SIEMENS

SINUMERIK

SINUMERIK 840D sl / 828D Arbeitsvorbereitung

Programmierhandbuch

Gültig für

Steuerung
SINUMERIK 840D sl / 840DE sl
SINUMERIK 828D

Software	*Version*
NCU Systemsoftware	2.6 SP1

03/2010
6FC5398-2BP20-1AA0

Rechtliche Hinweise

Warnhinweiskonzept

Dieses Handbuch enthält Hinweise, die Sie zu Ihrer persönlichen Sicherheit sowie zur Vermeidung von Sachschäden beachten müssen. Die Hinweise zu Ihrer persönlichen Sicherheit sind durch ein Warndreieck hervorgehoben, Hinweise zu alleinigen Sachschäden stehen ohne Warndreieck. Je nach Gefährdungsstufe werden die Warnhinweise in abnehmender Reihenfolge wie folgt dargestellt.

⚠GEFAHR
bedeutet, dass Tod oder schwere Körperverletzung eintreten **wird**, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

⚠WARNUNG
bedeutet, dass Tod oder schwere Körperverletzung eintreten **kann**, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

⚠VORSICHT
mit Warndreieck bedeutet, dass eine leichte Körperverletzung eintreten kann, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

VORSICHT
ohne Warndreieck bedeutet, dass Sachschaden eintreten kann, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

ACHTUNG
bedeutet, dass ein unerwünschtes Ergebnis oder Zustand eintreten kann, wenn der entsprechende Hinweis nicht beachtet wird.

Beim Auftreten mehrerer Gefährdungsstufen wird immer der Warnhinweis zur jeweils höchsten Stufe verwendet. Wenn in einem Warnhinweis mit dem Warndreieck vor Personenschäden gewarnt wird, dann kann im selben Warnhinweis zusätzlich eine Warnung vor Sachschäden angefügt sein.

Qualifiziertes Personal

Das zu dieser Dokumentation zugehörige Produkt/System darf nur von für die jeweilige Aufgabenstellung **qualifiziertem Personal** gehandhabt werden unter Beachtung der für die jeweilige Aufgabenstellung zugehörigen Dokumentation, insbesondere der darin enthaltenen Sicherheits- und Warnhinweise. Qualifiziertes Personal ist auf Grund seiner Ausbildung und Erfahrung befähigt, im Umgang mit diesen Produkten/Systemen Risiken zu erkennen und mögliche Gefährdungen zu vermeiden.

Bestimmungsgemäßer Gebrauch von Siemens-Produkten

Beachten Sie Folgendes:

⚠WARNUNG
Siemens-Produkte dürfen nur für die im Katalog und in der zugehörigen technischen Dokumentation vorgesehenen Einsatzfälle verwendet werden. Falls Fremdprodukte und -komponenten zum Einsatz kommen, müssen diese von Siemens empfohlen bzw. zugelassen sein. Der einwandfreie und sichere Betrieb der Produkte setzt sachgemäßen Transport, sachgemäße Lagerung, Aufstellung, Montage, Installation, Inbetriebnahme, Bedienung und Instandhaltung voraus. Die zulässigen Umgebungsbedingungen müssen eingehalten werden. Hinweise in den zugehörigen Dokumentationen müssen beachtet werden.

Marken

Alle mit dem Schutzrechtsvermerk ® gekennzeichneten Bezeichnungen sind eingetragene Marken der Siemens AG. Die übrigen Bezeichnungen in dieser Schrift können Marken sein, deren Benutzung durch Dritte für deren Zwecke die Rechte der Inhaber verletzen kann.

Haftungsausschluss

Wir haben den Inhalt der Druckschrift auf Übereinstimmung mit der beschriebenen Hard- und Software geprüft. Dennoch können Abweichungen nicht ausgeschlossen werden, so dass wir für die vollständige Übereinstimmung keine Gewähr übernehmen. Die Angaben in dieser Druckschrift werden regelmäßig überprüft, notwendige Korrekturen sind in den nachfolgenden Auflagen enthalten.

Siemens AG
Industry Sector
Postfach 48 48
90026 NÜRNBERG
DEUTSCHLAND

Dokumentbestellnummer: 6FC5398-2BP20-1AA0
℗

Vorwort

SINUMERIK-Dokumentation

Die SINUMERIK-Dokumentation ist in 3 Kategorien gegliedert:

- Allgemeine Dokumentation
- Anwender-Dokumentation
- Hersteller-/Service-Dokumentation

Unter dem Link http://www.siemens.com/motioncontrol/docu gibt es Informationen zu folgenden Themen:

- Dokumentation bestellen
  Hier finden Sie die aktuelle Druckschriftenübersicht.
- Dokumentation downloaden
  Weiterführende Links für den Download von Dateien aus Service & Support.
- Dokumentation online recherchieren
  Informationen zur DOConCD und direkten Zugriff auf die Druckschriften im DOConWEB.
- Dokumentation auf Basis der Siemens Inhalte individuell zusammenstellen mit dem My Documentation Manager (MDM), siehe http://www.siemens.com/mdm

  Der My Documentation Manager bietet Ihnen eine Reihe von Features zur Erstellung Ihrer eigenen Maschinendokumentation.
- Training und FAQs

  Informationen zum Trainingsangebot und zu FAQs (frequently asked questions) finden Sie über die Seitennavigation.

Zielgruppe

Die vorliegende Druckschrift wendet sich an:

- Programmierer
- Projekteure

Nutzen

Das Programmierhandbuch befähigt die Zielgruppe, Programme und Software-Oberflächen zu entwerfen, zu schreiben, zu testen und Fehler zu beheben.

Standardumfang

In der vorliegenden Programmieranleitung ist die Funktionalität des Standardumfangs beschrieben. Ergänzungen oder Änderungen, die durch den Maschinenhersteller vorgenommen werden, werden vom Maschinenhersteller dokumentiert.

Es können in der Steuerung weitere, in dieser Dokumentation nicht erläuterte Funktionen ablauffähig sein. Es besteht jedoch kein Anspruch auf diese Funktionen bei der Neulieferung bzw. im Servicefall.

Ebenso enthält diese Dokumentation aus Gründen der Übersichtlichkeit nicht sämtliche Detailinformationen zu allen Typen des Produkts und kann auch nicht jeden denkbaren Fall der Aufstellung, des Betriebes und der Instandhaltung berücksichtigen.

Technical Support

Bei Fragen wenden Sie sich bitte an folgende Hotline:

	Europa / Afrika
Telefon	+49 180 5050 - 222
Fax	+49 180 5050 - 223
0,14 €/Min. aus dem deutschen Festnetz, abweichende Mobilfunkpreise möglich	
Internet	http://www.siemens.de/automation/support-request

	Amerika
Telefon	+1 423 262 2522
Fax	+1 423 262 2200
E-Mail	mailto:techsupport.sea@siemens.com

	Asien / Pazifik
Telefon	+86 1064 757575
Fax	+86 1064 747474
E-Mail	mailto:support.asia.automation@siemens.com

Hinweis

Landesspezifische Telefonnummern für technische Beratung finden Sie im Internet: http://www.automation.siemens.com/partner

Fragen zur Dokumentation

Bei Fragen zur Dokumentation (Anregungen, Korrekturen) senden Sie bitte ein Fax oder eine E-Mail an folgende Adresse:

Fax:	+49 9131- 98 2176
E-Mail:	mailto:docu.motioncontrol@siemens.com

Eine Faxvorlage finden Sie im Anhang dieses Dokuments.

Internetadresse für SINUMERIK

http://www.siemens.com/sinumerik

Programmierhandbuch "Grundlagen" und "Arbeitsvorbereitung"

Die Beschreibungen zur NC-Programmierung sind auf zwei Handbücher verteilt:

1. **Grundlagen**

   Das Programmierhandbuch "Grundlagen" dient dem Maschinenfacharbeiter und setzt entsprechende Kenntnisse für Bohr-, Fräs- und Drehbearbeitungen voraus. An einfachen Programmierbeispielen werden die auch nach DIN 66025 bekannten Befehle und Anweisungen erläutert.

2. **Arbeitsvorbereitung**

   Das Programmierhandbuch "Arbeitsvorbereitung" dient dem Technologen mit Kenntnissen über die gesamten Programmiermöglichkeiten. Die SINUMERIK-Steuerung ermöglicht mit einer speziellen Programmiersprache die Programmierung eines komplexen Werkstückprogramms (z. B. Freiformflächen, Kanalkoordinierung, ...) und erleichtert dem Technologen eine aufwendige Programmierung.

Verfügbarkeit der beschriebenen NC-Sprachelemente

Alle im vorliegenden Handbuch beschriebenen NC-Sprachelemente stehen für SINUMERIK 840D sl zur Verfügung. Die Verfügbarkeit bezüglich SINUMERIK 828D ist der Spalte "828D" der "Liste der Anweisungen (Seite 719)" zu entnehmen.

Inhaltsverzeichnis

Flexible NC-Programmierung 1

1.1 Variablen

1.1.1 Allgemeine Informationen zu Variablen

Durch die Verwendung von Variablen, insbesondere in Verbindung mit Rechenfunktionen und Kontrollstrukturen, können sie Teileprogramme und Zyklen extrem flexibel gestalten. Dazu werden vom System drei unterschiedliche Arten von Variablen zur Verfügung gestellt:

- Systemvariablen

  Systemvariablen sind im System definierte und dem Anwender zur Verfügung gestellte Variablen mit einer fest vorgegebenen Bedeutung. Sie werden auch von der Systemsoftware gelesen und geschrieben. Beispiel: Maschinendaten

  Die Bedeutung einer Systemvariable ist vom System fest die Eigenschaften weitestgehend vorgegeben. Die Eigenschaften können aber in geringem Umfang vom Anwender durch Redefinition noch angepasst werden.

- Anwendervariablen

  Anwendervariablen sind Variablen, deren Bedeutung dem System nicht bekannt ist und vom System auch nicht ausgewertet werden. Die Bedeutung wird ausschließlich durch den Anwender festgelegt.

  Anwendervariablen sind unterteilt in:

  – Vordefinierte Anwendervariablen

    Vordefinierte Anwendervariablen sind im System bereits definierte Variablen, deren Anzahl über spezifische Maschinendaten vom Anwender nur noch parametriert werden muss. Die Eigenschaften dieser Variablen können vom Anwender weitestgehend angepasst werden.

  – Anwenderdefinierte Variablen

    Anwenderdefinierte Variablen sind Variablen die ausschließlich vom Anwender definierte und vom System erst zur Laufzeit angelegt werden. Ihre Anzahl, Datentyp, Sichtbarkeit und alle weiteren Eigenschaften werden ausschließlich durch den Anwender festgelegt.

1.1.2 Systemvariablen

Systemvariablen sind im System vordefiniert Variablen, die in Teileprogrammen und Zyklen Zugriff auf die aktuelle Parametrierung der Steuerung, sowie Maschinen-, Steuerungs- und Prozesszustände bieten.

Vorlaufvariablen

Als Vorlaufvariablen werden Systemvariablen bezeichnet, die im Kontext des Vorlaufs, d.h. zum Zeitpunkt der Interpretation des Teileprogrammsatzes in dem die Systemvariable programmiert ist, gelesen und geschrieben werden. Vorlaufvariablen lösen keinen Vorlaufstop aus.

Hauptlaufvariablen

Als Hauptlaufvariablen werden Systemvariablen bezeichnet, die im Kontext des Hauptlaufs, d.h. zum Zeitpunkt der Ausführung des Teileprogrammsatzes in dem die Systemvariable programmiert ist, gelesen oder geschrieben werden. Hauptlaufvariablen sind:

- Systemvariablen, die in Synchronaktionen programmiert werden können (Lesen/Schreiben)
- Systemvariablen, die im Teileprogramm programmiert werden können und Vorlaufstop auslösen (Lesen/Schreiben)
- Systemvariablen, die im Teileprogramm programmiert werden können und der Wert im Vorlauf ermittelt, aber erst im Hauptlauf geschrieben wird (Hauptlauf-synchron: nur Schreiben)

Prefix-Systematik

Zur besonderen Kennzeichnung von Systemvariablen ist dem Name im Normalfall ein Prefix vorangestellt, der sich aus dem $-Zeichen, gefolgt von einem oder zwei Buchstabe und einem Unterstrich, zusammensetzt:

$ + 1. Buchstabe	Bedeutung: Datenart
Systemvariablen, die im Vorlauf gelesen / geschrieben werden	
$M	Maschinendaten 1)
$S	Settingdaten, Schutzbereiche 1)
$T	Werkzeugverwaltungsdaten
$P	Programmierte Werte
$C	Zyklenvariablen der ISO-Hüllzyklen
$O	Optionsdaten
R	R-Parameter (Rechenparameter) 2)
Systemvariablen, die im Hauptlauf gelesen / geschrieben werden	
$$M	Maschinendaten 1)
$$S	Settingdaten 1)
$A	Aktuelle Hauptlaufdaten
$V	Servo-Daten

$ + 1. Buchstabe	Bedeutung: Datenart
$R	R-Parameter (Rechenparameter) 2)
1) Bei der Verwendung von Maschinen- und Settingdaten im Teileprogramm / Zyklus als Vorlaufvariable wird der Prefix mit einem $-Zeichen geschrieben. Bei der Verwendung in Synchronaktionen als Hauptlaufvariable wird der Prefix mit zwei $-Zeichen geschrieben. 2) Bei der Verwendung eines R-Parameters im Teileprogramm / Zyklus als Vorlaufvariable wird kein Prefix geschrieben, z.B. R10. Bei der Verwendung in einer Synchronaktion als Hauptlaufvariable wird als Prefix ein $-Zeichen geschrieben, z.B. $R10.	

2. Buchstabe	Bedeutung: Sichtbarkeit
N	NCK-globale Variable (**N**CK)
C	kanalspezifische Variable (**C**hannel)
A	achsspezifische Variable (**A**xis)

Randbedingungen

Ausnahmen in der Prefix-Systematik

Folgende Systemvariablen weichen von der oben genannten Prefix-Systematik ab:

- $TC_...: Der 2. Buchstabe C verweist hier nicht auf kanalspezifische, sondern auf Werkzeughalter-spezifische Systemvariablen (TC = Tool Carrier)
- $P_ ...: Kanalspezifische Systemvariablen

Verwendung von Maschinen- und Settingdaten in Synchronaktionen

Bei der Verwendung von Maschinen- und Settingdaten in Synchronaktionen kann durch den Prefix bestimmt werden, ob das Maschinen- oder Settingdatum vorlauf- oder hauptlaufsynchron gelesen/geschrieben wird.

Bleibt das Datum während der Bearbeitung unverändert, kann vorlaufsynchron gelesen werden. Der Prefix des Maschinen- oder Settingdatums wird dazu mit einem $-Zeichen geschrieben:

Programmcode

```
ID=1 WHENEVER G710 $AA_IM[z] < $SA_OSCILL_REVERSE_POS2[Z]-6 DO $AA_OVR[X]=0
```

Wird das Datum während der Bearbeitung verändert, muss hauptlaufsynchron gelesen / geschrieben werden. Der Prefix des Maschinen- oder Settingdatums wird dazu mit zwei $-Zeichen geschrieben:

Programmcode

```
ID=1 WHENEVER $AA_IM[z] < $$SA_OSCILL_REVERSE_POS2[Z]-6 DO $AA_OVR[X]=0
```

Hinweis

Schreiben von Maschinendaten

Beim Schreiben eines Maschinen- oder Settingdatums ist darauf zu achten, dass die aktive Zugriffsstufe beim Ausführen des Teileprogramms / Zyklus den Schreibzugriff erlaubt und die Wirksamkeit des Datums "IMMEDIATE" ist.

Literatur

Eine Auflistung der Eigenschaften aller Systemvariablen findet sich in:

/PGA1/ Listenhandbuch Systemvariable

1.1.3 Vordefinierte Anwendervariablen: Rechenparameter (R)

Funktion

Die Rechenparameter oder R-Parameter sind eine vordefinierte Anwendervariable mit der Bezeichnung R, definiert als Feld vom Datentyp `REAL`. Aus historischen Gründen ist für R-Parameter neben der Schreibweise mit Feldindex z. B. `R[10]` auch die Schreibweise ohne Feldindex z. B. `R10` erlaubt.

Bei der Verwendung in Synchronaktionen muss der Buchstabe $ vorangestellt werden z. B. `$R10`.

Syntax

Bei Verwendung als Vorlaufvariable:

```
R<n>
R[<Ausdruck>]
```

Bei Verwendung als Hauptlaufvariable:

```
$R<n>
$R[<Ausdruck>]
```

Bedeutung

`R`: Bezeichner bei Verwendung als Vorlaufvariable z.B. im Teileprogramm

`$R`: Bezeichner bei Verwendung als Hauptlaufvariable z.B. in Synchronaktionen

Typ: REAL

Wertebereich: Bei nicht-exponentieller Schreibweise:

± (0.000 0001 ... 9999 9999)

Hinweis:
Es sind maximal 8 Dezimalstellen erlaubt

Bei exponentieller Schreibweise:

± ($1*10^{-300}$ ... $1*10^{+300}$)

Hinweis:

- Schreibweise: <Mantisse>EX<exponent> z.B. 8.2EX-3
- Es sind maximal 10 Zeichen einschließlich Vorzeichen und Dezimalpunkt erlaubt.

`<n>`: Nummer des R-Parameter

Typ: INT

Wertebereich: 0 - MAX_INDEX

Hinweis
MAX_INDEX ergibt sich aus der parametrierten Anzahl an R-Parametern:
MAX_INDEX = (MD28050 $MN_MM_NUM_R_PARAM) - 1

`<Ausdruck>`: Feldindex

Als Feldindex kann ein beliebiger Ausdruck angegeben werden, solange das Ergebnis des Ausdrucks in den Datentyp `INT` gewandelt werden kann (`INT`, `REAL`, `BOOL`, `CHAR`)

Beispiel

Zuweisungen an R-Parameter und Verwendung von R-Parametern in mathematischen Funktionen:

```
Programmcode                Kommentar
R0=3.5678                   ;   Zuweisung im Vorlauf
R[1]=-37.3                  ;   Zuweisung im Vorlauf
R3=-7                       ;   Zuweisung im Vorlauf
$R4=-0.1EX-5                ;   Zuweisung im Hauptlauf: R4 = -0.1 * 10^-5
$R[6]=1.874EX8              ;   Zuweisung im Hauptlauf: R6 = 1.874 * 10^8
R7=SIN(25.3)                ;   Zuweisung im Vorlauf
```

Programmcode	Kommentar
R[R2]=R10	; Indirekte Adressierung über R-Parameter
R[(R1+R2)*R3]=5	; Indirekte Adressierung über math. Ausdruck
X=(R1+R2)	; Verfahre Achse X auf die Position die sich aus der Summe von R1 und R2 ergibt
Z=SQRT(R1*R1+R2*R2)	; Verfahre Achse Z auf Position Quadratwurzel(R1^2 + R2^2)

1.1.4 Vordefinierte Anwendervariablen: Link-Variablen

Funktion

Über Link-Variablen können im Rahmen der Funktion "NCU-Link" zyklisch Daten zwischen NCUs, die in einem Netzwerk miteinander verbunden sind, ausgetauscht werden. Sie ermöglichen dabei einen Datenformat-spezifischen Zugriff auf den Link-Variablen-Speicher. Der Link-Variablen-Speicher wird sowohl bezüglich der Größe und als auch der Datenstruktur vom Anwender / Maschinenhersteller anlagenspezifisch festgelegt.

Link-Variablen sind systemglobale Anwendervariablen, die bei projektierter Link-Kommunikation von allen NCUs des Link-Verbundes in Teileprogrammen und Zyklen gelesen und geschrieben werden können. Im Gegensatz zu globalen Anwendervariablen (GUD) können Link-Variablen auch in Synchronaktionen verwendet werden.

Bei Anlagen ohne aktiven NCU-Link, können Link-Variablen Steuerungs-lokal neben den globalen Anwendervariablen (GUD), als zusätzliche globale Anwendervariablen verwendet werden.

Syntax

```
$A_DLB[<Index>]
$A_DLW[<Index>]
$A_DLD[<Index>]
$A_DLR[<Index>]
```

Bedeutung

$A_DLB:	Link-Variable für Datenformat BYTE (1 Byte)	
	Datentyp:	UINT
	Wertebereich:	0 ... 255
$A_DLW:	Link-Variable für Datenformat WORD (2 Bytes)	
	Datentyp:	INT
	Wertebereich:	-32768 ... 32767
$A_DLD:	Link-Variable für Datenformat DWORD (4 Bytes)	
	Datentyp:	INT
	Wertebereich:	-2147483648 ... 2147483647

`$A_DLR`: Link-Variable für Datenformat REAL (8 Bytes)

Datentyp: REAL

Wertebereich: $\pm(2{,}2*10^{-308} \ldots 1{,}8*10^{+308})$

`<Index>`: Adressindex in Byte, gerechnet vom Anfang des Link-Variablen-Speichers

Datentyp: INT

Wertebereich: 0 - MAX_INDEX

Hinweis

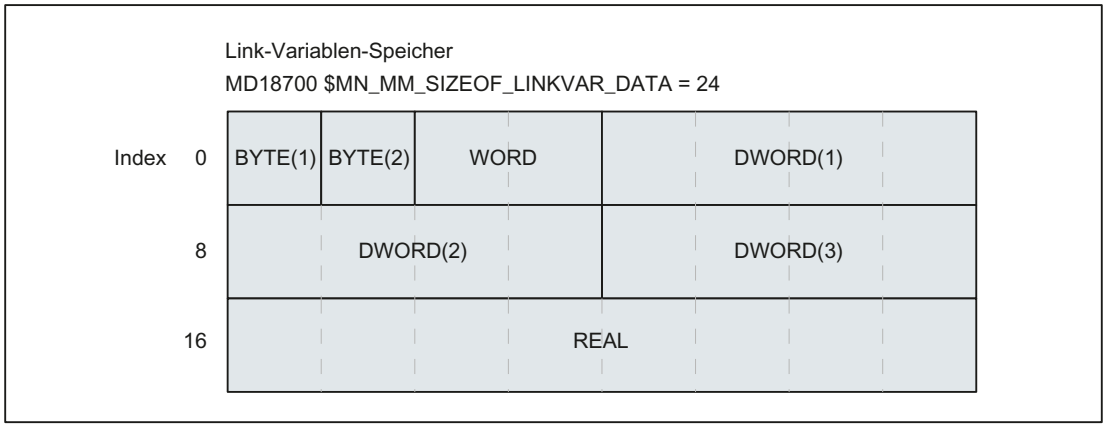
- MAX_INDEX ergibt sich aus der parametrierten Größe des Link-Variablen-Speichers: MAX_INDEX = (MD18700 $MN_MM_SIZEOF_LINKVAR_DATA) - 1
- Es dürfen nur Indizes programmiert werden, so dass die im Link-Variablen-Speicher adressierten Bytes auf einer Datenformatgrenze liegen ⇒
  Index = n * Bytes, mit n = 0, 1, 2, ...
  - `$A_DLB[i]`: i = 0, 1, 2, ...
  - `$A_DLW[i]`: i = 0, 2, 4, ...
  - `$A_DLD[i]`: i = 0, 4, 8, ...
  - `$A_DLR[i]`: i = 0, 8, 16, ...

Beispiel

In der Automatisierungsanlage sind 2 NCUs (NCU1 und NCU2) vorhanden. An NCU1 ist Maschinenachse AX2 angeschlossen, die als Link-Achse von NCU2 verfahren wird.

NCU1 schreibt zyklisch den Stromistwert ($VA_CURR) der Achse AX2 in den Link-Variablen-Speicher. NCU2 liest zyklisch den per Link-Kommunikation übertragenen Stromistwert und zeigt bei Überschreitung des Grenzwertes Alarm 61000 an.

Die Datenstruktur im Link-Variablen-Speicher ist im folgenden Bild dargestellt. Der Stromistwert wird über den REAL-Wert übertragen.



NCU1
NCU1 schreibt in einer statischen Synchronaktion zyklisch im IPO-Takt den Stromistwert der Achse AX2 über die Link-Variable $A_DLR[16] in den Link-Variablen-Speicher.

Programmcode

```
N111 IDS=1 WHENEVER TRUE DO $A_DLR[16]=$VA_CURR[AX2]
```

NCU2
NCU2 liest in einer statischen Synchronaktion zyklisch im IPO-Takt den Stromistwert der Achse AX2 über die Link-Variable $A_DLR[16] aus dem Link-Variablen-Speicher. Ist der Stromistwert größer als 23.0 A, wird der Alarm 61000 angezeigt.

Programmcode

```
N222 IDS=1 WHEN $A_DLR[16] > 23.0 DO SETAL(61000)
```

1.1.5 Definition von Anwendervariablen (DEF)

Funktion

Mit dem Befehl DEF können sie eigene Variablen definieren und mit Werten belegen. In Abgrenzung zu den Systemvariablen werden diese als anwenderdefinierte Variablen oder Anwendervariablen (User Data) bezeichnet.

Entsprechend dem Gültigkeitsbereich, d. h. dem Bereich in dem die Variable sichtbar ist, gibt es folgende Kategorien von Anwendervariablen:

- Lokale Anwendervariablen (LUD)

  Lokale Anwendervariablen (LUD) sind Variablen, die in einem Teileprogramm definiert sind, das zum Zeitpunkt der Abarbeitung nicht das Hauptprogramm ist. Sie werden beim Aufruf des Teileprogramms angelegt und mit dem Ende des Teileprogramms bzw. NC-Reset gelöscht. Auf LUD kann nur innerhalb des Teileprogramms zugegriffen werden, in dem sie definiert sind.

- Programmglobale Anwendervariablen (PUD)

  Programmglobale Anwendervariablen (PUD) sind Variablen, die in einem als Hauptprogramm verwendeten Teileprogramm definiert sind. Sie werden mit Teileprogrammstart angelegt und mit Teileprogrammende bzw. NC-Reset gelöscht. Auf PUD kann im Hauptprogramm und in allen Unterprogrammen zugegriffen werden.

- Globale Anwendervariablen (GUD)

  Globale Anwendervariablen (GUD) sind NC- bzw. Kanal-globale Variablen, die in einem Datenbaustein (SGUD, MGUD, UGUD, GUD4 ... GUD9) definiert sind und auch über Power On hinaus erhalten bleiben. Auf GUD kann in allen Teileprogrammen zugegriffen werden.

Anwendervariablen müssen vor ihrer Verwendung (Lesen / Schreiben) definiert worden sein. Folgende Regeln sind dabei zu beachten:

- GUD müssen in einer Definitionsdatei, z. B. _N_DEF_DIR/_M_SGUD_DEF, definiert werden.
- PUD und LUD müssen im Definitionsteil eines Teileprogramms definiert werden.
- Die Datendefinition muss in einem eigenen Satz erfolgen.
- Pro Datendefinition darf nur ein Datentyp verwendet werden.
- Pro Datendefinition können mehrere Variablen des gleichen Datentyps definiert werden.

Syntax

```
DEF <Bereich> <Typ> <VL_Stopp> <Init_Zeitpunkt> <Phys_Einheit>
<Grenzwerte> <Zugriffsrechte>
<Name>[<Wert_1>,<Wert_2>,<Wert_3>]=<Init_Wert>
```

Bedeutung

`DEF`:	Befehl zur Definition von Anwendervariablen GUD, PUD, LUD	
`<Bereich>`:	Gültigkeitsbereich, nur relevant für GUD:	
	`NCK`:	NC-globale Anwendervariable
	`CHAN`:	Kanal-globale Anwendervariable
`<Typ>`:	Datentyp:	
	`INT`:	Ganzzahliger Wert mit Vorzeichen
	`REAL`:	Real-Zahl (LONG REAL nach IEEE)
	`BOOL`:	Wahrheitswert TRUE (1) / FALSE (0)
	`CHAR`:	ASCII-Zeichen
	`STRING[<MaxLänge>]`:	Zeichenkette definierter Länge
	`AXIS`:	Achs-/Spindelbezeichner
	`FRAME`:	Geometrische Angaben für eine statische Koordinatentransformation
`<VL_Stopp>`:	Vorlaufstopp, nur relevant für GUD (optional)	
	`SYNR`:	Vorlaufstopp beim Lesen
	`SYNW`:	Vorlaufstopp beim Schreiben
	`SYNRW`:	Vorlaufstopp beim Lesen/Schreiben
`<Init_Zeitpunkt>`:	Zeitpunkt zu dem die Variable reinitialisiert wird (optional)	
	`INIPO`:	Power On
	`INIRE`:	Hauptprogrammende, NC-Reset oder Power On
	`INICF`:	NewConfig oder Hauptprogrammende, NC-Reset oder Power On
	`PRLOC`:	Hauptprogrammende, NC-Reset nach lokaler Änderung oder Power On
`<Phys_Einheit>`:	Physikalische Einheit (optional)	
	`PHU` <Einheit>:	

<Grenzwerte>:	unterer und oberer Grenzwert (optional)
	LLI <Grenzwert>: unterer Grenzwert (lower limit)
	ULI <Grenzwert>: oberer Grenzwert (upper limit)
<Zugriffsrechte>:	Zugriffsrechte für das Lesen / Schreiben von GUD über Teileprogramm oder BTSS (optional)
	APRP <Schutzstufe>: Lesen: Teileprogramm
	APWP <Schutzstufe>: Schreiben: Teileprogramm
	APRB <Schutzstufe>: Lesen: BTSS
	APWB <Schutzstufe>: Schreiben: BTSS
	Schutzstufe Wertebereich: 0 ... 7
<Name>:	Name der Variablen **Hinweis** • Maximal 31 Zeichen • Die beiden ersten Zeichen müssen ein Buchstabe und/oder ein Unterstrich sein. • Das "$"-Zeichen ist für Systemvariablen reserviert und darf nicht verwendet werden.
[<Wert_1>, <Wert_2>, <Wert_3>]:	Angabe der Feldgrößen für 1- bis max. 3-dimensionale Feldvariablen (optional)
<Init_Wert>:	Initialisierungswert (optional)

Beispiele

Beispiel 1: Definitionen von Anwendervariablen im Datenbaustein für Maschinenhersteller

Programmcode

```
%_N_MGUD_DEF                                          ; GUD-Baustein: Maschinenhersteller
$PATH=/_N_DEF_DIR
DEF CHAN REAL PHU 24 LLI 0 ULI 10 STROM_1, STROM_2
;  Beschreibung
;  Definition zweier GUD: STROM_1, STROM_2
;  Gültigkeitsbereich: Kanalweit
;  Datentyp: REAL
;  VL-Stopp: nicht programmiert => Defaultwert = kein VL-Stopp
;  Phys. Einheit: 24 = [A]
;  Grenzwerte: Low = 0.0, High = 10.0
;  Zugriffsrechte: nicht programmiert => Defaultwert = 7 = Schlüsselschalterstellung 0
;  Initialisierungswert: nicht programmiert => Defaultwert = 0.0

DEF NCK REAL PHU 13 LLI 10 APWP 3 APRP 3 APWB 0 APRB 2 ZEIT_1=12, ZEIT_2=45
;  Beschreibung
;  Definition zweier GUD: ZEIT_1, ZEIT_2
;  Gültigkeitsbereich: NCK-weit
;  Datentyp: REAL
```

Programmcode

```
;  VL-Stopp: nicht programmiert => Defaultwert = kein VL-Stopp
;  Phys. Einheit: 13 = [s]
;  Grenzwerte: Low = 10.0, High = nicht programmiert => obere Definitionsbereichsgrenze
;  Zugriffsrechte:
;     Teileprogramm: Schreiben/Lesen = 3 = Endanwender
;     BTSS: Schreiben = 0 = Siemens, Lesen = 3 = Endanwender
;  Initialisierungswert: ZEIT_1 = 12.0, ZEIT_2 = 45.0

DEF NCK APWP 3 APRP 3 APWB 0 APRB 3 STRING[5] GUD5_NAME = "COUNTER"
;  Beschreibung
;  Definition eines GUD: GUD5_NAME
;  Gültigkeitsbereich: NCK-weit
;  Datentyp: STRING, max. 5 Zeichen
;  VL-Stopp: nicht programmiert => Defaultwert = kein VL-Stopp
;  Phys. Einheit: nicht programmiert => Defaultwert = 0 = keine phys. Einheit
;  Grenzwerte: nicht programmiert => Definitionsbereichsgrenzen: Low = 0, High = 255
;  Zugriffsrechte:
;     Teileprogramm: Schreiben/Lesen = 3 = Endanwender
;     BTSS: Schreiben = 0 = Siemens, Lesen = 3 = Endanwender
;  Initialisierungswert: "COUNTER"
M30
```

Beispiel 2: Programm-globale und -lokale Anwendervariablen (PUD / LUD)

Programmcode	Kommentar
PROC MAIN	; Hauptprogramm
DEF INT VAR1	; PUD-Definition
...	
SUB2	; Unterprogrammaufruf
...	
M30	

Programmcode	Kommentar
PROC SUB2	; Unterprogramm SUB2
DEF INT VAR2	; LUD-DEFINITION
...	
IF (VAR1==1)	; PUD lesen
VAR1=VAR1+1	; PUD lesen und schreiben
VAR2=1	; LUD schreiben
ENDIF	
SUB3	; Unterprogrammaufruf
...	
M17	

Programmcode	Kommentar
PROC SUB3	; Unterprogramm SUB3
...	
IF (VAR1==1)	; PUD lesen
VAR1=VAR1+1	; PUD lesen und schreiben
VAR2=1	; Fehler: LUD aus SUB2 nicht bekannt
ENDIF	
...	
M17	

Beispiel 3: Definition und Verwendung von Anwendervariablen vom Datentyp `AXIS`

Programmcode	Kommentar
DEF AXIS ABSZISSE	; 1. Geometrieachse
DEF AXIS SPINDLE	; Spindel
...	
IF ISAXIS(1) == FALSE GOTOF WEITER	
ABSZISSE = $P_AXN1	
WEITER:	
...	
SPINDLE=(S1)	1. Spindel
OVRA[SPINDLE]=80	; Spindeloverride = 80%
SPINDLE=(S3)	3. Spindel

Randbedingungen

Globale Anwendervariablen (GUD)

Im Rahmen der Definition von globalen Anwendervariablen (GUD) sind folgende Maschinendaten zu berücksichtigen:

Nr.	Bezeichner: $MN_	Bedeutung
11140	GUD_AREA_ SAVE_TAB	zusätzliche Sicherung für GUD-Bausteine
18118 [1)]	MM_NUM_GUD_MODULES	Anzahl GUD-Dateien im aktiven Filesystem
18120 [1)]	MM_NUM_GUD_NAMES_NCK	Anzahl der globalen GUD-Namen
18130 [1)]	MM_NUM_GUD_NAMES_CHAN	Anzahl der kanalspez. GUD-Namen
18140 [1)]	MM_NUM_GUD_NAMES_AXIS	Anzahl der achsspez. GUD-Namen
18150 [1)]	MM_GUD_VALUES_MEM	Speicherplatz für globale GUD-Werte
18660 [1)]	MM_NUM_SYNACT_GUD_REAL	Anzahl projektierbare GUD Datentyp REAL
18661 [1)]	MM_NUM_SYNACT_GUD_INT	Anzahl projektierbare GUD Datentyp INT
18662 [1)]	MM_NUM_SYNACT_GUD_BOOL	Anzahl projektierbare GUD Datentyp BOOL
18663 [1)]	MM_NUM_SYNACT_GUD_AXIS	Anzahl projektierbare GUD Datentyp AXIS
18664 [1)]	MM_NUM_SYNACT_GUD_CHAR	Anzahl projektierbare GUD Datentyp CHAR
18665 [1)]	MM_NUM_SYNACT_GUD_STRING	Anzahl projektierbare GUD Datentyp STRING

[1)] Für SINUMERIK 828D nicht verfügbar.

Programmglobale Anwendervariablen (PUD)

ACHTUNG

Sichtbarkeit von programmlokalen Anwendervariablen (PUD)

Im Hauptprogramm definierte programmlokale Anwendervariablen (PUD) sind nur dann auch in den Unterprogrammen sichtbar, wenn folgendes Maschinendatum gesetzt ist:

MD11120 $MN_LUD_EXTENDED_SCOPE = 1

Mit MD11120 = 0 sind die im Hauptprogramm definierten programmlokalen Anwendervariablen nur im Hauptprogramm sichtbar.

Kanalübergreifende Verwendung einer NCK-globale Anwendervariablen vom Datentyp AXIS

Eine NCK-globale Anwendervariable vom Datentyp `AXIS`, die bei der Definition im Dateinbaustein mit einem Achsbezeichner initialisiert wurde, kann nur dann in unterschiedlichen Kanälen der NC verwendet werden, wenn die Achse in diesen Kanälen die gleiche Kanalachsnummer hat.

Ist dies nicht der Fall, muss die Variable am Teileprogrammanfang geladen oder, wie im folgenden Beispiel, die Funktion AXNAME(...) verwendet werden.

```
Programmcode                                  Kommentar
DEF NCK STRING[5] ACHSE="X"                   ;  Definition im Datenbaustein

N100 AX[AXNAME(ACHSE)]=111 G00                ;  Verwendung im Teileprogramm
```

1.1.6 Redefinition von Systemvariablen, Anwendervariablen und NC-Sprachbefehlen (REDEF)

Funktion

Mit dem Befehl REDEF können sie die Attribute von Systemvariablen, Anwendervariablen und NC-Sprachbefehle geändert. Grundvoraussetzung für eine Redefinition ist, dass sie zeitlich nach der entsprechenden Definition ausgeführt wird.

Bei einer Redefinition können nicht mehrere Attribute gleichzeitg geändert werden. Für jedes zu ändernde Attribut muss eine eigene REDEF-Anweisung programmiert werden.

Werden mehrere konkurrierende Attributänderungen programmiert, wird immer die letzte Änderung aktiv.

Redefinierbare Attribute

Siehe "Übersicht definierbarer und redefinierbarer Attribute (Seite 43)"

Lokale Anwendervariablen (PUD / LUD)

Für lokale Anwendervariablen (PUD / LUD) dürfen keine Redefinitionen vorgenommen werden.

Syntax

```
REDEF <Name> <VL_Stop>
REDEF <Name> <Phys_Einheit>
REDEF <Name> <Grenzwerte>
REDEF <Name> <Zugriffsrechte>
REDEF <Name> <Init_Zeitpunkt>
REDEF <Name> <Init_Zeitpunkt> <Init_Wert>
```

Bedeutung

REDEF:	Befehl zur Redefinition eines bestimmten Attributs von Systemvariablen, Anwendervariablen und NC-Sprachbefehle	
<Name>:	Name einer bereites definierten Variablen oder eines NC-Sprachbefehls	
<VL-Stopp>:	Vorlaufstopp	
	SYNR:	Vorlaufstopp beim Lesen
	SYNW:	Vorlaufstopp beim Schreiben
	SYNRW:	Vorlaufstopp beim Lesen/Schreiben

<Phys_Einheit>: Physikalische Einheit

PHU <Einheit>:

siehe "Attribut: Physikalische Einheit (PHU) (Seite 35)"

Hinweis
Nicht redefinierbar für:
- Systemvariablen
- globale Anwenderdaten (GUD)
- Datentypen: BOOL, AXIS, STRING, FRAME

<Grenzwerte>: Unterer und/oder oberer Grenzwert

LLI <Grenzwert>: unterer Grenzwert (lower limit)

ULI <Grenzwert>: oberer Grenzwert (upper limit)

siehe "Attribut: Grenzwerte (LLI, ULI) (Seite 34)"

Hinweis
Nicht redefinierbar für:
- Systemvariablen
- globale Anwenderdaten (GUD)
- Datentypen: BOOL, AXIS, STRING, FRAME

<Zugriffsrechte>: Zugriffsrechte für das Lesen / Schreiben über Teileprogramm oder BTSS

APX <Schutzstufe>: Ausführen: NC-Sprachelement

APRP <Schutzstufe>: Lesen: Teileprogramm

APWP <Schutzstufe>: Schreiben: Teileprogramm

APRB <Schutzstufe>: Lesen: BTSS

APWB <Schutzstufe>: Schreiben: BTSS

Schutzstufe Wertebereich: 0 ... 7

siehe "Attribut: Zugriffsrechte (APR, APW, APRP, APWP, APRB, APWB) (Seite 38)"

<Init_Zeitpunkt>: Zeitpunkt zu dem die Variable reinitialisiert wird

INIPO: PowerOn

INIRE: Hauptprogrammende, NC-Reset oder PowerOn

INICF: NewConfig oder Hauptprogrammende, NC-Reset oder PowerOn

PRLOC: Hauptprogrammende, NC-Reset nach lokaler Änderung oder PowerOn

siehe "Attribut: Initialisierungswert (Seite 30)"

<Init_Wert>: Initialisierungswert

Bei Redefinition des Initialisierungswertes muss immer auch ein Initialisierungszeitpunkt (siehe <Init_Zeitpunkt>) angegeben werden.

siehe "Attribut: Initialisierungswert (Seite 30)"

Hinweis
Nicht redefinierbar für:
- Systemvariablen, ausgenommen Settingdaten

Beispiel

Redefinitionen der Systemvariable $TC_DPC1 im Datenbaustein für Maschinenhersteller

Programmcode

```
%_N_MGUD_DEF                                          ; GUD-Baustein: Maschinenhersteller
$PATH=/_N_DEF_DIR
REDEF $TC_DPC1 APWB 2 APWP 3
REDEF $TC_DPC1 PHU 21
REDEF $TC_DPC1 LLI 0 ULI 200
REDEF $TC_DPC1 INIPO (100, 101, 102, 103)
;  Beschreibung
;  Zugriffsrecht Schreiben: BTSS = Schutzstufe 2, Teileprogramm = Schutzstufe 3
;  Hinweis
;  Bei Verwendung von ACCESS-Dateien muss die Redefinition der Zugriffsrechte von
;  _N_MGUD_DEF nach _N_MACCESS_DEF verlagert werden
;  Physikalische Einheit = [ % ]
;  Grenzwerte: unterer = 0, oberer = 200
;  Die Feldvariable wird bei PowerOn mit den vier Werten initialisiert
M30
```

Randbedingungen

Granularität

Eine Redefinition bezieht sich immer auf die gesamte, durch ihren Namen eindeutig gekennzeichnete Variable. Es ist nicht möglich z.B. bei Feldvariablen für einzelne Feldelemente unterschiedliche Attributwerte zuzuweisen.

1.1.7 Attribut: Initialisierungswert

Definition (DEF) von Anwendervariablen

Bei der Definition kann für folgende Anwendervariablen ein Initialisierungswert vorgegeben werden:

- globale Anwendervariablen (GUD)
- programmglobale Anwendervariablen (PUD)
- lokale Anwendervariablen (LUD)

Redefinition (REDEF) von System- und Anwendervariablen

Bei der Redefinition kann für folgende Variablen ein Initialisierungswert vorgegeben werden:

- Systemdaten
  - Settingdaten
- Anwenderdaten
  - R–Parameter
  - Synchronaktionsvariable ($AC_MARKER, $AC_PARAM, $AC_TIMER)
  - Synchronaktions–GUD (SYG_xy[ ], mit x=R, I, B, A, C, S und y=S, M, U, 4, ..., 9)
  - EPS–Parameter
  - Werkzeugdaten–OEM
  - Magazindaten–OEM
  - globale Anwendervariablen (GUD)

Reinitialisierungszeitpunkt
Bei der Redefinition kann Zeitpunkt angegeben werden, zu dem die Variable reinitialisiert, d.h. wieder auf den Initialisierungswert gesetzt werden soll:

- INIPO (Power On)

  Die Variable wird bei PowerOn reinitialisiert.

- INIRE (Reset)

  Die Variable wird bei NC-Reset, BAG-Reset, Teileprogrammende (M02 / M30) oder PowerOn reinitialisiert.

- INICF (NewConfig)

  Die Variable wird bei NewConf-Anforderung über HMI, Teileprogramm-Befehl NEWCONFIG oder NC-Reset, BAG-Reset, Teileprogrammende (M02 / M30) oder PowerOn reinitialisiert.

- PRLOC (progammlokale Änderung)

  Die Variable wird nur dann bei NC-Reset, BAG-Reset oder Teileprogrammende (M02 / M30) reinitialisiert, wenn sie im Rahmen des aktuellen Teileprogramms verändert worden ist.

  Das Attribut PRLOC darf nur in Zusammenhang mit programmierbaren Settingdaten (siehe folgende Tabelle) verwendet werden.

Tabelle 1- 1 Programmierbare Settingdaten

Nummer	Bezeichner	G-Befehl 1)
42000	$SC_THREAD_START_ANGLE	SF
42010	$SC_THREAD_RAMP_DISP	DITS / DITE
42400	$SA_PUNCH_DWELLTIME	PDELAYON
42800	$SA_SPIND_ASSIGN_TAB	SETMS
43210	$SA_SPIND_MIN_VELO_G25	G25
43220	$SA_SPIND_MAX_VELO_G26	G26
43230	$SA_SPIND_MAX_VELO_LIMS	LIMS
43300	$SA_ASSIGN_FEED_PER_REV_SOURCE	FPRAON
43420	$SA_WORKAREA_LIMIT_PLUS	G26
43430	$SA_WORKAREA_LIMIT_MINUS	G25
43510	$SA_FIXED_STOP_TORQUE	FXST
43520	$SA_FIXED_STOP_WINDOW	FXSW
43700	$SA_OSCILL_REVERSE_POS1	OSP1
43710	$SA_OSCILL_REVERSE_POS2	OSP2
43720	$SA_OSCILL_DWELL_TIME1	OST1
43730	$SA_OSCILL_DWELL_TIME2	OST2
43740	$SA_OSCILL_VELO	FA
43750	$SA_OSCILL_NUM_SPARK_CYCLES	OSNSC
43760	$SA_OSCILL_END_POS	OSE
43770	$SA_OSCILL_CTRL_MASK	OSCTRL
43780	$SA_OSCILL_IS_ACTIVE	OS
43790	$SA_OSCILL_START_POS	OSB
1) mit diesem G-Befehl wird das Settingdatum angesprochen		

Randbedingungen

Initialisierungswert: globale Anwendervariablen (GUD)

- Für globale Anwendervariable (GUD) mit dem Gültigkeitsbereich NCK, kann als Initialisierungszeitpunkt nur INIPO (Power On) vorgegeben werden.
- Für globalen Anwendervariablen (GUD) mit dem Gültigkeitsbereich CHAN kann als Initialisierungszeitpunkt neben INIPO (Power On) auch INIRE (Reset) oder INICF (NewConfig) vorgegeben werden.
- Bei globalen Anwendervariablen (GUD) mit dem Gültigkeitsbereich CHAN und Initialisierungszeitpunkt INIRE (Reset) oder INICF (NewConfig), werden bei NC-Reset, BAG-Reset und NewConfig die Variablen nur in den Kanälen neu initialisiert, in denen die genannten Ereignisse ausgelöst wurden.

Initialisierungswert: Datentyp FRAME

Für Variablen vom Datentyp FRAME darf kein Initialisierungswert angegeben werden. Variablen vom Datentyp FRAME werden implizit immer mit dem Defaultframe initialisiert.

Initialisierungswert: Datentyp CHAR

Für Variablen vom Datentyp CHAR kann statt des ASCII-Codes (0...255) auch das entsprechende ASCII-Zeichen in Anführungszeichen programmiert werden, z.B. "A"

Initialisierungswert: Datentyp STRING

Bei Variablen vom Datentyp STRING muss die Zeichenkette in Anführungszeichen gesetzt werden z.B.: ...= "MASCHINE_1"

Initialisierungswert: Datentyp AXIS

Für Variablen vom Datentyp AXIS muss bei erweiterter Adressschreibweise der Achsbezeichner in Klammern gesetzt werden, z.B.: ...=(X3)

Initialisierungswert: Systemvariable

Für Systemvariable können durch Redefinition keine anwenderspezifischen Initialisierungswerte vorgegeben werden. Die Initialisierungswerte der Systemvariablen sind vom System fest vorgegeben. Durch Redefinition kann aber der Zeitpunkt (INIRE, INICF) zu dem die Systemvariable reinitialisiert wird geändert werden.

Impliziter Initialisierungswert: Datentyp AXIS

Für Variablen vom Datentyp AXIS wird folgender implizite Initialisierungwert verwendet:

- Systemdaten: "erste Geometrieachse"
- Synchronaktions-GUD (Bezeichnung: SYG_A*), PUD, LUD:
  Achsbezeichner aus Maschinendatum: MD20082
  $MC_AXCONF_CHANAX_DEFAULT_NAME

Impliziter Initialisierungswert: Werkzeug- und Magazindaten

Für Werkzeug– und Magazindaten können Initialisierungstwerte über folgendes Maschinendatum vorgegeben werden: MD17520 $MN_TOOL_DEFAULT_DATA_MASK

ACHTUNG
Synchronisation Die Synchronisation von Ereignissen die eine Reinitialisierung einer globalen Variable auslösen mit dem Lesen dieser Variable an anderer Stelle, liegt ausschließlich in der Verantwortung des Anwenders / Maschinenherstellers.

1.1.8 Attribut: Grenzwerte (LLI, ULI)

Ein oberer und unterer Grenzwert des Definitionsbereichs kann nur für folgende Datentypen vorgegeben werden:

- INT
- REAL
- CHAR

Definition (DEF) von Anwendervariablen: Grenzwerte und implizite Initialisierungswerte

Wird bei der Definition einer Anwendervariablen von einem der oben genannten Datentypen kein expliziter Initialisierungswert definiert, wird die Variable auf den impliziten Initialisierungswert des Datentyps gesetzt:

- INT: 0
- REAL: 0.0
- CHAR: 0

Liegt der implizite Initialisierungswert ausserhalb des durch die programmierten Grenzwerte festgelegten Definitionsbereichs, wird die Variable mit dem Grenzwert initialisiert, der dem implizite Initialisierungswert am nächsten liegt:

- impliziter Initialisierungswert < unterer Grenzwert (LLI) ⇒
  Initialisierungswert = unterer Grenzwert
- impliziter Initialisierungswert > oberer Grenzwert (ULI) ⇒
  Initialisierungswert = oberer Grenzwert

Beispiele:

```
Programmcode                Kommentar
DEF REAL GUD1               ;   unterer Grenzwert = Definitionsbereichsgrenze
                            ;   oberer Grenwert = Definitionsbereichsgrenze
                            ;   kein Initialisierungswert programmiert
                            ;   => impliziter Initialisierungswert = 0.0
DEF REAL LLI 5.0 GUD2       ;   unterer Grenzwert = 5.0
                            ;   oberer Grenwert = Definitionsbereichsgrenze
                            ;   => Initialisierungswert = 5.0
DEF REAL ULI -5 GUD3        ;   unterer Grenwert = Definitionsbereichsgrenze
                            ;   oberer Grenzwert = -5.0
                            ;   => Initialisierungswert = -5.0
```

Redefinition (REDEF) von Anwendervariablen: Grenzwerte und aktuelle Istwerte

Werden bei der Redefinition der Grenzwerte einer Anwendervariablen diese so geändert, dass der aktuelle Istwert ausserhalb des neuen Definitionsbereichs liegt, erfolgt ein Alarm und die Grenzwerte werden nicht übernommen.

Hinweis

Redefinition (REDEF) von Anwendervariablen

Bei der Redefinition der Grenzwerte einer Anwendervariablen ist auf das konsistente Ändern der folgenden Werte zu achten:

- Grenzwerte
- Istwert
- Initialsierungswert beim Redefinieren und beim automatischen Reinitialsieren aufgrund von INIPO, INIRE oder INICF

1.1.9 Attribut: Physikalische Einheit (PHU)

Eine physikalische Einheit kann nur für Variablen von folgende Datentypen vorgegeben werden:

- INT
- REAL

Programmierbare physikalische Einheiten (PHU)

Die Angabe der physikalische Einheit erfolgt als Festkommazahl: `PHU <Einheit>`

Folgende physikalische Einheiten können programmiert werden:

<Einheit>	Bedeutung	Physikalische Einheit
0	keine physikalische Einheit	-
1	Linear– oder Winkel–Position 1)2)	[mm], [inch], [Grad]
2	Linear–Position 2)	[mm], [inch]
3	Winkel–Position	[Grad]
4	Linear– oder Winkel–Geschwindigkeit 1)2)	[mm/min], [inch/min], [U/min]
5	Linear–Geschwindigkeit 2)	[mm/min]
6	Winkel–Geschwindigkeit	[U/min]
7	Linear– oder Winkel–Beschleunigung 1)2)	[m/s^2], [$inch/s^2$], [U/s^2]
8	Linear–Beschleunigung 2)	[m/s^2], [$inch/s^2$]
9	Winkel–Beschleunigung	[U/s^2]
10	Linear– oder Winkel–Ruck 1)2)	[m/s^3], [$inch/s^3$], [U/s^3]

<Einheit>	Bedeutung	Physikalische Einheit
11	Linear–Ruck 2)	[m/s³], [inch/s³]
12	Winkel–Ruck	[U/s³]
13	Zeit	[s]
14	Lageregler–Verstärkung	[16.667/s]
15	Umdrehungsvorschub 2)	[mm/U], [inch/U]
16	Temperaturkompensation 1)2)	[mm], [inch]
18	Kraft	[N]
19	Masse	[kg]
20	Trägheitmoment 3)	[kgm²]
21	Prozent	[%]
22	Frequenz	[Hz]
23	Spannung	[V]
24	Strom	[A]
25	Temperatur	[°C]
26	Winkel	[Grad]
27	KV	[1000/min]
28	Linear– oder Winkel–Position 3)	[mm], [inch], [Grad]
29	Schnittgeschwindigkeit 2)	[m/min], [feet/min]
30	Umfangsgeschwindigkeit 2)	[m/s], [feet/s]
31	Widerstand	[Ohm]
32	Induktivitaet	[mH]
33	Drehmoment 3)	[Nm]
34	Drehmomentkonstante 3)	[Nm/A]
35	Stromreglerverstärkung	[V/A]
36	Drehzahlreglerverstärkung 3)	[Nm/(rad*s)]
37	Drehzahl	[U/min]
42	Leistung	[kW]
43	Strom, klein	[µA]
46	Drehmoment, klein 3)	[µNm]
48	Promille	-
49	-	[Hz/s]
65	Durchfluss	[l/min]
66	Druck	[bar]
67	Volumen 3)	[cm³]
68	Streckenverstärkung 3)	[mm/(V*min)]
69	Streckenverstärkung Kraftregler	[N/V]
155	Gewindesteigung 3)	[mm/U], [inch/U]
156	Gewindesteigungsänderung 3)	[mm/U / U], [inch/U / U]

<Einheit>	Bedeutung	Physikalische Einheit
1) Die physikalische Einheit ist abhängig vom Achstyp: Linear- oder Rundachse		
2) Maßsystem-Umschaltung G70/G71(inch/metrisch) Nach einer Umschaltung des Grundsystems (MD10240 $MN_SCALING_SYSTEM_IS_METRIC) mit G70/G71 erfolgt bei Schreib/Lesezugriffen auf längenbehaftete System- und Anwendervariablen **keine** Umrechnung der Werte (Istwert, Defaultwert und Grenzwerte) G700/G710(inch/metrisch) Nach einer Umschaltung des Grundsystems (MD10240 $MN_SCALING_SYSTEM_IS_METRIC) mit G700/G710 erfolgt bei Schreib/Lesezugriffen auf längenbehaftete System- und Anwendervariablen **eine** Umrechnung der Werte (Istwert, Defaultwert und Grenzwerte)		
3) Die Variable wird **nicht** automatische in das aktuelle Maßsystem der NC (inch/metrisch) umgerechnet. Die Umrechnung liegt ausschließlich in der Verantwortung des Anwenders / Maschinenherstellers.		

Hinweis

Ebenenüberlauf durch Formatumrechnung

Das interne Ablageformat für alle Anwendervariablen (GUD / PUD / LUD) mit längenbehafteten physikalischen Einheiten ist metrisch. Eine exzessive Verwendung derartiger Variablen im Hauptlauf des NCK, z.B. in Synchronaktionen, kann bei einer Maßsystemumschaltung zu einem Rechenzeitüberlauf der Interpolatorebene, Alarm 4240, führen.

ACHTUNG

Kompatibilität von Einheiten

Bei der Verwendung von Variablen (Zuweisung, Vergleich, Berechnung etc.) erfolgt keine Prüfung auf Kompatibilität der beteiligten Einheiten. Eine gegebenenfalls erforderliche Umrechnung liegt ausschließlich in der Verantwortung des Anwenders / Maschinenherstellers.

1.1.10 Attribut: Zugriffsrechte (APR, APW, APRP, APWP, APRB, APWB)

Den Zugriffsrechten entsprechen folgende bei der Programmierung anzugebende Schutzstufen:

Zugriffsrecht	Schutzstufe
Kennwort System	0
Kennwort Maschinenhersteller	1
Kennwort Service	2
Kennwort Endanwender	3
Schlüsselschalter Stellung 3	4
Schlüsselschalter Stellung 2	5
Schlüsselschalter Stellung 1	6
Schlüsselschalter Stellung 0	7

Definition (DEF) von Anwendervariablen

Zugriffsrechte (APR... / APW...) können für folgende Variablen definiert werden:

- globale Anwenderdaten (GUD)

Redefinition (REDEF) von System- und Anwendervariablen

Zugriffsrechte (APR... / APW...) können für folgende Variablen redefiniert werden:

- Systemdaten
  - Maschinendaten
  - Settingdaten
  - FRAME
  - Prozessdaten
  - Spindelsteigungsfehlerkompensation (EEC)
  - Durchhangkompensation (CEC)
  - Quadrantenfehlerkompensation (QEC)
  - Magazindaten
  - Werkzeugdaten
  - Schutzbereiche
  - orientierbare Werkzeugträger
  - kinematische Ketten
  - 3D-Schutzbereiche
  - Arbeitsfeldbegrenzung
  - ISO-Werkzeugdaten

- Anwenderdaten
  - R–Parameter
  - Synchronaktionsvariable ($AC_MARKER, $AC_PARAM, $AC_TIMER)
  - Synchronaktions–GUD (SYG_xy[ ], mit x=R, I, B, A, C, S und y=S, M, U, 4, ..., 9)
  - EPS–Parameter
  - Werkzeugdaten–OEM
  - Magazindaten–OEM
  - globale Anwendervariablen (GUD)

Hinweis

Bei der Redefinition kann das Zugriffsrecht auf eine Variable zwischen der niedrigsten Schutzstufe 7 und der eigenen Schutzstufe, z.B. 1 (Maschinenhersteller), frei vergeben werden.

Redefinition (REDEF) von NC-Sprachbefehlen

Das Zugriffs- bzw. Ausführungsrecht (APX) kann für folgende NC-Sprachbefehle redefiniert werden:

- G-Funktionen / Wegbedingungen

  Literatur:

  /PG/ Programmieranleitung Grundlagen; Kapitel: G-Funktionen / Wegbedingungen
- Vordefinierte Funktionen

  Literatur:

  /PG/ Programmieranleitung Grundlagen; Kapitel: Vordefinierte Funktionen
- Vordefinierte Unterprogrammaufrufe

  Literatur:

  /PG/ Programmieranleitung Grundlagen; Kapitel: Vordefinierte Unterprogrammaufrufe
- Anweisung DO bei Synchronaktionen
- Programmbezeichner von Zyklen

  Der Zyklus muss in einem Zyklenverzeichniss abgelegt sein und eine PROC-Anweisung enthalten.

Zugriffsrechte bezüglich Teileprogrammen und Zyklen (APRP, APWP)

Die unterschiedlichen Zugriffsrechte haben für den Zugriff in einem Teileprogramm bzw. Zyklus folgende Auswirkungen:

- APRP 0 / APWP 0
  - beim Abarbeiten des Teileprogramms muss das System-Kennwort gesetzt sein
  - der Zyklus muss im Verzeichnis _N_CST_DIR (System) abgelegt sein
  - für das Verzeichnis _N_CST_DIR muss im MD11160 $MN_ACCESS_EXEC_CST das Ausführungsrecht auf System eingestellt sein
- APRP 1 / APWP 1 bzw. APRP 2 / APWP 2
  - beim Abarbeiten des Teileprogramms muss das Maschinenhersteller- bzw. Service-Kennwort gesetzt sein
  - der Zyklus muss im Verzeichnis _N_CMA_DIR (Maschinenhersteller) oder _N_CST_DIR abgelegt sein
  - für die Verzeichnisse _N_CMA_DIR bzw. _N_CST_DIR müssen in den Maschinendaten MD11161 $MN_ACCESS_EXEC_CMA bzw. MD11160 $MN_ACCESS_EXEC_CST die Ausführungsrechte mindestens auf Maschinenhersteller eingestellt sein
- APRP 3 / APWP 3
  - beim Abarbeiten des Teileprogramms muss das Endanwender-Kennwort gesetzt sein
  - der Zyklus muss im Verzeichnis _N_CUS_DIR (Anwender), _N_CMA_DIR oder _N_CST_DIR abgelegt sein
  - für die Verzeichnisse _N_CUS_DIR, _N_CMA_DIR bzw. _N_CST_DIR müssen in den Maschinendaten MD11162 $MN_ACCESS_EXEC_CUS, MD11161 $MN_ACCESS_EXEC_CMA bzw. MD11160 $MN_ACCESS_EXEC_CST die Ausführungsrechte mindestens auf Endanwender eingestellt sein
- APRP 4...7 / APWP 4...7
  - beim Abarbeiten des Teileprogramms muss Schlüsselschalterstellung 3 ... 0 eingestellt sein
  - der Zyklus muss im Verzeichnis _N_CUS_DIR, _N_CMA_DIR oder _N_CST_DIR abgelegt sein
  - für die Verzeichnisse _N_CUS_DIR, _N_CMA_DIR bzw. _N_CST_DIR müssen in den Maschinendaten MD11162 $MN_ACCESS_EXEC_CUS, MD11161 $MN_ACCESS_EXEC_CMA bzw. MD11160 $MN_ACCESS_EXEC_CST die Ausführungsrechte mindestens auf die entsprechende Schlüsselschalterstellung eingestellt sein

Zugriffsrechte bezüglich BTSS (APRB, APWB)

Die Zugriffsrechte (APRB, APWB) beschränken den Zugriff auf System- und Anwendervariablen über BTSS für alle Systemkomponenten (HMI, PLC, externe Rechner, EPS-Dienste, etc.) gleichermaßen.

Hinweis

HMI-lokale Zugriffsrechte

Bei Änderungen von Zugriffsrechten von Systemdaten muss darauf geachtet werden, daß diese konsistent zu den über HMI-Mechanismen festgelegten Zugriffsrechten erfolgt.

Zugriffsattribute APR / APW

Aus Kombatibilitätsgründen werden die Attribute APR und APW implizit auf die Attribute APRP / APRB und APWP / APWB abgebildet:

- APR x ⇒ APRP x APRB x
- APW y ⇒ APWP y APWB y

Zugriffsrechte über ACCESS-Dateien einstellen

Bei der Verwendung von ACCES-Dateien für die Vergabe von Zugriffsrechten, dürfen Redefinitionen von Zugriffsrechten für Systemdaten, Anwenderdaten und NC-Sprachbefehlen nur noch in diesen ACCESS-Dateien programmiert werden. Eine Ausnahme bilden globale Anwenderdaten (GUD). Für diese muss, falls dies erforderlich erscheint, die Redefinition der Zugriffsrechte weiterhin in den entsprechenden Definitionsdateien programmiert werden.

Für einen durchgehenden Zugriffsschutz müssen die Maschinendaten für die Ausführungsrechte und den Zugriffsschutz der entsprechenden Verzeichnisse konsistenten angepasst werden.

Es ergibt sich folgende prinzipielle Vorgehensweise:

- Erstellen der benötigten Definitionsdateien:
  - _N_DEF_DIR/_N_SACCESS_DEF
  - _N_DEF_DIR/_N_MACCESS_DEF
  - _N_DEF_DIR/_N_UACCESS_DEF
- Parametrieren des Schreibrechtes für die Definitiondateien auf den für die Redefinition erforderlichen Wert:
  - MD11170 $MN_ACCESS_WRITE_SACCESS
  - MD11171 $MN_ACCESS_WRITE_MACCESS
  - MD11172 $MN_ACCESS_WRITE_UACCESS

- Für Zugriffe auf geschützte Elemente aus Zyklen heraus müssen die Ausführungs- und Schreibrechte der Zyklenverzeichnisse _N_CST_DIR, _N_CMA_DIR und _N_CST_DIR angepasst werden:

  Ausführungsrechte

  – MD11160 $MN_ACCESS_EXEC_CST
  – MD11161 $MN_ACCESS_EXEC_CMA
  – MD11162 $MN_ACCESS_EXEC_CUS

  Schreibrechte

  – MD11165 $MN_ACCESS_WRITE_CST
  – MD11166 $MN_ACCESS_WRITE_CMA
  – MD11167 MN_ACCESS_WRITE_CUS

  Das Ausführungsrecht muss mindestens auf die gleiche Schutzstufe wie die höchste Schutzstufe des verwendeten Elementes gesetzt werden.

  Das Schreibrecht muss mindestens auf die gleiche Schutzstufe wie das Ausführungsrecht gesetzt werden.

- Die Schreibrechte der HMI-lokalen Zyklenverzeichnisse müssen auf die gleiche Schutzstufe wie die der NC-lokalen Zyklenverzeichnisse gesetzt werden.

  Literatur

  /BAD/ Bedienhandbuch HMI-Advanced,

  Kapitel: Bedienbereich Dienste > Daten verwalten > Eigenschaften ändern

Unterprogrammaufrufe in ACCESS-Dateien

Für die weitere Strukturierung des Zugriffsschutzes können in den ACCESS-Dateien auch Unterprogramme (Kennung SPF oder MPF) aufgerufen werden. Die Unterprogramme erben dabei die Ausführungsrechte der aufrufenden ACCESS-Datei.

Hinweis

In den ACCESS-Dateien können nur die Zugriffsrechte redefiniert werden. Alle anderen Attribute müssen weiterhin in den entsprechenden Definitionsdateien programmiert bzw. redefiniert werden.

1.1.11 Übersicht definierbarer und redefinierbarer Attribute

Die folgenden Tabellen zeigen bei welchen Datenarten welche Attribute definiert (DEF) und/oder redefiniert (REDEF) werden können.

Systemdaten

Datenart	Init.Wert	Grenzwerte	phys. Einheit	Zugriffsrechte
Maschinendaten	---	---	---	REDEF
Settingdaten	REDEF	---	---	REDEF
FRAME-Daten	---	---	---	REDEF
Prozessdaten	---	---	---	REDEF
Spindelsteigungsfehlerkomp. (EEC)	---	---	---	REDEF
Durchhangkompensation (CEC)	---	---	---	REDEF
Quadrantenfehlerkompensation (QEC)	---	---	---	REDEF
Magazindaten	---	---	---	REDEF
Werkzeugdaten	---	---	---	REDEF
Schutzbereiche	---	---	---	REDEF
orientierbare Werkzeugträger	---	---	---	REDEF
kinematische Ketten	---	---	---	REDEF
3D-Schutzbereiche	---	---	---	REDEF
Arbeitsfeldbegrenzung	---	---	---	REDEF
ISO-Werkzeugdaten	---	---	---	REDEF

Anwenderdaten

Datenart	Init.Wert	Grenzwerte	phys. Einheit	Zugriffsrechte
R–Parameter	REDEF	REDEF	REDEF	REDEF
Synchronaktionsvariable ($AC_...)	REDEF	REDEF	REDEF	REDEF
Synchronaktions–GUD (SYG_...)	REDEF	REDEF	REDEF	REDEF
EPS–Parameter	REDEF	REDEF	REDEF	REDEF
Werkzeugdaten–OEM	REDEF	REDEF	REDEF	REDEF
Magazindaten–OEM	REDEF	REDEF	REDEF	REDEF
globale Anwendervariablen (GUD)	DEF / REDEF	DEF	DEF	DEF / REDEF
lokale Anwendervariablen (PUD / LUD)	DEF	DEF	DEF	---

1.1.12 Definition und Initialisierung von Feldvariablen (DEF, SET, REP)

Funktion

Eine Anwendervariable kann als 1- bis maximal 3-dimensionales Feld (Array) definiert werden:

- 1-dimensional: `DEF <Datentyp> <Variablenname>[<n>]`
- 2-dimensional: `DEF <Datentyp> <Variablenname>[<n>,<m>]`
- 3-dimensional: `DEF <Datentyp> <Variablenname>[<n>,<m>,<o>]`

Hinweis

Anwendervariable vom Datentyp STRING können maximal als 2-dimensionales Feld definiert werden.

Datentypen

Anwendervariable können als Felder für folgende Datentypen definiert werden: BOOL, CHAR, INT, REAL, STRING, AXIS, FRAME

Wertzuweisung an Feldelemente

Wertzuweisungen an Feldelemente können zu folgenden Zeitpunkten vorgenommen werden:

- bei der Felddefinition (Initialisierungswerte)
- während des Programmablaufs

Wertzuweisung können dabei erfolgen über:

- explizite Angabe eines Feldelements
- explizite Angabe eines Feldelements als Startelement und Angabe einer Werteliste (`SET`)
- explizite Angabe eines Feldelements als Startelement und Angabe eines Wertes und der Häufigkeit seiner Wiederholung (`REP`)

Hinweis

Anwendervariablen vom Datentyp FRAME können keine Initialisierungswerte zugewiesen werden.

Syntax (`DEF`)

```
DEF <Datentyp> <Variablenname>[<n>,<m>,<o>]
DEF  STRING[<Stringlänge>] <Variablenname>[<n>,<m>]
```

Syntax (DEF...=SET...)

Verwendung einer Werteliste:

- bei der Definition:

```
DEF <Datentyp> <Variablenname>[<n>,<m>,<o>] = SET(<Wert1>,<Wert2>,...)
```

gleichbedeutend mit:

```
DEF <Datentyp> <Variablenname>[<n>,<m>,<o>] = (<Wert1>,<Wert2>,...)
```

Hinweis

Bei der Initialisierung über eine Werteliste ist die Angabe von SET optional.

- bei einer Wertzuweisung:

```
<Variablenname>[<n>,<m>,<o>] = SET(<WERT1>,<Wert2>,...)
```

Syntax (DEF...=REP...)

Verwendung eines Werte mit Wiederholung

- bei der Definition:

```
DEF <Datentyp> <Variablenname>[<n>,<m>,<o>] = REP(<Wert>)

DEF <Datentyp> <Variablenname>[<n>,<m>,<o>] = REP(<Wert>,<Anzahl_Feldelemente>)
```

- bei einer Wertzuweisung:

```
<Variablenname>[<n>,<m>,<o>] = REP(<Wert>)
<Variablenname>[<n>,<m>,<o>] = REP(<Wert>,<Anzahl_Feldelemente>)
```

Bedeutung

DEF:	Befehl zur Definition von Variablen
<Datentyp>:	Datentyp der Variablen Wertebereich: • bei Systemvariablen: BOOL, CHAR, INT, REAL, STRING, AXIS • bei GUD- oder LUD-Variablen: BOOL, CHAR, INT, REAL, STRING, AXIS, FRAME
<Stringlänge>:	Maximale Anzahl der Zeichen beim Datentyp STRING
<Variablenname>:	Variablenname
[<n>,<m>,<o>]:	Feldgrößen bzw. Feldindizes
<n>:	Feldgröße bzw. Feldindex für 1. Dimension Typ: INT (bei Systemvariablen auch AXIS) Wertebereich: Max. Feldgröße: 65535 Feldindex: 0 ≤ n ≤ 65534

`<m>`:
Feldgröße bzw. Feldindex für 2. Dimension
Typ: INT (bei Systemvariablen auch AXIS)
Wertebereich: Max. Feldgröße: 65535
Feldindex: 0 ≤ m ≤ 65534

`<o>`:
Feldgröße bzw. Feldindex für 3. Dimension
Typ: INT (bei Systemvariablen auch AXIS)
Wertebereich: Max. Feldgröße: 65535
Feldindex: 0 ≤ o ≤ 65534

`SET`:
Wertzuweisung über die angegebenen Werteliste

`(<Wert1>,<Wert2>,...)`:
Werteliste

`REP`:
Wertzuweisung über den angegebenen `<Wert>`

`<Wert>`:
Wert, mit dem die Feldelemente bei der Initialisierung mit `REP` beschrieben werden sollen.

`<Anzahl_Feldelemente>`:
Anzahl der Feldelemente, die mit dem angegebenen `<Wert>` beschrieben werden sollen. Für die restlichen Feldelemente gilt abhängig vom Zeitpunkt:

- Initialisierung bei der Felddefinition:

  → Die restlichen Feldelemente werden mit Null beschrieben
- Zuweisung während des Programmlaufs:

  → Die aktuellen Werte der Feldelemente bleiben unverändert.

Ist der Parameter nicht programmiert, werden alle Feldelemente mit `<Wert>` beschrieben.

Ist der Parameter gleich Null, gilt abhängig vom Zeitpunkt:

- Initialisierung bei der Felddefinition:

  → Alle Elemente werden mit Null vorbelegt
- Zuweisung während des Programmlaufs:

  → Die aktuellen Werte der Feldelemente bleiben unverändert.

Feldindex

Die implizite Reihenfolge der Feldelemente z.B. bei einer Wertzuweisung über SET oder REP erfolgt durch Iteration der Feldindizes von rechts nach links.

Beispiel: Initialisierung eines 3-dimensionalen Feldes mit 24 Feldelementen:

```
DEF INT FELD[2,3,4] = REP(1,24)
  FELD[0,0,0] = 1          1. Feldelement
  FELD[0,0,1] = 1          2. Feldelement
  FELD[0,0,2] = 1          3. Feldelement
  FELD[0,0,3] = 1          4. Feldelement
  ...
  FELD[0,1,0] = 1          5. Feldelement
  FELD[0,1,1] = 1          6. Feldelement
  ...
  FELD[0,2,3] = 1          12. Feldelement
  FELD[1,0,0] = 1          13. Feldelement
  FELD[1,0,1] = 1          14. Feldelement
  ...
  FELD[1,2,3] = 1          24. Feldelement
```

entsprechend:

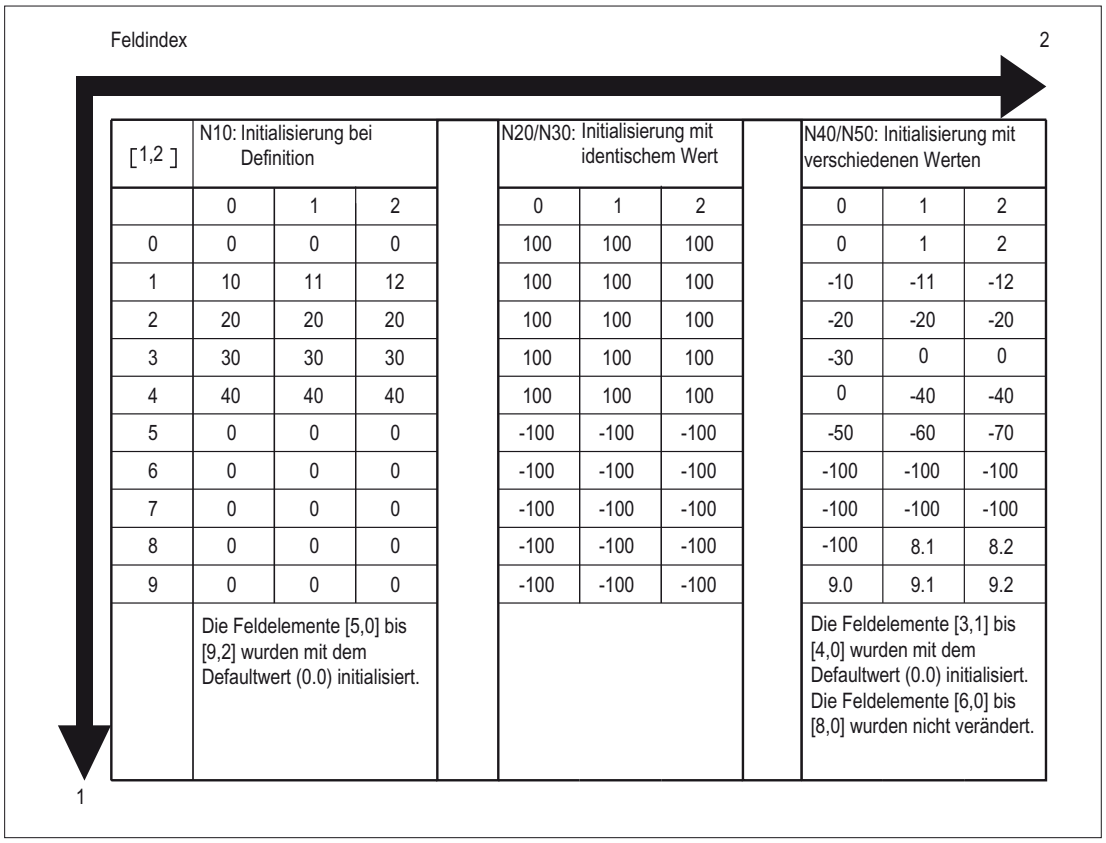
```
FOR n=0 TO 1
  FOR m=0 TO 2
     FOR o=0 TO 3
       FELD[n,m,o] = 1
     ENDFOR
  ENDFOR
ENDFOR
```

Beispiel: Initialisierung kompletter Variablenfelder

Aktuelle Belegung siehe Abbildung.

