein existiert ausschließlich zur Gewährleistung der Kompatibilität mit bestehenden Projekten. Für neue Projekte verwenden Sie bitte den Baustein ItpResetEx2 [▶ 239].

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    nChnId          : UDINT;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird ein Reset des NC-Kanals durchgeführt

nChnId: Kanal-ID

tTimeOut: ADS Timeout-Delay

Hinweis Bei einem Reset werden alle Tabellen in der NC gelöscht. Die Achsen werden instantan abgebremst. Deshalb sollte ein Reset nur im Fehlerfall oder bei stehenden Achsen durchgeführt werden.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bErr        : BOOL;
    nErrId      : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.4.13 ItpResetEx

Der Baustein 'ItpResetEx' führt einen Kanal-Reset aus und löscht damit alle vorhandenen Tabellen des NC-Kanals. Im Gegensatz zum herkömmlichen ItpReset [▶ 288] wird ein aktiver Kanal erst gestoppt, bevor der Reset ausgeführt wird. Damit wird die Programmierung in der SPS vereinfacht, da nicht explizit überprüft werden muss, ob die Achsen noch in Bewegung sind.

Veraltete Version

Dieser Funktionsbaustein existiert ausschließlich zur Gewährleistung der Kompatibilität mit bestehenden Projekten. Für neue Projekte verwenden Sie bitte den Baustein ItpResetEx2 [▶ 239].

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    nGrpId          : UDINT;
    nChnId          : UDINT;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt

nGrpId: Gruppen-ID

nChnId: Kanal-ID

tTimeOut: ADS Timeout-Delay (das bBusy-Signal kann länger anliegen als tTimeOut)

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy      : BOOL;
    bErr       : BOOL;
    nErrId     : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

siehe auch: ItpStartStop [▶ 297]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.4.14 ItpResetFastMFunc

Dieser Funktionsbaustein stellt eine Alternative zum Autoreset bzw. dem Zurücksetzen mit einer anderen M-Funktion dar (Resetliste bei der Parametrierung der M-Funktion). Aus Gründen der Übersichtlichkeit vermeiden Sie einen Mischbetrieb zwischen dem Zurücksetzen mit einer M-Funktion und diesem Funktionsbaustein.

Mit einer steigenden Flanke am Eingang **bExecute** wird die schnelle M-Funktion [▶ 167] **nMFuncNo** zurückgesetzt. Für den Fall, dass die M-Funktion nicht anliegt, wird **kein** Fehler zurückgegeben.

Veraltete Version

Dieser Funktionsbaustein existiert ausschließlich zur Gewährleistung der Kompatibilität mit bestehenden Projekten. Für neue Projekte verwenden Sie bitte den Baustein ItpResetFastMFuncEx [▶ 240].

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    nChnId          : UDINT;
    nMFuncNo        : UINT;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt

nChnId: Kanal-ID

nMFuncNo: Fliegende M-Funktion, die zurückgesetzt werden soll

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy      : BOOL;
    bErr       : BOOL;
    nErrId     : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.4.15 ItpSetBottleNeckLookAhead

Der Baustein ItpSetBottleNeckLookAhead legt fest, wie viel Segmente maximal für die Flaschenhalserkennung (Kontur-Kollisions-Überwachung) in die Zukunft geschaut darf. Dabei ist zu beachten, dass Segmente, die aufgrund der Radiuskorrektur eingefügt wurden (z.B. Zusatzsegmente an spitzen Winkeln) mit berücksichtigt werden.

Eine weitere Beschreibung ist in der Interpreter [▶ 196]-Dokumentation zu finden.

Veraltete Version

Dieser Funktionsbaustein existiert ausschließlich zur Gewährleistung der Kompatibilität mit bestehenden Projekten. Für neue Projekte verwenden Sie bitte den Baustein ItpSetBottleNeckLookAheadEx [▶ 241].

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    nChnId          : UDINT;
    nLookAhead      : UDINT;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt

nChnId: Kanal-ID

nLookAhead: Legt die Größe des Look-Aheads fest

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy      : BOOL;
    bErr       : BOOL;
    nErrId     : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.4.16 ItpSetBottleNeckMode

Der Baustein ItpSetBottleNeckMode legt die Verhaltensweise bei einer auftretenden Kontur-Kollision (Flaschenhals) fest.

Eine weitere Beschreibung ist in der Interpreter [▶ 196]-Dokumentation zu finden.

Veraltete Version

Dieser Funktionsbaustein existiert ausschließlich zur Gewährleistung der Kompatibilität mit bestehenden Projekten. Für neue Projekte verwenden Sie bitte den Baustein ItpSetBottleNeckModeEx [▶ 242].

VAR_INPUT

```
VAR_INPUT
    bExecute       : BOOL;
    nChnId         : UDINT;
    eBottleNeckMode: E_ItpBnMode
    tTimeOut       : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt

nChnId: Kanal-ID

eBottleNeckMode: Enum für die Verhaltensweise bei einer auftretenden Kontur-Kollision

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy     : BOOL;
    bErr      : BOOL;
    nErrId    : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

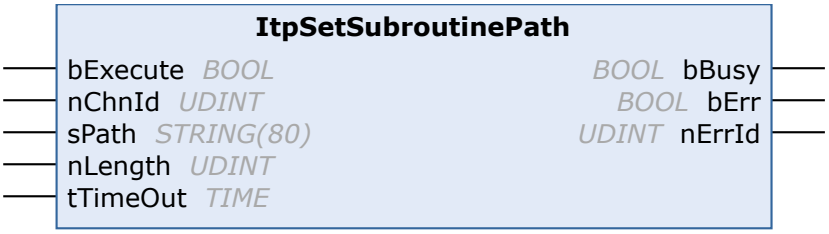
nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

```
TYPE E_ItpBnMode:
(
    ItpBnm_Abort   := 0,
    ItpBnm_Adjust  := 1,
    ItpBnm_Leave   := 2
);
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.4.17 ItpSetSubroutinePath



Mit dem Baustein ItpSetSubroutinePath wird optional der Suchpfad für Unterprogramme gesetzt.

Wenn ein Unterprogramm noch eingebunden werden muss, wird in folgender Reihenfolge nach der Datei gesucht:

- optionaler Suchpfad (ItpSetSubroutinePath)
- Pfad aus dem das Hauptprogramm geladen wurde
- TwinCAT\Mc\Nci -Verzeichnis

Es kann immer nur ein optionaler Pfad wirken und dieser bleibt bestehen, bis er

- mit einem anderen Pfad oder
- mit einem Leerstring

überschrieben wird.

Nach einem TwinCAT-Restart muss der Pfad neu zugewiesen werden.

Veraltete Version

Dieser Funktionsbaustein existiert ausschließlich zur Gewährleistung der Kompatibilität mit bestehenden Projekten. Für neue Projekte verwenden Sie bitte den Baustein ItpSetSubroutinePathEx [▶ 247].

Interface

```
VAR_INPUT
    bExecute        : BOOL;
    nChnId          : UDINT;
    sPath           : STRING;
    nLength         : UDINT;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt

nChnId: Kanal-ID

sPath: optionaler Pfad für Unterprogramme, wird mit einem Leerstring deaktiviert.

nLength: Stringlänge

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy      : BOOL;
    bErr       : BOOL;
    nErrId     : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.4.18 ItpSetToolDescNull

Der FB ItpSetToolDescNull überschreibt alle Werkzeugparameter (inkl. Nummer & Type) des Kanals mit Null.

> **Veraltete Version**
>
> Dieser Funktionsbaustein existiert ausschließlich zur Gewährleistung der Kompatibilität mit bestehenden Projekten. Für neue Projekte verwenden Sie bitte den Baustein ItpSetToolDescNullEx [▶ 248].

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    nChnId          : UDINT;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Mit einer steigenden Flanke werden alle Werkzeugparameter des NC-Kanals mit Null überschrieben.

nChnId: ID des NC-Kanals

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy      : BOOL;
    bErr       : BOOL;
    nErrId     : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

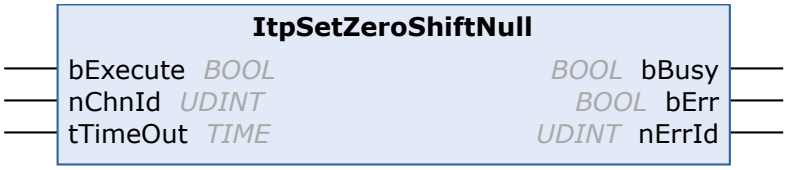
Siehe auch:

- ItpWriteToolDesc [▶ 300],
- ItpReadToolDesc [▶ 286]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.4.19 ItpSetZeroShiftNull



Der FB ItpSetZeroShiftNull überschreibt alle Nullpunktverschiebungen des Kanals mit Null.

Veraltete Version

Dieser Funktionsbaustein existiert ausschließlich zur Gewährleistung der Kompatibilität mit bestehenden Projekten. Für neue Projekte verwenden Sie bitte den Baustein ItpSetZeroShiftNullEx.

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    nChnId          : UDINT;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Mit einer steigenden Flanke werden alle Nullpunktverschiebungen des NC-Kanals mit Null überschrieben.

nChnId: ID des NC-Kanals

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy      : BOOL;
    bErr       : BOOL;
    nErrId     : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Siehe auch:

- ItpWriteZeroShift [▶ 301]
- ItpReadZeroShift [▶ 287]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.4.20 ItpStartStop

Veraltete Version

Dieser Funktionsbaustein existiert ausschließlich zur Gewährleistung der Kompatibilität mit bestehenden Projekten. Für neue Projekte verwenden Sie bitte den Baustein ItpStartStopEx [▶ 251].

Interface

```
VAR_INPUT
    bStart          : BOOL;
    bStop           : BOOL;
    nChnId          : UDINT;
    tTimeOut        : TIME;
END_VAR
```

bStart: eine positive Flanke startet den NC-Kanal

bStop: eine positive Flanke stoppt den NC-Kanal. Bei einem Stop-Befehl werden alle Tabellen in der NC gelöscht und die Achsen geregelt angehalten.

nChnId: Kanal-ID

tTimeOut: ADS Timeout-Delay

HINWEIS! Der Eingang bStop hat eine höhere Priorität als der Eingang bStart, d.h. wenn beide Eingänge eine positive Flanke haben, wird ein Kanal-Stop ausgeführt.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy     : BOOL;
    bErr      : BOOL;
    nErrId    : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.4.21 ItpStepOnAfterEStop

Der Baustein ItpStepOnAfterEStop ermöglicht die weitere Abarbeitung des Teileprogramms nach einem programmierten EStop.

Veraltete Version

Dieser Funktionsbaustein existiert ausschließlich zur Gewährleistung der Kompatibilität mit bestehenden Projekten. Für neue Projekte verwenden Sie bitte den Baustein ItpStepOnAfterEStopEx [▶ 252].

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    nGrpId          : UDINT;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt

nGrpId: Gruppen-ID

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy      : BOOL;
    bErr       : BOOL;
    nErrId     : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

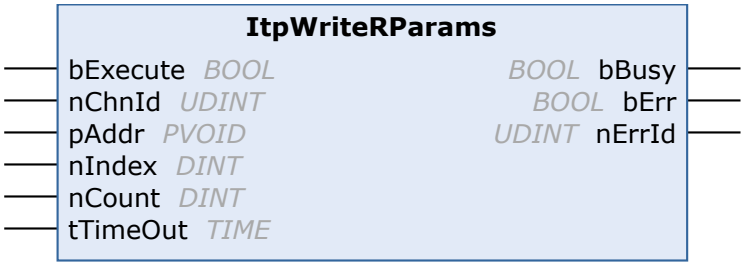
Siehe auch:

- ItpEStop [▶ 278]
- ItpIsEStop [▶ 283]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.4.22 ItpWriteRParams



Der Baustein ItpWriteRParams schreibt R-Parameter in die NC.

> **Veraltete Version**
>
> Dieser Funktionsbaustein existiert ausschließlich zur Gewährleistung der Kompatibilität mit bestehenden Projekten. Für neue Projekte verwenden Sie bitte den Baustein ItpWriteRParamsEx [▶ 253].

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    nChnId          : UDINT;
    pAddr           : PVOID;
    nIndex          : DINT;
    nCount          : DINT;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Mit einer steigenden Flanke wird der Schreibvorgang gestartet.

nChnId: ID des NC-Kanals dessen R-Parameter beschrieben werden sollen.

pAddr: Adresse der Variablen, die die zu schreibenden Daten enthält. Dabei werden die Daten direkt ab der angegebenen Adresse verwendet. D.h. nIndex ist nicht als Offset zu pAddr zu sehen. Die Daten befinden sich für gewöhnlich in einem Array vom Typ LREAL, das vom Anwender definiert werden muss.

nIndex: Beschreibt den Index des R-Parameters der aus NC Sicht beschrieben werden soll.

nCount: Anzahl der zu beschreibenden R-Parameter

tTimeOut: ADS Timeout-Delay

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy      : BOOL;
    bErr       : BOOL;
    nErrId     : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Beispiel

In diesem Beispiel werden aus NC Sicht die Parameter R90 bis R99 beschrieben.

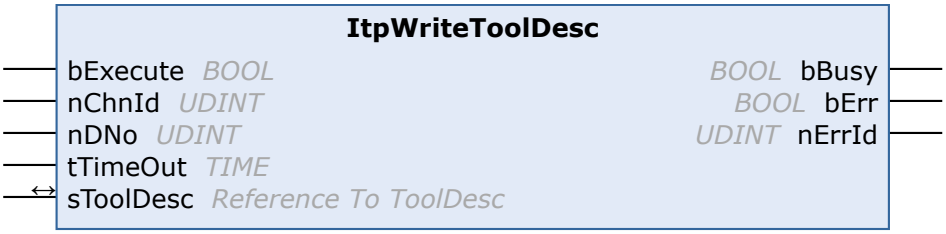
```
VAR
    arrfRParam90to99 : ARRAY[0..9] OF LREAL;
    fbWriteRParam    : ItpWriteRParams;
    n                : INT := 0;
    bWriteParam      : BOOL := FALSE;
END_VAR
```

```
FOR n:=0 TO 9 DO
    arrfRParam90to99[n] := 90 + n;
END_FOR

fbWriteRParam(
 bExecute := bWriteParam,
 nChnId     := 2,
 pAddr     := ADR( arrfRParam90to99[0] ),
 nIndex     := 90,
 nCount     := 10,
 tTimeOut := T#200ms );
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.4.23 ItpWriteToolDesc



Der Baustein **ItpWriteToolDesc** schreibt einen Block von Werkzeugparametern.

> **Veraltete Version**
>
> Dieser Funktionsbaustein existiert ausschließlich zur Gewährleistung der Kompatibilität mit bestehenden Projekten. Für neue Projekte verwenden Sie bitte den Baustein ItpWriteToolDescEx [▶ 254].