Programmcode

```
N10 DEF REAL FELD1[10,3]=SET(0,0,0,10,11,12,20,20,20,30,30,30,40,40,40,)
N20 FELD1[0,0]=REP(100)
N30 FELD1[5,0]=REP(-100)
N40 FELD1[0,0]=SET(0,1,2,-10,-11,-12,-20,-20,-20,-30, , , ,-40,-40,-50,-60,-70)
N50 FELD1[8,1]=SET(8.1,8.2,9.0,9.1,9.2)
```



[1,2]	N10: Initialisierung bei Definition			N20/N30: Initialisierung mit identischem Wert			N40/N50: Initialisierung mit verschiedenen Werten		
	0	1	2	0	1	2	0	1	2
0	0	0	0	100	100	100	0	1	2
1	10	11	12	100	100	100	-10	-11	-12
2	20	20	20	100	100	100	-20	-20	-20
3	30	30	30	100	100	100	-30	0	0
4	40	40	40	100	100	100	0	-40	-40
5	0	0	0	-100	-100	-100	-50	-60	-70
6	0	0	0	-100	-100	-100	-100	-100	-100
7	0	0	0	-100	-100	-100	-100	-100	-100
8	0	0	0	-100	-100	-100	-100	8.1	8.2
9	0	0	0	-100	-100	-100	9.0	9.1	9.2
	Die Feldelemente [5,0] bis [9,2] wurden mit dem Defaultwert (0.0) initialisiert.						Die Feldelemente [3,1] bis [4,0] wurden mit dem Defaultwert (0.0) initialisiert. Die Feldelemente [6,0] bis [8,0] wurden nicht verändert.		

1.1.13 Definition und Initialisierung von Feldvariablen (DEF, SET, REP): Weitere Informationen

Weitere Informationen (SET)

Initialisierung bei der Definition

- Es werden, beginnend beim 1. Feldelement, so viele Feldelemente mit den Werten aus der Werteliste initialisiert, wie Elemente in der Werteliste programmiert sind.
- Feldelemente ohne explizit angegebene Werte in der Werteliste (Lücken in der Werteliste) werden mit 0 belegt.
- Bei Variablen vom Datentyp AXIS sind Lücken in der Werteliste nicht zugelassen.
- Enthält die Werteliste mehr Werte als Feldelemente definiert sind, wird ein Alarm angezeigt.

Wertzuweisung im Programmablauf

Bei der Wertzuweisung im Programmablauf gelten die oben bei der Definition beschriebenen Regeln. Zusätzlich gibt es folgende Möglichkeiten:

- Als Elemente in der Werteliste sind auch Ausdrücke erlaubt.
- Die Wertzuweisung beginnt bei dem programmierten Feldindex. Hierdurch lassen sich gezielt Teilfelder mit Werten belegen.

Beispiel:

Programmcode	**Kommentar**
`DEF INT FELD[5,5]`	`; Felddefinition`
`FELD[0,0]=SET(1,2,3,4,5)`	`; Wertzuweisung an die ersten 5 Feldelemente [0,0] - [0,4]`
`FELD[0,0]=SET(1,2, , ,5)`	`; Wertzuweisung mit Lücke an die ersten 5 Feldelemente [0,0] - [0,4], Feldelemente [0,2] und [0,3] = 0`
`FELD[2,3]=SET(VARIABLE,4*5.6)`	`; Wertzuweisung mit Variable und Ausdruck ab Feldindex [2,3]: [2,3] = VARIABLE [2,4] = 4 * 5.6 = 22.4`

Weitere Informationen (REP)

Initialisierung bei der Definition

- Alle oder die optional angegebene Anzahl an Feldelementen werden mit dem angegebenen Wert (Konstante) initialisiert.
- Variablen vom Datentyp FRAME können nicht initialisiert werden.

Beispiel:

Programmcode	Kommentar
`DEF REAL varName[10]=REP(3.5,4)`	`; Felddefinition und Feldelemente [0] bis [3] mit Wert 3,5 initialisieren`

Wertzuweisung im Programmablauf

Bei der Wertzuweisung im Programmablauf gelten die oben bei der Definition beschriebenen Regeln. Zusätzlich gibt es folgende Möglichkeiten:

- Als Elemente in der Werteliste sind auch Ausdrücke erlaubt.
- Die Wertzuweisung beginnt bei dem programmierten Feldindex. Hierdurch lassen sich gezielt Teilfelder mit Werten belegen.

Beispiele:

Programmcode	Kommentar
`DEF REAL varName[10]`	`; Felddefinition`
`varName[5]=REP(4.5,3)`	`; Feldelemente [5] bis [7] = 4,5`
`R10=REP(2.4,3)`	`; R-Parameter R10 bis R12 = 2,4`
`DEF FRAME FRM[10]`	`; Felddefinition`
`FRM[5]=REP(CTRANS(X,5))`	`; Feldelemente [5] bis [9] = CTRANS(X,5)`

Weitere Informationen (allgemein)

Wertzuweisungen an axiale Maschinendaten

Axiale Maschinendaten haben prinzipiell einen Feldindex vom Datentyp AXIS. Bei Wertzuweisungen an ein axiales Maschinendatum mittels SET oder REP wird dieser Feldindex ignoriert bzw. nicht durchlaufen.

Beispiel: Wertzuweisung an Maschinendatum MD36200 $MA_AX_VELO_LIMIT

```
$MA_AX_VELO_LIMIT[1,AX1]=SET(1.1, 2.2, 3.3)
```

Entspricht:

```
$MA_AX_VELO_LIMIT[1,AX1]=1.1
$MA_AX_VELO_LIMIT[2,AX1]=2.2
$MA_AX_VELO_LIMIT[3,AX1]=3.3
```

ACHTUNG

Wertzuweisungen an axiale Maschinendaten

Bei Wertzuweisungen an axiale Maschinendaten mittels SET oder REP wird der Feldindex vom Datentyp AXIS ignoriert bzw. nicht durchlaufen.

Speicherbedarf

Datentyp	Speicherbedarf pro Element
BOOL	1 Byte
CHAR	1 Byte
INT	4 Bytes
REAL	8 Bytes
STRING	(Stringlänge + 1) Bytes
FRAME	~ 400 Bytes, abhängig von Achsanzahl
AXIS	4 Bytes

1.1.14 Datentypen

Folgende Datentypen stehen in der NC zur Verfügung:

Datentyp	Bedeutung	Wertebereich
INT	ganzzahliger Wert mit Vorzeichen	-2147483648 ... +2147483647
REAL	Real-Zahl (LONG REAL nach IEEE)	±(~2,2*10^{-308} … ~1,8*10^{+308})
BOOL	Wahrheitswert TRUE (1) und FALSE (0)	1, 0
CHAR	ASCII-Zeichen	ASCII-Code 0 ... 255
STRING	Zeichenkette definierter Länge	maximal 200 Zeichen (keine Sonderzeichen)
AXIS	Achs-/Spindelbezeichner	Kanalachsbezeichner
FRAME	Geometrische Angaben für eine statische Koordinatentransformation (Verschieben, Drehen, Skalieren, Spiegeln)	---

Implizite Datentypwandlungen

Folgende Datentypwandlungen sind möglich und werden bei Zuweisungen und Parameterübergaben implizit vorgenommen:

von ↓/ nach →	REAL	INT	BOOL
REAL	x	o	&
INT	x	x	&
BOOL	x	x	x

x: ohne Einschränkungen möglich

o: Datenverlust durch Überschreitung des Wertebereichs möglich ⇒ Alarm;
Rundung: Nachkommawert ≥ 0,5 ⇒ aufrunden, Nachkommawert < 0,5 ⇒ abrunden

&: Wert ≠ 0 ⇒ TRUE, Wert == 0 ⇒ FALSE

1.2 Indirekte Programmierung

1.2.1 Indirekte Programmierung von Adressen

Funktion

Bei der indirekten Programmierung von Adressen wird die erweiterte Adresse (Index) durch eine Variable geeigneten Typs ersetzt.

Hinweis

Die indirekte Programmierung von Adressen ist nicht möglich bei:

- N (Satznummer)
- L (Unterprogramm)
- Einstellbaren Adressen
  (z. B. X[1] anstelle von X1 ist nicht zulässig)

Syntax

```
<ADRESSE>[<Index>]
```

Bedeutung

`<ADRESSE>[...]`:	Feste Adresse mit Erweiterung (Index)
`<Index>`:	Variable z. B. für Spindelnummer, Achse, ...

Beispiele

Beispiel 1: Indirekte Programmierung einer Spindelnummer

Direkte Programmierung:

Programmcode	Kommentar
`S1=300`	`; Drehzahl 300 U/min für die Spindel mit Nummer 1.`

Indirekte Programmierung:

Programmcode	Kommentar
`DEF INT SPINU=1`	`; Definition der Variablen vom Typ INT und Wertzuweisung.`
`S[SPINU]=300`	`; Drehzahl 300 U/min für die Spindel, deren Nummer in der Variablen SPINU abgelegt ist (in diesem Beispiel die Spindel mit Nummer 1).`

Beispiel 2: Indirekte Programmierung einer Achse

Direkte Programmierung:

Programmcode	Kommentar
FA[U]=300	; Vorschub 300 für die Achse "U".

Indirekte Programmierung:

Programmcode	Kommentar
DEF AXIS AXVAR2=U	; Definition einer Variablen vom Typ AXIS und Wertzuweisung.
FA[AXVAR2]=300	; Vorschub 300 für die Achse, deren Adressname in der Variablen mit dem Namen AXVAR2 abgelegt ist.

Beispiel 3: Indirekte Programmierung einer Achse

Direkte Programmierung:

Programmierung	Kommentar
$AA_MM[X]	; Messtaster-Messwert (MKS) der Achse "X" lesen.

Indirekte Programmierung:

Programmcode	Kommentar
DEF AXIS AXVAR3=X	; Definition einer Variablen vom Typ AXIS und Wertzuweisung.
$AA_MM[AXVAR3]	; Messtaster-Messwert (MKS) lesen für die Achse, deren Name in der Variablen AXVAR3 abgelegt ist.

Beispiel 4: Indirekte Programmierung einer Achse

Direkte Programmierung:

Programmcode
X1=100 X2=200

Indirekte Programmierung:

Programmcode	Kommentar
DEF AXIS AXVAR1 AXVAR2	; Definition zweier Variablen vom Typ AXIS.
AXVAR1=(X1) AXVAR2=(X2)	; Zuweisung der Achsnamen.
AX[AXVAR1]=100 AX[AXVAR2]=200	; Verfahren der Achsen, deren Adressnamen in den Variablen mit den Namen AXVAR1 und AXVAR2 abgelegt sind.

Beispiel 5: Indirekte Programmierung einer Achse

Direkte Programmierung:

Programmcode
G2 X100 I20

Indirekte Programmierung:

Programmcode	Kommentar
DEF AXIS AXVAR1=X	; Definition einer Variablen vom Typ AXIS und Wertzuweisung.
G2 X100 IP[AXVAR1]=20	; Indirekte Programmierung der Mittelpunktsangabe für die Achse, deren Adressname in der Variablen mit dem Namen AXVAR1 abgelegt ist

Beispiel 6: Indirekte Programmierung von Feldelementen

Direkte Programmierung:

Programmcode	Kommentar
DEF INT FELD1[4,5]	; Definition von Feld 1.

Indirekte Programmierung:

Programmcode	Kommentar
DEFINE DIM1 AS 4	; Bei Felddimensionen müssen Feldgrößen als feste Werte angegeben werden.
DEFINE DIM2 AS 5	
DEF INT FELD[DIM1,DIM2]	
FELD[DIM1-1,DIM2-1]=5	

Beispiel 7: Indirekter Unterprogrammaufruf

Programmcode	Kommentar
CALL "L" << R10	; Aufruf des Programms, dessen Nummer in R10 steht (Stringverkettung).

1.2.2 Indirekte Programmierung von G-Codes

Funktion

Die indirekte Programmierung von G-Codes ermöglicht eine effektive Zyklenprogrammierung.

Syntax

```
G[<Gruppe>]=<Nummer>
```

Bedeutung

`G[...]`:	G-Befehl mit Erweiterung (Index)
`<Gruppe>`:	Index-Parameter: G-Funktionsgruppe Typ: INT
`<Nummer>`:	Variable für die G-Code-Nummer Typ: INT oder REAL

Hinweis

Es können i. d. R. nur nicht-syntaxbestimmende G-Codes indirekt programmiert werden.

Von den syntaxbestimmenden G-Codes sind nur die der G-Funktionsgruppe 1 möglich. Die syntaxbestimmenden G-Codes der G-Funktionsgruppen 2, 3 und 4 sind nicht möglich.

Hinweis

In der indirekten G-Code-Programmierung sind keine Arithmetik-Funktionen erlaubt. Eine notwendige Berechnung der G-Code-Nummer muss in einer eigenen Teileprogrammzeile vor der indirekten G-Code-Programmierung erfolgen.

Beispiele

Beispiel 1: Einstellbare Nullpunktverschiebung (G-Funktionsgruppe 8)

```
Programmcode                              Kommentar
N1010 DEF INT INT_VAR
N1020 INT_VAR=2
...
N1090 G[8]=INT_VAR G1 X0 Y0               ; G54
N1100 INT_VAR=INT_VAR+1                   ; G-Code-Berechnung
N1110 G[8]=INT_VAR G1 X0 Y0               ; G55
```

Beispiel 2: Ebenenanwahl (G-Funktionsgruppe 6)

Programmcode	Kommentar
N2010 R10=$P_GG[6]	; Aktive G-Funktion der G-Funktions-Gruppe 6 lesen
...	
N2090 G[6]=R10	

Literatur

Informationen zu den G-Funktionsgruppen siehe:
Programmierhandbuch Grundlagen; Kapitel "G-Funktionsgruppen"

1.2.3 Indirekte Programmierung von Positionsattributen (GP)

Funktion

Positionsattribute, wie z. B. die inkrementelle oder absolute Programmierung der Achsposition, können in Verbindung mit dem Schlüsselwort `GP` indirekt als Variablen programmiert werden.

Anwendung

Die indirekte Programmierung von Positionsattributen findet Verwendung in **Ersetzungszyklen**, da hier folgender Vorteil gegenüber der Programmierung von Positionsattributen als Schlüsselwort (z. B. `IC`, `AC`, ...) besteht:

Durch die indirekte Programmierung als Variablen wird **keine** `CASE`-Anweisung benötigt, die über alle möglichen Positionsattribute verzweigt.

Syntax

```
<POSITIONIERBEFEHL>[<Achse/Spindel>]=
GP(<Position>,<Positionsattribut)
<Achse/Spindel>=GP(<Position>,<Positionsattribut)
```

Bedeutung

`<POSITIONIERBEFEHL>[]`:	Folgende Positionierbefehle können zusammen mit dem Schlüsselwort GP programmiert werden:
	`POS`, `POSA`, `SPOS`, `SPOSA`
	Außerdem möglich:
	• alle im Kanal vorhandenen Achs-/Spindelbezeichner:
	`<Achse/Spindel>`
	• variabler Achs-/Spindelbezeichner `AX`

`<Achse/Spindel>`:	Achse/Spindel, die positioniert werden soll
`GP()`:	Schlüsselwort zur Positionierung
`<Position>`:	Parameter 1 Achs-/Spindelposition als Konstante oder Variable
`<Positionsattribut>`:	Parameter 2 Positionsattribut (z. B. Positionsanfahrmodus) als Variable (z. B. $P_SUB_SPOSMODE) oder als Schlüsselwort (IC, AC, ...)

Die von den Variablen gelieferten Werte haben folgende Bedeutung:

Wert	Bedeutung	Zulässig bei:
0	Keine Änderung des Positionsattributs	
1	AC	`POS`, `POSA`, `SPOS`, `SPOSA`, `AX`, Achsadresse
2	IC	`POS`, `POSA`, `SPOS`, `SPOSA`, `AX`, Achsadresse
3	DC	`POS`, `POSA`, `SPOS`, `SPOSA`, `AX`, Achsadresse
4	ACP	`POS`, `POSA`, `SPOS`, `SPOSA`, `AX`, Achsadresse
5	ACN	`POS`, `POSA`, `SPOS`, `SPOSA`, `AX`, Achsadresse
6	OC	-
7	PC	-
8	DAC	`POS`, `POSA`, `AX`, Achsadresse
9	DIC	`POS`, `POSA`, `AX`, Achsadresse
10	RAC	`POS`, `POSA`, `AX`, Achsadresse
11	RIC	`POS`, `POSA`, `AX`, Achsadresse
12	CAC	`POS`, `POSA`
13	CIC	`POS`, `POSA`
14	CDC	`POS`, `POSA`
15	CACP	`POS`, `POSA`
16	CACN	`POS`, `POSA`

Beispiel

Bei einer aktiven Synchronspindelkopplung zwischen der Leitspindel S1 und der Folgespindel S2 wird durch den SPOS-Befehl im Hauptprogramm der folgende Ersetzungszyklus zur Positionierung der Spindeln aufgerufen.

Die Positionierung erfolgt über die Anweisung in N2230:
```
SPOS[1]=GP($P_SUB_SPOSIT,$P_SUB_SPOSMODE)
SPOS[2]=GP($P_SUB_SPOSIT,$P_SUB_SPOSMODE)
```

Die anzufahrende Position wird aus der Systemvariablen $P_SUB_SPOSIT, der Positionsanfahrmodus wird aus der Systemvariablen $P_SUB_SPOSMODE gelesen.

```
Programmcode                                                    Kommentar
N1000 PROC LANG_SUB DISPLOF SBLOF
...
N2100 IF($P_SUB_AXFCT==2)
N2110                                                           ; Ersetzung des SPOS / SPOSA / M19-
                                                                  Befehls bei aktiver
                                                                  Synchronspindelkopplung
N2185 DELAYFSTON                                                ; Beginn Stopp-Delay-Bereich
N2190 COUPOF(S2,S1)                                             ; Synchronspindelkopplung deaktivieren
N2200                                                           ; Leit- und Folgespindel positionieren
N2210 IF($P_SUB_SPOS==TRUE) OR ($P_SUB_SPOSA==TRUE)
N2220                                                           ; Spindel mit SPOS positionieren:
N2230    SPOS[1]=GP($P_SUB_SPOSIT,$P_SUB_SPOSMODE)
         SPOS[2]=GP($P_SUB_SPOSIT,$P_SUB_SPOSMODE)
N2250 ELSE
N2260                                                           ; Spindel mit M19 positionieren:
N2270 M1=19 M2=19                                               ; Leit- und Folgespindel positionieren
N2280 ENDIF
N2285 DELAYFSTOF                                                ; Ende Stopp-Delay-Bereich
N2290 COUPON(S2,S1)                                             ; Synchronspindelkopplung aktivieren
N2410 ELSE
N2420                                                           ; Abfrage auf weitere Ersetzungen
...
N3300 ENDIF
...
N9999 RET
```

Randbedingungen

- In Synchronaktionen ist die indirekte Programmierung von Positionsattributen nicht möglich.

Literatur

Funktionshandbuch Grundfunktionen; BAG, Kanal, Programmbetrieb, Reset-Verhalten (K1), Kapitel: Ersetzung von NC-Funktionen durch Unterprogramme

1.2.4 Indirekte Programmierung von Teileprogrammzeilen (EXECSTRING)

Funktion

Mit dem Teileprogrammbefehl EXECSTRING ist es möglich, eine zuvor erzeugte String-Variable als Teileprogrammzeile auszuführen.

Syntax

EXECSTRING wird in einer eigenen Teileprogrammzeile programmiert:
EXECSTRING(<String-Variable>)

Bedeutung

EXECSTRING:	Befehl zur Ausführung einer String-Variablen als Teileprogrammzeile
<String-Variable>:	Variable vom Typ STRING, die die eigentlich auszuführende Teileprogrammzeile enthält

Hinweis

Mit EXECSTRING können alle Teileprogramm-Konstrukte abgesetzt werden, die im Programmteil eines Teileprogramms programmiert werden können. Ausgeschlossen sind damit PROC- und DEF-Anweisungen sowie generell die Verwendung in INI- und DEF-Dateien.

Beispiel

```
Programmcode                                  Kommentar
N100 DEF STRING[100] BLOCK                    ; Definition der String-Variablen zur
                                                Aufnahme der auszuführenden
                                                Teileprogrammzeile.
N110 DEF STRING[10] MFCT1="M7"
...
N200 EXECSTRING(MFCT1 << "M4711")             ; Teileprogrammzeile "M7 M4711"
                                                ausführen.
...
N300 R10=1
N310 BLOCK="M3"
N320 IF(R10)
N330 BLOCK = BLOCK << MFCT1
N340 ENDIF
N350 EXECSTRING(BLOCK)                        ; Teileprogrammzeile "M3 M7" ausführen.
```

1.3 Rechenfunktionen

Funktion

Die Rechenfunktionen sind vorrangig für R-Parameter und Variable (oder Konstante und Funktionen) vom Typ REAL anwendbar. Zulässig sind auch die Typen INT und CHAR.

Operator / Rechenfunktion	Bedeutung
+	Addition
-	Subtraktion
*	Multiplikation
/	Division **Achtung**: (Typ INT)/(Typ INT)=(Typ REAL); Beispiel: 3/4 = 0.75
DIV	Division, für Variablentyp INT und REAL **Achtung**: (Typ INT)DIV(Typ INT)=(Typ INT); Beispiel: 3 DIV 4 = 0
MOD	Modulo-Division (nur für Typ INT) liefert Rest einer INT-Division Beispiel: 3 MOD 4 = 3
:	Kettungsoperator (bei FRAME-Variablen)
Sin()	Sinus
COS()	Cosinus
TAN()	Tangens
ASIN()	Arcussinus
ACOS()	Arcuscosinus
ATAN2(,)	Arcustangens2
SQRT()	Quadratwurzel
ABS()	Betrag
POT()	2. Potenz (Quadrat)
TRUNC()	ganzzahliger Teil Genauigkeiten bei Vergleichsbefehlen einstellbar mit TRUNC (siehe "Genauigkeitskorrektur bei Vergleichsfehlern (TRUNC) (Seite 66)")
ROUND()	Runden auf ein Ganzzahliges
LN()	natürlicher Logarithmus
EXP()	Exponentialfunktion
MINVAL()	kleinerer Wert zweier Variablen (siehe "Minimum, Maximum und Bereich von Variablen (MINVAL, MAXVAL, BOUND) (Seite 68)")
MAXVAL()	größerer Wert zweier Variablen (siehe "Minimum, Maximum und Bereich von Variablen (MINVAL, MAXVAL, BOUND) (Seite 68)")

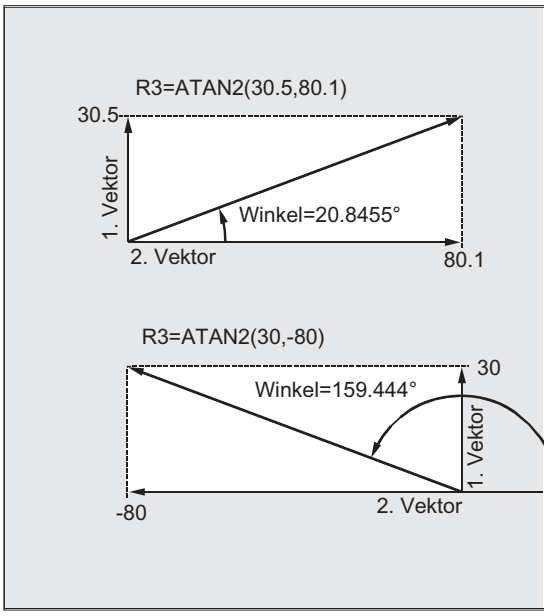
`BOUND()`	Variablenwert, der im definierten Wertebereich liegt (siehe "Minimum, Maximum und Bereich von Variablen (MINVAL, MAXVAL, BOUND) (Seite 68)")
`CTRANS()`	Verschiebung
`CROT()`	Drehung
`CSCALE()`	Maßstabsveränderung
`CMIRROR()`	Spiegeln

Programmierung

Bei den Rechenfunktionen gilt die übliche mathematische Schreibweise. Prioritäten in der Abarbeitung werden durch runde Klammern gesetzt. Für die trigonometrischen und deren inverse Funktionen gilt die Gradangabe (rechter Winkel = 90°).

Beispiele

Beispiel 1: ATAN2



Die Rechenfunktion ATAN2 berechnet aus zwei aufeinander senkrecht stehenden Vektoren den Winkel des Summenvektors.

Das Ergebnis liegt im Bereich von vier Quadranten (-180° < 0 < +180°).

Basis für den Winkelbezug ist immer der 2. Wert in positiver Richtung.

Beispiel 2: Initialisierung kompletter Variablenfelder

Programmcode	**Kommentar**
`R1=R1+1`	`; Neues R1 = altes R1 +1`
`R1=R2+R3 R4=R5-R6 R7=R8*R9`	
`R10=R11/R12 R13=SIN(25.3)`	
`R14=R1*R2+R3`	`; Punktrechnung geht vor Strichrechnung.`
`R14=(R1+R2)*R3`	`; Klammern werden zuerst berechnet.`
`R15=SQRT(POT(R1)+POT(R2))`	`; Innere Klammern werden zuerst aufgelöst:` `R15 = Quadratwurzel aus (R1+R2)`
`RESFRAME=FRAME1:FRAME2` `FRAME3=CTRANS(…):CROT(…)`	`; Mit dem Kettungsoperator werden Frames zu einem resultierenden Frame verknüpft oder den Frame-Komponenten Werte zugewiesen.`

1.4 Vergleichs- und logische Operationen

Funktion

Vergleichsoperationen können z. B. zur Formulierung einer Sprungbedingung benutzt werden. Vergleichbar sind dabei auch komplexe Ausdrücke.

Die Vergleichsoperationen sind für Variable vom Typ `CHAR, INT, REAL` und `BOOL` anwendbar. Beim Typ `CHAR` wird der Codewert verglichen.
Bei den Typen `STRING`, `AXIS` und `FRAME` sind möglich: **==** und **<>**, die für Operationen vom Typ `STRING` auch in Synchronaktionen angewendet werden können.

Das Ergebnis von vergleichenden Operationen ist immer vom Typ `BOOL`.

Logische Operatoren dienen zur Verknüpfung von Wahrheitswerten.

Die logischen Operationen sind nur auf Variable vom Typ `BOOL` anwendbar. Über interne Typenkonvertierung sind sie auch auf die Datentypen `CHAR, INT`, und `REAL` anwendbar.

Bei den logischen (boolschen) Operationen gilt für die Datentypen `BOOL`, `CHAR`, `INT` und `REAL`:

- 0 entspricht: FALSE
- ungleich 0 entspricht: TRUE

Bitweise logische Operatoren

Mit den Variablen vom Typ `CHAR` und `INT` können auch bitweise logische Operationen vorgenommen werden. Gegebenenfalls erfolgt eine Typkonvertierung automatisch.

Programmierung

Vergleichsoperator	Bedeutung
==	gleich
<>	ungleich
>	größer
<	kleiner
>=	größer oder gleich
<=	kleiner oder gleich

Logischer Operator	Bedeutung
AND	UND
OR	ODER
NOT	Negation
XOR	Exklusiv-ODER

Bitweise logischer Operator	Bedeutung
B_AND	bitweises UND
B_OR	bitweises ODER
B_NOT	bitweise Negation
B_XOR	bitweises Exklusiv-ODER

Hinweis

In arithmetischen Ausdrücken kann durch runde Klammern die Abarbeitungsreihenfolge aller Operatoren festgelegt und damit von den normalen Prioritätsregeln abgewichen werden.

Hinweis

Zwischen BOOLSCHEN Operanden und Operatoren müssen Zwischenräume geschrieben werden.

Hinweis

Der Operator B_NOT bezieht sich auf nur einen Operanden. Dieser steht nach dem Operator.

Beispiele

Beispiel 1: Vergleichsoperatoren

```
IF R10>=100 GOTOF ZIEL
```

oder

```
R11=R10>=100
IF R11 GOTOF ZIEL
```

Das Ergebnis des Vergleichs R10>=100 wird zunächst in R11 zwischengespeichert.

Beispiel 2: Logische Operatoren

```
IF (R10<50) AND ($AA_IM[X]>=17.5) GOTOF ZIEL
```

oder

```
IF NOT R10 GOTOB START
```

NOT bezieht sich nur auf einen Operanden.

Beispiel 3: Bitweise logische Operatoren

```
IF $MC_RESET_MODE_MASK B_AND 'B10000' GOTOF ACT_PLANE
```

1.5 Genauigkeitskorrektur bei Vergleichsfehlern (TRUNC)

Funktion

Der TRUNC-Befehl schneidet den mit einem Genauigkeitsfaktor multiplizierten Operanden ab.

Einstellbare Genauigkeit bei Vergleichsbefehlen

Teileprogrammdaten vom Typ REAL werden intern im IEEE-Format mit 64 Bit dargestellt. Aufgrund dieser Darstellungsform können Dezimalzahlen ungenau abgebildet werden, die bei einem Vergleich mit ideal gerechneten Werten zu unerwarteten Ergebnissen führen können.

Relative Gleichheit

Damit die durch die Darstellungsform hervorgerufenen Ungenauigkeiten den Programmfluß nicht verfälschen, wird bei den Vergleichsbefehlen nicht auf absolute Gleichheit, sondern auf eine relative Gleichheit geprüft.

Syntax

Genauigkeitskorrektur bei Vergleichsfehlern

```
TRUNC (R1*1000)
```

Bedeutung

`TRUNC`: Abschneiden der Nachkommastellen

Berücksichtigte relative Gleichheit von 10^{-12} bei

- Gleichheit: (==)
- Ungleichheit: (<>)
- Größer-Gleich: (>=)
- Kleiner-Gleich: (<=)
- Größer/Kleiner: (><) mit absoluter Gleichheit
- Größer: (>)
- Kleiner: (<)

Kompatibilität

Aus Kompatibilitätsgründen kann die Prüfung auf relative Gleichheit bei (>) und (<) durch Setzen von Maschinendatum MD10280 $MN_ PROG_FUNCTION_MASK Bit0 = 1 deaktiviert werden.

Hinweis

Vergleiche mit Daten vom Typ REAL sind aus den genannten Gründen generell mit einer gewissen Ungenauigkeit behaftet. Bei nicht akzeptablen Abweichungen muss auf INTEGER-Rechnung ausgewichen werden, indem die Operanden mit einem Genauigkeitsfaktor multipliziert und danach mit TRUNC abgeschnitten werden.

Synchronaktionen

Das beschriebene Verhalten der Vergleichsbefehle gilt auch bei Synchronaktionen.

Beispiele

Beispiel 1: Genauigkeitsbetrachtungen

Programmcode	Kommentar
N40 R1=61.01 R2=61.02 R3=0.01	; Zuweisung der Anfangswerte
N41 IF ABS(R2-R1) > R3 GOTOF FEHLER	; Sprung würde bisher ausgeführt werden
N42 M30	; Programmende
N43 FEHLER: SETAL(66000)	;
R1=61.01 R2=61.02 R3=0.01	; Zuweisung der Anfangswerte
R11=TRUNC(R1*1000) R12=TRUNC(R2*1000) R13=TRUNC(R3*1000)	; Genauigkeitskorrektur
IF ABS(R12-R11) > R13 GOTOF FEHLER	; Sprung wird nicht mehr ausgeführt
M30	; Programmende
FEHLER: SETAL(66000)	;

Beispiel 2: Quotient beider Operanden bilden und auswerten

Programmcode	Kommentar
R1=61.01 R2=61.02 R3=0.01	; Zuweisung der Anfangswerte
IF ABS((R2-R1)/R3)-1) > 10EX-5 GOTOF FEHLER	; Sprung wird nicht ausgeführt
M30	; Programmende
FEHLER: SETAL(66000)	;

1.6 Minimum, Maximum und Bereich von Variablen (MINVAL, MAXVAL, BOUND)

Funktion

Mit den Befehlen MINVAL und MAXVAL können die Werte zweier Variablen miteinander verglichen werden. Als Ergebnis wird der kleinere Wert (bei MINVAL) bzw. größere Wert (bei MAXVAL) zurückgeliefert.

Mit dem Befehl BOUND kann geprüft werden, ob der Wert einer Prüfvariablen innerhalb eines definierten Wertebereichs liegt.

Syntax

```
<Kleinerer Wert>=MINVAL(<Variable1>,<Variable2>)
<Größerer Wert>=MAXVAL(<Variable1>,<Variable2>)
<Rückgabewert>=<BOUND>(<Minimum>,<Maximum>,<Prüfvariable>)
```

Bedeutung

MINVAL:	Ermittelt den **kleineren** Wert zweier Variablen (<Variable1>, <Variable2>)
<Kleinerer Wert>:	Ergebnisvariable für den Befehl MINVAL
	Wird auf den kleineren Variablenwert gesetzt.
MAXVAL:	Ermittelt den **größeren** Wert zweier Variablen (<Variable1>, <Variable2>)
<Größerer Wert>:	Ergebnisvariable für den Befehl MAXVAL
	Wird auf den größeren Variablenwert gesetzt.
BOUND:	Prüft, ob eine Variable (<Prüfvariable>) innerhalb eines definierten Wertebereichs liegt.
<Minimum>:	Variable, die den Minimalwert des Wertebereichs definiert
<Maximum>:	Variable, die den Maximalwert des Wertebereichs definiert
<Rückgabewert>:	Ergebnisvariable für den Befehl BOUND
	Wenn der Wert der Prüfvariablen innerhalb des definierten Wertebereichs liegt, dann wird die Ergebnisvariable auf den Wert der Prüfvariablen gesetzt.
	Wenn der Wert der Prüfvariablen größer als der Maximalwert ist, dann wird die Ergebnisvariable auf den Maximalwert des Definitionsbereichs gesetzt.
	Wenn der Wert der Prüfvariablen kleiner als der Minimalwert ist, dann wird die Ergebnisvariable auf den Minimalwert des Definitionsbereichs gesetzt.

Hinweis

MINVAL, MAXVAL und BOUND können auch in Synchronaktionen programmiert werden.

Hinweis

Verhalten bei Gleichheit

Bei Gleichheit wird bei MINVAL/MAXVAL dieser gleiche Wert geliefert. Bei BOUND wird der Wert der zu prüfenden Variablen wieder zurückgegeben.

Beispiel

```
Programmcode                                    Kommentar
DEF REAL rVar1=10.5, rVar2=33.7, rVar3, rVar4, rVar5, rValMin, rValMax, rRetVar
rValMin=MINVAL(rVar1,rVar2)                     ; rValMin wird auf den Wert 10.5 gesetzt.
rValMax=MAXVAL(rVar1,rVar2)                     ; rValMax wird auf den Wert 33.7 gesetzt.
rVar3=19.7
rRetVar=BOUND(rVar1,rVar2,rVar3)                ; rVar3 liegt innerhalb der Grenzen, rRetVar wird auf
                                                  19.7 gesetzt.
rVar3=1.8
rRetVar=BOUND(rVar1,rVar2,rVar3)                ; rVar3 liegt unterhalb der Minimumgrenze, rRetVar wird
                                                  auf 10.5 gesetzt.

rVar3=45.2
rRetVar=BOUND(rVar1,rVar2,rVar3)                ; rVar3 liegt oberhalb der Maximumgrenze, rRetVar wird
                                                  auf 33.7 gesetzt.
```

1.7 Priorität der Operationen

Funktion

Jedem Operator ist eine Priorität zugeordnet. Bei der Auswertung eines Ausdrucks werden stets die Operatoren höherer Priorität zuerst angewandt. Bei gleichrangigen Operatoren erfolgt die Auswertung von links nach rechts.

In arithmetischen Ausdrücken kann durch runde Klammern die Abarbeitungsreihenfolge aller Operatoren festgelegt und damit von den normalen Prioritätsregeln abgewichen werden.

Reihenfolge der Operatoren

Von der höchsten zur niedrigsten Priorität

1.	NOT, B_NOT	Verneinung, bitweise Verneinung
2.	*, /, DIV, MOD	Multiplikation, Division
3.	+, –	Addition, Subtraktion
4.	B_AND	bitweises UND
5.	B_XOR	bitweises exklusives ODER
6.	B_OR	bitweises ODER
7.	AND	UND
8.	XOR	exklusives ODER
9.	OR	ODER
10.	<<	Verkettung von Strings, Ergebnistyp STRING
11.	==, <>, >, <, >=, <=	Vergleichsoperatoren

Hinweis

Der Kettungsoperator ":" für Frames darf nicht mit anderen Operatoren in einem Ausdruck vorkommen. Eine Prioritätseinstufung für diesen Operator ist deshalb nicht erforderlich.

Beispiel If-Anweisung

```
If (otto==10) and (anna==20) gotof end
```

1.8 Mögliche Typenkonvertierungen

Funktion

Typkonvertierung bei Zuweisung

Der konstante Zahlenwert, die Variable oder der Ausdruck, der einer Variablen zugewiesen wird, muss mit dem Typ dieser Variablen verträglich sein. Ist dies gegeben, so wird bei der Zuweisung der Typ automatisch umgewandelt.

Mögliche Typkonvertierungen

nach	REAL	INT	BOOL	CHAR	STRING	AXIS	FRAME
von							
REAL	ja	ja*	ja[1])	ja*	–	–	–
INT	ja	ja	ja[1])	ja[2])	–	–	–
BOOL	ja	ja	ja	ja	ja	–	–
CHAR	ja	ja	ja[1])	ja	ja	–	–
STRING	–	–	ja[4])	ja[3])	ja	–	–
AXIS	–	–	–	–	–	ja	–
FRAME	–	–	–	–	–	–	ja

Erklärungen

* Bei Typumwandlung von REAL nach INT wird bei gebrochenem Wert >= 0.5 aufgerundet, ansonsten wird abgerundet (vgl. Funktion ROUND)

1) Wert <> 0 entspricht TRUE, Wert == 0 entspricht FALSE

2) Wenn der Wert im zulässigen Zahlenbereich liegt

3) Wenn nur 1 Zeichen

4) Stringlänge 0 = >FALSE, ansonsten TRUE

Hinweis

Ist beim Konvertieren ein Wert größer als der Zielbereich, erfolgt eine Fehlermeldung.

Treten in einem Ausdruck gemischte Typen auf, so wird eine Typanpassung automatisch durchgeführt. Typumwandlungen sind auch in Synchronaktionen möglich, siehe Kapitel Bewegungssynchronaktionen, Implizite Typwandlung".

1.9 Stringoperationen

Stringoperationen

Neben den klassischen Operationen "Zuweisung" und "Vergleich" sind folgende Stringoperationen möglich:

- Typenkonvertierung nach STRING (AXSTRING)
- Typenkonvertierung von STRING (NUMBER, ISNUMBER, AXNAME)
- Verkettung von Strings (<<)
- Wandlung in Klein-/Großbuchstaben (TOLOWER, TOUPPER)
- Länge eines Strings bestimmen (STRLEN)
- Zeichen/String in String suchen (INDEX, RINDEX, MINDEX, MATCH)
- Auswahl eines Teilstrings (SUBSTR)
- Selektion eines Einzelzeichens (STRINGVAR, STRINGFELD)

Sonderbedeutung des 0-Zeichens

Das 0-Zeichen wird intern als Enderkennung eines Strings interpretiert. Wird ein Zeichen durch das 0-Zeichen ersetzt, wird der String damit verkürzt.

Beispiel:

Programmcode	**Kommentar**
`DEF STRING[20] STRG="Achse . steht"`	
`STRG[6]="X"`	
`MSG(STRG)`	`; Liefert die Meldung "Achse X steht".`
`STRG[6]=0`	
`MSG(STRG)`	`; Liefert die Meldung "Achse".`

1.9.1 Typenkonvertierung nach STRING (AXSTRING)

Funktion

Durch die Funktion "Typkonvertierung nach STRING" lassen sich Variablen unterschiedlichen Typs als Bestandteil einer Meldung (MSG) nutzen.

Erfolgt bei Verwendung des Operators **<<** implizit für die Datentypen INT, REAL, CHAR und BOOL (siehe " Verkettung von Strings (<<) (Seite 75) ").

Ein INT-Wert wird in die normal lesbare Form umgewandelt. Bei REAL-Werten werden bis zu 10 Nachkommastellen angegeben.

Mit dem Befehl `AXSTRING` können Variable vom Typ AXIS nach STRING gewandelt werden.

Syntax

```
<STRING_ERG> = << <bel._Typ>
<STRING_ERG> = AXSTRING(<Achsbezeichner>)
```

Bedeutung

`<STRING_ERG>`:	Variable für das Ergebnis der Typkonvertierung Typ: STRING
`<bel._Typ>`:	Variablen-Typen INT, REAL, CHAR, STRING und BOOL
`AXSTRING`:	Der Befehl `AXSTRING` liefert den angegebenen Achsbezeichner als String.
`<Achsbezeichner>`:	Variable für Achsbezeichner Typ: AXIS

Hinweis

FRAME-Variablen können nicht konvertiert werden.

Beispiele

Beispiel 1:

```
MSG("Position:"<<$AA_IM[X])
```

Beispiel 2: AXSTRING

Programmcode	Kommentar
`DEF STRING[32] STRING_ERG`	
`STRING_ERG=AXSTRING(X)`	`; STRING_ERG == "X"`

1.9.2 Typenkonvertierung von STRING (NUMBER, ISNUMBER, AXNAME)

Funktion

Mit dem Befehl NUMBER wird von STRING nach REAL konvertiert. Die Konvertierbarkeit kann mit dem Befehl ISNUMBER überprüft werden.

Mit dem Befehl AXNAME wird ein String in den Datentyp AXIS konvertiert.

Syntax

```
<REAL_ERG>=NUMBER("<String>")
<BOOL_ERG>=ISNUMBER("<String>")
<AXIS_ERG>=AXNAME("<String>")
```

Bedeutung

NUMBER:	Der Befehl NUMBER liefert die durch den <String> dargestellte Zahl als REAL-Wert zurück.		
<String>:	Zu konvertierende Variable vom Typ STRING		
<REAL_ERG>:	Variable für das Ergebnis der Typkonvertierung mit NUMBER		
	Typ:	REAL	
ISNUMBER:	Mit dem Befehl ISNUMBER kann überprüft werden, ob der <String> in eine gültige Zahl gewandelt werden kann.		
<BOOL_ERG>:	Variable für das Ergebnis der Abfrage mit ISNUMBER		
	Typ:	BOOL	
	Wert:	TRUE	ISNUMBER liefert den Wert TRUE, wenn der <String> eine nach den Regeln der Sprache gültige REAL-Zahl darstellt.
		FALSE	Liefert ISNUMBER den Wert FALSE, wird bei Aufruf von NUMBER mit dem gleichen <String> Alarm ausgelöst.
AXNAME:	Der Befehl AXNAME wandelt den angegebenen <String> in einen Achsbezeichner.		
	Hinweis: Kann der <String> keinem projektierten Achsbezeichner zugeordnet werden, wird ein Alarm ausgelöst.		
<AXIS_ERG>:	Variable für das Ergebnis der Typkonvertierung mit AXNAME		
	Typ:	AXIS	

Beispiel

```
Programmcode                                    Kommentar
DEF BOOL BOOL_ERG
DEF REAL REAL_ERG
DEF AXIS AXIS_ERG
BOOL_ERG=ISNUMBER("1234.9876Ex-7")              ; BOOL_ERG == TRUE
BOOL_ERG=ISNUMBER("1234XYZ")                    ; BOOL_ERG == FALSE
REAL_ERG=NUMBER("1234.9876Ex-7")                ; REAL_ERG == 1234.9876Ex-7
AXIS_ERG=AXNAME("X")                            ; AXIS_ERG == X
```

1.9.3 Verkettung von Strings (<<)

Funktion

Die Funktion "Verkettung von Strings" schafft die Möglichkeit, einen String aus einzelnen Bestandteilen zusammensetzen zu können.

Realisiert wird die Verkettung über den Operator "<<". Dieser Operator hat für alle Kombinationen der Basistypen CHAR, BOOL, INT, REAL und STRING als Zieltyp STRING. Eine eventuell notwendige Konvertierung wird nach den bestehenden Regeln vorgenommen.

Syntax

```
<bel._Typ> << <bel._Typ>
```

Bedeutung

`<bel._Typ>`: Variable vom Typ CHAR, BOOL, INT, REAL oder STRING

`<<` : Operator für die Verkettung von Variablen (<bel._Typ>) zu einer zusammengesetzten Zeichenkette (Typ STRING).

Dieser Operator ist auch alleinig als sog. "unäre" Variante verfügbar. So ist es möglich, eine explizite Typwandlung nach STRING auszuführen (nicht für FRAME und AXIS):

```
<< <bel._Typ>
```

Beispielsweise lässt sich so eine Meldung oder ein Kommando aus Textlisten zusammensetzen und Parameter (etwa ein Bausteinname) einfügen:

```
MSG(STRG_TAB[LOAD_IDX]<<BAUSTEIN_NAME)
```

VORSICHT
Die Zwischenergebnisse bei der Stringverkettung dürfen die maximale Stringlänge nicht überschreiten.

Hinweis

Die Typen FRAME und AXIS können nicht zusammen mit dem Operator "<<" verwendet werden.

Beispiele

Beispiel 1: Verkettung von Strings

```
Programmcode                                    Kommentar
DEF INT IDX=2
DEF REAL VALUE=9.654
DEF STRING[20] STRG="INDEX:2"
IF STRG=="Index:"<<IDX GOTOF NO_MSG
MSG("Index:"<<IDX<<"/Wert:"<<VALUE)             ; Anzeige:
                                                  "Index:2/Wert:9.654"
NO_MSG:
```

Beispiel 2: Explizite Typkonvertierung mit <<

```
Programmcode            Kommentar
DEF REAL VALUE=3.5
<<VALUE                 ; Die angegebene Variable vom Typ REAL wird in den Typ
                          STRING konvertiert.
```

1.9.4 Wandlung in Klein-/Großbuchstaben (TOLOWER, TOUPPER)

Funktion

Die Funktion "Wandlung in Klein-/Großbuchstaben" erlaubt es, alle Buchstaben einer Zeichenkette in eine einheitliche Darstellung zu wandeln.

Syntax

```
<STRING_ERG>=TOUPPER("<String>")
<STRING_ERG>=TOLOWER("<String>")
```

Bedeutung

TOUPPER:	Mit dem Befehl TOUPPER werden alle Buchstaben einer Zeichenkette in **Groß**buchstaben umgewandelt.
TOLOWER:	Mit dem Befehl TOLOWER werden alle Buchstaben einer Zeichenkette in **Klein**buchstaben umgewandelt.
<String>:	Zeichenkette, die umgewandelt werden soll Typ: STRING
<STRING_ERG>:	Variable für das Ergebnis der Umwandlung Typ: STRING

Beispiel

Da es auch möglich ist, Benutzereingaben an der Bedienoberfläche anzustoßen, kann eine einheitliche Darstellung mit Klein- oder Großbuchstaben erreicht werden:

Programmcode

```
DEF STRING [29] STRG
...
IF "LEARN.CNC"==TOUPPER(STRG) GOTOF LOAD_LEARN
```

1.9.5 Länge eines Strings bestimmen (STRLEN)

Funktion

Mit dem Befehl STRLEN ist es möglich, die Länge einer Zeichenkette zu bestimmen.

Syntax

```
<INT_ERG>=STRLEN("<STRING>")
```

Bedeutung

STRLEN:	Mit dem Befehl STRLEN wird die Länge der angegebenen Zeichenkette bestimmt. Es wird die Anzahl der Zeichen zurückgegeben, die - vom Anfang der Zeichenkette an gezählt - keine 0-Zeichen sind.
<String>:	Zeichenkette, deren Länge bestimmt werden soll Typ: STRING
<INT_ERG>:	Variable für das Ergebnis der Bestimmung Typ: INT

Beispiel

Die Funktion im Zusammenhang mit dem Einzelzeichenzugriff ermöglicht es, das Ende einer Zeichenkette zu bestimmen:

Programmcode

```
IF (STRLEN(BAUSTEIN_NAME)>10) GOTOF FEHLER
```

1.9.6 Zeichen/String in String suchen (INDEX, RINDEX, MINDEX, MATCH)

Funktion

Diese Funktionalität erlaubt es, einzelne Zeichen bzw. einen String in einem weiteren String zu suchen. Die Funktionsergebnisse geben an, an welcher Position des Strings das Zeichen/der String im zu untersuchenden String gefunden wurde.

Syntax

```
INT_ERG=INDEX(STRING,CHAR) ; Ergebnistyp: INT
INT_ERG=RINDEX(STRING,CHAR) ; Ergebnistyp: INT
INT_ERG=MINDEX(STRING,STRING) ; Ergebnistyp: INT
INT_ERG=MATCH(STRING,STRING) ; Ergebnistyp: INT
```

Semantik

Suchfunktionen: Sie liefern die Position im String (erster Parameter) zurück, wo die Suche erfolgreich war. Kann das Zeichen/der String nicht gefunden werden, wird der Wert -1 zurückgegeben. Das erste Zeichen hat dabei die Position 0.

Bedeutung

INDEX:	sucht das als zweiten Parameter angegebene Zeichen (von vorne) im ersten Parameter.
RINDEX:	sucht das als zweiten Parameter angegebene Zeichen (von hinten) im ersten Parameter.
MINDEX:	entspricht der Funktion INDEX, außer, dass eine Liste von Zeichen (als String) übergeben wird, von denen der Index des ersten gefundenen Zeichens zurückgegeben wird.
MATCH:	sucht einen String in einem String.

So lassen sich Strings nach bestimmten Kriterien zerlegen, etwa an Positionen mit Leerzeichen oder Pfadtrennzeichen ("/").

Beispiel

Zerlegen einer Eingabe in Pfad- und Bausteinnamen

```
Programmcode                                                    Kommentar
DEF INT PFADIDX, PROGIDX
DEF STRING[26] EINGABE
DEF INT LISTIDX
EINGABE = "/_N_MPF_DIR/_N_EXECUTE_MPF"
LISTIDX = MINDEX (EINGABE, "M,N,O,P") + 1                       ; Als Wert in LISTIDX wird 3
                                                                  zurückgeliefert; da "N" das erste
                                                                  Zeichen im Parameter EINGABE, aus der
                                                                  Auswahlliste von vorne, ist.
PFADIDX = INDEX (EINGABE, "/") +1                               ; damit gilt: PFADIDX = 1
PROGIDX = RINDEX (EINGABE, "/") +1                              ; damit gilt: PROGIDX = 12
                                                                  mit Hilfe der im nächsten Abschnitt
                                                                  eingeführten Funktion SUBSTR läßt
                                                                  sich
                                                                  die Variable EINGABE in die
                                                                  Komponenten ;"Pfad";und "Baustein"
                                                                  zerlegen:
VARIABLE = SUBSTR (EINGABE, PFADIDX, PROGIDX-PFADIDX-1)         ; liefert dann "_N_MPF_DIR"
VARIABLE = SUBSTR (EINGABE, PROGIDX)                            ; liefert dann "_N_EXECUTE_MPF"
```

1.9.7 Auswahl eines Teilstrings (SUBSTR)

Funktion

Diese Funktionalität erlaubt es, einen Teilstring aus einem String herauszulösen. Dazu wird der Index des ersten Zeichens und ggf. die gewünschte Länge angegeben. Wird die Längeninformation nicht angegeben, ist der Reststring gemeint.

Syntax

`STRING_ERG = SUBSTR (STRING,INT)` ; **Ergebnistyp: INT**

`STRING_ERG = SUBSTR(STRING,INT, INT)` ; **Ergebnistyp: INT**

Semantik

Im ersten Fall wird der Teilstring ab der Position, die durch den zweiten Parameter festgelegt ist, bis zum Ende des Strings zurückgegeben.

Im zweiten Fall ist der Ergebnisstring auf die maximale Länge, gegeben durch den dritten Parameter, begrenzt.

Liegt die Anfangsposition hinter dem Stringende, wird der Leerstring (" ") zurückgegeben.

Ist die Anfangsposition oder die Länge negativ, wird ein Alarm ausgelöst.

Beispiel

```
Programmcode                                          Kommentar
DEF STRING[29] ERG
ERG = SUBSTR ("QUITTUNG:10 bis 99", 10, 2)            ; damit gilt: ERG == "10"
```

1.9.8 Selektion eines Einzelzeichens (STRINGVAR, STRINGFELD)

Funktion

Diese Funktionalität erlaubt es, die einzelnen Zeichen eines Strings zu selektieren. Dies trifft sowohl auf den lesenden als auch auf den schreibenden Zugriff zu.

Syntax

CHAR_ERG = STRINGVAR [IDX] ; **Ergebnistyp: CHAR**

CHAR_ERG = STRINGFELD [IDX_FELD, IDX_CHAR] ; **Ergebnistyp: CHAR**

Semantik

Es wird das Zeichen innerhalb des Strings gelesen/geschrieben, das an der angegebenen Stelle steht. Ist die Positionsangabe negativ oder größer als der String, wird ein Alarm ausgelöst.

Beispiel Meldungen:

Einsetzen eines Achsbezeichners in einem vorgefertigten String.

```
Programmcode                                                    Kommentar
DEF STRING [50] MELDUNG = "Achse n hat Position erreicht"
MELDUNG [6] = "X"
MSG (MELDUNG)                                                   ; liefert die Meldung "Achse X
                                                                  hat Position erreicht"
```

Parameter

Der Einzelzeichenzugriff ist nur auf vom Anwender definierte Variablen (LUD-,GUD- und PUD-Daten) möglich.

Außerdem ist diese Art des Zugriffs bei einem Unterprogrammaufruf nur für Parameter vom Typ "Call-By-Value" möglich.

Beispiele

Beispiel 1: Einzelzeichenzugriff auf ein System-, Maschinendatum, …

```
Programmcode                                    Kommentar
DEF STRING [50] STRG
DEF CHAR QUITTUNG
…
STRG = $P_MMCA
QUITTUNG = STRG [0]                             ;  Auswerten der Quittungskomponente
```

Beispiel 2: Einzelzeichenzugriff bei Call-By-Reference-Parameter

```
Programmcode                                    Kommentar
DEF STRING [50] STRG
DEF CHAR CHR1
EXTERN UP_CALL (VAR CHAR1)                      ;  Call-By-Reference-Parameter!
…
CHR1 = STRG [5]
UP_CALL (CHR1)                                  ;  Call-By-Reference
STRG [5] = CHR1
```

1.10 Programmsprünge und -verzweigungen

1.10.1 Rücksprung auf Programmanfang (GOTOS)

Funktion

Mit dem Befehl GOTOS ist es möglich, zur Programmwiederholung an den Anfang eines Haupt- oder Unterprogramms zurückzuspringen.

Über Maschinendaten kann eingestellt werden, dass bei jedem Rücksprung auf den Programmanfang:

- die Programmlaufzeit auf "0" gesetzt wird.
- die Werkstückzählung um den Wert "1" erhöht wird.

Syntax

```
GOTOS
```

Bedeutung

GOTOS: Sprunganweisung mit Sprungziel Programmanfang.

Die Ausführung wird gesteuert über das NC/PLC-Nahtstellensignal:

DB21, ... DBX384.0 (Programmverzweigung steuern)

Wert:	Bedeutung:
0	Kein Rücksprung auf den Programmanfang. Die Programmbearbeitung wird mit dem nächsten Teileprogrammsatz nach GOTOS fortgeführt.
1	Rücksprung auf den Programmanfang. Das Teileprogramm wird wiederholt.

Randbedingungen

- GOTOS löst intern ein STOPRE (Vorlaufstopp) aus.
- Bei einem Teileprogramm mit Datendefinitionen (LUD-Variablen) wird mit GOTOS auf den ersten Programmsatz nach dem Definitionsabschnitt gesprungen, d. h. die Datendefinitionen werden nicht erneut ausgeführt. Die definierten Variablen behalten daher den im GOTOS-Satz erreichten Wert und werden nicht auf die im Definitionsabschnitt programmierten Standardwerte zurückgesetzt.
- In Synchronaktionen und Technologie-Zyklen steht der Befehl GOTOS nicht zur Verfügung.

Beispiel

Programmcode	Kommentar
N10 ...	; Programmanfang.
...	
N90 GOTOS	; Sprung an den Programmanfang.
...	

1.10.2 Programmsprünge auf Sprungmarken (GOTOB, GOTOF, GOTO, GOTOC)

Funktion

In einem Programm können Sprungmarken (Labels) gesetzt werden, auf die von anderen Stellen innerhalb desselben Programms mit dem Befehlen GOTOF, GOTOB, GOTO bzw. GOTOC gesprungen werden kann. Die Programmbearbeitung wird dann mit der Anweisung fortgesetzt, die unmittelbar nach der Sprungmarke folgt. Dadurch sind Verzweigungen innerhalb des Programms realisierbar.

Neben den Sprungmarken sind als Sprungziele auch Haupt- und Nebensatznummern möglich.

Wenn vor der Sprunganweisung eine Sprungbedingung (IF ...) formuliert ist, dann erfolgt der Programmsprung nur dann, wenn die Sprungbedingung erfüllt ist.

Syntax

```
GOTOB <Sprungziel>
IF <Sprungbedingung> = TRUE GOTOB <Sprungziel>
GOTOF <Sprungziel>
IF <Sprungbedingung> = TRUE GOTOF <Sprungziel>
GOTO <Sprungziel>
IF <Sprungbedingung> = TRUE GOTO <Sprungziel>
GOTOC <Sprungziel>
IF <Sprungbedingung> = TRUE GOTOC <Sprungziel>
```

Bedeutung

GOTOB:	Sprunganweisung mit Sprungziel in Richtung Programmanfang.
GOTOF:	Sprunganweisung mit Sprungziel in Richtung Programmende.
GOTO:	Sprunganweisung mit Sprungzielsuche. Die Suche erfolgt erst in Richtung Programmende, dann in Richtung Programmanfang.
GOTOC:	Wirkung wie GOTO mit dem Unterschied, dass der Alarm 14080 "Sprungziel nicht gefunden" unterdrückt wird. Das bedeutet, dass die Programmbearbeitung im Falle einer ergebnislosen Sprungzielsuche nicht abgebrochen wird, sondern mit der auf den Befehl GOTOC folgenden Programmzeile fortgesetzt wird.

`<Sprungziel>`:	Sprungzielparameter	
	Mögliche Angaben sind:	
	<Sprungmarke>:	Sprungziel ist die im Programm gesetzte Sprungmarke mit benutzerdefiniertem Namen: `<Sprungmarke>:`
	<Satznummer>:	Sprungziel ist eine Haupt- oder Nebensatznummer (z. B.: `200`, `N300`)
	Variable vom Typ STRING:	Variables Sprungziel. Die Variable steht für eine Sprungmarke oder eine Satznummer.
`IF`:	Schlüsselwort zur Formulierung der Sprungbedingung.	
	Die Sprungbedingung lässt alle Vergleichs- und logischen Operationen zu (Ergebnis: TRUE oder FALSE). Der Programmsprung wird ausgeführt, wenn das Ergebnis dieser Operation TRUE ist.	

Hinweis

Sprungmarken (Labels)

Sprungmarken stehen immer am Anfang eines Satzes. Wenn eine Programmnummer vorhanden ist, steht die Sprungmarke unmittelbar nach der Satznummer.

Für die Benennung von Sprungmarken gelten folgende Regeln:

- Anzahl an Zeichen:
  - mindestens 2
  - höchstens 32
- Erlaubte Zeichen sind:
  - Buchstaben
  - Ziffern
  - Unterstriche
- Die ersten beiden Zeichen müssen Buchstaben oder Unterstriche sein.
- Nach dem Namen der Sprungmarke folgt ein Doppelpunkt (":").

Randbedingungen

- Sprungziel kann nur ein Satz mit Sprungmarke oder Satznummer sein, der **innerhalb** des Programms liegt.
- Eine Sprunganweisung ohne Sprungbedingung muss in einem separaten Satz programmiert werden. Bei Sprunganweisungen mit Sprungbedingungen gilt diese Einschränkung nicht. Hier können mehrere Sprunganweisungen in einem Satz formuliert werden.
- Bei Programmen mit Sprunganweisungen ohne Sprungbedingungen muss das Programmende `M2`/`M30` nicht zwangsläufig am Programmende stehen.

Beispiele

Beispiel 1: Sprünge auf Sprungmarken

```
Programmcode                    Kommentar
N10 …
N20 GOTOF Label_1               ; Sprung in Richtung Programmende zur Sprungmarke
                                  "Label_1".
N30 …
N40 Label_0: R1=R2+R3           ; Sprungmarke "Label_0" gesetzt.
N50 …
N60 Label_1:                    ; Sprungmarke "Label_1" gesetzt.
N70 …
N80 GOTOB Label_0               ; Sprung in Richtung Programmanfang zur Sprungmarke
                                  "Label_0".
N90 …
```

Beispiel 2: Indirekter Sprung auf Satznummer

```
Programmcode                    Kommentar
N5 R10=100
N10 GOTOF "N"<<R10              ; Sprung auf den Satz, dessen Satznummer in R10
                                  steht.
...
N90 ...
N100 ...                        ; Sprungziel
N110 ...
...
```

Beispiel 3: Sprung auf variables Sprungziel

```
Programmcode                    Kommentar
DEF STRING[20] ZIEL
ZIEL = "Marke2"
GOTOF ZIEL                      ; Sprung in Richtung Programmende zum variablen
                                  Sprungziel ZIEL.
Marke1: T="Bohrer1"
...
Marke2: T="Bohrer2"             ; Sprungziel
...
```

Beispiel 4: Sprung mit Sprungbedingung

```
Programmcode                                    Kommentar
N40 R1=30 R2=60 R3=10 R4=11 R5=50 R6=20         ; Zuweisung der Anfangswerte.
N41 LA1: G0 X=R2*COS(R1)+R5 Y=R2*SIN(R1)+R6     ; Sprungmarke LA1 gesetzt.
N42 R1=R1+R3 R4=R4-1
N43 IF R4>0 GOTOB LA1                           ; Wenn Sprungbedingung erfüllt,
                                                  dann Sprung in Richtung
                                                  Programmanfang zur Sprungmarke
                                                  LA1.
N44 M30                                         ; Programmende
```

1.10.3 Programmverzweigung (CASE ... OF ... DEFAULT ...)

Funktion

Die CASE-Funktion bietet die Möglichkeit, den aktuellen Wert (Typ: INT) einer Variablen oder einer Rechenfunktion zu überprüfen und abhängig vom Ergebnis an unterschiedliche Stellen im Programm zu springen.

Syntax

```
CASE(<Ausdruck>) OF <Konstante_1> GOTOF <Sprungziel_1> <Konstante_2>
GOTOF <Sprungziel_2> ... DEFAULT GOTOF <Sprungziel_n>
```

Bedeutung

CASE:	Sprunganweisung
<Ausdruck>:	Variable oder Rechenfunktion
OF:	Schlüsselwort zur Formulierung der bedingten Programmverzweigungen
<Konstante_1>:	Erster angegebener konstanter Wert für die Variable oder Rechenfunktion Typ: INT
<Konstante_2>:	Zweiter angegebener konstanter Wert für die Variable oder Rechenfunktion Typ: INT
DEFAULT:	Für die Fälle, in denen die Variable oder Rechenfunktion keinen der angegebenen konstanten Werte annimmt, kann mit der Anweisung DEFAULT ein Sprungziel bestimmt werden. **Hinweis:** Falls die DEFAULT-Anweisung nicht programmiert ist, wird in diesen Fällen der auf die CASE-Anweisung folgende Satz zum Sprungziel.

GOTOF:	Sprunganweisung mit Sprungziel in Richtung Programmende. Statt GOTOF sind auch alle anderen GOTO-Befehle programmierbar (siehe Thema "Programmsprünge auf Sprungmarken").
<Sprungziel_1>:	Auf dieses Sprungziel wird verzweigt, wenn der Wert der Variablen oder Rechenfunktion der ersten angegebenen Konstanten entspricht. Das Sprungziel kann wie folgt angegeben werden: <Sprungmarke>: Sprungziel ist die im Programm gesetzte Sprungmarke mit benutzerdefiniertem Namen: <Sprungmarke>: <Satznummer>: Sprungziel ist eine Haupt- oder Nebensatznummer (z. B.: 200, N300) Variable vom Typ STRING: Variables Sprungziel. Die Variable steht für eine Sprungmarke oder eine Satznummer.
<Sprungziel_2>:	Auf dieses Sprungziel wird verzweigt, wenn der Wert der Variablen oder Rechenfunktion der zweiten angegebenen Konstanten entspricht.
<Sprungziel_n>:	Auf dieses Sprungziel wird verzweigt, wenn der Wert der Variablen keinen der angegebenen konstanten Werte annimmt.

Beispiel

Programmcode

```
...
N20 DEF INT VAR1 VAR2 VAR3
N30 CASE(VAR1+VAR2-VAR3) OF 7 GOTOF Label_1 9 GOTOF Label_2 DEFAULT GOTOF Label_3
N40 Label_1: G0 X1 Y1
N50 Label_2: G0 X2 Y2
N60 Label_3: G0 X3 Y3
...
```

Die CASE-Anweisung aus N30 definiert folgende Programmverzeigungsmöglichkeiten:

1. Wenn der Wert der Rechenfunktion VAR1+VAR2-VAR3 = 7, dann springe zu dem Satz mit Sprungmarkendefinition "Label_1" (→ N40).
2. Wenn der Wert der Rechenfunktion VAR1+VAR2-VAR3 = 9, dann springe zu dem Satz mit Sprungmarkendefinition "Label_2" (→ N50).
3. Wenn der Wert der Rechenfunktion VAR1+VAR2-VAR3 weder 7 noch 9 beträgt, dann springe zu dem Satz mit Sprungmarkendefinition "Label_3" (→ N60).

1.11 Programmteilwiederholung (REPEAT, REPEATB, ENDLABEL, P)

Funktion

Die Programmteilwiederholung ermöglicht die Wiederholung bereits geschriebener Programmteile innerhalb eines Programms in beliebiger Zusammensetzung.

Die zu wiederholenden Programmzeilen bzw. Programmbereiche werden durch Sprungmarken (Labels) gekennzeichnet.

Hinweis

Sprungmarken (Labels)

Sprungmarken stehen immer am Anfang eines Satzes. Wenn eine Programmnummer vorhanden ist, steht die Sprungmarke unmittelbar nach der Satznummer.

Für die Benennung von Sprungmarken gelten folgende Regeln:

- Anzahl an Zeichen:
  - mindestens 2
  - höchstens 32
- Erlaubte Zeichen sind:
  - Buchstaben
  - Ziffern
  - Unterstriche
- Die ersten beiden Zeichen müssen Buchstaben oder Unterstriche sein.
- Nach dem Namen der Sprungmarke folgt ein Doppelpunkt (":").

Syntax

1. Einzelne Programmzeile wiederholen:

```
<Sprungmarke>: ...
...
REPEATB <Sprungmarke> P=<n>
...
```

2. Programmbereich zwischen Sprungmarke und REPEAT-Anweisung wiederholen:

```
<Sprungmarke>: ...
...
REPEAT <Sprungmarke> P=<n>
...
```

3. Bereich zwischen zwei Sprungmarken wiederholen:

```
<Start-Sprungmarke>: ...
...
<End-Sprungmarke>: ...
...
REPEAT <Start-Sprungmarke> <End-Sprungmarke> P=<n>
...
```

Hinweis

Die REPEAT-Anweisung mit den beiden Sprungmarken zu klammern, ist nicht möglich. Wird die <Start-Sprungmarke> vor der REPEAT-Anweisung gefunden und wird die <End-Sprungmarke> nicht vor der REPEAT-Anweisung erreicht, dann wird die Wiederholung zwischen <Start-Sprungmarke> und REPEAT-Anweisung durchgeführt.

4. Bereich zwischen Sprungmarke und ENDLABEL wiederholen:

```
<Sprungmarke>: ...
...
ENDLABEL: ...
...
REPEAT <Sprungmarke> P=<n>
...
```

Hinweis

Die REPEAT-Anweisung mit der <Sprungmarke> und dem ENDLABEL zu klammern, ist nicht möglich. Wird die <Sprungmarke> vor der REPEAT-Anweisung gefunden und wird ENDLABEL nicht vor der REPEAT-Anweisung erreicht, dann wird die Wiederholung zwischen <Sprungmarke> und REPEAT-Anweisung durchgeführt.

Bedeutung

REPEATB:	Befehl zum Wiederholen einer Programmzeile
REPEAT:	Befehl zum Wiederholen eines Programmbereichs
<Sprungmarke>:	Die <Sprungmarke> kennzeichnet: • die zu wiederholende Programmzeile (bei REPEATB) bzw. • den Beginn des zu wiederholenden Programmbereichs (bei REPEAT) Die mit der <Sprungmarke> gekennzeichnete Programmzeile kann vor oder nach der REPEAT-/REPEATB-Anweisung stehen. Gesucht wird zunächst in Richtung Programmanfang. Wird die Sprungmarke in dieser Richtung nicht gefunden, dann wird in Richtung Programmende gesucht. **Ausnahme:** Wenn der Programmbereich zwischen Sprungmarke und REPEAT-Anweisung wiederholt werden soll (siehe 2. unter Syntax), dann muss die mit der <Sprungmarke> gekennzeichnete Programmzeile **vor** der REPEAT-Anweisung stehen, da in diesem Fall **nur** in Richtung Programmanfang gesucht wird. Enthält die Zeile mit der <Sprungmarke> weitere Anweisungen, so werden diese bei jeder Wiederholung erneut ausgeführt.
ENDLABEL:	Schlüsselwort, welches das Ende eines zu wiederholenden Programmbereichs markiert Enthält die Zeile mit dem ENDLABEL weitere Anweisungen, so werden diese bei jeder Wiederholung erneut ausgeführt. ENDLABEL kann mehrfach im Programm verwendet werden.
P:	Adresse zur Angabe der Wiederholungsanzahl
<n>:	Anzahl an Programmteilwiederholungen Typ: INT Der zu wiederholende Programmteil wird <n> mal wiederholt. Nach der letzten Wiederholung wird das Programm mit der auf die REPEAT-/REPEATB-Zeile folgenden Zeile fortgesetzt. **Hinweis:** Ist kein P=<n> angegeben, wird der zu wiederholende Programmteil genau einmal wiederholt.

Beispiele

Beispiel 1: Einzelne Programmzeile wiederholen

```
Programmcode                          Kommentar
N10 POSITION1: X10 Y20
N20 POSITION2: CYCLE(0,,9,8)          ; Positionszyklus
N30 ...
N40 REPEATB POSITION1 P=5             ; Führe SATZ N10 fünfmal aus.
N50 REPEATB POSITION2                 ; Führe Satz N20 einmal aus.
N60 ...
N70 M30
```

Beispiel 2: Programmbereich zwischen Sprungmarke und REPEAT-Anweisung wiederholen

```
Programmcode                  Kommentar
N5 R10=15
N10 Begin: R10=R10+1          ; Breite
N20 Z=10-R10
N30 G1 X=R10 F200
N40 Y=R10
N50 X=-R10
N60 Y=-R10
N70 Z=10+R10
N80 REPEAT BEGIN P=4          ; Führe Bereich N10 bis N70 viermal aus.
N90 Z10
N100 M30
```

Beispiel 3: Bereich zwischen zwei Sprungmarken wiederholen

```
Programmcode                  Kommentar
N5 R10=15
N10 Begin: R10=R10+1              ; Breite
N20 Z=10-R10
N30 G1 X=R10 F200
N40 Y=R10
N50 X=-R10
N60 Y=-R10
N70 END: Z=10
N80 Z10
N90 CYCLE(10,20,30)
N100 REPEAT BEGIN END P=3         ; Führe Bereich N10 bis N70 dreimal aus.
N110 Z10
N120 M30
```

Beispiel 4: Bereich zwischen Sprungmarke und ENDLABEL wiederholen

```
Programmcode                    Kommentar
N10 G1 F300 Z-10
N20 BEGIN1:
N30 X10
N40 Y10
N50 BEGIN2:
N60 X20
N70 Y30
N80 ENDLABEL: Z10
N90 X0 Y0 Z0
N100 Z-10
N110 BEGIN3: X20
N120 Y30
N130 REPEAT BEGIN3 P=3      ; Führe Bereich N110 bis N120 dreimal aus.
N140 REPEAT BEGIN2 P=2      ; Führe Bereich N50 bis N80 zweimal aus.
N150 M100
N160 REPEAT BEGIN1 P=2      ; Führe Bereich N20 bis N80 zweimal aus.
N170 Z10
N180 X0 Y0
N190 M30
```

Beispiel 5: Fräsbearbeitung, Bohrposition mit verschiedenen Technologien bearbeiten

```
Programmcode                Kommentar
N10 ZENTRIERBOHRER()        ; Zentrierbohrer einwechseln.
N20 POS_1:                  ; Bohrpositionen 1
N30 X1 Y1
N40 X2
N50 Y2
N60 X3 Y3
N70 ENDLABEL:
N80 POS_2:                  ; Bohrpositionen 2
N90 X10 Y5
N100 X9 Y-5
N110 X3 Y3
N120 ENDLABEL:
N130 BOHRER()               ; Bohrer wechseln und Bohrzyklus.
N140 GEWINDE(6)             ; Gewindebohrer M6 einwechseln und Gewindezyklus.
N150 REPEAT POS_1           ; Wiederhole Programmabschnitt ab POS_1 einmal bis
                              ENDLABEL.
N160 BOHRER()               ; Bohrer wechseln und Bohrzyklus.
```

```
Programmcode                Kommentar
N170 GEWINDE(8)             ; Gewindebohrer M8 einwechseln und Gewindezyklus.
N180 REPEAT POS_2           ; Wiederhole Programmabschnitt ab POS_2 einmal bis
                              ENDLABEL.
N190 M30
```

Weitere Informationen

- Programmteilwiederholung kann geschachtelt aufgerufen werden. Jeder Aufruf belegt eine Unterprogrammebene.
- Ist während der Bearbeitung einer Programmteilwiederholung `M17` oder `RET` programmiert, so wird die Programmteilwiederholung abgebrochen. Das Programm wird mit dem auf die `REPEAT`-Zeile folgenden Satz fortgesetzt.
- In der aktuellen Programm-Anzeige wird die Programmteilwiederholung als eigene Unterprogrammebene angezeigt.
- Wird während der Programmteil-Bearbeitung Ebenenabbruch ausgelöst, so wird das Programm nach dem Aufruf der Programmteilbearbeitung fortgesetzt.

  Beispiel:

```
Programmcode                Kommentar
N5 R10=15
N10 BEGIN: R10=R10+1        ; Breite
N20 Z=10-R10
N30 G1 X=R10 F200
N40 Y=R10                   ; Ebenenabbruch
N50 X=-R10
N60 Y=-R10
N70 END: Z10
N80 Z10
N90 CYCLE(10,20,30)
N100 REPEAT BEGIN END P=3
N120 Z10                    ; Programmbearbeitung fortsetzen.
N130 M30
```

- Kontrollstrukturen und Programmteilwiederholung können kombiniert genutzt werden. Es sollte jedoch keine Überschneidungen geben. Eine Programmteilwiederholung sollte innerhalb eines Kontrollstruktur-Zweigs liegen bzw. eine Kontrollstruktur innerhalb einer Programmteilwiederholung.
- Bei der Mischung von Sprüngen und Programmteilwiederholung werden die Sätze rein sequentiell abgearbeitet. Erfolgt z. B. ein Sprung aus einer Programmteilwiederholung, so wird solange bearbeitet, bis das programmierte Programmteilende gefunden wird.

Beispiel:

Programmcode

```
N10 G1 F300 Z-10
N20 BEGIN1:
N30 X=10
N40 Y=10
N50 GOTOF BEGIN2
N60 ENDLABEL:
N70 BEGIN2:
N80 X20
N90 Y30
N100 ENDLABEL: Z10
N110 X0 Y0 Z0
N120 Z-10
N130 REPEAT BEGIN1 P=2
N140 Z10
N150 X0 Y0
N160 M30
```

Hinweis

Die `REPEAT`-Anweisung sollte hinter den Verfahrsätzen stehen.

1.12 Kontrollstrukturen

Funktion

Die Steuerung arbeitet die NC-Sätze standardmäßig in der programmierten Reihenfolge ab.

Diese Reihenfolge kann durch die Programmierung von alternativen Programmblöcken und Programmschleifen variiert werden. Die Programmierung dieser Kontrollstrukturen erfolgt mit den Kontrollstrukturelementen (Schlüsselwörtern) `IF...ELSE`, `LOOP`, `FOR`, `WHILE` und `REPEAT`.

⚠ VORSICHT

Kontrollstrukturen sind nur innerhalb des Anweisungsteils eines Programms möglich. Definitionen im Programmkopf können nicht bedingt oder wiederholt ausgeführt werden.

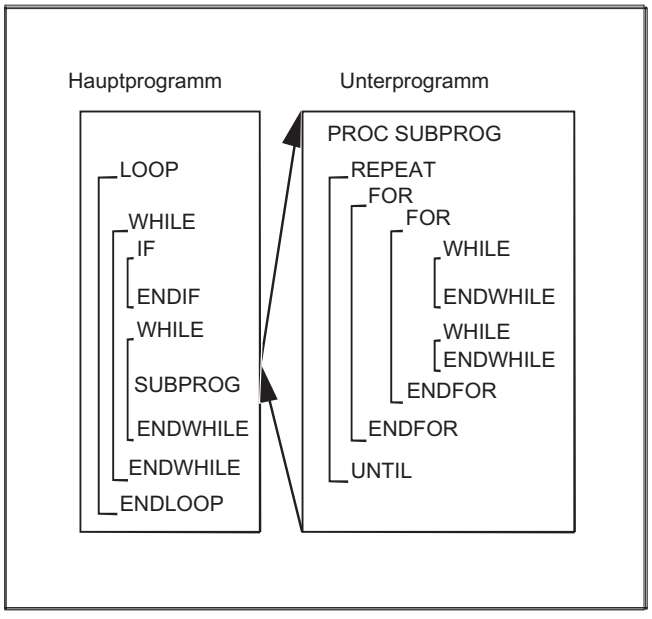
Schlüsselworte für Kontrollstrukturen dürfen ebenso wie Sprungziele nicht mit Makros überlagert werden. Eine Abprüfung bei der Makrodefinition findet nicht statt.

Wirksamkeit

Kontrollstrukturen gelten programm-lokal.

Schachtelungstiefe

Innerhalb jeder Unterprogrammebene ist eine Schachtelungstiefe von bis zu 16 Kontrollstrukturen möglich.



Laufzeitverhalten

Im standardmäßig aktiven Interpreterbetrieb kann durch Verwendung von Programmsprüngen ein schnellerer Programmablauf als mit Kontrollstrukturen erreicht werden.

In vorkompilierten Zyklen ist kein Unterschied zwischen Programmsprüngen und Kontrollstrukturen vorhanden.

Randbedingungen

- Sätze mit Kontrollstrukturelementen können nicht ausgeblendet werden.
- Sprungmarken (Labels) sind in Sätzen mit Kontrollstrukturelementen nicht erlaubt.
- Kontrollstrukturen werden interpretativ abgearbeitet. Bei Erkennen eines Schleifenendes wird unter Berücksichtigung der dabei gefundenen Kontrollstrukturen nach dem Schleifenanfang gesucht. Daher wird im Interpreterbetrieb die Blockstruktur eines Programms nicht komplett geprüft.
- Grundsätzlich empfiehlt sich, Kontrollstrukturen und Programmsprünge nicht gemischt zu verwenden.
- Bei Vorverarbeitung von Zyklen kann die korrekte Schachtelung von Kontrollstrukturen überprüft werden.

1.12.1 Programmschleife mit Alternative (IF, ELSE, ENDIF)

Funktion

Eine Konstruktion mit `IF` und `ELSE` wird verwendet, wenn die Programmschleife einen alternativen Programmblock enthalten soll: Wenn die `IF`-Bedingung erfüllt ist, dann wird der auf `IF` folgende Programmblock ausgeführt. Wenn die `IF`-Bedingung **nicht** erfüllt ist, dann wird der auf `ELSE` folgende alternative Programmblock ausgeführt.

Hinweis

Wenn keine Alternative erforderlich ist, dann kann eine IF-Schleife auch ohne `ELSE`-Anweisung und dem auf `ELSE` folgenden Programmblock programmiert werden.

Syntax

```
IF <Bedingung>
...
ELSE
...
ENDIF
```

Bedeutung

IF:	Leitet die IF-Schleife ein.
ELSE:	Leitet den alternativen Programmblock ein.
ENDIF:	Markiert das Ende der IF-Schleife und bewirkt Rücksprung auf den Schleifenanfang.
<Bedingung>:	Bedingung, die darüber entscheidet, welcher Programmblock durchlaufen wird.

Beispiel

Werkzeugwechselunterprogramm

Programmcode	**Kommentar**
PROC L6	; Werkzeugwechselroutine
N500 DEF INT TNR_AKTUELL	; Variable für aktive T-Nummer
N510 DEF INT TNR_VORWAHL	; Variable für vorgewählte T-Nummer
	; Aktuelles Werkzeug ermitteln
N520 STOPRE	
N530 IF $P_ISTEST	; Im Programmtest-Betrieb wird ...
N540 TNR_AKTUELL = $P_TOOLNO	; ... aus dem Programmkontext das "aktuelle" Werkzeug gelesen.
N550 ELSE	; Andernfalls wird ...
N560 TNR_AKTUELL = $TC_MPP6[9998,1]	; ... das Werkzeug der Spindel ausgelesen.
N570 ENDIF	
N580 GETSELT(TNR_VORWAHL)	; T-Nummer des vorgewählten Werkzeugs auf der Spindel lesen.
N590 IF TNR_AKTUELL <> TNR_VORWAHL	; Wenn das vorgewählte Werkzeug noch nicht das aktuelle Werkzeug ist, dann ...
N600 G0 G40 G60 G90 SUPA X450 Y300 Z300 D0	; ... Werkzeugwechselpunkt anfahren ...
N610 M206	; ... und Werkzeugwechsel ausführen.
N620 ENDIF	
N630 M17	

1.12.2 Endlos-Programmschleife (LOOP, ENDLOOP)

Funktion

Die Endlos-Schleife findet Verwendung in Endlos-Programmen. Am Schleifenende findet immer wieder der Rücksprung zum Schleifenanfang statt.

Syntax

```
LOOP
...
ENDLOOP
```

Bedeutung

`LOOP`:	Leitet die Endlosschleife ein.
`ENDLOOP`:	Markiert das Ende der Schleife und bewirkt Rücksprung auf den Schleifenanfang.

Beispiel

Programmcode

```
...
LOOP
MSG("keine Werkzeugschneide aktiv")
M0
STOPRE
ENDLOOP
...
```

1.12.3 Zählschleife (FOR ... TO ..., ENDFOR)

Funktion

Die Zählschleife wird verwendet, wenn ein Arbeitsablauf mit einer festen Anzahl von Durchläufen wiederholt werden soll.

Syntax

```
FOR <Variable> = <Anfangswert> TO <Endwert>
...
ENDFOR
```

Bedeutung

FOR:	Leitet die Zählschleife ein.
ENDFOR:	Markiert das Ende der Schleife und bewirkt Rücksprung auf den Schleifenanfang, solange der Endwert der Zählung noch nicht erreicht ist.
<Variable>:	Zählvariable, die vom Anfangs- bis zum Endwert hochgezählt wird und sich bei jedem Durchlauf um den Wert "1" erhöht. Typ INT oder REAL **Hinweis:** Der Typ REAL wird genommen, wenn z. B. R-Parameter für eine Zählschleife programmiert werden. Ist die Zählvariable vom Typ REAL, wird ihr Wert auf einen ganzzahligen Wert gerundet.
<Anfangswert>:	Anfangswert der Zählung Bedingung: Der Anfangswert muss kleiner sein als der Endwert.
<Endwert>:	Endwert der Zählung

Beispiele

Beispiel 1: INTEGER-Variable oder R-Parameter als Zählvariable

INTEGER-Variable als Zählvariable:

```
Programmcode                          Kommentar
DEF INT iVARIABLE1
R10=R12-R20*R1 R11=6
FOR iVARIABLE1= R10 TO R11            ; Zählvariable = INTEGER-Variable
 R20=R21*R22+R33
ENDFOR
M30
```

R-Parameter als Zählvariable:

```
Programmcode                          Kommentar
R11=6
FOR R10=R12-R20*R1 TO R11             ; Zählvariable = R-Parameter (Realvariable)
 R20=R21*R22+R33
ENDFOR
M30
```

Beispiel 2: Fertigung einer festen Teilestückzahl

```
Programmcode                          Kommentar
DEF INT STUECKZAHL                    ; Definiert Variable vom Typ INT mit Namen
                                        "STUECKZAHL".
FOR STUECKZAHL = 0 TO 100             ; Leitet die Zählschleife ein. Die Variable
                                        "STUECKZAHL" wird vom Anfangswert "0" bis zum
                                        Endwert "100" hochgezählt.
G01 …
ENDFOR                                ; Ende der Zählschleife.
M30
```

1.12.4 Programmschleife mit Bedingung am Schleifenanfang (WHILE, ENDWHILE)

Funktion

Bei einer WHILE-Schleife steht die Bedingung am Schleifenanfang. Solange die Bedingung erfüllt ist, wird die WHILE-Schleife durchlaufen.

Syntax

```
WHILE <Bedingung>
...
ENDWHILE
```

Bedeutung

`WHILE`:	Leitet die Programmschleife ein.
`ENDWHILE`:	Markiert das Ende der Schleife und bewirkt Rücksprung auf den Schleifenanfang.
`<Bedingung>`:	Bedingung, die erfüllt sein muss, damit die WHILE-Schleife durchlaufen wird.

Beispiel

```
Programmcode                                    Kommentar
...
WHILE $AA_IW[BOHRACHSE] > -10                   ; Aufruf der WHILE-Schleife unter folgender
                                                  Bedingung: der aktuelle WKS-Sollwert für
                                                  die Bohrachse muss größer -10 sein.
G1 G91 F250 AX[BOHRACHSE] = -1
ENDWHILE
...
```

1.12.5 Programmschleife mit Bedingung am Schleifenende (REPEAT, UNTIL)

Funktion

Bei einer REPEAT-Schleife steht die Bedingung am Schleifenende. Die REPEAT-Schleife wird einmal durchlaufen und solange wiederholt, bis die Bedingung erfüllt ist.

Syntax

```
REPEAT
...
UNTIL <Bedingung>
```

Bedeutung

`REPEAT`:	Leitet die Programmschleife ein.
`UNTIL`:	Markiert das Ende der Schleife und bewirkt Rücksprung auf den Schleifenanfang.
`<Bedingung>`:	Bedingung, die erfüllt sein muss, damit die REPEAT-Schleife nicht mehr durchlaufen wird.

Beispiel

```
Programmcode                    Kommentar
...
REPEAT                          ; Aufruf der REPEAT-Schleife.
...
UNTIL ...                       ; Prüfung, ob Bedingung erfüllt ist.
...
```

1.12.6 Programmbeispiel mit verschachtelten Kontrollstrukturen

```
Programmcode                                    Kommentar
LOOP
 IF NOT $P_SEARCH                               ; kein Satzsuchlauf
 G01 G90 X0 Z10 F1000
 WHILE $AA_IM[X] <= 100
 G1 G91 X10 F500                                ; Bohrbild
 Z-F100
 Z5
 ENDWHILE
 Z10
 ELSE
 MSG("Im Suchlauf wird nicht gebohrt")
 ENDIF
 $A_OUT[1] = 1                                  ; nächste Bohrplatte
 G4 F2
ENDLOOP
M30
```

1.13 Programmkoordinierung (INIT, START, WAITM, WAITMC, WAITE, SETM, CLEARM)

Funktion

Kanäle

Ein Kanal kann sein eigenes Programm, unabhängig von anderen Kanälen, abarbeiten. Damit sind die ihm zeitweise zugeordneten Achsen und Spindeln über das Programm steuerbar.

Bei der Inbetriebnahme können für die Steuerung zwei oder mehr Kanäle eingerichtet werden.

Programmkoordinierung

Sind mehrere Kanäle an der Fertigung eines Werkstücks beteiligt, so kann eine Synchronisation der Programmabläufe erforderlich werden.

Für diese Programmkoordinierung gibt es besondere Anweisungen (Kommandos). Sie stehen jeweils in einem Satz für sich.

Hinweis

Programmkoordinierung ist auch im eigenen Kanal möglich.

Anweisungen zur Programmkoordinierung

- Angabe mit absoluter Pfadangabe

	Dabei wird der absolute Pfad nach folgenden Regeln gebildet:
INIT (n,"/_HUGO_DIR/_N_name_MPF") oder	- aktuelles Directory/_N_name_MPF "aktuelles Directory" steht für das angewählte Werkstückdirectory oder das Standarddirectory /_N_MPF_DIR.
INIT (n,"/_N_MPF_DIR/_N_name_MPF")	- Anwahl eines bestimmten Programms zur Abarbeitung in einem bestimmten Kanal: n: Nummer des Kanals, Wert je nach Steuerungskonfiguration - Kompletter Programmname
Beispiel:	**bis SW 3:**
INIT(2,"/_N_WKS_DIR/_ABRICHT_MPF") G01F0.1 START	Zwischen einem **init**-Befehl (ohne Synchronisation) und einem **NC-Start** muss mindestens ein ausführbarer Satz stehen. Bei Unterprogrammaufrufen muss "_SPF" in der Pfadangabe ergänzt werden
INIT (2,"/_N_WKS_DIR/_N_UNTER_1_SPF")	

- Angabe mit relativer Pfadangabe

	Bei relativer Pfadangabe gelten dieselben Regeln wie für Unterprogrammaufrufe.
Beispiel:	
INIT(2,"ABRICHT")	
INIT(3,"UNTER_1_SPF")	Bei Unterprogrammaufrufe muss "_SPF" im Programmnamen ergänzt werden.

Parameter

Zum Datenaustausch zwischen den Programmen können die Variablen benutzt werden, über die Kanäle gemeinsam verfügen (NCK-spezifische globale Variable). Ansonsten wird die Programmerstellung für jeden Kanal getrennt vorgenommen.

`INIT(n, Pfadangabe, Quittungsmodus)`	Anweisung zur Abarbeitung in einem Kanal. Anwahl eines bestimmten Programms mit absoluter oder relativer Pfadangabe.
`START (n, n)`	Starten der angewählten Programme in den anderen Kanälen. n,n: Aufzählung der Kanalnummern: Wert je nach Steuerungskonfiguration
`WAITM (Marker-Nr., n, n, ...)`	Setzen des Markers "Marker-Nr." im eigenen Kanal. Vorhergehenden Satz mit Genauhalt beenden. Warten auf die Marker mit der gleichen "Marker-Nr." in den angegebenen Kanälen "n" (eigener Kanal muss nicht angegeben werden). Marker wird nach Synchronisation gelöscht. Gleichzeitig können max. 10 Marker pro Kanal gesetzt werden.
`WAITMC (Marker-Nr., n, n,`	Setzen des Markers "Marker-Nr." im eigenen Kanal. Genauhalt wird nur eingeleitet, wenn die anderen Kanäle den Marker noch nicht erreicht haben. Warten auf den Marker mit der gleichen "Marker-Nr." in den angegebenen Kanälen "n" (eigener Kanal muss nicht angegeben werden). Sobald Marker "Marker-Nr." in angegebenen Kanälen erreicht ist, die Bearbeitung ohne Beenden des Genauhalts fortsetzen.

`WAITE (n, n, ...)`	Warten auf das Programmende der angegebenen Kanäle (eigenen Kanal nicht angeben). Beispiel:Programmierung einer Verweilzeit nach dem Start-Befehl.
	`N30 START(2)`
	`N31 G4 F0.01`
	`N40 WAITE(2)`

`SETM (Marker-Nr., Marker-Nr.,`	Setzen der Marker "Marker-Nr." im eigenen Kanal, ohne Einfluss auf die laufende Bearbeitung. SETM() behält Gültigkeit über RESET und NC-START hinweg.

CLEARM (Marker-Nr., Marker-Nr.,	Löschen der Marker "Marker-Nr." im eigenen Kanal, ohne Einfluss auf die laufende Bearbeitung. Alle Marker im Kanal können mit CLEARM() gelöscht werden. CLEARM (0) löscht den Marker "0". CLEARM() behält Gültigkeit über RESET und NC-START hinweg.
n	Entsprechende Kanalnummer oder Kanalname

Hinweis

Alle obigen Befehle müssen in eigenständigen Sätzen stehen.

Die Anzahl der Marker ist abhängig von der eingebauten CPU.

Kanalnummern

Für die zu koordinierenden Kanäle können bis zu 10 Kanäle als Kanalnummer (Integerwert) angegeben werden.

Kanalnamen

Kanalnamen müssen über Variable (siehe Kapitel "Variable und Rechenparameter") in Nummern gewandelt werden oder anstelle von Kanalnummern können auch die über $MC_CHAN_NAME definierten Kanalnamen (Bezeichner oder Schlüsselwort) programmiert werden. Die definierten Namen müssen den NC-Sprachkonventionen entsprechen (D. h. die ersten beiden Zeichen müssen entweder aus Buchstaben oder ein Unterstrich bestehen).

⚠VORSICHT

Die Nummernzuordnung ist vor leichtfertiger Änderung zu sichern.

Die Namen dürfen nicht bereits in der NC in anderer Bedeutung wie z.B. als Schlüsselwort, Sprachbefehl, Achsname etc. vorhanden sein.

SETM() und CLEARM()

`SETM()` und `CLEARM()` können auch aus einer Synchronaktion heraus programmiert werden. Siehe Kapitel "Wartemarken setzen/ Löschen: SETM CLEARM"

Beispiel

Kanal mit Namen "MASCHINE" soll Kanalnummer 1 erhalten,

Kanal mit Namen "LADER" soll Kanalnummer 2 erhalten:

```
DEF INT MASCHINE=1, LADER=2
```

Die Variablen erhalten den gleichen Namen wie die Kanäle.

Damit lautet beispielsweise die Anweisung `START`:

```
START(MASCHINE)
```

Beispiel Programmkoordinierung

Kanal 1:

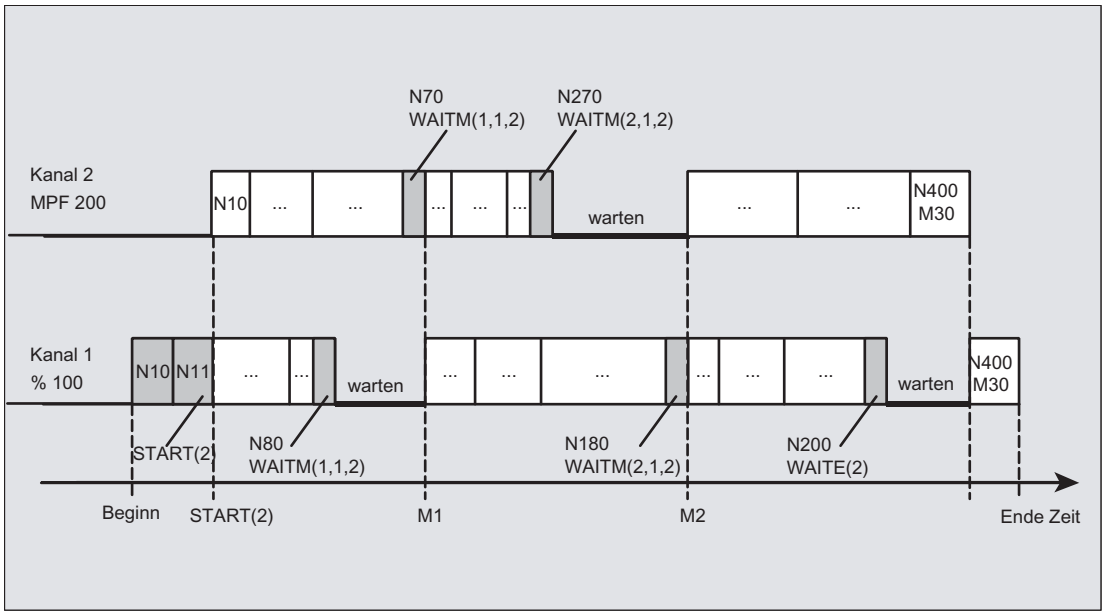
_N_MPF100_MPF

```
Programmcode                Kommentar
N10 INIT(2,"MPF200")
N11 START(2)                ; Bearbeiten im Kanal 2
...
N80 WAITM(1,1,2)            ; Warten auf WAIT-Marke 1 im Kanal 1 und
                              im Kanal 2 weiteres Bearbeiten in Kanal 1
...
N180 WAITM(2,1,2)           ; Warten auf WAIT-Marke 2 im Kanal 1 und
                              im Kanal 2 weiteres Bearbeiten in Kanal 1
...
N200 WAITE(2)               ; Warten auf Programmende des Kanals 2
N201 M30                    ; Programmende Kanal 1, Gesamtende
...
```

Kanal 2:

_N_MPF200_MPF

```
Programmcode          Kommentar
;$PATH=/_N_MPF_DIR
                      ; Bearbeiten im Kanal 2
N70 WAITM(1,1,2)      ; Warten auf WAIT-Marke 1 im Kanal 1 und im Kanal 2 weiteres
                        Bearbeiten in Kanal 1
...
N270 WAITM(2,1,2)     ; Warten auf WAIT-Marke 2 im Kanal 1 und im Kanal 2 weiteres
                        Bearbeiten in Kanal 2
...
N400 M30              ; Programmende des Kanals 2
```



Beispiel: Programm aus Werkstück

Programmcode
N10 INIT(2,"/_N_WKS_DIR/_N_WELLE1_WPD/_N_ABSPAN1_MPF")

Beispiel: INIT-Befehl mit relativer Pfadangabe

Im Kanal 1 ist das Programm `/_N_MPF_DIR/_N_MAIN_MPF` angewählt

Programmcode	**Kommentar**
N10 INIT(2,"MYPROG")	; Programm /_N_MPF_DIR/_N_MYPROG_MPF in Kanal 2 anwählen

Beispiel: Kanalname und Kanalnummer mit Integer Variable

```
$MC_CHAN_NAME[0]= "CHAN_X" ;Name des 1. Kanals
$MC_CHAN_NAME[1]= "CHAN_Y" ;Name des 2. Kanals
```

Programmcode	Kommentar
START(1, 2)	; Start im 1. und 2. Kanal ausführen

Analog dazu Programmierung mit den Kanalbezeichnern:

Programmcode	Kommentar
START(CHAN_X, CHAN_Y)	; Start im 1. und 2. Kanal ausführen ; Die Bezeichner Kanal_X und Kanal_Y repräsentieren aufgrund des Maschinendatums $MC_CHAN_NAME intern die Kanalnummer 1 und 2. Dementsprechend führen die ebenfalls einen Start im 1. und 2. Kanal aus

Programmierung mit Integer-Variable:

Programmcode	Kommentar
DEF INT chanNo1, chanNo2)	; Kanalnummer definieren
chanNo1=CHAN_X chanNo2=CHAN_Y	
START(chanNo1, chanNo2)	

1.14 Interruptroutine (ASUP)

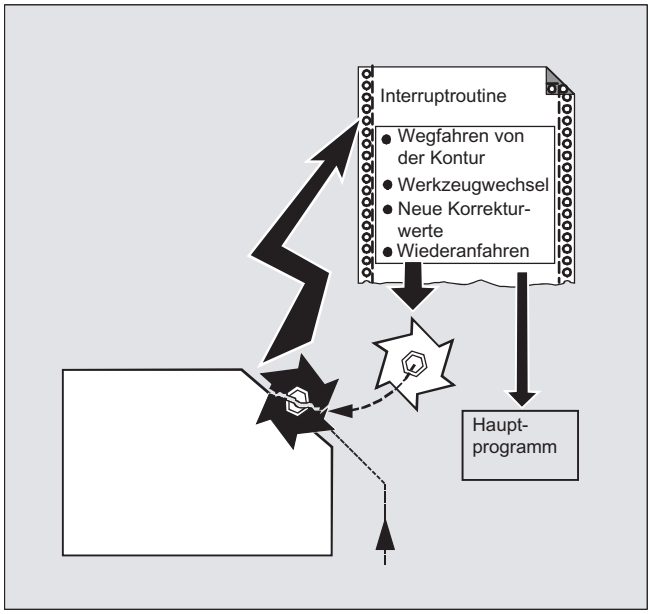
1.14.1 Funktion einer Interruptroutine

Hinweis

Die in der folgenden Beschreibung abwechselnd vorkommenden Begriffe "Asynchrones Unterprogramm (ASUP)" und "Interruptroutine" kennzeichnen die gleiche Funktionalität.

Funktion

Die Funktion einer Interruptroutine soll anhand eines typischen Beispiels verdeutlicht werden:



Während der Bearbeitung bricht das Werkzeug. Hierdurch wird ein Signal ausgelöst, das den laufenden Bearbeitungsablauf stoppt und gleichzeitig ein Unterprogramm – die sogenannte Interruptroutine – startet. In diesem Unterprogramm stehen alle Anweisungen, die in diesem Fall ausgeführt werden sollen.

Ist das Unterprogramm abgearbeitet (und hierdurch die Betriebsbereitschaft hergestellt), springt die Steuerung in das Hauptprogramm zurück und setzt die Bearbeitung – je nach REPOS-Befehl – an der Unterbrechungsstelle fort (siehe " Wiederanfahren an Kontur (Seite 476) ").

⚠ VORSICHT

Wenn im Unterprogramm kein REPOS-Befehl programmiert ist, dann wird auf den Endpunkt des Satzes positioniert, der auf den unterbrochenen Satz folgt.

Literatur

Funktionshandbuch Grundfunktionen; BAG, Kanal, Programmbetrieb, Reset-Verhalten (K1), Kapitel: "Asynchrone Unterprogramme (ASUPs), Interruptroutinen"

1.14.2 Interruptroutine erstellen

Interruptroutine als Unterprogramm erstellen

Die Interruptroutine wird bei der Definition wie ein Unterprogramm gekennzeichnet.

Beispiel:

```
Programmcode        Kommentar
PROC ABHEB_Z        ; Programmname "ABHEB_Z"
N10 ...             ; Danach folgen die NC-Sätze.
...
N50 M17             ; Zum Schluss Programmende und Rückkehr ins Hauptprogramm.
```

Modale G-Funktionen sichern (SAVE)

Die Interruptroutine kann bei der Definition mit SAVE gekennzeichnet werden.

Das Attribut SAVE bewirkt, dass die vor dem Aufruf der Interruptroutine aktiven modalen G-Funktionen gesichert und nach dem Ende der Interruptroutine wieder reaktiviert werden (siehe " Unterprogramme mit SAVE-Mechanismus (SAVE) (Seite 157) ").

Dadurch ist es möglich, die Bearbeitung nach Ablauf der Interruptroutine an der Unterbrechungsstelle fortzusetzen.

Beispiel:

```
Programmcode
PROC ABHEB_Z SAVE
N10 ...
...
N50 M17
```

Weitere Interruptroutinen zuordnen (SETINT)

Innerhalb der Interruptroutine können SETINT-Anweisungen (siehe "Interruptroutine zuordnen und starten (SETINT)" (Seite 111)) programmiert und hierdurch weitere Interruptroutinen scharf geschaltet werden. Das Auslösen erfolgt erst durch den Eingang.

Literatur

Für weitere Informationen zur Erstellung von Unterprogrammen siehe Kapitel "Unterprogrammtechnik, Makrotechnik".

1.14.3 Interruptroutine zuordnen und starten (SETINT, PRIO, BLSYNC)

Funktion

Die Steuerung verfügt über Signale (Eingang 1…8), die eine Unterbrechung des laufenden Programms auslösen und eine entsprechende Interruptroutine starten können.

Die Zuordnung, welcher Eingang welches Programm startet, erfolgt im Teileprogramm mit dem Befehl SETINT.

Falls im Teileprogramm mehrere SETINT-Anweisungen stehen und dadurch mehrere Signale gleichzeitig eintreffen können, müssen den zugeordneten Interruptroutinen Prioritätswerte zugewiesen werden, die die Reihenfolge bei der Abarbeitung festlegen: PRIO=<Wert>

Treffen während der Interruptbearbeitung neue Signale ein, unterbrechen Routinen höherer Priorität die aktuelle Interruptroutine.

Syntax

```
SETINT(<n>) PRIO=<Wert> <NAME>
SETINT(<n>) PRIO=<Wert> <NAME> BLSYNC
SETINT(<n>) PRIO=<Wert> <NAME> LIFTFAST
```

Bedeutung

SETINT(<n>):	Befehl: Eingang <n> einer Interruptroutine zuordnen. Die zugeordnete Interruptroutine startet, wenn Eingang <n> schaltet. **Hinweis:** Wird einem belegten Eingang eine neue Routine zugeordnet, ist die alte Zuordnung automatisch unwirksam.
<n>:	Parameter: Nummer des Eingangs Typ: INT Wertebereich: 1 ... 8
PRIO= :	Befehl: Festlegung der Priorität
<Wert>:	Prioritätswert Typ: INT Wertebereich: 1 ... 128 Priorität 1 entspricht der höchsten Priorität.
<NAME>:	Name des Unterprogramms (Interruptroutine), das abgearbeitet werden soll.
BLSYNC:	Wenn die SETINT-Anweisung zusammen mit BLSYNC programmiert wird, dann wird beim Eintreffen des Interruptsignals der laufende Programmsatz noch abgearbeitet und erst danach die Interruptroutine gestartet.
LIFTFAST:	Wenn die SETINT-Anweisung zusammen mit LIFTFAST programmiert wird, dann wird beim Eintreffen des Interruptsignals vor dem Start der Interruptroutine ein "Schnellabheben des Werkzeugs von der Kontur" durchgeführt (siehe " Schnellabheben von der Kontur (SETINT LIFTFAST, ALF) (Seite 115)").

Beispiele

Beispiel 1: Interruptroutinen zuordnen und Priorität festlegen

Programmcode	Kommentar
...	
N20 SETINT(3) PRIO=1 ABHEB_Z	; Wenn Eingang 3 schaltet, dann soll die Interruptroutine "ABHEB_Z" starten.
N30 SETINT(2) PRIO=2 ABHEB_X	; Wenn Eingang 2 schaltet, dann soll die Interruptroutine "ABHEB_X" starten.
...	

Die Interruptroutinen werden in der Reihenfolge der Prioritätswerte nacheinander abgearbeitet, wenn die Eingänge gleichzeitig anstehen: zuerst "ABHEB_Z", dann "ABHEB_X".

Beispiel 2: Interruptroutine neu zuordnen

Programmcode	Kommentar
...	
N20 SETINT(3) PRIO=2 ABHEB_Z	; Wenn Eingang 3 schaltet, dann soll die Interruptroutine "ABHEB_Z" starten.
…	
N120 SETINT(3) PRIO=1 ABHEB_X	; Eingang 3 wird eine neue Interruptroutine zugeordnet: statt "ABHEB_Z" soll "ABHEB_X" starten, wenn Eingang 3 schaltet.

1.14.4 Zuordnung einer Interruptroutine deaktivieren/reaktivieren (DISABLE, ENABLE)

Funktion

Eine SETINT-Anweisung kann mit DISABLE deaktiviert und mit ENABLE wieder aktiviert werden, ohne dass die Zuordnung Eingang → Interruptroutine verloren geht.

Syntax

```
DISABLE(<n>)
ENABLE(<n>)
```

Bedeutung

DISABLE(<n>):	Befehl: **Deaktivieren** der Interruptroutinen-Zuordnung von Eingang <n>	
ENABLE(<n>):	Befehl: **Reaktivieren** der Interruptroutinen-Zuordnung von Eingang <n>	
<n>:	Parameter: Nummer des Eingangs	
	Typ:	INT
	Wertebereich:	1 ... 8

Beispiel

```
Programmcode                              Kommentar
...
N20 SETINT(3) PRIO=1 ABHEB_Z              ; Wenn Eingang 3 schaltet, dann soll die
                                            Interruptroutine "ABHEB_Z" starten.
...
N90 DISABLE(3)                            ; Die SETINT-Anweisung aus N20 wird
                                            deaktiviert.
...
N130 ENABLE(3)                            ; Die SETINT-Anweisung aus N20 wird wieder
                                            aktiviert.
...
```

1.14.5 Zuordnung einer Interruptroutine löschen (CLRINT)

Funktion

Eine mit SETINT definierte Zuordnung Eingang → Interruptroutine kann mit CLRINT gelöscht werden.

Syntax

```
CLRINT(<n>)
```

Bedeutung

CLRINT(<n>):	Befehl: Löschen der Interruptroutinen-Zuordnung von Eingang <n>	
<n>:	Parameter: Nummer des Eingangs	
	Typ:	INT
	Wertebereich:	1 ... 8

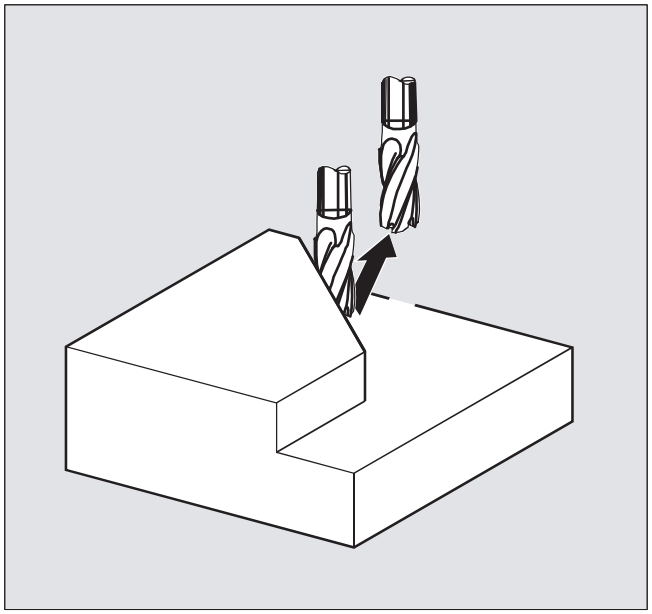
Beispiel

```
Programmcode                          Kommentar
...
N20 SETINT(3) PRIO=2 ABHEB_Z          ;
...
N50 CLRINT(3)                         ; Die Zuordnung zwischen Eingang "3" und der
                                        Interruptroutine "ABHEB_Z" ist gelöscht.
...
```

1.14.6 Schnellabheben von der Kontur (SETINT LIFTFAST, ALF)

Funktion

Bei einer SETINT-Anweisung mit LIFTFAST wird beim Schalten des Eingangs das Werkzeug durch schnelles Abheben von der Werkstückkontur weggefahren.



Der weitere Ablauf ist davon abhängig, ob die SETINT-Anweisung neben LIFTFAST eine Interruptroutine enthält:

Mit Interruptroutine:	**Nach** dem Schnellabheben wird die Interruptroutine ausgeführt.
Ohne Interruptroutine:	Die Bearbeitung wird nach dem Schnellabheben mit Alarm gestoppt.

Syntax

```
SETINT(<n>) PRIO=1 LIFTFAST
SETINT(<n>) PRIO=1 <NAME> LIFTFAST
```

Bedeutung

SETINT(<n>):	Befehl: Eingang <n> einer Interruptroutine zuordnen. Die zugeordnete Interruptroutine startet, wenn Eingang <n> schaltet.	
<n>:	Parameter: Nummer des Eingangs	
	Typ:	INT
	Wertebereich:	1 ... 8

PRIO= :	Festlegung der Priorität
<Wert>:	Prioritätswert
	Wertebereich: 1 ... 128
	Priorität 1 entspricht der höchsten Priorität.
<NAME>:	Name des Unterprogramms (Interruptroutine), das abgearbeitet werden soll.
LIFTFAST:	Befehl: Schnellabheben von der Kontur
ALF=… :	Befehl: Programmierbare Verfahrrichtung (steht im Bewegungssatz)
	Zu den Programmiermöglichkeiten mit ALF siehe Thema " Verfahrrichtung beim Schnellabheben von der Kontur (Seite 117) ".

Randbedingungen

Verhalten bei aktivem Frame mit Spiegelung

Bei der Bestimmung der Abheberichtung wird geprüft, ob ein Frame mit Spiegelung aktiv ist. In diesem Fall werden bei der Abheberichtung bezogen auf die Tangentenrichtung rechts und links vertauscht. Die Richtungsanteile in Werkzeugrichtung werden nicht gespiegelt. Aktiviert wird dieses Verhalten durch die MD-Einstellung:

MD21202 $MC_LIFTFAST_WITH_MIRROR = TRUE

Beispiel

Ein abgebrochenes Werkzeug soll automatisch durch ein Schwesterwerkzeug ersetzt werden. Die Bearbeitung wird dann mit dem neuen Werkzeug fortgesetzt.

Hauptprogramm:

Hauptprogramm	**Kommentar**
N10 SETINT(1) PRIO=1 W_WECHS LIFTFAST	; Wenn Eingang 1 schaltet, wird sofort das Werkzeug mit Schnellabheben (Code Nr. 7 für Werkzeugradiuskorrektur G41) von der Kontur weggefahren. Dann wird die Interruptroutine "W_WECHS" abgearbeitet.
N20 G0 Z100 G17 T1 ALF=7 D1	
N30 G0 X-5 Y-22 Z2 M3 S300	
N40 Z-7	
N50 G41 G1 X16 Y16 F200	
N60 Y35	
N70 X53 Y65	
N90 X71.5 Y16	
N100 X16	
N110 G40 G0 Z100 M30	

Unterprogramm:

```
Unterprogramm                    Kommentar
PROC W_WECHS SAVE                ; Unterprogramm mit Speicherung des
                                   aktuellen Betriebszustandes
N10 G0 Z100 M5                   ; Werkzeugwechselposition, Spindelstopp
N20 T11 M6 D1 G41                ; Werkzeug wechseln
N30 REPOSL RMB M3                ; Kontur wiederanfahren und Rücksprung
                                   ins Hauptprogramm (wird in einem Satz
                                   programmiert)
```

1.14.7 Verfahrrichtung beim Schnellabheben von der Kontur

Rückzugsbewegung

Die Ebene der Rückzugsbewegung wird durch folgende G-Codes bestimmt:

- LFTXT

  Die Ebene der Rückzugsbewegung wird aus der Bahntangente und der Werkzeugrichtung bestimmt (Standardeinstellung).

- LFWP

  Die Ebene der Rückzugsbewegung ist die aktive Arbeitsebene, die mit den G-Codes G17, G18 oder G19 ausgewählt wird. Die Richtung der Rückzugsbewegung ist unabhängig von der Bahntangente. Damit ist ein achsparalleles Schnellabheben programmierbar.

- LFPOS

  Rückzug der mit POLFMASK / POLFMLIN bekannt gemachten Achse auf die mit POLF programmierte absolute Achsposition.

  ALF hat keinen Einfluss auf die Abheberichtung für mehrere Achsen sowie für mehrere Achsen im linearen Zusammenhang.

  Literatur:
  Programmierhandbuch Grundlagen; Kapitel: "Schnellrückzug für Gewindeschneiden"

Programmierbare Verfahrrichtung (ALF=…)

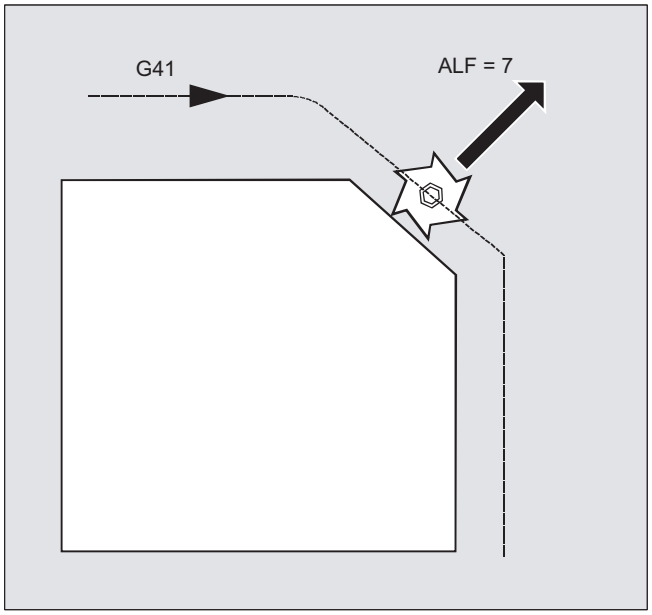
In der Ebene der Rückzugsbewegung wird mit ALF die Richtung in diskreten Schritten von 45 Grad programmiert.

Die möglichen Verfahrrichtungen sind in der Steuerung unter speziellen Code-Nummern gespeichert und unter dieser Nummer abrufbar.

Beispiel:

```
Programmcode
N10 SETINT(2) PRIO=1 ABHEB_Z LIFTFAST
ALF=7
```

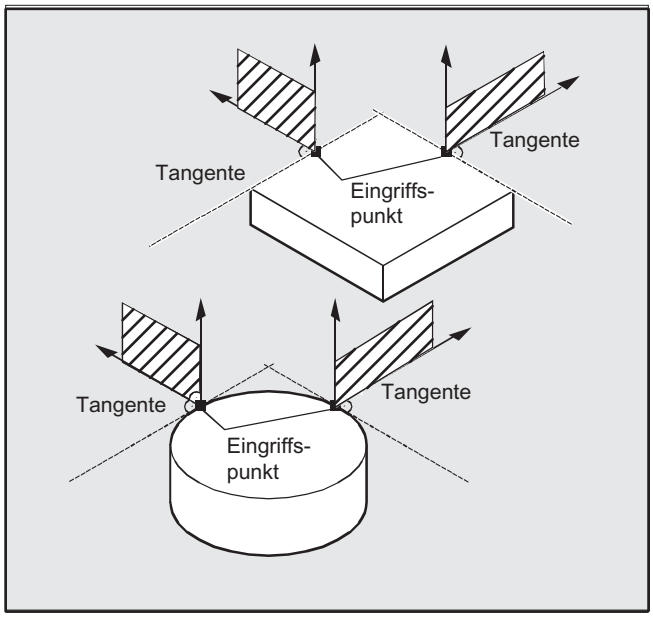
Das Werkzeug fährt bei eingeschaltetem `G41` (Bearbeitungsrichtung links von der Kontur) senkrecht von der Kontur weg.



Bezugsebene für die Beschreibung der Verfahrrichtungen bei LFTXT

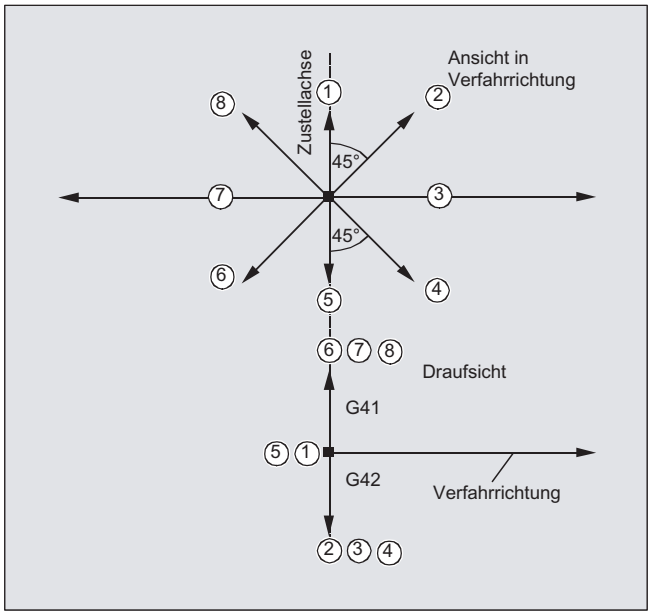
Im Eingriffspunkt des Werkzeugs an der programmierten Kontur wird eine Ebene aufgespannt, die als Bezug für die Angabe der Abhebebewegung mit der entsprechenden Code-Nummer dient.

Die Bezugsebene wird aufgespannt aus der Werkzeuglängsachse (Zustellrichtung) und einem Vektor, der zu dieser und senkrecht zur Tangente im Eingriffspunkt des Werkzeugs an der Kontur steht.



Code-Nummern mit Verfahrrichtungen bei LFTXT

Ausgehend von der Bezugsebene finden Sie in folgender Abbildung die Code-Nummern mit Verfahrrichtungen.



Für `ALF=1` ist der Rückzug in Werkzeugrichtung festgelegt.

Mit `ALF=0` ist die Funktion "Schnellabheben" ausgeschaltet.

⚠ VORSICHT

Bei eingeschalteter Werkzeugradiuskorrektur sollten:

- bei `G41` die Codierungen 2, 3, 4
- bei `G42` die Codierungen 6, 7, 8

nicht verwendet werden, da in diesen Fällen das Werkzeug zur Kontur hinfahren und mit dem Werkstück kollidieren würde.

Code-Nummern mit Verfahrrichtungen bei LFWP

Bei LFWP ergibt sich die Richtung in der Arbeitsebene nach folgender Zuordnung:

- G17: X/Y-Ebene

  ALF=1: Rückzug in X-Richtung

  ALF=3: Rückzug in Y-Richtung

- G18: Z/X-Ebene

  ALF=1: Rückzug in Z-Richtung

  ALF=3: Rückzug in X-Richtung

- G19: Y/Z-Ebene

  ALF=1: Rückzug in Y-Richtung

  ALF=3: Rückzug in Z-Richtung

1.14.8 Bewegungsablauf bei Interruptroutinen

Interruptroutine ohne LIFTFAST

Die Achsbewegungen werden auf der Bahn bis zum Stillstand abgebremst. Anschließend startet die Interruptroutine.

Die Stillstandsposition wird als Unterbrechungsposition abgespeichert und wird bei REPOS mit RMI am Ende der Interruptroutine angefahren.

Interruptroutine mit LIFTFAST

Die Achsbewegungen werden auf der Bahn abgebremst. Gleichzeitig wird die LIFTFAST-Bewegung als überlagerte Bewegung ausgeführt. Wenn die Bahnbewegung und LIFTFAST-Bewegung zum Stillstand gekommen sind, wird die Interruptroutine gestartet.

Als Unterbrechungsposition wird die Position auf der Kontur abgespeichert, bei der die LIFTFAST-Bewegung gestartet und dadurch die Bahn verlassen wurde.

Die Interruptroutine verhält sich mit LIFTFAST und ALF=0 identisch wie die Interruptroutine ohne LIFTFAST.

Hinweis

Der Betrag, um den die Geometrieachsen beim Schnellabheben von der Kontur wegfahren, ist über ein Maschinendatum einstellbar.

1.15 Achstausch, Spindeltausch (RELEASE, GET, GETD)

Funktion

Eine oder mehrere Achsen bzw. Spindeln können immer nur in einem Kanal interpoliert werden. Muss eine Achse wechselweise in zwei verschiedenen Kanälen arbeiten (z. B. Palettenwechsler), so muss sie zunächst im aktuellen Kanal freigegeben und dann in den anderen Kanal übernommen werden. Die Achse wird zwischen den Kanälen getauscht.

Achstauscherweiterungen

Eine Achse/Spindel kann mit Vorlaufstopp und Synchronisation zwischen Vorlauf und Hauptlauf oder alternativ auch ohne Vorlaufstopp getauscht werden. Außerdem ist ein Achstausch auch möglich über

- Achscontainer-Drehung AXCTSWE bzw. AXCTWED mittels impliziten `GET/GETD`.
- Frame mit Rotation, wenn diese Achse hierüber mit anderen Achsen verknüpft ist.
- Synchronaktionen, siehe Bewegungssynchronaktionen, "Achstausch `RELEASE`, `GET`".

Maschinenhersteller

Bitte beachten Sie die Angaben des Maschinenherstellers. Über projektierbare Maschinendaten muss eine Achse für den Achstausch in allen Kanälen eindeutig definiert sein und das Achstauschverhalten ist auch über Maschinedaten veränderbar einstellbar.

Syntax

`RELEASE(Achsname, Achsname, ...)` oder `RELEASE(S1)`

`GET(Achsname, Achsname, ...)` oder `GET(S2)`

`GETD(Achsname, Achsname, ...)` oder `GETD(S3)`

Mit GETD (GET Directly) wird eine Achse direkt aus einem anderen Kanal geholt. Das bedeutet, dass zu diesem GETD kein passendes RELEASE in einem anderen Kanal programmiert sein muss. Es bedeutet aber auch, dass jetzt eine andere Kanalkommunikation aufgebaut werden muss (z. B. Waitmarken).

Bedeutung

`RELEASE (Achsname, Achsname, …)`:	Freigeben der Achse(n)
`GET (Achsname, Achsname, …)`:	Übernehmen der Achse(n)
`GETD (Achsname, Achsname, …)`:	Direktes Übernehmen der Achse(n)
`Achsname`:	Achszuordnung im System: AX1, AX2, … oder Angabe der Maschinenachsnamen
`RELEASE(S1)` :	Freigeben der Spindel S1, S2, …
`GET(S2)` :	Übernehmen der Spindel S1, S2, …
`GETD(S3)` :	Direktes Übernehmen der Spindel S1, S2, …

GET-Anforderung ohne Vorlaufstopp

Wird nach einer GET-Anforderung **ohne** Vorlaufstopp die Achse mit `RELEASE(Achse)` oder `WAITP(Achse)` wieder freigegeben, so führt ein nachfolgender `GET` zu einem `GET` **mit** Vorlaufstopp.

⚠VORSICHT

Eine mit GET übernommene Achse bzw. Spindel bleibt auch nach einem Tasten- oder Programm-RESET diesem Kanal zugeordnet.

Bei neuem Programmstart muss die Zuordnung der getauschten Achsen bzw. Spindeln programmtechnisch erfolgen, falls die Achse in ihrem Grundkanal benötigt wird.

Bei POWER ON wird sie dem im Maschinendatum hinterlegten Kanal zugeordnet.

Beispiele

Beispiel 1: Achstausch zwischen zwei Kanälen

Von 6 Achsen werden in Kanal 1 zur Bearbeitung benutzt: 1., 2., 3. und 4. Achse.
5. und 6. Achse werden in Kanal 2 zum Werkstückwechsel benutzt.

Achse 2 soll zwischen beiden Kanälen getauscht werden können und nach POWER ON dem Kanal 1 zugeordnet sein.

Programm "MAIN" in Kanal 1:

```
Programmcode                    Kommentar
INIT (2,"TAUSCH2")              ; Programm TAUSCH2 im Kanal 2 anwählen.
N… START (2)                    ; Programm in Kanal 2 starten.
N… GET (AX2)                    ; Achse AX2 übernehmen.
...
N… RELEASE (AX2)                ; Achse AX2 freigeben.
N… WAITM (1,1,2)                ; Warten auf WAIT-Marke in Kanal 1 und 2 zur
                                  Synchronisation in den beiden Kanälen.
...                             ; Weiterer Ablauf nach Achstausch.
N… M30
```

Programm "TAUSCH2" in Kanal 2:

```
Programmierung                  Kommentar
N… RELEASE(AX2)
N160 WAITM(1,1,2)               ; Warten auf WAIT-Marke in Kanal 1 und 2 zur
                                  Synchronisation in den beiden Kanälen.
N150 GET(AX2)                   ; Achse AX2 übernehmen.
...                             ; Weiterer Ablauf nach Achstausch.
N… M30
```

Beispiel 2: Achstausch ohne Synchronisierung

Wenn die Achse nicht synchronisiert werden muss, wird durch GET kein Vorlaufstopp erzeugt.

```
Programmierung              Kommentar
N01 G0 X0
N02 RELEASE(AX5)
N03 G64 X10
N04 X20
N05 GET(AX5)                ; Wenn keine Synchronisation nötig, wird dies kein
                              ausführbarer Satz.
N06 G01 F5000               ; Kein ausführbarer Satz.
N07 X20                     ; Kein ausführbarer Satz, da X-Position wie in N04.
N08 X30                     ; Erster ausführbarer Satz nach N05.
...
```

Beispiel 3: Aktivierung eines Achstausches ohne Vorlaufstopp

Voraussetzung: Der Achstausch ohne Vorlaufstopp muss über ein Maschinendatum projektiert werden.

```
Programmierung              Kommentar
N010 M4 S100
N011 G4 F2
N020 M5
N021 SPOS=0
N022 POS[B]=1
N023 WAITP(B)               ; Achse B wird zur neutralen Achse.
N030 X1 F10
N031 X100 F500
N032 X200
N040 M3 S500                ; Achse löst kein Vorlaufstopp/REORG aus.
N041 G4 F2
N050 M5
N099 M30
```

Wird die Spindel bzw. Achse B unmittelbar nach dem Satz N023 als **PLC-Achse** z. B. auf 180 Grad und zurück auf 1 Grad verfahren, dann wird diese Achse wieder zur neutralen Achse und löst im Satz `N40` keinen Vorlaufstopp auf.

Voraussetzung

Voraussetzungen für den Achstausch

- Die Achse muss über Maschinendaten in allen Kanälen definiert sein, die Achse verwenden wollen.
- Über das **achs**spezifische Maschinendatum muss festgelegt sein, welchem Kanal die Achse nach POWER ON zugeordnet werden soll.

Beschreibung

Achse freigeben: RELEASE

Bei der Achsfreigabe ist zu beachten:

1. Die Achse darf an keiner Transformation beteiligt sein.
2. Bei Achskopplungen (Tangentialsteuerung), müssen alle Achsen des Verbands freigegeben werden.
3. Eine konkurrierende Positionierachse kann in diesem Zustand nicht getauscht werden.
4. Bei einer Gantry-Masterachse werden auch alle Folgeachsen getauscht.
5. Bei Achskopplungen (Mitschleppen, Leitwertkopplung, Elektronisches Getriebe) kann nur die Leitachse des Verbandes freigegeben werden.

Achse übernehmen: GET

Mit diesem Befehl wird der eigentliche Achstausch durchgeführt. Die Verantwortung für die Achse liegt vollständig bei dem Kanal, in dem der Befehl programmiert wurde.

Auswirkungen von GET:

Achstausch mit Synchronisierung:

Eine Achse muss immer dann synchronisiert werden, wenn sie zwischenzeitlich in einem anderen Kanal oder der PLC zugeordnet war, und vor dem GET keine Synchronisierung durch "WAITP", G74 oder Restweglöschen stattgefunden hat.

- Ein Vorlaufstopp erfolgt (wie bei STOPRE).
- Die Bearbeitung wird so lange unterbrochen, bis der Tausch vollständig ausgeführt ist.

Automatisches "GET"

Wenn eine Achse prinzipiell im Kanal verfügbar, jedoch derzeitig nicht als "Kanal-Achse" vorhanden ist, wird automatisch ein GET ausgeführt. Falls die Achse(n) schon synchronisiert ist (sind), wird kein Vorlaufstopp erzeugt.

Achstauschverhalten veränderbar einstellen

Der Abgabezeitpunkt von Achsen lässt sich über ein Maschinendatum wie folgt einstellen:

- Automatischer Achstausch findet zwischen zwei Kanälen auch dann statt, wenn die Achse durch WAITP in einen neutralen Zustand gebracht wurde (Verhalten wie bisher)
- Bei der Anforderung einer Achs-Containerdrehung werden alle dem ausführenden Kanal zuordenbaren Achsen des Achs-Containers mittels impliziten GET bzw. GETD in den Kanal geholt. Ein anschließender Achstausch ist erst nach dem Abschluss der Achs-Containerdrehung wieder erlaubt.
- Nach einem eingeschobenen Zwischensatz im Hauptlauf wird geprüft, ob ein Reorganisieren erforderlich ist oder nicht. Nur wenn die Achszustände dieses Satzes mit den aktuellen Achszuständen **nicht** übereinstimmen, ist ein Reorganisieren erforderlich.
- Statt eines GET-Satzes mit Vorlaufstopp und Synchronisation zwischen Vorlauf und Hauptlauf kann ein Achstausch auch ohne Vorlaufstopp erfolgen. Es wird dann nur ein Zwischensatz mit der GET-Anforderung erzeugt. Im Hauptlauf wird bei Abarbeitung dieses Satzes überprüft, ob die Zustände der Achse im Satz mit den aktuellen Achszuständen übereinstimmen.

Weitere Informationen zur Funktionalität eines Achs- oder Spindeltausches siehe /FB2/ Funktionshandbuch Erweiterungsfunktionen; BAGs, Kanäle, Achstausch (K5).

1.16 Achse einem anderen Kanal übergeben (AXTOCHAN)

Funktion

Mit dem Sprachbefehl `AXTOCHAN` kann eine Achse angefordert werden, um diese Achse einem anderen Kanal zu übergeben. Die Achse kann sowohl vom NC-Teileprogramm als auch aus einer Synchronaktion heraus in den entsprechenden Kanal gebracht werden.

Syntax

```
AXTOCHAN(Achsname,Kanalnummer[,Achsname,Kanalnummer[,...]])
```

Bedeutung

`AXTOCHAN`:	Achse für einen bestimmten Kanal anfordern
`Achsname`:	Achszuordnung im System: X, Y, … oder Angabe der beteiligten Maschinenachsnamen. Der auszuführende Kanal muss nicht der eigene Kanal sein und es muss auch nicht der Kanal sein, der aktuell das Interpolationsrecht für die Achse besitzt
`Kanalnummer`:	Nummer des Kanals, dem die Achse zugeordnet werden soll

Hinweis

Konkurrierende Positionierachse und ausschließlich PLC kontrollierte Achse

Eine PLC-Achse kann als konkurrierende Positionierachse den Kanal nicht wechseln. Eine ausschließlich von der PLC kontrollierte Achse kann nicht dem NC-Programm zugeordnet werden.

Literatur
Funktionshandbuch Erweiterungsfunktionen; Positionierachsen (P2)

Beispiel

AXTOCHAN im NC-Programm

Die Achsen X und Y sind im 1. Kanal und im 2. Kanal bekannt. Aktuell hat der Kanal 1 das Interpolationsrecht und im Kanal 1 wird folgendes Programm gestartet:

```
Programmcode                    Kommentar
N110 AXTOCHAN(Y,2)              ; Y-Achse in den 2. Kanal schieben.
N111 M0
N120 AXTOCHAN(Y,1)              ; Y-Achse wieder zurückholen (neutral).
N121 M0
N130 AXTOCHAN(Y,2,X,2)          ; Y-Achse und X-Achse in den 2. Kanal schieben
                                  (Achsen neutral).
```

Programmcode	Kommentar
N131 M0	
N140 AXTOCHAN(Y,2)	; Y-Achse in den 2. Kanal schieben (NC-Programm).
N141 M0	

Weitere Informationen

AXTOCHAN im NC-Programm

Dabei wird nur bei einer Anforderung der Achse für das NC-Programm im eigenen Kanal ein GET durchgeführt und damit auch auf die tatsächliche Zustandsänderung gewartet. Wird die Achse für einen anderen Kanal angefordert oder soll sie zur neutralen Achse im eigenen Kanal werden, dann nur wird die Anforderung entsprechend abgesetzt.

AXTOCHAN aus einer Synchronaktion

Wird eine Achse für den eigenen Kanal angefordert so wird AXTOCHAN aus einer Synchronaktion auf ein GET aus einer Synchronaktion abgebildet. In diesem Fall wird die Achse bei der ersten Anforderung für den eigenen Kanal zur neutralen Achse. Bei der zweiten Anforderung wird die Achse dem NC-Programm analog zur GET-Anforderung im NC-Programm zugeordnet. Zur GET-Anforderung aus einer Synchronaktion siehe Kapitel "Bewegungssynchronaktionen".

1.17 Maschinendaten wirksam setzen (NEWCONF)

Funktion

Mit dem Befehl NEWCONF werden alle Maschinendaten der Wirksamkeitsstufe "NEW_CONFIG" wirksam gesetzt. Die Funktion kann auch in der Bedienoberfläche HMI durch Betätigen des Softkeys "MD wirksam setzen" aktiviert werden.

Bei der Ausführung der Funktion "NEWCONF" erfolgt ein impliziter Vorlaufstopp, d. h. die Bahnbewegung wird unterbrochen.

Syntax

```
NEWCONF
```

Bedeutung

NEWCONF: Befehl zum Wirksamsetzen aller Maschinendaten der Wirksamkeitsstufe "NEW_CONFIG"

NEWCONF aus dem Teileprogramm kanalübergreifend ausführen

Werden axiale Maschinendaten aus dem Teileprogramm heraus verändert und anschließend mit NEWCONF aktiviert, so setzt der Befehl NEWCONF nur die Maschinendaten aktiv, die Änderungen für den Kanal des Teileprogramms bewirken.

Hinweis

Um alle Änderungen sicher wirksam werden zu lassen, muss der Befehl NEWCONF in jedem Kanal ausgeführt werden, in dem auch die von den veränderten Maschinendaten betroffenen Achsen oder Funktionen aktuell gerechnet werden.

Bei NEWCONF werden keine axialen Maschinendaten wirksam gesetzt.

Für PLC-kontrollierte Achsen muss ein axialer RESET ausgeführt werden.

Beispiel

Fräsbearbeitung: Bohrposition mit verschiedenen Technologien bearbeiten

```
Programmcode                          Kommentar
N10 $MA_CONTOUR_TOL[AX]=1.0           ; Maschinendatum ändern.
N20 NEWCONF                           ; Maschinendaten wirksam setzen.
...
```

1.18 Datei schreiben (WRITE)

Funktion

Mit dem WRITE-Befehl können Sätze/Daten aus dem Teileprogramm an das Ende einer angegebenen Datei (Protokolldatei) bzw. des gerade in Abarbeitung befindlichen Teileprogramms geschrieben werden. Die Sätze/Daten werden am Dateiende eingefügt, also nach M30.

Hinweis

Eine per WRITE-Befehl zu beschreibende Datei wird neu angelegt, wenn sie nicht in der NC existiert.

Ablageort ist der statische NC-Speicher. Bei SINUMERIK 840D sl ist dies die CompactFlash Card. Gegenüber SINUMERIK 840D erhöht sich dadurch die Laufzeit des WRITE-Befehls um ca. 75 ms.

Existiert eine Datei gleichen Namens auf der Festplatte, wird diese nach dem Schließen der Datei (in der NC) überschrieben. Abhilfe: Im Bedienbereich "Dienste" über den Softkey "Eigenschaften" Namen in der NC ändern.

Voraussetzung

Die aktuell eingestellte Schutzstufe muss gleich oder größer dem WRITE-Recht der Datei sein. Ist dies nicht der Fall, wird der Zugriff mit Fehlermeldung (Rückgabewert der Fehlervariablen = 13) abgelehnt.

Syntax

```
DEF INT <Fehler>
WRITE(<Fehler>,"<Dateiname>","<Satz/Daten>")
```

Bedeutung

WRITE:	Befehl zum Anfügen eines Satzes bzw. von Daten an das Ende der angegebenen Datei

<Fehler>: Variable für die Rückgabe des Fehlerwerts

Typ. INT

Wert:		
	0	kein Fehler
	1	Pfad nicht erlaubt
	2	Pfad nicht gefunden
	3	Datei nicht gefunden
	4	falscher Dateityp
	10	Datei ist voll
	11	Datei wird benutzt
	12	keine Ressourcen frei
	13	keine Zugriffsrechte
	20	sonstiger Fehler

<Dateiname>: Name der Datei, in der der angegebene Satz bzw. die angegebenen Daten angefügt werden sollen

Typ: STRING

Bei der Angabe des Dateinamens sind folgende Punkte zu beachten:

- Der angegebene Dateiname darf keine Leer- oder Steuerzeichen (Zeichen mit ASCII-Code ≤ 32) enthalten, da sonst der WRITE-Befehl mit Fehlerkennung 1 "Pfad nicht erlaubt" abgebrochen wird.
- Der Dateiname kann mit Pfadangabe und Datei-Kennung angegeben werden:
  - Pfadangaben

    Pfadangaben müssen absolut sein, d. h. sie beginnen mit "/".

    Ohne Pfadangabe wird die Datei im aktuellen Verzeichnis (= Verzeichnis des angewählten Programms) abgelegt.
  - Datei-Kennung

    Enthält der Dateiname keine Domain-Kennung ("_N_"), wird er entsprechend ergänzt.

    Enthält der Dateinamen als viertletztes Zeichen einen Unterstrich "_", so werden die nachfolgenden drei Zeichen als Datei-Kennung interpretiert. Um bei allen Datei-Befehlen denselben Dateinamen verwenden zu können, z. B. über eine Variable vom Typ STRING, dürfen nur die Datei-Kennungen _SPF und _MPF verwendet werden.

    Ist keine Kennung "_MPF" oder "_SPF" angegeben, wird automatisch _MPF ergänzt.
- Die Länge des Dateinamens darf maximal 32 Bytes, die Länge der Pfadangabe maximal 128 Bytes betragen.

Beispiel:

```
"PROTFILE"
"_N_PROTFILE"
"_N_PROTFILE_MPF"
"/_N_MPF_DIR_/_N_PROTFILE_MPF/"
```

<Satz/Daten>: Satz bzw. Daten, die in der angegebenen Datei angefügt werden sollen.

Typ: STRING

Hinweis:
Intern wird noch LF angehängt, d. h. die Zeichenkette wird um 1 Zeichen länger.

Randbedingungen

- **Maximale Dateigröße (→ Maschinenhersteller!)**

  Die maximal mögliche Dateigröße von Protokolldateien wird eingestellt mit dem Maschinendatum:

  MD11420 $MN_LEN_PROTOCOL_FILE

  Die maximale Dateigröße gilt für alle Dateien, die mit dem WRITE-Befehl angelegt werden. Bei Überschreitung wird eine Fehlermeldung ausgegeben und der Satz bzw. die Daten werden nicht abgespeichert. Sofern der Speicher ausreicht, kann eine neue Datei angelegt werden.

Beispiele

Beispiel 1: WRITE-Befehl ohne absolute Pfadangabe

```
Programmcode                                                    Kommentar
N10 DEF INT ERROR                                               ; Definition der
                                                                  Fehlervariablen.
N20 WRITE(ERROR,"TEST1","PROTOKOLL VOM 7.2.97")                 ; Schreibe den Text
                                                                  "PROTOKOLL VOM 7.2.97"
                                                                  in die Datei
                                                                  _N_TEST1_MPF.
N30 IF ERROR                                                    ; Fehlerauswertung.
N40 MSG ("Fehler bei WRITE-Befehl:" << ERROR)
N50 M0
N60 ENDIF
...
```

Beispiel 2: WRITE-Befehl mit absoluter Pfadangabe

```
Programmcode
...
WRITE(ERROR,"/_N_WKS_DIR/_N_PROT_WPD/_N_PROT_MPF","PROTOKOLL VOM 7.2.97")
...
```

1.19 Datei löschen (DELETE)

Funktion

Mit dem DELETE-Befehl können alle Dateien gelöscht werden, egal, ob diese per WRITE-Befehl entstanden sind oder nicht. Auch Dateien, die unter höherer Zugriffsstufe erstellt wurden, können mit DELETE gelöscht werden.

Syntax

```
DEF INT <Fehler>
DELETE(<Fehler>,"<Dateiname>")
```

Bedeutung

DELETE:	Befehl zum Löschen der angegebenen Datei		
<Fehler>:	Variable für die Rückgabe des Fehlerwerts		
	Typ.	INT	
	Wert:	0	kein Fehler
		1	Pfad nicht erlaubt
		2	Pfad nicht gefunden
		3	Datei nicht gefunden
		4	falscher Dateityp
		11	Datei wird benutzt
		12	keine Ressourcen frei
		20	sonstiger Fehler

<Dateiname>: Name der zu löschenden Datei

Typ: STRING

Bei der Angabe des Dateinamens sind folgende Punkte zu beachten:

- Der angegebene Dateiname darf keine Leer- oder Steuerzeichen (Zeichen mit ASCII-Code ≤ 32) enthalten, da sonst der DELETE-Befehl mit Fehlerkennung 1 "Pfad nicht erlaubt" abgebrochen wird.
- Der Dateiname kann mit Pfadangabe und Datei-Kennung angegeben werden:
  - Pfadangaben

    Pfadangaben müssen absolut sein, d. h. sie beginnen mit "/".

    Ohne Pfadangabe wird die Datei im aktuellen Verzeichnis (= Verzeichnis des angewählten Programms) gesucht.
  - Datei-Kennung

    Enthält der Dateiname keine Domain-Kennung ("_N_"), wird er entsprechend ergänzt.

    Enthält der Dateinamen als viertletztes Zeichen einen Unterstrich "_", so werden die nachfolgenden drei Zeichen als Datei-Kennung interpretiert. Um bei allen Datei-Befehlen denselben Dateinamen verwenden zu können, z. B. über eine Variable vom Typ STRING, dürfen nur die Datei-Kennungen _SPF und _MPF verwendet werden.

    Ist keine Kennung "_MPF" oder "_SPF" angegeben, wird automatisch _MPF ergänzt.
- Die Länge des Dateinamens darf maximal 32 Bytes, die Länge der Pfadangabe maximal 128 Bytes betragen.

Beispiel:

```
"PROTFILE"
"_N_PROTFILE"
"_N_PROTFILE_MPF"
"/_N_MPF_DIR_/_N_PROTFILE_MPF/"
```

Beispiel

Programmcode	Kommentar
N10 DEF INT ERROR	; Definition der Fehlervariablen.
N15 STOPRE	; Vorlaufstopp.
N20 DELETE(ERROR,"/_N_SPF_DIR/_N_TEST1_SPF")	; Lösche die Datei TEST1 im Unterprogrammverzeichnis.
N30 IF ERROR	; Fehlerauswertung.
N40 MSG("Fehler bei DELETE-Befehl:" <<ERROR)	
N50 M0	
N60 ENDIF	

1.20 Zeilen in Datei lesen (READ)

Funktion

Der READ-Befehl liest in der angegebenen Datei eine oder mehrere Zeilen und legt die gelesenen Informationen in einem Feld vom Typ STRING ab. Jede gelesene Zeile belegt in diesem Feld ein Feldelement.

Hinweis

Die Datei muss sich im statischen Anwenderspeicher des NCK (Passives Filesystem) befinden.

Voraussetzung

Die aktuell eingestellte Schutzstufe muss gleich oder größer dem READ-Recht der Datei sein. Ist dies nicht der Fall, wird der Zugriff mit Fehlermeldung (Rückgabewert der Fehlervariablen = 13) abgelehnt.

Syntax

```
DEF INT <Fehler>
DEF STRING[<Stringlänge>] <Ergebnis>[<n>,<m>]
READ(<Fehler>,"<Dateiname>",<Anfangszeile>,<Zeilenanzahl>,<Ergebnis>
)
```

Bedeutung

READ:	Befehl zum Lesen von Zeilen der angegebenen Datei und zur Ablage dieser Zeilen in einem Variablenfeld.

<Fehler>: Variable für die Rückgabe des Fehlerwerts (Call-By-Reference-Parameter)

Typ. INT

Wert:
- 0 kein Fehler
- 1 Pfad nicht erlaubt
- 2 Pfad nicht gefunden
- 3 Datei nicht gefunden
- 4 falscher Dateityp
- 13 Zugriffsrechte nicht ausreichend
- 21 Zeile nicht vorhanden (Parameter <Anfangszeile> oder <Zeilenanzahl> größer als Anzahl der Zeilen in der angegebenen Datei).
- 22 Feldlänge der Ergebnisvariablen (<Ergebnis>) ist zu klein.
- 23 Zeilenbereich zu groß (Parameter <Zeilenanzahl> so groß gewählt, dass über das Dateiende hinausgelesen wird).

<Dateiname>: Name der zu lesenden Datei (Call-By-Value-Parameter)

Typ: STRING

Bei der Angabe des Dateinamens sind folgende Punkte zu beachten:

- Der angegebene Dateiname darf keine Leer- oder Steuerzeichen (Zeichen mit ASCII-Code ≤ 32) enthalten, da sonst der READ-Befehl mit Fehlerkennung 1 "Pfad nicht erlaubt" abgebrochen wird.
- Der Dateiname kann mit Pfadangabe und Datei-Kennung angegeben werden:
  - Pfadangaben

    Pfadangaben müssen absolut sein, d. h. sie beginnen mit "/".

    Ohne Pfadangabe wird die Datei im aktuellen Verzeichnis (= Verzeichnis des angewählten Programms) gesucht.
  - Datei-Kennung

    Enthält der Dateiname keine Domain-Kennung ("_N_"), wird sie entsprechend ergänzt.

    Enthält der Dateinamen als viertletztes Zeichen einen Unterstrich "_", so werden die nachfolgenden drei Zeichen als Datei-Kennung interpretiert. Um bei allen Datei-Befehlen denselben Dateinamen verwenden zu können, z. B. über eine Variable vom Typ STRING, dürfen nur die Datei-Kennungen _SPF und _MPF verwendet werden.

    Ist keine Kennung "_MPF" oder "_SPF" angegeben, wird automatisch _MPF ergänzt.
- Die Länge des Dateinamens darf maximal 32 Bytes, die Länge der Pfadangabe maximal 128 Bytes betragen.

Beispiel:

```
"PROTFILE"
"_N_PROTFILE"
"_N_PROTFILE_MPF"
"/_N_MPF_DIR_/_N_PROTFILE_MPF/"
```

<Anfangszeile>: Anfangszeile des zu lesenden Dateibereichs (Call-By-Value-Parameter)

Typ: INT

Wert: 0 Es werden die mit dem Parameter <Zeilenanzahl> angegebene Anzahl an Zeilen vor dem Dateiende gelesen.

1 … n Nummer der ersten zu lesenden Zeile.

<Zeilenanzahl>: Anzahl der zu lesenden Zeilen (Call-By-Value-Parameter)

Typ: INT

<Ergebnis>: Ergebnisvariable (Call-By-Reference-Parameter)

Variablenfeld, in dem der gelesene Text abgelegt wird.

Typ: STRING (max. Länge: 255)

Wenn im Parameter <Zeilenanzahl> weniger Zeilen angegeben sind als die Feldgröße [<n>,<m>] der Ergebnisvariablen beträgt, dann werden die restlichen Feldelemente nicht verändert.

Der Abschluss einer Zeile durch die Steuerzeichen "LF" (Line Feed) oder "CR LF" (Carriage Return Line Feed) wird **nicht** in der Ergebnisvariablen abgelegt.

Gelesene Zeilen werden abgeschnitten, wenn die Zeile länger ist als die definierte Stringlänge. Es erfolgt keine Fehlermeldung.

Hinweis

Binäre Files können nicht eingelesen werden. Es wird der Fehler "falscher Dateityp" (Rückgabewert der Fehlervariablen = 4) ausgegeben. Folgenden Dateitypen sind nicht lesbar: _BIN, _EXE, _OBJ, _LIB, _BOT, _TRC, _ACC, _CYC, _NCK.

Beispiel

Programmcode	Kommentar
N10 DEF INT ERROR	; Definition der Fehlervariablen.
N20 DEF STRING[255] RESULT[5]	; Definition der Ergebnisvariablen.
N30 READ(ERROR,"/_N_CST_DIR/_N_TESTFILE_MPF",1,5,RESULT)	; Dateiname mit Domain-, Dateikennung und Pfadangabe.
N40 IF ERROR <>0	; Fehlerauswertung.
N50 MSG("FEHLER"<<ERROR<<"BEI READ-BEFEHL")	
N60 M0	
N70 ENDIF	
...	

1.21 Vorhandensein einer Datei prüfen (ISFILE)

Funktion

Mit dem ISFILE-Befehl kann geprüft werden, ob eine Datei im statischen Anwenderspeicher des NCK (passives Filesystem) existiert.

Syntax

```
<Ergebnis>=ISFILE("<Dateiname>")
```

Bedeutung

ISFILE: Befehl zur Prüfung, ob die angegebene Datei im passiven Filesystem existiert.

<Dateiname>: Name der Datei, deren Vorhandensein im passiven Filesystem geprüft werden soll.

Typ: STRING

Bei der Angabe des Dateinamens sind folgende Punkte zu beachten:

- Der angegebene Dateiname darf keine Leer- oder Steuerzeichen (Zeichen mit ASCII-Code ≤ 32) enthalten.
- Der Dateiname kann mit Pfadangabe und Datei-Kennung angegeben werden:
  - Pfadangaben

    Pfadangaben müssen absolut sein, d. h. sie beginnen mit "/".

    Ohne Pfadangabe wird die Datei im aktuellen Verzeichnis (= Verzeichnis des angewählten Programms) gesucht.
  - Datei-Kennung

    Enthält der Dateiname keine Domain-Kennung ("_N_"), wird er entsprechend ergänzt.

    Enthält der Dateinamen als viertletztes Zeichen einen Unterstrich "_", so werden die nachfolgenden drei Zeichen als Datei-Kennung interpretiert. Um bei allen Datei-Befehlen denselben Dateinamen verwenden zu können, z. B. über eine Variable vom Typ STRING, dürfen nur die Datei-Kennungen _SPF und _MPF verwendet werden.

    Ist keine Kennung "_MPF" oder "_SPF" angegeben, wird automatisch _MPF ergänzt.
- Die Länge des Dateinamens darf maximal 32 Bytes, die Länge der Pfadangabe maximal 128 Bytes betragen.