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    nChnId          : UDINT;
    nDNo            : UDINT;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt

nChnId: Kanal-ID

nDNo: D-Wort für das die Werkzeugparameter ausgelesen werden sollen. nDoNo kann Werte von 1 bis einschließlich 255 annehmen.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
    sToolDesc       : ToolDesc;
END_VAR
```

sToolDesc: Struktur, die die neuen Werkzeugparameter enthält. Auf diese Struktur wird nur lesend zugegriffen. Die Bedeutung der Parameter ist vom Werkzeugtyp abhängig und kann den Werkzeugdaten [▶ 184] entnommen werden.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy     : BOOL;
    bErr      : BOOL;
    nErrId    : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

```
TYPE ToolDesc:
STRUCT
    nToolNumber     : UDINT; (*valid range from 0 .. 65535*)
    nToolType       : UDINT;
    fParam          : ARRAY [2..15] OF LREAL;
END_STRUCT
END_TYPE
```

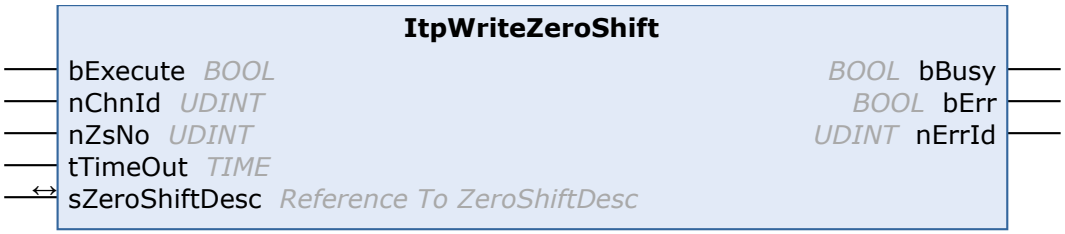
Siehe auch:

- ItpReadToolDesc [▶ 286]
- ItpSetToolDescNull [▶ 295]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.4.24 ItpWriteZeroShift



Der Baustein **ItpWriteZeroShift** schreibt für die angegebene Nullpunktverschiebung die Verschiebungskomponenten X, Y und Z.

Aus Kompatibilitätsgründen gibt es für jede einstellbare Nullpunktverschiebung pro Achse zwei Parameter (grob & fein). Beim Schreiben einer neuen Nullpunktverschiebung mit diesem Baustein wird der neue Wert in den 'Fein-Parameter' geschrieben. In den 'Grob-Parameter' wird eine 0.0 eingetragen. Auf diese Weise ist es möglich, dass z.B. mit dem ItpReadZeroShift [▶ 287] Baustein die Nullpunktverschiebung gelesen, modifiziert und erneut der NC übergeben wird.

Veraltete Version

Dieser Funktionsbaustein existiert ausschließlich zur Gewährleistung der Kompatibilität mit bestehenden Projekten. Für neue Projekte verwenden Sie bitte den Baustein ItpWriteZeroShiftEx [▶ 256].

VAR_INPUT

```
VAR_INPUT
    bExecute        : BOOL;
    nChnId          : UDINT;
    nZsNo           : UDINT;
    tTimeOut        : TIME;
END_VAR
```

bExecute: Durch eine steigende Flanke an diesem Eingang wird der Befehl ausgeführt

nChnId: Kanal-ID

nZsNo: Nummer der Nullpunktverschiebung

NC-seitig sind G54 bis G59 Nullpunktverschiebungen, wobei G58 und G59 nur aus dem NC-Programm editiert werden können. Der gültige Wertebereich für 'nZsNo' ist deshalb von 54 bis 57.

tTimeOut: ADS Timeout-Delay

VAR_IN_OUT

```
VAR_IN_OUT
    sZeroShiftDesc : ZeroShiftDesc;
END_VAR
```

sZeroShiftDesc: Struktur mit den Komponenten der Nullpunktverschiebung. Auf diese Struktur wird nur lesend zugegriffen.

VAR_OUTPUT

```
VAR_OUTPUT
    bBusy     : BOOL;
    bErr      : BOOL;
    nErrId    : UDINT;
END_VAR
```

bBusy: Dieser Ausgang bleibt solange auf TRUE, bis der Baustein eine Befehlsanforderung ausführt, längstens aber für die Dauer der an dem 'Timeout'-Eingang angelegten Zeit. Während Busy = TRUE wird an den Eingängen kein neuer Befehl angenommen. Bitte beachten Sie, dass nicht die Ausführung des Dienstes, sondern nur dessen Annahme zeitlich überwacht wird.

bErr: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in 'nErrId' enthalten. Wird durch das Ausführen eines Befehls an den Eingängen auf FALSE zurückgesetzt.

nErrId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS Fehlerdokumentation oder in der NC Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

```
TYPE ZeroShiftDesc:
STRUCT
    fShiftX : LREAL;
    fShiftY : LREAL;
    fShiftZ : LREAL;
END_STRUCT
END_TYPE
```

Siehe auch:

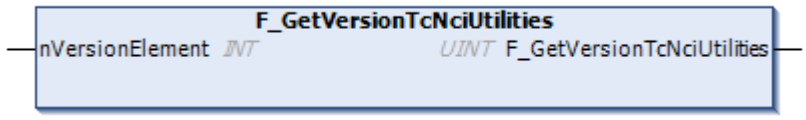
- ItpReadZeroShift [▶ 287]
- ItpSetZeroShiftNull [▶ 301]

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.5 Obsolete

6.1.5.1 F_GetVersionTcNciUtilities



Diese Funktion gibt eine Stelle der dreiteiligen Versionsnummer der TwinCAT 2 SPS-Bibliothek TcNciUtilities.lib als UINT zurück.

> **Veraltete Version**
>
> Diese Funktion existiert ausschließlich zur Gewährleistung der Kompatibilität mit bestehenden Projekten. Für neue Projekte verwenden Sie bitte die globale Struktur stLibVersion_Tc2_NCI.

VAR_INPUT

```
FUNCTION F_GetVersionNciUtilities

VAR_INPUT
    nVersionElement      : INT;
END_VAR
```

nVersionElement: Stelle der Versionsnummer die ausgelesen werden soll (Wertebereich: [1..3])

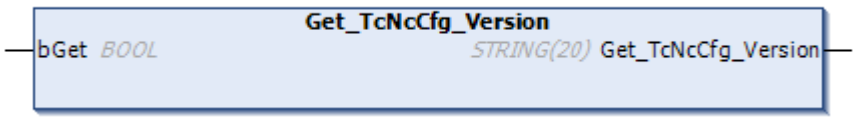
Rückgabewert

F_GetVersionNciUtilities: Versionsnummer

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.5.2 Get_TcNcCfg_Version



Diese Funktion gibt die Versionsnummer der TwinCAT 2 SPS-Bibliothek TcNcCfg.lib als String zurück.

> **Veraltete Version**
>
> Diese Funktion existiert ausschließlich zur Gewährleistung der Kompatibilität mit bestehenden Projekten. Für neue Projekte verwenden Sie bitte die globale Struktur stLibVersion_Tc2_NCI.

Rückgabewert

Get_TcNcCfg_Version: Versionsnummer

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.1.5.3 ItpGetVersion

ItpGetVersion ist eine Funktion, die die Versionsnummer der TwinCAT2 SPS-Bibliothek TcNCI.lib als String zurückgibt.

> **Veraltete Version**
>
> Diese Funktion existiert ausschließlich zur Gewährleistung der Kompatibilität mit bestehenden Projekten. Für neue Projekte verwenden Sie bitte die globale Struktur stLibVersion_Tc2_NCI.

VAR_INPUT

```
FUNCTION ItpGetVersion

VAR_INPUT
END_VAR
```

Rückgabewert

ItpGetVersion: Versionsnummer

Beispiel

```
VAR
    strVersion: STRING(20);
END_VAR
strVersion := ItpGetVersion();
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_NCI

6.2 PLC Library: Tc2_PlcInterpolation

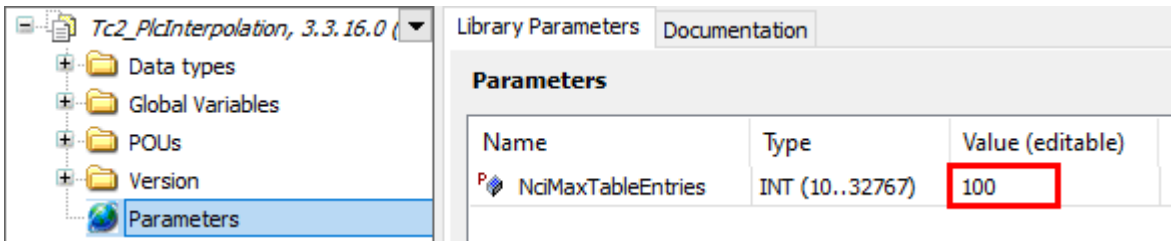
Die Tc2_PlcInterpolation-Bilbliothek bietet eine Alternative zur Verwendung von G-Code (DIN 66025). Mit dieser Bibliothek ist es möglich, interpolierte Fahrbefehle unmittelbar aus der SPS und ohne die Verwendung von G-Code auszuführen.

> **Alternative für Pick-and-Place-Anwendungen**
>
> TF5420 TwinCAT 3 Motion Pick-and-Place führt mehrdimensionale Bewegungen aus. Es wurde speziell für die Anforderungen von Pick-and-Place-Anwendungen entwickelt und stellt für diesen Anwendungsfall eine Alternative zur Tc2_PlcInterpolation-Bibliothek dar. Alle zugehörigen Funktionsbausteine sind in der Bibliothek Tc3_McCoordinatedMotion enthalten.

In einem ersten Schritt wird eine Tabelle verschiedener Fahrbefehle und Zusatzfunktionen beschrieben. Hierzu werden Strukturen, wie zum Beispiel ST_NciGeoLine, an den FB NciFeedTablePreparation übergeben. Hierdurch wird der Fahrbefehl an die Tabelle angehängt. Wenn die Tabelle voll ist oder alle erforderlichen Einträge in der Tabelle sind, wird NciFeedTable dazu aufgerufen, den Tabelleninhalt zum NC-Kern zu übertragen. Mit der Datenübertragung wird die Ausführung direkt gestartet.

NciMaxTableEntries können angepasst werden

Ab der Bibliotheks-Version 3.3.16.0 (enthalten ab TC3.1.4024.11) ist die Anzahl an maximalen Tabelleneinträgen im Bereich von 10 bis 32767 editierbar. Der Default-Wert beträgt 100 Einträge.



Funktionsbausteine

Funktionsbausteine, die zur Gruppierung von Achsen (oder zur Kanalsteuerung (Kanaloverride) benötigt werden, befinden sich in der PLC Library: Tc2_NCI [▶ 203].

Funktionsbaustein	Beschreibung
FB_NciFeedTablePreparation [▶ 306]	Füllt eine Tabelle mit NCI-Bewegungen in der SPS
FB_NciFeedTable [▶ 307]	Übergibt eine zuvor geschriebene Tabelle an den NC-Kern und startet die Bewegung

Datenstrukturen

Folgende Strukturen können als Eingangsparameter für den Baustein NciFeedTablePreparation benutzt werden:

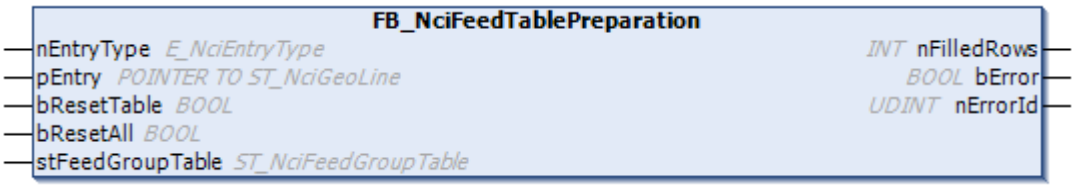
Strukturen	Enum	Beschreibung
Organisation		
	E_NciEntryTypeNone	Keine Funktion
ST_NciGeoStart [▶ 310]	E_NciEntryTypeGeoStart	Stellt die Startposition des ersten Geometrieeintrags ein
ST_NciEndOfTables [▶ 320]	E_NciEntryTypeEndOfTables	Signalisiert das Ende der Geometrietabelle
Fahrbefehle		
ST_NciGeoLine [▶ 310]	E_NciEntryTypeGeoLine	Beschreibt eine Gerade
ST_NciGeoCirclePlane [▶ 311]	E_NciEntryTypeGeoCirclePlane	Beschreibt einen Kreis in der Hauptebene (Mittelpunktprogrammierung)
ST_NciGeoCircleCIP [▶ 312]	E_NciEntryTypeGeoCircleCIP	Beschreibt einen frei im Raum liegenden Kreis
ST_NciGeoBezier3 [▶ 313]	E_NciEntryTypeGeoBezier3	Beschreibt einen Bezier 3. Ordnung mit Kontrollpunkten
ST_NciGeoBezier5 [▶ 314]	E_NciEntryTypeGeoBezier5	Beschreibt einen Bezier 5. Ordnung mit Kontrollpunkten
ST_NciDwellTime [▶ 319]	E_NciEntryTypeDwellTime	Beschreibt eine Verweilzeit
Bahnparameter		
ST_NciBaseFrame [▶ 317]	E_NciEntryTypeBaseFrame	Beschreibt eine Nullpunktverschiebung und Rotation
ST_NciVertexSmoothing [▶ 317]	E_NciEntryTypeVertexSmoothing	Aktiviert eine Verschleifung an Segmentübergängen
ST_NciTangentialFollowingDesc [▶ 319]	E_NciEntryTypeTfDesc	Aktiviert die tangentiale Nachführung des Werkzeugs
Dynamik		
ST_NciDynOvr [▶ 316]	E_NciEntryTypeDynOvr	Verändert den dynamischen Override
ST_NciAxisDynamics [▶ 318]	E_NciEntryTypeAxisDynamics	Limitiert die Dynamik der Achsen
ST_NciPathDynamics [▶ 318]	E_NciEntryTypePathDynamics	Limitiert die Bahndynamik

Strukturen	Enum	Beschreibung
ST_NciFeedrateIpol [▶ 319]	E_NciEntryTypeFeedrateIpol	Stellt die Art der Vorschubinterpolation ein
Parameterbefehle		
ST_NciHParam [▶ 316]	E_NciEntryTypeHParam	Setzt einen H-Parameter (DINT)
ST_NciSParam [▶ 316]	E_NciEntryTypeSParam	Setzt einen S-Parameter (WORD)
ST_NciTParam [▶ 316]	E_NciEntryTypeTParam	Setzt einen T-Parameter (WORD)
ST_NciMFuncFast [▶ 315]	E_NciEntryTypeMFuncFast	Parametriert eine schnelle M-Funktion (kein Handshake)
ST_NciMFuncHsk [▶ 315]	E_NciEntryTypeMFuncHsk	Parametriert eine M-Funktion mit Handshake
ST_NciMFuncResetAllFast [▶ 315]	E_NciEntryTypeResetAllFast	Setzt alle schnellen M-Funktionen zurück

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_PlcInterpolation

6.2.1 FB_NciFeedTablePreparation



Der Funktionsbaustein FB_NciFeedTablePreparation hängt einen Eintrag eines spezifischen Typs an die Feed-Tabelle (stFeedGroupTable). Ein angehängter Eintrag kann mehr als eine Zeile in der Tabelle erzeugen. Falls sich nicht genug freie Zeilen in der Tabelle befinden, wird ein Fehler zurückgeliefert und es wird kein Eintrag in der Tabelle hinzugefügt. Dann muss der Eintrag entweder einer anderen Tabelle hinzugefügt werden oder derselben, nachdem FB_NciFeedTable ausgeführt wurde. Dieser Funktionsbaustein kümmert sich um Modalfunktionen wie die tangentiale Nachführung. Es ist deshalb wichtig, immer dieselbe Instanz dieses Funktionsbausteins zu verwenden. Der Funktionsbaustein kann in einem SPS-Zyklus mehrfach aufgerufen werden.