Beispiel:

```
"PROTFILE"
"_N_PROTFILE"
"_N_PROTFILE_MPF"
"/_N_MPF_DIR_/_N_PROTFILE_MPF/"
```

<Ergebnis>:	Ergebnisvariable zur Aufnahme des Prüfergebnisses		
	Typ.	BOOL	
	Wert:	TRUE	Datei vorhanden
		FALSE	Datei nicht vorhanden

Beispiel

Programmcode	Kommentar
N10 DEF BOOL RESULT	; Definition der Ergebnisvariablen.
N20 RESULT=ISFILE("TESTFILE")	
N30 IF(RESULT==FALSE)	
N40 MSG("DATEI NICHT VORHANDEN")	
N50 M0	
N60 ENDIF	
...	

oder:

Programmcode	Kommentar
N10 DEF BOOL RESULT	; Definition der Ergebnisvariablen.
N20 RESULT=ISFILE("TESTFILE")	
N30 IF(NOT ISFILE("TESTFILE"))	
N40 MSG("DATEI NICHT VORHANDEN")	
N50 M0	
N60 ENDIF	
...	

1.22 Datei-Informationen auslesen (FILEDATE, FILETIME, FILESIZE, FILESTAT, FILEINFO)

Funktion

Über die Befehle FILEDATE, FILETIME, FILESIZE, FILESTAT und FILEINFO können bestimmte Datei-Informationen wie Datum / Uhrzeit des letzten schreibenden Zugriffs, aktuelle Dateigröße, Datei-Status oder die Summe dieser Informationen ausgelesen werden.

Hinweis

Die Datei muss sich im statischen Anwenderspeicher des NCK (Passives Filesystem) befinden.

Voraussetzung

Die aktuell eingestellte Schutzstufe muss gleich oder größer dem Show-Recht des übergeordneten Verzeichnisses sein. Ist dies nicht der Fall, wird der Zugriff mit Fehlermeldung (Rückgabewert der Fehlervariablen = 13) abgelehnt.

Syntax

```
DEF INT <Fehler>
DEF STRING[<Stringlänge>] <Ergebnis>
FILE....(<Fehler>,"<Dateiname>",<Ergebnis>)
```

Bedeutung

FILEDATE:	Der Befehl FILEDATE liefert das **Datum** des letzten schreibenden Zugriffs auf die angegebene Datei.
FILETIME:	Der Befehl FILETIME liefert die **Uhrzeit** des letzten schreibenden Zugriffs auf die angegebene Datei.
FILESIZE:	Der Befehl FILESIZE liefert die **aktuelle Größe** der angegebenen Datei.
FILESTAT:	Der Befehl FILESTAT liefert für die angegebene Datei den **Status** bezüglich Lese-, Schreib- und Ausführrechte.
FILEINFO:	Der Befehl FILEINFO liefert für die angegebene Datei die **Summe der Datei-Informationen**, die über FILEDATE, FILETIME, FILESIZE und FILESTAT auslesbar sind.

`<Fehler>`: Variable für die Rückgabe des Fehlerwerts (Call-By-Reference-Parameter)

Typ. INT

Wert:		
	0	kein Fehler
	1	Pfad nicht erlaubt
	2	Pfad nicht gefunden
	3	Datei nicht gefunden
	4	falscher Dateityp
	13	Zugriffsrechte nicht ausreichend
	22	Stringlänge der Ergebnisvariablen (`<Ergebnis>`) ist zu klein.

`<Dateiname>`: Name der Datei, von der Datei-Information(en) ausgelesen werden soll(en).

Typ: STRING

Bei der Angabe des Dateinamens sind folgende Punkte zu beachten:

- Der angegebene Dateiname darf keine Leer- oder Steuerzeichen (Zeichen mit ASCII-Code ≤ 32) enthalten, da sonst der `FILE...`-Befehl mit Fehlerkennung 1 "Pfad nicht erlaubt" abgebrochen wird.
- Der Dateiname kann mit Pfadangabe und Datei-Kennung angegeben werden:
  - Pfadangaben

    Pfadangaben müssen absolut sein, d. h. sie beginnen mit "/".

    Ohne Pfadangabe wird die Datei im aktuellen Verzeichnis (= Verzeichnis des angewählten Programms) gesucht.
  - Datei-Kennung

    Enthält der Dateiname keine Domain-Kennung ("_N_"), wird er entsprechend ergänzt.

    Enthält der Dateinamen als viertletztes Zeichen einen Unterstrich "_", so werden die nachfolgenden drei Zeichen als Datei-Kennung interpretiert. Um bei allen Datei-Befehlen denselben Dateinamen verwenden zu können, z. B. über eine Variable vom Typ STRING, dürfen nur die Datei-Kennungen _SPF und _MPF verwendet werden.

    Ist keine Kennung "_MPF" oder "_SPF" angegeben, wird automatisch _MPF ergänzt.
- Die Länge des Dateinamens darf maximal 32 Bytes, die Länge der Pfadangabe maximal 128 Bytes betragen.

Beispiel:

```
"PROTFILE"
"_N_PROTFILE"
"_N_PROTFILE_MPF"
"/_N_MPF_DIR_/_N_PROTFILE_MPF/"
```

`<Ergebnis>`: Ergebnisvariable (Call-By-Reference-Parameter)

Variable, in der die angeforderte Datei-Information abgelegt wird.

Typ: STRING bei: `FILEDATE`
Format: "dd.mm.yy"
⇒ Stringlänge muss **8** sein.
`FILETIME`
Format: " hh:mm.ss "
⇒ Stringlänge muss **8** sein.
`FILESTAT`
Format: "rwxsd"
(r: **r**ead, w: **w**rite, x: e**x**ecute, s: **s**how, d: **d**elete)
⇒ Stringlänge muss **5** sein.
`FILEINFO`
Format: "rwxsd nnnnnnnn dd.mm.yy hh:mm:ss"
⇒ Stringlänge muss **32** sein.

INT bei: `FILESIZE`
Die Dateigröße wird in Byte ausgegeben.

Beispiel

Programmcode	**Kommentar**
`N10 DEF INT ERROR`	`; Definition der Fehlervariablen.`
`N20 STRING[32] RESULT`	`; Definition der Ergebnisvariablen.`
`N30 FILEINFO(ERROR,"/_N_MPF_DIR/_N_TESTFILE_MPF",RESULT)`	`; Dateiname mit Domain-, Dateikennung und Pfadangabe.`
`N40 IF ERROR <>0`	`; Fehlerauswertung`
`N50 MSG("FEHLER"<<ERROR<<"BEI FILEINFO-BEFEHL")`	
`N60 M0`	
`N70 ENDIF`	
`...`	

Das Beispiel könnte in der Ergebnisvariablen RESULT z. B. folgendes Ergebnis liefern:

"77777 12345678 26.05.00 13:51:30"

1.23 Checksummenberechnung über ein Feld (CHECKSUM)

Funktion

Mit dem Befehl `CHECKSUM` kann die Checksumme über ein Feld berechnet werden. Durch den Vergleich dieser Checksumme mit dem Ergebnis einer früheren Checksummenberechnung kann festgestellt werden, ob sich die Daten des Feldes verändert haben.

Anwendung

Prüfung, ob sich beim Abspanen die Eingangskontur geändert hat.

Syntax

```
DEF INT <Fehler>
DEF STRING[<Stringlänge>] <Checksumme>
DEF ... <Feld>[<n>,<m>,<o>]
<Fehler>=CHECKSUM(<Checksumme>,"<Feld>"[,<Anfangsspalte>,<Endspalte>
])
```

Bedeutung

`CHECKSUM`: Befehl zur Berechnung der Checksumme über ein Feld

`<Fehler>`: Variable für die Rückgabe des Fehlerwerts

Typ. INT

Wert:		
	0	kein Fehler
	1	Symbol nicht gefunden
	2	kein Feld
	3	Index 1 zu groß
	4	Index 2 zu groß
	5	ungültiger Datentyp
	10	Überlauf der Checksumme

`<Checksumme>`: Ergebnisvariable zur Aufnahme des Ergebnisses der Checksummenberechnung (Call-By-Reference-Parameter)

Typ: STRING

Erforderliche Stringlänge: 16

Die Checksumme wird als Zeichenkette von 16 Hex-Ziffern dargestellt. Es werden aber keine Formatzeichen mit angegeben.

Beispiel: "A6FC3404E534047C"

<Feld>:	Name des Feldes, über das die Checksumme gebildet werden soll (Call-By-Value-Parameter) Typ: STRING Max. Stringlänge: 32 Zulässige Felder sind 1- bis 3-dimensionale Felder der Typen: BOOL, CHAR, INT, REAL, STRING **Hinweis:** Felder von Maschinendaten sind nicht zulässig.
<Anfangsspalte>:	Nummer der Anfangsspalte des Feldes für die Berechnung der Checksumme (optionaler Parameter)
<Endspalte>:	Nummer der Endspalte des Feldes für die Berechnung der Checksumme (optionaler Parameter)

Hinweis

Die Parameter <Anfangsspalte> und <Endspalte> sind optional. Werden keine Spaltenindizes angegeben, so wird die Checksumme über das komplette Feld gebildet.

Das Ergebnis der Checksumme ist immer eindeutig. Bei Änderungen eines Feldelements ergibt sich auch ein anderer Ergebnisstring.

Beispiel

```
Programmcode                                    Kommentar
N10 DEF INT ERROR                               ; Definition der
                                                  Fehlervariablen.
N20 DEF STRING[16] MY_CHECKSUM                  ; Definition der
                                                  Ergebnisvariablen.
N30 DEF INT MY_VAR[4,4]                         ; Felddefinition.
N40 MY_VAR=...
N50 ERROR=CHECKSUM(MY_CHECKSUM,"MY_VAR",0,2)
...
```

Das Beispiel könnte in der Ergebnisvariablen MY_CHECKSUM z. B. folgendes Ergebnis liefern:

"A6FC3404E534047C"

1.24 Aufrunden (ROUNDUP)

Funktion

Mit der Funktion "ROUNDUP" können Eingabewerte vom Typ REAL (gebrochene Zahlen mit Dezimalpunkt) auf die nächste größere ganze Zahl aufgerundet werden.

Syntax

```
ROUNDUP(<Wert>)
```

Bedeutung

`ROUNDUP`:	Befehl zum Aufrunden eines Eingabewerts
`<Wert>`:	Eingabewert vom Typ REAL

Hinweis

Eingabewerte vom Typ INTEGER (eine ganze Zahl) werden unverändert zurückgeliefert.

Beispiele

Beispiel 1: Verschiedene Eingabewerte und deren Rundungsergebnisse

Beispiel	Rundungsergebnis
`ROUNDUP(3.1)`	4.0
`ROUNDUP(3.6)`	4.0
`ROUNDUP(-3.1)`	-3.0
`ROUNDUP(-3.6)`	-3.0
`ROUNDUP(3.0)`	3.0
`ROUNDUP(3)`	3.0

Beispiel 2: ROUNDUP im NC-Programm

Programmcode

```
N10 X=ROUNDUP(3.5) Y=ROUNDUP(R2+2)
N15 R2=ROUNDUP($AA_IM[Y])
N20 WHEN X=100 DO Y=ROUNDUP($AA_IM[X])
...
```

1.25 Unterprogrammtechnik

1.25.1 Allgemeines

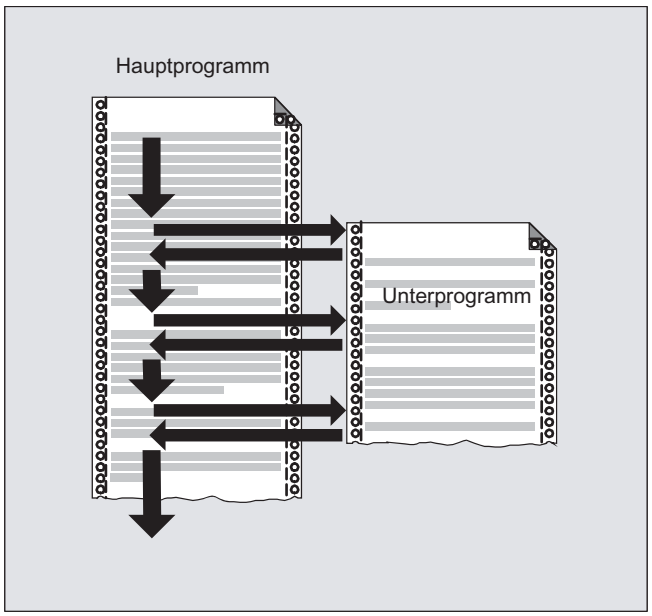
1.25.1.1 Unterprogramm

Funktion

Die Bezeichnung "Unterprogramm" stammt noch aus der Zeit, als Teileprogramme fest in Haupt- und Unterprogramme unterteilt waren. Hauptprogramme waren dabei die Teileprogramme, die an der Steuerung zum Abarbeiten angewählt und dann gestartet wurden. Unterprogramme waren die Teileprogramme, die vom Hauptprogramm aus aufgerufen wurden.

Diese feste Einteilung besteht mit der heutigen SINUMERIK NC-Sprache nicht mehr. Jedes Teileprogramm kann prinzipiell als Hauptprogramm angewählt und gestartet oder als Unterprogramm von einem anderen Teileprogramm aus aufgerufen werden.

Somit wird im weiteren Verlauf mit Unterprogramm ein Teileprogramm bezeichnet, das von einem anderen Teileprogramm aus aufgerufen wird.



Anwendung

Wie in allen höheren Programmiersprachen werden auch in der NC-Sprache Unterprogramme dazu angewandt, um Programmteile, die mehrfach verwendet werden, in eigenständige, in sich abgeschlossene Programme auszulagern.

Unterprogrammen bieten folgende Vorteile:

- Erhöhen die Übersichtlichkeit und Lesbarkeit der Programme
- Erhöhen die Qualität durch Wiederverwendung getesteter Programmteile
- Bieten die Möglichkeit zur Schaffung spezifischer Bearbeitungsbibliotheken
- Sparen Speicherplatz

1.25.1.2 Unterprogrammnamen

Regeln zur Benennung

Bei der Namensgebung von Unterprogrammen sind folgenden Regeln zu beachten:

- Die ersten beiden Zeichen müssen Buchstaben (A - Z, a - z) sein.
- Die folgenden Zeichen können in beliebiger Kombination Buchstaben, Ziffern (0 - 9) und Unterstrich ("_") sein.
- Es dürfen maximal 31 Zeichen verwendet werden.

Hinweis

In der SINUMERIK NC-Sprache wird **nicht** zwischen Groß- und Kleinschreibung unterschieden.

Erweiterungen des Programmnamens

Der bei der Programmerstellung vergebene Programmname wird steuerungsintern mit einem Pre- und Postfix erweitert:

- Prefix: _N_
- Postfix:
  - Hauptprogramme: _MPF
  - Unterprogramme: _SPF

Verwendung des Programmnamens

Bei der Verwendung des Programmnamens, z. B. bei einem Unterprogrammaufruf, sind alle Kombinationen von Prefix, Programmnamen und Postfix möglich.

Beispiel:

Das Unterprogramm mit dem Programmnamen "SUB_PROG" kann über folgende Aufrufe gestartet werden:

1. `SUB_PROG`
2. `_N_SUB_PROG`
3. `SUB_PROG_SPF`
4. `_N_SUB_PROG_SPF`

Hinweis

Namensgleichheit von Haupt- und Unterprogrammen

Existieren Hauptprogramme (.MPF) und Unterprogramme (.SPF) mit gleichem Programmnamen, muss bei Verwendung des Programmnamens im Teileprogramm der jeweilige Postfix angegeben werden, um das Programm eindeutig zu kennzeichnen.

1.25.1.3 Schachtelung von Unterprogrammen

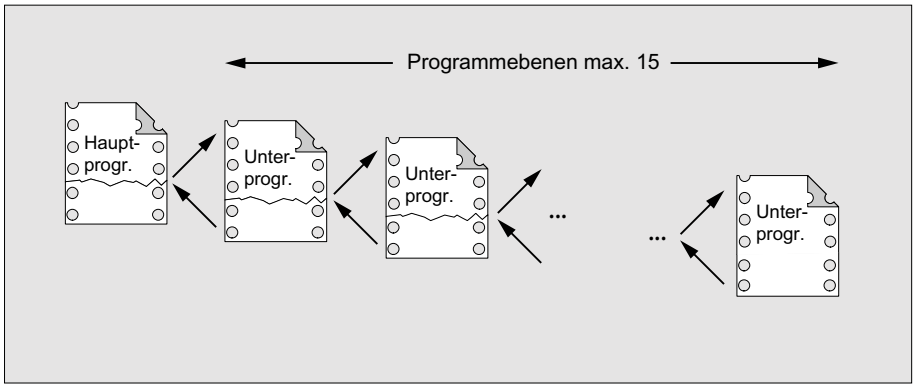
Ein Hauptprogramm kann Unterprogramme aufrufen, die wiederum Unterprogramme aufrufen. Die Abläufe der Programme sind somit ineinander geschachtelt. Jedes Programm läuft dabei in einer eigenen Programmebene.

Schachtelungstiefe

Die NC-Sprache stellt aktuell 16 Programmebenen zur Verfügung. Das Hauptprogramm läuft immer in der obersten Programmebene 0. Ein Unterprogramm läuft immer in der dem Aufruf folgenden nächstniedrigeren Programmebene. Die Programmebene 1 ist somit die erste Unterprogrammebene.

Unterteilung der Programmebenen:

- Programmebene 0: Hauptprogrammebene
- Programmebene 1 - 15: Unterprogrammebene 1 - 15



Interruptroutinen (ASUP)

Wird im Rahmen einer Interruptroutine ein Unterprogramm aufgerufen, wird dieses nicht in der aktuellen im Kanal aktiven Programmebene (n), sondern ebenfalls in der nächstniedrigeren Programmebene (n+1) abgearbeitet. Damit dies auch in der untersten Programmebene noch möglich ist, stehen im Zusammenhang mit Interruptroutinen 2 zusätzliche Programmebenen (16 und 17) zur Verfügung.

Werden mehr als 2 Programmebenen benötigt, muss dies explizit in der Strukturierung des im Kanal abgearbeiteten Teileprogramms berücksichtigt werden. D. h. es darf dann maximal nur so viele Programmebenen beanspruchen, dass noch ausreichend Programmebenen für die Interruptbearbeitung zur Verfügung stehen.

Benötigt die Interruptbearbeitung z. B. 4 Programmebenen, muss das Teileprogramm so strukturiert werden, dass es maximal Programmebene 13 belegt. Erfolgt dann ein Interrupt, stehen diesem die benötigten 4 Programmebenen (14 bis 17) zur Verfügung.

Siemens-Zyklen

Siemens-Zyklen benötigen 3 Programmebenen. Der Aufruf eines Siemens-Zyklus muss daher spätestens erfolgen in:

- Teileprogrammbearbeitung: Programmebene 12
- Interruptroutine: Programmebene 14

1.25.1.4 Suchpfad

Beim Aufruf eines Unterprogramms ohne Pfadangabe sucht die Steuerung in der angegebenen Reihenfolge in folgenden Verzeichnissen:

Reihenfolge	Verzeichnis	Beschreibung
1.	Aktuelles Verzeichnis	Verzeichnis des aufrufenden Programms
2.	/_N_SPF_DIR /	Globales Unterprogrammverzeichnis
3.	/_N_CUS_DIR /	Anwender-Zyklen
4.	/_N_CMA_DIR /	Hersteller-Zyklen
5.	/_N_CST_DIR /	Standard-Zyklen

1.25.1.5 Formal- und Aktualparameter

Von Formal- und Aktualparameter spricht man im Zusammenhang mit der Definition und dem Aufruf von Unterprogrammen mit Parameterübergabe.

Formalparameter

Bei der Definition eines Unterprogramms müssen die dem Unterprogramm zu übergebenden Parameter, die sogenannten Formalparameter, mit Typ und Parameternamen definiert werden.

Die Formalparameter definieren somit die Schnittstelle des Unterprogramms.

Beispiel:

```
Programmcode                          Kommentar
PROC KONTUR (REAL X, REAL Y)          ; Formalparameter: X und Y beide vom Typ REAL
N20 X1=X Y1=Y                         ; Verfahren der Achse X1 auf Position X und der
                                        Achse Y1 auf Position Y
...
N100 RET
```

Aktualparameter

Beim Aufruf eines Unterprogramms müssen dem Unterprogramm absolute Werte oder Variablen, die sogenannten Aktualparameter, übergeben werden.

Die Aktualparameter befüllen somit beim Aufruf die Schnittstelle des Unterprogramms mit aktuellen Werten.

Beispiel:

```
Programmcode                          Kommentar
N10 DEF REAL BREITE                   ; Variablendefinition
N20 BREITE=20.0                       ; Variablenzuweisung
N30 KONTUR(5.5, BREITE)               ; Unterprogrammaufruf mit Aktualparametern: 5.5
                                        und BREITE
...
N100 M30
```

1.25.1.6 Parameterübergabe

Definition eines Unterprogramms mit Parameterübergabe

Die Definition eines Unterprogramms mit Parameterübergabe erfolgt mit dem Schlüsselwort `PROC` und einer vollständigen Auflistung aller vom Unterprogramm erwarteten Parameter.

Unvollständige Parameterübergabe

Beim Aufruf des Unterprogramms müssen nicht immer alle in der Unterprogrammschnittstelle definierten Parameter explizit übergeben werden. Wird ein Parameter weggelassen, wird für diesen Parameter der Standardwert "0" übergeben.

Zur eindeutigen Kennzeichnung der Reihenfolge der Parameter müssen allerdings die Kommas als Trennzeichen der Parameter immer mit angegeben werden. Eine Ausnahme bildet der letzte Parameter. Wird dieser beim Aufruf weggelassen, kann auch das letzte Komma entfallen.

Beispiel:

Unterprogramm:

```
Programmcode                                    Kommentar
PROC SUB_PROG (REAL X, REAL Y, REAL Z)          ;  Formalparameter: X, Y und Z
...
N100 RET
```

Hauptprogramm:

```
Programmcode                        Kommentar
PROC MAIN_PROG
...
N30 SUB_PROG(1.0,2.0,3.0)           ; Unterprogrammaufruf mit vollständiger
                                      Parameterübergabe:
                                      X=1.0, Y=2.0, Z=3.0
…
N100 M30
```

Beispiele für den Unterprogrammaufruf in `N30` mit unvollständiger Parameterübergabe:

```
N30 SUB_PROG( ,2.0,3.0)          ; X=0.0, Y=2.0, Z=3.0
N30 SUB_PROG(1.0, ,3.0)          ; X=1.0, Y=0.0, Z=3.0
N30 SUB_PROG(1.0,2.0)            ; X=1.0, Y=2.0, Z=0.0
N30 SUB_PROG( , ,3.0)            ; X=0.0, Y=0.0, Z=3.0
N30 SUB_PROG( , , )              ; X=0.0, Y=0.0, Z=0.0
```

VORSICHT
Parameterübergabe Call-by-Reference Parameter, die über Call-by-Reference übergeben werden, dürfen beim Unterprogramm-Aufruf nicht weggelassen werden.

VORSICHT
Datentyp AXIS Parameter vom Datentyp AXIS dürfen beim Unterprogramm-Aufruf nicht weggelassen werden.

Überprüfung der Übergabeparameter

Über die Systemvariable $P_SUBPAR [n] mit n = 1, 2, ... kann im Unterprogramm überprüft werden, ob ein Parameter explizit übergeben oder weggelassen wurde. Der Index n bezieht sich auf die Reihenfolge der Formalparameter. Index n = 1 bezieht sich auf den 1. Formalparameter, Index n = 2 auf den 2. Formalparameter usw.

Der folgende Programmausschnitt zeigt beispielhaft für den 1. Formalparameter, wie eine Überprüfung realisiert werden kann:

```
Programmierung                                  Kommentar
PROC SUB_PROG (REAL X, REAL Y, REAL Z)          ; Formalparameter: X, Y und Z
N20 IF $P_SUBPAR[1]==TRUE                       ; Überprüfung des 1.Formalparameters X.
...                                             ; Diese Aktionen werden ausgeführt,
                                                  wenn der Formalparameter X explizit
                                                  übergeben wurde.
N40 ELSE
...                                             ; Diese Aktionen werden ausgeführt,
                                                  wenn der Formalparameter X nicht
                                                  übergeben wurde.
N60 ENDIF
...                                             ; Allgemeine Aktionen
N100 RET
```

1.25.2 Definition eines Unterprogramms

1.25.2.1 Unterprogramm ohne Parameterübergabe

Funktion

Bei der Definition von Unterprogrammen ohne Parameterübergabe kann die Definitionszeile am Programmanfang entfallen.

Syntax

```
[PROC <Programmname>]
...
```

Bedeutung

`PROC`:	Definitionsanweisung am Anfang eines Programms
`<Programmname>`:	Name des Programms

Beispiel

Beispiel 1: Unterprogramm mit `PROC`-Anweisung

Programmcode	**Kommentar**
`PROC SUB_PROG`	`; Definitionszeile`
`N10 G01 G90 G64 F1000`	
`N20 X10 Y20`	
`...`	
`N100 RET`	`; Unterprogrammrücksprung`

Beispiel 2: Unterprogramm ohne `PROC`-Anweisung

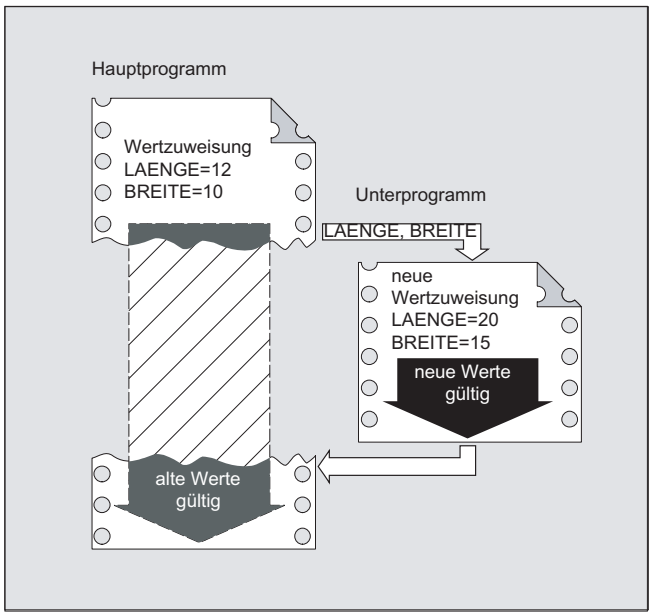
Programmcode	**Kommentar**
`N10 G01 G90 G64 F1000`	
`N20 X10 Y20`	
`...`	
`N100 RET`	`; Unterprogrammrücksprung`

1.25.2.2 Unterprogramm mit Parameterübergabe Call-by-Value (PROC)

Funktion

Die Definition eines Unterprogramms mit Parameterübergabe Call-by-Value erfolgt mit dem Schlüsselwort `PROC`, gefolgt vom Programmnamen und einer vollständigen Auflistung aller vom Unterprogramm erwarteten Parameter mit Typ und Namen. Die Definitionsanweisung muss in der ersten Programmzeile stehen.

Die Parameterübergabe Call-by-Value hat keine Rückwirkungen auf das aufrufende Programm. Das aufrufende Programm übergibt dem Unterprogramm nur die Werte der Aktualparameter.



Hinweis

Es können maximal 127 Parameter übergeben werden.

Syntax

```
PROC <Programmname> (<Parametertyp> <Parametername>, ...)
```

Bedeutung

`PROC`:	Definitionsanweisung am Anfang eines Programms
`<Programmname>`:	Name des Programms
`<Parametertyp>`:	Datentyp des Parameters (z. B. REAL, INT, BOOL)
`<Parametername>`:	Name des Parameters

ACHTUNG
Der nach dem Schlüsselwort PROC angegebene Programmname muss mit dem an der Bedienoberfläche vergebenen Programmnamen übereinstimmen.

Beispiel

Definition eines Unterprogramms mit 2 Parametern vom Typ REAL:

Programmcode	**Kommentar**
`PROC SUB_PROG (REAL LAENGE, REAL BREITE)`	`; Parameter 1: Typ: REAL, Name: LAENGE` `Parameter 2: Typ: REAL, Name: BREITE`
`...`	
`N100 RET`	`; Unterprogrammrücksprung`

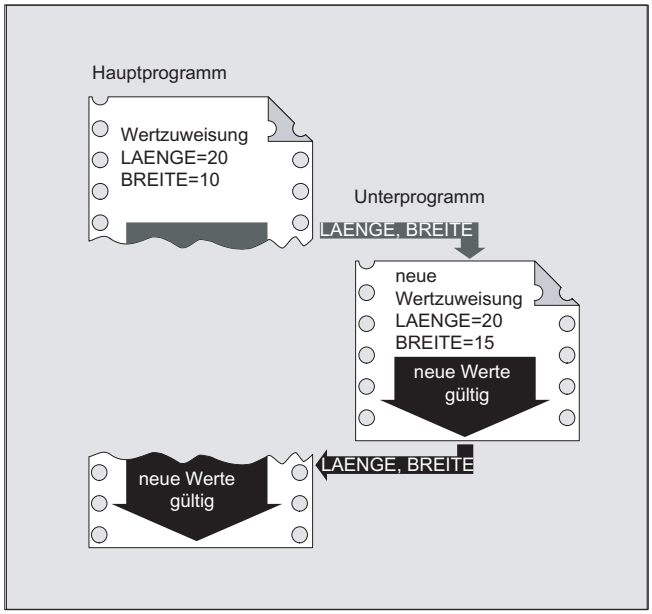
1.25.2.3 Unterprogramm mit Parameterübergabe Call-by-Reference (PROC, VAR)

Funktion

Die Definition eines Unterprogramms mit Parameterübergabe Call-by-Reference erfolgt mit dem Schlüsselwort `PROC`, gefolgt vom Programmnamen und einer vollständigen Auflistung aller vom Unterprogramm erwarteten Parameter mit Schlüsselwort `VAR`, Typ und Namen. Die Definitionsanweisung muss in der ersten Programmzeile stehen.

Bei der Parameterübergabe Call-by-Reference können auch Referenzen auf Felder übergeben werden.

Die Parameterübergabe Call-by-Reference hat Rückwirkungen auf das aufrufende Programm. Das aufrufende Programm übergibt dem Unterprogramm eine Referenz auf den Aktualparameter und ermöglicht dem Unterprogramm somit einen direkten Zugriff auf die entsprechende Variable.



Hinweis

Es können maximal 127 Parameter übergeben werden.

Hinweis

Eine Parameterübergabe Call-by-Reference ist nur dann erforderlich, wenn die übergebene Variable im aufrufenden Programm definiert wurde (LUD). Kanal-globale oder NC-globale Variablen müssen nicht übergeben werden, da auf diese auch direkt vom Unterprogramm aus zugegriffen werden kann.

Syntax

```
PROC <Programmname> (VAR <Parametertyp> <Parametername>, ...)
PROC <Programmname> (VAR <Feldtyp> <Feldname> [<m>,<n>,<o>], ...)
```

Bedeutung

PROC:	Definitionsanweisung am Anfang eines Programms
VAR:	Schlüsselwort für die Parameterübergabe per Referenz
<Programmname>:	Name des Programms
<Parametertyp>:	Datentyp des Parameters (z. B. REAL, INT, BOOL)
<Parametername>:	Name des Parameters
<Feldtyp>:	Datentyp der Feldelemente (z. B. REAL, INT, BOOL)
<Feldname>:	Name des Feldes
[<m>,<n>,<o>]:	Feldgröße Aktuell sind maximal 3-dimensionale Felder möglich: <m>: Feldgröße für 1. Dimension <n>: Feldgröße für 2. Dimension <o>: Feldgröße für 3. Dimension

ACHTUNG

Der nach dem Schlüsselwort PROC angegebene Programmname muss mit dem an der Bedienoberfläche vergebenen Programmnamen übereinstimmen.

Hinweis

Mit Feldern unbestimmter Feldlänge als Formalparameter können Unterprogramme Felder variabler Länge bearbeiten. Dazu wird bei der Definition z. B. eines zweidimensionalen Feldes als Formalparameter die Länge der 1. Dimension nicht angegeben. Das Komma aber muss geschrieben werden.

Beispiel: `PROC <Programmname> (VAR REAL FELD[ ,5])`

Beispiel

Definition eines Unterprogramms mit 2 Parameter als Referenz auf Typ REAL:

Programmcode	**Kommentar**
`PROC SUB_PROG(VAR REAL LAENGE, VAR REAL BREITE)`	`; Parameter 1: Referenz auf Typ: REAL, Name: LAENGE` `Parameter 2: Referenz auf Typ: REAL, Name: BREITE`
`...`	
`N100 RET`	

1.25.2.4 Modale G-Funktionen sichern (SAVE)

Funktion

Das Attribut SAVE bewirkt, dass die vor dem Unterprogrammaufruf aktiven modalen G-Funktionen gesichert und nach dem Unterprogrammende wieder reaktiviert werden.

VORSICHT
Unterbrechung des Bahnsteuerbetriebs Wird bei aktivem Bahnsteuerbetrieb ein Unterprogramme mit Attribut SAVE aufgerufen, wird der Bahnsteuerbetrieb am Ende des Unterprogramms (Rücksprung) unterbrochen.

Syntax

```
PROC <Unterprogrammname> SAVE
```

Bedeutung

SAVE: Sichern der modalen G-Funktionen vor dem Unterprogrammaufruf und Wiederherstellen nach Unterprogrammende

Beispiel

Im Unterprogramm KONTUR wirkt die modale G-Funktion G91 (Kettenmaß). Im Hauptprogramm wirkt die modale G-Funktion G90 (Absolutmaß). Durch die Unterprogrammdefinition mit SAVE wirkt nach dem Unterprogrammende im Hauptprogramm wieder G90.

Unterprogramm-Definition:

```
Programmcode                              Kommentar
PROC KONTUR (REAL WERT1) SAVE             ; Unterprogramm-Definition mit Parameter SAVE
N10 G91 ...                               ; Modale G-Funktion G91: Kettenmaß
N100 M17                                  ; Unterprogrammende
```

Hauptprogramm:

```
Programmcode                              Kommentar
N10 G0 X... Y... G90                      ; Modale G-Funktion G90: Absolutmaß
N20 ...
...
N50 KONTUR (12.4)                         ; Unterprogrammaufruf
N60 X... Y...                             ; Modale G-Funktion G90 durch SAVE reaktiviert
```

Randbedingungen

Frames

Das Verhalten von Frames bezüglich Unterprogrammen mit dem Attribut SAVE ist abhängig vom Typ des Frames und kann über Maschinendaten eingestellt werden.

Literatur

Funktionshandbuch Grundfunktionen; Achsen, Koordinatensysteme, Frames (K2), Kapitel: "Unterprogrammrücksprung mit SAVE"

1.25.2.5 Einzelsatzbearbeitung unterdrücken (SBLOF, SBLON)

Funktion

Einzelsatzunterdrückung für das gesamte Programm

Mit SBLOF gekennzeichnete Programme werden bei aktiver Einzelsatzbearbeitung wie ein Satz komplett abgearbeitet, d. h. für das gesamte Programm wird die Einzelsatzbearbeitung unterdrückt.

SBLOF steht in der PROC-Zeile und gilt bis zum Ende oder Abbruch des Unterprogramms. Mit dem Rücksprung-Befehl wird entschieden, ob am Ende des Unterprogramms angehalten wird oder nicht:

Rücksprung mit M17:	Stopp am Ende des Unterprogramms
Rücksprung mit RET:	Kein Stopp am Ende des Unterprogramms

Einzelsatzunterdrückung innerhalb des Programms

SBLOF muss allein im Satz stehen. Ab diesem Satz wird Einzelsatz ausgeschaltet bis:

- zum nächsten SBLON

  oder

- zum Ende der aktiven Unterprogrammebene

Syntax

Einzelsatzunterdrückung für das gesamte Programm:

```
PROC ... SBLOF
```

Einzelsatzunterdrückung innerhalb des Programms:

```
SBLOF
...
SBLON
```

Bedeutung

PROC:	Erste Anweisung eines Programms
SBLOF:	Befehl zum Ausschalten der Einzelsatzbearbeitung SBLOF kann in einem PROC-Satz oder allein im Satz stehen.
SBLON:	Befehl zum Einschalten der Einzelsatzbearbeitung SBLON muss in einem eigenen Satz stehen.

Randbedingungen

- **Einzelsatzunterdrückung und Satzanzeige**

  Die aktuelle Satzanzeige kann in Zyklen/Unterprogrammen mit DISPLOF unterdrückt werden. Wird DISPLOF zusammen mit SBLOF programmiert, so wird bei Einzelsatz-Stopps innerhalb des Zyklus/Unterprogramms nach wie vor der Aufruf des Zyklus/Unterprogramms angezeigt.

- **Einzelsatzunterdrückung im System-ASUP oder Anwender-ASUP**

  Wenn der Einzelsatz-Stopp im System- oder Anwender-ASUP über die Einstellungen im Maschinendatum MD10702 $MN_IGNORE_SINGLEBLOCK_MASK unterdrückt wird (Bit0 = 1 bzw. Bit1 = 1), dann kann der Einzelsatz-Stopp durch Programmierung von SBLON im ASUP wieder aktiviert werden.

  Wird der Einzelsatz-Stopp im Anwender-ASUP über die Einstellung im Maschinendatum MD20117 $MC_IGNORE_SINGLEBLOCK_ASUP unterdrückt, dann kann der Einzelsatz-Stopp durch Programmierung von SBLON im ASUP **nicht** wieder aktiviert werden.

- **Besonderheiten der Einzelsatzunterdrückung bei den verschiedenen Einzelsatzbearbeitungstypen**

  Bei aktiver Einzelsatzbearbeitung SBL2 (Stopp nach jedem Teileprogrammsatz) wird im SBLON-Satz **nicht** angehalten, wenn im MD10702 $MN_IGNORE_SINGLEBLOCK_MASK (Einzelsatzstopp verhindern) Bit 12 auf "1" gesetzt ist.

  Bei aktiver Einzelsatzbearbeitung SBL3 (Stopp nach jedem Teileprogrammsatz auch im Zyklus) wird der Befehl SBLOF unterdrückt.

Beispiele

Beispiel 1: Einzelsatzunterdrückung innerhalb eines Programms

```
Programmcode                    Kommentar
N10 G1 X100 F1000
N20 SBLOF                       ; Einzelsatz ausschalten
N30 Y20
N40 M100
N50 R10=90
N60 SBLON                       ; Einzelsatz wieder einschalten
N70 M110
N80 ...
```

Der Bereich zwischen N20 und N60 wird im Einzelsatzbetrieb als ein Schritt bearbeitet.

Beispiel 2: Zyklus soll für den Anwender wie ein Befehl wirken

Hauptprogramm:

```
Programmcode
N10 G1 X10 G90 F200
N20 X-4 Y6
N30 CYCLE1
N40 G1 X0
N50 M30
```

Zyklus CYCLE1:

```
Programmcode                                Kommentar
N100 PROC CYCLE1 DISPLOF SBLOF              ; Einzelsatz unterdrücken
N110 R10=3*SIN(R20)+5
N120 IF (R11 <= 0)
N130 SETAL(61000)
N140 ENDIF
N150 G1 G91 Z=R10 F=R11
N160 M17
```

Der Zyklus CYCLE1 wird bei aktiver Einzelsatzbearbeitung abgearbeitet, d. h. es muss für die Bearbeitung von CYCLE1 einmal die Start-Taste gedrückt werden.

Beispiel 3:

Ein von der PLC gestartetes ASUP zum Aktivieren von geänderten Nullpunktverschiebung und Werkzeugkorrekturen soll nicht sichtbar sein.

```
Programmcode
N100 PROC NV SBLOF DISPLOF
N110 CASE $P_UIFRNUM OF     0 GOTOF _G500
                            1 GOTOF _G54
                            2 GOTOF _G55
                            3 GOTOF _G56
                            4 GOTOF _G57
                            DEFAULT GOTOF END
N120 _G54: G54 D=$P_TOOL T=$P_TOOLNO
N130 RET
N140 _G54: G55 D=$P_TOOL T=$P_TOOLNO
N150 RET
N160 _G56: G56 D=$P_TOOL T=$P_TOOLNO
N170 RET
N180 _G57: G57 D=$P_TOOL T=$P_TOOLNO
N190 RET
```

```
Programmcode
N200 END: D=$P_TOOL T=$P_TOOLNO
N210 RET
```

Beispiel 4: Mit MD10702 Bit 12 = 1 wird nicht angehalten

Ausgangssituation:

- Einzelsatzbearbeitung ist aktiv.
- MD10702 $MN_IGNORE_SINGLEBLOCK_MASK Bit12 = 1

Hauptprogramm:

Programmcode	Kommentar
N10 G0 X0	; In dieser Teileprogrammzeile stoppen.
N20 X10	; In dieser Teileprogrammzeile stoppen.
N30 CYCLE	; Vom Zyklus generierter Verfahrsatz.
N50 G90 X20	; In dieser Teileprogrammzeile stoppen.
M30	

Zyklus CYCLE:

Programmcode	Kommentar
PROC CYCLE SBLOF	; Einzelsatz-Stopp unterdrücken
N100 R0 = 1	
N110 SBLON	; Wegen MD10702 Bit12=1 wird in dieser Teileprogrammzeile nicht gestoppt.
N120 X1	; In dieser Teileprogrammzeile wird gestoppt.
N140 SBLOF	
N150 R0 = 2	
RET	

Beispiel 5: Einzelsatzunterdrückung bei Programmschachtelung

Ausgangssituation:

Einzelsatzbearbeitung ist aktiv.

Programmverschachtelung:

```
Programmcode                                  Kommentar
N10 X0 F1000                                  ; In diesem Satz wird gestoppt.
N20 UP1(0)
      PROC UP1(INT _NR) SBLOF                 ; Einzelsatz-Stopp unterdrücken.
      N100 X10
      N110 UP2(0)
            PROC UP2(INT _NR)
            N200 X20
            N210 SBLON                        ; Einzelsatz-Stopp einschalten.
            N220 X22                          ; In diesem Satz wird gestoppt.
            N230 UP3(0)
                  PROC UP3(INT _NR)
                  N300 SBLOF                  ; Einzelsatz-Stopp unterdrücken.
                  N305 X30
                  N310 SBLON                  ; Einzelsatz-Stopp einschalten.
                  N320 X32                    ; In diesem Satz wird gestoppt.
                  N330 SBLOF                  ; Einzelsatz-Stopp unterdrücken.
                  N340 X34
                  N350 M17                    ; SBLOF ist aktiv.
            N240 X24                          ; In diesem Satz wird gestoppt. SBLON ist
                                                aktiv.
            N250 M17                          ; In diesem Satz wird gestoppt. SBLON ist
                                                aktiv.
      N120 X12
      N130 M17                                ; In diesem Rücksprungssatz wird
                                                gestoppt. SBLOF der PROC-Anweisung ist
                                                aktiv.
N30 X0                                        ; In diesem Satz wird gestoppt.
N40 M30                                       ; In diesem Satz wird gestoppt.
```

Weitere Informationen

Einzelsatzsperre für asynchrone Unterprogramme

Um ein ASUP im Einzelsatz in einem Schritt abzuarbeiten, muss im ASUP eine PROC-Anweisung mit SBLOF programmiert werden. Dies gilt auch für die Funktion "Editierbares System-ASUP" (MD11610 $MN_ASUP_EDITABLE).

Beispiel für ein editierbares System-ASUP:

```
Programmcode                                Kommentar
N10 PROC ASUP1 SBLOF DISPLOF
N20 IF $AC_ASUP=='H200'
N30 RET                                     ; Kein REPOS bei BA-Wechsel.
N40 ELSE
N50 REPOSA                                  ; REPOS in allen übrigen Fällen.
N60 ENDIF
```

Programmbeeinflussungen im Einzelsatz

In der Einzelsatzbearbeitung kann der Anwender das Teileprogramm satzweise abarbeiten. Es existieren folgende Einstellungsarten:

- SBL1: IPO-Einzelsatz mit Stopp nach jedem Maschinenfunktionssatz.
- SBL2: Einzelsatz mit Stopp nach jedem Satz.
- SBL3: Halt im Zyklus (durch die Anwahl von SBL3 wird der SBLOF-Befehl unterdrückt).

Einzelsatzunterdrückung bei Programmschachtelung

Wurde in einem Unterprogramm SBLOF in der PROC-Anweisung programmiert, so wird auf den Unterprogrammrücksprung mit M17 angehalten. Damit wird verhindert, dass im aufrufenden Programm bereits der nächste Satz ausgeführt wird. Wird in einem Unterprogramm mit SBLOF, ohne SBLOF in der PROC-Anweisung, eine Einzelsatzunterdrückung aktiviert, wird erst nach dem nächsten Maschinenfunktionssatz des aufrufenden Programms angehalten. Ist dies nicht erwünscht, muss im Unterprogramm noch vor dem Rücksprung (M17) wieder SBLON programmiert werden. Bei einem Rücksprung mit RET in ein übergeordnetes Programm wird nicht angehalten.

1.25.2.6 Aktuelle Satzanzeige unterdrücken (DISPLOF, DISPLON, ACTBLOCNO)

Funktion

In der Satzanzeige wird standardmäßig der aktuelle Programmsatz angezeigt. In Zyklen bzw. Unterprogrammen kann die Anzeige des aktuellen Satzes mit dem Befehl DISPLOF unterdrückt werden. Anstelle des aktuellen Satzes wird dann der Aufruf des Zyklus bzw. Unterprogramms angezeigt. Mit dem Befehl DISPLON kann die Unterdrückung der Satzanzeige wieder aufgehoben werden.

DISPLOF bzw. DISPLON wird in der Programmzeile mit der PROC-Anweisung programmiert und wirkt für das gesamte Unterprogramm und implizit für alle von diesem Unterprogramm aufgerufenen Unterprogramme, die keinen DISPLON- bzw. DISPLOF-Befehl enthalten. Dieses Verhalten gilt auch für ASUPs.

Syntax

```
PROC … DISPLOF
PROC … DISPLOF ACTBLOCNO
PROC … DISPLON
```

Bedeutung

DISPLOF: Befehl zum Unterdrücken der aktuellen Satzanzeige.

Platzierung: Am Ende der Programmzeile mit der PROC-Anweisung

Wirksamkeit: Bis zum Rücksprung aus dem Unterprogramm oder Programmende.

Hinweis:
Wenn aus dem Unterprogramm mit dem DISPLOF-Befehl weitere Unterprogramme aufgerufen werden, dann wird auch in diesen Unterprogrammen die aktuelle Satzanzeige unterdrückt, sofern in diesen nicht explizit DISPLON programmiert ist.

DISPLON: Befehl zum Aufheben der Unterdrückung der aktuellen Satzanzeige

Platzierung: Am Ende der Programmzeile mit der PROC-Anweisung

Wirksamkeit: Bis zum Rücksprung aus dem Unterprogramm oder Programmende.

Hinweis:
Wenn aus dem Unterprogramm mit dem DISPLON-Befehl weitere Unterprogramme aufgerufen werden, dann wird auch in diesen Unterprogrammen der aktuelle Programmsatz angezeigt, sofern in diesen nicht explizit DISPLOF programmiert ist.

ACTBLOCNO: DISPLOF zusammen mit dem Attribut ACTBLOCNO bewirkt, dass im Falle eines Alarms die Nummer des aktuellen Satzes ausgegeben wird, in dem der Alarm aufgetreten ist. Dies gilt auch dann, wenn in einer niedrigeren Programmebene nur DISPLOF programmiert ist.

Bei DISPLOF ohne ACTBLOCNO wird dagegen die Satznummer des Zyklen- bzw. Unterprogrammaufrufs aus der letzten nicht mit DISPLOF gekennzeichneten Programmebene angezeigt.

Beispiele

Beispiel 1: Aktuelle Satzanzeige im Zyklus unterdrücken

Programmcode	Kommentar
`PROC CYCLE(AXIS TOMOV, REAL POSITION) SAVE DISPLOF`	`; Aktuelle Satzanzeige unterdrücken. Stattdessen soll der Zyklus-Aufruf angezeigt werden, z. B.: CYCLE(X,100.0)`
`DEF REAL DIFF`	`; Zyklen-Inhalt`
`G01 ...`	
`...`	
`RET`	`; Unterprogramm-Rücksprung. In der Satzanzeige wird der auf den Zyklus-Aufruf folgende Satz angezeigt.`

Beispiel 2: Satzanzeige bei der Alarmausgabe

Unterprogramm SUBPROG1 (mit ACTBLOCNO):

Programmcode	Kommentar
`PROC SUBPROG1 DISPLOF ACTBLOCNO`	
`N8000 R10 = R33 + R44`	
`...`	
`N9040 R10 = 66 X100`	`; Alarm 12080 auslösen`
`...`	
`N10000 M17`	

Unterprogramm SUBPROG2 (ohne ACTBLOCNO):

Programmcode	Kommentar
`PROC SUBPROG2 DISPLOF`	
`N5000 R10 = R33 + R44`	
`...`	
`N6040 R10 = 66 X100`	`; Alarm 12080 auslösen`
`...`	
`N7000 M17`	

Hauptprogramm:

```
Programmcode                    Kommentar
N1000 G0 X0 Y0 Z0
N1010 ...
...
N2050 SUBPROG1                  ; Alarmausgabe = "12080 Kanal K1 Satz N9040
                                  Syntaxfehler bei Text R10="
N2060 ...
N2350 SUBPROG2                  ; Alarmausgabe = "12080 Kanal K1 Satz N2350
                                  Syntaxfehler bei Text R10="
...
N3000 M30
```

Beispiel 3: Unterdrückung der aktuellen Satzanzeige aufheben

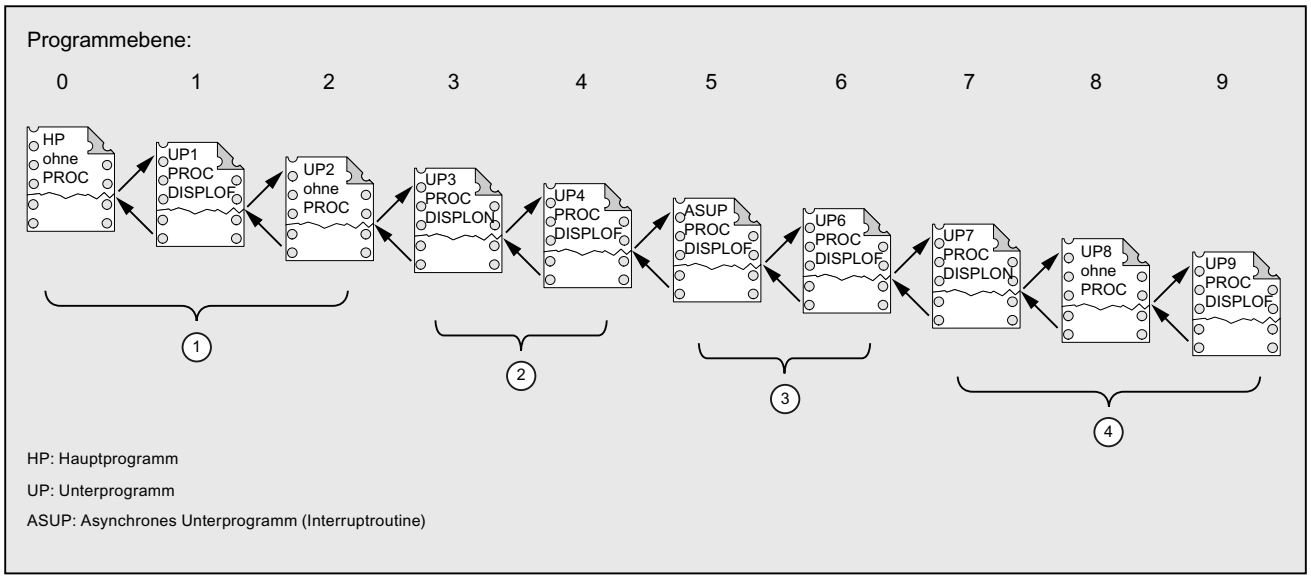
Unterprogramm SUB1 mit Unterdrückung:

```
Programmcode                    Kommentar
PROC SUB1 DISPLOF               ; Aktuelle Satzanzeige im Unterprogramm SUB1
                                  unterdrücken. Stattdessen soll der Satz mit dem
                                  SUB1-Aufruf angezeigt werden.
...
N300 SUB2                       ; Unterprogramm SUB2 aufrufen.
...
N500 M17
```

Unterprogramm SUB2 ohne Unterdrückung:

```
Programmcode                    Kommentar
PROC SUB2 DISPLON               ; Unterdrückung der aktuellen Satzanzeige im
                                  Unterprogramm SUB2 aufheben.
...
N200 M17                        ; Rücksprung ins Unterprogramm SUB1. In SUB1 wird die
                                  aktuelle Satzanzeige wieder unterdrückt.
```

Beispiel 4: Anzeigeverhalten bei unterschiedlichen DISPLON/DISPLOF-Kombinationen



① In der aktuellen Satzanzeige werden die Teileprogrammzeilen aus Programmebene 0 angezeigt.
② In der aktuellen Satzanzeige werden die Teileprogrammzeilen aus Programmebene 3 angezeigt.
③ In der aktuellen Satzanzeige werden die Teileprogrammzeilen aus Programmebene 3 angezeigt.
④ In der aktuellen Satzanzeige werden die Teileprogrammzeilen aus Programmebene 7/8 angezeigt.

1.25.2.7 Unterprogramme mit Vorbereitung kennzeichnen (PREPRO)

Funktion

Mit dem Schlüsselwort PREPRO können im Hochlauf am Ende der PROC-Anweisungszeile alle Dateien gekennzeichnet werden.

Hinweis

Diese Art der Programmvorbereitung ist vom entsprechend eingestellten Maschinendatum abhängig. Bitte Angaben des Maschinenherstellers beachten.

Literatur:
Funktionshandbuch Sonderfunktionen; Vorverarbeitung (V2)

Syntax

```
PROC … PREPRO
```

Bedeutung

PREPRO: Schlüsselwort für Kennzeichnung aller im Hochlauf vorbereiteten Dateien, der in Zyklenverzeichnissen abgelegten NC-Programme

Unterprogramme mit Vorbereitung einlesen und Unterprogrammaufruf

Sowohl im Hochlauf vorbereiteter Unterprogramme mit Parametern als auch beim Unterprogrammaufruf werden die Zyklenverzeichnissen in der gleichen Reihenfolge behandelt:

1. _N_CUS_DIR Anwenderzyklen
2. _N_CMA_DIR Herstellerzyklen
3. _N_CST_DIR Standardzyklen

Im Falle gleichnamiger NC-Programme mit unterschiedlicher Ausprägung wird die zuerst gefundene PROC-Anweisung aktiviert und die andere PROC-Anweisung wird ohne Alarmmeldung überlesen.

1.25.2.8 Unterprogrammrücksprung M17

Funktion

Am Ende eines Unterprogramms steht der Rücksprung-Befehl M17 (bzw. der Teileprogrammende-Befehl M30). Er bewirkt den Rücksprung in das aufrufende Programm auf den Teileprogrammsatz nach dem Unterprogrammaufruf.

Hinweis

M17 und M30 werden in der NC-Sprache gleichwertig behandelt.

Syntax

```
PROC <Programmname>
...
M17/M30
```

Randbedingungen

Auswirkung des Unterprogrammrücksprungs auf den Bahnsteuerbetrieb

Steht M17 (bzw. M30) alleine im Teilprogrammsatz, wird dadurch ein im Kanal aktiver Bahnsteuerbetrieb unterbrochen.

Um zu vermeiden, dass der Bahnsteuerbetrieb unterbrochen wird, ist M17 (bzw. M30) mit in den letzten Verfahrsatz zu schreiben. Zusätzlich muss folgendes Maschinendatum auf "0" gesetzt sein:

MD20800 $MC_SPF_END_TO_VDI = 0 (keine M30/M17-Ausgabe an die NC/PLC-Nahtstelle)

Beispiel

1. Unterprogramm mit M17 im eigenen Satz

```
Programmcode                Kommentar
N10 G64 F2000 G91 X10 Y10
N20 X10 Z10
N30 M17                     ; Rücksprung mit Unterbrechung des Bahnsteuerbetriebs.
```

2. Unterprogramm mit M17 im letzten Verfahrsatz

```
Programmcode                Kommentar
N10 G64 F2000 G91 X10 Y10
N20 X10 Z10 M17             ; Rücksprung ohne Unterbrechung des Bahnsteuerbetriebs.
```

1.25.2.9 Unterprogrammrücksprung RET

Funktion

Als Ersatz für den Rücksprungsbefehl M17 kann im Unterprogramm auch der Befehl RET verwendet werden. RET muss in einem eigenen Teileprogrammsatz programmiert werden. Wie M17 bewirkt RET den Rücksprung in das aufrufende Programm auf den Teileprogrammsatz nach dem Unterprogrammaufruf.

Hinweis

Durch die Programmierung von Parametern kann das Rücksprungverhalten von RET geändert werden (siehe "Parametrierbarer Unterprogrammrücksprung (RET ...) (Seite 171)").

Anwendung

Die RET-Anweisung ist dann zu benutzen, wenn ein G64-Bahnsteuerbetrieb (G641 ... G645) durch den Rücksprung nicht unterbrochen werden soll.

Voraussetzung

Der Befehl RET kann nur in Unterprogrammen verwendet werden, die nicht mit dem Attribut SAVE definiert wurden.

Syntax

```
PROC <Programmname>
...
RET
```

Beispiel

Hauptprogramm:

Programmcode	**Kommentar**
PROC MAIN_PROGRAM	; Programmanfang
...	
N50 SUB_PROG	; Unterprogrammaufruf: SUB_PROG
N60 ...	
...	
N100 M30	; Programmende

Unterprogramm:

```
Programmcode                    Kommentar
PROC SUB_PROG
...
N100 RET                        ; Rücksprung erfolgt auf Satz N60 im Hauptprogramm.
```

1.25.2.10 Parametrierbarer Unterprogrammrücksprung (RET ...)

Funktion

Im Allgemeinen wird aus einem Unterprogramm mit einem Unterprogrammende RET oder M17 in das Programm zurückgesprungen, aus dem das Unterprogramm aufgerufen wurde, und die Bearbeitung wird mit der auf den Unterprogrammaufruf folgenden Programmzeile fortgesetzt.

Daneben gibt es jedoch auch Anwendungsfälle, wo die Programmbearbeitung an einer anderen Stelle fortgesetzt werden soll, z. B.:

- Fortsetzung der Programmbearbeitung nach Aufruf der Abspanzyklen im ISO-Dialekt-Modus (nach der Konturbeschreibung).
- Rücksprung ins Hauptprogramm aus einer beliebigen Unterprogrammebene (auch nach ASUP) beim Fehlerhandling.
- Rücksprung über mehrere Programmebenen für spezielle Anwendungen in Compilezyklen und im ISO-Dialekt-Modus.

In solchen Fällen wird der Befehl RET zusammen mit "Rücksprungsparametern" programmiert.

Syntax

```
RET("<Zielsatz>")
RET("<Zielsatz>",<Satz nach Zielsatz>)
RET("<Zielsatz>",<Satz nach Zielsatz>,<Anzahl der Rücksprungebenen>)
RET("<Zielsatz>", ,<Anzahl der Rücksprungebenen>)
RET("<Zielsatz>",<Satz nach Zielsatz>,<Anzahl der Rücksprungebenen>,
<Rücksprung auf Programmanfang>)
RET( ,
,<Anzahl der Rücksprungebenen>,<Rücksprung auf Programmanfang>)
```

Bedeutung

`RET`: Unterprogrammende (Verwendung statt `M17`)

`<Zielsatz>`: Rücksprungsparameter 1

Nennt als Sprungziel den Satz, an dem die Programmbearbeitung fortgesetzt werden soll.

Wenn Rücksprungsparameter 3 nicht programmiert ist, dann befindet sich das Sprungziel in dem Programm, aus dem das aktuelle Unterprogramm aufgerufen wurde.

Mögliche Angaben sind:

"<Satznummer>"	Nummer des Zielsatzes
"<Sprungmarke>"	Sprungmarke, die im Zielsatz gesetzt sein muss.
"<Zeichenkette>"	Zeichenkette, die im Programm bekannt sein muss (z. B. Programm- oder Variablenname). Für die Programmierung der Zeichenkette im Zielsatz gelten folgende Regeln: • **Leerzeichen am Ende** (im Unterschied zur Sprungmarke, die durch einen ":" am Ende gekennzeichnet ist). • **Vor der Zeichenkette** dürfen nur eine Satznummer und/oder eine Sprungmarke gesetzt sein, **keine Programmbefehle**.

`<Satz nach Zielsatz>`: Rücksprungsparameter 2

Bezieht sich auf den Rücksprungsparameter 1.

Typ: INT

Wert:	
0	Rücksprung erfolgt auf den Satz, der mit dem Rücksprungsparameter 1 angegeben wurde.
> 0	Rücksprung erfolgt auf den Satz, der auf den mit dem Rücksprungsparameter 1 angegeben Satz folgt.

`<Anzahl der Rücksprungebenen>`:

Rücksprungsparameter 3

Nennt die Anzahl an Ebenen, die zurückgesprungen werden soll, um zu der Programmebene zu gelangen, in der die Programmbearbeitung fortgesetzt werden soll.

Typ: INT

Wert:
- 1: Das Programm wird in der "aktuellen Programmebene - 1" fortgesetzt (wie `RET` ohne Parameter).
- 2: Das Programm wird in der "aktuellen Programmebene - 2" fortgesetzt, d. h. es wird eine Ebene übersprungen.
- 3: Das Programm wird in der "aktuellen Programmebene - 3" fortgesetzt, d. h. es werden zwei Ebenen übersprungen.
- ...

Werte-bereich: 1 ... 15

`<Rücksprung auf Programmanfang>`:

Rücksprungsparameter 4

Typ: BOOL

Wert:
- 1: Wenn der Rücksprung ins Hauptprogramm erfolgt und dort ein **ISO-Dialekt-Modus** aktiv ist, wird auf den Programmanfang verzweigt.

Hinweis

Bei einem Unterprogrammrücksprung mit einer Zeichenkette als Angabe für die Zielsatzsuche wird im aufrufenden Programm immer zuerst nach einer Sprungmarke gesucht.

Wenn ein Sprungziel durch eine Zeichenkette eindeutig definiert sein soll, darf die Zeichenkette daher nicht mit dem Namen einer Sprungmarke übereinstimmen, da sonst der Unterprogrammrücksprung immer auf die Sprungmarke und nicht auf die Zeichenkette ausgeführt wird (siehe Beispiel 2).

Randbedingungen

Beim Rücksprung über mehrere Programmebenen werden die SAVE-Anweisungen der einzelnen Programmebenen ausgewertet.

Ist bei einem Rücksprung über mehrere Programmebenen ein modales Unterprogramm aktiv und ist in einem der übersprungenen Unterprogramme der Abwahlbefehl MCALL für das modale Unterprogramm programmiert, bleibt das modale Unterprogramm weiterhin aktiv.

⚠ VORSICHT

Der Programmierer muss darauf achten, dass beim Rücksprung über mehrere Programmebenen mit den richtigen modalen Einstellungen fortgesetzt wird. Dies wird z. B. durch Programmierung eines entsprechenden Hauptsatzes erreicht.

Beispiele

Beispiel 1: Wiederaufsetzen im Hauptprogramm nach ASUP-Bearbeitung

```
Programmierung                              Kommentar
N10010 CALL "UP1"                           ; Programmebene 0 (Hauptprogramm)
  N11000 PROC UP1                           ; Programmebene 1
  N11010 CALL "UP2"
    N12000 PROC UP2                         ; Programmebene 2
    ...
      N19000 PROC ASUP                      ; Programmebene 3 (ASUP-Bearbeitung)
      ...
      N19100 RET("N10900", ,$P_STACK)       ; Unterprogrammrücksprung
N10900                                      ; Wiederaufsetzen im Hauptprogramm.
N10910 MCALL                                ; Modales Unterprogramm ausschalten.
N10920 G0 G60 G40 M5                        ; Weitere modale Einstellungen
                                              korrigieren.
```

Beispiel 2: Zeichenkette (<String>) als Angabe für die Zielsatzsuche

Hauptprogramm:

```
Programmcode                                Kommentar
PROC MAIN_PROGRAM
N1000 DEF INT iVar1=1, iVar2=4
N1010 ...
N1200 subProg1                              ; Aufruf von Unterprogramm "subProg1"
N1210 M2 S1000 X10 F1000
N1220 ......
N1400 subProg2                              ; Aufruf von Unterprogramm "subProg2"
N1410 M3 S500 Y20
N1420 ..
```

```
Programmcode                          Kommentar
N1500 lab1: iVar1=R10*44
N1510 F500 X5
N1520 ...
N1550 subprog1: G1 X30                ; "subProg1" ist hier als Sprungmarke
                                        definiert.
N1560 ...
N1600 subProg3                          Aufruf von Unterprogramm "subProg3"
N1610 ...
N1900 M30
```

Unterprogramm subProg1:

```
Programmcode                    Kommentar
PROC subProg1
N2000 R10=R20+100
N2010 ...
N2200 RET("subProg2")           ; Rücksprung ins Hauptprogramm auf den Satz N1400
```

Unterprogramm subProg2:

```
Programmcode                    Kommentar
PROC subProg2
N2000 R10=R20+100
N2010 ...
N2200 RET("iVar1")              ; Rücksprung ins Hauptprogramm auf den Satz N1500
```

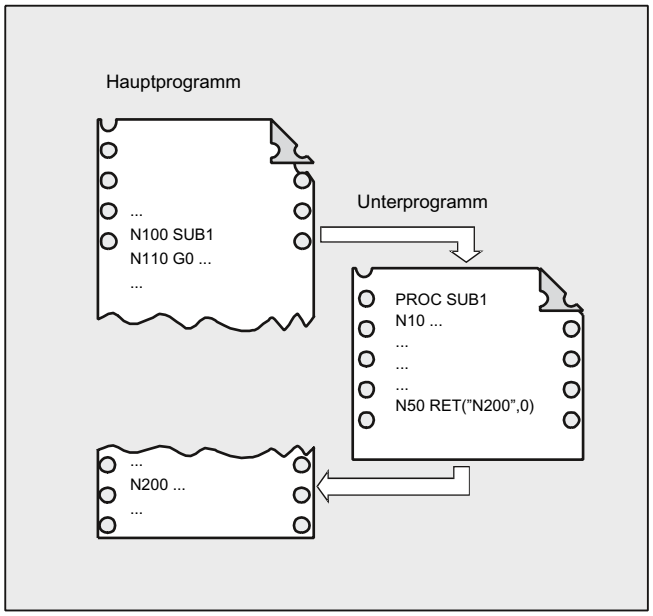
Unterprogramm subProg3:

```
Programmcode                    Kommentar
PROC subProg3
N2000 R10=R20+100
N2010 ...
N2200 RET("subProg1")           ; Rücksprung ins Hauptprogramm auf den Satz N1550
```

Weitere Informationen

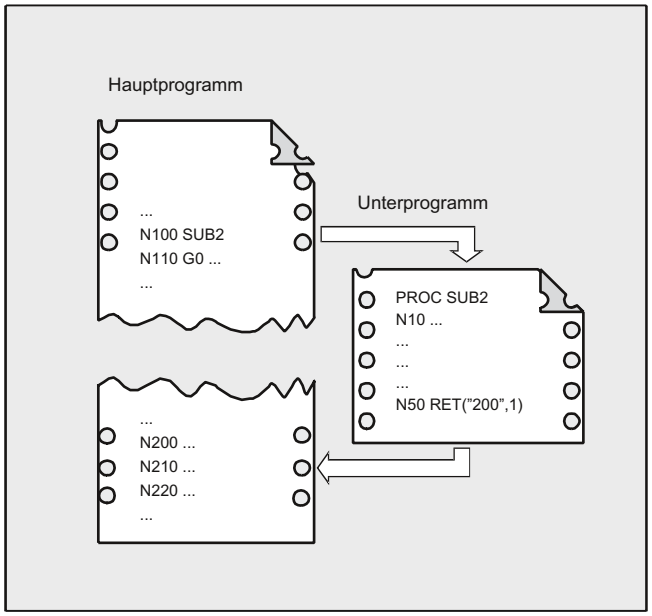
Die folgenden Grafiken sollen die unterschiedlichen Wirkungen der Rücksprungsparameter 1 bis 3 veranschaulichen.

1. Rücksprungsparameter 1 = "N200", Rücksprungsparameter 2 = 0



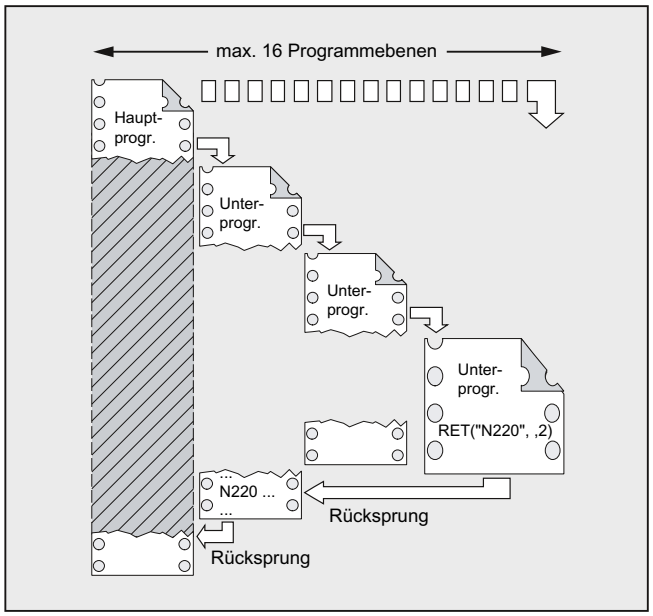
Nach dem `RET`-Befehl wird die Programmbearbeitung mit dem Satz `N200` im Hauptprogramm fortgesetzt.

2. Rücksprungsparameter 1 = "N200", Rücksprungsparameter 2 = 1



Nach dem RET-Befehl wird die Programmbearbeitung mit dem Satz (N210) fortgesetzt, der auf den Satz N200 im Hauptprogramm folgt.

3. Rücksprungsparameter 1 = "N220", Rücksprungsparameter 3 = 2



Nach dem RET-Befehl wird zwei Programmebenen zurückgesprungen und die Programmbearbeitung wird mit dem Satz N220 fortgesetzt.

1.25.3 Aufruf eines Unterprogramms

1.25.3.1 Unterprogrammaufruf ohne Parameterübergabe

Funktion

Der Aufruf eines Unterprogramms erfolgt entweder mit Adresse L und Unterprogrammnummer oder durch Angabe des Programmnamens.

Auch ein Hauptprogramm kann als Unterprogramm aufgerufen werden. Das im Hauptprogramm gesetzte Programmende `M2` oder `M30` wird in diesem Fall wie `M17` (Programmende mit Rücksprung ins aufrufende Programm) gewertet.

Hinweis

Entsprechend kann ein Unterprogramm auch als Hauptprogramm gestartet werden.

Suchstrategie der Steuerung:

Gibt es *_MPF?

Gibt es *_SPF?

Daraus folgt: Falls der Name des aufzurufenden Unterprogramms mit dem Namen des Hauptprogramms identisch ist, dann wird das aufrufende Hauptprogramm wieder aufgerufen. Dieser in der Regel nicht gewünschte Effekt muss durch eindeutige Namenswahl über Unterprogramme und Hauptprogramme vermieden werden.

Hinweis

Unterprogramme, die keine Parameterübergabe erfordern, können auch aus einer Initialisierungsdatei aufgerufen werden.

Syntax

```
L<Nummer>/<Programmname>
```

Hinweis

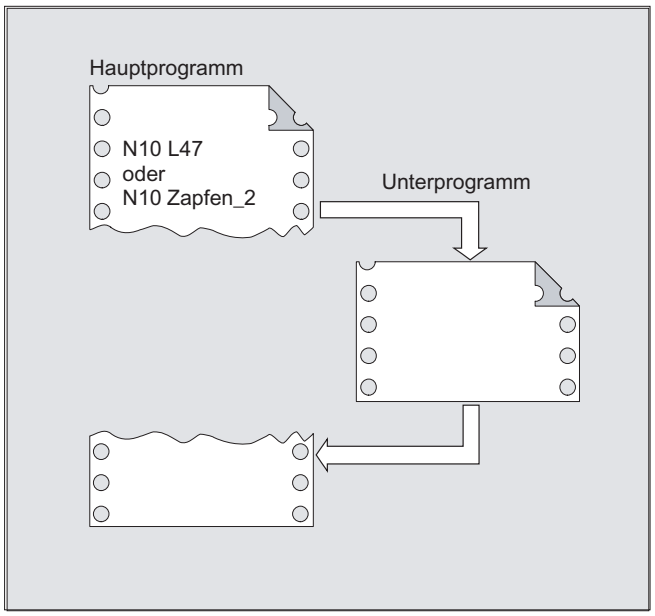
Der Aufruf eines Unterprogramms muss immer im eigenen NC-Satz programmiert werden.

Bedeutung

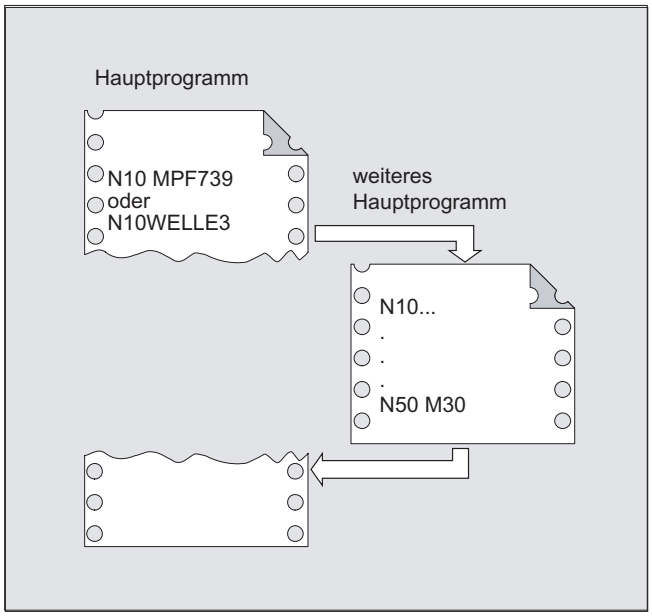
`L`:	Adresse für den Unterprogrammaufruf
`<Nummer>`:	Nummer des Unterprogramms Typ: INT Wert: Maximal 7 Dezimalstellen **Achtung:** Führende Nullen sind bei der Namensgebung von Bedeutung (⇒ L123, L0123 und L00123 sind drei verschiedene Unterprogramme).
`<Programmname>`:	Name des Unterprogramms (oder Hauptprogramms)

Beispiele

Beispiel 1: Aufruf eines Unterprogramms ohne Parameterübergabe



Beispiel 2: Aufruf eines Hauptprogramms als Unterprogramm



1.25.3.2 Unterprogrammaufruf mit Parameterübergabe (EXTERN)

Funktion

Beim Unterprogrammaufruf mit Parameterübergabe können Variablen oder Werte direkt übergeben werden (nicht bei VAR-Parametern).

Unterprogramme mit Parameterübergabe müssen vor dem Aufruf im Hauptprogramm mit EXTERN bekannt gemacht werden (z. B. am Programmanfang). Angegeben werden dabei der Name des Unterprogramms und die Variablentypen in der Reihenfolge der Übergabe.

⚠ VORSICHT
Sowohl die Variablentypen als auch die Reihenfolge der Übergabe muss mit den Definitionen, die im Unterprogramm unter PROC vereinbart wurden, übereinstimmen. Die Parameternamen können in Haupt- und Unterprogramm unterschiedlich sein.

Syntax

```
EXTERN <Programmname>(<Typ_Par1>,<Typ_Par2>,<Typ_Par3>)
...
<Programmname>(<Wert_Par1>,<Wert_Par2>,<Wert_Par3>)
```

⚠ VORSICHT
Der Unterprogrammaufruf muss immer im eigenen NC-Satz programmiert werden.