VAR_INPUT

```
VAR_INPUT
    nEntryType       : E_NciEntryType;
    pEntry           : POINTER TO ST_NciGeoLine;
    bResetTable      : BOOL;
    bResetAll        : BOOL;
END_VAR
```

nEntryType: Spezifiziert den Eintragstyp, z.B. Linie, Kreis, tangentiale Nachführung

pEntry: Pointer auf Eintragsstruktur – muss übereinstimmen mit nEntryType

bResetTable: Wenn bResetTable = TRUE dann wird die Tabelle ‚stFeedGroupTable' auf Null gesetzt und nFilledRows wird ebenfalls auf Null gesetzt. Wenn nErrorId = ErrNciFeedTableFull ist, wird dieser Fehler zurückgesetzt. Alle Modalflags (wie tangentiale Nachführung) bleiben konstant.

bResetAll: Genau wie bResetTable. Daneben werden alle Modalflags auf ihren Standardwert gesetzt und alle Fehler-IDs zurückgesetzt.

VAR_IN_OUT

```
VAR_IN_OUT
    stFeedGroupTable : ST_NciFeedGroupTable
END_VAR
```

stFeedGroupTable: Tabelle, die die Zeilen für den NC-Kernel enthält.

VAR_OUTPUT

```
VAR_OUTPUT
    nFilledRows      : INT;
    bError           : BOOL;
    nErrorId         : UDINT;
END_VAR
```

nFilledRows: Anzahl gefüllter Zeilen.

bError: Wird TRUE, wenn ein Fehler aufgetreten ist.

nErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS-Fehlerdokumentation oder in der NC-Fehlerdokumentation (Fehlercodes ab 0x4000) nachgeschlagen werden.

Hinweis Wenn bResetTable, bResetAll, oder bError TRUE ist, werden keine weiteren Einträge akzeptiert.

Hinweis Der Fehlercode 0x4B72 zeigt an, dass die Tabelle voll ist, und der letzte Eintrag nicht akzeptiert wurde.

Beispiel:

```
stGeoLine.nDisplayIndex := 1;
stGeoLine.fEndPosX := 0;
stGeoLine.fEndPosY := 400;
stGeoLine.fEndPosZ := 100;
stGeoLine.fEndPosQ1 :=-90;
stGeoLine.fVelo := 1000; (*mm per sec*)

fbFeedTablePrep(
    nEntryType := E_NciEntryTypeGeoLine,
    pEntry := ADR(stGeoLine),
    bResetTable:= FALSE,
    stFeedGroupTable:= stNciFeedGroupTable,
    nFilledRows=> nFilledRows,
    bError => bError,
    nErrorId => nErrorId);
```

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_PlcInterpolation

6.2.2 FB_NciFeedTable

Der Funktionsbaustein **FB_NciFeedTable** übergibt eine gegebene Tabelle zum NC-Kern. Wenn der Override gesetzt und die Freigaben aktiviert sind, wird die Ausführung sofort gestartet. Nachdem die Übergabe abgeschlossen ist, wird bFeedingDone TRUE. Dieses Signal kann für einen erneutes Überschreiben der Tabelle mit NciFeedTablePreparation [▶ 306] verwendet werden. In NciFeedTablePreparation muss die Tabelle vorher zurückgesetzt werden.

bChannelDone signalisiert die vollständige Ausführung im NC-Kern der Tabellen. Deshalb muss der Identifier ST_NciEndOfTables [▶ 308] an das Ende der letzten Tabelle gesetzt werden.

VAR_INPUT

```
VAR_INPUT
    bExecute          : BOOL;
    bReset            : BOOL;
    bLogFeederEntries : BOOL;
END_VAR
```

bExecute: Der Befehl wird durch eine steigende Flanke an diesem Eingang ausgelöst.

bReset: Löst einen Kanalreset aus und setzt ebenfalls den Funktionsbaustein zurück.

bLogFeederEntries: Wenn TRUE, wird eine Logdatei ‚PlcItpFeed.log' in den TwinCAT\Mc\Nci-Ordner geschrieben. Sie enthält alle Einträge, die über ADS an den NC-Kern gesendet werden. Wenn bLogFeederEntries = TRUE, dann ist mehr Zeit erforderlich, bis bFeedingDone TRUE wird.

VAR_IN_OUT

```
VAR_IN_OUT
    stFeedGroupTable  : ST_NciFeedGroupTable;
    stNciToPlc        : NCTOPLC_NCICHANNEL_REF;
END_VAR
```

stFeedGroupTable: Tabelle, die die Zeilen für den NC-Kern enthält.

stNciToPlc: Die Struktur des zyklischen Kanalinterfaces von der NCI zur SPS.

VAR_OUTPUT

```
VAR_OUTPUT
    bFeedingDone      : BOOL;
    bChannelDone      : BOOL;
    bFeedBusy:        : BOOL;
    bResetBusy:       : BOOL;
    bError            : BOOL;
    nErrorId          : UDINT;
END_VAR
```

bFeedingDone: Wird TRUE, wenn alle Zeilen der Tabelle an den NC-Kern gesendet wurden.

bChannelDone: Wird TRUE, wenn alle Einträge der Tabelle im NC-Kern ausgeführt wurden und ST_NciEndOfTables erkannt wurde.

bFeedBusy: Wird TRUE, wenn der Funktionsbaustein Einträge an den NC-Kern sendet.

bResetBusy: Wird TRUE, während ein Reset ausgeführt wird.

bError: Wird TRUE, wenn ein Fehler aufgetreten ist.

nErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das Ausführen eines Befehls an den Eingängen auf 0 zurückgesetzt. Die Fehlernummern in ErrId können in der ADS-Fehlerdokumentation oder in der NC-Fehlerdokumentation nachgeschlagen werden.

Voraussetzungen

Entwicklungsumgebung	Zielplattform	Einzubindende SPS-Bibliotheken
TwinCAT V3.1.0	PC oder CX (x86 oder x64)	Tc2_PlcInterpolation

6.2.3 Typen und Enums

E_NciEntryType

```
TYPE E_NciEntryType :
(
    E_NciEntryTypeNone := 0,
    E_NciEntryTypeGeoStart := 1,
    E_NciEntryTypeGeoLine := 2,
    E_NciEntryTypeGeoCirclePlane := 3,
    E_NciEntryTypeGeoCircleCIP := 4,
    E_NciEntryTypeGeoBezier3 := 10,
    E_NciEntryTypeGeoBezier5 := 11,
    E_NciEntryTypeMFuncHsk := 20,
```

```
    E_NciEntryTypeMFuncFast := 21,
    E_NciEntryTypeMFuncResetAllFast := 23,
    E_NciEntryTypeHParam := 24,
    E_NciEntryTypeSParam := 25,
    E_NciEntryTypeTParam := 26,
    E_NciEntryTypeDynOvr := 50,
    E_NciEntryTypeVertexSmoothing := 51,
    E_NciEntryTypeBaseFrame := 52,
    E_NciEntryTypePathDynamics := 53,
    E_NciEntryTypeAxisDynamics := 55,
    E_NciEntryTypeDwellTime := 56,
    E_NciEntryTypeFeedrateIpol := 57,
    E_NciEntryTypeTfDesc := 100,
    E_NciEntryTypeEndOfTables := 1000
);
END_TYPE
```

Strukturen	Enum	Beschreibung
Organisation		
	E_NciEntryTypeNone	Keine Funktion
ST_NciGeoStart [▶ 310]	E_NciEntryTypeGeoStart	Stellt die Startposition des ersten Geometrieeintrags ein
ST_NciEndOfTables [▶ 320]	E_NciEntryTypeEndOfTables	Signalisiert das Ende der Geometrietabelle
Fahrbefehle		
ST_NciGeoLine [▶ 310]	E_NciEntryTypeGeoLine	Beschreibt eine Gerade
ST_NciGeoCirclePlane [▶ 311]	E_NciEntryTypeGeoCirclePlane	Beschreibt einen Kreis in der Hauptebene (Mittelpunktprogrammierung)
ST_NciGeoCircleCIP [▶ 312]	E_NciEntryTypeGeoCircleCIP	Beschreibt einen frei im Raum liegenden Kreis
ST_NciGeoBezier3 [▶ 313]	E_NciEntryTypeGeoBezier3	Beschreibt einen Bezier 3. Ordnung mit Kontrollpunkten
ST_NciGeoBezier5 [▶ 314]	E_NciEntryTypeGeoBezier5	Beschreibt einen Bezier 5. Ordnung mit Kontrollpunkten
ST_NciDwellTime [▶ 319]	E_NciEntryTypeDwellTime	Beschreibt eine Verweilzeit
Bahnparameter		
ST_NciBaseFrame [▶ 317]	E_NciEntryTypeBaseFrame	Beschreibt eine Nullpunktverschiebung und Rotation
ST_NciVertexSmoothing [▶ 317]	E_NciEntryTypeVertexSmoothing	Aktiviert eine Verschleifung an Segmentübergängen
ST_NciTangentialFollowingDesc [▶ 319]	E_NciEntryTypeTfDesc	Aktiviert die tangentiale Nachführung des Werkzeugs
Dynamik		
ST_NciDynOvr [▶ 316]	E_NciEntryTypeDynOvr	Verändert den dynamischen Override
ST_NciAxisDynamics [▶ 318]	E_NciEntryTypeAxisDynamics	Limitiert die Dynamik der Achsen
ST_NciPathDynamics [▶ 318]	E_NciEntryTypePathDynamics	Limitiert die Bahndynamik
ST_NciFeedrateIpol [▶ 319]	E_NciEntryTypeFeedrateIpol	Stellt die Art der Vorschubinterpolation ein
Parameterbefehle		
ST_NciHParam [▶ 316]	E_NciEntryTypeHParam	Setzt einen H-Parameter (DINT)
ST_NciSParam [▶ 316]	E_NciEntryTypeSParam	Setzt einen S-Parameter (WORD)
ST_NciTParam [▶ 316]	E_NciEntryTypeTParam	Setzt einen T-Parameter (WORD)
ST_NciMFuncFast [▶ 315]	E_NciEntryTypeMFuncFast	Parametriert eine schnelle M-Funktion (kein Handshake)
ST_NciMFuncHsk [▶ 315]	E_NciEntryTypeMFuncHsk	Parametriert eine M-Funktion mit Handshake

Strukturen	Enum	Beschreibung
ST_NciMFuncResetAllFast [▶ 315]	E_NciEntryTypeResetAllFast	Setzt alle schnellen M-Funktionen zurück

ST_NciGeoStart

Stellt die Startposition des ersten Geometrie-Eintrags ein. Dies ist erforderlich, wenn der erste Geometrie-Eintrag ein Kreis ist oder die tangentiale Nachführung im ersten Segment ON ist. Diese Struktur kann optional bei jedem Start der ersten Tabelle geschrieben werden.

```
TYPE ST_NciGeoStart :
STRUCT
    nEntryType: E_NciEntryType := E_NciEntryTypeGeoStart; (*do not override this parameter *)
    fPosX: LREAL;
    fPosY: LREAL;
    fPosZ: LREAL;
    fPosQ1: LREAL;
    fPosQ2: LREAL;
    fPosQ3: LREAL;
    fPosQ4: LREAL;
    fPosQ5: LREAL;
END_STRUCT
END_TYPE
```

nEntryType: Diesen Parameter nicht überschreiben (Typ: E_NciEntryType [▶ 308])

fPosX: Startposition X

fPosY: Startposition Y

fPosZ: Startposition Z

fPosQ1: Startposition Q1

fPosQ2: Startposition Q2

fPosQ3: Startposition Q3

fPosQ4: Startposition Q4

fPosQ5: Startposition Q5

ST_NciGeoLine

Beschreibt eine Gerade mit spezifizierter Geschwindigkeit.

```
TYPE ST_NciGeoLine :
STRUCT
    nEntryType: E_NciEntryType := E_NciEntryTypeGeoLine; (*do not override this parameter *)
    nDisplayIndex: UDINT;
    fEndPosX: LREAL;
    fEndPosY: LREAL;
    fEndPosZ: LREAL;
    fEndPosQ1: LREAL;
    fEndPosQ2: LREAL;
    fEndPosQ3: LREAL;
    fEndPosQ4: LREAL;
    fEndPosQ5: LREAL;
    fVelo: LREAL;
    bRapidTraverse: BOOL;
    bAccurateStop: BOOL; (* VeloEnd := 0 *)
END_STRUCT
END_TYPE
```

nEntryType: Diesen Parameter nicht überschreiben (Typ: E_NciEntryType [▶ 308])

nDisplayIndex: Zu Anzeigezwecken, wie Satznummer im G-Code

fEndPosX: Zielposition X

fEndPosY: Zielposition Y

fEndPosZ: Zielposition Z

fEndPosQ1: Zielposition Q1

fEndPosQ2: Zielposition Q2

fEndPosQ3: Zielposition Q3

fEndPosQ4: Zielposition Q4

fEndPosQ5: Zielposition Q5

fVelo: Zielbahngeschwindigkeit, wie F in G-Code, allerdings in Basiseinheiten pro Sekunde (z.B. mm/s)

bRapidTraverse: TRUE hat dieselbe Wirkung wie G0, FALSE behandelt diesen Eintrag wie G01

bAccurateStop: Genauhalt (TRUE hat dieselbe Wirkung wie G09)

ST_NciGeoCirclePlane

Beschreibt einen Kreis in der Hauptebene. Der Mittelpunkt ist in absoluten Koordinaten angegeben.

Die orthogonale Komponente beim Mittelpunkt wird intern zugewiesen. Wenn z.B. ein Kreis in der XY-Ebene programmiert wird, so wird ‚`fCenterZ`' intern zugewiesen. Falls der Anwender den Wert explizit zugewiesen hat, wird der Wert dennoch vom Funktionsblock überschrieben. Mit der Programmierung der Höhe, lässt sich auch eine Helix beschreiben. Wird z.B. eine Helix in der XY-Ebene programmiert, wird die Hubhöhe der Helix mit ‚`fEndPosZ`' absolut vorgegeben.

```
TYPE ST_NciGeoCirclePlane :
STRUCT
    nEntryType: E_NciEntryType := E_NciEntryTypeGeoCirclePlane; (*do not override this parameter *)
    nDisplayIndex: UDINT;
    fEndPosX: LREAL;
    fEndPosY: LREAL;
    fEndPosZ: LREAL;
    fCenterX: LREAL;
    fCenterY: LREAL;
    fCenterZ: LREAL;
    fEndPosQ1: LREAL;
    fEndPosQ2: LREAL;
    fEndPosQ3: LREAL;
    fEndPosQ4: LREAL;
    fEndPosQ5: LREAL;
    fVelo: LREAL;
    bClockwise: BOOL;
    bAccurateStop: BOOL; (* VeloEnd := 0 *)
    nPlane: E_NciGeoPlane := E_NciGeoPlaneXY;
END_STRUCT
END_TYPE
```

nEntryType: Diesen Parameter nicht überschreiben (Typ: E_NciEntryType [▶ 308])

nDisplayIndex: Zu Anzeigezwecken, wie Satznummer im G-Code

fEndPosX: Zielposition X

fEndPosY: Zielposition Y

fEndPosZ: Zielposition Z

fCenterX: Mittelposition X in absoluten Koordinaten

fCenterY: Mittelposition Y in absoluten Koordinaten

fCenterZ: Mittelposition Z in absoluten Koordinaten

fEndPosQ1: Zielposition Q1

fEndPosQ2: Zielposition Q2

fEndPosQ3: Zielposition Q3

fEndPosQ4: Zielposition Q4

fEndPosQ5: Zielposition Q5

fVelo: Zielbahngeschwindigkeit in Basiseinheiten pro Sekunde (z.B. mm/s), wie F in G-Code

bClockwise: Wenn TRUE, wird der Kreis im Uhrzeigersinn gezogen, ansonsten gegen den Uhrzeigersinn (ähnlich wie G02, G03)

bAccurateStop: Genauhalt [▶ 144] (TRUE hat dieselbe Wirkung wie G09)

nPlane: Spezifiziert die Ebene: XY, YZ, oder ZX (ähnlich wie G17..G19) (Typ: E_NciGeoPlane [▶ 312])

> **Kreissegment als Startsegment**
>
> Ist das erste Geometrie-Segment ein Kreis, müssen Sie die Startposition mit ST_NciGeoStart [▶ 310] setzen.

E_NciGeoPlane

```
TYPE E_NciGeoPlane :
(
    E_NciGeoPlaneXY := 17,
    E_NciGeoPlaneZX := 18,
    E_NciGeoPlaneYZ := 19
);
END_TYPE
```

ST_NciGeoCircleCIP

Mit dem CIP-Kreis ist es möglich, einen Kreis frei im Raum zu beschreiben. Er muss sich nicht in der Hauptebene befinden. Damit der Kreis eindeutig beschrieben werden kann, dürfen alle 3 Punkte (Anfangspunkt ist implizit vorgegeben) nicht auf einer Geraden liegen. Es lässt sich also auf diese Weise kein Vollkreis programmieren.

```
TYPE ST_NciGeoCircleCIP :
STRUCT
    nEntryType: E_NciEntryType := E_NciEntryTypeGeoCircleCIP;    (* do not overwrite this parameter
*)
    nDisplayIndex: UDINT;
    fEndPosX:      LREAL;
    fEndPosY:      LREAL;
    fEndPosZ:      LREAL;
    fCIPPosX:      LREAL;
    fCIPPosY:      LREAL;
    fCIPPosZ:      LREAL;
    fEndPosQ1:     LREAL;
    fEndPosQ2:     LREAL;
    fEndPosQ3:     LREAL;
    fEndPosQ4:     LREAL;
    fEndPosQ5:     LREAL;
    fVelo:         LREAL;
    bAccurateStop: BOOL;    (* VeloEnd := 0 *)
END_STRUCT
END_TYPE
```

nEntryType: Diesen Parameter nicht überschreiben (Typ: E_NciEntryType [▶ 308])

nDisplayIndex: Zu Anzeigezwecken, wie Satznummer im G-Code

fCIPPosX: X-Position in absoluten Koordinaten (Punkt auf Kreisbahn)

fCIPPosY: Y-Position in absoluten Koordinaten (Punkt auf Kreisbahn)

fCIPPosZ: Z-Position in absoluten Koordinaten (Punkt auf Kreisbahn)

fEndPosX: Zielposition X

fEndPosY: Zielposition Y

fEndPosZ: Zielposition Z

fEndPosQ1: Zielposition Q1

fEndPosQ2: Zielposition Q2

fEndPosQ3: Zielposition Q3

fEndPosQ4: Zielposition Q4

fEndPosQ5: Zielposition Q5

fVelo: Zielbahngeschwindigkeit in Basiseinheiten pro Sekunde (z.B. mm/s), wie F in G-Code

bAccurateStop: Genauhalt [▶ 144] (TRUE hat dieselbe Wirkung wie G09)

Kreissegment als Startsegment

Ist das erste Geometrie-Segment ein Kreis, müssen Sie Startposition mit ST_NciGeoStart [▶ 310] setzen.

ST_NciGeoBezier3

Beschreibt eine Bezierkurve dritter Ordnung mit Hilfe von Kontrollpunkten. Die Startposition ergibt sich aus dem vorherigen Segment. Der dritte Kontrollpunkt wird durch die Zielposition vorgegeben.

```
TYPE ST_NciGeoBezier3:
STRUCT
    nEntryType: E_NciEntryType := E_NciEntryTypeGeoBezier3; (*do not override this parameter *)
    nDisplayIndex: UDINT;
    fControlPoint1X: LREAL;
    fControlPoint1Y: LREAL;
    fControlPoint1Z: LREAL;
    fControlPoint2X: LREAL;
    fControlPoint2Y: LREAL;
    fControlPoint2Z: LREAL;
    fEndPosX: LREAL;
    fEndPosY: LREAL;
    fEndPosZ: LREAL;
    fEndPosQ1: LREAL;
    fEndPosQ2: LREAL;
    fEndPosQ3: LREAL;
    fEndPosQ4: LREAL;
    fEndPosQ5: LREAL;
    fVelo: LREAL;
    bAccurateStop: BOOL; (* VeloEnd := 0 *)
END_STRUCT
END_TYPE
```

Eine Bezier3-Kurve ist nicht mit dem Typ ST_NciVertexSmoothing vom Typ Bezier 3. und 5. Ordnung kompatibel. In diesem Fall muss ein anderer Typ beim VertexSmoothing ausgewählt werden.

nEntryType: Diesen Parameter nicht überschreiben (Typ: E_NciEntryType [▶ 308])

nDisplayIndex: Zu Anzeigezwecken, wie Satznummer im G-Code

fControlPoint1X: X Komponente Kontrollpunkt 1

fControlPoint1Y: Y Komponente Kontrollpunkt 1

...

fControlPoint2Z: Z Komponente Kontrollpunkt 2

fEndPosX: Zielposition X

fEndPosY: Zielposition Y

fEndPosZ: Zielposition Z

fEndPosQ1: Zielposition Q1

fEndPosQ2: Zielposition Q2

fEndPosQ3: Zielposition Q3

fEndPosQ4: Zielposition Q4

fEndPosQ5: Zielposition Q5

fVelo: Zielbahngeschwindigkeit in Basiseinheiten pro Sekunde (z.B. mm/s), wie F in G-Code

bAccurateStop: Genauhalt [▶ 144] (TRUE hat dieselbe Wirkung wie G09)

ST_NciGeoBezier5

Beschreibt eine Bezierkurve 5ter Ordnung mit Hilfe von Kontrollpunkten. Die Startposition ergibt sich aus dem vorherigen Segment. Der fünfte Kontrollpunkt wird durch die Zielposition vorgegeben.

```
TYPE ST_NciGeoBezier5:
STRUCT
    nEntryType: E_NciEntryType := E_NciEntryTypeGeoBezier5; (*do not override this parameter *)
    nDisplayIndex: UDINT;
    fControlPoint1X: LREAL;
    fControlPoint1Y: LREAL;
    fControlPoint1Z: LREAL;
    fControlPoint2X: LREAL;
    fControlPoint2Y: LREAL;
    fControlPoint2Z: LREAL;
    fControlPoint3X: LREAL;
    fControlPoint3Y: LREAL;
    fControlPoint3Z: LREAL;
    fControlPoint4X: LREAL;
    fControlPoint4Y: LREAL;
    fControlPoint4Z: LREAL;
    fEndPosX: LREAL;
    fEndPosY: LREAL;
    fEndPosZ: LREAL;
    fEndPosQ1: LREAL;
    fEndPosQ2: LREAL;
    fEndPosQ3: LREAL;
    fEndPosQ4: LREAL;
    fEndPosQ5: LREAL;
    fVelo: LREAL;
    bAccurateStop: BOOL; (* VeloEnd := 0 *)
END_STRUCT
END_TYPE
```

nEntryType: Diesen Parameter nicht überschreiben(Typ: E_NciEntryType [▶ 308])

nDisplayIndex: Zu Anzeigezwecken, wie Satznummer im G-Code

fControlPoint1X: X Komponente Kontrollpunkt 1

fControlPoint1Y: Y Komponente Kontrollpunkt 1

...

fControlPoint4Z: Z Komponente Kontrollpunkt 4

fEndPosX: Zielposition X

fEndPosY: Zielposition Y

fEndPosZ: Zielposition Z

fEndPosQ1: Zielposition Q1

fEndPosQ2: Zielposition Q2

fEndPosQ3: Zielposition Q3

fEndPosQ4: Zielposition Q4

fEndPosQ5: Zielposition Q5

fVelo: Zielbahngeschwindigkeit in Basiseinheiten pro Sekunde (z.B. mm/s), wie F in G-Code

bAccurateStop: Genauhalt [▶ 144] (TRUE hat dieselbe Wirkung wie G09)

ST_NciMFuncHsk

Beschreibt eine M-Funktion [▶ 167] des Typs Handshake. Die M-Funktionsnummer liegt zwischen 0 und 159.

```
TYPE ST_NciMFuncHsk :
STRUCT
    nEntryType: E_NciEntryType := E_NciEntryTypeMFuncHsk; (*do not override this parameter *)
    nDisplayIndex: UDINT;
    nMFunc: INT;
END_STRUCT
END_TYPE
```

nEntryType: Diesen Parameter nicht überschreiben (Typ: E_NciEntryType [▶ 308])

nDisplayIndex: Zu Anzeigezwecken, wie Satznummer im G-Code

nMFunc: M-Funktionsnummer (0..159)

M-Funktionen in der PlcInterpolation-Bibliothek

Werden in der PlcInterpolation-Bibliothek M-Funktionen verwendet, müssen diese nicht im User Interface der XAE eingetragen werden. Eine M-Funktion wirkt immer an der programmierten Stelle.

ST_NciMFuncFast

Parametriert bis zu 8 schnelle M-Funktionen [▶ 167]. Die erste M-Funktion muss nMFuncIn0 zugewiesen werden, die zweite nMFuncIn1 usw. -1 gibt das Ende der Zuweisungen an.

```
TYPE ST_NciMFuncFast :
STRUCT
    nEntryType: E_NciEntryType := E_NciEntryTypeMFuncFast; (*do not override this parameter *)
    nDisplayIndex: UDINT;
    nMFuncIn0: INT;
    nMFuncIn1: INT;
    nMFuncIn2: INT;
    nMFuncIn3: INT;
    nMFuncIn4: INT;
    nMFuncIn5: INT;
    nMFuncIn6: INT;
    nMFuncIn7: INT;
END_STRUCT
END_TYPE
```

nEntryType: Diesen Parameter nicht überschreiben (Typ: E_NciEntryType [▶ 308])

nDisplayIndex: Zu Anzeigezwecken, wie Satznummer im G-Code

nMFuncIn0: schnelle M-Funktionsnummer (0..159)

nMFuncIn1: schnelle M-Funktionsnummer (0..159) -1 gibt das Ende der Liste an.

nMFuncIn2: schnelle M-Funktionsnummer (0..159) -1 gibt das Ende der Liste an.

nMFuncIn3: schnelle M-Funktionsnummer (0..159) -1 gibt das Ende der Liste an.

nMFuncIn4: schnelle M-Funktionsnummer (0..159) -1 gibt das Ende der Liste an.

nMFuncIn5: schnelle M-Funktionsnummer (0..159) -1 gibt das Ende der Liste an.

nMFuncIn6: schnelle M-Funktionsnummer (0..159) -1 gibt das Ende der Liste an.

nMFuncIn7: schnelle M-Funktionsnummer (0..159) -1 gibt das Ende der Liste an.

M-Funktionen in der PlcInterpolation-Bibliothek

Werden in der PlcInterpolation-Bibliothek M-Funktionen verwendet, müssen diese nicht im User Interface der XAE eingetragen werden. Eine M-Funktion wirkt immer an der programmierten Stelle.

ST_NciMFuncResetAllFast

Setzt alle schnellen M-Funktionen [▶ 167] zurück.

```
TYPE ST_NciMFuncResetAllFast :
STRUCT
    nEntryType: E_NciEntryType := E_NciEntryTypeMFuncResetAllFast; (*do not override this parameter
*)
    nDisplayIndex: UDINT;
END_STRUCT
END_TYPE
```

nEntryType: Diesen Parameter nicht überschreiben (Typ: E_NciEntryType [▶ 308])

nDisplayIndex: Zu Anzeigezwecken, wie Satznummer im G-Code

ST_NciHParam

Setzt einen H-Parameter [▶ 172] im zyklischen Kanalinterface.

```
TYPE ST_NciHParam :
STRUCT
    nEntryType: E_NciEntryType := E_NciEntryTypeHParam; (*do not override this parameter *)
    nDisplayIndex: UDINT;
    nHParam: UDINT;
END_STRUCT
END_TYPE
```

nEntryType: Diesen Parameter nicht überschreiben (Typ: E_NciEntryType [▶ 308])

nDisplayIndex: Zu Anzeigezwecken, wie Satznummer im G-Code

nHParam: H-Parameter von der NC zur SPS

ST_NciSParam

Setzt einen S-Parameter [▶ 172] im zyklischen Kanalinterface.

```
TYPE ST_NciSParam :
STRUCT
    nEntryType: E_NciEntryType := E_NciEntryTypeSParam; (*do not override this parameter *)
    nDisplayIndex: UDINT;
    nSParam: UINT;
END_STRUCT
END_TYPE
```

nEntryType: Diesen Parameter nicht überschreiben (Typ: E_NciEntryType [▶ 308])

nDisplayIndex: Zu Anzeigezwecken, wie Satznummer im G-Code

nSParam: S-Parameter von der NC zur SPS

ST_NciTParam

Setzt einen T-Parameter [▶ 172] im zyklischen Kanalinterface.

```
TYPE ST_NciTParam :
STRUCT
    nEntryType: E_NciEntryType := E_NciEntryTypeTParam; (*do not override this parameter *)
    nDisplayIndex: UDINT;
    nTParam: UINT;
END_STRUCT
END_TYPE
```

nEntryType: Diesen Parameter nicht überschreiben (Typ: E_NciEntryType [▶ 308])

nDisplayIndex: Zu Anzeigezwecken, wie Satznummer im G-Code

nTParam: T-Parameter von der NC zur SPS

ST_NciDynOvr

Modalfunktion zur Änderung der Bahndynamik.