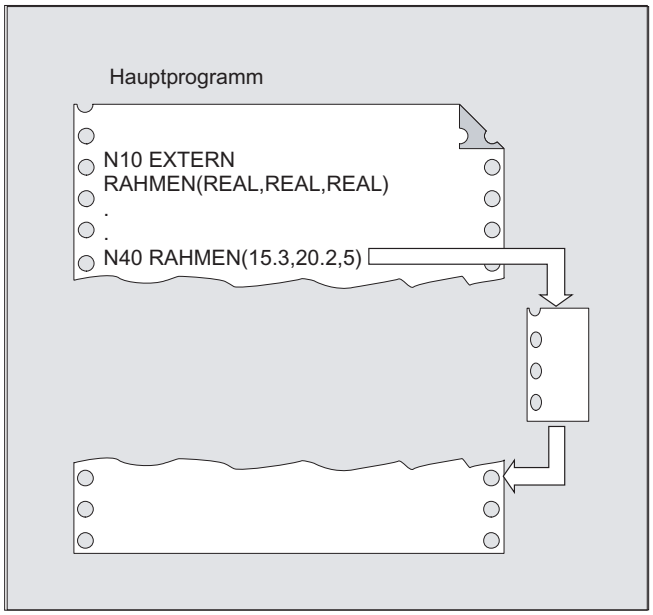
Bedeutung

`<Programmname>`:	Name des Unterprogramms
`EXTERN`:	Schlüsselwort für die Bekanntmachung eines Unterprogramms mit Parameterübergabe **Hinweis:** `EXTERN` muss nur dann angegeben werden, wenn das Unterprogramm im Werkstück- oder im globalen Unterprogrammverzeichnis steht. Zyklen müssen nicht als `EXTERN` erklärt werden.
`<Typ_Par1>,<Typ_Par2>,<Typ_Par3>`:	Variablentypen der zu übergebenden Parameter in der Reihenfolge der Übergabe
`<Wert_Par1>,<Wert_Par2>,<Wert_Par3>`:	Variablenwerte für die zu übergebenden Parameter

Beispiele

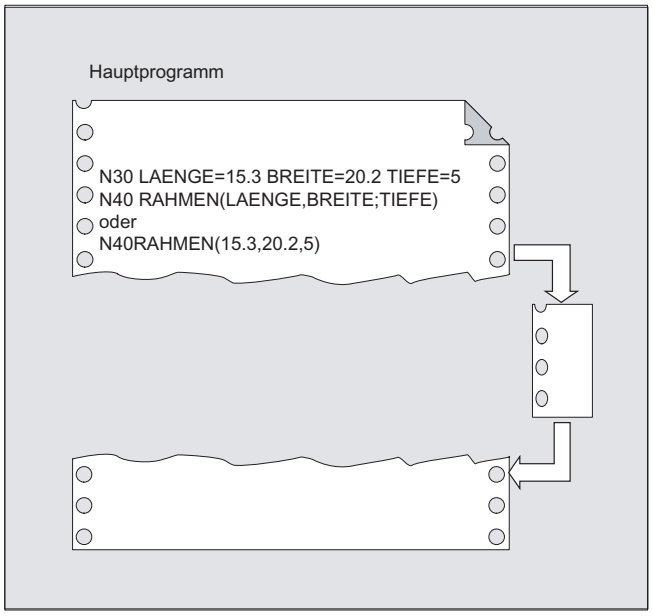
Beispiel 1: Unterprogrammaufruf mit vorhergehender Bekanntmachung

Programmcode	**Kommentar**
`N10 EXTERN RAHMEN(REAL,REAL,REAL)`	`; Angabe des Unterprogramms.`
`...`	
`N40 RAHMEN(15.3,20.2,5)`	`; Aufruf des Unterprogramms mit Parameterübergabe.`



Beispiel 2: Unterprogrammaufruf ohne Bekanntmachung

Programmcode	**Kommentar**
`N10 DEF REAL LAENGE, BREITE, TIEFE`	
`N20 …`	
`N30 LAENGE=15.3 BREITE=20.2 TIEFE=5`	
`N40 RAHMEN(LAENGE,BREITE,TIEFE)`	`; oder: N40 RAHMEN(15.3,20.2,5)`



1.25.3.3 Anzahl der Programmwiederholungen (P)

Funktion

Soll ein Unterprogramm mehrfach hintereinander abgearbeitet werden, kann im Satz mit dem Unterprogrammaufruf unter der Adresse `P` die gewünschte Anzahl der Programmwiederholungen programmiert werden.

> **⚠ VORSICHT**
>
> **Unterprogrammaufruf mit Programmwiederholung und Parameterübergabe**
>
> Parameter werden nur beim Programmaufruf bzw. ersten Durchlauf übergeben. Für die weiteren Wiederholungen bleiben die Parameter unverändert. Falls Sie bei Programmwiederholungen die Parameter verändern wollen, müssen Sie im Unterprogramm entsprechende Vereinbarungen festlegen.

Syntax

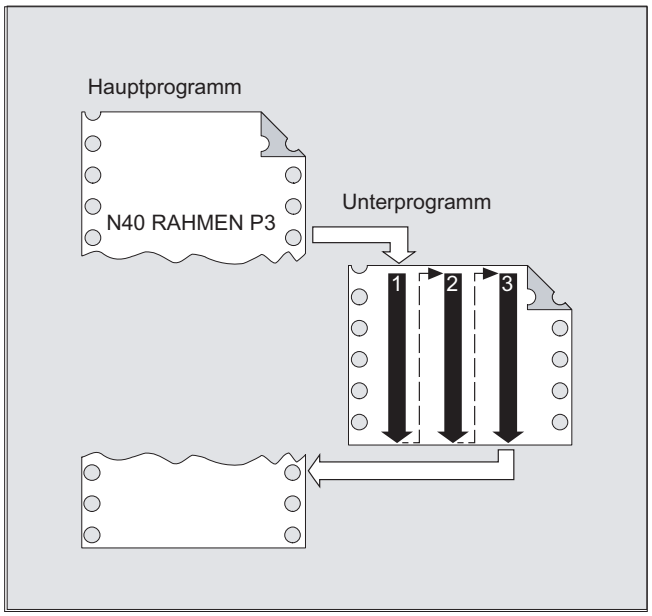
```
<Programmname> P<Wert>
```

Bedeutung

`<Programmname>`:	Unterprogrammaufruf	
`P`:	Adresse für die Programmierung von Programmwiederholungen	
`<Wert>`:	Anzahl der Programmwiederholungen	
	Typ:	INT
	Wertebereich:	1 … 9999 (ohne Vorzeichen)

Beispiel

```
Programmcode          Kommentar
...
N40 RAHMEN P3         ; Das Unterprogramm RAHMEN soll dreimal hintereinander
                        abgearbeitet werden.
...
```



1.25.3.4 Modaler Unterprogrammaufruf (MCALL)

Funktion

Bei einem modalen Unterprogrammaufruf mit MCALL wird das Unterprogramm nach jedem Satz mit Bahnbewegung automatisch aufgerufen und abgearbeitet. Hierdurch lässt sich der Aufruf von Unterprogrammen automatisieren, die an unterschiedlichen Werkstückpositionen abgearbeitet werden sollen (zum Beispiel für die Herstellung von Bohrbildern).

Das Ausschalten der Funktion erfolgt mit MCALL ohne Unterprogrammaufruf oder durch Programmierung eines neuen modalen Unterprogrammaufrufs für ein neues Unterprogramm.

⚠ VORSICHT

In einem Programmablauf kann gleichzeitig nur ein MCALL-Aufruf wirken. Parameter werden nur einmal beim MCALL-Aufruf übergeben.

Das modale Unterprogramm wird in folgenden Situationen auch ohne Programmierung einer Bewegung aufgerufen:

- Bei Programmierung der Adressen S und F wenn G0 oder G1 aktiv ist.
- Wenn G0/G1 allein im Satz oder mit weiteren G-Codes programmiert wurde.

Syntax

```
MCALL <Programmname>
```

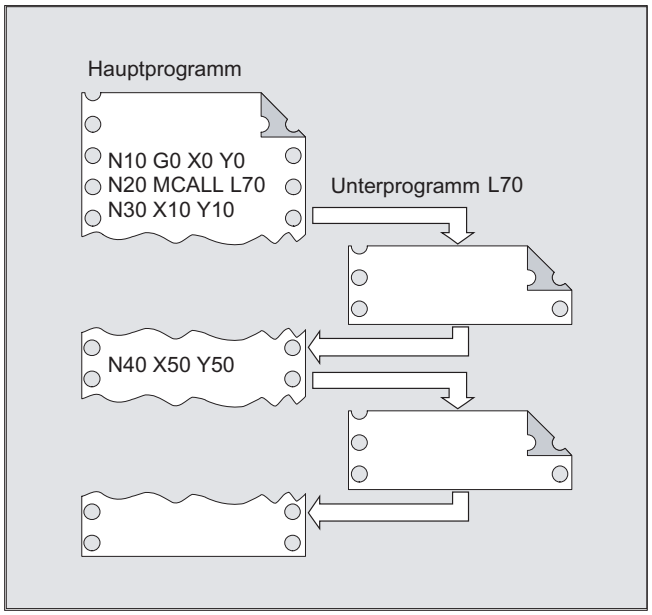
Bedeutung

MCALL:	Befehl für den modalen Unterprogrammaufruf
<Programmname>:	Name des Unterprogramms

Beispiele

Beispiel 1:

Programmcode	Kommentar
N10 G0 X0 Y0	
N20 MCALL L70	; Modaler Unterprogrammaufruf.
N30 X10 Y10	; Die programmierte Position wird angefahren und anschließend das Unterprogramm L70 abgearbeitet.
N40 X50 Y50	; Die programmierte Position wird angefahren und anschließend das Unterprogramm L70 abgearbeitet.



Beispiel 2:

Programmcode
N10 G0 X0 Y0
N20 MCALL L70
N30 L80

In diesem Beispiel stehen die nachfolgenden NC-Sätze mit programmierten Bahnachsen in Unterprogramm L80. L70 wird durch L80 aufgerufen.

1.25.3.5 Indirekter Unterprogrammaufruf (CALL)

Funktion

In Abhängigkeit von den gegebenen Bedingungen können an einer Stelle unterschiedliche Unterprogramme aufgerufen werden. Hierzu wird der Name des Unterprogramms in einer Variablen vom Typ STRING hinterlegt. Der Unterprogrammaufruf erfolgt mit `CALL` und dem Variablennamen.

⚠VORSICHT

Der indirekte Unterprogrammaufruf ist nur für Unterprogramme ohne Parameterübergabe möglich. Für den direkten Aufruf eines Unterprogramms hinterlegen Sie den Namen in einer STRING-Konstanten.

Syntax

```
CALL <Programmname>
```

Bedeutung

`CALL`:	Befehl für den indirekten Unterprogrammaufruf
`<Programmname>`:	Name des Unterprogramms (Variable oder Konstante) Typ: STRING

Beispiel

Direkter Aufruf mit STRING-Konstante:

```
Programmcode                                              Kommentar
…
CALL "/_N_WKS_DIR/_N_SUBPROG_WPD/_N_TEIL1_SPF"            ; Unterprogramm TEIL1 mit
                                                            CALL direkt aufrufen.
…
```

Indirekter Aufruf über Variable:

```
Programmcode                                                    Kommentar
…
DEF STRING[100] PROGNAME                                        ; Variable definieren.
PROGNAME="/_N_WKS_DIR/_N_SUBPROG_WPD/_N_TEIL1_SPF"              ; Unterprogramm TEIL1
                                                                  der Variablen
                                                                  PROGNAME zuordnen.
CALL PROGNAME                                                   ; Unterprogramm TEIL1
                                                                  über CALL und die
                                                                  Variable PROGNAME
                                                                  indirekt aufrufen.
…
```

1.25.3.6 Indirekter Unterprogrammaufruf mit Angabe des auszuführenden Programmteils (CALL BLOCK ... TO ...)

Funktion

Mit CALL und der Schlüsselwortkombination BLOCK ... TO wird ein Unterprogramm indirekt aufgerufen und der mit Start- und Endmarke gekennzeichnete Programmteil ausgeführt.

Syntax

```
CALL <Programmname> BLOCK <Startmarke> TO <Endmarke>
CALL BLOCK <Startmarke> TO <Endmarke>
```

Bedeutung

CALL:	Befehl für den indirekten Unterprogrammaufruf
<Programmname>:	Name des Unterprogramms (Variable oder Konstante), das den zu bearbeitenden Programmteil enthält (**Angabe optional**). Typ: STRING **Hinweis:** Ist kein <Programmname> programmiert, wird der mit <Startmarke> und <Endmarke> gekennzeichnete Programmteil im aktuellen Programm gesucht und ausgeführt.
BLOCK ... TO ... :	Schlüsselwortkombination für indirekte Programmteilausführung
<Startmarke>:	Variable, die auf den Beginn des zu bearbeitenden Programmteils verweist. Typ: STRING
<Endmarke>:	Variable, die auf das Ende des zu bearbeitenden Programmteils verweist. Typ: STRING

Beispiel

Hauptprogramm:

```
Programmcode                                          Kommentar
...
DEF STRING[20] STARTLABEL, ENDLABEL                   ; Variablendefinition für die
                                                        Start- und Endmarke.
STARTLABEL="LABEL_1"
ENDLABEL="LABEL_2"
...
CALL "CONTUR_1" BLOCK STARTLABEL TO ENDLABEL          ; Indirekter
                                                        Unterprogrammaufruf und
                                                        Kennzeichnung des
                                                        auszuführenden
                                                        Programmteils.
...
```

Unterprogramm:

```
Programmcode            Kommentar
PROC CONTUR_1 ...
LABEL_1                 ; Startmarke: Beginn der Programmteilausführung
N1000 G1 ...
...
LABEL_2                 ; Endmarke: Ende der Programmteilausführung
...
```

1.25.3.7 Indirekter Aufruf eines in ISO-Sprache programmierten Programms (ISOCALL)

Funktion

Mit dem indirekten Programmaufruf `ISOCALL` kann ein in einer ISO-Sprache programmiertes Programm aufgerufen werden. Dabei wird der in den Maschinendaten eingestellte ISO-Modus aktiviert. Am Programmende wird wieder der ursprüngliche Bearbeitungsmodus wirksam. Ist in den Maschinendaten kein ISO-Modus eingestellt, erfolgt der Aufruf des Unterprogramms im Siemens-Modus.

Weitere Informationen zum ISO-Modus siehe:
Literatur:
Funktionsbeschreibung ISO-Dialekte

Syntax

```
ISOCALL <Programmname>
```

Bedeutung

`ISOCALL`:	Schlüsselwort für indirekten Unterprogrammaufruf, mit dem der in den Maschinendaten eingestellte ISO-Modus aktiviert wird
`<Programmname>`:	Name des in einer ISO-Sprache programmierten Programms (Variable oder Konstante vom Typ STRING)

Beispiel: Kontur mit Zyklenprogrammierung aus dem ISO-Modus heraus aufrufen

Programmcode	**Kommentar**
0122_SPF	; Konturbeschreibung im ISO-Modus
N1010 G1 X10 Z20	
N1020 X30 R5	
N1030 Z50 C10	
N1040 X50	
N1050 M99	
N0010 DEF STRING[5] PROGNAME = "0122"	; Siemens-Teileprogramm (-Zyklus)
...	
N2000 R11 = $AA_IW[X]	
N2010 ISOCALL PROGNAME	
N2020 R10 = R10+1	; Programm 0122.spf im ISO-Modus bearbeiten
...	
N2400 M30	

1.25.3.8 Unterprogramm mit Pfadangabe und Parametern aufrufen (PCALL)

Funktion

Mit PCALL können Unterprogramme mit absoluter Pfadangabe und Parameterübergabe aufgerufen werden.

Syntax

```
PCALL <Pfad/Programmname>(<Parameter 1>,…,<Parameter n>)
```

Bedeutung

PCALL:	Schlüsselwort für Unterprogrammaufruf mit absoluter Pfadangabe.
<Pfad/Programmname>:	Absolute Pfadangabe beginnend mit "/", einschließlich Unterprogrammnamen. Wurde kein absoluter Pfad angegeben, verhält sich PCALL wie ein Standard-Unterprogrammaufruf mit Programmbezeichner. Der Programmbezeichner wird ohne Vorspann _N_ und ohne Erweiterung angegeben. Soll der Programmname mit Vorspann und Erweiterung programmiert werden, so muss er explizit mit Vorspann und Erweiterung mit dem Befehl EXTERN erklärt werden.
<Parameter 1>, ...:	Aktual-Parameter entsprechend der PROC-Anweisung des Unterprogramms.

Beispiel

Programmcode
PCALL/_N_WKS_DIR/_N_WELLE_WPD/WELLE(parameter1,parameter2,…)

1.25.3.9 Suchpfad bei Unterprogrammaufrufen erweitern (CALLPATH)

Funktion

Mit dem Befehl CALLPATH kann der Suchpfad für Unterprogrammaufrufe erweitert werden.

Damit können auch Unterprogramme aus einem nicht ausgewählten Werkstückverzeichnis aufgerufen werden, ohne den vollständigen, absoluten Pfadnamen des Unterprogramms anzugeben.

Die Suchpfaderweiterung erfolgt vor dem Eintrag für Anwenderzyklen (_N_CUS_DIR).

Durch folgende Ereignisse wird die Suchpfaderweiterung wieder abgewählt:

- CALLPATH mit Leerzeichen
- CALLPATH ohne Parameter
- Teileprogrammende
- Reset

Syntax

```
CALLPATH("<Pfadname>")
```

Bedeutung

CALLPATH:	Schlüsselwort für die programmierbare Suchpfaderweiterung. Wird in einer eigenen Teileprogrammzeile programmiert.
<Pfadname>:	Konstante oder Variable vom Typ STRING. Enthält die absolute Pfadangabe eines Verzeichnisses, um das der Suchpfad erweitert werden soll. Die Pfadangabe beginnt mit "/". Der Pfad muss vollständig mit Präfixen und Suffixen angegeben werden. Die maximale Pfadlänge beträgt 128 Bytes. Enthält der <Pfadname> ein Leerzeichen oder wird CALLPATH ohne Parameter aufgerufen, wird die Suchpfadanweisung wieder zurückgesetzt.

Beispiel

```
Programmcode
CALLPATH("/_N_WKS_DIR/_N_MYWPD_WPD")
```

Damit wird folgender Suchpfad eingestellt (Position 5. ist neu):

1. Aktuelles Directory/unterprogrammbezeichner
2. Aktuelles Directory/unterprogrammbezeichner_SPF
3. Aktuelles Directory/unterprogrammbezeichner_MPF
4. /_N_SPF_DIR/unterprogrammbezeichner_SPF
5. **/_N_WKS_DIR/_N_MYWPD/unterprogrammbezeichner_SPF**
6. /N_CUS_DIR/_N_MYWPD/unterprogrammbezeichner_SPF
7. /_N_CMA_DIR/unterprogrammbezeichner_SPF
8. /_N_CST_DIR/unterprogrammbezeichner_SPF

Randbedingungen

- CALLPATH prüft, ob der programmierte Pfadname tatsächlich vorhanden ist. Im Fehlerfall wird die Teileprogrammbearbeitung mit Korrektursatz-Alarm 14009 abgebrochen.
- CALLPATH kann auch in INI-Dateien programmiert werden. Er wirkt dann für die Bearbeitungsdauer der INI-Datei (WPD-INI-Datei oder Initialisierungsprogramm für NC-aktive Daten, z. B. Frames im 1. Kanal _N_CH1_UFR_INI). Danach wird der Suchpfad wieder zurückgesetzt.

1.25.3.10 Externes Unterprogramm abarbeiten (EXTCALL)

Funktion

Mit dem Befehl EXTCALL ist es möglich, ein Unterprogramm von einem externen Programmspeicher (Lokales Laufwerk, Netzlaufwerk, USB-Laufwerk) nachzuladen und abzuarbeiten.

Der Pfad zum externen Unterprogrammverzeichnis kann voreingestellt werden mit dem Settingdatum:

SD42700 $SC_EXT_PROG_PATH

Zusammen mit dem beim EXTCALL-Aufruf angegebenen Unterprogrammpfad bzw. -bezeichner ergibt sich daraus der Gesamtpfad des aufzurufenden Programms.

Hinweis

Externe Unterprogramme dürfen keine Sprunganweisungen wie GOTOF, GOTOB, CASE, FOR, LOOP, WHILE oder REPEAT enthalten.

IF-ELSE-ENDIF-Konstrukte sind möglich.

Unterprogrammaufrufe und geschachtelte EXTCALL-Aufrufe sind möglich.

Syntax

```
EXTCALL("<Pfad/><Programmname>")
```

Bedeutung

`EXTCALL`:	Befehl zum Aufrufen eines externen Unterprogramms	
`"<Pfad/><Programmname>"`:	Konstante/Variable vom Typ STRING	
	`<Pfad/>`:	Absolute oder relative Pfadangabe (**optional**)

	`<Programmname>`:	Der Programmname wird ohne Präfix "_N_" angegeben.
		Die Dateierweiterung ("MPF", "SPF") kann mit dem Zeichen "_" oder "." am Programmnamen angefügt werden (**optional**).
		Beispiel:
		`"WELLE"`
		oder
		`"WELLE_SPF"` bzw. `"WELLE.SPF"`

Hinweis

Pfadangabe: Kurzbezeichnungen

Bei der Pfadangabe können folgende Kurzbezeichnungen verwendet werden:

- **LOCAL_DRIVE:** für lokales Laufwerk
- **CF_CARD:** für CompactFlash-Card
- **USB:** für USB Front-Anschluss

CF_CARD: und **LOCAL_DRIVE:** sind alternativ verwendbar.

Hinweis

Abarbeiten von Extern über USB-Laufwerk

Sollen externe Programme von einem externen USB-Laufwerk über USB-Schnittstelle übertragen werden, so darf hierfür nur die Schnittstelle über X203 mit dem Namen "TCU_1" verwendet werden.

ACHTUNG

Abarbeiten von Extern über USB-FlashDrive (an USB Front-Anschluss)

Ein direktes Abarbeiten von einem USB-FlashDrive wird nicht empfohlen.

Es gibt keine Absicherung gegen Kontaktschwierigkeiten, Herausfallen, Abbrechen durch Anstoßen oder versehentliches Abziehen des USB-FlashDrive während des laufenden Betriebs.

Das Trennen während der Werkzeugbearbeitung führt zum sofortigen Stopp der Bearbeitung und somit auch zum Werkstückschaden.

Beispiel

Abarbeiten von lokalem Laufwerk

Hauptprogramm:

Programmcode

```
N010 PROC MAIN
N020 ...
N030 EXTCALL ("SCHRUPPEN")
N040 ...
N050 M30
```

Externes Unterprogramm:

Programmcode

```
N010 PROC SCHRUPPEN
N020 G1 F1000
N030 X= ... Y= ... Z= ...
N040 ...
...
...
N999999 M17
```

Das Hauptprogramm "MAIN.MPF" befindet sich im NC-Speicher und ist zur Abarbeitung angewählt.

Das nachzuladende Unterprogramm "SCHRUPPEN.SPF" bzw. "SCHRUPPEN.MPF" befindet sich auf dem lokalen Laufwerk in dem Verzeichnis "/user/sinumerik/data/prog/WKS.DIR/WST1.WPD".

Der Pfad zu dem Unterprogramm ist im SD42700 voreingestellt:

SD42700 $SC_EXT_PROG_PATH = "LOCAL_DRIVE:WKS.DIR/WST1.WPD"

Hinweis

Ohne Pfadangabe im SD42700 müsste die `EXTCALL`-Anweisung für dieses Beispiel wie folgt programmiert werden:

```
EXTCALL("LOCAL_DRIVE:WKS.DIR/WST1.WPD/SCHRUPPEN")
```

Weitere Informationen

EXTCALL-Aufruf mit absoluter Pfadangabe

Wenn das Unterprogramm unter dem angegebenen Pfad existiert, dann wird es nach dem EXTCALL-Aufruf ausgeführt. Wenn es nicht existiert, dann wird die Programmausführung abgebrochen.

EXTCALL-Aufruf mit relativer Pfadangabe / ohne Pfadangabe

Bei einem EXTCALL-Aufruf mit relativer Pfadangabe bzw. ohne Pfadangabe werden die vorhandenen Programmspeicher nach folgendem Muster durchsucht:

- Wenn in SD42700 $SC_EXT_PROG_PATH eine Pfadangabe voreingestellt ist, dann wird zuerst ausgehend von diesem Pfad nach der Angabe im EXTCALL-Aufruf (Programmname ggf. mit relativer Pfadangabe) gesucht. Der absolute Pfad ergibt sich dann durch Zeichenverkettung aus:
  - der in SD42700 voreingestellten Pfadangabe
  - dem Zeichen "/" als Trennzeichen
  - dem bei EXTCALL angegebenen Unterprogrammpfad bzw. -bezeichner
- Wurde das aufgerufene Unterprogramm unter dem voreingestellten Pfad nicht gefunden, werden als nächstes die Verzeichnisse des Anwenderspeichers nach der Angabe im EXTCALL-Aufruf durchsucht.
- Die Suche endet, wenn das Unterprogramm erstmalig gefunden wurde. Sollte die Suche keinen Treffer ergeben, kommt es zum Programmabbruch.

Einstellbarer Nachladespeicher (FIFO-Puffer)

Für die Bearbeitung eines Programms im Modus "Abarbeiten von Extern" (Hauptprogramm oder Unterprogramm) wird im NCK ein Nachladespeicher benötigt. Die Größe des Nachladespeichers ist mit 30 kByte voreingestellt und kann wie weitere speicherrelevante Maschinendaten nur vom Maschinenhersteller bedarfsorientiert verändert werden.

Für alle Programme (Hauptprogramme oder Unterprogramme), die gleichzeitig im Modus "Abarbeiten von Extern" bearbeitet werden, muss jeweils ein Nachladespeicher eingestellt werden.

RESET, POWER ON

Durch RESET und POWER ON werden externe Unterprogrammaufrufe abgebrochen und die jeweiligen Nachladespeicher gelöscht.

Ein für "Abarbeiten von Extern" selektiertes Unterprogramm bleibt über RESET / Teileprogrammende weiterhin für "Abarbeiten von Extern" angewählt. Durch POWER ON geht die Anwahl verloren.

Literatur

Weitere Informationen zu "Abarbeiten von Extern" siehe:

Funktionshandbuch Grundfunktionen; BAG, Kanal, Programmbetrieb, Reset-Verhalten (K1)

1.25.4 Zyklen

1.25.4.1 Zyklen: Anwenderzyklen parametrieren

Funktion

Mit den Dateien cov.com und uc.com können eigene Zyklen parametriert werden.

Die Datei cov.com wird mit den Standardzyklen geliefert und ist entsprechend zu erweitern. Die Datei uc.com ist vom Anwender selbst zu erstellen.

Beide Dateien sind im passiven Filesystem in das Verzeichnis "Anwenderzyklen" zu laden (bzw. mit entsprechender Pfadangabe):

```
;$PATH=/_N_CUS_DIR
```

im Programm zu versehen.

Dateien und Pfade

`cov.com_COM`	Übersicht über die Zyklen
`uc.com`	Zyklenaufrufbeschreibung

Anpassung cov.com - Übersicht der Zyklen

Die mit den Standardzyklen ausgelieferte Datei cov.com hat folgende Struktur:

`%_N_COV_COM`	Dateiname
`;$PATH=/_N_CST_DIR`	Pfadangabe
`;Vxxx 11.12.95 Sca Zyklenuebersicht`	Kommentarzeile
`C1(CYCLE81) Bohren, Zentrieren`	Aufruf für 1. Zyklus
`C2(CYCLE82) Bohren, Plansenken`	Aufruf für 2. Zyklus
`...`	
`C24(CYCLE98) Ketten von Gewinden`	Aufruf für letzten Zyklus
`M17`	Dateiende

Syntax

Für jeden neu hinzukommenden Zyklus ist eine Zeile in folgender Syntax einzufügen:

```
C<Nummer> (<Zyklusname>) Kommentartext
```

Nummer: eine beliebige ganze Zahl, sie darf bisher in der Datei noch nicht verwendet worden sein;

Zyklusname: der Programmname des einzubindenden Zyklus

Kommentartext: wahlweise ein Kommentartext zum Zyklus

Beispiel:

```
C25 (MEIN_ZYKLUS_1) Anwenderzyklus_1
C26 (SPEZIALZYKLUS)
```

Beispiel Datei uc.com - Anwenderzyklenbeschreibung

Die Erläuterung erfolgt anhand der Fortsetzung des
Beispiels:
Für die folgenden beiden Zyklen soll eine Zyklenparametrierung neu erstellt werden:

```
Programmierung          Kommentar
PROC MEIN_ZYKLUS_1 (REAL PAR1, INT PAR2, CHAR PAR3, STRING[10] PAR4)
Der Zyklus hat folgende Übergabeparameter:
PAR1:                ; Real-Wert im Bereich von -1000.001 <= PAR2 <= 123.456,
                       Vorbelegung mit 100
PAR2:                ; positiver ganzzahliger Wert zwischen 0 <= PAR3 <= 999999,
                       Vorbelegung mit 0
PAR3:                ; 1 ASCII-Zeichen
PAR4:                ; String der Länge 10 für einen Unterprogrammnamen
...
M17                  ;
```

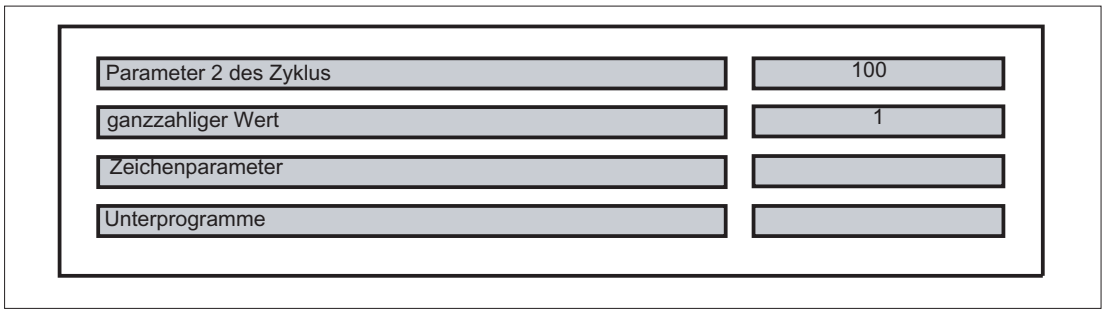
```
Programmierung        Kommentar
PROC SPEZIALZYKLUS (REAL WERT1, INT WERT2)
Der Zyklus hat folgende Übergabeparameter:
WERT1:               ; Real-Wert ohne Wertebereichsbeschränkung und Vorbelegung
WERT2:               ; ganzzahliger Wert ohne Wertebereichsbeschränkung und
                       Vorbelegung
...
M17                  ;
```

Zugehörige Datei uc.com:

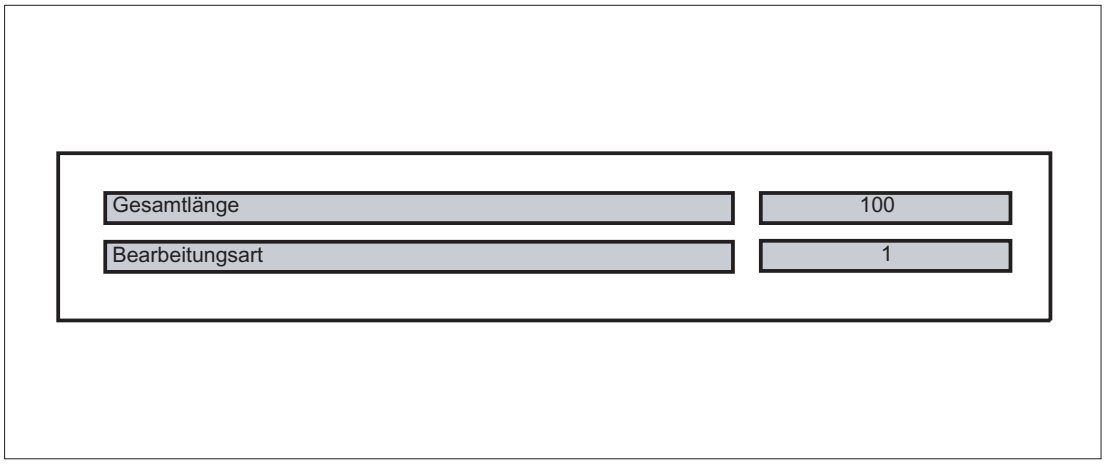
```
Programmierung
%_N_UC_COM
;$PATH=/_N_CUS_DIR
//C25(MEIN_ZYKLUS_1) Anwenderzyklus_1
(R/-1000.001 123.456 / 100 /Parameter_2 des Zyklus)
(I/0 999999 / 1 / ganzzahliger Wert)
(C//"A" / Zeichenparameter)
(S///Unterprogrammname)
//C26(SPEZIALZYKLUS)
(R///Gesamtlänge)
(I/*123456/3/Bearbeitungsart)
M17
```

Beispiel Beide Zyklen

Anzeigemaske für Zyklus `MEIN_ZYKLUS_1`



Anzeigemaske für Zyklus `SPEZIALZYKLUS`



Syntaxbeschreibung für die Datei uc.com - Anwenderzyklenbeschreibung

Kopfzeile pro Zyklus:

wie in der Datei cov.com mit vorgesetztem "//"

```
//C <Nummer> (<Zyklusname>) Kommentartext
```

Beispiel:

```
//C25 (MEIN_ZYKLUS_1) Anwenderzyklus_
```

Zeile für Beschreibung pro Parameter:

```
(<Datentypkennung> / <Minimalwert> <Maximalwert>
/ <Vorbelegungswert> /<Kommentar>)
```

Datentypkennung:

R	für Real
I	für Integer
C	für Charakter (1 Zeichen)
S	für String

Minimalwert, Maximalwert (kann entfallen)

Grenzen des einzugebenden Wertes, die bei der Eingabe überprüft werden; Werte außerhalb dieses Bereichs können nicht eingegeben werden. Es können Aufzählungswerte angegeben werden, die mit der Toggle-Taste bedient werden können; diese werden beginnend mit "*" aufgezählt, andere Werte sind dann nicht zulässig.

Beispiel:

```
(I/*123456/1/Bearbeitungsart)
```

Bei den Typen String und Charakter gibt es keine Grenzen.

Vorbelegungswert (kann entfallen)

Wert, der bei Aufruf des Zyklus in der entsprechenden Maske vorbesetzt ist; er kann per Bedienung geändert werden.

Kommentar

Text, maximal 50 Zeichen, der in der Aufrufmaske für den Zyklus vor dem Eingabefeld für den Parameter angezeigt wird.

1.26 Makrotechnik (DEFINE ... AS)

⚠ VORSICHT
Mit Makrotechnik kann die Programmiersprache der Steuerung stark verändert werden! Setzen Sie deshalb die Makrotechnik mit großer Sorgfalt ein!

Funktion

Als Makro bezeichnet man die Zusammenfassung von einzelnen Anweisungen zu einer neuen Gesamtanweisung mit eigenem Namen. Auch G-, M- und H-Funktionen oder L-Unterprogrammnamen können als Makros angelegt werden. Bei Aufruf des Makros im Programmablauf werden die unter dem Makronamen programmierten Anweisungen nacheinander abgearbeitet.

Anwendung

Anweisungsfolgen, die sich wiederholen, programmiert man nur einmal als Makro in einem eigenen Makrobaustein (Makrodatei) oder einmal am Programmanfang. Das Makro kann dann in jedem beliebigen Haupt- oder Unterprogramm aufgerufen und abgearbeitet werden.

Aktivierung

Um die Makros einer Makrodatei im NC-Programm verwenden zu können, muss die Makrodatei in die NC geladen werden.

Syntax

Makro-Definition:
```
DEFINE <Makroname> AS <Anweisung 1> <Anweisung 2> ...
```
Aufruf im NC-Programm:
```
<Makroname>
```

Bedeutung

`DEFINE ... AS` :	Schlüsselwort-Kombination zur Definition eines Makros
`<Makroname>`:	Name des Makros Als Makronamen sind nur Bezeichner zulässig. Mit dem Makronamen wird das Makro aus dem NC-Programm heraus aufgerufen.
`<Anweisung>`:	Programmieranweisung, die im Makro enthalten sein soll.

Regeln zur Makro-Definition

- Im Makro können beliebige Bezeichner, G-, M-, H-Funktionen und L-Programmnamen definiert werden.
- Makros können auch im NC-Programm definiert werden.
- G-Funktions-Makros können nur steuerungsglobal im Makrobaustein definiert werden.
- H- und L-Funktionen sind 2-stellig programmierbar.
- M- und G-Funktionen können 3-stellig programmiert werden.

⚠ VORSICHT
Schlüsselworte und reservierte Namen dürfen nicht mit Makros überdefiniert werden.

Randbedingungen

Eine Schachtelung von Makros ist nicht möglich.

Beispiele

Beispiel 1: Makrodefinition am Programmanfang

```
Programmcode                          Kommentar
DEFINE LINIE AS G1 G94 F300           ; Makro-Definition
...
...
N70 LINIE X10 Y20                     ; Makro-Aufruf
...
```

Beispiel 2: Makrodefinitionen in einer Makrodatei

```
Programmcode                          Kommentar
DEFINE M6 AS L6                       ; Beim Werkzeugwechsel wird ein Unterprogramm
                                        aufgerufen, das den nötigen Datentransfer
                                        übernimmt. Im Unterprogramm wird die eigentliche
                                        Werkzeugwechsel-M-Funktion ausgegeben (z. B.
                                        M106).
DEFINE G81 AS DRILL(81)               ; Nachbildung der DIN-G-Funktion.
DEFINE G33 AS M333 G333               ; Beim Gewindeschneiden wird Synchronisation mit
                                        der PLC angefordert. Die ursprüngliche G-Funktion
                                        G33 wurde per MD in G333 umbenannt, die
                                        Programmierung bleibt für den Anwender gleich.
```

Beispiel 3: Externe Makrodatei

Nach dem Einlesen der externen Makrodatei in die Steuerung muss die Makrodatei in die NC geladen werden. Erst dann können die Makros im NC-Programm verwendet werden.

Programmcode	**Kommentar**
`%_N_UMAC_DEF`	
`;$PATH=/_N_DEF_DIR`	`; Kundenspezifische Makros`
`DEFINE PI AS 3.14`	
`DEFINE TC1 AS M3 S1000`	
`DEFINE M13 AS M3 M7`	`; Spindel rechts, Kühlmittel ein`
`DEFINE M14 AS M4 M7`	`; Spindel links, Kühlmittel ein`
`DEFINE M15 AS M5 M9`	`; Spindel Halt, Kühlmittel aus`
`DEFINE M6 AS L6`	`; Aufruf des Werkzeugwechselprogramms`
`DEFINE G80 AS MCALL`	`; Abwahl Bohrzyklus`
`M30`	

Datei- und Programmverwaltung 2

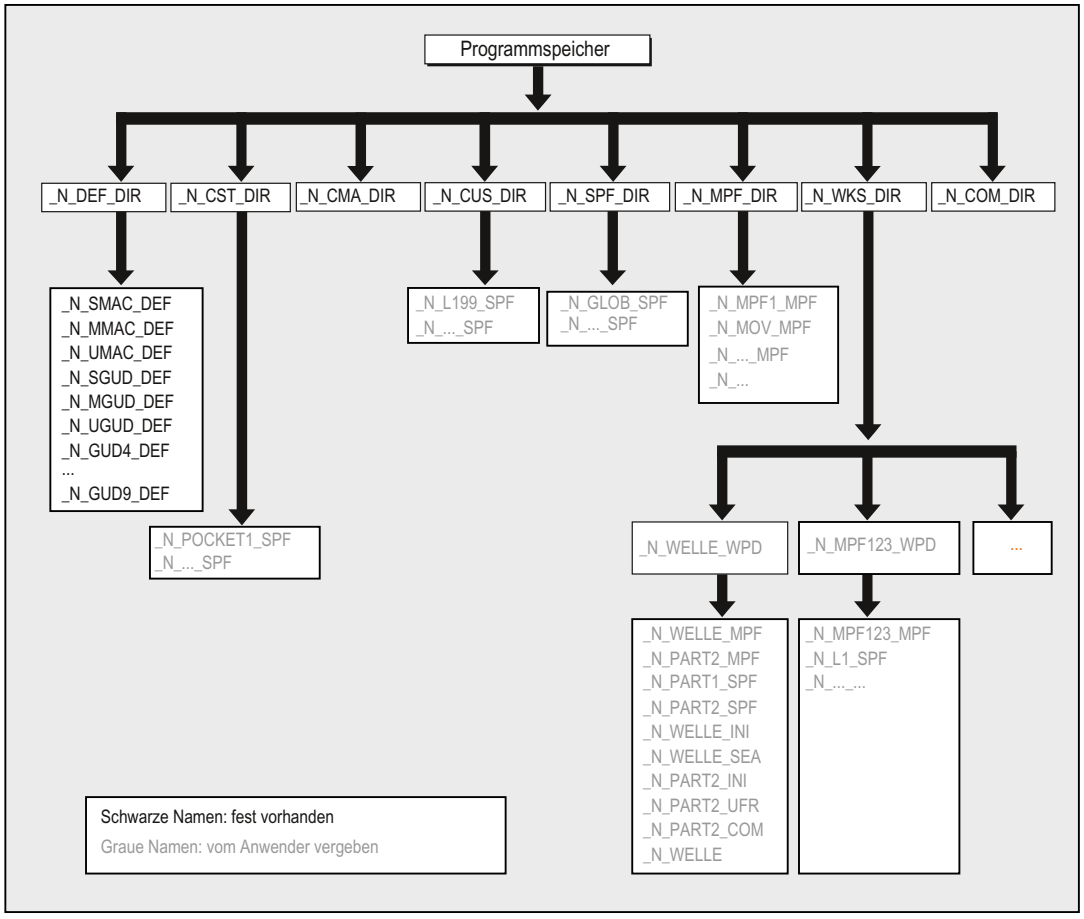
2.1 Programmspeicher

Funktion

Im Programmspeicher werden Dateien und Programme (z. B. Haupt- und Unterprogramme, Makro-Definitionen) persistent gespeichert (→ Passives Filesystem).

Literatur:
Funktionshandbuch Erweiterungsfunktionen; Speicherkonfiguration (S7)

Daneben gibt es eine Anzahl von Dateitypen, die hier zwischengespeichert werden können und bei Bedarf (z. B. bei Bearbeitung eines bestimmten Werkstückes) in den Arbeitsspeicher zu übertragen sind (z. B. für Initialisierungszwecke).



Standard-Verzeichnisse

Folgende Verzeichnisse sind standardmäßig vorhanden:

Verzeichnis	Inhalt
_N_DEF_DIR	Datenbausteine und Makrobausteine
_N_CST_DIR	Standard-Zyklen
_N_CMA_DIR	Hersteller-Zyklen
_N_CUS_DIR	Anwender-Zyklen
_N_WKS_DIR	Werkstücke
_N_SPF_DIR	Globale Unterprogramme
_N_MPF_DIR	Hauptprogramme
_N_COM_DIR	Kommentare

Dateitypen

Im Programmspeicher können folgende Dateitypen eingebracht werden:

Dateityp	Beschreibung
name_MPF	Hauptprogramm
name_SPF	Unterprogramm
name_TEA	Maschinendaten
name_SEA	Settingdaten
name_TOA	Werkzeugkorrekturen
name_UFR	Nullpunktverschiebungen/Frame
name_INI	Initialisierungsdatei
name_GUD	Globale Anwenderdaten
name_RPA	R-Parameter
name_COM	Kommentar
name_DEF	Definitionen für globale Anwenderdaten und Makros

Werkstück-Hauptverzeichnis (_N_WKS_DIR)

Das Werkstück-Hauptverzeichnis ist standardmäßig unter der Bezeichnung `_N_WKS_DIR` im Programmspeicher eingerichtet. Das Werkstück-Hauptverzeichnis enthält für alle Werkstücke, die Sie programmiert haben, die entsprechenden Werkstückverzeichnisse.

Werkstückverzeichnisse (..._WPD)

Für eine flexiblere Handhabung von Daten und Programmen können bestimmte Daten und Programme gebündelt oder in einzelnen Werkstückverzeichnissen abgelegt werden.

Ein Werkstückverzeichnis enthält alle Dateien, die zum Bearbeiten eines Werkstückes notwendig sind. Dies können Hauptprogramme, Unterprogramme, beliebige Initialisierungs-Programme und Kommentar-Dateien sein.

Initialisierungsprogramme werden nach der Programmanwahl mit dem ersten Teileprogrammstart einmalig ausgeführt (entsprechend Maschinendatum MD11280 $MN_WPD_INI_MODE).

Beispiel:

Das Werkstückverzeichnis _N_WELLE_WPD, das für das Werkstück WELLE angelegt wurde, enthält folgende Dateien:

Datei	Beschreibung
_N_WELLE_MPF	Hauptprogramm
_N_PART2_MPF	Hauptprogramm
_N_PART1_SPF	Unterprogramm
_N_PART2_SPF	Unterprogramm
_N_WELLE_INI	Allgemeines Initialisierungsprogramm der Daten für das Werkstück
_N_WELLE_SEA	Initialisierungsprogramm Settingdaten
_N_PART2_INI	Allgemeines Initialisierungsprogramm der Daten für Programm Part 2
_N_PART2_UFR	Initialisierungsprogramm für Frame-Daten für Programm Part 2
_N_WELLE_COM	Kommentardatei

Werkstückverzeichnisse am externen PC anlegen

Die nachstehend beschriebene Vorgehensweise wird an einer externen Datenstation durchgeführt. Für die Datei- und Programmverwaltung (vom PC zur Steuerung), direkt an der Steuerung, finden Sie die Informationen in Ihrer Bedienungsanleitung.

Werkstückverzeichnis anlegen mit Pfadangabe ($PATH=…)

In der zweiten Zeile einer Datei wird der Zielpfad mit `$PATH=…` angegeben. Die Datei wird dann unter dem angegebenen Pfad abgelegt.

Beispiel:

Programmcode

```
%_N_WELLE_MPF
;$PATH=/_N_WKS_DIR/_N_WELLE_WPD
N10 G0 X… Z…
...
M2
```

Die Datei _N_WELLE_MPF wird im Verzeichnis /_N_WKS_DIR/_N_WELLE_WPD abgelegt.

Werkstückverzeichnis anlegen ohne Pfadangabe

Fehlt die Pfadangabe, so werden Dateien mit der Endung _SPF im Verzeichnis /_N_SPF_DIR, Dateien mit der Endung _INI im Arbeitsspeicher und alle übrigen Dateien im Verzeichnis /_N_MPF_DIR abgelegt.

Beispiel:

Programmcode

```
%_N_WELLE_SPF
...
M17
```

Die Datei _N_WELLE_SPF wird im Verzeichnis /_N_SPF_DIR abgelegt.

Werkstück für die Bearbeitung anwählen

Ein Werkstückverzeichnis kann für die Abarbeitung in einem Kanal angewählt werden. Befindet sich in diesem Verzeichnis ein Hauptprogramm **gleichen Namens** oder nur ein einziges Hauptprogramm (_MPF), so wird dieses automatisch für die Abarbeitung angewählt.

Literatur:
/BAD/ Bedienungshandbuch HMI Advanced; Kapitel "Jobliste" sowie "Programm zur Abarbeitung anwählen"

Suchpfade beim Unterprogrammaufruf

Wird der Aufruf-Pfad nicht explizit im Teileprogramm beim Aufruf eines Unterprogramms (oder auch Initialisierungsdatei) angegeben, so wird das aufgerufene Programm nach einem festen Suchpfad ermittelt.

Unterprogrammaufruf mit absoluter Pfadangabe

Beispiel:

Programmcode

```
...
CALL"/_N_CST_DIR/_N_CYCLE1_SPF"
...
```

Unterprogrammaufruf ohne absoluter Pfadangabe

In der Regel werden die Programme ohne Pfadangabe aufgerufen.

Beispiel:

Programmcode

```
...
CYCLE1
...
```

Die Verzeichnisse werden nach dem aufgerufenen Programm in der folgenden Reihenfolge durchsucht:

Nr.	Verzeichnis	Beschreibung
1	aktuelles Directory / *name*	Werkstück-Hauptverzeichnis oder Standard-Verzeichnis _N_MPF_DIR
2	aktuelles Directory / *name_SPF*	
3	aktuelles Directory / *name_MPF*	
4	/_N_SPF_DIR / *name_SPF*	Globale Unterprogramme
5	/_N_CUS_DIR / *name_SPF*	Anwender-Zyklen
6	/_N_CMA_DIR / *name_SPF*	Hersteller-Zyklen
7	/_N_CST_DIR / *name_SPF*	Standard-Zyklen

Suchpfade beim Unterprogrammaufruf programmieren (CALLPATH)

Der Suchpfad beim Unterprogrammaufruf kann mit dem Teileprogrammbefehl `CALLPATH` erweitert werden.

Beispiel:

Programmcode

```
CALLPATH ("/_N_WKS_DIR/_N_MYWPD_WPD")
...
```

Der Suchpfad wird vor Position 5 (Anwender-Zyklus) entsprechend der angegebenen Programmierung abgelegt.

Weitere Informationen zum programmierbaren Suchpfad bei Unterprogrammaufrufen mit `CALLPATH` siehe Kapitel "Suchpfad bei Unterprogrammaufrufen mit CALLPATH erweitern".

2.2 Arbeitsspeicher (CHANDATA, COMPLETE, INITIAL)

Funktion

Der Arbeitsspeicher enthält die aktuellen System- und Anwenderdaten, mit denen die Steuerung betrieben wird (aktives Filesystem), z. B.:

- Aktive Maschinendaten
- Werkzeugkorrekturdaten
- Nullpunktverschiebungen
- ...

Initialisierungsprogramme

Hierbei handelt es sich um Programme, mit denen die Daten des Arbeitsspeichers vorbesetzt (initialisiert) werden. Hierfür können folgende Dateitypen verwendet werden:

Dateityp	Beschreibung
name_TEA	Maschinendaten
name_SEA	Settingdaten
name_TOA	Werkzeugkorrekturen
name_UFR	Nullpunktverschiebungen/Frame
name_INI	Initialisierungsdatei
name_GUD	Globale Anwenderdaten
name_RPA	R-Parameter

Informationen zu allen Dateitypen finden Sie im Bedienhandbuch zur Bedienoberfläche.

Datenbereiche

Die Daten können in unterschiedliche Bereiche eingegliedert werden, in denen sie gelten sollen. Beispielsweise kann eine Steuerung über mehrere Kanäle verfügen oder gewöhnlich auch über mehrere Achsen.

Es gibt:

Kennung	Datenbereiche
NCK	NCK-spezifische Daten
CH<n>	Kanalspezifische Daten (<n> gibt die Kanalnummer an)
AX<n>	Achsspezifische Daten (<n> gibt die Nummer der Maschinenachse an)
TO	Werkzeugdaten
COMPLETE	Alle Daten

Initialisierungsprogramm am externen PC erzeugen

Mit Hilfe von Datenbereichskennung und Datentypenkennung können die Bereiche bestimmt werden, die bei der Datensicherung als Einheit betrachtet werden:

_N_AX5_TEA_INI	Maschinendaten für Achse 5
_N_CH2_UFR_INI	Frames des Kanals 2
_N_COMPLETE_TEA_INI	Alle Maschinendaten

Nach Inbetriebnahme der Steuerung ist ein Datensatz im Arbeitsspeicher vorhanden, der den ordnungsgemäßen Betrieb der Steuerung gewährleistet.

Vorgehensweise bei mehrkanaligen Steuerungen (CHANDATA)

`CHANDATA(<Kanalnummer>)` für mehrere Kanäle ist nur in der Datei _N_INITIAL_INI zulässig. Das ist die Inbetriebnahmedatei, mit der alle Daten der Steuerung initialisiert werden.

```
Programmcode                              Kommentar
%_N_INITIAL_INI
CHANDATA(1)
                                          ; Maschinenachszuordnung Kanal 1:
$MC_AXCONF_MACHAX_USED[0]=1
$MC_AXCONF_MACHAX_USED[1]=2
$MC_AXCONF_MACHAX_USED[2]=3
CHANDATA(2)
                                          ; Maschinenachszuordnung Kanal 2:
$MC_AXCONF_MACHAX_USED[0]=4
$MC_AXCONF_MACHAX_USED[1]=5
CHANDATA(1)
                                          ; Axiale Maschinendaten:
                                          ; Genauhaltfenster grob:
$MA_STOP_LIMIT_COARSE[AX1]=0.2            ; Achse 1
$MA_STOP_LIMIT_COARSE[AX2]=0.2            ; Achse 2
                                          ; Genauhaltfenster fein:
$MA_STOP_LIMIT_FINE[AX1]=0.01             ; Achse 1
$MA_STOP_LIMIT_FINE[AX1]=0.01             ; Achse 2
```

⚠ VORSICHT

CHANDATA-Anweisung

Im Teileprogramm darf die `CHANDATA`-Anweisung nur für den Kanal gesetzt werden, auf dem das NC-Programm abgearbeitet wird, d. h. die Anweisung kann dazu benutzt werden, NC-Programme davor zu schützen, dass sie auf einem nicht vorgesehenen Kanal abgearbeitet werden.

Im Fehlerfall wird die Programmabarbeitung abgebrochen.

Hinweis

INI-Dateien in Joblisten enthalten keine CHANDATA-Anweisungen.

Initialisierungsprogramme sichern (COMPLETE, INITIAL)

Die Dateien des Arbeitsspeichers können auf einem externen PC gesichert und von dort wieder eingelesen werden.

- Die Dateien werden mit COMPLETE gesichert.
- Mit INITIAL wird über alle Bereiche eine INI-Datei (_N_INITIAL_INI) erzeugt.

Initialisierungsprogramme einlesen

ACHTUNG

Wird die Datei mit dem Namen "INITIAL_INI" eingelesen, so werden alle Daten, die in der Datei nicht versorgt werden, mit Standarddaten initialisiert. Ausgenommen davon sind nur die Maschinendaten. Es werden also **Settingdaten, Werkzeugdaten, NPV, GUD-Werte, ...** mit Standarddaten (normalerweise "NULL") versorgt.

Zum Einlesen von einzelnen Maschinendaten eignet sich z. B. die Datei COMPLETE_TEA_INI. In dieser Datei erwartet die Steuerung nur Maschinendaten. Damit bleiben die anderen Datenbereiche in diesem Fall unberührt.

Initialisierungsprogramme laden

Die INI-Programme können auch als Teileprogramme angewählt und aufgerufen werden, wenn sie nur Daten eines Kanals verwenden. So ist es auch möglich, programmgesteuerte Daten zu initialisieren.

2.3 Strukturierungsanweisung im Stepeditor (SEFORM)

Funktion

Die Strukturierungsanweisung SEFORM wird im Stepeditor (editorbasierte Programmunterstützung) ausgewertet, um daraus die Schrittansicht für HMI Advanced zu generieren. Die Schrittansicht dient zur besseren Lesbarkeit des NC-Unterprogramms.

Syntax

```
SEFORM(<Abschnittsname>,<Ebene>,<Icon>)
```

Bedeutung

SEFORM()	Funktionsaufruf der Strukturierungsanweisung mit den Parametern <Abschnittsname>, <Ebene> und <Icon>		
<Abschnittsname>	Bezeichner des Arbeitsschritts		
	Typ:	STRING	
<Ebene>	Index für die Haupt- oder Unterebene		
	Typ:	INT	
	Wert:	0	Hauptebene
		1, ..., <n>	Unterebene 1, ... , Unterebene <n>
<Icon>	Name des Icons, welches für diesen Abschnitt angezeigt werden soll.		
	Typ:	STRING	

Hinweis

SEFORM-Anweisungen werden im Stepeditor erzeugt.

Die mit dem Parameter <Abschnittsname> übergebene Zeichenkette wird analog zur MSG-Anweisung hauptlaufsynchron in der BTSS-Variable abgelegt. Die Information bleibt bis zum Überschreiben der nächsten SEFORM-Anweisung bestehen. Mit Reset und Teileprogrammende wird der Inhalt gelöscht.

Die Parameter <Ebene> und <Icon> werden bei der Teileprogrammbearbeitung vom NCK geprüft, aber nicht weiterverarbeitet.

Literatur

Weitere Informationen zur editorbasierten Programmierunterstützung siehe:
Bedienhandbuch HMI Advanced

Schutzbereiche 3

3.1 Festlegung der Schutzbereiche (CPROTDEF, NPROTDEF)

Funktion

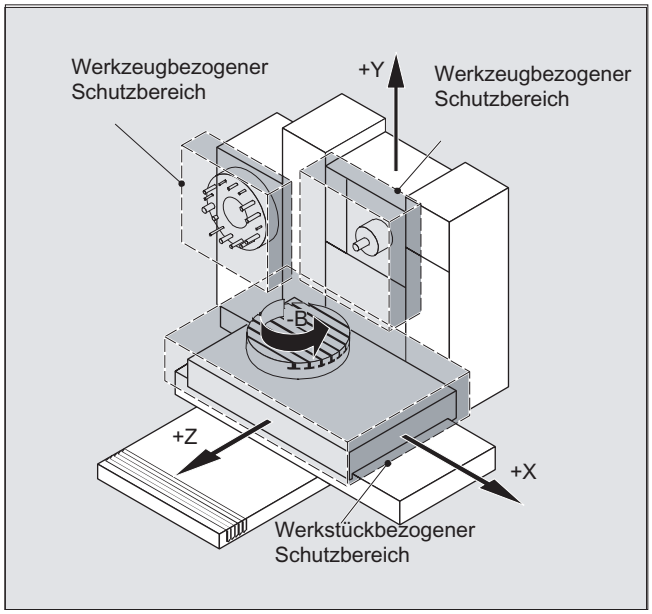
Mit Hilfe von Schutzbereichen lassen sich verschiedene Elemente an der Maschine, die Ausrüstung sowie das Werkstück vor falschen Bewegungen schützen.

Werkzeugbezogene Schutzbereiche:

Für Teile, die zum Werkzeug gehören (z. B. Werkzeug, Werkzeugträger).

Werkstückbezogene Schutzbereiche:

Für Teile, die zum Werkstück gehören (z. B. Teile des Werkstücks, Aufspanntisch, Spannpratzen, Spindelfutter, Reitstock).



Syntax

```
DEF INT NOT_USED
G17/G18/G19
CPROTDEF/NPROTDEF(<n>,<t>,<applim>,<applus>,<appminus>)
G0/G1/... X/Y/Z...
...
EXECUTE(NOT_USED)
```

Bedeutung

DEF INT NOT_USED:	Lokale Variable, Datentyp INTEGER definieren (vgl. Kapitel "Bewegungssynchronaktionen (Seite 551)")
G17/G18/G19:	Die gewünschte Ebene wird vor CPROTDEF bzw. NPROTDEF mit G17/G18/G19 angewählt und darf vor EXECUTE nicht geändert werden. Eine Programmierung der Applikate zwischen CPROTDEF bzw. NPROTDEF und EXECUTE ist nicht zulässig.
CPROTDEF:	Kanalspezifische Schutzbereiche (nur für NCU 572/573) definieren
NPROTDEF:	Maschinenspezifische Schutzbereiche definieren
G0/G1/... X/Y/Z... ...:	Die Kontur der Schutzbereiche wird mit maximal 11 Verfahrbewegungen in der angewählten Ebene angegeben. Dabei ist die erste Verfahrbewegung die Bewegung an die Kontur. Als Schutzbereich gilt dabei der Bereich links von der Kontur. **Hinweis:** Die zwischen CPROTDEF bzw. NPROTDEF und EXECUTE stehenden Verfahrbewegungen werden nicht ausgeführt, sondern definieren den Schutzbereich.
EXECUTE:	Definition beenden
<n>:	Nummer des definierten Schutzbereichs
<t>:	Typ des Schutzbereichs TRUE: **Werkzeug**bezogener Schutzbereich FALSE: **Werkstück**bezogener Schutzbereich
<applim>:	Art der Begrenzung in der 3. Dimension 0: keine Begrenzung 1: Begrenzung in Plus-Richtung 2: Begrenzung in Minus-Richtung 3: Begrenzung in Plus- und Minus-Richtung
<applus>:	Wert der Begrenzung in Plus-Richtung der 3. Dimension
<appminus>:	Wert der Begrenzung in Minus-Richtung der 3. Dimension
NOT_USED:	Die Fehlervariable ist bei Schutzbereichen mit EXECUTE wirkungslos

Randbedingungen

Während der Definition der Schutzbereiche darf:

- keine Fräserradius- bzw. Schneidenradiuskorrektur aktiv sein.
- keine Transformation aktiv sein.
- kein Frame aktiv sein.

Es darf auch nicht Referenzpunktanfahren (G74), Festpunktanfahren (G75), Satzvorlauf-Stopp oder Programmende programmiert sein.

Weitere Informationen

Definition von Schutzbereichen

Zur Definition von Schutzbereichen gehören:

- `CPROTDEF` für kanalspezifische Schutzbereiche
- `NPROTDEF` für maschinenspezifische Schutzbereiche
- Konturbeschreibung des Schutzbereichs
- Abschluss der Definition mit `EXECUTE`

Bei Aktivierung des Schutzbereichs im NC-Teileprogramm können Sie den Bezugspunkt des Schutzbereichs relativ verschieben.

Bezugspunkt der Konturbeschreibung

Die werkstückbezogenen Schutzbereiche werden im Basiskoordinatensystem definiert.

Die werkzeugbezogenen Schutzbereiche werden bezogen auf den Werkzeugträgerbezugspunkt F angegeben.

Zulässige Konturelemente

Für die Konturbeschreibung des Schutzbereichs sind zulässig:

- `G0`, `G1` für gerade Konturelemente
- `G2` für Kreisabschnitte im Uhrzeigersinn (nur für werkstückbezogene Schutzbereiche)
- `G3` für Kreisabschnitte gegen den Uhrzeigersinn

Hinweis

Soll ein Vollkreis den Schutzbereich beschreiben, so ist er in zwei Teilkreise aufzuteilen. Die Folge `G2`, `G3` bzw. `G3`, `G2` ist nicht zulässig. Hier ist ggf. ein kurzer `G1`-Satz einzuschieben.

Der letzte Punkt der Konturbeschreibung muss mit dem ersten Punkt zusammenfallen.

Außenschutzbereiche

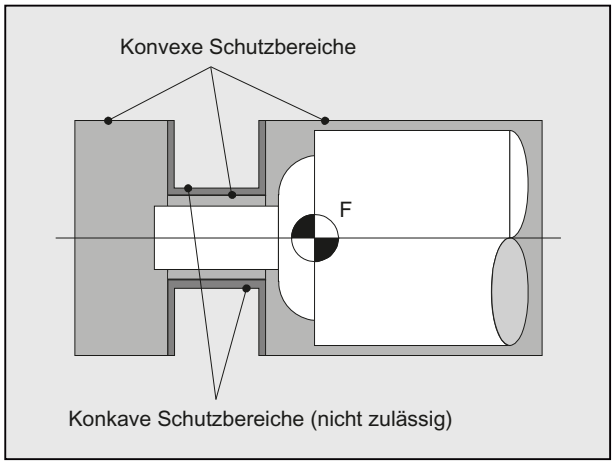
Außenschutzbereiche (nur bei werkstückbezogenen Schutzbereichen möglich) sind im Uhrzeigersinn zu definieren.

Rotationssymmetrische Schutzbereiche

Bei rotationssymmetrischen Schutzbereichen (z. B. Spindelfutter) muss die Gesamtkontur beschrieben werden (nicht nur bis zur Drehmitte!).

Werkzeugbezogene Schutzbereiche

Werkzeugbezogene Schutzbereiche müssen immer konvex sein. Falls ein konkaver Schutzbereich gewünscht ist, ist dieser in mehrere konvexe Schutzbereiche zu zerlegen.



3.2 Schutzbereiche aktivieren/deaktivieren (CPROT, NPROT)

Funktion

Vorher definierte Schutzbereiche zur Kollisionsüberwachung aktivieren, voraktivieren oder aktive Schutzbereiche deaktivieren.

Die maximale Anzahl der gleichzeitig in einem Kanal aktiven Schutzbereiche wird über Maschinendatum festgelegt.

Ist kein werkzeugbezogener Schutzbereich aktiv, so wird die Werkzeugbahn gegen die werkstückbezogenen Schutzbereiche geprüft.

Hinweis

Ist kein werkstückbezogener Schutzbereich aktiv, so findet keine Schutzbereichsüberwachung statt.

Syntax

```
CPROT(<n>,<state>,<xMov>,<yMov>,<zMov>)
NPROT(<n>,<state>,<xMov>,<yMov>,<zMov>)
```

Bedeutung

`CPROT`:	Aufruf kanalspezifischer Schutzbereich (nur für NCU 572/573)
`NPROT`:	Aufruf maschinenspezifischer Schutzbereich
`<n>`:	Nummer des Schutzbereichs
`<state>`:	Statusangabe 0: Schutzbereich deaktivieren 1: Schutzbereich voraktivieren 2: Schutzbereich aktivieren 3: Schutzbereich voraktivieren mit bedingtem Stopp
`<xMov>,<yMov>,<zMov>`:	Bereits definierten Schutzbereich in den Geometrieachsen verschieben

Randbedingungen

Schutzbereichsüberwachung bei aktiver Werkzeugradiuskorrektur

Bei aktiver Werkzeugradiuskorrektur ist eine funktionsfähige Schutzbereichsüberwachung nur möglich, wenn die Ebene der Werkzeugradiuskorrektur identisch ist mit der Ebene der Schutzbereichsdefinitionen.

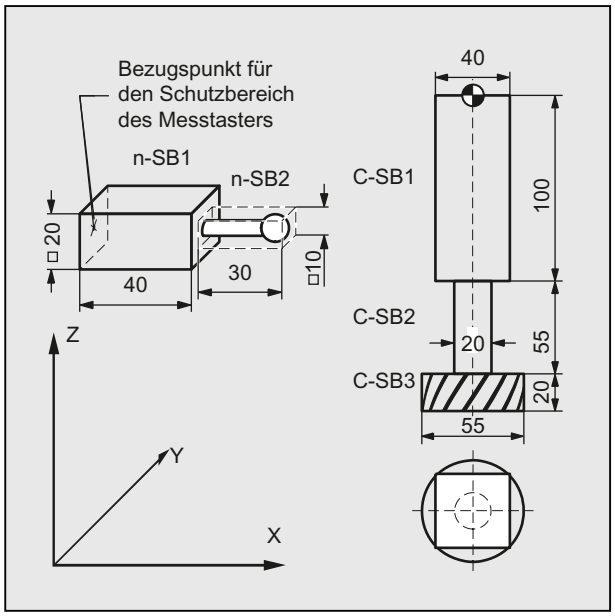
Beispiel

Für eine Fräsmaschine soll eine mögliche Kollision des Fräsers mit dem Messtaster überwacht werden. Die Lage des Messtasters soll bei der Aktivierung durch eine Verschiebung angegeben werden. Es werden dafür folgende Schutzbereiche definiert:

- Jeweils ein maschinenspezifischer und werkstückbezogener Schutzbereich für den Messtasterhalter (n-SB1) und für den Messtaster selbst (n-SB2).
- Jeweils ein kanalspezifischer und werkzeugbezogener Schutzbereich für den Fräserhalter (c-SB1), den Fräserschaft (c-SB2) und für den Fräser selbst (c-SB3).

Die Orientierung aller Schutzbereiche liegt in Z-Richtung.

Die Lage des Bezugspunkts des Messtasters bei der Aktivierung soll bei X = -120, Y = 60 und Z = 80 liegen.



Programmcode	**Kommentar**
`DEF INT SCHUTZB`	`; Definition einer Hilfsvariablen`
`Definition der SchutzbereicheG17`	`; Orientierung einstellen`
`NPROTDEF(1,FALSE,3,10,-10)G01 X0 Y-10`	`; Schutzbereich n-SB1`
`X40`	
`Y10`	
`X0`	
`Y-10`	
`EXECUTE(SCHUTZB)`	

Programmcode	**Kommentar**
NPROTDEF(2,FALSE,3,5,-5)	; Schutzbereich n-SB2
G01 X40 Y-5	
X70	
Y5	
X40	
Y-5	
EXECUTE(SCHUTZB)	
CPROTDEF(1,TRUE,3,0,-100)	; Schutzbereich c-SB1
G01 X-20 Y-20	
X20	
Y20	
X-20	
Y-20	
EXECUTE(SCHUTZB)	
CPROTDEF(2,TRUE,3,-100,-150)	; Schutzbereich c-SB2
G01 X0 Y-10	
G03 X0 Y10 J10	
X0 Y-10 J-10	
EXECUTE(SCHUTZB)	
CPROTDEF(3,TRUE,3,-150,-170)	; Schutzbereich c-SB3
G01 X0 Y-27,5	
G03 X0 Y27,5 J27,5	
X0 Y27,5 J-27,5	
EXECUTE(SCHUTZB)	
Aktivierung der Schutzbereiche:	
NPROT(1,2,-120,60,80)	; Schutzbereich n-SB1 m. Versch. aktivieren
NPROT(2,2,-120,60,80)	; Schutzbereich n-SB2 m. Versch. aktivieren
CPROT(1,2,0,0,0)	; Schutzbereich c-SB1 m. Versch. aktivieren
CPROT(2,2,0,0,0)	; Schutzbereich c-SB2 m. Versch. aktivieren
CPROT(3,2,0,0,0)	; Schutzbereich c-SB3 m. Versch. aktivieren

Weitere Informationen

Aktivierungsstatus (<state>)

- **<state>=2**

  Ein Schutzbereich wird im Allgemeinen im Teileprogramm mit Status = 2 aktiviert.

  Der Status ist immer kanalspezifisch, auch bei maschinenbezogenen Schutzbereichen.

- **<state>=1**

  Wenn durch das PLC-Anwenderprogramm vorgesehen ist, dass ein Schutzbereich durch das PLC-Anwenderprogramm wirksam gesetzt werden kann, so erfolgt die dafür erforderliche Voraktivierung durch den Status = 1.

- **<state>=3**

  Bei der Voraktivierung mit bedingtem Stopp wird nicht grundsätzlich vor einem verletzten, voraktivierten Schutzbereich angehalten. Der Stopp erfolgt nur dann, wenn der Schutzbereich wirksam gesetzt worden ist. Dies ermöglicht eine unterbrechungsfreie Bearbeitung, wenn die Schutzbereiche nur in besonderen Fällen wirksam gesetzt werden. Zu beachten ist, dass infolge der Bremsrampe ggf. in einen Schutzbereich gefahren wird, falls der Schutzbereich erst unmittelbar vor dem Einfahren wirksam gesetzt worden ist.

  Die Voraktivierung mit bedingtem Stopp erfolgt durch den Status = 3.

- **<state>=0**

  Die Deaktivierung und damit das Ausschalten der Schutzbereiche erfolgt durch den Status = 0. Es ist dabei keine Verschiebung notwendig.

Verschiebung von Schutzbereichen beim (Vor-)Aktivieren

Die Verschiebung kann in 1, 2 oder 3 Dimensionen erfolgen. Die Angabe der Verschiebung bezieht sich auf:

- den Maschinennullpunkt bei werkstückspezifischen Schutzbereichen.
- den Werkzeugträgerbezugspunkt F bei werkzeugspezifischen Schutzbereichen.

Status nach dem Hochlaufen

Schutzbereiche können bereits nach dem Hochlaufen und anschließendem Referenzpunktanfahren aktiviert sein. Es muss dafür die Systemvariable $SN_PA_ACTIV_IMMED[<n>] bzw. $SC_PA_ACTIV_IMMED[<n>] auf TRUE gesetzt sein. Sie werden immer mit dem Status = 2 aktiviert und haben keine Verschiebung.

Mehrfache Aktivierung von Schutzbereichen

Ein Schutzbereich kann gleichzeitig auch in mehreren Kanälen wirksam sein (z. B. Pinole bei zwei gegenüberliegenden Schlitten). Die Überwachung der Schutzbereiche erfolgt nur, wenn alle Geometrieachsen referiert sind.

Dabei gilt:

- Der Schutzbereich ist in einem Kanal nicht gleichzeitig mehrfach mit verschiedenen Verschiebungen aktivierbar.
- Maschinenbezogene Schutzbereiche müssen in beiden Kanälen die gleiche Orientierung aufweisen.

3.3 Überprüfung auf Schutzbereichsverletzung, Arbeitsfeldbegrenzung und Softwarelimits (CALCPOSI)

Funktion

Die Funktion CALCPOSI dient dazu zu überprüfen, ob ausgehend von einem gegebenen Startpunkt die Geometrieachsen einen vorgegebenen Weg verfahren können, ohne die Achsgrenzen (Softwarelimits), Arbeitsfeldbegrenzungen oder Schutzbereiche zu verletzen.

Für den Fall, dass der vorgegebene Weg nicht gefahren werden kann, wird der maximal zulässige Wert zurückgeben.

Die Funktion CALCPOSI ist ein vordefiniertes Unterprogramm. Sie muss alleine in einem Satz stehen.

Syntax

```
Status=CALCPOSI(_STARTPOS, _MOVDIST, _DLIMIT, _MAXDIST, _BASE_SYS, _TESTLIM)
```

Bedeutung

`Status`	0: Funktion o. k., der vorgegebene Weg kann vollständig abgefahren werden. –: In _DLIMIT ist mindestens eine Komponente negativ –: In einer Transformationsberechnung ist ein Fehler aufgetreten Kann der vorgegebene Weg nicht vollständig abgefahren werden, wird ein positiver, dezimal codierter Wert zurückgegeben: **Einerstelle (Art der verletzten Grenze):** 1: Softwarelimits begrenzen den Verfahrweg. 2: Arbeitsfeldbegrenzung begrenzt den Verfahrweg. 3: Schutzbereiche begrenzen den Verfahrweg. Sind gleichzeitig mehrere Grenzen verletzt (z. B. Softwarelimits und Schutzbereiche), wird in der Einerstelle die Grenze gemeldet, die zur stärksten Einschränkung des vorgegebenen Verfahrweges führt.

Zehnerstelle

10:

Der Anfangswert verletzt die Grenze

20:

Die vorgegebene Gerade verletzt die Grenze. Dieser Wert wird auch dann zurückgegeben, wenn der Endpunkt selbst keine Grenze verletzt, auf dem Weg vom Start–zum Endpunkt aber eine Verletzung eines Grenzwertes auftreten würde (z. B. Durchfahren eines Schutzbereiches, gekrümmte Softwarelimits im WKS bei nichtlinearen Transformationen, z. B. Transmit).

Hunderterstelle

100:

Der positive Grenzwert ist verletzt (nur, wenn die Einerstelle 1 oder 2 ist, d. h. bei Softwarelimits und Arbeitsfeldbegrenzung)

100:

Es ist ein NCKSchutzbereich verletzt (nur, wenn die Einerstelle 3 ist).

200:

Der negative Grenzwert ist verletzt (nur, wenn die Einerstelle 1 oder 2 ist, d. h. bei Softwarelimits und Arbeitsfeldbegrenzung)

200:

Es ist ein kanalspezifischer Schutzbereich verletzt (nur, wenn die Einerstelle 3 ist).

Tausenderstelle

1000:

Faktor, mit dem die Nummer der Achse multipliziert wird, die die Grenze verletzt (nur, wenn die Einerstelle 1 oder 2 ist, d. h. bei Softwarelimits und Arbeitsfeldbegrenzung).

Die Zählung der Achsen beginnt bei 1 und bezieht sich bei verletzten Softwarelimits (Einerstelle = 1) auf die Maschinenachsen und bei verletzter Arbeitsfeldbegrenzung (Einerstelle =2) auf die Geometrieachsen.

1000:

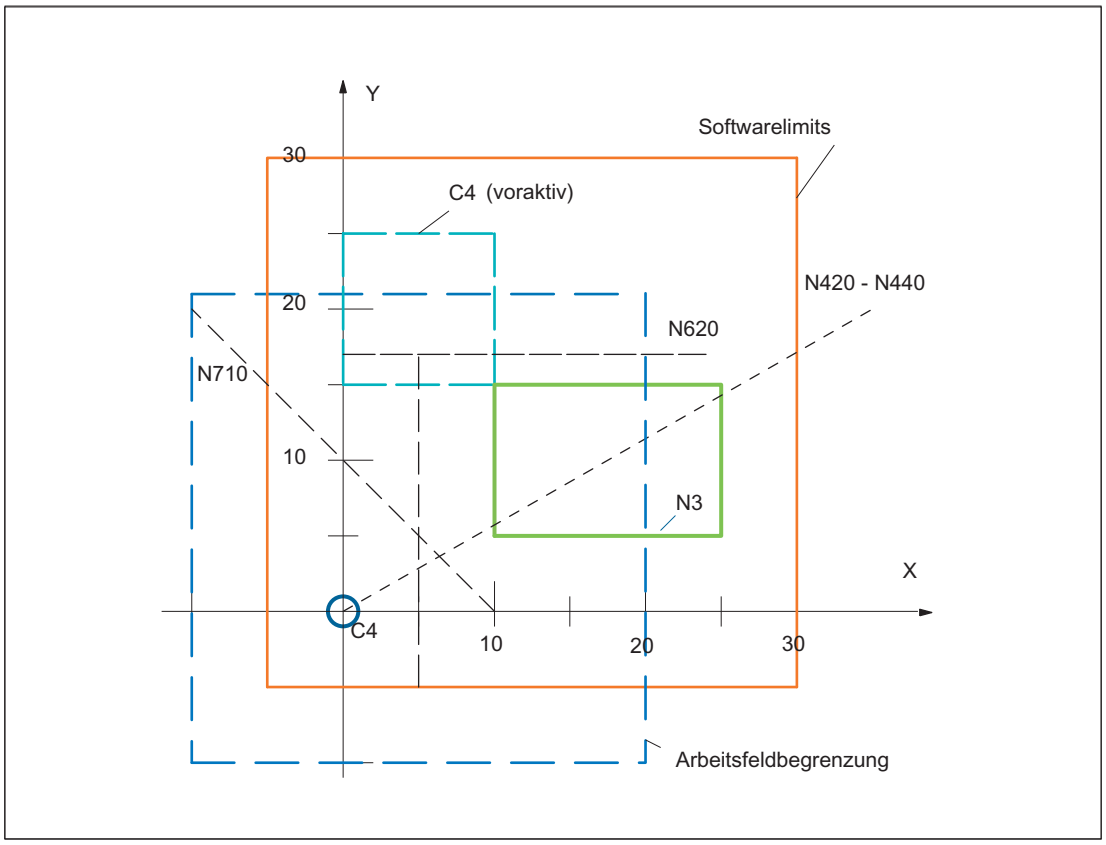
Faktor, mit dem die Nummer des verletzten Schutzbereiches multipliziert wird (nur, wenn die Einerstelle 3 ist).