Vergleiche DynOvr [▶ 179] in der Interpreter-Dokumentation [▶ 128].

```
TYPE ST_NciDynOvr :
STRUCT
    nEntryType: E_NciEntryType := E_NciEntryTypeDynOvr; (*do not override this parameter*)
    nDisplayIndex: UDINT;
    fDynOvr: LREAL;
END_STRUCT
END_TYPE
```

nEntryType: Diesen Parameter nicht überschreiben (Typ: E_NciEntryType [▶ 308])

nDisplayIndex: Zu Anzeigezwecken, wie Satznummer im G-Code

fDynOvr: Wert für dynamischen Override (0.01 < fDynOvr <= 1)

ST_NciVertexSmoothing

Modalfunktion zur Aktivierung der Verschleifung am Segmentübergang. Die Verschleifung wirkt, bis sie durch Setzen des Radius auf 0 wieder aufgehoben wird.

Es befindet sich eine detaillierte Beschreibung des Parameters in der Interpreter-Dokumentation [▶ 128]. (paramVertexSmoothing [▶ 155]).

```
TYPE ST_NciVertexSmoothing :
STRUCT
    nEntryType: E_NciEntryType := E_NciEntryTypeVertexSmoothing; (*do not override this parameter *)
    nDisplayIndex: UDINT;
    nType: UDINT; (*type of smoothing, e.g. parabola, bi-quad *)
    nSubtype: UDINT; (*e.g. adaptive, constant radius *)
    fRadius: LREAL; (*max. radius for tolerance ball *)
END_STRUCT
END_TYPE
```

nEntryType: Diesen Parameter nicht überschreiben (Typ: E_NciEntryType [▶ 308])

nDisplayIndex: Zu Anzeigezwecken, wie Satznummer im G-Code

nType: Verschleifungsart: 2: Parabel, 3: Bi-quadratisch, 4: Bezier 3. Ordnung, 5: Bezier 5. Ordnung

nSubtype: 1: konstanter Toleranzradius, 2: Abstand Schnittpunkt zum Scheitelpunkt, 3: adaptiver Toleranzradius

fRadius: Radius der Verschleifungskugel in Basiseinheiten (z.B. mm)

ST_NciBaseFrame

Die Struktur ST_NciBaseFrame beschreibt eine modale Nullpunktverschiebung und Rotation. Dabei ist die Wirkungsweise die gleiche wie die Nullpunktverschiebung und Rotation im Interpreter. D.h. der Punkt um den rotiert wird, ist der aktuelle Nullpunkt (vergl. Rotation [▶ 151] in der Interpreter-Dokumentation [▶ 128]).

```
TYPE ST_NciBaseFrame:
STRUCT
    nEntryType: E_NciEntryType := E_NciEntryTypeBaseFrame; (*Do not override this parameter *)
    nDisplayIndex: UDINT;
    fShiftX: LREAL;
    fShiftY: LREAL;
    fShiftZ: LREAL;
    fRotX: LREAL;
    fRotY: LREAL;
    fRotZ: LREAL;
    fShiftQ1: LREAL;
    fShiftQ2: LREAL;
    fShiftQ3: LREAL;
    fShiftQ4: LREAL;
    fShiftQ5: LREAL;
END_STRUCT
END_TYPE
```

nEntryType: Diesen Parameter nicht überschreiben (Typ: E_NciEntryType [▶ 308])

nDisplayIndex: Zu Anzeigezwecken, wie Satznummer im G-Code

fShiftX: Nullpunktverschiebung in X-Richtung

fShiftY: Nullpunktverschiebung in Y-Richtung

fShiftZ: Nullpunktverschiebung in Z-Richtung

fRotX: Rotation der X-Achse

fRotY: Rotation der Y-Achse

fRotZ: Rotation der Z-Achse

fShiftQ1: Offset der Q1-Achse

fShfitQ2: Offset der Q2-Achse

fShiftQ3: Offset der Q3-Achse

fShiftQ4: Offset der Q4-Achse

fShiftQ5: Offset der Q5-Achse

ST_NciPathDynamics

Die Struktur *ST_NciPathDynamics* setzt die Dynamik (Beschleunigung, Verzögerung, Ruck) auf der Bahn. Dabei ist die Wirkungsweise die gleiche wie *paramPathDynamics* im Interpreter (vergl. paramPathDynamics [▶ 179] in der Interpreter-Dokumentation [▶ 128]).

```
TYPE ST_NciPathDynamics:
STRUCT
    nEntryType: E_NciEntryType := E_NciEntryTypePathDynamics; (*do not override this parameter *)
    nDisplayIndex: UDINT;
    fAcc: LREAL;
    fDec: LREAL;
    fJerk: LREAL;
END_STRUCT
END_TYPE
```

nEntryType: Diesen Parameter nicht überschreiben (Typ: E_NciEntryType [▶ 308])

nDisplayIndex: Zu Anzeigezwecken, wie Satznummer im G-Code

fAcc: Maximal erlaubte Bahnbeschleunigung

fDec: Maximal erlaubte Bahnverzögerung

fJerk: Maximal erlaubter Bahnruck

ST_NciAxisDynamics

Die Struktur *ST_NciAxisDynamics* setzt die Dynamik (Beschleunigung, Verzögerung, Ruck) auf der Bahnachsen. Dabei ist die Wirkungsweise die gleiche wie *paramAxisDynamics* im Interpreter (vergl. paramAxisDynamics [▶ 179] in der Interpreter-Dokumentation [▶ 128])

```
TYPE ST_NciAxisDynamics:
STRUCT
    nEntryType: E_NciEntryType := E_NciEntryTypeAxisDynamics; (*Do not override this parameter*)
    nDisplayIndex: UDINT;
    nAxis: UDINT;
    fAcc: LREAL;
    fDec: LREAL;
    fJerk: LREAL;
END_STRUCT
END_TYPE
```

nEntryType: Diesen Parameter nicht überschreiben (Typ: E_NciEntryType [▶ 308])

nDisplayIndex: Zu Anzeigezwecken, wie Satznummer im G-Code

nAxis: Achse in der Interpolationsgruppe X:0 Y:1 Z:2 Q1:3 ... Q5:7

fAcc: Maximal erlaubte Achsbeschleunigung

fDec: Maximal erlaubte Achsverzögerung

fJerk: Maximal erlaubter Achsruck

ST_NciDwellTime

Mit der Struktur *ST_NciDwellTime* wird eine Verweilzeit in Sekunden eingeschaltet (vergl. Verweilzeit [▶ 143] in der Interpreter-Dokumentation [▶ 128])

```
TYPE ST_NciDwellTime:
STRUCT
    nEntryType: E_NciEntryType := E_NciEntryTypeDwellTime; (*Do not override this parameter *)
    nDisplayIndex: UDINT;
    fDwellTime: LREAL;
END_STRUCT
END_TYPE
```

nEntryType: Diesen Parameter nicht überschreiben (Typ: E_NciEntryType [▶ 308])

nDisplayIndex: Zu Anzeigezwecken, wie Satznummer im G-Code

fDwellTime: Verweilzeit in Sekunden

ST_NciFeedrateIpol

Mit der Struktur *ST_NciFeedrateIpol* kann die Vorschubinterpolation eingestellt werden (vergl. Vorschubinterpolation [▶ 144]).

```
TYPE ST_NciFeedrateIpol :
STRUCT
    nEntryType: E_NciEntryType := E_NciEntryTypeFeedrateIpol;(*Do not overwrite this parameter*)
    nDisplayIndex: UDINT;
    eFeedrateIpol: E_NciFeedrateIpol;(*E_NciFeedrateIpolConstant = FCONST,
E_NciFeedrateIpolLinear=FLIN *)
END_STRUCT
END_TYPE
```

nEntryType: Diesen Parameter nicht überschreiben (Typ: E_NciEntryType [▶ 308])

eFeedrateIpol: Legt den Vorschubinterpolationstyp fest.

```
TYPE E_NciFeedRateIpol :(
    E_NciFeedrateIpolConstant,
    E_NciFeedrateIpolLinear
)
END_TYPE
```

ST_NciTangentialFollowingDesc

Dies ist ein Modalbefehl zur Ein-/Ausschaltung der tangentialen Nachführung.

```
TYPE ST_NciTangentialFollowingDesc :
STRUCT
    nEntryType: E_NciEntryType := E_NciEntryTypeTfDesc; (*do not override this parameter *)
    bTangOn: BOOL;
    nTangAxis: E_NciAxesInGroup; (*axis used for tangential following *)
    nPathAxis1: E_NciAxesInGroup; (*describing the plane e.g. x*)
    nPathAxis2: E_NciAxesInGroup; (*e.g. y ==> g17, xy plane*)
    fOffset: LREAL; (*geo tangent is 0 degree, counting is mathmatical positive *)
    fCriticalAngle1: LREAL;
    nTfBehavior: E_TangentialFollowingBehavior; (*what to do if angle becomes bigger than critical
angle 1 *)
END_STRUCT
END_TYPE
```

nEntryType: Diesen Parameter nicht überschreiben (Typ: E_NciEntryType [▶ 308]).

bTangOn: Wenn TRUE, wird die tangentiale Nachführung eingeschaltet.

nTangAxis: Achse (Q1..Q5), die als Tangentialachse verwendet wird (Typ: E_NciAxesInGroup [▶ 320]).

nPathAxis1: Erste Bahnachse, die die Ebene und Ausrichtung zur Berechnung der Tangente beschreibt.

nPathAxis2: Zweite Bahnachse, die die Ebene und Ausrichtung zur Berechnung der Tangente beschreibt.

fOffset: Offset der Tangentialachse

fCriticalAngle1: Kritischer Winkel 1. Die Reaktion, wenn der Winkel zwischen zwei Segmenten größer ist als fCriticalAngle1, wird mit nTfBehavior spezifiziert.

nTfBehavior: siehe fCriticalAngle1 (Typ: E_TangentialFollowingBehavior [▶ 320])

E_NciAxesInGroup

```
TYPE E_NciAxesInGroup :
(
    NoneAxis := 0,
    XAxis,
    YAxis,
    ZAxis,
    Q1Axis,
    Q2Axis,
    Q3Axis,
    Q4Axis,
    Q5Axis
);
END_TYPE
```

E_TangentialFollowingBehavior

```
TYPE E_TangentialFollowingBehavior :
(
    E_TfIngoreAll, (*ignore critical angle *)
    E_TfErrorOnCritical1 (*if angle becomes bigger than critical angle 1 ==> error *)
);
END_TYPE
```

E_TfIngoreAll: Der kritische Winkel wird ignoriert.

E_TfErrorOnCritical1: Wird der kritische Winkel überschritten, so wird ein Fehler zurückgegeben.

ST_NciEndOfTables

Gibt den letzten Eintrag der letzten Tabelle an. Wird verwendet zur Signalisierung der bChannelDone-Flag in FB_NciFeedTable [▶ 307].

```
TYPE ST_NciEndOfTables :
STRUCT
    nEntryType: E_NciEntryType := E_NciEntryTypeEndOfTables; (*do not override this parameter *)
END_STRUCT
END_TYPE
```

nEntryType: Diesen Parameter nicht überschreiben (Typ: E_NciEntryType [▶ 308])

7 Beispiele

NCI: NCISimpleSample

Download:

https://infosys.beckhoff.com/content/1031/TF5100_TC3_NC_I/Resources/3438746891.zip

Das Beispiel NCISimpleSample zeigt, wie ein G-Code-Programm aus der SPS geladen und die Abarbeitung gestartet wird.

Voraussetzung ist, dass Sie das beigefügte Teileprogramm first.nc in das TwinCAT\Mc\Nci-Verzeichnis kopieren, ansonsten wird das Teileprogramm beim Laden nicht gefunden. Alternativ können Sie den Pfad im SPS-Programm anpassen.

PLC Interpolation: PlcInterpolationSimpleSample

Download:

https://infosys.beckhoff.com/content/1031/TF5100_TC3_NC_I/Resources/2944140171.zip

Das Beispiel zeigt, wie eine Bewegung mit der Bibliothek Tc2_PlcInterpolation direkt aus der SPS abgefahren werden kann.

8 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

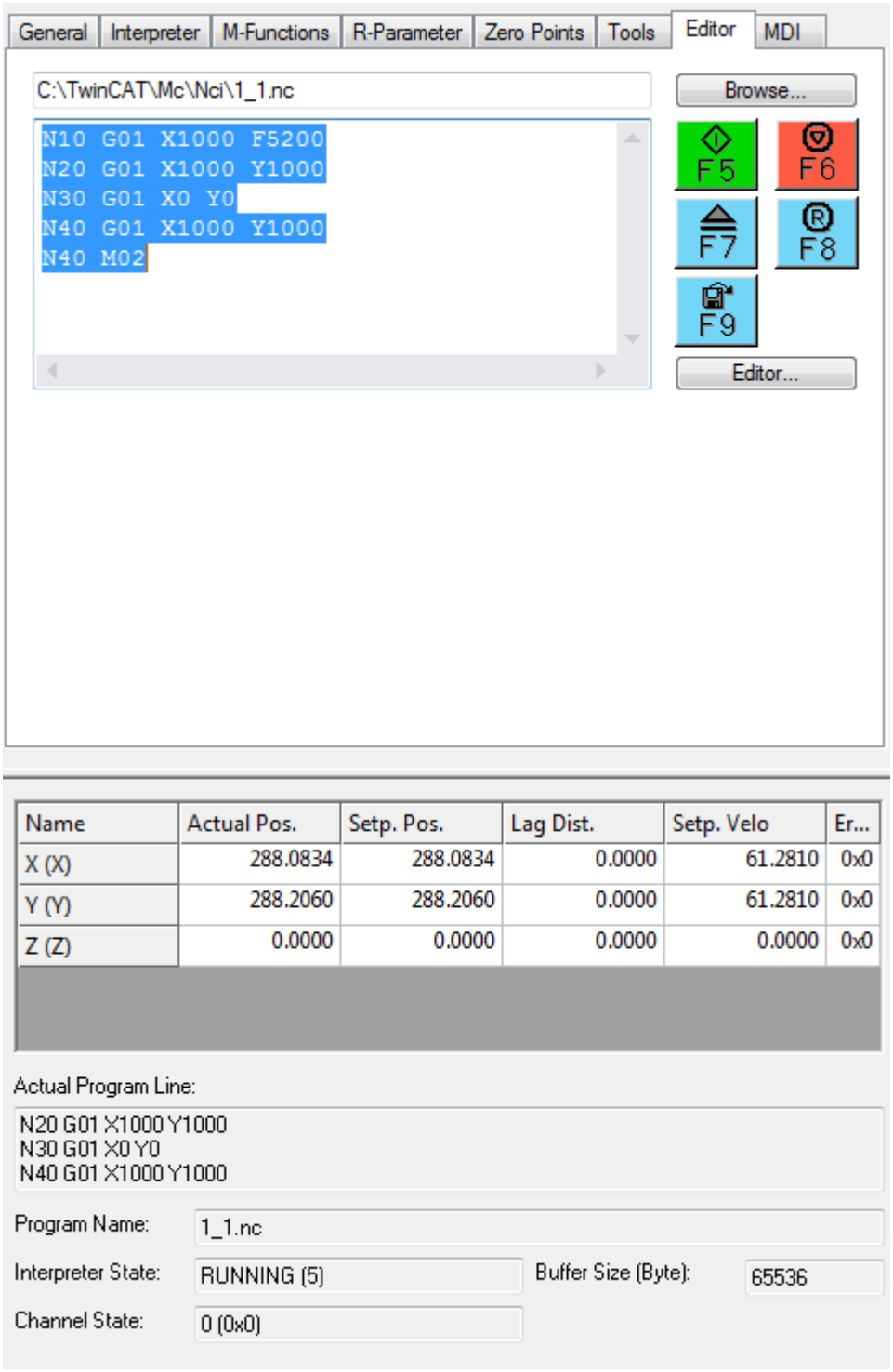
9 Anhang

9.1 Anzeige des Teileprogramms

Auslesen der aktuellen NC-Zeile via ADS

Mit diesem ADS-Read-Befehl werden maximal 3 Zeilen des aktuellen Teileprogramms zurückgegeben. Dabei handelt es sich um die aktuelle Programmzeile und ggf. zwei zuvor abgearbeitete Zeilen.

Funktion	ADS-Read
Port	500 (dez)
Index Group	0x2300 + Kanal ID
Index Offset	0x2000 0001
Daten	String (min. 30 byte)



Auslesen des aktuellen Programmnamens

Mit diesem ADS-Read-Befehl wird der Programmname des aktuellen NC-Hauptprogramms zurückgegeben (hier 1_1.nc).

Funktion	ADS-Read
Port	500 (dez)
Index Group	0x2100 + Kanal ID
Index Offset	0x7
Daten	String, max. 100 Zeichen

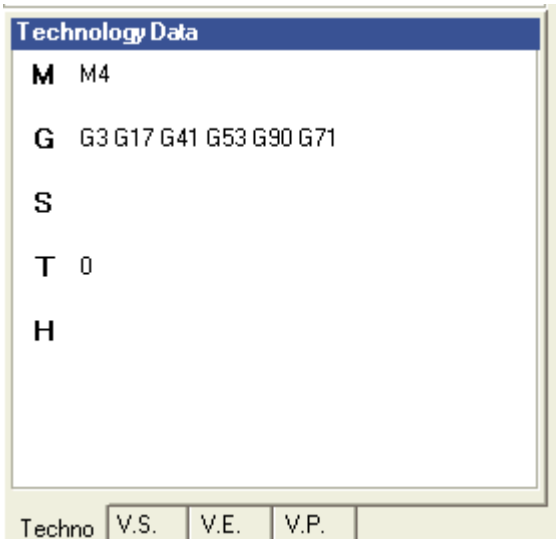
Auslesen der aktuellen Datei-Information

Im Gegensatz zu der Funktion 'Auslesen der aktuellen NC-Zeile' wird hier nicht die Zeile, sondern eine Zeileninformation ausgelesen. D.h. als Rückgabewert erhält man den aktuellen Programmnamen (z.B. Dateiname des Unterprogramms) und einen Dateioffset. Die Bedienoberfläche kann mit dieser Information die dazugehörige Datei öffnen und die entsprechende Zeile hervorheben. Damit ist man bei der Darstellung nicht mehr auf 3 Zeilen beschränkt, sondern kann beliebig viele Zeile zur Anzeige bringen.

Für den Fall, dass in der NCI ein Lade- bzw. Laufzeitfehler aufgetreten ist, können über diesen Weg Informationen zu der dazugehörigen Programmzeile erfragt werden.

Funktion	ADS-Read	
Port	500 (dez)	
Index Group	0x2100 + Kanal ID	
Index Offset	0x12	
Daten	UINT32	Aktuelle Anzeige von 1: SAF- 2: Interpreter 3: Fehleroffset
	UINT32	Dateioffset
	char[260]	Pfad + Programmname

9.2 Anzeige von Technologiedaten



Die aktuell wirkenden Technologiedaten wie G-Funktionen, Nullpunktverschiebungen und Rotation können via ADS ausgelesen werden.

Aktivierung zum Auslesen der Technologiedaten

Um die oben genannten Parameter auszulesen, ist zuvor eine Aktivierung via ADS erforderlich.

Die Funktion muss spätestens vor dem Start des NC-Programms aktiviert werden. Sie bleibt solange aktiv, bis entweder ein TwinCAT Restart durchgeführt oder die Funktion explizit zurückgesetzt wird.

Funktion	ADS-Write
Port	500 (dez)
Index Group	0x2000 + Kanal ID
Index Offset	0x0053
Daten	DWORD 0: disable (default) 1: enable

Auslesen der aktuell wirkenden Nullpunktverschiebung

Mit diesem Befehl wird die wirkende Nullpunktverschiebung (NPV) des aktuell in der Satzausführung (SAF) verfahrenen Segments ausgelesen. Wenn keine NPV aktiv ist (G53), so enthält die Struktur für die einzelnen Komponenten einen Null-Vektor. Diese Daten können z.B. für eine Umschaltung der Anzeige zwischen Maschinen-Koordinaten und Programmierten-Koordinaten verwendet werden.

Die Daten, die z.B. mit dem Funktionsbaustein 'ItpReadZeroShift' ausgelesen werden, können sich von diesen Werten unterscheiden, da mit dem Funktionsbaustein die Interpreter-Daten ausgelesen werden und dieser ggf. schon neue Verschiebungen berücksichtigt.

Funktion	ADS-Read	
Port	500 (dez)	
Index Group	0x2100 + Kanal ID	
Index Offset	0x0014	
Daten	{	
	UINT32	Satzzähler
	UINT32	dummy
	LREAL[3]	Nullpunktverschiebung G54..G57
	LREAL[3]	Nullpunktverschiebung G58
	LREAL[3]	Nullpunktverschiebung G59
	}	

Auslesen der aktuell wirkenden Rotation

Mit diesem Befehl wird die wirkende Rotation des aktuell in der SAF verfahrenen Segments ausgelesen.

Funktion	ADS-Read	
Port	500 (dez)	
Index Group	0x2100 + Kanal ID	
Index Offset	0x0015	
Daten	{	
	UINT32	Satzzähler
	UINT32	dummy
	LREAL[3]	Rotation in Grad von X, Y & Z
	}	

Auslesen des aktuell wirkenden G-Codes

Der G-Code ist in Gruppen unterteilt. So bilden z. B. die modal wirkenden Geometrietypen (G01, G02...) und die Ebenenanwahl (G17..G19) eigene Gruppen. Mit dem Auslesen der G-Code-Information wird der Enumerator für die Gruppen ausgelesen. Diese können dann applikationsspezifisch zur Anzeige gebracht werden.

Da mit dem Read-Befehl eine zu lesende Größe mitgegeben wird, müssen nicht alle Gruppen ausgelesen werden. Es wird immer von Gruppe 1 der mitgelieferte Speicher gefüllt. Werden z. B. 3x8 Byte als Speichergröße übertragen, so werden die Daten des Satzzählers, Gruppe 1 und 2 zurückgeliefert.

Funktion	ADS-Read	
Port	500 (dez)	
Index Group	0x2100 + Kanal ID	
Index Offset	0x0013	
Daten	{	
	UINT32	Satzzähler
	UINT32	Gruppe 1: ModalGeoTypes
	UINT32	Gruppe 2: BlockwiseGeoTypes
	UINT32	Gruppe 3: ModalPlaneSelection

	UINT32	Gruppe 4: ModalToolCompensation
	UINT32	Gruppe 5: ModalToolFeedDirection
	UINT32	Gruppe 6: ModalZeroShift
	UINT32	Gruppe 7: ModalAccurateStop
	UINT32	Gruppe 8: BlockwiseAccurateStop
	UINT32	Gruppe 9: ModalDesignationAbsInc
	UINT32	Gruppe 10: ModalDesignationInchMetric
	UINT32	Gruppe 11: ModalFeedRateInCurve
	UINT32	Gruppe 12: ModalCenterpointCorr
	UINT32	Gruppe 13: ModalCircleCpAbsInc
	UINT32	Gruppe 14: ModalCollisionDetection
	UINT32	Gruppe 15: ModalRotation
	UINT32	Gruppe 16: ModalCalcExRot
	UINT32	Gruppe 17: ModalDiam
	UINT32	Gruppe 18: ModalFeedrateIpol
	UINT32	Gruppe 19: ModalMirror
	}	

```
#define GCodeOffset 0x1000
#define CommonIdentOffset 0x2000 // used for non-g-code commands, like rot, cfc...
```

Gruppe 1: ModalGeoTypes

```
enum GCodeGroup_ModalGeoTypes
{
    ModalGeoTypeUndefined = 0,
    ModalGeoTypeG0  = 0 + GCodeOffset, // line - rapid traverse
    ModalGeoTypeG01 = 1 + GCodeOffset, // straight line
    ModalGeoTypeG02 = 2 + GCodeOffset, // circle clockwise
    ModalGeoTypeG03 = 3 + GCodeOffset  // circle anticlockwise
};
```

Gruppe 2: BlockwiseGeoTypes

```
enum GCodeGroup_BlockwiseGeoTypes
{
    BlockwiseGeoTypeNone = 0,
    BlockwiseGeoTypeG04  = 4  + GCodeOffset,      // dwell time
    BlockwiseGeoTypeG74  = 74 + GCodeOffset,      // homing
    BlockwiseGeoTypeCip  = 1  + CommonIdentOffset // circle parametrized with 3 points
};
```

Gruppe 3: ModalPlaneSelection

```
enum GCodeGroup_ModalPlaneSelection
{
    ModalPlaneSelectUndefined = 0,
    ModalPlaneSelectG17       = 17 + GCodeOffset, // xy-plane
    ModalPlaneSelectG18       = 18 + GCodeOffset, // zx-plane
    ModalPlaneSelectG19       = 19 + GCodeOffset  // yz-plane
};
```

Gruppe 4: ModalToolCompensation

```
enum GCodeGroup_ModalToolCompensation
{
    ModalToolCompUndefined = 0,
    ModalToolCompG40       = 40 + GCodeOffset, // tool compensation off
    ModalToolCompG41       = 41 + GCodeOffset, // tool compensation left
    ModalToolCompG42       = 42 + GCodeOffset  // tool compensation right
};
```

Gruppe 5: ModalToolFeedDirection

```
enum GCodeGroup_ModalToolFeedDirection
{
    ModalToolFeedDirUndefined = 0,
    ModalToolFeedDirPos       = 2 + CommonIdentOffset, // tool feed direction positive
    ModalToolFeedDirNeg       = 3 + CommonIdentOffset  // tool feed direction negative
};
```

Gruppe 6: ModalZeroShift

```
enum GCodeGroup_ModalZeroShift
{
    ModalZeroShiftUndefined = 0,
    ModalZeroShiftG53       = 53 + GCodeOffset, // zero shift off
    ModalZeroShiftG54G58G59 = 54 + GCodeOffset, // zero shift G54 + G58+ G59
    ModalZeroShiftG55G58G59 = 55 + GCodeOffset, // zero shift G55 + G58+ G59
    ModalZeroShiftG56G58G59 = 56 + GCodeOffset, // zero shift G56 + G58+ G59
    ModalZeroShiftG57G58G59 = 57 + GCodeOffset  // zero shift G57 + G58+ G59
};
```

Gruppe 7: ModalAccurateStop

```
enum GCodeGroup_ModalAccurateStop
{
    ModalAccurateStopNone = 0,
    ModalAccurateStopG60  = 60 + GCodeOffset // modal accurate stop
};
```

Gruppe 8: BlockwiseAccurateStop

```
enum GCodeGroup_BlockwiseAccurateStop
{
    BlockwiseAccurateStopNone = 0,
    BlockwiseAccurateStopG09  = 9 + GCodeOffset,      // common accurate stop
    BlockwiseAccurateStopTpm  = 4 + CommonIdentOffset // target position monitoring
};
```

Gruppe 9: ModalDesignationAbsInc

```
enum GCodeGroup_ModalDesignationAbsInc
{
    ModalDesignAbsIncUndefined = 0,
    ModalDesignAbsIncG90       = 90 + GCodeOffset, // absolute designation
    ModalDesignAbsIncG91       = 91 + GCodeOffset  // incremental designation
};
```

Gruppe 10: ModalDesignationInchMetric

```
enum
GCodeGroup_ModalDesignationInchMetric
{
    ModalDesignInchMetricUndefined = 0,
    ModalDesignInchMetricG70       = 70  + GCodeOffset, // designation inch
    ModalDesignInchMetricG71       = 71  + GCodeOffset, // designation metric
    ModalDesignInchMetricG700      = 700 + GCodeOffset, // designation inch & feedrate recalculated
    ModalDesignInchMetricG710      = 710 + GCodeOffset  // designation metric & feedrate
recalculated
};
```

Gruppe 11: ModalFeedRateInCurve

```
enum GCodeGroup_ModalFeedRateInCurve
{
    ModalFeedRateInCurveUndefined = 0,
    ModalFeedRateInCurveCfc       = 5 + CommonIdentOffset, // constant feed contour
    ModalFeedRateInCurveCfin      = 6 + CommonIdentOffset, // constant feed inner contour
    ModalFeedRateInCurveCftcp     = 7 + CommonIdentOffset  // constant feed tool center point
};
```

Gruppe 12: ModalCenterpointCorr

```
enum GCodeGroup_ModalCenterpointCorr
{
    ModalCenterpointCorrUndefined = 0,
    ModalCenterpointCorrOn        = 8 + CommonIdentOffset, // circle centerpoint correction on
    ModalCenterpointCorrOff       = 9 + CommonIdentOffset  // circle centerpoint correction off
};
```