Sind mehrere Schutzbereiche verletzt, wird in den Hunderter– und Tausenderstellen der Schutzbereich gemeldet, der zur stärksten Einschränkung des vorgegebenen Verfahrweges führt.

_STARTPOS	Anfangswert für Abszisse [0], Ordinate [1] und Applikate [2] im (WKS)
_MOVEDIST	Wegvorgabe inkrementell für Abszisse [0], Ordinate [1] und Applikate [2]
_DLIMIT	[0] - [2]: Mindestabstände die den Geometrieachsen zugeordnet sind. [3]: Mindestabstand, der einer linearen Maschinenachse zugeordnet wird bei einer nicht linearen Transformation, wenn keine Geometrieachse eindeutig zugeordnet werden kann. [4]: Mindestabstand, der einer rotatorischen Maschinenachse zugeordnet wird bei einer nicht linearen Transformation, wenn keine Geometrieachse eindeutig zugeordnet werden kann. Nur bei speziellen Transformationen, wenn SW-Limits überwacht werden sollen.
_MAXDIST	Feld [0] - [2] für Rückgabewert. Inkrementeller Weg in allen drei Geometrieachsen, ohne dass der vorgegebene Mindestabstand von einer Achsgrenze in den beteiligten Maschinenachsen unterschritten wird. Ist der Verfahrweg nicht eingeschränkt, ist der Inhalt dieses Rückgabeparameters gleich dem Inhalt von _MOVDIST.
_BASE_SYS	FALSE oder Parameter nicht angegeben: Bei der Bewertung der Positions- und Längenangaben wird der G-Code der Gruppe 13 (G70, G71, G700, G710; inch/metrisch) ausgewertet. Bei aktivem G70 und metrischem Grundsystem (bzw. aktivem G71 und inch) werden die WKSSystemvariablen $AA_IW[X] und $AA_MW[X]) im Grundsystem geliefert und müssen gegebenenfalls zur Verwendung durch die Funktion CALCPOSI umgerechnet werden. TRUE: Bei der Bewertung der Positions–und Längenangaben wird stets das Grundsystem der Steuerung unabhängig vom Wert des aktiven Gder Gruppe 13 verwendet.
_TESTLIM	Zu überprüfende Begrenzungen (binär codiert): 1: Software Limits überwachen 2: Arbeitsfeldbegrenzungen überwachen 3: Aktivierte Schutzbereiche überwachen 4: Voraktivierte Schutzbereiche überwachen Kombinationen durch addieren der Werte. Default: 15; alle prüfen.

Beispiel

Im Beispiel (siehe Bild) sind in der XSoftwarelimits und Arbeitsfeldbegrenzungen eingezeichnet. Zusätzlich sind drei Schutzbereiche definiert, die beiden kanalspezifischen Schutzbereiche C2 und C4 sowie der NCKSchutzbereich N3. C2 ist ein kreisförmiger aktiver, werkzeugbezogener Schutzbereich mit 2 mm Radius. C4 ist ein quadratischer, voraktivierter und werkstückbezogener Schutzbereich mit 10 mm Seitenlänge und N3 ist ein rechteckiger aktiver Schutzbereich mit 10 mm bzw. 15 mm Seitenlänge. Im folgenden NCwerden zunächst die Schutzbereiche und Arbeitsfeldbegrenzungen wie skizziert definiert, und anschließend wird die Funktion CALCPOSI mit verschiedenen Parametrierungen aufgerufen. Die Ergebnisse der einzelnen Aufrufe von CALCPOSI sind in der Tabelle am Beispielende zusammengefasst.



Programmcode	Kommentar
N10 def real _STARTPOS[3]	
N20 def real _MOVDIST[3]	
N30 def real _DLIMIT[5]	
N40 def real _MAXDIST[3]	
N50 def int _SB	
N60 def int _STATUS	

Programmcode	**Kommentar**
N70 cprotdef(2, true, 0)	; werkzeugbezogener Schutzbereich
N80 g17 g1 x-y0	
N90 g3 i2 x2	
N100 i-x-	
N110 execute(_SB)	
N120 cprotdef(4, false, 0)	; werkstückbezogener Schutzbereich
N130 g17 g1 x0 y15	
N140 x10	
N150 y25	
N160 x0	
N170 y15	
N180 execute(_SB)	
N190 nprotdef(3, false, 0)	; maschinenbezogener Schutzbereich
N200 g17 g1 x10 y5	
N210 x25	
N220 y15	
N230 x10	
N240 y5	
N250 execute(_SB)	
N260 cprot(2,2,0, 0, 0)	; Schutzbereiche aktivieren bzw. voraktivieren
N270 cprot(4,1,0, 0, 0)	
N280 nprot(3,2,0, 0, 0)	
N290 g25 XX=-YY=-	; Arbeitsfeldbegrenzungen definieren
N300 g26 xx= 20 yy= 21	
N310 _STARTPOS[0] = 0.	
N320 _STARTPOS[1] = 0.	
N330 _STARTPOS[2] = 0.	
N340 _MOVDIST[0] = 35.	
N350 _MOVDIST[1] = 20.	
N360 _MOVDIST[2] = 0.	
N370 _DLIMIT[0] = 0.	
N380 _DLIMIT[1] = 0.	
N390 _DLIMIT[2] = 0.	
N400 _DLIMIT[3] = 0.	
N410 _DLIMIT[4] = 0.	
;Verschiede Funktionsaufrufe	; Anderer Startpunkt
N420 _STATUS = calcposi(_STARTPOS,_MOVDIST, _DLIMIT, _MAXDIST)	
N430 _STATUS = calcposi(_STARTPOS,_MOVDIST, _DLIMIT, _MAXDIST,,3)	
N440 _STATUS = calcposi(_STARTPOS,_MOVDIST, _DLIMIT, _MAXDIST,,1)	

Programmcode	**Kommentar**
N450 _STARTPOS[0] = 5.	; Anderes Ziel
N460 _STARTPOS[1] = 17.	
N470 _STARTPOS[2] = 0.	
N480 _MOVDIST[0] = 0.	
N490 _MOVDIST[1] =-.	
N500 _MOVDIST[2] = 0.	
;Verschiede Funktionsaufrufe	
N510 _STATUS = calcposi(_STARTPOS,_MOVDIST, _DLIMIT, _MAXDIST,,14)	
N520 _STATUS = calcposi(_STARTPOS,_MOVDIST, _DLIMIT, _MAXDIST,, 6)	
N530 _DLIMIT[1] = 2.	
N540 _STATUS = calcposi(_STARTPOS,_MOVDIST, _DLIMIT, _MAXDIST,, 6)	
N550 _STARTPOS[0] = 27.	
N560 _STARTPOS[1] = 17.1	
N570 _STARTPOS[2] = 0.	
N580 _MOVDIST[0] =-.	
N590 _MOVDIST[1] = 0.	
N600 _MOVDIST[2] = 0.	
N610 _DLIMIT[3] = 2.	
N620 _STATUS = calcposi(_STARTPOS,_MOVDIST, _DLIMIT, _MAXDIST,, 12)	
N630 _STARTPOS[0] = 0.	
N640 _STARTPOS[1] = 0.	
N650 _STARTPOS[2] = 0.	
N660 _MOVDIST[0] = 0.	
N670 _MOVDIST[1] = 30.	
N680 _MOVDIST[2] = 0.	
N690 trans x10	
N700 arot z45	
N710 _STATUS = calcposi(_STARTPOS,_MOVDIST, _DLIMIT, _MAXDIST)	
N720 M30	

Ergebnisse der Prüfungen im Beispiel:

Satznr. N...	_STATUS	_MAXDIST [0] (= X)	_MAXDIST [1] (= Y)	Bemerkungen
420	3123	8.040	4.594	Schutzbereich SB N3 wird verletzt.
430	1122	20.000	11.429	Keine SB–Überwachung,-Arbeitsfeld-begrenzung wird verletzt.
440	1121	30.000	17.143	Nur noch Überwachung der Softwarelimits aktiv.
510	4213	0.000	0.000	Startpunkt verletzt SB C4
520	0000	0.000	–.000	Voraktivierter SB C4 wird nicht überwacht. Vorgegebener Weg kann vollständig verfahren werden.
540	2222	0.000	–.000	Wegen _DLIMIT[1]=2 wird der Verfahrweg durch Arbeitsfeldbegrenzung eingeschränkt.
620	4223	–.000	0.000	Abstand zu C4 wegen C2 und _DLIMIT[3] insgesamt 4 mm. Abstand C2 –N3 von 0.1 mm führt nicht zur Beschränkung des Verfahrwegs.
710	1221	0.000	21.213	Frame mit Translation und Rotation aktiv. Der zulässige Verfahrweg in _MOVDIST gilt im verschobenen und gedrehten Koordinatensystem (WKS).

Sonderfälle und weitere Details

Alle Wegangaben sind immer im Radiusmaß auch bei einer Planachse mit aktivem G"DIAMON". Kann der Weg einer der beteiligten Achsen nicht vollständig verfahren werden, werden im Rückgabewert _MAXDIST auch die Wege der anderen Achsen entsprechend reduziert, so dass der resultierende Endpunkt auf der vorgegebenen Bahn liegt.

Es ist zulässig, dass für eine oder mehrere der beteiligten Achsen keine Softwarelimits bzw. Arbeitsfeldbegrenzungen oder Schutzbereiche definiert sind. Sämtliche Grenzen werden nur überwacht, wenn die beteiligten Achsen referiert sind. Eventuell beteiligte Rundachsen werden nur überwacht, wenn sie keine Moduloachsen sind.

Die Überwachung der Softwarelimits und der Arbeitsfeldbegrenzungen ist wie im normalen Verfahrbetrieb abhängig von aktiven Einstellungen (Interfacesignale zur Auswahl der Softwarelimits 1 bzw. Softwarelimits 2, GWALIMON/WALIMOF, Settingdaten zur individuellen Aktivierung der Arbeitsfeldgrenzen und zur Festlegung, ob bei der Überwachung der Arbeitsfeldbegrenzungen der Radius des aktiven Werkzeuges berücksichtigt werden soll oder nicht).

Bei bestimmten kinematischen Transformationen (z.B. TRANSMIT) kann die Position der Maschinenachsen aus den Positionen im Werkstückkoordinatensystem (WKS) nicht eindeutig bestimmt werden (Mehrdeutigkeit). Im normalen Verfahrbetrieb ergibt sich die Eindeutigkeit in der Regel aus der Vorgeschichte und der Bedingung, dass einer kontinuierlichen Bewegung im WKS eine kontinuierliche Bewegung der Maschinenachsen entsprechen muss. Bei der Überwachung der Softwarelimits mit Hilfe der Funktion CALCPOSI wird deshalb in derartigen Fällen die gegenwärtige Maschinenposition zur Auflösung der Mehrdeutigkeit herangezogen. Gegebenenfalls muss deshalb vor CALCPOSI ein **STOPRE** programmiert werden, um die Funktion mit gültigen Maschinenachspositionen versorgen zu können.

Es ist nicht sichergestellt, dass zu den Schutzbereichen bei einer Bewegung auf dem vorgegebenen Verfahrweg der in _DLIMIT[3] spezifizierte Abstand überall eingehalten wird. Dafür kann bei Verlängerung des in _MOVDIST zurückgelieferten Endpunktes um diese Distanz kein Schutzbereich verletzt werden. Die Gerade kann in ihrem Verlauf aber beliebig dicht an einem Schutzbereich vorbei führen.

Hinweis

Details zu Arbeitsfeldbegrenzungen finden Sie im
/PG/ Programmierhandbuch Grundlagen,

zu den Softwarelimits in
/FB1/ Funktionshandbuch Grundfunktionen; Achsüberwachungen, Schutzbereiche (A3).

Spezielle Wegbefehle 4

4.1 Codierte Positionen anfahren (CAC, CIC, CDC, CACP, CACN)

Funktion

Über die folgenden Befehle können Sie Linear- und Rundachsen über Positionsnummern auf in Maschinendaten-Tabellen hinterlegte feste Achspositionen verfahren. Diese Art der Programmierung wird als "Anfahren von codierten Positionen" bezeichnet.

Syntax

```
CAC(<n>)
CIC(<n>)
CACP(<n>)
CACN(<n>)
```

Bedeutung

`CAC(<n>)`	Codierte Position von Positionsnummer n anfahren
`CIC(<n>)`	Codierte Position, ausgehend von der aktuellen Positionsnummer, n-Positionsplätze vor (+n) oder zurück (–n) anfahren
`CDC(<n>)`	Codierte Position von Positionsnummer n auf kürzestem Weg anfahren (nur für Rundachsen)
`CACP(<n>)`	Codierte Position von Positionsnummer n in positiver Richtung anfahren (nur für Rundachsen)
`CACN(<n>)`	Codierte Position von Positionsnummer n in negativer Richtung anfahren (nur für Rundachsen)
`<n>`	Positionsnummer innerhalb der Maschinendaten-Tabelle Wertebereich: 0, 1, … (max. Anzahl Tabellenplätze - 1)

Beispiel: Anfahren von codierten Positionen einer Positionierachse

```
Programmiercode                 Kommentar
N10 FA[B]=300                   ; Vorschub für Positionierachse B
N20 POS[B]=CAC(10)              ; Codierte Position von Positionsnummer 10 anfahren
N30 POS[B]=CIC(-4)              ; Codierte Position von "aktuelle Positionsnummer" -
                                  4 anfahren
```

Literatur

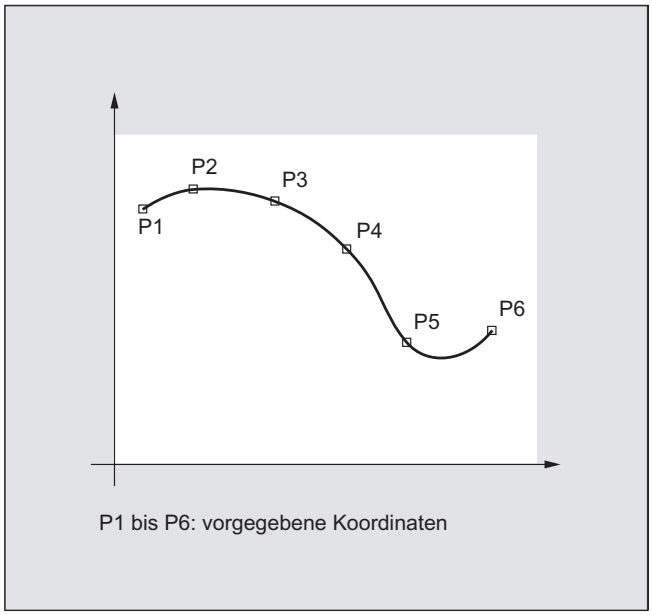
- Funktionshandbuch Erweiterungsfunktionen; Teilungsachsen (T1)
- Funktionshandbuch Synchronaktionen

4.2 Spline-Interpolation (ASPLINE, BSPLINE, CSPLINE, BAUTO, BNAT, BTAN, EAUTO, ENAT, ETAN, PW, SD, PL)

Funktion

Beliebig gekrümmte Konturen an Werkstücken können nicht analytisch exakt beschrieben werden. Derartige Konturen werden daher durch eine begrenzte Anzahl von Stützpunkten, z. B. beim Digitalisieren von Oberflächen, angenähert. Zur Erzeugung der digitalisierten Oberfläche an einem Werkstück müssen die Stützpunkte zu einer Konturbeschreibung verbunden werden. Dies ermöglicht die Spline-Interpolation.

Ein Spline definiert eine Kurve, die aus Polynomen 2. oder 3. Grades zusammengesetzt wird. Die Eigenschaften an den Stützpunkten eines Splines sind **abhängig vom verwendeten Spline-Typ** definierbar.



Folgende Spline-Typen stehen bei SINUMERIK solution line zur Verfügung:

- A-Spline
- B-Spline
- C-Spline

Syntax

Allgemein:

```
ASPLINE X... Y... Z... A... B... C...
BSPLINE X... Y... Z... A... B... C...
CSPLINE X... Y... Z... A... B... C...
```

Bei B-Spline zusätzlich programmierbar:

```
PW=<n>
SD=2
PL=<Wert>
```

Bei A- und C-Spline zusätzlich programmierbar:

`BAUTO` / `BNAT` / `BTAN`

`EAUTO` / `ENAT` / `ETAN`

Bedeutung

Spline-Interpolationstyp:

`ASPLINE`	Befehl zum Einschalten der A-Spline-Interpolation
`BSPLINE`	Befehl zum Einschalten der B-Spline-Interpolation
`CSPLINE`	Befehl zum Einschalten der C-Spline-Interpolation
	Die Befehle `ASPLINE`, `BSPLINE` und `CSPLINE` sind modal wirksam und gehören zur Gruppe der Wegbefehle.

Stützpunkte bzw. Kontrollpunkte:

`X... Y... Z...` `A... B... C...`	Positionen in kartesischen Koordinaten

Punktgewicht (nur B-Spline):

`PW`	Mit dem Befehl `PW` ist für jeden Stützpunkt die Programmierung eines sogenannten "Punktgewichts" möglich.		
`<n>`	"Punktgewicht"		
	Wertebereich:	0 ≤ n ≤ 3	
	Schrittweite:	0.0001	
	Wirkung:	n > 1	Die Kurve wird vom Kontrollpunkt stärker angezogen.
		n < 1	Die Kurve wird vom Kontrollpunkt weniger stark angezogen.

Spline-Grad (nur B-Spline):

`SD`	Standardmäßig wird ein Polygon 3. Grades verwendet. Durch Programmierung von `SD=2` kann aber auch ein Polygon 2. Grades verwendet werden.

Knotenabstand (nur B-Spline):

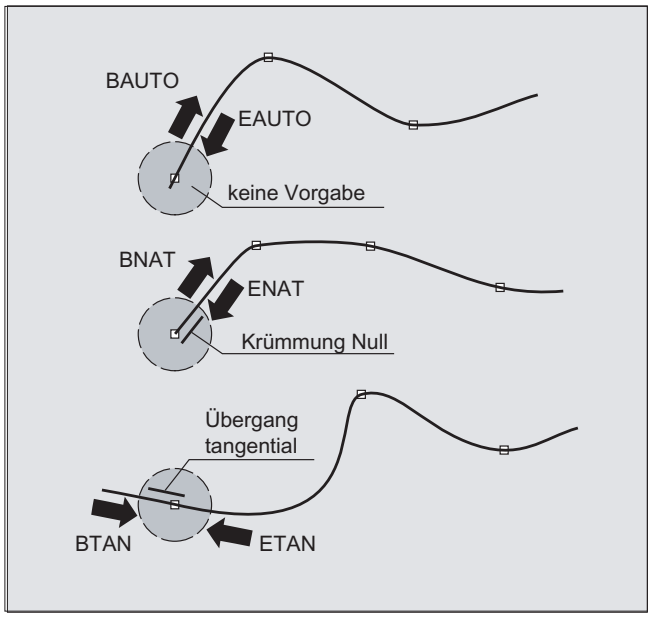
`PL`	Die Knotenabstände werden intern geeignet berechnet. Die Steuerung kann aber auch vorgegebene Knotenabstände verarbeiten, die mit dem Befehl `PL` als sog. Parameter-Intervall-Länge angegeben werden.
`<Wert>`	Parameter-Intervall-Länge Wertebereich: wie Wegmaß

Übergangsverhalten am Beginn der Spline-Kurve (nur A- oder C-Spline):

`BAUTO`	Keine Vorgabe für das Übergangsverhalten. Der Anfang ergibt sich aus der Lage des ersten Punkts.
`BNAT`	Krümmung Null
`BTAN`	Tangentialer Übergang zum Satz vorher (Löschstellung)

Übergangsverhalten am Ende der Spline-Kurve (nur A- oder C-Spline):

`EAUTO`	Keine Vorgabe für das Übergangsverhalten. Das Ende ergibt sich aus der Lage des letzten Punkts.
`ENAT`	Krümmung Null
`ETAN`	Tangentialer Übergang zum Satz vorher (Löschstellung)



Hinweis

Das programmierbare Übergangsverhalten hat keinen Einfluss auf den B-Spline. Der B-Spline ist in Start- und Endpunkt immer tangential zum Kontrollpolygon.

Randbedingungen

- Die Werkzeugradiuskorrektur ist einsetzbar.
- Kollisionsüberwachung erfolgt in der Projektion auf die Ebene.

Beispiele

Beispiel 1: B-Spline

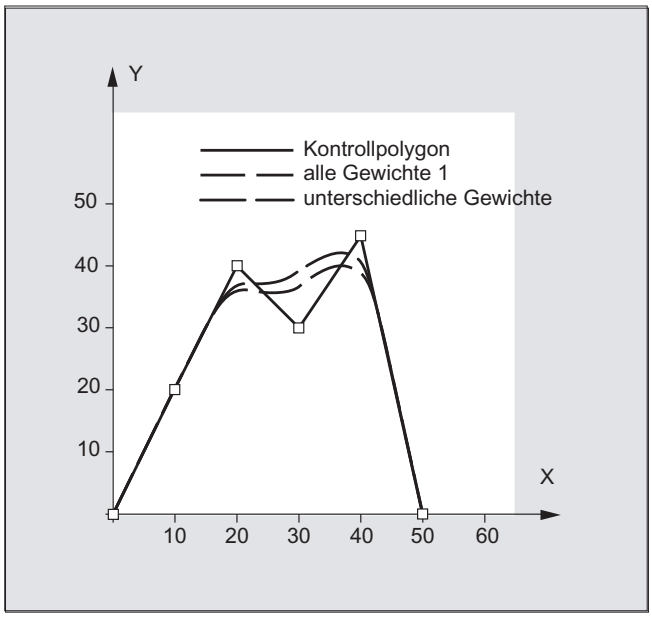
Programmcode 1 (alle Gewichte 1)

```
N10 G1 X0 Y0 F300 G64
N20 BSPLINE
N30 X10 Y20
N40 X20 Y40
N50 X30 Y30
N60 X40 Y45
N70 X50 Y0
```

Programmcode 2 (unterschiedliche Gewichte)

```
N10 G1 X0 Y0 F300 G64
N20 BSPLINE
N30 X10 Y20 PW=2
N40 X20 Y40
N50 X30 Y30 PW=0.5
N60 X40 Y45
N70 X50 Y0
```

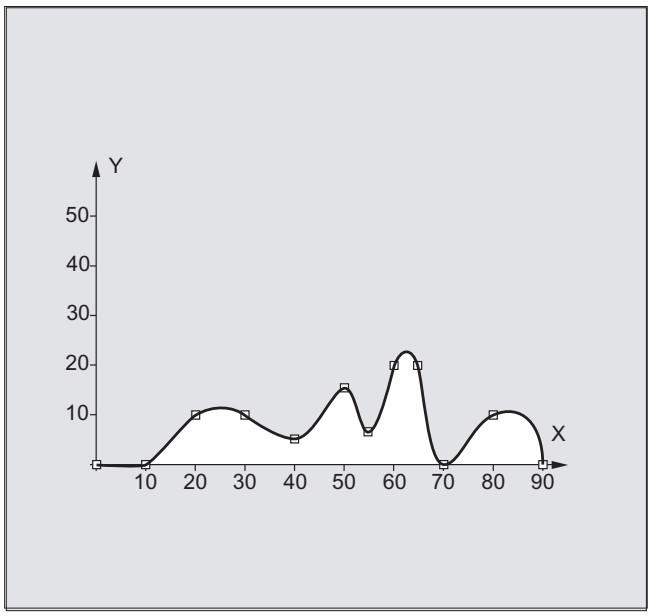
Programmcode 3 (Kontrollpolygon)	**Kommentar**
N10 G1 X0 Y0 F300 G64	
N20	; entfällt
N30 X10 Y20	
N40 X20 Y40	
N50 X30 Y30	
N60 X40 Y45	
N70 X50 Y0	



Beispiel 2: C-Spline, am Anfang und am Ende Krümmung Null

Programmcode

```
N10 G1 X0 Y0 F300
N15 X10
N20 BNAT ENAT
N30 CSPLINE X20 Y10
N40 X30
N50 X40 Y5
N60 X50 Y15
N70 X55 Y7
N80 X60 Y20
N90 X65 Y20
N100 X70 Y0
N110 X80 Y10
N120 X90 Y0
N130 M30
```



Beispiel 3: Spline-Interpolation (A-Spline) und Koordinatentransformation (ROT)

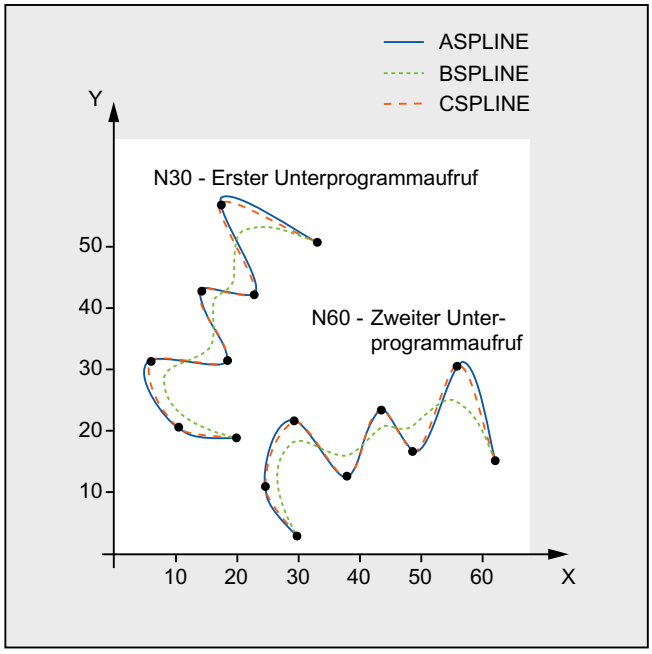
Hauptprogramm:

Programmcode	Kommentar
N10 G00 X20 Y18 F300 G64	; Startpunkt anfahren.
N20 ASPLINE	; Interpolationstyp A-Spline aktivieren.
N30 KONTUR	; Erster Aufruf des Unterprogramms.
N40 ROT Z-45	; Koordinatentransformation: Drehung des WKS um -45° um die Z-Achse.
N50 G00 X20 Y18	; Konturstartpunkt anfahren.
N60 KONTUR	; Zweiter Aufruf des Unterprogramms.
N70 M30	; Programmende

Unterprogramm "Kontur" (enthält die Stützpunkt-Koordinaten):

Programmcode
N10 X20 Y18
N20 X10 Y21
N30 X6 Y31
N40 X18 Y31
N50 X13 Y43
N60 X22 Y42
N70 X16 Y58
N80 X33 Y51
N90 M1

In der folgenden Abbildung sind neben der Spline-Kurve, die aus dem Programmbeispiel resultiert (ASPLINE), auch die Spline-Kurven enthalten, die sich bei Aktivierung einer B- oder C-Spline-Interpolation ergeben hätten (BSPLINE, CSPLINE):



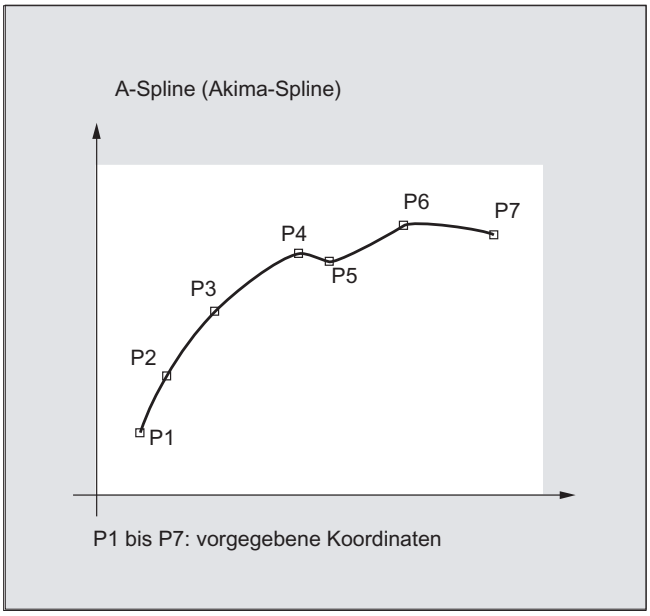
Weitere Informationen

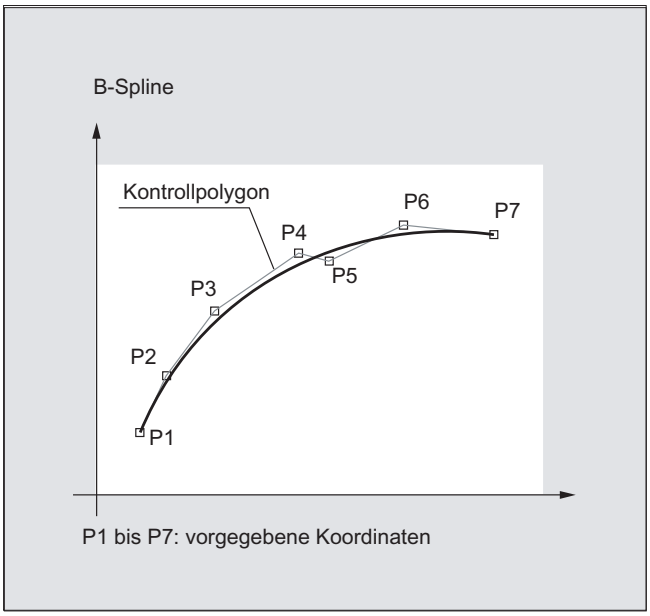
Vorteile der Spline-Interpolation

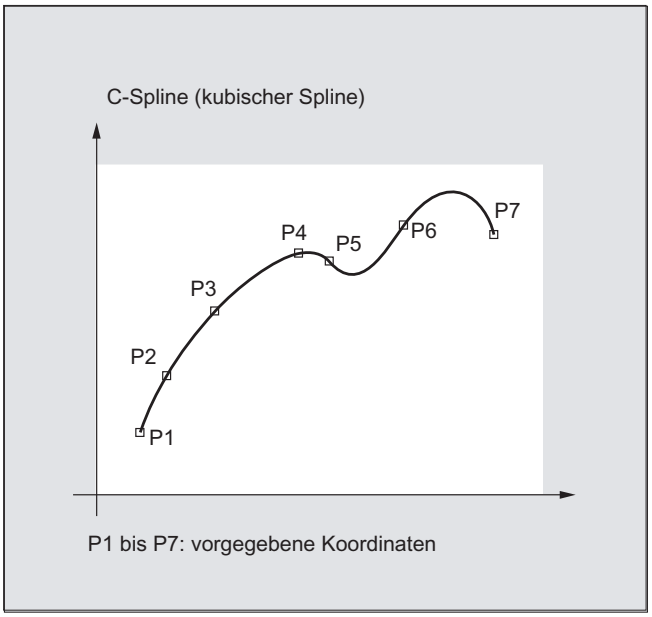
Durch Verwendung der Spline-Interpolation lassen sich, im Gegensatz zur Verwendung von Geradensätzen `G01`, folgende Vorteile erzielen:

- Reduzierung der Anzahl von benötigten Teileprogrammsätzen zur Beschreibung der Kontur
- Weicher, mechanikschonender Kurvenverlauf beim Übergange zwischen den Teileprogrammsätzen

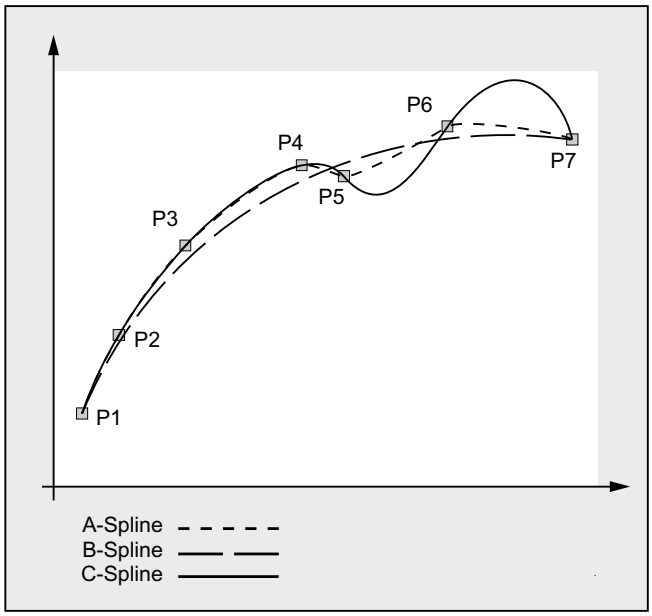
Eigenschaften und Anwendung der verschiedenen Spline-Typen

Spline-Typ	Eigenschaften und Anwendung
A-Spline	 **Eigenschaften:** • Verläuft exakt durch die vorgegebenen Stützpunkte. • Der Kurvenverlauf ist tangenten- aber nicht krümmungsstetig. • Erzeugt kaum ungewollte Schwingungen. • Der Einflussbereich von Stützpunktänderungen ist lokal, d. h. Veränderung eines Stützpunkts wirkt sich nur auf bis zu max. 6 benachbarte Stützpunkte aus. **Anwendung:** Der A-Spline eignet sich vor allem für die Interpolation von Kurvenverläufen mit großen Steigungsänderungen (z. B. treppenförmige Kurvenverläufe).

Spline-Typ	Eigenschaften und Anwendung
B-Spline	 **Eigenschaften:** • Verläuft nicht durch die vorgegebenen Stützpunkte, sondern nur in deren Nähe. Die Kurve wird durch die Stützpunkte angezogen. Durch Gewichtung der Stützpunkte mit einem Faktor, kann der Kurvenverlauf zusätzlich beeinflusst werden. • Der Kurvenverlauf ist tangenten- und krümmungsstetig. • Erzeugt keine ungewollten Schwingungen. • Der Einflussbereich von Stützpunktänderungen ist lokal, d. h. Veränderung eines Stützpunkts wirkt sich nur auf bis zu max. 6 benachbarte Stützpunkte aus. **Anwendung:** Der B-Spline ist primär als Schnittstelle zu CAD-Systemen gedacht.

Spline-Typ	Eigenschaften und Anwendung
C-Spline	 **Eigenschaften:** • Verläuft exakt durch die vorgegebenen Stützpunkte. • Der Kurvenverlauf ist tangenten- und krümmungsstetig. • Erzeugt häufig ungewollten Schwingungen, besonders an Stellen mit großen Steigungsänderungen. • Der Einflussbereich von Stützpunktänderungen ist global, d. h. Veränderung eines Stützpunkts wirkt sich auf den gesamten Kurvenverlauf aus. **Anwendung:** Der C-Spline kann dann gut eingesetzt werden, wenn die Stützpunkte auf einer analytisch bekannten Kurve liegen (Kreis, Parabel, Hyperbel)

Gegenüberstellung der drei Spline-Typen bei gleichen Stützpunkten



Mindestanzahl an Spline-Sätzen

Die G-Codes ASPLINE, BSPLINE und CSPLINE verbinden Satzendpunkte mit Splines. Dazu müssen im Vorlauf eine Reihe von Sätzen (Endpunkte) gleichzeitig berechnet werden. Die Größe des Puffers für die Berechnung beträgt standardmäßig 10 Sätze. Nicht jede Satzinformation ist ein Spline-Endpunkt. Die Steuerung benötigt jedoch von 10 Sätzen eine bestimmte Anzahl an Spline-Endpunkt-Sätzen:

Spline-Typ	Mindestanzahl an Spline-Sätzen
A-Spline:	Von je 10 Sätzen müssen mindestens **4** Spline-Sätze sein. Kommentarsätze und Parameterrechnungen zählen hierbei nicht.
B-Spline:	Von je 10 Sätzen müssen mindestens **6** Spline-Sätze sein. Kommentarsätze und Parameterrechnungen zählen hierbei nicht.
C-Spline:	Die benötigte Mindestanzahl an Spline-Sätzen ergibt sich aus folgender Summe: Wert aus MD20160 $MC_CUBIC_SPLINE_BLOCKS + 1 Im MD20160 wird die Anzahl der Punkte eingetragen, über die der Spline-Abschnitt berechnet wird. Die Standardeinstellung beträgt 8. Von je 10 Sätzen müssen daher im Standardfall mindestens **9** Spline-Sätze sein.

Hinweis

Bei Unterschreitung des tolerierbaren Werts wird ein Alarm ausgegeben, ebenso, wenn eine am Spline beteiligte Achse als Positionierachse programmiert wird.

Zusammenfassung kurzer Spline-Sätze

Bei der Spline-Interpolation können kurze Spline-Sätze entstehen, die zu einer unnötigen Reduzierung der Bahngeschwindigkeit führen. Mit der Funktion "Zusammenfassung kurzer Spline-Sätze" können diese Sätze so zusammengefasst werden, dass die resultierende Satzlänge ausreichend groß ist und nicht zu einer Verringerung der Bahngeschwindigkeit führt.

Die Funktion wird aktiviert über das kanalspezifische Maschinendatum:

MD20488 $MC_SPLINE_MODE (Einstellung für Spline-Interpolation)

Literatur:
Funktionshandbuch Grundfunktionen; Bahnsteuerbetrieb, Genauhalt, LookAhead (B1), Kapitel: Zusammenfassung kurzer Spline-Sätze

4.3 Spline-Verbund (SPLINEPATH)

Funktion

Die im Spline-Verbund zu interpolierenden Achsen werden mit dem Befehl `SPLINEPATH` ausgewählt. Bis zu acht Bahnachsen sind bei der Spline-Interpolation möglich.

Hinweis

Wird `SPLINEPATH` nicht explizit programmiert, so werden die ersten drei Achsen des Kanals als Spline-Verbund verfahren.

Syntax

Die Festlegung des Spline-Verbundes erfolgt in einem gesonderten Satz:

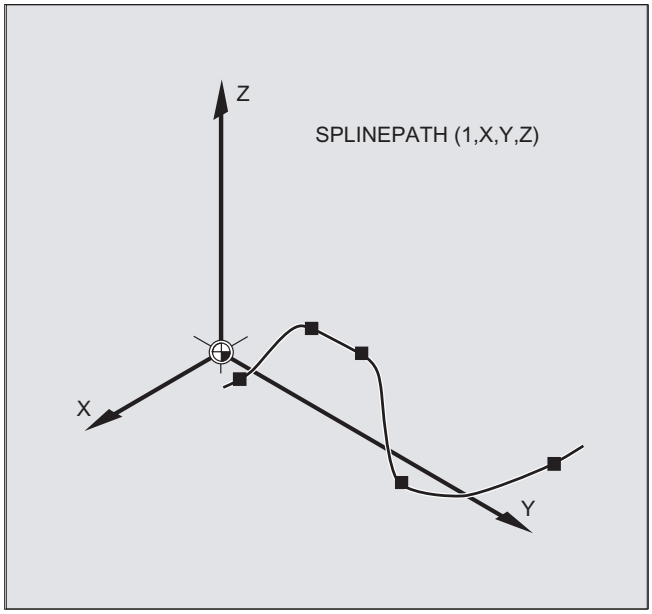
```
SPLINEPATH(n,X,Y,Z,…)
```

Bedeutung

`SPLINEPATH`	Befehl zur Festlegung eines Spline-Verbundes
`n`	=1 (fester Wert)
`X,Y,Z,…`	Bezeichner der im Spline-Verbund zu interpolierenden Bahnachsen

Beispiel: Spline-Verbund mit drei Bahnachsen

Programmcode	**Kommentar**
N10 G1 X10 Y20 Z30 A40 B50 F350	
N11 SPLINEPATH(1,X,Y,Z)	; Spline-Verbund
N13 CSPLINE BAUTO EAUTO X20 Y30 Z40 A50 B60	; C-Spline
N14 X30 Y40 Z50 A60 B70	; Stützpunkte
…	
N100 G1 X… Y…	; Abwahl Spline-Interpolation



4.4 NC-Satz-Kompression (COMPON, COMPCURV, COMPCAD, COMPOF)

Funktion

CAD/CAM-Systeme liefern in der Regel Linearsätze, welche die parametrierte Genauigkeit einhalten. Dies führt bei komplexen Konturen zu einer erheblichen Datenmenge und zu eventuell kurzen Bahnabschnitten. Diese kurzen Bahnabschnitte begrenzen die Abarbeitungsgeschwindigkeit.

Durch die Anwendung einer Kompressor-Funktion erfolgt eine Annäherung an die durch Linear-Sätze vorgegebene Kontur durch Polynom-Sätze. Dadurch ergeben sich folgende Vorteile:

- Reduzierung der Anzahl von benötigten Teileprogrammsätzen zur Beschreibung der Werkstückkontur
- Stetige Satzübergänge
- Erhöhung der maximal möglichen Bahngeschwindigkeiten

Folgende Kompressor-Funktionen stehen zur Verfügung:

- COMPON

  Die Satzübergänge sind nur stetig in der **Geschwindigkeit**, während die Beschleunigung der beteiligten Achsen an den Satzübergängen Sprünge machen kann.

- COMPCURV

  Die Satzübergänge sind **beschleunigungsstetig**. Damit ist sowohl ein glatter Verlauf der Geschwindigkeit als auch der Beschleunigung aller Achsen an den Satzübergängen gewährleistet.

- COMPCAD

  Rechenzeit- und speicherplatzintensive Kompression, die bezüglich Oberflächengüte und Geschwindigkeit optimiert. COMPCAD sollte nur eingesetzt werden, wenn Maßnahmen zur Oberflächenverbesserung vom CAD/CAM-Programm nicht vorab geleistet werden können.

Beendet wird die Kompressor-Funktion mit `COMPOF`.

Syntax

```
COMPON
COMPCURV
COMPCAD
COMPOF
```

Bedeutung

COMPON:	Befehl zum Einschalten der Kompressor-Funktion COMPON.
	Wirksamkeit: modal
COMPCURV:	Befehl zum Einschalten der Kompressor-Funktion COMPCURV.
	Wirksamkeit: modal
COMPCAD:	Befehl zum Einschalten der Kompressor-Funktion COMPCAD.
	Wirksamkeit: modal
COMPOF:	Befehl zum Ausschalten der aktuell aktiven Kompressor-Funktion.

Hinweis

Zur zusätzlichen Verbesserung der Oberflächengüte kann die Überschleiffunktion `G642` und die Ruckbegrenzung `SOFT` verwendet werden. Diese Befehle sind am Programmanfang zu schreiben.

Randbedingungen

- Die NC-Satz-Kompression wird i. d. R. für Linearsätze (`G1`) durchgeführt.
- Es werden nur Sätze komprimiert, die einer einfachen Syntax genügen:

  ```
  N... G1X... Y... Z... F... ;Kommentar
  ```

  Alle anderen Sätze werden unverändert abgearbeitet (ohne Kompression).
- Bewegungssätze mit erweiterten Adressen wie `C=100` oder `A=AC(100)` werden auch komprimiert.
- Positionswerte müssen nicht direkt programmiert werden, sondern können auch indirekt über Parameterzuweisungen angegeben werden, z. B. `X=R1*(R2+R3)`.
- Wenn die Option "Orientierungstransformation" zur Verfügung steht, dann können auch NC-Sätze komprimiert werden, in denen die Werkzeugorientierung (und ggf. auch die Werkzeugdrehung) mittels Richtungsvektoren programmiert ist (siehe " Komprimierung der Orientierung (Seite 353) ").
- Der Kompressionsvorgang wird unterbrochen durch jede andere NC-Anweisung, z. B. eine Hilfsfunktionsausgabe.

Beispiele

Beispiel 1: COMPON

```
Programmcode                          Kommentar
N10 COMPON                            ; Kompressor-Funktion COMPON ein.
N11 G1 X0.37 Y2.9 F600                ; G1 vor Endpunkt und Vorschub.
N12 X16.87 Y-.698
N13 X16.865 Y-.72
N14 X16.91 Y-.799
…
N1037 COMPOF                          ; Kompressor-Funktion aus.
…
```

Beispiel 2: COMPCAD

```
Programmcode                          Kommentar
G00 X30 Y6 Z40
G1 F10000 G642                        ; Überschleiffunktion G642 ein.
SOFT                                  ; Ruckbegrenzung SOFT ein.
COMPCAD                               ; Kompressor-Funktion COMPCAD ein.
STOPFIFO
N24050 Z32.499
N24051 X41.365 Z32.500
N24052 X43.115 Z32.497
N24053 X43.365 Z32.477
N24054 X43.556 Z32.449
N24055 X43.818 Z32.387
N24056 X44.076 Z32.300
…
COMPOF                                ; Kompressor-Funktion aus.
G00 Z50
M30
```

Literatur

Funktionshandbuch Grundfunktionen; Bahnsteuerbetrieb, Genauhalt, LookAhead (B1), Kapitel: "NC-Satz-Kompression"

4.5 Polynom-Interpolation (POLY, POLYPATH, PO, PL)

Funktion

Im eigentlichen Sinn handelt es sich bei der Polynom-Interpolation (`POLY`) nicht um eine Spline-Interpolationsart. Sie ist in erster Linie als Schnittstelle für die Programmierung extern erzeugter Spline-Kurven gedacht. Hierbei können die Spline-Abschnitte direkt programmiert werden.

Diese Interpolationsart entlastet die NC von der Berechnung der Polynom-Koeffizienten. Sie ist dann optimal einsetzbar, wenn die Koeffizienten direkt von einem CAD-System oder Post-Prozessor kommen.

Syntax

Polynom 3. Grades:

```
POLY PO[X]=(xe,a2,a3) PO[Y]=(ye,b2,b3) PO[Z]=(ze,c2,c3) PL=n
```

Polynome 5.Grades und neue Polynomsyntax:

```
POLY X=PO(xe,a2,a3,a4,a5) Y=PO(ye,b2,b3,b4,b5) Z=PO(ze,c2,c3,c4,c5)
PL=n
POLYPATH("AXES","VECT")
```

Hinweis

Die Summe der in einem NC-Satz programmierten Polynom-Koeffizienten und Achsen darf die maximal erlaubte Achsanzahl pro Satz nicht überschreiten.

Bedeutung

`POLY` :	Einschalten der Polynom-Interpolation mit einem Satz mit POLY.
`POLYPATH` :	Polynom-Interpolation selektierbar für die beiden Achsgruppen AXIS oder VECT
`PO[Achsbezeichner/Variable]` :	Endpunkte und Polynom-Koeffizienten
`X, Y, Z` :	Achsbezeichner
`xe, ye, ze` :	Angabe der Endposition für die jeweilige Achse; Wertebereich wie Wegmaß

`a2, a3, a4, a5` :	Die Koeffizienten a_2, a_3, a_4, und a_5 werden mit ihrem Wert geschrieben; Wertebereich wie Wegmaß. Der jeweils letzte Koeffizient kann entfallen, wenn er den Wert Null hat.
`PL` :	Länge des Parameterintervalls, auf dem die Polynome definiert sind (Definitionsbereich der Funktion f(p)). Das Intervall beginnt immer bei 0, p kann Werte von 0 bis PL annehmen. Theoretischer Wertebereich für PL: 0,0001 … 99 999,9999 **Hinweis:** Der PL-Wert gilt für den Satz, in dem er steht. Ist kein PL programmiert, wirkt PL=1.

Ein-/Ausschalten der Polynom-Interpolation

Die Polynom-Interpolation wird im Teileprogramm durch den G-Befehl `POLY` eingeschaltet.

Der G-Befehl `POLY` gehört zusammen mit `G0`, `G1`, `G2`, `G3`, `ASPLINE`, `BSPLINE` und `CSPLINE` zur 1. G-Gruppe.

Achsen, die nur mit Namen und Endpunkt programmiert sind (z.B. `X10`), werden linear verfahren. Sind alle Achsen eines NC-Satzes so programmiert, verhält sich die Steuerung wie bei G1.

Die Polynom-Interpolation wird durch die Programmierung eines anderen Befehls der 1. G-Gruppe (z. B. `G0`, `G1`) implizit wieder ausgeschaltet.

Polynomkoeffizient

Der `PO`-Wert `(PO[]=)` bzw. `...=PO(...)` gibt alle Polynom-Koeffizienten für eine Achse an. Entsprechend dem Grad des Polynoms werden mehrere Werte durch Kommata getrennt angegeben. Innerhalb eines Satzes sind unterschiedliche Polynomgrade für verschiedene Achsen möglich.

Unterprogramm POLYPATH

Mit `POLYPATH(...)` kann die Polynom-Interpolation selektiv für bestimmte Achsgruppen freigegeben werden:

Nur Bahnachsen und Zusatzachsen:	`POLYPATH("AXES")`
Nur Orientierungsachsen: (beim Verfahren mit Orientierungs-Transformation)	`POLYPATH("VECT")`

Die jeweils nicht freigegebenen Achsen werden linear verfahren.

Standardmäßig ist die Polynom-Interpolation für beide Achsgruppen freigegeben.

Durch Programmierung ohne Parameter `POLYPATH( )` wird die Polynom-Interpolation für alle Achsen deaktiviert.

Beispiel

```
Programmcode                                                    Kommentar
N10 G1 X… Y… Z… F600
N11 POLY PO[X]=(1,2.5,0.7) PO[Y]=(0.3,1,3.2) PL=1.5             ; Polynom-Interpolation ein
N12 PO[X]=(0,2.5,1.7) PO[Y]=(2.3,1.7) PL=3
...
N20 M8 H126 …
N25 X70 PO[Y]=(9.3,1,7.67) PL=5                                 ; gemischte Angaben für die Achsen
N27 PO[X]=(10,2.5) PO[Y]=(2.3)                                  ; kein PL programmiert; es wirkt PL=1
N30 G1 X… Y… Z.                                                 ; Polynom-Interpolation aus
…
```

Beispiel: Neue Polynomsyntax

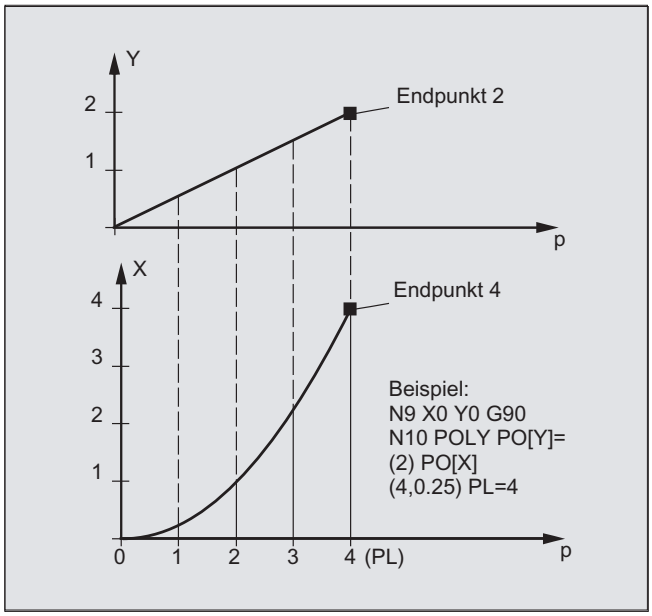
Weiterhin gültige Polynomsyntax	Neue Polynomsyntax
PO[Achsbezeichner]=(.. , ..)	Achsbezeichner=PO(.. , ..)
PO[PHI]=(.. , ..)	PHI=PO(.. , ..)
PO[PSI]=(.. , ..)	PSI=PO(.. , ..)
PO[THT]=(.. , ..)	THT=PO(.. , ..)
PO[]=(.. , ..)	PO(.. , ..)
PO[variable]=IC(.. , ..)	variable=PO IC(.. , ..)

Beispiel: Kurve in der X/Y-Ebene

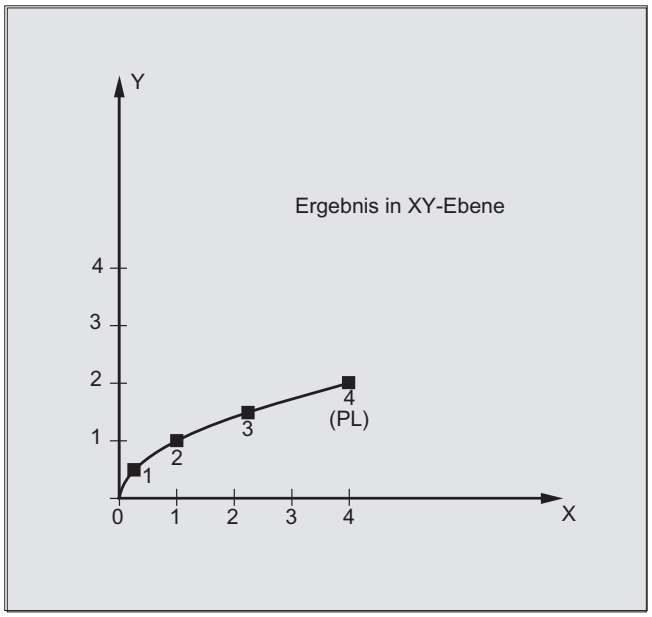
Programmierung

```
Programmcode
N9 X0 Y0 G90 F100
N10 POLY PO[Y]=(2) PO[X]=(4,0.25) PL=4
```

Verlauf der Kurven X(p) und Y(p)



Verlauf der Kurve in der XY-Ebene



Beschreibung

Die allgemeine Form der Polynom-Funktion lautet:

$f(p)= a_0 + a_1p + a_2p^2 +. . . + a_np^n$

mit: a_n: konstante Koeffizienten
p: Parameter

In der Steuerung können maximal Polynome 5. Grades programmiert werden:

$f(p)= a_0 + a_1p + a_2p^2 + a_3p^3 + a_4p^4 + a_5p^5$

Durch Belegen der Koeffizienten mit konkreten Werten sind verschieden Kurvenverläufe, wie Geraden, Parabeln und Potenzfunktionen, erzeugbar.

Eine Gerade wird erzeugt durch $a_2 = a_3 = a_4 = a_5 = 0$:

$f(p)= a_0 + a_1p$

Weiter gilt:

a_0: Achsposition am Ende des vorangehenden Satzes

$p = PL$

$a_1 = (x_E - a_0 - a_2*p^2 - a_3*p^3) / p$

Es ist möglich Polynome zu programmieren, **ohne** dass die Polynom-Interpolation durch den G-Befehl `POLY` aktiviert wurde. In diesem Fall werden nicht die programmierten Polynome interpoliert, sondern die programmierten Endpunkte der Achsen linear angefahren (`G1`). Erst nach expliziter Aktivierung der Polynom-Interpolation im Teileprogramm (`POLY`) werden die programmierten Polynome auch als solche verfahren.

Besonderheit: Nenner-Polynom

Für die Geometrieachsen kann mit `PO[]=(…)` ohne Angabe eines Achsnamens auch ein gemeinsames Nenner-Polynom programmiert werden, d. h. die Bewegung der Geometrieachsen wird als Quotient zweier Polynome interpoliert.

Damit lassen sich z. B. Kegelschnitte (Kreis, Ellipse, Parabel, Hyperbel) exakt darstellen.

Beispiel:

Programmcode	Kommentar
`POLY G90 X10 Y0 F100`	`; Geometrieachsen verfahren linear auf die Position X10 Y0.`
`PO[X]=(0,-10) PO[Y]=(10) PO[]=(2,1)`	`; Geometrieachsen verfahren im Viertelkreis auf X0 Y10.`

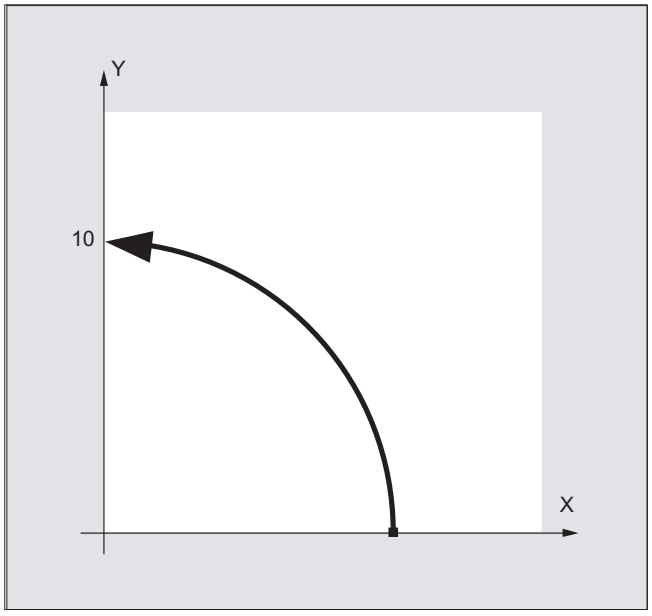
Der konstante Koeffizient (a_0) des Nenner-Polynoms wird stets mit 1 angenommen. Der programmierte Endpunkt ist unabhängig von `G90` / `G91`.

Aus den programmierten Werten berechnen sich X(p) und Y(p) zu:

$X(p) = (10 - 10 * p^2) / (1 + p^2)$
$Y(p) = 20 * p / (1 + p^2)$
mit $0 \leq p \leq 1$

Aufgrund der programmierten Anfangspunkte, Endpunkte, Koeffizient a_2 und PL=1 ergeben sich folgende Zwischenergebnisse:

Zähler (X) = $10 + 0 * p - 10 * p^2$
Zähler (Y) = $0 + 20 * p + 0 * p^2$
Nenner = $1 + p^2$



Bei eingeschalteter Polynom-Interpolation wird die Programmierung eines Nenner-Polynoms mit Nullstellen innerhalb des Intervalls `[0,PL]` mit einem Alarm abgelehnt. Auf die Bewegung von Zusatzachsen hat das Nenner-Polynom keinen Einfluss.

Hinweis

Eine Werkzeugradiuskorrektur ist bei der Polynom-Interpolation mit `G41`, `G42` einschaltbar und wie für Geraden- oder Kreisinterpolation verwendbar.

4.6 Einstellbarer Bahnbezug (SPATH, UPATH)

Funktion

Während Polynominterpolation können vom Anwender zwei unterschiedliche Beziehungen zwischen den geschwindigkeitsbestimmenden FGROUP-Achsen und den übrigen Bahnachsen gewünscht sein: Letztere sollen entweder synchron zum Bahnweg S oder synchron zum Kurvenparameter U der FGROUP-Achsen geführt werden.

Beide Arten der Bahninterpolation werden in unterschiedlichen Applikationen gebraucht und können durch die beiden in der 45. G-Code-Gruppe enthaltenen modal wirksamen Sprachbefehle SPATH und UPATH eingestellt/umgeschaltet werden.

Syntax

```
SPATH
UPATH
```

Bedeutung

SPATH:	Bahnbezug für FGROUP-Achsen ist Bogenlänge
UPATH:	Bahnbezug für FGROUP-Achsen ist Kurvenparameter

Hinweis

UPATH und SPATH bestimmen auch den Zusammenhang des F-Wort-Polynoms (FPOLY, FCUB, FLIN) mit der Bahnbewegung.

Randbedingungen

Der eingestellte Bahnbezug hat keine Bedeutung:

- bei Linear- und Kreisinterpolation
- in Gewindesätzen
- wenn alle Bahnachsen in FGROUP enthalten sind.

Beispiele

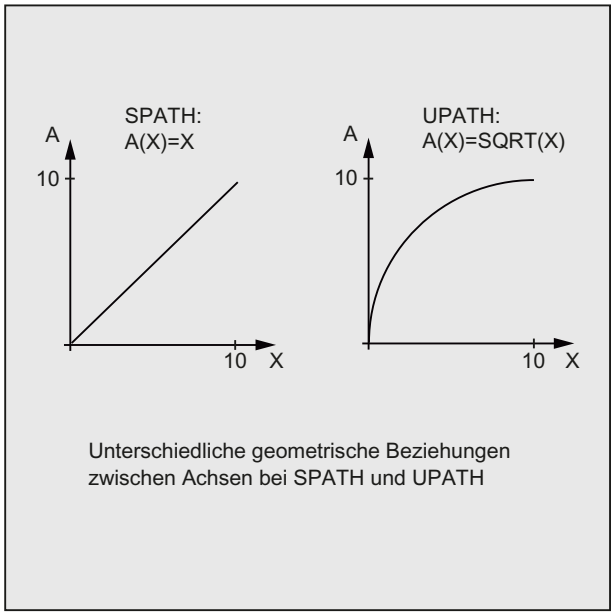
Beispiel 1:

Im nachfolgenden Beispiel wird ein Quadrat mit 20 mm Kantenlänge mit G643 überschliffen. Die maximalen Abweichungen von der exakten Kontur werden dabei durch das achsspezifische Maschinendatum MD33100 $MA_COMPRESS_POS_TOL[<n>] für jede Achse festgelegt.

Programmcode	Kommentar
N10 G1 X… Y… Z… F500	
N20 G643	; Satzinternes Überschleifen mit G643
N30 XO Y0	
N40 X20 Y0	; Kantenlänge (mm) für die Achsen
N50 X20 Y20	
N60 X0 Y20	
N70 X0 Y0	
N100 M30	

Beispiel 2:

Das folgende Beispiel illustriert den Unterschied zwischen den beiden Arten der Bewegungsführung. Beide Male sei die Voreinstellung FGROUP(X,Y,Z) aktiv.



Unterschiedliche geometrische Beziehungen zwischen Achsen bei SPATH und UPATH

Programmcode

```
N10 G1 X0 A0 F1000 SPATH
N20 POLY PO[X]=(10,10) A10
```

Bzw.:

Programmcode

```
N10 G1 X0 F1000 UPATH
N20 POLY PO[X]=(10,10) A10
```

Im Satz N20 hängt der Weg S der FGROUP-Achsen vom Quadrat des Kurvenparameters U ab. Daher ergeben sich entlang des Wegs von X unterschiedliche Positionen der Synchronachse A, je nachdem, ob SPATH oder UPATH aktiv ist.

Weitere Informationen

Während Polynominterpolation - und damit seien immer die Polynominterpolation im engeren Sinne (POLY), alle Spline-Interpolationsarten (ASPLINE, BSPLINE, CSPLINE) und Linearinterpolation mit Kompressorfunktion (COMPON, COMPCURV) verstanden - sind die Positionen aller Bahnachsen i durch Polynome pi(U) vorgegeben. Der Kurvenparameter U bewegt sich dabei innerhalb eines NC-Satzes von 0 bis 1, ist also normiert.

Durch den Sprachbefehl FGROUP können innerhalb der Bahnachsen diejenigen Achsen ausgewählt werden, auf die sich der programmierte Bahnvorschub beziehen soll. Eine Interpolation mit konstanter Geschwindigkeit auf dem Weg S dieser Achsen bedeutet während Polynominterpolation jedoch in der Regel eine nicht konstante Änderung des Kurvenparameters U.

Steuerungsverhalten bei Reset und Maschinen-/Optionsdaten

Nach Reset ist der durch MD20150 $MC_GCODE_RESET_VALUES[44] bestimmte G-Code wirksam (45. G-Code-Gruppe). Um kompatibel zu bestehenden Anlagen zu bleiben, wird hier als Standardwert SPATH voreingestellt.

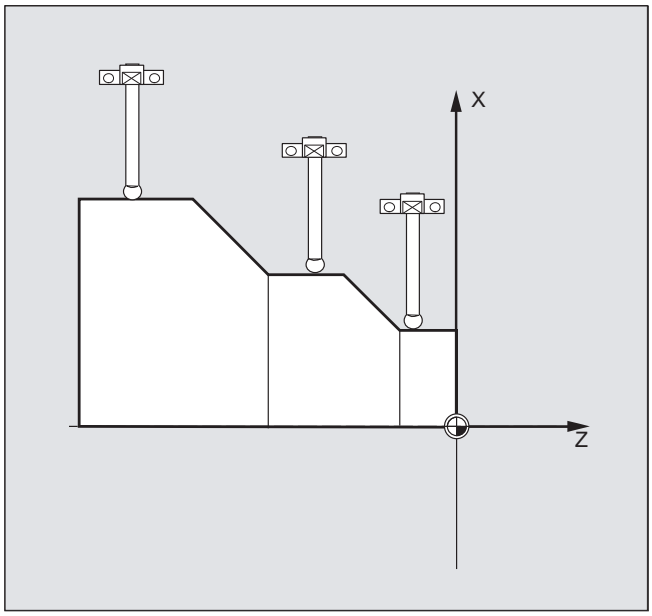
Der Grundstellungswert für die Art des Überschleifens wird mit MD20150 $MC_GCODE_RESET_VALUES[9] festgelegt (10. G-Code-Gruppe).

Das achsspezifische Maschinendatum MD33100 $MA_COMPRESS_POS_TOL[<n>] hat eine erweiterte Bedeutung: es enthält die Toleranzen für die Kompressorfunktion und für das Überschleifen mit G642.

4.7 Messen mit schaltendem Taster (MEAS, MEAW)

Funktion

Mit der Funktion "Messen mit schaltendem Taster" werden Istpositionen am Werkstück angefahren und bei der Schaltflanke des Messtasters werden für alle im Messsatz programmierten Achsen die Positionen gemessen und für jede Achse in die entsprechende Speicherzelle geschrieben.



Messsätze programmieren

Für die Programmierung der Funktion stehen die beiden folgenden Befehle zur Verfügung:

- `MEAS`

  Mit dem Befehl `MEAS` wird der Restweg zwischen Ist- und Sollposition gelöscht.

- `MEAW`

  Der Befehl `MEAW` wird für Messaufgaben eingesetzt, bei denen in jedem Fall die programmierte Position angefahren werden soll.

`MEAS` und `MEAW` sind satzweise wirksam und werden zusammen mit Bewegungsanweisungen programmiert. Die Vorschübe und Interpolationsarten (G0, G1, …), ebenso wie die Anzahl der Achsen, müssen dabei dem jeweiligen Messproblem angepasst sein.

Messergebnisse lesen

Die Messergebnisse für die mit Messtaster erfassten Achsen stehen unter folgenden Variablen zur Verfügung:

- `$AA_MM[<Achse>]`
  Messergebnisse im Maschinenkoordinatensystem
- `$AA_MW[<Achse>]`
  Messergebnisse im Werkstückkoordinatensystem

Beim Lesen dieser Variablen wird intern kein Vorlaufstopp erzeugt.

Hinweis

Mit `STOPRE` muss im NC-Programm an geeigneter Stelle ein Vorlaufstopp programmiert werden. Ansonsten werden falsche Werte gelesen.

Syntax

```
MEAS=<TE> G... X... Y... Z...
MEAW=<TE> G... X... Y... Z...
```

Bedeutung

`MEAS`	Befehl: Messen **mit** Restweglöschen Wirksamkeit: satzweise
`MEAW`	Befehl: Messen **ohne** Restweglöschen Wirksamkeit: satzweise
`<TE>`	Trigger-Ereignis zur Auslösung der Messung Typ: INT Wertebereich: -2, -1, 1, 2 **Hinweis:** Es existieren maximal 2 Messtaster (je nach Ausbaustufe). Bedeutung: (+)1 steigende Flanke von Messtaster 1 (auf Messeingang 1) -1 fallende Flanke von Messtaster 1 (auf Messeingang 1) (+)2 steigende Flanke von Messtaster 2 (auf Messeingang 2) -2 fallende Flanke von Messtaster 2 (auf Messeingang 2) **Hinweis:** Es existieren maximal 2 Messtaster (je nach Ausbaustufe).
`G...`	Interpolationsart, z. B. `G0`, `G1`, `G2` oder `G3`
`X... Y... Z...`	Endpunkt in kartesischen Koordinaten

Beispiel

```
Programmcode                                    Kommentar
N10 MEAS=1 G1 F1000 X100 Y730 Z40               ; Messsatz mit Messtaster des ersten
                                                  Messeingangs und Geradeninterpolation.
                                                  Vorlaufstopp wird automatisch erzeugt.
...
```

Weitere Informationen

Messauftragsstatus

Ist im Programm eine Auswertung erforderlich, ob der Messtaster geschaltet hat oder nicht, kann die Zustandsvariable `$AC_MEA[n]` (n = Nummer des Messtasters) abgefragt werden:

Wert	Bedeutung
0	Messauftrag nicht erfüllt
1	Messauftrag erfolgreich beendet (Messtaster hat geschaltet)

Hinweis

Wird der Messtaster im Programm ausgelenkt, wird die Variable auf 1 gesetzt. Beim Start eines Messsatzes wird die Variable automatisch auf den Anfangszustand des Tasters gesetzt.

Messwerteaufnahme

Es werden die Positionen aller verfahrenen Bahn- und Positionierachsen des Satzes (maximale Anzahl an Achsen je nach Steuerungskonfiguration) erfasst. Bei `MEAS` wird die Bewegung nach dem Schalten des Messtasters definiert abgebremst.

Hinweis

Ist in einem Messsatz eine GEO-Achse programmiert, werden die Messwerte für alle aktuellen GEO-Achsen abgelegt.

Ist in einem Messsatz eine an einer Transformation beteiligte Achse programmiert, werden die Messwerte aller an dieser Transformation beteiligten Achsen abgelegt.

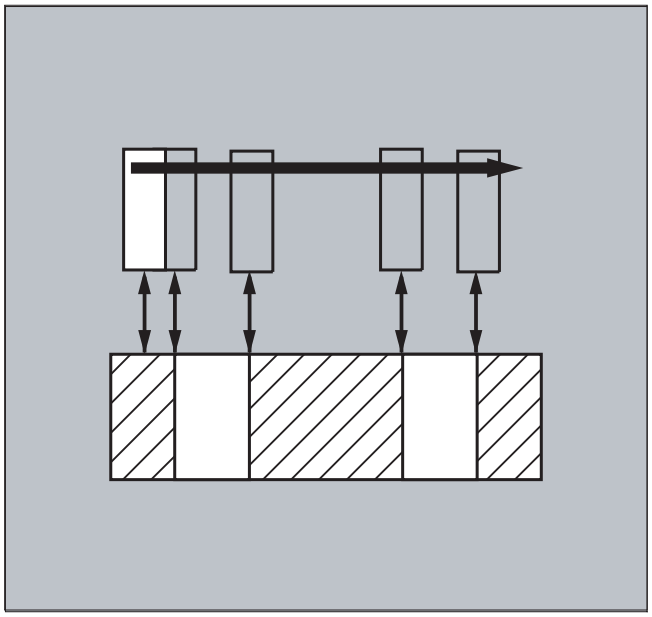
4.8 Erweiterte Messfunktion (MEASA, MEAWA, MEAC) (Option)

Funktion

Beim axialen Messen können mehrere Messtaster und mehrere Messsysteme benutzt werden.

Mit dem Befehl `MEASA` bzw. `MEAWA` werden für die jeweils programmierte Achse bis zu vier Messwerte pro Messung erfasst und passend zum Trigger-Ereignis in Systemvariablen abgelegt.

Kontinuierliche Messaufträge können mit dem Befehl `MEAC` durchgeführt werden. In diesem Fall werden die Messergebnisse in FIFO-Variablen abgelegt. Auch für `MEAC` sind pro Messung maximal vier Messwerte möglich.



Messergebnisse lesen

Die Messergebnisse stehen unter folgenden Variablen zur Verfügung:

- $AA_MM1...4[<Achse>]

  Messergebnisse im Maschinenkoordinatensystem

- $AA_MW1...4[<Achse>]

  Messergebnisse im Werkstückkoordinatensystem

Syntax

```
MEASA[<Achse>]=(<Modus>,<TE1>,...,<TE4>)
MEAWA[<Achse>]=(<Modus>,<TE1>,...,<TE4>)
MEAC[<Achse>]=(<Modus>,<Messspeicher>,<TE1>,...,<TE4>)
```

Hinweis

MEASA und MEAWA sind satzweise wirksam und können zusammen in einem Satz programmiert werden. Wird dagegen MEASA/MEAWA zusammen mit MEAS/MEAW in einem Satz programmiert, kommt es zu einer Fehlermeldung.

Bedeutung

MEASA
Befehl: Axiales Messen mit Restweglöschen
Wirksamkeit: satzweise

MEAWA
Befehl: Axiales Messen ohne Restweglöschen
Wirksamkeit: satzweise

MEAC
Befehl: Axiales kontinuierliches Messen ohne Restweglöschen
Wirksamkeit: satzweise

<Achse>
Name der zur Messung verwendeten Kanalachse

<Modus>
Zweistellige Ziffer zur Angabe des Betriebsmodus (Messmodus und Messsystem)

Messmodus (Einerdekade):

0 Messauftrag abbrechen.

1 Bis zu 4 verschiedene gleichzeitig aktivierbare Triggerereignisse.

2 Bis zu 4 nacheinander aktivierbare Triggerereignisse.

3 Bis zu 4 nacheinander aktivierbare Triggerereignisse, jedoch keine Überwachung von Triggerereignis 1 beim START (Alarme 21700/21703 werden unterdrückt).

Hinweis:
Dieser Modus ist bei MEAC nicht möglich.

Messsystem (Zehnerdekade):

0 (oder keine Angabe)	aktives Messsystem
1	Messsystem 1
2	Messsystem 2
3	beide Messsysteme

<TE>
Trigger-Ereignis zur Auslösung der Messung
Typ: INT
Wertebereich: -2, -1, 1, 2
Bedeutung:

(+)1	steigende Flanke von Messtaster 1
-1	fallende Flanke von Messtaster 1
(+)2	steigende Flanke von Messtaster 2
-2	fallende Flanke von Messtaster 2

<Messspeicher>
Nummer des FIFO (Umlaufspeichers)

Beispiele

Beispiel 1: Axiales Messen mit Restweglöschen im Modus 1 (Auswertung in zeitlicher Reihenfolge)

a) mit 1 Messsystem

Programmcode	Kommentar
...	
N100 MEASA[X]=(1,1,-1) G01 X100 F100	; Messen im Modus 1 mit aktivem Messsystem. Warten auf Messsignal mit steigender/fallender Flanke von Messtaster 1 auf dem Verfahrweg nach X=100.
N110 STOPRE	; Vorlaufstopp
N120 IF $AC_MEA[1]==FALSE GOTOF ENDE	; Erfolg der Messung kontrollieren.
N130 R10=$AA_MM1[X]	; Zum ersten programmierten Triggerereignis (steigende Flanke) gehörigen Messwert speichern.
N140 R11=$AA_MM2[X]	; Zum zweiten programmierten Triggerereignis (fallende Flanke) gehörigen Messwert speichern.
N150 ENDE:	

b) mit 2 Messsystemen

Programmcode	Kommentar
...	
N200 MEASA[X]=(31,1,-1) G01 X100 F100	; Messen im Modus 1 mit beiden Messsystemen. Warten auf Messsignal mit steigender/fallender Flanke von Messtaster 1 auf dem Verfahrweg nach X=100.
N210 STOPRE	; Vorlaufstopp
N220 IF $AC_MEA[1]==FALSE GOTOF ENDE	; Erfolg der Messung kontrollieren.
N230 R10=$AA_MM1[X]	; Messwert des Messsystems 1 bei steigender Flanke speichern.
N240 R11=$AA_MM2[X]	; Messwert des Messsystems 2 bei steigender Flanke speichern.
N250 R12=$AA_MM3[X]	; Messwert des Messsystems 1 bei fallender Flanke speichern.
N260 R13=$AA_MM4[X]	; Messwert des Messsystems 2 bei fallender Flanke speichern.
N270 ENDE:	

Beispiel 2: Axiales Messen mit Restweglöschen im Modus 2 (Auswertung in programmierter Reihenfolge)

Programmcode	Kommentar
...	
N100 MEASA[X]=(2,1,-1,2,-2) G01 X100 F100	; Messen im Modus 2 mit aktivem Messsystem. Warten auf Messsignal in der Reihenfolge steigende Flanke von Messtaster 1, fallende Flanke Messtaster 1, steigende Flanke von Messtaster 2, fallende Flanke Messtaster 2 auf dem Verfahrweg nach X=100.
N110 STOPRE	; Vorlaufstopp
N120 IF $AC_MEA[1]==FALSE GOTOF MESSTASTER2	; Erfolg der Messung mit Messtaster 1 kontrollieren.
N130 R10=$AA_MM1[X]	; Zum ersten programmierten Triggerereignis (steigende Flanke Messtaster 1) gehörigen Messwert speichern.
N140 R11=$AA_MM2[X]	; Zum zweiten programmierten Triggerereignis (steigende Flanke Messtaster 1) gehörigen Messwert speichern.
N150 MESSTASTER2:	
N160 IF $AC_MEA[2]==FALSE GOTOF ENDE	; Erfolg der Messung mit Messtaster 2 kontrollieren.
N170 R12=$AA_MM3[X]	; Zum dritten programmierten Triggerereignis (steigende Flanke Messtaster 2) gehörigen Messwert speichern.
N180 R13=$AA_MM4[X]	; Zum vierten programmierten Triggerereignis (steigende Flanke Messtaster 2) gehörigen Messwert speichern.
N190 ENDE:	

Beispiel 3: Axiales kontinuierliches Messen im Modus 1 (Auswertung in zeitlicher Reihenfolge)

a) Messen von bis zu 100 Messwerten

```
Programmcode                                    Kommentar
...
N110 DEF REAL MESSWERT[100]
N120 DEF INT Schleife=0
N130 MEAC[X]=(1,1,-1) G01 X1000 F100            ; Messen im Modus 1 mit aktivem Messsystem, Speichern
                                                  der Messwerte unter $AC_FIFO1, Warten auf Messsignal
                                                  mit fallender Flanke von Messtaster 1 auf dem
                                                  Verfahrweg nach X=1000.
N135 STOPRE
N140 MEAC[X]=(0)                                ; Messung nach Erreichen der Achsposition abbrechen.
N150 R1=$AC_FIFO1[4]                            ; Anzahl aufgelaufener Messwerte in Parameter R1
                                                  speichern.
N160 FOR Schleife=0 TO R1-1
N170 MESSWERT[Schleife]=$AC_FIFO1[0]            ; Messwerte aus dem $AC_FIFO1 auslesen und
                                                  abspeichern.
N180 ENDFOR
```

b) Messen mit Restweglöschen nach 10 Messwerten

```
Programmcode                                                    Kommentar
...
N10 WHEN $AC_FIFO1[4]>=10 DO MEAC[x]=(0) DELDTG(x)              ; Restweg löschen.
N20 MEAC[x]=(1,1,1,-1) G01 X100 F500
N30 MEAC[X]=(0)
N40 R1=$AC_FIFO1[4]                                             ; Anzahl Messwerte.
...
```

Weitere Informationen

Messauftrag

Die Programmierung eines Messauftrags kann im Teileprogramm oder aus einer Synchronaktion (siehe Kapitel "Bewegungssynchronaktionen") heraus erfolgen. Pro Achse kann dabei zu ein- und demselben Zeitpunkt nur ein Messauftrag aktiv sein.