Gruppe 13: ModalCircleCpAbsInc

```
enum GCodeGroup_ModalCircleCpAbsInc
{
    ModalCircleCpUndefined   = 0,
    ModalCircleCpIncremental = 10 + CommonIdentOffset, // circle centerpoint incremental to start
point
    ModalCircleCpAbsolute    = 11 + CommonIdentOffset  // circle centerpoint absolute
};
```

Gruppe 14: ModalCollisionDetection

```
enum GCodeGroup_ModalCollisionDetection
{
    ModalCollisionDetectionUndefined = 0,
    ModalCollisionDetectionOn        = 12 + CommonIdentOffset, //collision detection on
    ModalCollisionDetectionOff       = 13 + CommonIdentOffset  //collision detection off
};
```

Gruppe 15: ModalRotation

```
enum GCodeGroup_ModalRotation
{
    ModalRotationUndefined = 0,
    ModalRotationOn        = 14 + CommonIdentOffset, // rotation is turned on
    ModalRotationOff       = 15 + CommonIdentOffset  // rotation is turned off
};
```

Gruppe 16: ModalCalcExRot

```
enum GCodeGroup_ModalCalcExRot
{
    ModalCalcExRotUndefined = 0,
    ModalCalcExRotOn        = 16 + CommonIdentOffset, // extended calculation for rotation turned on
    ModalCalcExRotOff       = 17 + CommonIdentOffset  // extended calculation for rotation turned
off
};
```

Gruppe 17: ModalDiam

```
enum GCodeGroup_ModalDiam
{
    ModalDiamUndefined = 0,
    ModalDiamOn        = 18 + CommonIdentOffset, // diameter programming on
    ModalDiamOff       = 19 + CommonIdentOffset  // diameter programming off
};
```

Group 18: ModalFeedrateIpol

```
enum GCodeGroup_ModalFeedrateIpol
{
    ModalFeedrateIpolUndefined = 0,
    ModalFeedrateIpolConst     = 20 + CommonIdentOffset, // federate interpolation constant
(default)
    ModalFeedrateIpolLinear    = 21 + CommonIdentOffset  // federate interpoaltion linear to
remaining path
};
```

Group 19: ModalMirror

```
enum GCodeGroup_ModalMirror
{
    // value - (32+CommonIdentOffset) shows the bitmask for mirrored axes
    // that's why the sequence seems to be strange...

    ModalMirrorUndefined = 0,
    ModalMirrorOff       = 32 + CommonIdentOffset,
    ModalMirrorX         = 33 + CommonIdentOffset,
    ModalMirrorY         = 34 + CommonIdentOffset,
    ModalMirrorXY        = 35 + CommonIdentOffset,
    ModalMirrorZ         = 36 + CommonIdentOffset,
    ModalMirrorZX        = 37 + CommonIdentOffset,
    ModalMirrorYZ        = 38 + CommonIdentOffset,
    ModalMirrorXYZ       = 39 + CommonIdentOffset
};
```

9.3 Anzeige der verbleibenden Bahnlänge

Wenn die Berechnung der verbleibenden Bahnlänge aktiv geschaltet ist, wird diese bis zum nächsten Genauhalt bzw. bis zum letzten Geometriesegment im Speicher (Satzvorbereitung) berechnet. Ein Genauhalt wird z.B. mit G09 bzw. G60 generiert. Aber auch M-Funktionen vom Typ Handshake, Dekodierstopps und G04 erzeugen implizit ein Genauhalt.

Aktivierung:

Index Group: 0x3000 + Group ID
Index Offset: 0x0508

vergl. Index Offset Spezifikation für Gruppen-Parameter

Auslesen der verbleibenden Bahnlänge:
Das Auslesen erfolgt ebenfalls via ADS und kann auch mit dem TwinCAT Scope aufgezeichnet werden.

Index Group: 0x3100 + Group ID
Index Offset: 0x0522

Außerdem kann die verbleibende Bahnlänge durch ItpSetCyclicLrealOffsets [▶ 243] mit dem zyklischen Kanalinterface in die SPS übertragen werden.
vergl. Index Offset Spezifikation für Gruppen-Zustand

9.4 Parametrierung

Die Parametrierung der NCI umfasst die Standard-Dynamikparameter (Beschleunigung, Verzögerung, Ruck) und deren online Wechsel, sowie die Mindestgeschwindigkeit und die Parameter zur Reduktion der Bahngeschwindigkeit und deren online Wechsel.

Generelles Verhalten an Segmentübergängen

- Geschwindigkeit: Die Segmentsollgeschwindigkeit VS wechselt vom Segmentübergang von VS_in in VS_out. Am Segmentübergang wird die Geschwindigkeit immer auf den niedrigeren der beiden Werte reduziert.
- Beschleunigung: die aktuelle Bahnbeschleunigung wird am Segmentübergang immer auf $a = 0$ geführt.
- Ruck: der Ruckeinheit *J* wird in Abhängigkeit von der Geometrie am Segmentübergang gewechselt. Das kann einen merkbaren Dynamiksprung bedingen.
- Es gibt die Möglichkeit der Glättung von Segmentübergängen. [▶ 131]

Tab. 1: NCI-Gruppen-Parameter

Parameter	Bedeutung und Randbedingungen
Kurvengeschwindigkeitsreduktionsmodus [▶ 332]	Coulomb, Cosinus oder VELOJUMP
Mindestgeschindigkeit [▶ 331]	Bahngeschwindigkeit die (bis auf Spitzen mit Bewegungsumkehr) nicht unterschritten werden darf: $V\_min \geq 0.0$
Reduktionsmethode für C1-Übergänge [▶ 331]	Reduktionsfaktor für *C1*-Übergänge: $C1 \geq 0.0$
VELOJUMP: C0-Reduktionsfaktoren *C0X*, C0Y, C0Z	Reduktionsfaktoren für *C0*-Übergänge für *X*-, Y-, *Z*-Achse: $C0X \geq 0.0$, $C0Y \geq 0.0$, $C0Z \geq 0.0$ *(Achsparameter, Online-Änderung im* Interpreter [▶ 180] möglich).
DEVIATIONANGLE: -Reduktionsfaktor *C0 C0*	Bahn-Reduktionsfaktor für *C0*-Übergänge: $1.0 \geq C0 \geq 0.0$
DEVIATIONANGLE: Kritischer Winkel (tief) φ_l	Winkel ab dem eine Geschwindigkeitsreduktion am Segmentübergang einsetzt: $0 \leq \varphi\_l < \varphi\_h \leq \pi$
DEVIATIONANGLE: Kritischer Winkel (hoch) φ_h	Winkel ab dem die Geschwindigkeit am Segmentübergang (*v_link) auf 0.0* reduziert wird: $0 \leq \varphi\_l < \varphi\_h \leq \pi$

Parameter	Bedeutung und Randbedingungen
Toleranzkugelradius [▶ 158] *TBR*	Radius der Toleranzkugeln: 1000.0 mm ≥*TBR* ≥ *0.1 mm*
C2-Reduktionsfaktor [▶ 180] *C2*	Reduktionsfaktor für geglättete Übergänge: *C2* ≥ *0.0*
Globale Software-Endlagen der Bahn [▶ 332]	Schaltet die Überwachung der globalen Software-Endlagen für die Bahnachsen

Mindestgeschwindigkeit

Jede NCI-Gruppe hat eine Mindestbahngeschwindigkeit *V_min*≥ *0.0* die niemals unterschritten werden sollte. Vom Benutzer vorgegebene Ausnahmen sind: programmierter Halt am Segmentübergang, Bahnende und Overrideanforderungen die unter die Mindestgeschwindigkeit führen. Systembedingte Ausnahme ist eine Bewegungsumkehr. Bei der Reduktionsmethode DEVIATIONANGLE: gilt für den Ablenkungswinkel $\varphi \geq \varphi\_h$, dann wird die Mindestgeschwindigkeit nicht beachtet. *V_min* muss kleiner als die Bahnsollgeschwindigkeit (F-Wort) jedes Segments sein.

Die Mindestgeschwindigkeit kann jederzeit im NC-Programm auf einen neuen Wert *V_min*≥ *0.0* in Einheiten *mm/sec* gesetzt werden.

Klassifikation der Segmentübergänge

Im Allgemeinen ist der Übergang von einem Segment zum nächsten nicht beliebig glatt, so dass zur Vermeidung von dynamischen Instabilitäten die Geschwindigkeit im Übergangspunkt geeignet reduziert werden muss. Dazu werden die Übergänge geometrisch klassifiziert und in drei Schritten die effektive Übergangsgeschwindigkeit V_link festgelegt.

Segmente - als geometrische Objekte - werden hier als Kurven im Sinne der Differentialgeometrie aufgefasst, die mittels der Bogenlänge parametriert sind.
Ein Segmentübergang von einem Segment *S_in* zu einem Segment *S_out* heißt vom geometrischen Typ *Ck*, wobei *k* eine natürliche Zahl (inklusive *0*) ist, wenn jedes Segment *k* mal stetig nach der Bogenlänge differenzierbar ist und die *k*-ten Ableitungen am Übergangspunkt übereinstimmen.

C0-Übergänge haben am Übergangspunkt einen Knick.

C1-Übergänge sehen glatt aus, sind aber dynamisch nicht glatt. Ein Beispiel ist der Übergang Gerade-Halbkreis im Stadion: am Übergangspunkt gibt es einen Sprung in der Beschleunigung.

C2-Übergänge (und natürlich *Ck*-Übergänge mit *k* > *2*) sind dynamisch glatt (ruckbegrenzt).

Reduktionsmethode für C2-Übergänge

Wie an allen Übergängen wird an *C2*-Übergängen *V_link* gleich dem Minimum der beiden Segmentsollgeschwindigkeiten gesetzt: *V_link* = *min(V_in,V_out)*. Weiter wird nicht reduziert.

Reduktionsmethode für C1-Übergänge

Zuerst wird*V_link* gleich dem Minimum der beiden Segmentsollgeschwindigkeiten gesetzt: *V_link* = min(*V_in,V_out*). In Abhängigkeit von den Geometrietypen *G_in* und *G_out* auf den zu verbindenden Segmenten und den Ebenenanwahlen auf *G_in* und *G_out* wird der geometrisch induzierte absolute Beschleunigungssprung *AccJump* im Segmentübergang unter der Geschwindigkeit *V_link* berechnet. Ist dieser größer als *C1* mal der für die Geometrien und Ebenen zulässigen Bahn-Beschleunigung / (absolute)Verzögerung *AccPathReduced*, dann wird die Geschwindigkeit *V_link* so reduziert, dass der sich ergebende Beschleunigungssprung gleich *AccPathReduced* ist. Ist dieser Wert kleiner als *V_min*, dann hat *V_min* Priorität.

Hinweis **Bei Wechsel der Dynamikparameter ändert sich automatisch die für die Geometrien und Ebenen zulässige Bahn-Beschleunigung und damit das Verhalten der Reduktion.**

Interface: XAE [▶ 23] und Interpreter [▶ 180]

Reduktionsmodi für C0-Übergänge

Es gibt für *C0*-Übergänge mehrere Reduktionsmethoden. Die Reduktionsmethode VELOJUMP reduziert die Geschwindigkeit nach erlaubten Geschwindigkeitssprüngen pro Achse. Die Reduktionsmethode DEVIATIONANGLE reduziert die Geschwindigkeit in Abhängigkeit vom Ablenkungswinkel φ (Winkel zwischen der normierten End-Tangente *T_in* des einlaufenden Segments *S_in* und der normierten Anfangs-Tangente *T_out* des auslaufenden Segments *S_out*). Die Cosinus Reduktionsmethode ist ein rein geometrisches Verfahren (siehe Kurvengeschwindigkeitsreduktionsmethode [▶ 24]).

Für mechanisch unabhängige Achsen empfiehlt sich die Methode VELOJUMP, während für mechanisch gekoppelte Achsen (z.B. ist die *Y*-Achse auf der *X*-Achse befestigt) in der Regel die Methode DEVIATIONANGLE empfehlenswert ist.

Reduktionsmethode für C0-Übergänge: VELOJUMP

Sei *V_link = min(V_in,V_out)* und für jede Achse *V_jump[i] = C0[i] * min(A+[i],-A-[i]) * T* der erlaubte absolute Geschwindigkeitssprung der Achse *[i]*, wobei *C0[i]* der Reduktionsfaktor und *A+[i], A-[i]* dieBeschleunigungs-/Verzögerungsbegrenzungen der Achse *[i]*, sowie *T* die Zykluszeit ist. Die Reduktionsmethode VELOJUMP sorgt dafür, dass die Bahngeschwindigkeit am Segmentübergang *V_link* soweit reduziert wird, dass der absolute Sprung in der Achssollgeschwindigkeit der Achse *[i]* höchstens *V_jump[i]* ist. Allerdings hat *V_min* Priorität: ist *V_link* kleiner als *V_min*, dann wird *V_link = V_min* gesetzt. Bei Bewegungsumkehr ohne programmierten Halt springt die Achsgeschwindigkeit.

Hinweis Bei Wechsel der Dynamikparameter werden die maximal erlaubten Achsgeschwindigkeitssprünge automatisch mitgewechselt.

Reduktionsmethode für C0-Übergänge: DEVIATIONANGLE

Hinweis Bei Wechsel der Dynamikparameter werden die Reduktionsfaktoren nicht automatisch mitgewechselt.

Wechsel der Parameter für C0-Übergänge: DEVIATIONANGLE

Tab. 2: Parameter

Parameter	Bedeutung und Randbedingungen
DEVIATIONANGLE: -Reduktionsfaktor *C0 C0*	Bahn-Reduktionsfaktor für *C0*-Übergänge: 1.0 ≥ *C0* ≥ *0.0*
DEVIATIONANGLE: Kritischer Winkel (tief) *φ_l*	Winkel ab dem die Reduktion einsetzt: *0* ≤ *φ_l* < *φ_h* ≤ *π*
DEVIATIONANGLE: Kritischer Winkel (hoch) *φ_h*	Winkel ab dem auf *v_link = 0.0* reduziert wird: *0* ≤ *φ_l* < *φ_h* ≤ *π*

Interface: Interpreter [▶ 180]

Reduktionsmethode Cosinus

Siehe hier [▶ 24].

Toleranzkugelradius und C2-Reduktionsfaktor

Diese Parameter werden unter dem Thema Glättung von Segmentübergängen [▶ 131] beschrieben.

Globale Software-Endlagen der Bahn

Mit der Funktion *'Globale Software-Endlagenüberwachung der Bahn' gibt* es zwei verschiedene Arten der Endlagenüberwachung.

Endlagenüberwachung der SAF-Task

Diese Art der Endlagenüberwachung ist immer dann aktiv, wenn für die Achse die Endlagen aktiv geschaltet sind (Achsparameter). Die Überprüfung erfolgt Komponentenweise aus der SAF-Task. D.h. tritt eine Verletzung der Endlage auf, so wird die Bahngeschwindigkeit sofort auf 0 gesetzt und die gesamte Interpolationsgruppe hat einen Fehler.