Hinweis

Der Vorschub ist dem jeweiligen Messproblem anzupassen.

Bei MEASA und MEAWA können korrekte Ergebnisse nur bei Vorschüben gewährleistet werden, bei denen nicht mehr als ein gleiches und nicht mehr als 4 verschiedene Trigger-Ereignisse pro Lagereglertakt eintreffen.

Beim kontinuierlichen Messen mit MEAC darf das Verhältnis zwischen Interpolationstakt und Lagereglertakt nicht größer als 8:1 werden.

Trigger-Ereignis

Ein Triggerereignis setzt sich zusammen aus der Nummer des Messtasters und dem Auslösekriterium (steigende oder fallende Flanke) des Messsignals.

Für jede Messung können jeweils bis zu 4 Trigger-Ereignisse der angesprochenen Messtaster verarbeitet werden, also bis zu zwei Messtaster mit je zwei Messflanken. Die Reihenfolge der Verarbeitung sowie die maximale Anzahl der Trigger-Ereignisse sind dabei abhängig vom gewählten Modus.

Hinweis

Für Messmodus 1 gilt: Ein gleiches Trigger-Ereignis darf nur einmal in einem Messauftrag programmiert werden!

Betriebsmodus

Mit der ersten Ziffer (Zehnerdekade) des Betriebsmodus wird das gewünschte Messsystem angewählt. Ist nur ein Messsystem vorhanden, jedoch das zweite programmiert, wird automatisch das vorhandene eingesetzt.

Mit der zweiten Ziffer (Einerdekade) wird der gewünschte Messmodus angewählt. Damit wird der Messvorgang an die Möglichkeiten der jeweiligen Steuerung angepasst:

- **Modus 1**

  Die Auswertung der Trigger-Ereignisse erfolgt in der zeitlichen Reihenfolge ihres Auftretens. In diesem Modus ist bei Einsatz von Sechsachsbaugruppen nur ein Trigger-Ereignis programmierbar bzw. wird bei Angabe mehrerer Trigger-Ereignisse automatisch in Modus 2 umgesetzt (ohne Meldung).

- **Modus 2**

  Die Auswertung der Trigger-Ereignisse erfolgt in der programmierten Reihenfolge.

- **Modus 3**

  Die Auswertung der Trigger-Ereignisse erfolgt in der programmierten Reihenfolge, jedoch keine Überwachung von Trigger-Ereignis 1 beim START.

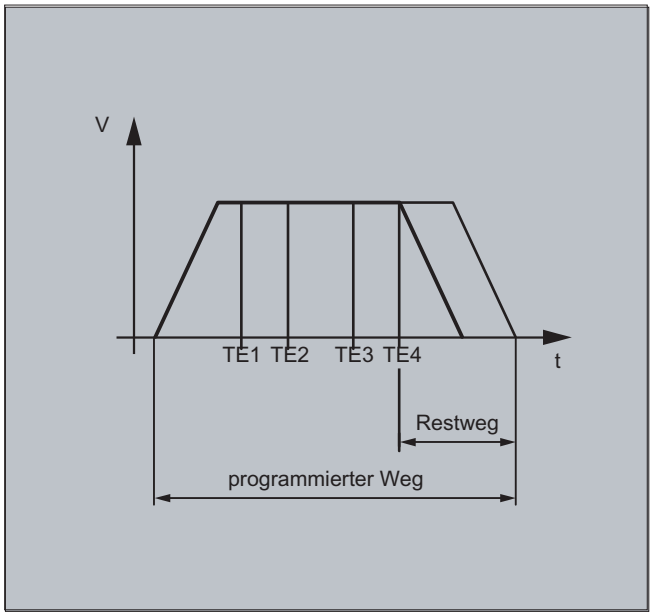
Hinweis

Bei Einsatz von 2 Messsystemen sind nur zwei Trigger-Ereignisse programmierbar.

Messen mit und ohne Restweglöschen

Bei der Programmierung von `MEASA` wird Restweglöschen erst nach der Erfassung aller geforderten Messwerte durchgeführt.

Für spezielle Messaufgaben, bei denen in jedem Fall die programmierte Position angefahren werden soll, wird `MEAWA` eingesetzt.



Hinweis

MEASA ist nicht in Synchronaktionen programmierbar. Ersatzweise kann MEAWA plus Restweglöschen als Synchronaktion programmiert werden.

Wird der Messauftrag mit MEAWA aus den Synchronaktionen gestartet, sind die Messwerte nur im Maschinen-Koordinatensystem verfügbar.

Messergebnisse für MEASA, MEAWA

Die Messergebnisse stehen unter folgenden Systemvariablen zur Verfügung:

- im Maschinen-Koordinatensystem:

$AA_MM1[<Achse>]	Messwert des programmierten Messsystems bei Trigger-Ereignis 1
...	...
$AA_MM4[<Achse>]	Messwert des programmierten Messsystems bei Trigger-Ereignis 4

- im Werkstück-Koordinatensystem:

$AA_WM1[<Achse>]	Messwert des programmierten Messsystems bei Trigger-Ereignis 1
...	...
$AA_WM4[<Achse>]	Messwert des programmierten Messsystems bei Trigger-Ereignis 4

Hinweis

Beim Lesen dieser Variablen wird intern kein Vorlaufstopp erzeugt. Mit STOPRE muss an geeigneter Stelle ein Vorlaufstopp programmiert werden. Ansonsten werden falsche Werte eingelesen.

Geometrieachsen / Transformationen

Soll das axiale Messen für eine Geometrieachse gestartet werden, muss der gleiche Messauftrag explizit für alle restlichen Geometrieachsen programmiert werden. Das gleiche gilt für Achsen, die an einer Transformation beteiligt sind.

Beispiel:

```
N10 MEASA[Z]=(1,1) MEASA[Y]=(1,1) MEASA[X]=(1,1) G0 Z100
```

oder

```
N10 MEASA[Z]=(1,1) POS[Z]=100
```

Messauftrag mit 2 Messsystemen

Wird ein Messauftrag mit zwei Messsystemen durchgeführt, wird jedes der beiden möglichen Trigger-Ereignisse von beiden Messsystemen der jeweiligen Achse erfasst. Die Belegung der reservierten Variablen ist damit vorgegeben:

`$AA_MM1[<Achse>]`	bzw.	`$AA_MW1[<Achse>]`	Messwert von Messsystem 1 bei Trigger-Ereignis 1
`$AA_MM2[<Achse>]`	bzw.	`$AA_MW2[<Achse>]`	Messwert von Messsystem 2 bei Trigger-Ereignis 1
`$AA_MM3[<Achse>]`	bzw.	`$AA_MW3[<Achse>]`	Messwert von Messsystem 1 bei Trigger-Ereignis 2
`$AA_MM4[<Achse>]`	bzw.	`$AA_MW4[<Achse>]`	Messwert von Messsystem 2 bei Trigger-Ereignis 2

Messtasterstatus

Der Messtasterstatus steht unter der folgenden Systemvariablen zur Verfügung:

$A_PROBE[<n>]

<n>=Messtaster

Wert	Bedeutung
1	Messtaster ausgelenkt
0	Messtaster nicht ausgelenkt

Messauftragsstatus bei MEASA, MEAWA

Ist im Programm eine Auswertung erforderlich, so kann der Messauftragsstatus über$AC_MEA[<n>], mit <n> = Nummer des Messtasters, abgefragt werden. Sobald alle in einem Satz programmierten Trigger-Ereignisse der Messtaster <n> erfolgt sind, liefert diese Variable den Wert 1. Anderenfalls ist der Wert 0.

Hinweis

Wird Messen aus Synchronaktionen gestartet, wird $AC_MEA nicht mehr aktualisiert. In diesem Fall sind neue PLC-Statussignale DB31, … DBX62.3 bzw. die gleichwertige Variable $AA_MEAACT[<Achse>] abzufragen.

Bedeutung:

$AA_MEAACT==1: Messen aktiv

$AA_MEAACT==0: Messen nicht aktiv

Kontinuierliches Messen (MEAC)

Die Messwerte liegen bei `MEAC` im Maschinenkoordinatensystem vor und werden im angegebenen FIFO[n]-Speicher (Umlaufspeicher) abgelegt. Sind für die Messung zwei Messtaster projektiert, werden die Messwerte des zweiten Messtasters getrennt im zusätzlich dafür projektierten (über MD einstellbar) FIFO[n+1]-Speicher abgelegt.

FIFO-Speicher ist ein Umlaufspeicher, in den Messwerte im Umlaufprinzip in $AC_FIFO-Variablen eingetragen werden, siehe Kapitel "Bewegungssynchronaktionen".

Hinweis

Der FIFO-Inhalt kann nur einmal aus dem Umlaufspeicher ausgelesen werden. Zur Mehrfachverwendung der Messdaten müssen diese in den Anwenderdaten zwischengespeichert werden.

Überschreitet die Anzahl der Messwerte für den FIFO-Speicher die im Maschinendatum festgelegte Höchstzahl, so wird die Messung automatisch beendet.

Endloses Messen lässt sich durch zyklisches Auslesen von Messwerten realisieren. Das Auslesen muss dabei mindestens in der gleichen Häufigkeit wie der Eingang von neuen Messwerten erfolgen.

Erkannte Fehlerprogrammierungen

Folgende Fehlprogrammierungen werden erkannt und mit einem Fehler angezeigt:

- MEASA/MEAWA zusammen mit MEAS/MEAW in einem Satz programmiert

  Beispiel:

  ```
  N01 MEAS=1 MEASA[X]=(1,1) G01 F100 POS[X]=100
  ```

- MEASA/MEAWA mit Parameteranzahl <2 oder >5

  Beispiel:

  ```
  N01 MEAWA[X]=(1) G01 F100 POS[X]=100
  ```

- MEASA/MEAWA mit Trigger-Ereignis ungleich 1/ -1/ 2/ -2

  Beispiel:

  ```
  N01 MEASA[B]=(1,1,3) B100
  ```

- MEASA/MEAWA mit falschem Modus

  Beispiel:

  ```
  N01 MEAWA[B]=(4,1) B100
  ```

- MEASA/MEAWA mit doppelt programmierten Trigger-Ereignis

  Beispiel:

  ```
  N01 MEASA[B]=(1,1,-1,2,-1) B100
  ```

- MEASA/MEAWA und fehlende GEO-Achse

  Beispiel:

  ```
  N01 MEASA[X]=(1,1) MEASA[Y]=(1,1) G01 X50 Y50 Z50 F100 ;GEO-Achse X/Y/Z
  ```

- Uneinheitlicher Messauftrag bei GEO-Achsen

  Beispiel:

  ```
  N01 MEASA[X]=(1,1) MEASA[Y]=(1,1) MEASA[Z]=(1,1,2) G01 X50 Y50 Z50 F100
  ```

4.9 Spezielle Funktionen für den OEM-Anwender (OEMIPO1, OEMIPO2, G810 bis G829)

Funktion

OEM-Adressen

Die Bedeutung der OEM-Adressen bestimmt der OEM-Anwender. Die Funktionalität wird über Compile-Zyklen eingebracht. 5 OEM-Adressen sind reserviert. Die Adressbezeichner sind einstellbar. OEM-Adressen sind in jedem Satz zulässig.

Parameter

Reservierte G-Gruppen

Gruppe 1 mit `OEMIPO1`, `OEMIPO2`

Der OEM-Anwender kann zwei zusätzliche Namen der G-Funktionen `OEMIPO1`, `OEMIPO2` definieren. Diese Funktionalität wird über Compile-Zyklen eingebracht und ist für den OEM-Anwender reserviert.

- Gruppe 31 mit **G810 bis G819**
- Gruppe 32 mit **G820 bis G829**

Für den OEM-Anwender sind zwei G-Gruppen mit jeweils 10 OEM-G-Funktionen reserviert. **Damit lassen sich die vom OEM-Anwender eingebrachten Funktionen zur Nutzung nach außen bringen.**

Funktionen und Unterprogramme

Zusätzlich können OEM-Anwender auch vordefinierte Funktionen und Unterprogramme mit Parameterübergabe anlegen.

4.10 Vorschubreduzierung mit Eckenverzögerung (FENDNORM, G62, G621)

Funktion

Bei der automatischen Eckenverzögerung wird der Vorschub glockenförmig kurz vor der betreffenden Ecke abgesenkt. Außerdem kann das Ausmaß des für die Bearbeitung relevanten Werkzeugverhaltens über Settingdaten parametriert werden. Dies sind:

- Beginn und Ende der Vorschubreduzierung
- Override, mit dem der Vorschub reduziert wird
- Erkennung der relevanten Ecke

Als relevante Ecken werden diejenigen Ecken berücksichtigt, deren Innenwinkel kleiner als die über Settingdatum parametrierte Ecke ist.

Mit dem Defaultwert `FENDNORM` wird die Funktion des automatischen Eckenoverride ausgeschaltet.

Literatur:
/FBFA/ Funktionsbeschreibung ISO-Dialekte

Syntax

```
FENDNORM
G62 G41
G621
```

Bedeutung

`FENDNORM`	Automatischer Eckenverzögerung aus
`G62`	Eckenverzögerung an Innenecken bei aktiver Werkzeugradiuskorrektur
`G621`	Eckenverzögerung an allen Ecken bei aktiver Werkzeugradiuskorrektur

G62 wirkt nur an den Innenecken mit

- aktiver Werkzeugradiuskorrektur `G41`, `G42` und
- aktiven Bahnsteuerbetrieb `G64`, `G641`

Die entsprechende Ecke wird mit dem abgesenkten Vorschub angefahren, der sich ergibt aus:

F * (Override zur Vorschubreduzierung) * Vorschuboverride

Die maximal mögliche Vorschubabsenkung wird genau dann erreicht, wenn das Werkzeug, bezogen auf die Mittelpunktsbahn, den Richtungswechsel an der betreffenden Ecke vornehmen soll.

G621 wirkt analog zu G62 an jeder Ecke, der durch `FGROUP` festgelegten Achsen.

4.11 Programmierbares Bewegungsendekriterium (FINEA, COARSEA, IPOENDA, IPOBRKA, ADISPOSA)

Funktion

Ähnlich dem Satzwechselkriterium bei Bahninterpolation (G601, G602 und G603) kann das Bewegungsendekriterium bei Einzelachsinterpolation in einem Teileprogramm bzw. in Synchronaktionen für Kommando-/PLC-Achsen programmiert werden.

Je nachdem, welches Bewegungsendekriterium eingestellt ist, werden Teileprogrammsätze bzw. Technologiezyklussätze mit Einzelachsbewegungen unterschiedlich schnell beendet. Gleiches gilt für PLC über FC15/16/18.

Syntax

```
FINEA[<Achse>]
COARSEA[<Achse>]
IPOENDA[<Achse>]
IPOBRKA(<Achse>[,<Zeitpunkt>])
ADISPOSA(<Achse>[,<Modus>,<Fenstergröße>])
```

Bedeutung

`FINEA`:	Bewegungsende bei Erreichen von "Genauhalt fein"
`COARSEA`:	Bewegungsende bei Erreichen von "Genauhalt grob"
`IPOENDA`:	Bewegungsende bei Erreichen von "Interpolator-Stopp"
`IPOBRKA`:	Satzwechsel in der Bremsrampe möglich
`ADISPOSA`:	Größe des Toleranzfensters zum Bewegungsendekriterium
`<Achse>`:	Kanalachsname (X, Y, ...)
`<Zeitpunkt>`:	Zeitpunkt des Satzwechsels, bezogen auf die Bremsrampe in %
`<Modus>`:	Modus Typ: INT Wertebereich: 0 Toleranzfenster nicht aktiv 1 Toleranzfenster bezüglich Sollposition 2 Toleranzfenster bezüglich Istposition
`<Fenstergröße>`:	Größe des Toleranzfensters Dieser Wert wird hauptlaufsynchron in das Settingdatum SD43610 $SA_ADISPOSA_VALUE eingetragen. Typ: REAL

Beispiele

Beispiel 1: Bewegungsende bei Erreichen von Interpolator-Stopp

```
Programmcode                                                    Kommentar
...
N110 G01 POS[X]=100 FA[X]=1000 ACC[X]=90 IPOENDA[X]             Fahren auf Position
                                                                X100 mit einer
                                                                Bahngeschwindigkeit
                                                                von 1000 U/min mit
                                                                einem
                                                                Beschleunigungswert
                                                                von 90% und dem
                                                                Bewegungsende bei
                                                                Erreichen von
                                                                Interpolator-Stopp.
...
N120 EVERY $A_IN[1] DO POS[X]=50 FA[X]=2000 ACC[X]=140 IPOENDA[X]  ; Fahren auf Position
                                                                X50, wenn der
                                                                Eingang 1 aktiv ist,
                                                                mit einer
                                                                Bahngeschwindigkeit
                                                                von 2000 U/min und
                                                                einem
                                                                Beschleunigungswert
                                                                von 140% und dem
                                                                Bewegungsende bei
                                                                Erreichen von
                                                                Interpolator-Stopp.
...
```

Beispiel 2: Satzwechselkriterium Bremsrampe im Teileprogramm

```
Programmcode            Kommentar
                        ; Defaulteinstellung wirksam
N40 POS[X]=100          ; Satzwechsel erfolgt, wenn X-Achse die Position 100 und Genauhalt fein
                          erreicht hat.
N20 IPOBRKA(X,100)      ; Satzwechselkriterium Bremsrampe aktivieren.
N30 POS[X]=200          ; Satzwechsel erfolgt, sobald die X-Achse zu bremsen beginnt.
N40 POS[X]=250          ; die X-Achse bremst nicht auf Position 200, sondern fährt weiter auf
                          Position 250, sobald die X-Achse zu bremsen beginnt erfolgt der
                          Satzwechsel.
N50 POS[X]=0            ; die X-Achse bremst und fährt auf Position 0 zurück der Satzwechsel
                          erfolgt bei Position 0und Genauhalt fein.
N60 X10 F100
N70 M30
...
```

Beispiel 3: Satzwechselkriterium Bremsrampe in Synchronaktionen

```
Programmcode                Kommentar
                            ; Im Technologie-Zyklus:
FINEA                       ; Bewegungsendekriterium Genauhalt fein.
POS[X]=100                  ; Technologie-Zyklus-Satzwechsel erfolgt, wenn die X-Achse die Position
                              100 und Genauhalt fein erreicht hat.
IPOBRKA(X,100)              ; Satzwechselkriterium Bremsrampe aktivieren.
POS[X]=100                  ; POS[X]=100; Technologie-Zyklus-Satzwechsel erfolgt, sobald die X-Achse
                              zu bremsen beginnt.
POS[X]=250                  ; die X-Achse bremst nicht auf Position 200 sondern fährt weiter auf
                              Position 250, sobald die X-Achse zu bremsen beginnt, erfolgt der
                              Satzwechsel im Technologie-Zyklus.
POS[X]=250                  ; die X-Achse bremst und fährt auf Position 0 zurück der Satzwechsel
                              erfolgt bei Position 0 und Genauhalt fein.
M17
```

Weitere Informationen

Bewegungsendekriterium lesen

Das eingestellte Bewegungsendekriterium kann mit der Systemvariablen$AA_MOTEND[<Achse>] abgefragt werden:

Wert	Bedeutung
1	Bewegungsende mit "Genauhalt fein"
2	Bewegungsende mit "Genauhalt grob"
3	Bewegungsende mit "IPO-Stopp"
4	Satzwechselkriterium Bremsrampe der Achsbewegung
5	Satzwechsel in der Bremsrampe mit Toleranzfenster bezüglich "Sollposition"
6	Satzwechsel in der Bremsrampe mit Toleranzfenster bezüglich "Istposition"

Hinweis

Nach RESET bleibt der letzte programmierte Wert bestehen.

Satzwechselkriterium in der Bremsrampe

Der prozentuelle Wert wird hauptlaufsynchron eingetragen ins Settingdatum:

SD43600 $SA_IPOBRAKE_BLOCK_EXCHANGE

Wird kein Wert angegeben, so wird der aktuelle Wert dieses Settingdatums wirksam.

Es ist ein Bereich von 0 % bis 100 % einstellbar.

Zusätzliches Toleranzfenster für IPOBRKA

Zum bereits bestehenden Satzwechselkriterium in der Bremsrampe kann auch ein zusätzliches Satzwechselkriterium "Toleranzfenster" angewählt werden. Die Freigabe erfolgt erst, wenn die Achse:

- wie bisher den vorgegebenen %-Wert ihrer Bremsrampe erreicht hat

  und

- ihre aktuelle Ist- oder Sollposition nicht weiter als eine Toleranz von der Endposition der Achse im Satz entfernt ist.

Literatur

Weitere Informationen zum Satzwechselkriterium von Positionierachsen siehe:

- Funktionshandbuch Erweiterungsfunktionen; Positionierachsen (P2)
- Programmierhandbuch Grundlagen; Kapitel "Vorschubregelung"

4.12 Programmierbarer Servo-Parametersatz (SCPARA)

Funktion

Mit SCPARA kann der Parametersatz (bestehend aus MDs) im Teileprogramm und in Synchronaktionen programmiert werden (bisher nur über PLC).

DB3n DBB9 Bit3

Damit es zu keinen Konflikten zwischen PLC und NCK kommt, wird ein weiteres Bit auf der PLC → NCK -Nahtstelle definiert:

DB3n DBB9 Bit3 "Parametersatzvorgabe durch SCPARA gesperrt".

Bei einer gesperrten Parametersatzvorgabe für SCPARA kommt es zu keiner Fehlermeldung, falls diese doch programmiert wird.

Syntax

```
SCPARA[<Achse>]=<Wert>
```

Bedeutung

SCPARA	Parametersatz festlegen
<Achse>	Kanalachsname (X, Y, ...)
<Wert>	gewünschter Parametersatz (1<= Wert <=6)

Hinweis

Der aktuelle Parametersatz kann mit der Systemvariablen $AA_SCPAR[<Achse>] abgefragt werden.

Bei G33, G331 bzw. G332 wird der geeignetste Parametersatz von der Steuerung ausgewählt.

Falls der **Servo-Parametersatz** sowohl in einem Teileprogramm bzw. in einer Synchronaktion als auch in der PLC **gewechselt** werden soll, muss das PLC-Anwenderprogramm erweitert werden.

Literatur:
/FB1/ Funktionshandbuch Grundfunktionen; Vorschübe (V1),
Kapitel "Vorschubbeeinflussung".

Beispiel

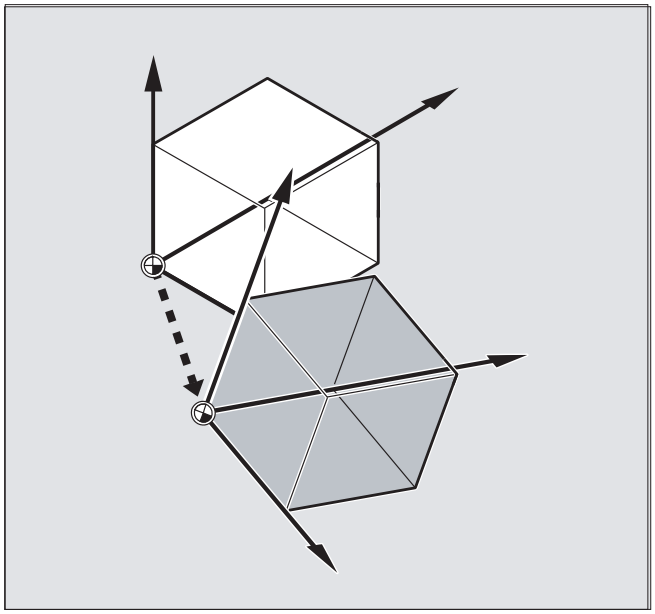
```
Programmcode            Kommentar
...
N110 SCPARA[X]= 3       ; Der 3.Parametersatz wird für die Achse X angewählt.
...
```

Koordinatentransformationen (FRAMES) 5

5.1 Koordinatentransformation über Framevariable

Funktion

Neben den im Programmierhandbuch "Grundlagen" bereits beschriebenen Programmiermöglichkeiten können Sie Koordinatensysteme auch mit vordefinierten Framevariablen festlegen.



Folgende Koordinatensysteme sind definiert:

MKS: Maschinen-Koordinatensystem

BKS: Basis-Koordinatensystem

BNS: Basisnullpunkt-Koordinatensystem

ENS: Einstellbares Nullpunkt-Koordinatensystem

WKS: Werkstück-Koordinatensystem

Was ist eine vordefinierte Framevariable?

Vordefinierte Framevariablen sind Schlüsselwörter, die im Sprachgebrauch der Steuerung mit entsprechender Wirkung bereits festgelegt sind und im NC-Programm verarbeitet werden können.

Mögliche Framevariable:

- Basisframe (Basisverschiebung)
- einstellbare Frames
- programmierbarer Frame

Wertzuweisungen und Istwerte auslesen

Zusammenhang Framevariable/Frame

Eine Koordinatentransformation kann durch Wertzuweisung eines Frames an eine Framevariable aktiviert werden.

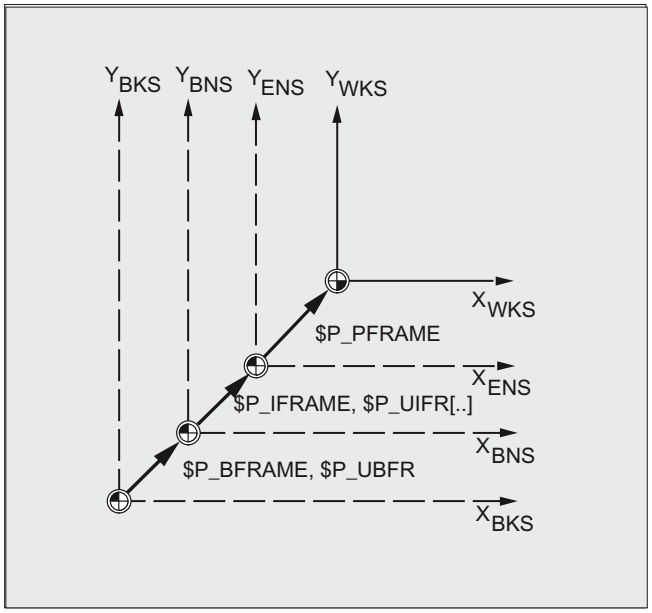
Beispiel: `$P_PFRAME=CTRANS(X,10)`

Framevariable:

`$P_PFRAME` bedeutet: aktueller programmierbarer Frame.

Frame:

`CTRANS(X,10)` bedeutet: programmierbare Nullpunktverschiebung der X-Achse um 10 mm.



Istwerte auslesen

Über vordefinierte Variable im Teileprogramm können die aktuellen Istwerte der Koordinatensysteme ausgelesen werden:

$AA_IM[Achse]: Lesen Istwert im MKS

$AA_IB[Achse]: Lesen Istwert im BKS

$AA_IBN[Achse:] Lesen Istwert im BNS

$AA_IEN[Achse]: Lesen Istwert im ENS

$AA_IW[Achse]: Lesen Istwert im WKS

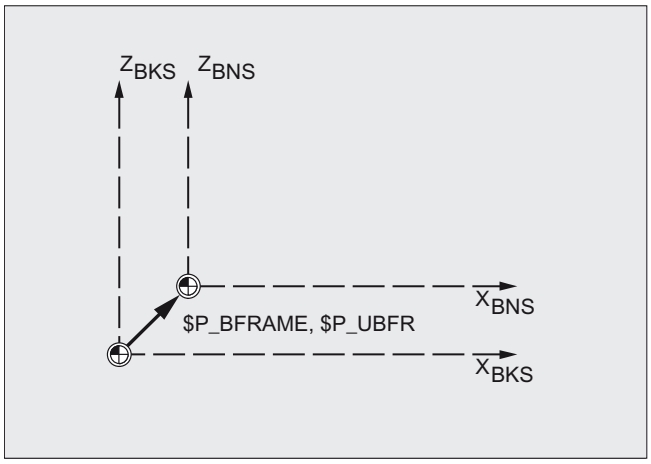
5.1.1 Vordefinierte Framevariable ($P_BFRAME, $P_IFRAME, $P_PFRAME, $P_ACTFRAME)

$P_BFRAME

Aktuelle Basisframevariable, die den Bezug zwischen Basiskoordinatensystem (BKS) und Basis-Nullpunktsystem (BNS) herstellt.

Soll der durch $P_UBFR beschriebene Basisframe sofort im Programm wirksam werden, muss entweder

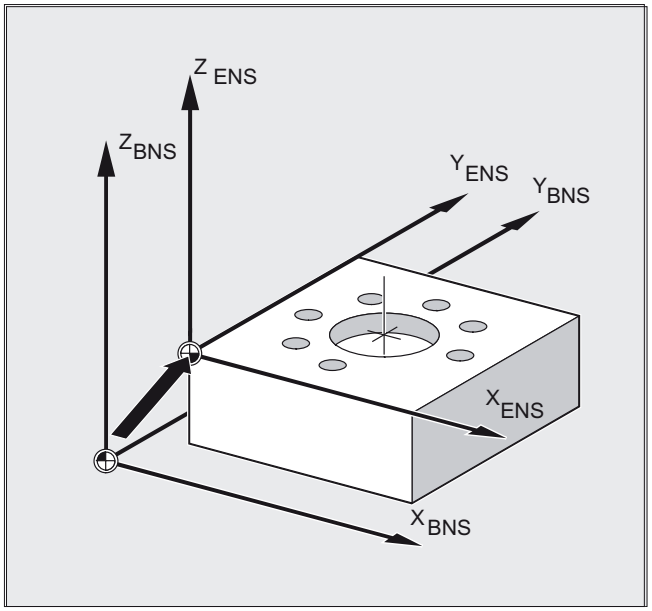
- ein `G500`, `G54...G599` programmiert werden oder
- `$P_BFRAME` mit `$ $P_UBFR` beschrieben werden.



$P_IFRAME

Aktuelle, einstellbare Framevariable, die den Bezug zwischen Basis-Nullpunktsystem (BNS) und Einstellbarem Nullpunktsystem (ENS) herstellt.

- `$P_IFRAME` entspricht `$P_UIFR[$P_IFRNUM]`
- `$P_IFRAME` enthält nach Programmierung von z. B. G54 die durch `G54` definierte Translation, Rotation, Skalierung und Spiegelung.

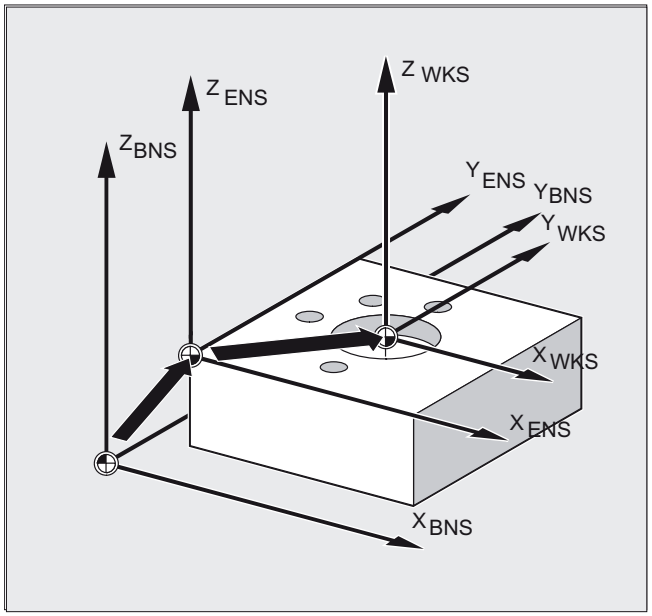


$P_PFRAME

Aktuelle, programmierbare Framevariable, die den Bezug zwischen dem Einstellbaren Nullpunktsystem (ENS) und dem Werkstückkoordinatensystem (WKS) herstellt.

`$P_PFRAME` enthält den resultierenden Frame, der sich

- **aus der Programmierung** von `TRANS/ATRANS`, `ROT/AROT`, `SCALE/ASCALE`, `MIRROR/AMIRROR` bzw.
- **aus der Zuweisung** von `CTRANS`, `CROT`, `CMIRROR`, `CSCALE` an den programmierbaren FRAME ergibt



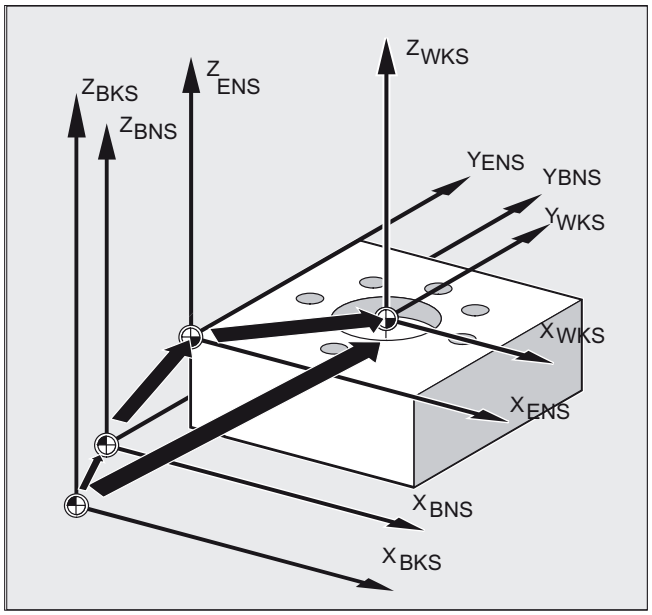
$P_ACTFRAME

Aktueller, resultierender Gesamtframe, der sich durch Verkettung aus

- der aktuellen Basisframevariablen `$P_BFRAME`,
- der aktuellen einstellbaren Framevariablen `$P_IFRAME` mit Systemframes und
- der aktuellen programmierbaren Framevariablen `$P_PFRAME` mit Systemframes

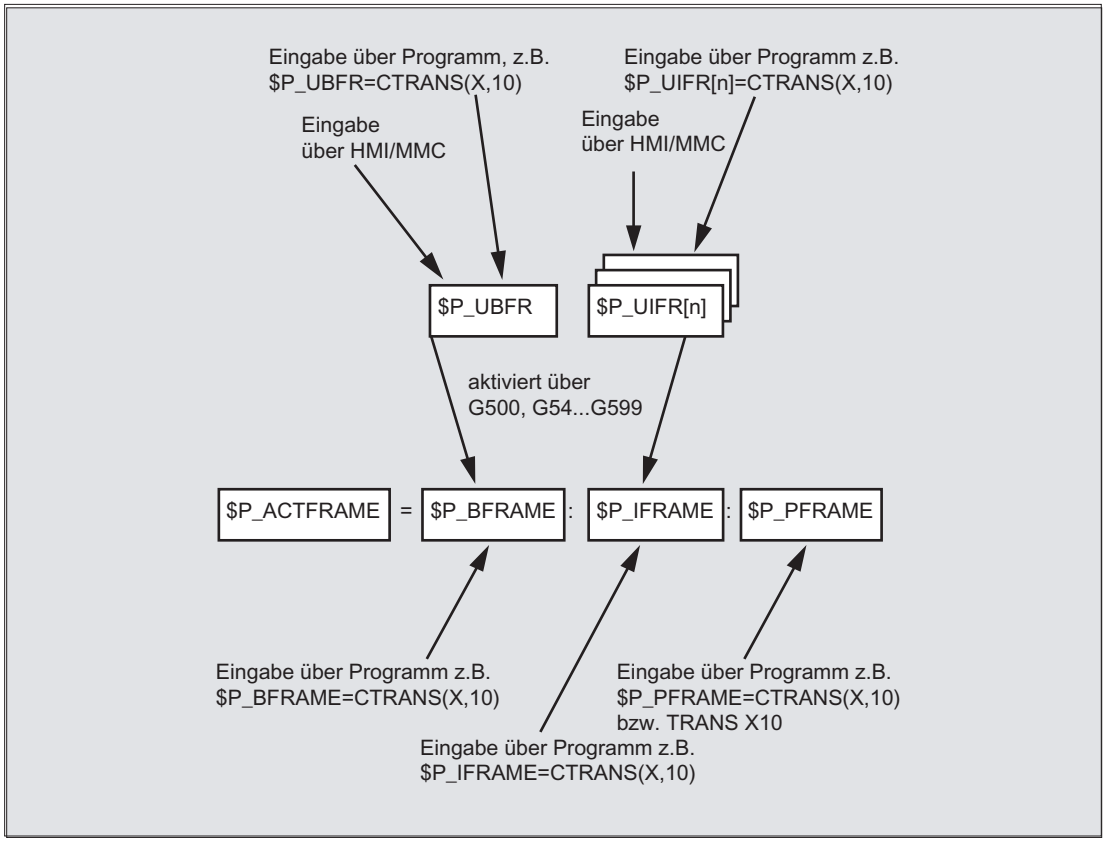
ergibt. Systemframes, siehe Kapitel "Im Kanal wirksame Frames"

`$P_ACTFRAME` beschreibt den aktuell gültigen Werkstücknullpunkt.



Falls `$P_BFRAME`, `$P_IFRAME` oder `$P_PFRAME` verändert werden, wird `$P_ACTFRAME` neu berechnet.

`$P_ACTFRAME` entspricht `$P_BFRAME`:`$P_IFRAME`:`$P_PFRAME`



Basisframe und einstellbarer Frame wirken nach Reset, wenn das MD 20110 RESET_MODE_MASK folgendermaßen eingestellt ist:

Bit0=1, Bit14=1 --> `$P_UBFR` (Basisframe) wirkt

Bit0=1, Bit5=1 --> `$P_UIFR[$P_UIFRNUM]` (einst. Frame) wirkt

Vordefinierte einstellbare Frames $P_UBFR

Mit $P_UBFR wird der Basisframe programmiert, er wird aber nicht gleichzeitig im Teileprogramm aktiv. Der mit $P_UBFR geschriebene Basisframe wird eingerechnet, wenn

- Reset geschaltet wurde und die Bits 0 und 14 des MD RESET_MODE_MASK gesetzt sind,
- die Anweisungen `G500`, `G54...G599` ausgeführt wurden.

Vordefinierte einstellbare Frames $P_UIFR[n]

Durch die vordefinierte Framevariable $P_UIFR[n] können die einstellbaren Nullpunktverschiebungen `G54` bis `G599` vom Teileprogramm aus gelesen oder geschrieben werden.

Diese Variablen stellen im Aufbau ein eindimensionales Feld vom Typ FRAME mit dem Namen `$P_UIFR[n]` dar.

Zuordnung zu den G-Befehlen

Standardmäßig sind 5 einstellbare Frames `$P_UIFR[0]... $P_UIFR[4]` bzw. 5 gleichbedeutende G-Befehle – `G500` und `G54` bis `G57` , unter deren Adressen Werte abgespeichert werden können.

`$P_IFRAME=$P_UIFR[0]` entspricht `G500`

`$P_IFRAME=$P_UIFR[1]` entspricht `G54`

`$P_IFRAME=$P_UIFR[2]` entspricht `G55`

`$P_IFRAME=$P_UIFR[3]` entspricht `G56`

`$P_IFRAME=$P_UIFR[4]` entspricht `G57`

Über Maschinendatum können Sie die Anzahl der Frames verändern:

`$P_IFRAME=$P_UIFR[5]` entspricht `G505`

`... ... ...`

`$P_IFRAME=$P_UIFR[99]` entspricht `G599`

Hinweis

Hierdurch lassen sich insgesamt 100 Koordinatensysteme erzeugen, die z. B. als Nullpunkt für verschiedene Vorrichtungen programmübergreifend aufgerufen werden können.

⚠ VORSICHT

Die Programmierung von Framevariablen und Frames erfordert im NC-Programm einen eigenen NC-Satz. **Ausnahme:** Programmierung eines einstellbaren Frames mit `G54`, `G55`, `...`

5.2 Framevariablen/Frames Werte zuweisen

5.2.1 Direkte Werte zuweisen (Achswert, Winkel, Maßstab)

Funktion

Im NC-Programm können Sie direkt Frames oder Framevariablen mit Werten belegen.

Syntax

```
$P_PFRAME=CTRANS (X, Achswert, Y, Achswert, Z, Achswert, …)
$P_PFRAME=CROT (X, Winkel, Y, Winkel, Z, Winkel, …)
$P_UIFR[..]=CROT (X, Winkel, Y, Winkel, Z, Winkel, …)
$P_PFRAME=CSCALE (X, Maßstab, Y, Maßstab, Z, Maßstab, …)
$P_PFRAME=CMIRROR (X, Y, Z)
```

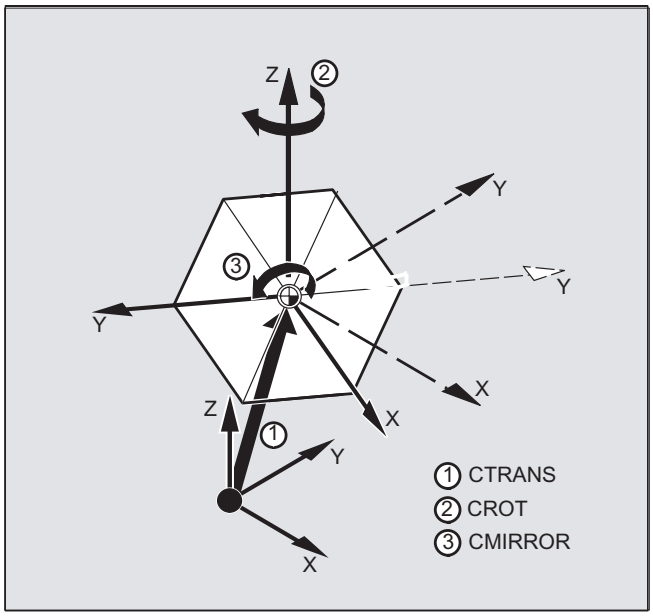
Die Programmierung von $P_BFRAME erfolgt analog wie $P_PFRAME.

Bedeutung

`CTRANS`	Verschiebung in den angegebenen Achsen
`CROT`	Drehung um die angegebenen Achsen
`CSCALE`	Maßstabsveränderung in den angegebenen Achsen
`CMIRROR`	Richtungsumkehr der angegebenen Achse
`X Y Z`	Verschiebewert in Richtung der angegebenen Geometrieachse
`Achswert`	Achswert der Verschiebung zuweisen
`Winkel`	Drehwinkel um die angegebenen Achsen zuweisen
`Maßstab`	Maßstab verändern

Beispiel

Durch Wertzuweisung an dem aktuellen programmierbaren Frame werden Translation, Drehung und Spiegelung aktiviert.




```
N10 $P_PFRAME=CTRANS(X,10,Y,20,Z,5):CROT(Z,45):CMIRROR(Y)
```

Frame-Rot-Komponenten mit anderen Werten vorbelegen

Mit CROT alle drei Komponenenten von UIFR mit Werten vorbelegen

Programmcode	**Kommentar**
`$P_UIFR[5]=CROT (X, 0, Y, 0, Z, 0)`	
`N100 $P_UIFR[5, y, rt]=0`	
`N100 $P_UIFR[5, x, rt]=0`	
`N100 $P_UIFR[5, z, rt]=0`	

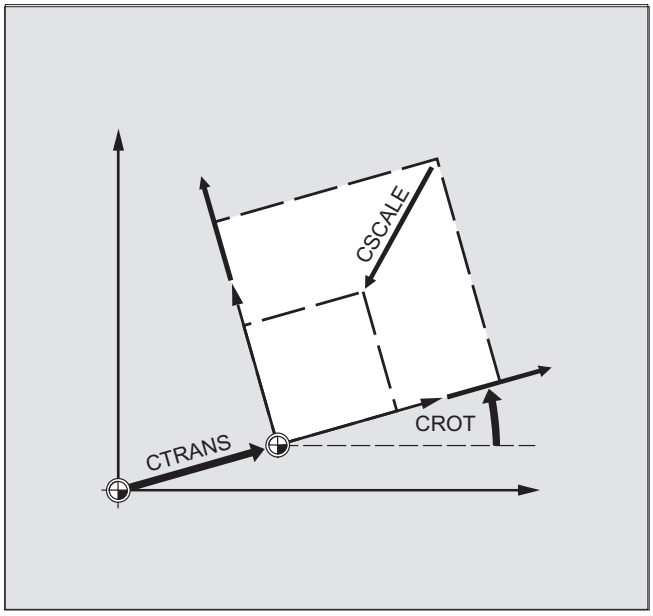
Beschreibung

Sie können mehrere Rechenvorschriften nacheinander programmieren.

Beispiel:

```
$P_PFRAME=CTRANS(...):CROT(...):CSCALE...
```

Beachten Sie, dass die Befehle durch den Kettungsoperator Doppelpunkt (…):(…) miteinander verbunden werden müssen. Dadurch werden die Befehle erstens miteinander verknüpft und zweitens in der programmierten Reihenfolge additiv ausgeführt.



Hinweis

Die mit den genannten Befehlen programmierten Werte werden den Frames zugewiesen und abgespeichert.

Aktiv werden die Werte erst, wenn sie dem Frame einer aktiven Framevariablen `$P_BFRAME` bzw. `$P_PFRAME` zugewiesen werden.

5.2.2 Framekomponenten lesen und verändern (TR, FI, RT, SC, MI)

Funktion

Sie haben die Möglichkeit, auf **einzelne** Daten eines Frames, z. B. auf einen bestimmten Verschiebewert oder Drehwinkel zuzugreifen. Diese Werte können Sie verändern oder einer anderen Variablen zuweisen.

Syntax

`R10=$P_UIFR[$P_UIFNUM,X,RT]`	Der Drehwinkel RT um die X-Achse aus der aktuell gültigen einstellbaren Nullpunktverschiebung $P_UIFRNUM soll der Variablen R10 zugewiesen werden.
`R12=$P_UIFR[25,Z,TR]`	Der Verschiebewert TR in Z aus dem Datensatz des eingestellten Frames Nr. 25 soll der Variablen R12 zugewiesen werden.
`R15=$P_PFRAME[Y,TR]`	Der Verschiebewert TR in Y des aktuellen programmierbaren Frames soll der Variablen R15 zugewiesen werden.
`$P_PFRAME[X,TR]=25`	Der Verschiebewert TR in X des aktuellen programmierbaren Frames soll verändert werden. Ab sofort gilt X25.

Bedeutung

`$P_UIFRNUM`	`Mit dieser Variablen wird automatisch der Bezug zur aktuell gültigen einstellbaren Nullpunktverschiebung hergestellt.`
`P_UIFR[n,…,…]`	`Durch Angabe der Framenummer n greifen Sie auf den einstellbaren Frame Nr. n zu.`
	`Angabe der Komponente, die gelesen oder verändert werden soll:`
`TR`	`TR Translation`
`FI`	`FI Translation Fine`
`RT`	`RT Rotation`
`SC`	`SC Scale Maßstabsveränderung`
`MI`	`MI Spiegelung`
`X Y Z`	`Zusätzlich (siehe Beispiele) wird die entsprechende Achse X, Y, Z angegeben.`

Wertebereich für Drehung RT

Drehung um 1. Geometrieachse:	-180° bis +180°
Drehung um 2. Geometrieachse:	-90° bis +90°
Drehung um 3. Geometrieachse:	-180° bis +180°

Beschreibung

Frame aufrufen

Durch Angabe der Systemvariablen $P_UIFRNUM können Sie direkt auf die mit $P_UIFR bzw. G54, G55, … aktuell eingestellte Nullpunktverschiebung zugreifen ($P_UIFRNUM enthält die Nummer des aktuell eingestellten Frames).

Alle anderen gespeicherten einstellbaren Frames $P_UIFR rufen Sie durch Angabe der entsprechenden Nummer $P_UIFR[n] auf.

Für vordefinierte Framevariable und eigendefinierte Frames geben Sie den Namen an, z. B. $P_IFRAME.

Daten aufrufen

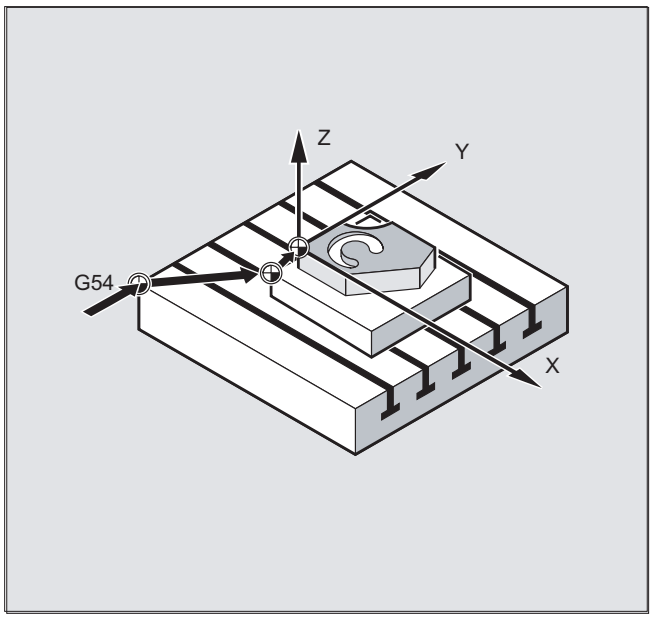
In den eckigen Klammern stehen Achsname und Framekomponente des Wertes, auf den Sie zugreifen oder den Sie verändern wollen, z. B. [X, RT] oder [Z, MI].

5.2.3 Verknüpfung von kompletten Frames

Funktion

Im NC-Programm kann ein kompletter Frame einem anderen Frame zugewiesen oder Frames miteinander verkettet werden.

Framekettungen eignen sich z. B. für die Beschreibung mehrerer Werkstücke, die auf einer Palette angeordnet sind und in einem Fertigungsablauf bearbeitet werden sollen.



Für die Beschreibung von Palettenaufgaben könnten die Framekomponenten z. B. nur bestimmte Teilwerte enthalten, durch deren Verkettung verschiedene Werkstücknullpunkte generiert werden.

Syntax

Frames zuweisen

```
DEF FRAME EINSTELLUNG1                    Dem aktuellen programmierbaren Frame
EINSTELLUNG1=CTRANS(X,10)                 werden die Werte des selbst
$P_PFRAME=EINSTELLUNG1                    definierten Frames EINSTELLUNG1
                                          zugewiesen.
DEF FRAME EINSTELLUNG4
EINSTELLUNG4=$P_PFRAME                    Der aktuelle programmierbare Frame
$P_PFRAME=EINSTELLUNG4                    wird zwischengespeichert und dann
                                          bei Bedarf wieder zurückgespeichert.
```

Frameketten

Die Frames werden in der programmierten Reihenfolge miteinander verkettet, die Framekomponenten wie z. B. Verschiebungen, Drehungen usw. werden nacheinander additiv ausgeführt.

$P_IFRAME=$P_UIFR[15]:$P_UIFR[16]	$P_UIFR[15] enthält z. B. Daten für Nullpunktverschiebungen. Anschließend werden –darauf aufbauend –die Daten von $P_UIFR[16] z. B. Daten für Rotationen verarbeitet.
$P_UIFR[3]=$P_UIFR[4]:$P_UIFR[5]	Der einstellbare Frame 3 wird durch Verkettung der einstellbaren Frames 4 und 5 erzeugt.

Hinweis

Beachten Sie, dass die Frames durch den Kettungsoperator Doppelpunkt **:** miteinander verbunden werden müssen.

5.2.4 Definition neuer Frames (DEF FRAME)

Funktion

Neben den bisher beschriebenen vordefinierten, einstellbaren Frames haben Sie auch die Möglichkeit, neue Frames zu erzeugen. Dabei handelt es sich um Variable vom Typ FRAME, die Sie mit freier Namensgebung definieren.

Mit den Funktionen CTRANS, CROT, CSCALE, CMIRROR können Sie Ihre Frames im NC-Programm mit Werten belegen.

Syntax

```
DEF FRAME PALETTE1
PALETTE1=CTRANS(…):CROT(…)…
```

Bedeutung

`DEF FRAME`	Neue Frames erzeugen.
`PALETTE1`	Name des neuen Frames
`=CTRANS(...):CROT(...)...`	Den möglichen Funktionen Werte zuweisen

5.3 Grob- und Feinverschiebung (CFINE, CTRANS)

Funktion

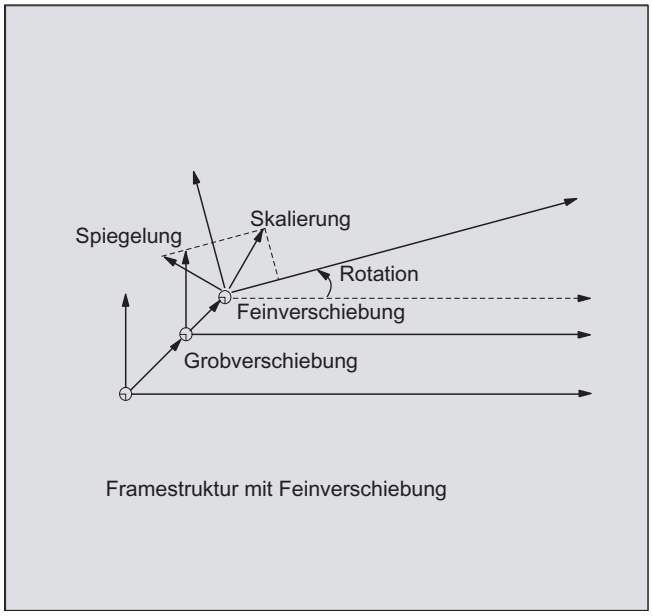
Feinverschiebung

Mit dem Befehl `CFINE(X, ...,Y ...)` kann eine Feinverschiebung des Basisframes und aller einstellbaren Frames programmiert werden.

Eine Feinverschiebung kann nur erfolgen, wenn das MD18600 $MN_MM_FRAME_FINE_TRANS=1 ist.

Grobverschiebung

Mit `CTRANS(...)` wird die Grobverschiebung festgelegt.



Framestruktur mit Feinverschiebung

Grob- und Feinverschiebung addieren sich zur Gesamtverschiebung.

Syntax

```
$P_UBFR=CTRANS(x, 10) : CFINE(x, 0.1) :
CROT(x, 45)                                ;Verkettung von Verschiebung,
                                           ;Feinverschiebung und Rotation

$P_UIFR[1]=CFINE(x, 0.5 y, 1.0, z, 0.1)    ;der gesamte Frame wird mit CFINE
                                           ;einschl. Grobverschiebung
                                           ;überschrieben
```

Der Zugriff auf die Einzelkomponenten der Feinverschiebung erfolgt durch die Komponentenangabe FI (Translation Fine).

```
DEF REAL FINEX                          ;Definition der Variable FINEX
FINEX=$P_UIFR[$P_UIFNUM, x, FI]         ;Auslesen der Feinverschiebung
                                        ;über die Variable FINEX
FINEX=$P_UIFR[3, x, FI]$P               ;Auslesen der Feinverschiebung
                                        ;der X-Achse im 3.Frame
                                        ;über die Variable FINEX
```

Bedeutung

`CFINE(x, Wert, y, Wert, z, Wert )`	Feinverschiebung für mehrere Achsen. Addive Verschiebung (Translatation).
`CTRANS(x, Wert, y, Wert, z, Wert )`	Grobverschiebung für mehrere Achsen. Absolute Verschiebung (Translatation).
`x y z`	Nullpunktverschiebung der Achsen (max. 8)
`Wert`	Translationsanteil

Maschinenhersteller

Mit dem MD18600 $MN_ MM_FRAME_FINE_TRANS kann die Feinverschiebung in folgenden Varianten projektiert werden:

0:
Feinverschiebung kann nicht eingegeben, bzw. nicht programmiert werden. G58 und G59 sind nicht möglich.

1:
Feinverschiebung für einstellbare Frames, Basisframes, programmierbare Frames, G58 und G59 kann eingegeben bzw. programmiert werden.

Beschreibung

Eine über die HMI-Bedienung geänderte Feinverschiebung wird erst nach Aktivierung des entsprechenden Frames aktiv, d. h. die Aktivierung erfolgt über G500, G54...G599. Eine aktivierte Feinverschiebung eines Frames ist solange aktiv, solange der Frame aktiv ist.

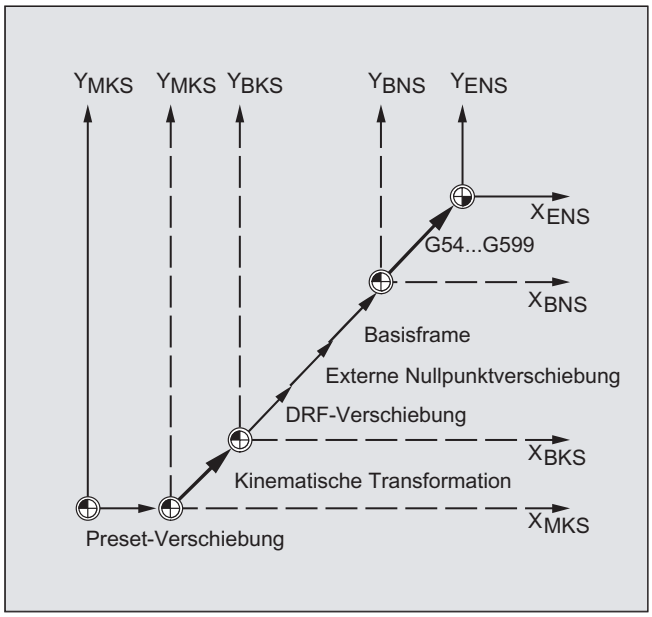
Keinen Feinverschiebungsanteil besitzt der programmierbare Frame. Wird dem programmierbaren Frame ein Frame mit Feinverschiebung zugewiesen, so wird seine Gesamtverschiebung aus der Summe der Grob- und Feinverschiebung gebildet. Beim Lesen des programmierbaren Frames ist die Feinverschiebung immer Null.

5.4 Externe Nullpunktverschiebung

Funktion

Hierdurch haben Sie eine weitere Möglichkeit, den Nullpunkt zwischen Basis- und Werkstückkoordinatensystem zu verschieben.

Bei der externen Nullpunktverschiebung können nur lineare Verschiebungen programmiert werden.



Programmierung

Die Programmierung der Verschiebewerte, $AA_ETRANS erfolgt über die Belegung der achsspezifischen Systemvariablen.

Verschiebewert zuweisen

```
$AA_ETRANS[Achse]=RI
```

RI ist die Rechenvariable vom Typ REAL, die den neuen Wert enthält.

Die externe Verschiebung wird in der Regel nicht im Teileprogramm angegeben, sondern von der PLC gesetzt.

Hinweis

Der im Teileprogramm geschriebene Wert wird erst wirksam, wenn an der VDI-Schnittstelle (NCU-PLC-Schnittstelle) das entsprechende Signal gesetzt ist.

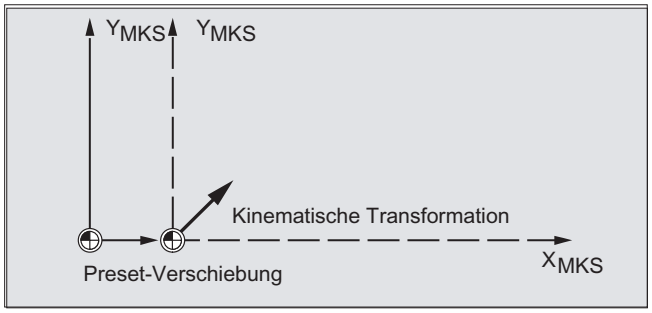
5.5 Preset-Verschiebung (PRESETON)

Funktion

Für spezielle Anwendungen kann es erforderlich werden, einer oder mehreren Achsen an der aktuellen Position (im Stillstand) einen neuen, programmierten Istwert zuzuweisen.

⚠ VORSICHT

Mit der Funktion PRESETON wird der Referenzpunkt ungültig. Deshalb sollten Sie diese Funktion nur für Achsen ohne Referenzpunktpflicht einsetzen. Soll das ursprüngliche System wiederhergestellt werden, müssen mit G74 die Referenzpunkte angefahren werden –siehe Kapitel "Datei- und Programmverwaltung".



Syntax

```
PRESETON(Achse, Wert, ...)
```

Bedeutung

`PRESETON`	Istwertsetzen
`Achse`	Angabe der Maschinenachse
`Wert`	Neuer Istwert, der für die angegebene Achse gelten soll

Hinweis

Das Istwertsetzen mit Synchronaktionen sollte nur mit dem Schlüsselwort "WHEN" oder "EVERY" erfolgen.

Beispiel

Die Zuweisung der Istwerte erfolgt im Maschinenkoordinatensystem –die Werte beziehen sich auf die Maschinenachsen.

```
N10 G0 A760
N20 PRESETON(A1,60)
```

Achse A fährt auf Position 760. Maschinenachse A1 erhält an Position 760 den neuen Istwert 60. Ab jetzt wird im neuen Istwertsystem positioniert.

5.6 Frame-Berechnung aus 3 Messpunkten im Raum (MEAFRAME)

Funktion

MEAFRAME ist eine Erweiterung der 840D-Sprache für die Unterstützung der Messzyklen.

Die Funktion MEAFRAME berechnet den Frame aus drei idealen und den korrespondierenden gemessenen Punkten.

Wird ein Werkstück für die Bearbeitung positioniert, ist seine Position relativ zum kartesischen Maschinenkoordinatensystem bezüglich seiner Idealposition i.a. sowohl verschoben als auch gedreht. Für exakte Bearbeitung oder Messung ist entweder eine kostspielige physikalische Justierung oder Änderung der Bewegungen im Teileprogramm nötig.

Ein Frame kann durch Abtasten dreier Punkte im Raum festgelegt werden, deren Idealpositionen bekannt sind. Abgetastet wird mit einem Berührungs- oder optischen Sensor, der spezielle, auf der Trägerplatte präzise fixierte Löcher oder Messkugeln berührt.

Syntax

```
MEAFRAME IDEAL_POINT,MEAS_POINT,FIT_QUALITY)
```

Bedeutung

`MEAFRAME`	Frame Berechnung von 3 Messpunkten im Raum	
`IDEAL_POINT`	dim. Real-Feld, das die drei Koordinaten der Ideal-Punkte enthält	
`MEAS_POINT`	dim. Real-Feld, das die drei Koordinaten der gemessenen Punkte enthält	
`FIT_QUALITY`	Real-Variable,	mit der folgende Informationen zurückgegeben werden:
	-1:	Die idealen Punkte liegen nahezu auf einer Geraden: Der Frame konnte nicht berechnet werden. Die zurückgegebene Framevariable enthält einen neutralen Frame.
	-2:	Die Messpunkte liegen nahezu auf einer Geraden: Der Frame konnte nicht berechnet werden. Die zurückgegebene Framevariable enthält einen neutralen Frame.
	-4:	Die Berechnung der Rotationsmatrix schlägt aus einem anderen Grund fehl.
	positiver Wert:	Summe der Verzerrungen (Abstände zwischen den Punkten), die zur Überführung des gemessenen Dreiecks in ein zum idealen Dreieck kongruentes benötigt wird.

Hinweis

Qualität der Messung

Damit die gemessenen den idealen Koordinaten mit einer kombinierten Rotation/Translation zugeordnet werden können, muss das von den Messpunkten aufgespannte Dreieck kongruent zum idealen Dreieck sein. Dies wird bewerkstelligt von einem Kompensationsalgorithmus, der die Summe der Quadrate der Abweichungen minimiert, die das gemessene in das ideale Dreieck überführen.

Die effektiv benötigte Verzerrung der Messpunkte kann als Indikator für die Qualität der Messung dienen und wird deshalb als zusätzliche Variable von `MEAFRAME` ausgegeben.

Hinweis

Das von `MEAFRAME` erzeugte Frame kann durch die Funktion `ADDFRAME` in ein anderes Frame in der Framekette transformiert werden.
Siehe Beispiel: Verkettung von Frames "Verkettung mit ADDFRAME".

Weitere Informationen zu den Parametern von `ADDFRAME(FRAME, STRING)` siehe /FB1/ Funktionshandbuch Grundfunktionen; Achsen, Koordinatensysteme, Frames (K2), Kapitel "FRAME-Kettung".

Beispiel

Programmcode	Kommentar
	; Teileprogramm 1
DEF FRAME CORR_FRAME	

Setzen von Messpunkten

Programmierung	Kommentar
DEF REAL IDEAL_POINT[3,3] = SET(10.0,0.0,0.0, 0.0,10.0,0.0, 0.0,0.0,10.0)	
DEF REAL MEAS_POINT[3,3] = SET (10.1,0.2,-0.2, -0.2,10.2,0.1, -0.2,0.2,9.8)	; für Test
DEF REAL FIT_QUALITY = 0	
DEF REAL ROT_FRAME_LIMIT = 5	; erlaubt max. 5 Grad Verdrehung der Teileposition
DEF REAL FIT_QUALITY_LIMIT = 3	; erlaubt max. 3 mm Verschiebung zwischen dem idealen und dem gemessenen Dreieck
DEF REAL SHOW_MCS_POS1[3]	
DEF REAL SHOW_MCS_POS2[3]	
DEF REAL SHOW_MCS_POS3[3]	

```
Programmcode                                          Kommentar
N100 G01 G90 F5000
N110 X0 Y0 Z0
N200 CORR_FRAME=MEAFRAME(IDEAL_POINT,MEAS
_POINT,FIT_QUALITY)

N230 IF FIT_QUALITY < 0
SETAL(65000)
GOTOF NO_FRAME
ENDIF
N240 IF FIT_QUALITY > FIT_QUALITY_LIMIT
SETAL(65010)
GOTOF NO_FRAME
ENDIF
N250 IF CORR_FRAME[X,RT] > ROT_FRAME_LIMIT      ;  Begrenzung des 1. RPY- Winkels
SETAL(65020)
GOTOF NO_FRAME
ENDIF
N260 IF CORR_FRAME[Y,RT] > ROT_FRAME_LIMIT      ;  Begrenzung des 2. RPY-Winkels
SETAL(65021)
GOTOF NO_FRAME
ENDIF
N270 IF CORR_FRAME[Z,RT] > ROT_FRAME_LIMIT      ;  Begrenzung des 3. RPY- Winkels
SETAL(65022)
GOTOF NO_FRAME
ENDIF
N300 $P_IFRAME=CORR_FRAME                       ;  Abtast-Frame mit einem setzbaren
                                                   Frame aktivieren
                                                ;  Frame prüfen durch Positionieren
                                                   der Geometrieachsen auf die
                                                   idealen Punkte
N400 X=IDEAL_POINT[0,0] Y=IDEAL_POINT[0,1]
Z=IDEAL_POINT[0,2]
N410 SHOW_MCS_POS1[0]=$AA_IM[X]
N420 SHOW_MCS_POS1[1]=$AA_IM[Y]
N430 SHOW_MCS_POS1[2]=$AA_IM[Z]
N500 X=IDEAL_POINT[1,0] Y=IDEAL_POINT[1,1]
Z=IDEAL_POINT[1,2]
N510 SHOW_MCS_POS2[0]=$AA_IM[X]
N520 SHOW_MCS_POS2[1]=$AA_IM[Y]
N530 SHOW_MCS_POS2[2]=$AA_IM[Z]
N600 X=IDEAL_POINT[2,0] Y=IDEAL_POINT[2,1]
Z=IDEAL_POINT[2,2]
N610 SHOW_MCS_POS3[0]=$AA_IM[X]
```

Programmcode	Kommentar
N620 SHOW_MCS_POS3[1]=$AA_IM[Y]	
N630 SHOW_MCS_POS3[2]=$AA_IM[Z]	
N700 G500	; Setzbaren Frame deaktivieren, da mit Nullframe (kein Wert eingetragen vorbesetzt).
No_FRAME	; Setzbaren Frame deaktivieren, da mit Nullframe (kein Wert eingetragen) vorbesetzt
M0	
M30	

Beispiel Verkettung von Frames

Verkettung von MEAFRAME für Korrekturen

Die Funktion `MEAFRAME( )` liefert ein Korrekturframe. Wird dieser Korrekturframe mit dem einstellbaren Frame `$P_UIFR[1]` verkettet, der bei Aufruf der Funktion aktiv war z. B. `G54`, so erhält man ein einstellbaren Frame für weitere Umrechnungen zum Verfahren oder Bearbeiten.

Verkettung mit ADDFRAME

Soll dieser Korrekturframe in der Framekette an einer anderen Stelle wirken oder sind vor dem einstellbaren Frame noch andere Frames aktiv, dann kann die Funktion `ADDFRAME( )` zum Einketten in einem der Kanal-Basisframes oder einem Systemframe genutzt werden.

In den Frames darf hierbei nicht aktiv sein:

- Spiegelung mit `MIRROR`
- Skalierung mit `SCALE`

Die Eingangsparameter für Soll- und Istwerte sind die Werkstückkoordinaten. Im Grundsystem der Steuerung sind diese Koordinaten stets

- metrisch oder in Inch `(G71/G70)` und als
- radiusbezogenes `(DIAMOF)`

Maß anzugeben.

5.7 NCU-globale Frames

Funktion

NCU-globale Frames gibt es pro NCU nur einmal für alle Kanäle. NCU-globale Frames können von allen Kanälen aus geschrieben und gelesen werden. Die Aktivierung der NCU-globalen Frames erfolgt im jeweiligen Kanal.

Durch globale Frames können **Kanalachsen und Maschinenachsen** mit Verschiebungen, skaliert und gespiegelt werden.

Geometrische Zusammenhänge und Frameketten

Bei globalen Frames existiert kein geometrischer Zusammenhang zwischen den Achsen. Deshalb können keine Drehungen und keine Programmierung von Geometrie-Achsbezeichnern ausgeführt werden.

- Auf globale Frames lassen sich keine Rotationen anwenden. Die Programmierung einer Rotation wird mit dem Alarm: "18310 Kanal %1 Satz %2 Frame: Rotation unzulässig", abgelehnt.
- Die Verkettung von globalen Frames und kanalspezifischen Frames ist möglich. Der resultierende Frame enthält alle Frameanteile inklusive der Rotationen für alle Achsen. Die Zuweisung eines Frames mit Rotationsanteilen an einen globalen Frame wird mit dem Alarm "Frame: Rotation unzulässig" abgelehnt.

NCU-globale Frames

NCU-globale Basisframes $P_NCBFR[n]

Es können bis zu 8 NCU-globale Basisframes projektiert werden:

Gleichzeitig können kanalspezifische Basisframes vorhanden sein.

Globale Frames können von allen Kanälen einer NCU geschrieben und gelesen werden. Beim Schreiben von globalen Frames ist vom Anwender für eine Kanalkoordinierung Sorge zu tragen. Dies kann z.B. durch Wait-Marken `(WAITMC)` realisiert werden.

Maschinenhersteller

Die Anzahl von globalen Basisframes wird über Maschinendaten projektiert siehe /FB1/ Funktionshandbuch Grundfunktionen; Achsen, Koordinatensysteme, Frames (K2).

NCU-globale Einstellbare Frames $P_UIFR[n]

Alle einstellbaren Frames `G500, G54...G599` können entweder NCU-global oder kanalspezifisch projektiert werden.

Maschinenhersteller

Alle einstellbaren Frames können mit Hilfe eines Maschinendatums $MN_MM_NUM_GLOBAL_USER_FRAMES zu globalen Frames umprojektiert werden.

Als Achsbezeichner bei den Frame-Programmbefehlen können Kanalachsbezeichner und Maschinenachsbezeichner verwendet werden. Die Programmierung von Geometrieachsbezeichnern wird mit einem Alarm abgelehnt.

5.7.1 Kanalspezifische Frames ($P_CHBFR, $P_UBFR)

Funktion

Einstellbare Frames oder Basisframes können

- über das Teileprogramm und
- über BTSS

von der Bedienung z. B. HMI Advanced und von der PLC geschrieben und gelesen werden.

Die Feinverschiebung ist auch für die globalen Frames möglich. Die Unterdrückung von globalen Frames erfolgt ebenso, wie bei kanalspezifischen Frames über `G53`, `G153`, `SUPA` und `G500`.

Maschinenhersteller

Über das MD28081 MM_NUM_BASE_FRAMES kann die Anzahl der Basisframes im Kanal projektiert werden. Die Standardkonfiguration ist so ausgelegt, dass es mindestens ein Basisframe pro Kanal gibt. Maximal sind 8 Basisframes pro Kanal möglich. Zusätzlich zu den 8 Basisframes im Kanal kann es noch 8 NCU-globale Basisframes geben.

Kanalspezifische Frames

$P_CHBFR[n]

Über die Systemvariable `$P_CHBFR[n]` können die Basisframes gelesen und geschrieben werden. Beim Schreiben eines Basisframes wird der verkettete Gesamt-Basisframe nicht aktiviert, sondern die Aktivierung erfolgt erst mit der Ausführung einer `G500`, `G54...G599`-Anweisung. Die Variable dient vorwiegend als Speicher für Schreibvorgänge auf das Basisframe von HMI oder PLC. Diese Frame-Variablen werden über die Datensicherung gesichert.

Erster Basisframe im Kanal

Ein Schreiben auf die vordefinierte Variable `$P_UBFR` aktiviert den Basisframe mit dem Feldindex 0 nicht gleichzeitig, sondern die Aktivierung erfolgt erst mit der Ausführung einer `G500`, `G54...G599`-Anweisung. Die Variable kann auch im Programm geschrieben und gelesen werden.

$P_UBFR

`$P_UBFR` ist identisch mit `$P_CHBFR[0]`. Standardmäßig gibt es immer einen Basisframe im Kanal, so dass die Systemvariable kompatibel zu älteren Ständen ist. Gibt es kein kanalspezifisches Basisframe, wird beim Schreiben oder Lesen der Alarm "Frame: Anweisung unzulässig" ausgegeben.

5.7.2 Im Kanal wirksame Frames

Funktion

Im Kanal wirksame Frames werden vom Teileprogramm über die betreffenden Systemvariablen dieser Frames eingegeben. Hierzu gehören auch Systemframes. Über diese Systemvariablen kann im Teileprogramm das aktuelle Systemframe gelesen und geschrieben werden.