Die Aktivierung dieser Überwachung erfolgt über die Achsparameter und **nicht** über den hier beschriebenen Gruppenparameter.

Software-Endlagen der Bahn

Um zu verhindern, dass bei einer Verletzung der Software-Endlagen die Bahngeschwindigkeit sofort auf 0 gesetzt wird, muss die Funktion *'Globale Software-Endlagenüberwachung der Bahn'* aktiviert werden. Ist diese aktiv, wird ausschließlich bis zu dem NC-Satz verfahren, in dem die Endlagen verletzt werden. Die Geschwindigkeit wird über eine Rampe abgebremst.

- Damit eine Überprüfung nur für die gewünschten Bahnachsen durchgeführt wird, müssen die Software-Endlagen der Achskomponenten angewählt sein (Achsparameter).
- Die Überprüfung wird für Standardgeometriesegmente durchgeführt. Dazu zählen

  Gerade

  Kreis

  Helix
- Verschleifungen mit Splines werden nicht überwacht. Die Splines sind sollwertseitig immer innerhalb der Toleranzkugel. Ansonsten greift noch die Endlagenüberwachung der SAF-Task.
- Da eine sinnvolle und allgemeingültige Überprüfung der Endlagen erst zur Laufzeit des NC-Programms durchgeführt werden kann (vor dem Lookahead), ist es möglich, dass die Bahnachsen bis ausschließlich zu dem NC-Satz verfahren, wo die Endlagen verletzt werden.
- Befindet man sich aus irgendeinem Grund außerhalb der Software-Endlagen, so kann mit einer Geraden wieder in den gültigen Bereich hineingefahren werden.

Parametrierung:

XAE: Gruppen Parameter [▶ 23]

9.4.1 Bahnoverride (Interpreter-Overridetypen)

Beim Bahnoverride handelt es sich um einen Geschwindigkeitsoverride. D.h. eine Overrideänderung bewirkt eine neue Geschwindigkeit, lässt aber dabei die Rampen (Beschleunigung, Ruck) unangetastet. Die verwendeten Overridetypen unterscheiden sich lediglich in der Bezugsgeschwindigkeit.

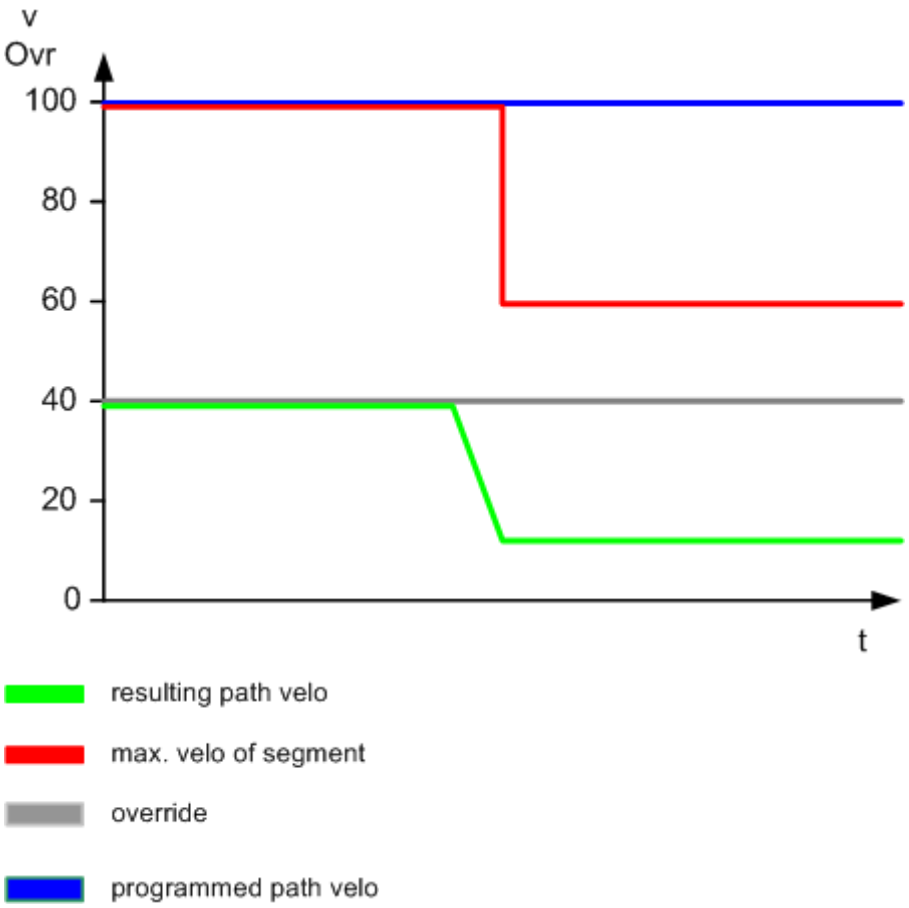
Die Parametrierung erfolgt im Interpolationskanal unter den Gruppenparametern [▶ 24].

Auswahl 'Reduziert' - bezogen auf die reduzierte Geschwindigkeit (default)

Aufgrund der gegebenen Dynamik (Bremsweg, Beschleunigung etc.) kann nicht in jedem Segment die programmierte Geschwindigkeit (blaue Linie) erreicht werden. Deshalb wird für jedes Geometriesegment eine evtl. reduzierte Geschwindigkeit (rote Linie) errechnet. Im Standardfall bezieht der Override sich auf diese Segmentgeschwindigkeit.

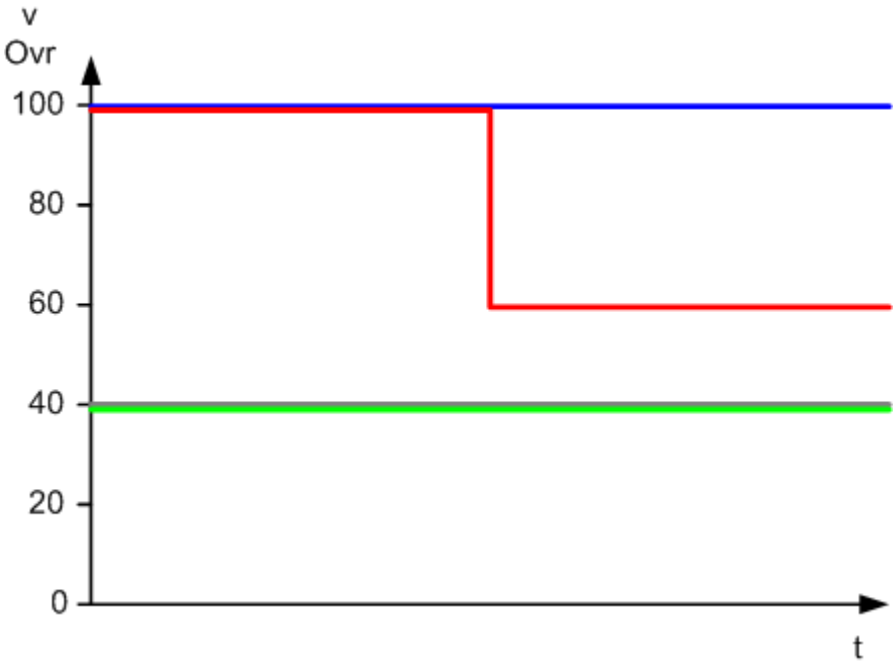
Dieser Overridetyp hat den Vorteil, dass bei kleineren Overridewerten die Maschine näherungsweise linear langsamer fährt und ist deshalb die richtige Einstellung für die meisten Anwendungen.

v_res = v_max * Override



Auswahl 'Original' - bezogen auf die programmierte Bahngeschwindigkeit

Der Overridewert wird auf die vom Anwender programmierte Geschwindigkeit bezogen. Die maximale Segmentgeschwindigkeit hat nur eine begrenzende Wirkung.



Auswahl 'Reduziert [0 ... >100%]' - bezogen auf intern reduzierte Geschwindigkeit mit der Option, einen Wert größer 100% vorzugeben

Der Overridetyp verhält sich wie 'Reduziert' [▶ 333]. Mit diesem Overridetyp ist es möglich, die Bahn schneller abzufahren, als diese im G-Code programmiert wurde. Es gibt hier keine Limitierung auf z.B. 120%. Die maximal mögliche Bahngeschwindigkeit wird durch die maximalen Geschwindigkeiten der Achskomponenten (G0-Geschwindigkeit) und deren Dynamik begrenzt.

Wenn Sie eine Begrenzung auf z.B. 120% wünschen, können Sie dies im PLC-Projekt entsprechend limitieren.

9.5 Zyklisches Kanal-Interface

Das Kanal-Interface ist für den zyklischen Datenaustausch zwischen der PLC und der NCI zuständig.

Von der NCI zur SPS (160 Bytes)

```
TYPE NCTOPLC_NCICHANNEL_REF :
STRUCT
BlockNo             : UDINT;
FastMFuncMask       : ARRAY [1..5] OF DWORD;
HskMFuncNo          : UINT;
HskMFuncReq         : WORD;
HFuncValue          : UDINT;
SpindleRpm          : UINT;
Tool                : UINT;
ChnState            : NCTOPLC_NCICHANNEL_REF_CHN_STATE;
IntParams           : ARRAY [0..3] OF UDINT;
DoubleParams        : ARRAY [0..3] OF LREAL;
PathVelo            : LREAL;
LoadedProg          : UDINT;
ItpMode             : WORD;
ItpState            : UINT;
ErrorCode           : UDINT;
ChnId               : UINT;
GrpId               : UINT;
ItfVersion          : UINT;
_reserved1          : UINT;
ChnOperationState   : UDINT;
McsAxisIDs          : ARRAY [0..7] OF USINT;
AcsAxisIDs          : ARRAY [0..7] OF USINT;
_reserved2          : ARRAY [1..24] OF USINT;
END_STRUCT
END_TYPE
```

Variablenname	Datentyp	Beschreibung
BlockNo	UDINT	Blocknummer
FastMFuncMask	ARRAY OF DWORD	Bitmaske zur Auswertung der schnellen M-Funktionen [▶ 167]
HskMFuncNo	UINT	Nummer der anliegenden synchronen M-Funktion (M-Funktion mit Handshake)
HskMFuncReq	WORD	Flag, mit dem angezeigt wird, dass eine synchrone M-Funktion anliegt 0: es liegt keine synchrone M-Funktion an 1: es liegt eine synchrone M-Funktion an
HFuncValue	DINT	Wert der Hilfsfunktion
SpindleRpm	WORD	Spindeldrehzahl
Tool	WORD	Werkzeugnummer
ChnState	NCTOPLC_NCICHANNEL_REF_CHN_STATE	DWORD mit Zustandsinformationen des Kanals (siehe Zustandsinformationen des Kanals (ChnState) [▶ 336]
IntParams	ARRAY [0..3] OF UDINT	Daten des frei konfigurierbaren Kanalinterfaces (s. ItpSetCyclicUDintOffsets [▶ 245])

Variablenname	Datentyp	Beschreibung
DoubleParams	ARRAY [0..3] OF LREAL	Daten des frei konfigurierbaren Kanalinterfaces (s. ItpSetCyclicLrealOffsets [▶ 243])
PathVelo	LREAL	Aktuelle Bahnsollgeschwindigkeit
LoadedProg	UDINT	Name des zurzeit abgearbeiteten NC-Programms. Falls der Name kein UDINT ist, so ist dieser Wert 0.
ItpMode	WORD	Bitmaske, die den Interpreter-Bearbeitungsmodus anzeigt.
ItpState	UINT	Status [▶ 15] des Interpreters
ErrorCode	UDINT	Error Code des Interpreterkanals
ChnId	UINT	Kanal-ID
GrpId	UINT	Gruppen-ID
ItfVersion	UINT	Version dieses zyklischen Kanal Interfaces
ChnOperationState	UDINT	Kanalzustand für einen Kanal der Kinematischen Transformation, hat keine Bedeutung für einen Interpolationskanal.
McsAxisIDs	ARRAY [0..7] OF USINT	Für einen Kanal der Kinematischen Transformation, IDs der MCS-Achsen, hat keine Bedeutung für einen Interpolationskanal.
AcsAxisIDs	ARRAY [0..7] OF USINT	Für einen Kanal der Kinematischen Transformation, IDs der ACS-Achsen, hat keine Bedeutung für einen Interpolationskanal.

Zustandsinformationen des Kanals (ChnState)

Die Zustandsinformationen des Kanals sind nur im XAE **mit Klarnamen lesbar und aus der SPS nur über die Bitnummer auslesbar.**

Name	Bitnummer (zero based)	Beschreibung
bIsInterpolationChannel	0	Zeigt an, dass der verknüpfte Kanal ein Interpolationskanal ist.
bIsKinematicChannel	1	Zeigt an, dass die Struktur mit einem Kanal für die Kinematische Transformation verknüpft ist.
bIsEStopRequested	8	Zeigt an, dass ein ItpEStop aufgerufen wurde - ohne eine Prüfung, ob sich die Achsen bereits im Stillstand befinden.
bIsFeedFromBackupList	10	Im Falle des Rückwärtsfahrens werden die aktuellen Einträge aus der Interpreter Backup Liste versendet.
bIsMovingBackward	11	Gibt an, dass die derzeitige Bewegung eine Rückwärtsbewegung ist.
bRetraceStartPosReached	12	Gibt an, dass beim Rückwärtsfahren der Programmanfang erreicht wurde.

Von der SPS zur NCI (128 Bytes)

```
TYPE PLCTONC_NCICHANNEL_REF :
STRUCT
SkipLine            : WORD; (* Mask to skip lines *)
ItpMode             : WORD;
MFuncGranted        : WORD; (* granted signal of the M-function *)
_reserved1          : UINT;
ChnAxesOvr          : UDINT; (* Channel override in percent * 100 *)
ChnSpindleOvr       : UDINT;
_reserved2          : ARRAY [1..112] OF USINT;
END_STRUCT
END_TYPE
```

Variablenname	Datentyp	Beschreibung
SkipLine	WORD	Bitmaske mit der aus der PLC die Satzunterdrückung [▶ 129] der NCI parametriert wird.
ItpMode	WORD	Bitmaske mit der der Interpreter-Bearbeitungsmodus geändert werden kann. Dies wird z.B. dann benötigt, wenn der Interpreter im Einzelsatz [▶ 135] arbeiten soll.
MFuncGranted	WORD	Flag, mit dem eine M-Funktion vom Typ 'Handshake' bestätigt wird. 0: nicht quittiert 1: Quittierung
ChnAxesOvr	UDINT	Kanal-Override für die Achsen von 0...1000000 (entspricht 0 - 100%).
ChnSpindleOvr	UDINT	Kanal-Override für die Spindel von 0...1000000 (entspricht 0 - 100%), aktuell nicht unterstützt.

Mehr Informationen:
www.beckhoff.com/tf5100

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