Aktuelle im Kanal wirksame Frames

Übersicht

Aktuelle Systemframes	für:
$P_PARTFRAME	TCARR und PAROT
$P_SETFRAME	Istwertsetzen und Ankratzen
$P_EXTFRAME	Externe Nullpunktverschiebung
$P_NCBFRAME[n]	Aktuelle NCU-globale Basisframes
$P_CHBFRAME[n]	Aktuelle Kanal-Basisframes
$P_BFRAME	Aktueller 1. Basisframe im Kanal
$P_ACTBFRAME	Gesamt-Basisframe
$P_CHBFRMASK und $P_NCBFRMASK	Gesamt-Basisframe
$P_IFRAME	Aktueller einstellbarer Frame
Aktuelle Systemframes	für:
$P_TOOLFRAME	TOROT und TOFRAME
$P_WPFRAME	Werkstückbezugspunkte
$P_TRAFRAME	Transformationen
$P_PFRAME	Aktueller programmierbarer Frame
Aktuelles Systemframe	für:
$P_CYCFRAME	Zyklen
P_ACTFRAME	Aktueller Gesamtframe
FRAME-Kettung	Aktuelles Frame setzt sich aus dem Gesamt-Basisframe zusammen

$P_NCBFRAME[n] Aktuelle NCU-globale Basisframes

Über die Systemvariable `$P_NCBFRAME[n]` können die aktuellen globalen Basisframe-Feldelemente gelesen und geschrieben werden. Das resultierende Gesamt-Basisframe wird durch den Schreibvorgang im Kanal eingerechnet.

Der geänderte Frame wird nur in dem Kanal, in dem der Frame programmiert wurde, aktiv. Soll der Frame für alle Kanäle einer NCU geändert werden, muss gleichzeitig $P_NCBFR[n] und $P_NCBFRAME[n] beschrieben werden. Die anderen Kanäle müssen dann noch den Frame mit z. B. G54 aktivieren. Beim Schreiben eines Basisframes wird der Gesamt-Basisframe neu berechnet.

$P_CHBFRAME[n] Aktuelle Kanal-Basisframes

Über die Systemvariable `$P_CHBFRAME[n]` können die aktuellen Kanal-Basisframe-Feldelemente gelesen und geschrieben werden. Der resultierende Gesamt-Basisframe wird durch den Schreibvorgang im Kanal eingerechnet. Beim Schreiben eines Basisframes wird der Gesamt-Basisframe neu berechnet.

$P_BFRAME Aktueller 1. Basisframe im Kanal

Über die vordefinierte Framevariable `$P_BFRAME` kann der aktuelle Basisframe mit dem Feldindex 0, der im Kanal gültig ist, im Teileprogramm gelesen und geschrieben werden. Der geschriebene Basisframe wird sofort eingerechnet.

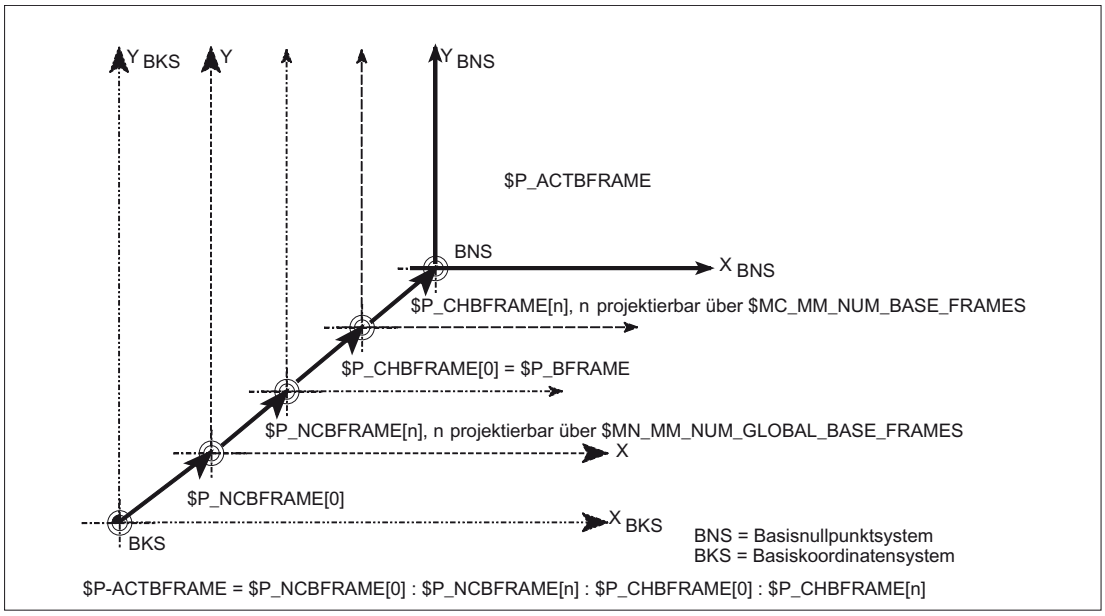
`$P_BFRAME` ist identisch mit `$P_CHBFRAME[0]`. Die Systemvariable hat standardmäßig immer einen gültigen Wert. Gibt es kein kanalspezifisches Basisframe, wird beim Schreiben oder Lesen der Alarm "Frame: Anweisung unzulässig" ausgegeben.

$P_ACTBFRAME Gesamt-Basisframe

Die Variable `$P_ACTFRAME` ermittelt das verkettete Gesamt-Basisframe. Die Variable ist nur lesbar.

`$P_ACTFRAME` entspricht

`$P_NCBFRAME[0]` : ... : `$P_NCBFRAME[n]` : `$P_CHBFRAME[0]` : ... : `$P_CHBFRAME[n]`.



$P_CHBFRMASK und $P_NCBFRMASK Gesamt-Basisframe

Über die Systemvariable `$P_CHBFRMASK` und `$P_NCBFRMASK` kann der Anwender auswählen, welche Basisframes er in die Berechnung des "Gesamt"-Basisframes mit einbeziehen möchte. Die Variablen können nur im Programm programmiert werden und über BTSS gelesen werden. Der Wert der Variablen wird als Bitmaske interpretiert und gibt an, welches Basisframe-Feldelement von `$P_ACTFRAME` in die Berechnung einfließt.

Mit `$P_CHBFRMASK` kann vorgegeben werden, welche kanalspezifischen Basisframes, und mit `$P_NCBFRMASK`, welche NCU-globalen Basisframes eingerechnet werden.

Mit der Programmierung der Variablen wird der Gesamt-Basisframe und der Gesamt-Frame neu berechnet. Nach Reset und in der Grundeinstellung ist der Wert von

`$P_CHBFRMASK = $MC_CHBFRAME_RESET_MASK` und

`$P_NCBFRMASK = $MC_CHBFRAME_RESET_MASK`.

z. B.

```
$P_NCBFRMASK = 'H81' ;$P_NCBFRAME[0] : $P_NCBFRAME[7]
$P_CHBFRMASK = 'H11' ;$P_CHBFRAME[0] : $P_CHBFRAME[4]
```

$P_IFRAME Aktueller einstellbarer Frame

Über die vordefinierte Framevariable `$P_IFRAME` kann der aktuelle einstellbare Frame, welcher im Kanal gültig ist, im Teileprogramm gelesen und geschrieben werden. Der geschriebene einstellbare Frame wird sofort eingerechnet.

Bei NCU-globalen einstellbaren Frames wirkt der geänderte Frame nur in dem Kanal, in dem der Frame programmiert wurde. Soll der Frame für alle Kanäle einer NCU geändert werden, muss gleichzeitig `$P_UIFR[n]` und `$P_IFRAME` beschrieben werden. Die anderen Kanäle müssen dann noch den entsprechenden Frame mit z. B. G54 aktivieren.

$P_PFRAME Aktueller programmierbarer Frame

`$P_PFRAME` ist der programmierbare Frame, der sich aus der Programmierung von `TRANS/ATRANS`, `G58/G59`, `ROT/AROT`, `SCALE/ASCALE`, `MIRROR/AMIRROR` bzw. aus der Zuweisung von `CTRANS`, `CROT`, `CMIRROR`, `CSCALE` an den programmierbaren FRAME ergibt.

Aktuelle, programmierbare Framevariable, die den Bezug zwischen dem Einstellbaren

- Nullpunktsystem (ENS) und dem
- Werkstückkoordinatensystem (WKS)

herstellt.

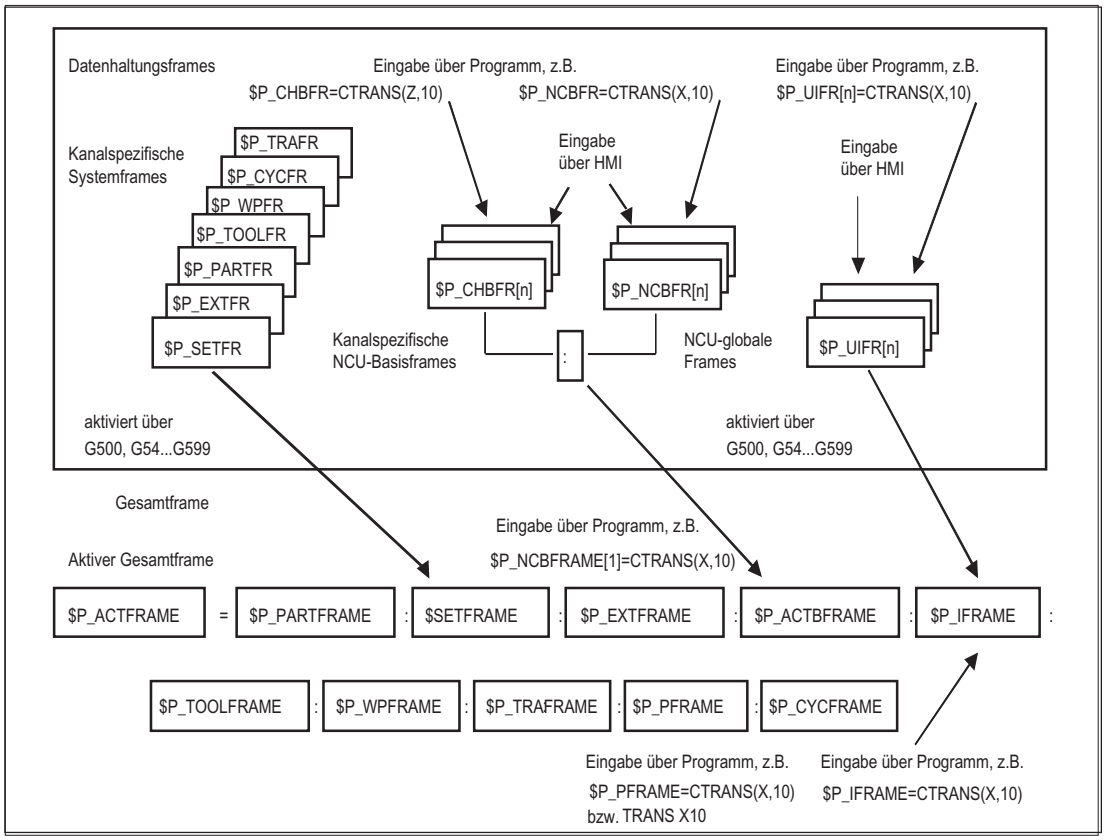
P_ACTFRAME Aktueller Gesamtframe

Der aktuelle resultierende Gesamtframe $P_ACTFRAME ergibt sich nun als Verkettung aller Basisframes, dem aktuellen einstellbaren Frame und dem programmierbaren Frame. Der aktuelle Frame wird immer dann aktualisiert, wenn sich ein Frameanteil ändert.

`$P_ACTFRAME` entspricht

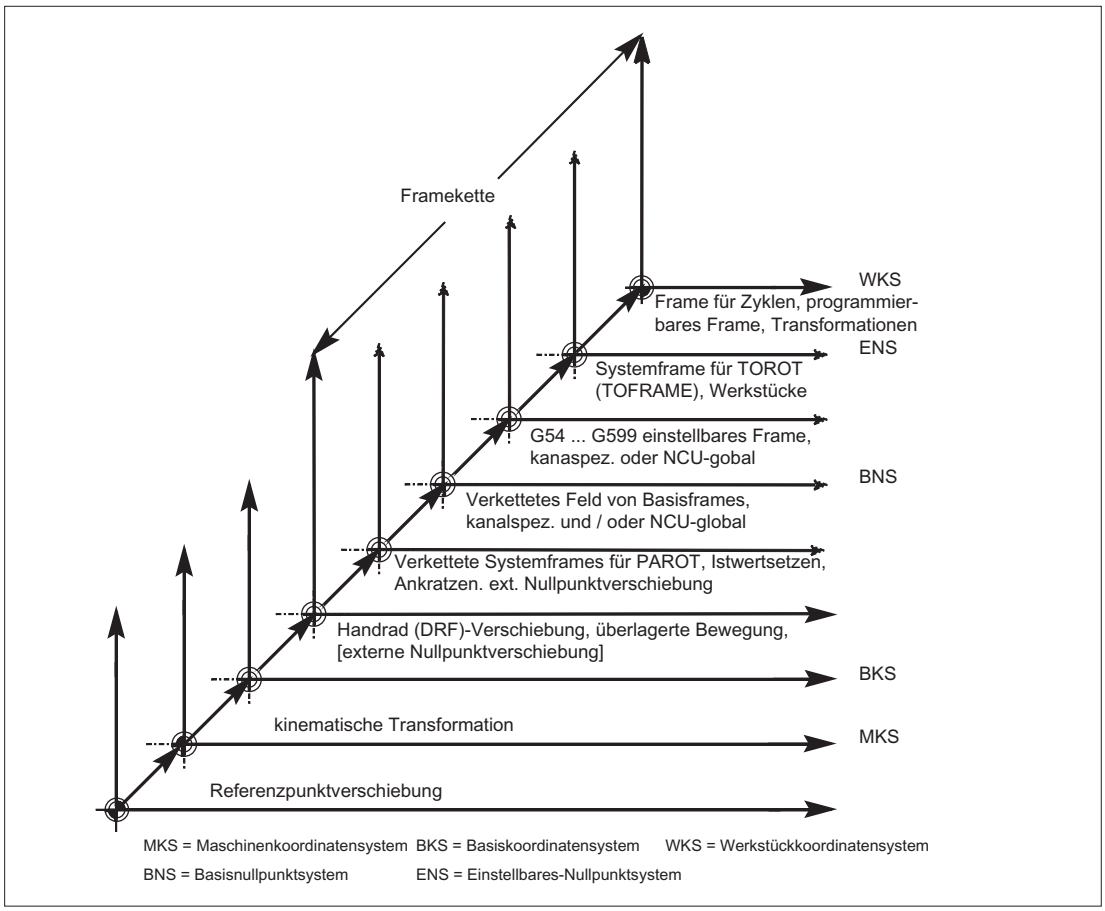
`$P_PARTFRAME : $P_SETFRAME : $P_EXTFRAME : $P_ACTBFRAME : $P_IFRAME :`

`$P_TOOLFRAME : $P_WPFRAME : $P_TRAFRAME : $P_PFRAME : $P_CYCFRAME`



Frame-Kettung

Der aktuelle Frame setzt sich aus dem Gesamt-Basisframe, dem einstellbaren Frame, dem Systemframe und dem programmierbaren Frame gemäß oben angegebenen aktuellen Gesamtframe zusammen.



Transformationen 6

6.1 Allgemeine Programmierung der Transformationsarten

Allgemeine Funktion

Zur Anpassung der Steuerung an verschiedene Maschinenkinematiken besteht die Auswahl Transformationsarten mit geeigneten Parametern zu programmieren. Über diese Parameter kann für die ausgewählte Transformation sowohl die Orientierung des Werkzeugs im Raum als auch die Orientierungsbewegungen der Rundachsen entsprechend vereinbart werden.

Bei den Drei-, Vier- und Fünf-Achs-Transformationen beziehen sich die programmierten, Positionsangaben immer auf die Spitze des Werkzeugs, welches orthogonal zur im Raum befindlichen Bearbeitungsfläche nachgeführt wird. Die kartesischen Koordinaten werden vom Basiskoordinatensystem ins Maschinenkoordinatensystem umgerechnet und beziehen sich auf die Geometrieachsen. Diese beschreiben den Arbeitspunkt. Virtuelle Rundachsen beschreiben die Orientierungen des Werkzeugs im Raum und werden mit TRAORI programmiert.

Bei der kinematischen Transformation können Positionen im kartesischen Koordinatensystem programmiert werden. Die Steuerung transformiert die mit TRANSMIT, TRACYL und TRAANG programmierten Verfahrbewegungen des kartesischen Koordinatensystems auf die Verfahrbewegungen der realen Maschinenachsen.

Programmierung

Drei-, Vier- und Fünf-Achs-Transformationen TRAORI

Die vereinbarte Orientierungstransformation wird mit dem Befehl TRAORI und den drei möglichen Parametern für Trafonummer, Orientierungsvektor und Rundachsoffsets aktiviert.

```
TRAORI(Trafonummer, Orientierungsvektor, Rundachsoffsets)
```

Kinematische Transformationen

Zu den Kinematischen Transformationengehören die vereinbarten Transformationen

```
TRANSMIT(Trafonummer)
TRACYL(Arbeitsdurchmesser, Trafonummer)
TRAANG(Winkel der schräg stehenden Achse, Trafonummer)
```

Aktive Transformation ausschalten

Mit `TRAFOOF` kann die gerade aktive Transformation ausgeschaltet werden.

Orientierungstransformation

Drei-, Vier- und Fünf- Achs-Transformationen TRAORI

Zur optimalen Bearbeitung räumlich geformter Flächen im Arbeitsraum der Maschine, benötigen Werkzeugmaschinen außer den drei Linearachsen X, Y und Z noch zusätzliche Achsen. Die zusätzlichen Achsen beschreiben die Orientierung im Raum und werden nachfolgend Orientierungsachsen genannt. Sie stehen als Drehachsen bei vier Maschinentypen mit verschiedener Kinematik zur Verfügung.

1. Zweiachsen-Schwenkkopf, z. B. Kardanischer Werkzeugkopf mit einer Rundachse parallel zu einer Linearachse bei festem Werkzeugtisch.
2. Zweiachsen-Drehtisch, z. B. fester Schwenkkopf mit drehbarem Werkzeugtisch um zwei Achsen.
3. Einachs-Schwenkkopf und Einachs-Drehtisch, z. B. ein drehbarer Schwenkkopf mit gedrehtem Werkzeug bei drehbarem Werkzeugtisch um eine Achse.
4. Zweiachsen-Schwenkkopf und Einachs-Drehtisch, z. B. bei drehbarem Werkzeugtisch um eine Achse und ein drehbarer Schwenkkopf mit drehbarem Werkzeug um sich selbst.

Die **3- und 4-Achs-Transformationen** sind Sonderformen der 5-Achs-Transformation und werden analog zu den 5-Achs-Transformationen programmiert.

Die "**Generische 3-/4-/5-/6-Achs-Transformation**" deckt mit ihrem Funktionsumfang für rechtwinklig angeordnete Rundachsen sowie die Transformationen für den Kardanischen Fräskopf ab und kann wie jede andere Orientierungstransformation auch für diese vier Maschinentypen mit TRAORI aktiviert werden. Bei der generischen 5/6-Achs-Transformationhat die Werkzeugorientierung einen weiteren dritten Freiheitsgrad, bei dem zur Werkzeugrichtung beliebig im Raum, das Werkzeug um die eigene Achse gedreht werden kann.

Literatur: /FB3/ Funktionshandbuch Sonderfunktionen; 3- bis 5-Achstransformation (F2)

Kinematikunabhängige Grundstellung der Werkzeugorientierung

ORIRESET

Ist mit TRAORI eine Orientierungstransformation aktiv, dann können mit ORIRESET die Grundstellungen von bis zu 3 Orientierungsachsen mit optionalen Parametern A, B, C angegeben werden. Die Zuordnung der Reihenfolge der programmierten Parameter zu den Rundachsen erfolgt gemäß der durch die Transformation festgelegten Reihenfolge der Orientierungsachsen. Die Programmierung von ORIRESET(A, B, C) bewirkt, dass die Orientierungsachsen linear und synchron von ihrer momentanen Position zu der angegebenen Grundstellungsposition fahren.

Kinematische Transformationen

TRANSMIT und TRACYL

Bei Fräsbearbeitungen an Drehmaschinen kann für die vereinbarte Transformation entweder

1. eine stirnseitige Bearbeitung in der Drehaufspannung mit TRANSMIT oder
2. eine Bearbeitung von beliebig verlaufenden Nuten an zylindrischen Körpern mit TRACYL

programmiert werden.

TRAANG

Soll die Zustellachse z. B. für die Technologie Schleifen auch schräg zustellbar sein, so kann mit TRAANG für die vereinbarte Transformation ein parametrierbarer Winkel programmiert werden.

Kartesisches PTP-Fahren

Zur kinematischen Transformation gehört auch das "Kartesisches PTP-Fahren" bei dem bis zu 8 unterschiedliche Gelenkstellungen STAT= programmiert werden können. Die Positionen werden im kartesischen Koordinatensystem programmiert, wobei die Bewegung der Maschine in Maschinenkoordinaten erfolgt.

Literatur:
/FB2/ Funktionshandbuch Erweiterungsfunktionen; Kinematische Transformation (M1)

Verkettete Transformationen

Es können jeweils zwei Transformationen hintereinander geschaltet werden. Bei der hierdurch verketteten zweiten Transformation werden die Bewegungsanteile der Achsen aus der ersten Transformation übernommen.

Als erste Transformation sind möglich:

- Orientierungstransformation TRAORI
- Polartransformation TRANSMIT
- Zylindertransformation TRACYL
- Transformation Schräge Achse TRAANG

Die zweite Transformation muss Schräge Achse TRAANG sein

6.1.1 Orientierungsbewegungen bei den Transformationen

Verfahrbewegungen und Orientierungsbewegungen

Die Verfahrbewegungen der programmierbaren Orientierungen hängen primär vom Maschinentyp ab. Bei der Drei-, Vier- und Fünf-Achs-Transformation mit TRAORI beschreiben die rotatorischen Achsen oder die schwenkbaren Linearachsen die Orientierungsbewegungen des Werkzeugs.

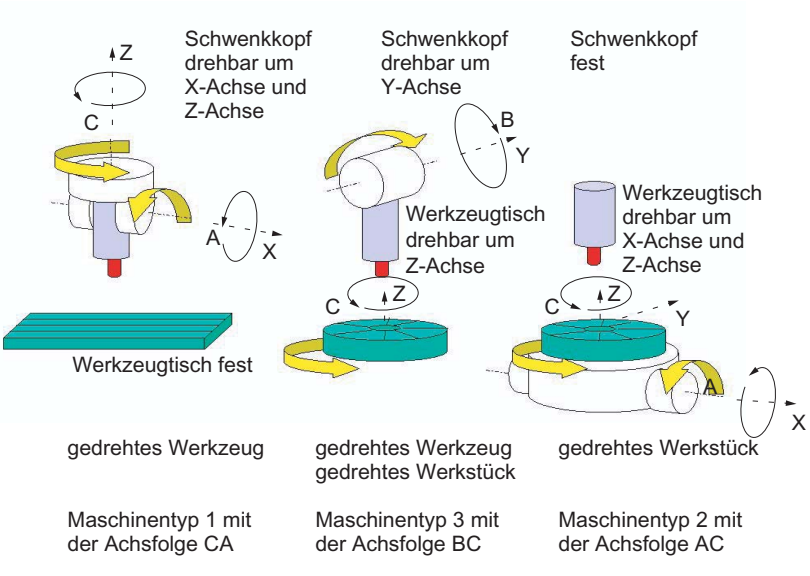
Änderungen der Positionen der an der Orientierungstransformation beteiligten Rundachsen führen zu Ausgleichsbewegungen der übrigen Maschinenachsen. Die Position der Werkzeugspitze bleibt dabei unverändert.

Orientierungsbewegungen des Werkzeugs können über die Rundachsbezeichner A…, B…, C… der virtuellen Achsen je nach Anwendung entweder durch Angabe von Euler- bzw. RPY-Winkeln oder Richtungs- bzw. Flächennormalenvektoren, Normierte Vektoren für die Drehachse eines Kegels oder für die Zwischenorientierung auf einer Kegelmantelfläche programmiert werden.

Bei der Kinematischen Transformation mit TRANSMIT, TRACYL und TRAANG transformiert die Steuerung die programmierten Verfahrbewegungen des kartesischen Koordinatensystems auf die Verfahrbewegungen der realen Maschinenachsen.

Maschinenkinematik bei Drei-, Vier- und Fünf-Achs-Transformation TRAORI

Es kann entweder das Werkzeug oder der Werkzeugtisch mit bis zu zwei Rundachsen drehbar sein. Eine Kombination von jeweils einachsigem Schwenkkopf und Drehtisch ist auch möglich.



Maschinentyp	Programmierung der Orientierung
Drei-Achs-Transformation Maschinentypen 1 und 2	Programmierung der Werkzeugorientierung nur in der Ebene, die senkrecht zu der rotatorischen Achse ist. Es existieren **zwei** translatorischen Achsen (Linearachsen) und **einer** rotatorischen Achse (Rundachse).
Vier-Achs-Transformation Maschinentypen 1 und 2	Programmierung der Werkzeugorientierung nur in der Ebene, die senkrecht zu der rotatorischen Achse ist. Es existieren **drei** translatorischen Achsen (Linearachsen) und **einer** rotatorischen Achse (Rundachse).
Fünf-Achs-Transformation Maschinentypen 3 Einachs-Schwenkkopf und Einachs-Drehtisch	Programmierung der Orientierungstransformation. Kinematik mit **drei** Linearachsen und **zwei** orthogonalen Rundachsen. Die Rundachsen sind parallel zu zwei der drei Linearachsen. Die erste Rundachse wird von zwei kartesischen Linearachsen bewegt. Sie dreht die dritte Linearachse mit dem Werkzeug. Die zweite Rundachse dreht das Werkstück.

Generische 5/6-AchsTransformationen

Maschinentyp	Programmierung der Orientierungstransformation
Generische Fünf-/Sechs-Achs Transformation Maschinentypen 4 Zweiachs-Schwenkkopf mit drehbarem Werkzeug um sich selbst und Einachs-Drehtisch	Programmierung der Orientierungstransformation. Kinematik mit **drei** Linearachsen und **drei** orthogonalen Rundachsen. Die Rundachsen sind parallel zu zwei der drei Linearachsen. Die erste Rundachse wird von zwei kartesischen Linearachsen bewegt. Sie dreht die dritte Linearachse mit dem Werkzeug. Die zweite Rundachse dreht das Werkstück. Die Grundorientierung des Werkzeugs kann durch eine zusätzliche Drehung um sich selbst mit dem Drehwinkel THETA programmiert werden.

Beim Aufruf der "Generischen Drei-, Vier- und Fünf-/Sechs-Achs Transformation" kann zusätzlich die Grundorientierung des Werkzeugs übergeben werden. Es gelten die Einschränkungen bezüglich der Richtungen der Rundachsen nicht mehr. Wenn die Rundachsen nicht exakt senkrecht aufeinander stehen oder vorhandene Rundachsen nicht exakt parallel zu den Linearachsen stehen, kann die "Generische Fünf-/Sechs-Achs Transformation" bessere Ergebnisse der Werkzeugorientierung liefern.

Kinematische Transformationen TRANSMIT, TRACYL und TRAANG

Für Fräsbearbeitungen an Drehmaschinen oder einer schräg zustellbaren Achse beim Schleifen gelten abhängig von der Transformation im Standardfall folgende Achsanordnungen:

TRANSMIT	Aktivierung der Polar-Transformation
stirnseitige Bearbeitung in der Drehaufspannung	eine Rundachse eine Zustellachse senkrecht zur Drehachse eine Längsachse parallel zur Drehachse

TRACYL	Aktivierung der Zylindermanteltransformation
Bearbeitung von beliebig verlaufenden Nuten an den zylindrischen Körper	eine Rundachse eine Zustellachse senkrecht zur Drehachse eine Längsachse parallel zur Drehachse

TRAANG	Aktivierung der Transformation Schräge Achse
Bearbeitung mit schräger Zustellachse	eine Rundachse eine Zustellachse mit parametrierbaren Winkel eine Längsachse parallel zur Drehachse

Kartesisches PTP-Fahren

Die Bewegung der Maschine erfolgt in Maschinenkoordinaten und wird programmiert mit:

TRAORI	Aktivierung der Transformation
PTP Punkt zu Punkt fahren	Position im kartesischen Koordinatensystem (MKS) anfahren
CP	Bahnbewegung der kartesischen Achsen im (BKS)
STAT	Stellung der Gelenke ist abhängig von der Transformation
TU	Um welchen Winkel die Achsen auf den kürzesten Weg verfahren

PTP-Fahren bei generischer 5/6-AchsTransformation

Die Bewegung der Maschine erfolgt in Maschinenkoordinaten und die Werkzeugorientierung kann sowohl mit Rundachspositionen als auch mit von der Kinematik unabhängigen Vektoren Euler bzw. RPY-Winkel oder den Richtungsvektoren programmiert werden.

Dabei sind Rundachsinterpolation, Vektoreninterpolation mit Großkreisinterpolation oder Interpolation des Orientierungsvektors auf einer Kegelmantelfläche möglich.

Beispiel Drei-, bis Fünf-Achs-Transformation bei einen Kardanischen Fräskopf

Die Werkzeugmaschine hat mindestens 5 Achsen, davon

- Drei translatorische Achsen für geradlinige Bewegungen, die den Arbeitspunkt an jede beliebige Position im Arbeitsaum bewegen.
- Zwei rotatorische Schwenkachsen, die unter einem projektierbaren Winkel (meist 45 Grad) angeordnet sind, ermöglichen dem Werkzeug Orientierungen im Raum einzunehmen, die sich bei 45 Grad Anordnung auf eine Halbkugel beschränken.

6.1.2 Übersicht der Orientierungstransformation TRAORI

Mögliche Programmierungsarten im Zusammenhang mit TRAORI

Maschinentyp	Programmierung bei aktiver Transformation TRAORI
Maschinentypen 1, 2 oder 3 Zweiachs-Schwenkkopf oder Zweiachs-Drehtisch oder eine Kombination von jeweils einachsigem Schwenkkopf und Drehtisch.	Achsfolge der Orientierungsachsen und die Orientierungsrichtung des Werkzeugs ist und entweder **maschinenbezogen** projektierbar über Maschinendaten abhängig von der Maschinenkinemtik oder **werkstückbezogen** mit programmierbarer Orientierung unabhängig von der Maschinenkinemtik Die Drehrichtungen der Orientierungsachsen im Bezugssystem wird programmiert mit: - ORIMKS Bezugssystem = Maschinenkoordinatensystem - ORIWKS Bezugssystem = Werkstückkoordinatensystem Die Grundeinstellung ist ORIWKS. Programmierung der Orientierungsachsen mit: A, B, C der Maschinenachspositionen direkt A2, B2, C2 Winkelprogrammierung virtueller Achsen mit - ORIEULER über Euler-Winkel (Standard) - ORIRPY über RPY-Winkel - ORIVIRT1 über virtuelle Orientierungsachsen 1. Definition - ORIVIRT2 über virtuelle Orientierungsachsen 2. Definition mit Unterscheidung der Interpolationsart: **lineare Interpolation** - ORIAXES von Orientierungsachsen oder Maschinenachsen **Großkreisinterpolation** (Interpolation des Orientierungsvektors) - ORIVECT von Orientierungsachsen Programmierung der Orientierungsachsen durch Angabe A3, B3, C3 der Vektorkomponenten (Richtung-/Flächennormale) Programmierung der resultierenden Werkzeugorientierung A4, B4, C4 des Flächennormalvektors am Satzanfang A5, B5, C5 des Flächennormalvektors am Satzende LEAD Voreilwinkel für die Werkzeugorientierung TILT Seitwärtswinkel für die Werkzeugorientierung

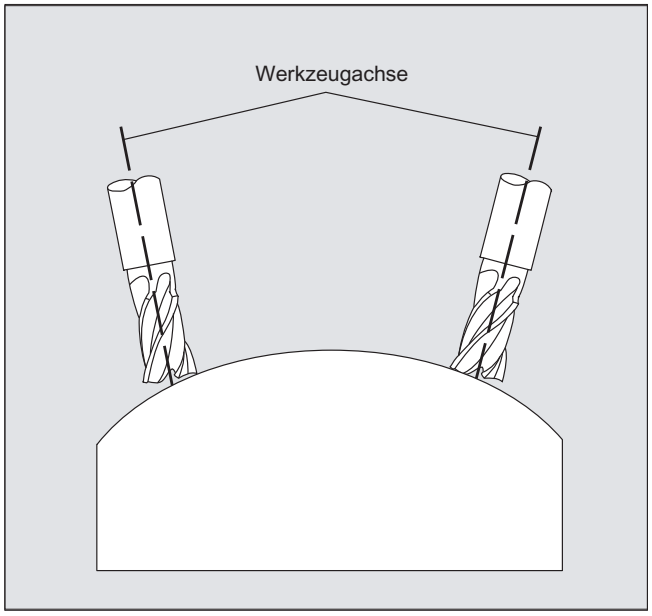
Maschinentyp	Programmierung bei aktiver Transformation TRAORI
	Interpolation des Orientierungsvektors auf einer Kegelmantelfläche Orientierungsänderungen auf einer **beliebig im Raum** befindlichen Kegelmantelfläche durch Interpolation: - ORIPLANE in der Ebene (Großreisinterpolation) - ORICONCW auf einer Kegelmantelfläche im Uhrzeigersinn - ORICONCCW auf einer Kegelmantelfläche gegen Uhrzeigersinn A6, B6, C6 Richtungsvektors (Drehachse des Kegels) -OICONIO Interpolation auf einer Kegelmantelfläche mit: A7, B7, C7 Zwischenvektoren (Start- und Endorientierung) oder - ORICONTO auf einer Kegelmantelfläche tangentialer Übergang Orientierungsänderungen bezogen **auf eine Bahn** mit - ORICURVE Vorgabe der Bewegung zweier Kontaktpunkte über PO[XH]=(xe, x2, x3, x4, x5) Orientierungspolynome bis 5.Grades PO[YH]=(ye, y2, y3, y4, y5) Orientierungspolynome bis 5.Grades PO[ZH]=(ze, z2, z3, z4, z5) Orientierungspolynome bis 5.Grades - ORIPATHS Glättung des Orientierungsverlaufs mit A8, B8, C8 Umorientierungsphase des Werkzeugs entspricht: Richtung und Weglänge des Werkzeugs bei der Abhebebewegung
Maschinentypen 1 und 3 Weitere Maschinentypen mit zusätzlicher Drehung des Werkzeugs um sich selbst erfordern eine 3. Rundachse Orientierungstransformation, wie z.B. generische 6-Achs-Transformation. Drehungen des Orientierungsvektors.	Programmierung **der Drehungen** der Werkzeugorientierung mit LEAD Voreilwinkel Winkel relativ zum Flächennormalenvektor PO[PHI] Programmierung eines Polynoms bis 5.Grades TILT Seitwärtswinkel Drehung um Bahntangente (Z-Richtung) PO[PSI] Programmierung eines Polynoms bis 5.Grades THETA Drehwinkel (Drehung um die Werkzeugrichtung in Z) THETA= Wert der am Satzende erreicht wird THETA=AC(...) Satzweise auf Maßangabe absolut umschalten THETA=IC(...) Satzweise auf Kettenmaßangabe umschalten THETA=Θ_e Programmierter Winkel G90/G91 interpolieren PO[THT]=(..) Programmierung eines Polynoms bis 5.Grades Programmierung des Drehvektors - ORIROTA Drehung absolut - ORIROTR relativer Drehvektor - ORIROTT tangentialer Drehvektor
Bahnrelative Orientierung für Orientierungsänderungen relativ zur Bahn oder Drehung des Drehvektors tangential zur Bahn	Orientierungsänderungen **relativ zur Bahn** mit - ORIPATH Werkzeugorientierung bezogen auf die Bahn - ORIPATHS zusätzlich bei einen Knick im Orientierungsverlauf Programmierung des Drehvektors - ORIROTC tangentialer Drehvektor, Drehung zur Bahntangente

6.2 Drei-, Vier- und Fünf-Achs-Transformation (TRAORI)

6.2.1 Allgemeine Zusammenhänge Kardanischer Werkzeugkopf

Funktion

Um optimale Schnittbedingungen beim Bearbeiten räumlich gekrümmter Flächen zu erzielen, muss der Anstellwinkel des Werkzeugs veränderbar sein.

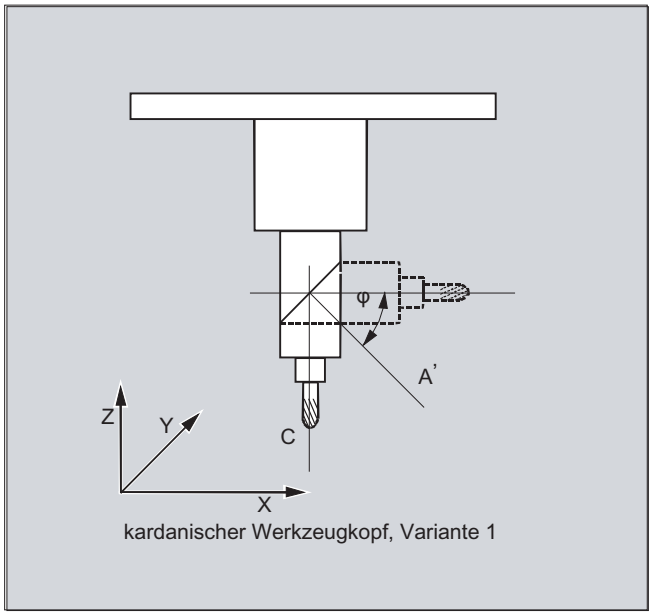


Mit welcher Maschinenkonstruktion dies erreicht wird, ist in den Achsdaten hinterlegt.

5-Achs-Transformation

Kardanischer Werkzeugkopf

Hier legen drei Linearachsen (X, Y, Z) und zwei Orientierungsachsen (C, A) den Anstellwinkel und Arbeitspunkt des Werkzeugs fest. Eine der beiden Orientierungsachsen ist als Schrägachse angelegt, hier im Beispiel A' - in vielen Fällen als 45°-Anordnung.

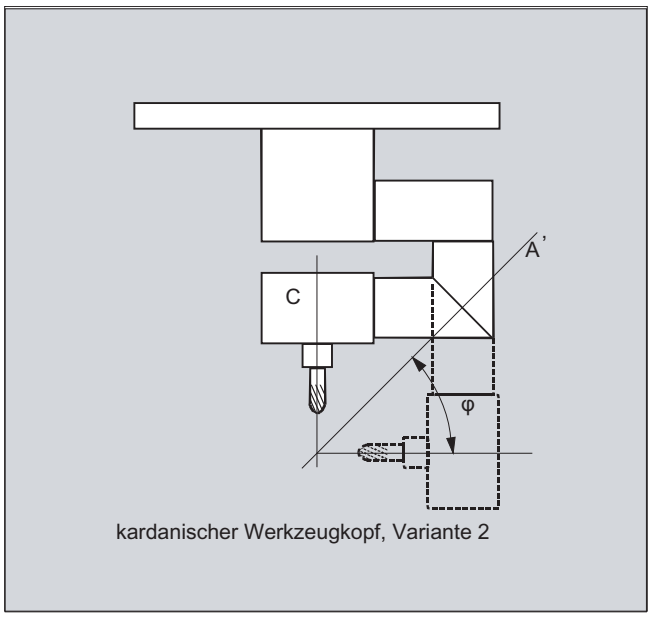


kardanischer Werkzeugkopf, Variante 1

In den hier gezeigten Beispielen sehen Sie die Anordnungen am Beispiel mit dem Kardanischen Werkzeugkopf der Maschinenkinematik CA!

Maschinenhersteller

Die Achsfolge der Orientierungsachsen und die Orientierungsrichtung des Werkzeugs kann abhängig von der Maschinenkinematik über Maschinendaten eingestellt werden.



kardanischer Werkzeugkopf, Variante 2

In diesem Beispiel liegt A' unter dem Winkel φ zur X-Achse

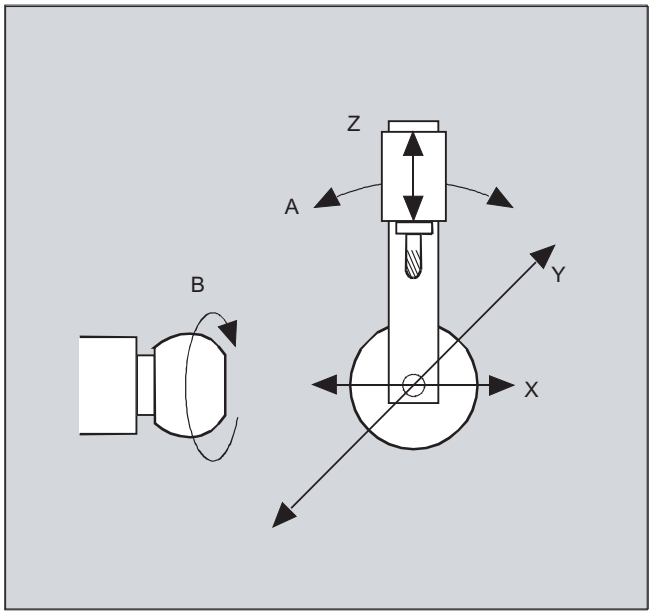
Allgemein gelten folgende mögliche Zusammenhänge:

A' liegt unter dem Winkel φ zur	X-Achse
B' liegt unter dem Winkel φ zur	Y-Achse
C' liegt unter dem Winkel φ zur	Z-Achse

Der Winkel φ kann im Bereich 0° bis +89° über Maschinendaten projektiert werden.

Mit schwenkbarer Linearachse

Hierbei handelt es sich um eine Anordnung mit bewegtem Werkstück und bewegtem Werkzeug. Die Kinematik setzt sich aus drei Linearachsen (X, Y, Z) und zwei rechtwinklig angeordneten Drehachsen zusammen. Die erste Rundachse wird z. B. über einen Kreuzschlitten von zwei Linearachsen bewegt, das Werkzeug steht parallel zur dritten Linearachse. Die zweite Drehachse dreht das Werkstück. Die dritte Linearachse (Schwenkachse) liegt in der Ebene des Kreuzschlittens.



Die Achsfolge der rotatorischen Achsen und die Orientierungsrichtung des Werkzeugs kann abhängig von der Maschinenkinematik über Maschinendaten eingestellt werden.

Es gelten folgende mögliche Zusammenhänge:

Achsen:	Achsfolgen:
1. Rundachse	A A B B C C
2. Rundachse	B C A C A B
Geschwenkte Linearachse	Z Y Z X Y X

Weitere Erläuterungen zu konfigurierbaren Achsfolgen für die Orientierungsrichtung des Werkzeugs siehe

Literatur: /FB3/ Funktionshandbuch Sonderfunktionen; 3- bis 5-Achs-Transformationen (F2), Kapitel Kardanischer Fräskopf, "Parametrierung".

6.2.2 Drei, Vier, und Fünf- Achs-Transformation (TRAORI)

Funktion

Der Anwender kann zwei bzw. drei translatorische Achsen und eine rotatorische Achse projektieren. Die Transformationen gehen davon aus, dass die rotatorische Achse orthogonal auf der Orientierungsebene steht.

Die Orientierung des Werkzeugs ist nur in der Ebene möglich, die senkrecht zur rotatorischen Achse ist. Die Transformation unterstützt die Maschinentypen mit beweglichem Werkzeug und beweglichem Werkstück.

Die Projektierung und Programmierung der Drei- und Vier-Achs-Transformationen sind analog zu den Fünf-Achs-Transformationen.

Literatur:
Funktionshandbuch Sonderfunktionen; Mehrachstransformationen (F2)

Syntax

```
TRAORI(<n>)
TRAORI(<n>,<X>,<Y>,<Z>,<A>,<B>)
TRAFOOF
```

Bedeutung

`TRAORI`:	Aktiviert die erste vereinbarte Orientierungstransformation
`TRAORI(<n>)`:	Aktiviert die mit n vereinbarte Orientierungstransformation
`<n>`:	Nummer der Transformation
	Wert: 1 oder 2
	Beispiel:
	TRAORI(1) aktiviert Orientierungstransformation 1
`<X>,<Y>,<Z>`:	Komponente des Orientierungsvektors, in die das Werkzeug zeigt.
`<A>,<B>`:	Programmierbarer Offset für die Rundachsen
`TRAFOOF`:	Transformation ausschalten

Werkzeugorientierung

Abhängig von der gewählten Orientierungsrichtung des Werkzeugs muss im NC-Programm die aktive Arbeitsebene (G17, G18, G19) so eingestellt werden, dass die Werkzeuglängenkorrektur in Richtung der Werkzeugorientierung wirkt.

Hinweis

Nach dem Einschalten der Transformation beziehen sich Positionsangaben (X, Y, Z) immer auf die Spitze des Werkzeugs. Änderung der Positionen der an der Transformation beteiligten Rundachsen führen zu Ausgleichsbewegungen der übrigen Maschinenachsen, wodurch die Position der Werkzeugspitze unverändert bleibt.

Die Orientierungstransformation ist immer von der Werkzeugspitze zur Werkzeugaufnahme gerichtet.

Offset für Orientierungsachsen

Bei Aktivierung der Orientierungstransformation kann ein zusätzlicher Offset für Orientierungsachsen direkt programmiert werden.

Es dürfen Parameter weggelassen werden, wenn bei der Programmierung die richtige Reihenfolge eingehalten wird.

Beispiel:

`TRAORI(, , , ,A,B)` ; wenn nur ein einziger Offset eingegeben werden soll

Alternativ zur direkten Programmierung kann der zusätzliche Offset für Orientierungsachsen auch aus der momentan aktiven Nullpunktverschiebung automatisch übernommen werden. Die Übernahme wird über Maschinendaten projektiert.

Beispiele

```
TRAORI(1,0,0,1)          ; Die Grundorientierung des Werkzeugs zeigt in Z-Richtung

TRAORI(1,0,1,0)          ; Die Grundorientierung des Werkzeugs zeigt in Z-Richtung

TRAORI(1,0,1,1)          ; Die Grundorientierung des Werkzeugs zeigt in Y/Z-Richtung
                           (entspricht Stellung -45°)
```

6.2.3 Varianten der Orientierungsprogrammierung und Grundstellung (ORIRESET)

Orientierungsprogrammierung der Werkzeugorientierung bei TRAORI

In Verbindung mit einer programmierbaren Orientierungstransformation TRAORI können zusätzlich zu den Linearachsen X, Y, Z auch über die Rundachsbezeichner A.., B..., C... Achspositionen oder virtuelle Achsen mit Winkeln oder Vektorkomponenten programmiert werden. Für Orientierungs- und Maschinenachsen sind verschiedene Interpolationsarten möglich. Unabhängig davon, welche Orientierungspolynome PO[Winkel] und Achspolynome PO[Achse] gerade aktiv sind, können mehrere unterschiedliche Polynomarten wie z.B. G1, G2, G3, CIP oder POLY programmiert sein.

Die Änderung der Orientierung des Werkzeuges kann auch über Orientierungsvektoren programmiert werden. Hierbei kann die Endorientierung jedes Satzes entweder durch direkte Programmierung des Vektors oder durch Programmierung der Rundachspositionen erfolgen.

Hinweis

Varianten der Orientierungsprogrammierung bei Drei- bis Fünf-Achs-Transformationen

Bei der Drei- bis Fünf-Achs-Transformation schließen sich die Varianten

1. A, B, C direkte Angabe der Maschinenachspositionen
2. A2, B2, C2 Winkelprogrammierung virtueller Achsen über Eulerwinkel oder RPY-Winkel
3. A3 ,B3, C3 Angabe der Vektorkomponenten
4. LEAD, TILT Angabe der Voreil- und Seitwärtswinkel bezogen auf die Bahn und Oberfläche
5. A4, B4, C4 und A5, B5, C5 Flächennormalenvektor am Satzanfang und am Satzende
6. A6, B6, C6 und A7, B7, C7 Interpolation des Orientierungsvektors auf einer Kegelmantelfläche
7. A8, B8, C8 Umorientierung des Werkzeugs, Richtung und Weglänge der Abhebebewegung

gegenseitig aus.

Gemischt programmierte Werte werden durch Alarmmeldungen verhindert.

Grundstellung der Werkzeugorientierung ORIRESET

Durch Programmierung von ORIRESET(A, B, C) werden Orientierungsachsen linear und synchron von ihrer momentanen Position zu der angegebenen Grundstellungsposition gefahren.

Wird für eine Achse keine Grundstellungsposition programmiert, dann wird definierte Position aus dem dazugehörigen Maschinendatum $MC_TRAFO5_ROT_AX_OFFSET_1/2 verwendet. Eventuell aktive Frames der Rundachsen werden dabei nicht berücksichtigt.

Hinweis

Nur wenn eine Orientierungstransformation mit TRAORI(...) aktiv ist, kann eine Grundstellung der Werkzeugorientierung kinematikunabhäng mit ORIRESET(...) ohne Alarm 14101 programmiert werden.

Beispiele

```
1. Beispiel für Maschinenkinematik CA (Kanalachsnamen C, A)
ORIRESET(90, 45)       ;C auf 90 Grad, A auf 45 Grad
ORIRESET(, 30)         ;C auf $MC_TRAFO5_ROT_AX_OFFSET_1/2[0], A auf 30 Grad
ORIRESET( )            ;C auf $MC_TRAFO5_ROT_AX_OFFSET_1/2[0],
                       ;A auf $MC_TRAFO5_ROT_AX_OFFSET_1/2[1]
2. Beispiel für Maschinenkinematik CAC (Kanalachsnamen C, A, B)
ORIRESET(90, 45, 90)   ;C auf 90 Grad, A auf 45 Grad, B auf 90 Grad
ORIRESET( )            ;C auf $MC_TRAFO5_ROT_AX_OFFSET_1/2[0],
                       ;A auf $MC_TRAFO5_ROT_AX_OFFSET_1/2[1],
                       ;B auf $MC_TRAFO5_ROT_AX_OFFSET_1/2[2]
```

Programmierung der Drehungen LEAD, TILT und THETA

Die Drehungen der Werkzeugorientierung werden bei der Drei- bis Fünf-Achs-Transformation mit den Voreilwinkel LEAD und den Seitwärtswinkel TILT programmiert.

Bei einer Transformation mit dritter Rundachse sind sowohl für die Orientierung mit Vektorkomponenten als auch mit Angabe der Winkel LEAD, TILT zusätzliche Programmierungen von C2 (Verdrehungen des Orientierungsvektors) erlaubt.

Mit einer zusätzlichen dritten Rundachse kann die Drehung des Werkzeugs um sich selbst mit dem Drehwinkel THETA programmiert werden.

6.2.4 Programmierung der Werkzeugorientierung (A..., B..., C..., LEAD, TILT)

Funktion

Für die Programmierung der Orientierung des Werkzeugs gibt es folgende Möglichkeiten:

1. Direkte Programmierung der Bewegung der Rundachsen. Die Orientierungsänderung erfolgt immer im Basis- bzw. Maschinen-Koordinatensystem. Die Orientierungsachsen werden als Synchronachsen verfahren.
2. Programmierung in Euler- oder RPY-Winkeln gemäß Winkeldefinition über `A2, B2, C2`.
3. Programmierung des Richtungsvektors über `A3, B3, C3`. Der Richtungsvektor zeigt von der Werkzeugspitze in Richtung Werkzeugaufnahme.
4. Programmierung des Flächennormalenvektors am Satzanfang mit `A4, B4, C4` und am Statzende mit `A5, B5, C5` (Stirnfräsen).
5. Programmierung über Voreilwinkel `LEAD` und Seitwärtswinkel `TILT`
6. Programmierung der Drehachse des Kegels als normierter Vektor über `A6, B6, C6` oder der Zwischenorientierung auf der Kegelmantelfläche über `A7, B7, C7`,
   siehe Kapitel "Orientierungsprogrammierung entlang einer Kegelmantelfläche (ORIPLANE, ORICONxx)".
7. Programmierung der Umorientierung, Richtung und Weglänge des Werkzeugs während der Abhebebewegung über `A8, B8, C8`,
   siehe Kapitel "Glättung des Orientierungsverlaufs (ORIPATHS A8=, B8=, C8=)"

Hinweis

In allen Fällen ist die Orientierungsprogrammierung nur zulässig, wenn eine Orientierungstransformation eingeschaltet ist.

Vorteil: Diese Programme sind auf jede Maschinenkinematik portierbar.

Definition der Werkzeugorientierung über G-Code

Hinweis

Maschinenhersteller

Über Maschinendatum kann zwischen Euler- oder RPY-Winkeln umgeschaltet werden. Bei entsprechenden Maschinendaten Einstellungen ist eine Umschaltung sowohl abhängig als auch unabhängig vom aktiven G-Code der Gruppe 50 möglich. Folgende Einstellmöglichkeiten stehen zur Auswahl:

1. Wenn beide Maschinendaten für die Definition der Orientierungsachsen und Definition der Orientierungswinkel über G-Code auf Null gesetzt sind:
   Die mit `A2, B2, C2` programmierten Winkel werden **abhängig vom Maschinendatum** Winkeldefinition der Orientierungsprogrammierung entweder als Euler- oder RPY-Winkeln interpretiert.
2. Wenn das Maschinendatum für die Definition der Orientierungsachsen über G-Code auf Eins gesetzt ist, erfolgt Umschaltung
   abhängig vom aktiven G-Code der Gruppe 50:
   Die mit `A2, B2, C2` programmierten Winkel werden gemäß eines der aktiven G-Codes `ORIEULER, ORIRPY, ORIVIRT1, ORIVIRT2, ORIAXPOS` und `ORIPY2` interpretiert. Die mit den Orientierungsachsen programmierten Werte werden entsprechend dem aktiven G-Code der Gruppe 50 auch als Orientierungswinkel interpretiert.
3. Wenn das Maschinendatum für die Definition der Orientierungswinkel über G-Code auf Eins und das Maschinendatum für die Definition der Orientierungsachsen über G-Code auf Null gesetzt ist erfolgt Umschaltung
   unabhängig vom aktiven G-Code der Gruppe 50:
   Die mit `A2, B2, C2` programmierten Winkel werden gemäß eines der aktiven G-Codes `ORIEULER, ORIRPY, ORIVIRT1, ORIVIRT2 ORIAXPOS` und `ORIPY2` interpretiert. Die mit den Orientierungsachsen programmierten Werte werden unabhängig vom aktiven G-Code der Gruppe 50 immer als Rundachspositionen interpretiert.

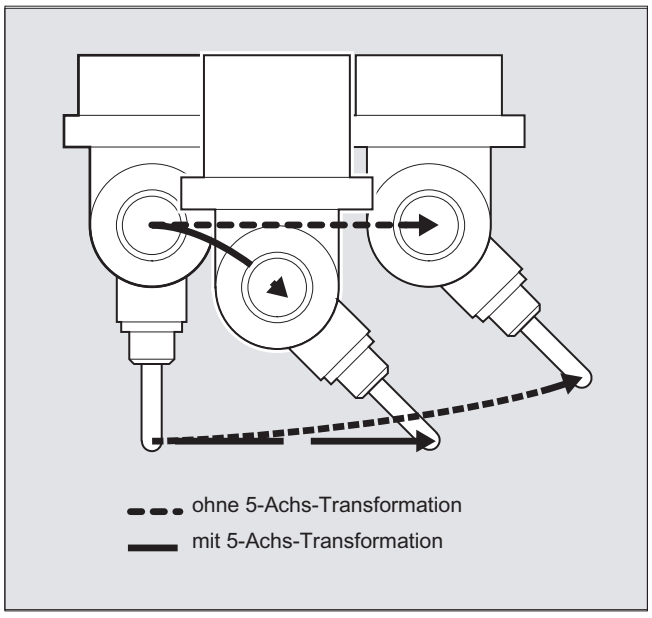
Programmierung

`G1 X Y Z A B C`	Programmierung der Bewegung der Rundachsen
`G1 X Y Z A2= B2= C2=`	Programmierung in Eulerwinkeln
`G1 X Y Z A3== B3== C3==`	Programmierung des Richtungsvektors
`G1 X Y Z A4== B4== C4==`	Programmierung des Flächennormalenvektors am Satzanfang
`G1 X Y Z A5== B5== C5==`	Programmierung des Flächennormalenvektors am Satzende
`LEAD=`	Voreilwinkel für die Programmierung der Werkzeugorientierung
`TILT=`	Seitwärtswinkel für die Programmierung der Werkzeugorientierung

Parameter

G....	Angabe der Bewegungsart der Rundachsen
X Y Z	Angabe der Linearachsen
A B C	Angabe der Maschinenachspositionen der Rundachsen
A2 B2 C2	Winkelprogrammierung (Euler- oder RPY-Winkel) virtueller Achsen bzw. Orientierungsachsen
A3 B3 C3	Angabe der Vektorkomponenten Richtungsvektors
A4 B4 C4	Angabe z. B. beim Stirnfräsen die Komponente des Flächennormalenvektors am Satzanfang
A5 B5 C5	Angabe z .B. beim Stirnfräsen die Komponente des Flächennormalenvektors am Satzende
LEAD	Winkel relativ zum Flächennormalenvektor, in der von Bahntangente und Flächennormalenvektor aufgespannten Ebene
TILT	Winkel in der Ebene, senkrecht zur Bahntangente relativ zum Flächennormalenvektor

Beispiel Gegenüberstellung ohne und mit 5-Achs-Transformation



Beschreibung

In der Regel werden 5-Achs-Programme von CAD/CAM-Systemen erzeugt und nicht an der Steuerung eingegeben. Deshalb wenden sich die folgenden Erklärungen hauptsächlich an Programmierer von Postprozessoren.

Die Art der Orientierungsprogrammierung wird in der G-Code Gruppe 50 festgelegt:

`ORIEULER` über Euler-Winkel
`ORIRPY` über RPY-Winkel (Drehreihenfolge ZYX)
`ORIVIRT1` über virtuelle Orientierungsachsen (Definition 1)
`ORIVIRT2` über virtuelle Orientierungsachsen (Definition 2)
`ORIAXPOS` über virtuelle Orientierungsachsen mit Rundachspositionen
`ORIPY2` über RPY-Winkel (Drehreihenfolge XYZ)

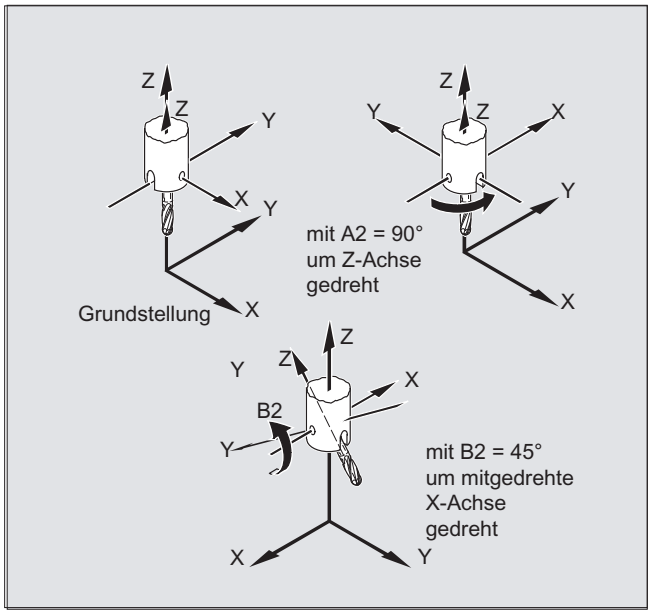
Maschinenhersteller

Über Maschinendaten können vom Maschinenhersteller verschiedene Varianten definiert werden. Bitte beachten Sie die Angaben des Maschinenherstellers.

Programmierung in Eulerwinkeln ORIEULER

Die bei der Orientierungsprogrammierung mit `A2, B2, C2` programmierten Werte werden als Eulerwinkel (in Grad) interpretiert.

Der Orientierungsvektor ergibt sich, indem ein Vektor in Z-Richtung zunächst mit `A2` um die Z-Achse, dann mit `B2` um die neue X-Achse und zuletzt mit `C2` um die neue Z-Achse gedreht wird.



In diesem Fall ist der Wert von `C2` (Drehung um neue Z-Achse) bedeutungslos und muss nicht programmiert werden.

Programmierung in RPY-Winkeln ORIRPY

Die bei der Orientierungsprogrammierung mit `A2, B2, C2` programmierten Werte werden als RPY-Winkel (in Grad) interpretiert.

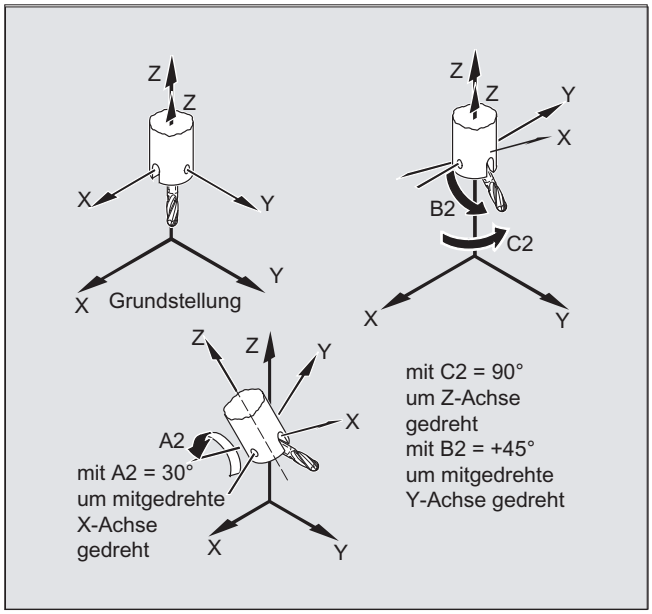
Hinweis

Im Gegensatz zur Eulerwinkel-Programmierung haben hier alle drei Werte Einfluss auf den Orientierungsvektor.

Maschinenhersteller

Bei Winkeldefinition mit Orientierungswinkel über RPY-Winkel gilt für die Orientierungsachsen mit $MC_ORI_DEF_WITH_G_CODE = 0

Der Orientierungsvektor ergibt sich, indem ein Vektor in Z-Richtung zunächst mit C2 um die Z-Achse, dann mit B2 um die neue Y-Achse und zuletzt mit A2 um die neue X-Achse gedreht wird.

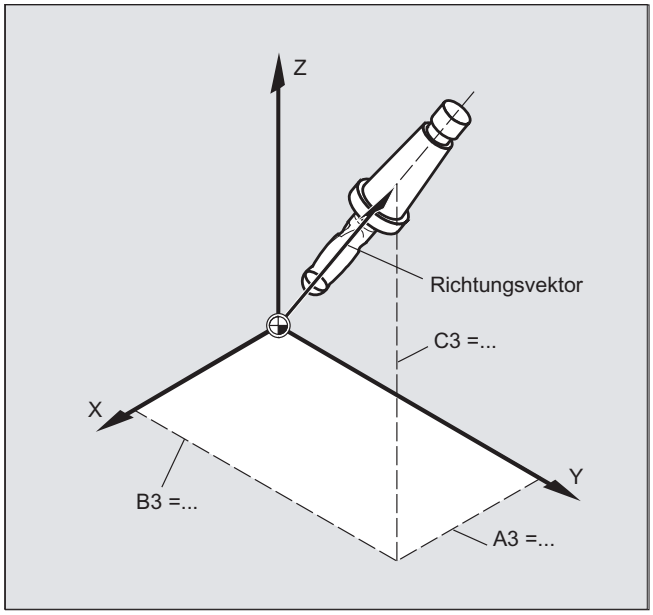


Ist das Maschinendatum zur über Definition der Orientierungsachsen über G-Code $MC_ORI_DEF_WITH_G_CODE = 1 dann gilt:
Der Orientierungsvektor ergibt sich, indem ein Vektor in Z-Richtung zunächst mit A2 um die Z-Achse, dann mit B2 um die neue Y-Achse und zuletzt mit C2 um die neue X-Achse gedreht wird.

Programmierung des Richtungsvektors

Die Komponenten des Richtungsvektors werden mit `A3, B3, C3` programmiert. Der Vektor zeigt in Richtung Werkzeugaufnahme; die Länge des Vektors ist dabei ohne Bedeutung.

Nicht programmierte Vektorkomponenten werden gleich Null gesetzt.



Programmierung der Werkzeugorientierung mit LEAD= und TILT=

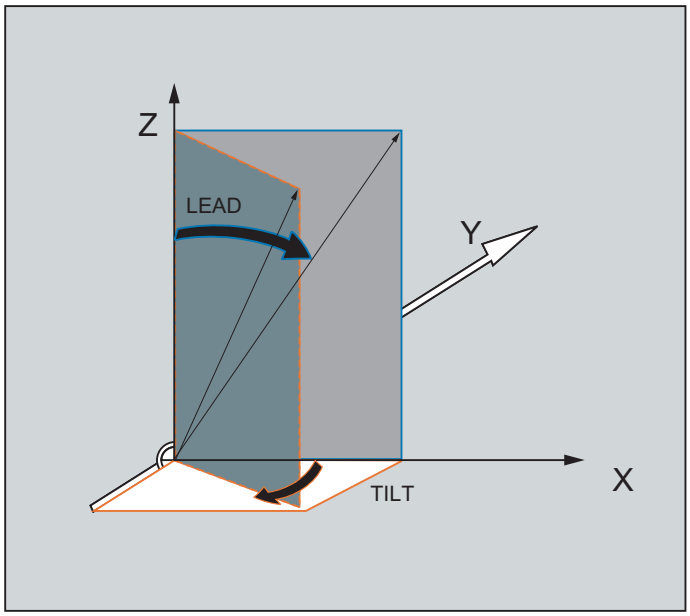
Die resultierende Werkzeugorientierung wird ermittelt aus:

- Bahntangente
- Flächennormalenvektor
  am Satzanfang `A4, B4, C4` und am Satzende `A5, B6, C5`
- Voreilwinkel `LEAD`
  in der von Bahntangente und Flächennormalenvektor aufgespannten Ebene
- Seitwärtswinkel `TILT` am Satzende
  senkrecht zur Bahntangente und relativ zum Flächennormalenvektor

Verhalten bei Innenecken (bei 3D-WZK)

Wenn der Satz an einer Innenecke verkürzt wird, wird die resultierende Werkzeugorientierung ebenso am Satzende erreicht.

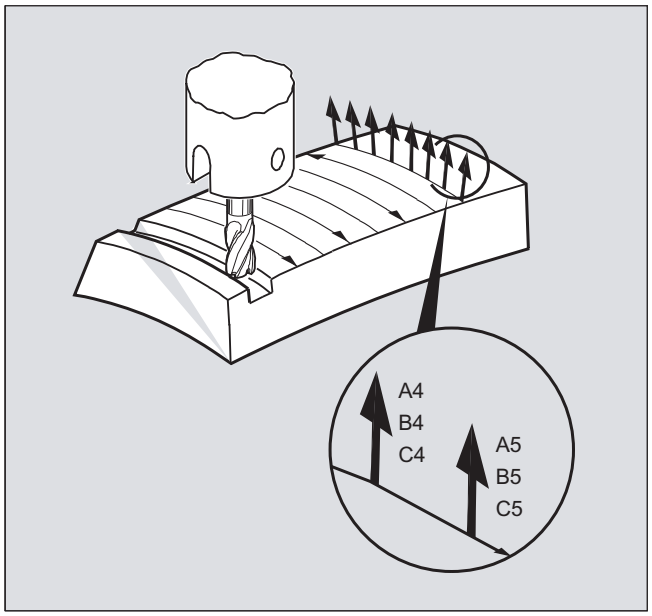
Definition der Werkzeugorientierung mit LEAD= und TILT=



6.2.5 Stirnfräsen (3D-Fräsen A4, B4, C4, A5, B5, C5)

Funktion

Stirnfräsen dient zur Bearbeitung beliebig gekrümmter Oberflächen.



Für diese Art des 3D-Fräsens benötigen Sie die zeilenweise Beschreibung der 3D-Bahnen auf der Werkstückoberfläche.

Die Berechnungen werden unter Berücksichtigung der Werkzeugform und Werkzeugabmessungen üblicherweise im CAM durchgeführt. Die fertig berechneten NC-Sätze werden dann über Postprozessoren in die Steuerung eingelesen.

Programmierung der Bahnkrümmung

Beschreibung der Flächen

Die Beschreibung der Bahnkrümmung erfolgt über Flächennormalenvektoren mit folgenden Komponenten:

`A4, B4, C4` Startvektor am Satzanfang

`A5, B5, C5` Endvektor am Satzende

Steht in einem Satz nur der Startvektor, bleibt der Flächennormalenvektor über den ganzen Satz konstant. Steht in einem Satz nur der Endvektor, so wird vom Endwert des vorherigen Satzes über Großkreisinterpolation zum programmierten Endwert interpoliert.

Sind Start- und Endvektor programmiert, so wird zwischen beiden Richtungen ebenfalls über Großkreisinterpolation interpoliert. Hierdurch lassen sich kontinuierlich glatte Bahnwege erzeugen.

In der Grundstellung zeigen Flächennormalenvektoren unabhängig von der aktiven Ebene G17 bis G19 in Z-Richtung.

Die Länge eines Vektors ist ohne Bedeutung.

Nicht programmierte Vektorkomponenten werden zu Null gesetzt.

Bei aktivem ORIWKS, siehe Kapitel "Bezug der Orientierungsachsen (ORIWKS, ORIMKS)" beziehen sich die Flächennormalenvektoren auf den aktiven Frame und werden bei Framedrehung mitgedreht.

Maschinenhersteller

Der Flächennormalenvektor muss innerhalb eines über Maschinendatum einstellbaren Grenzwertes senkrecht zur Bahntangente stehen, ansonsten wird Alarm ausgegeben.

6.2.6 Bezug der Orientierungsachsen (ORIWKS, ORIMKS)

Funktion

Bei Orientierungsprogrammierung im Werkstückkoordinatensystem über

- Euler- bzw. RPY-Winkel oder
- Orientierungsvektor

kann der Verlauf der Drehbewegung über ORIMKS/ORIWKS eingestellt werden.

Hinweis

Maschinenhersteller

Die Interpolationsart für die Orientierung wird festgelegt mit dem Maschinendatum:

MD21104 $MC_ORI_IPO_WITH_G_CODE

= FALSE: Bezug sind die G-Funktionen ORIWKS und ORIMKS

= TRUE: Bezug sind die G- Funktionen der 51. Gruppe (ORIAXES, ORIVECT, ORIPLANE, ...)

Syntax

```
ORIMKS=...
ORIWKS=...
```

Bedeutung

ORIMKS	Drehung im Maschinenkoordinatensystem
ORIWKS	Drehung im Werkstückkoordinatensystem

Hinweis

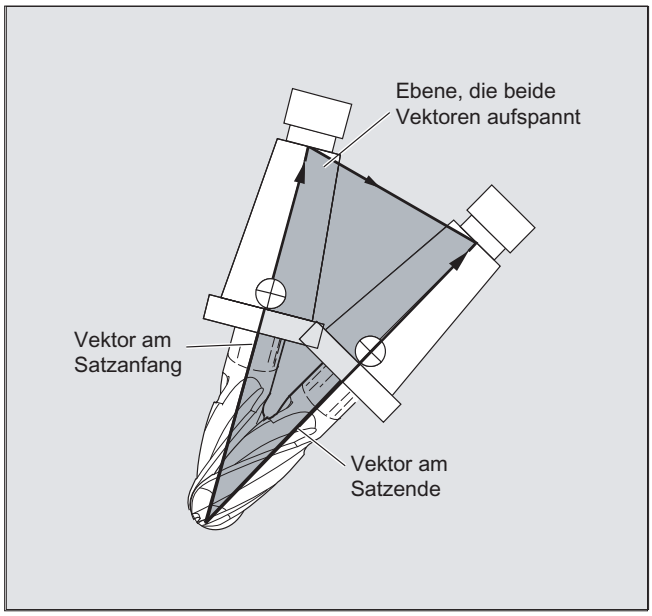
ORIWKS ist Grundeinstellung. Ist bei einem Fünf-Achs-Programm nicht von vornherein klar, auf welcher Maschine es ablaufen soll, so ist grundsätzlich ORIWKS zu wählen. Welche Bewegungen die Maschine tatsächlich ausführt, hängt von der Maschinenkinematik ab.

Mit ORIMKS können tatsächliche Maschinenbewegungen programmiert werden, z. B. um Kollisionen mit Vorrichtungen o. ä. zu vermeiden.

Beschreibung

Bei ORIMKS ist die ausgeführte Werkzeugbewegung von der Maschinenkinematik **abhängig**. Bei Orientierungsänderung mit raumfester Werkzeugspitze wird zwischen den Rundachspositionen linear interpoliert.

Bei ORIWKS ist die Werkzeugbewegung von der Maschinenkinematik **unabhängig**. Bei Orientierungsänderung mit raumfester Werkzeugspitze bewegt sich das Werkzeug in der vom Anfangs- und Endvektor aufgespannten Ebene.



Singuläre Stellungen

Hinweis

ORIWKS

Orientierungsbewegungen im Bereich der singulären Stellung der Fünf-Achs-Maschine erfordern große Bewegungen der Maschinenachsen. (Beispielsweise sind bei einem Drehschwenkkopf mit C als Drehachse und A als Schwenkachse alle Stellungen mit A=0 singulär.)

Maschinenhersteller

Um die Maschinenachsen nicht zu überlasten, senkt die Geschwindigkeitsführung die Bahngeschwindigkeit in der Nähe der singulären Stellen stark ab.

Mit den Maschinendaten

$MC_TRAFO5_NON_POLE_LIMIT

$MC_TRAFO5_POLE_LIMIT

kann die Transformation so parametriert werden, dass Orientierungsbewegungen in der Nähe des Pols durch den Pol gelegt werden und eine zügige Bearbeitung möglich ist.

Singuläre Stellen werden nur mit dem MD $MC_TRAFO5_POLE_LIMIT behandelt.

Literatur:
/FB3/ Funktionshandbuch Sonderfunktionen; 3- bis 5-Achs-Transformation (F2), Kapitel "Singuläre Stellen und ihre Behandlung".

6.2.7 Programmierung der Orientierungsachsen (ORIAXES, ORIVECT, ORIEULER, ORIRPY, ORIRPY2, ORIVIRT1, ORIVIRT2)

Funktion

Die Funktion Orientierungsachsen beschreibt die Orientierung des Werkzeugs im Raum und wird durch Programmierung der Offsets für die Rundachsen erreicht. Ein weiterer dritter Freiheitsgrad kann durch die zusätzliche Drehung des Werkzeugs um sich selbst erzielt werden. Diese Werkzeugorientierung erfolgt beliebig im Raum über eine dritte Rundachse und erfordert die Sechs-Achs-Transformation. Die Eigendrehung des Werkzeugs um sich selbst wird abhängig von der Interpolationsart der Drehvektoren mit dem Drehwinkel THETA festgelegt, siehe Kapitel "Drehungen der Werkzeugorientierung (ORIROTA/TR/TT, ORIROTC, THETA)".

Programmierung

Orientierungsachsen werden über die Achsbezeichner A2, B2, C2 programmiert.

`N... ORIAXES` oder `ORIVECT` `N... G1 X Y Z A B C`	Lineare oder Großkreisinterpolation
oder	oder
`N... ORIPLANE`	Orientierungsinterpolation der Ebene
oder	oder
`N... ORIEULER` oder `ORIRPY bzw. ORIRPY2` `N... G1 X Y Z A2= B2= C2=`	Orientierungswinkel Euler-/RPY-Winkel
oder	Winkelprogrammierung virtueller Achsen
`N... ORIVIRT1` oder `ORIVIRT2` `N... G1 X Y Z A3= B3= C3=`	oder virtuelle Orientierungsachsen Definition 1 oder 2 Richtungsvektorprogrammierung

Für Orientierungsänderungen entlang einer im Raum befindlichen Kegelmantelfläche können weitere Rundachsoffsets der Orientierungsachsen programmiert werden, siehe Kapitel "Orientierungsprogrammierung entlang einer Kegelmantelfläche (ORIPLANE, ORICONxx).

Parameter

`ORIAXES`	`Lineare Interpolation der Maschinen- oder Orientierungsachsen`
`ORIVECT`	`Großkreisinterpolation (identisch mit ORIPLANE)`
`ORIMKS`	`Drehung im Maschinenkoordinatensystem`
`ORIWKS`	`Drehung im Werkstückkoordinatensystem`
	`Beschreibung siehe Kap. Drehungen der Werkzeugorientierung`
`A= B= C=`	`Programmierung der Maschinenachsposition`

ORIEULER	Orientierungsprogrammierung über Euler-Winkel
ORIRPY	Orientierungsprogrammierung über RPY-Winkel. Die Drehreihenfolge ist XYZ, wobei gilt: A2 ist der Drehwinkel um X B2 ist der Drehwinkel um Y C2 ist der Drehwinkel um Z
ORIRPY2	Orientierungsprogrammierung über RPY-Winkel. Die Drehreihenfolge ist ZYX, wobei gilt: A2 ist der Drehwinkel um Z B2 ist der Drehwinkel um Y C2 ist der Drehwinkel um X
A2= B2= C2=	Winkelprogrammierung virtueller Achsen
ORIVIRT1 ORIVIRT2	Orientierungsprogrammierung über virtuelle Orientierungsachsen (Definition 1), Festlegung nach MD $MC_ORIAX_TURN_TAB_1 (Definition 2), Festlegung nach MD $MC_ORIAX_TURN_TAB_2
A3= B3= C3=	Richtungsvektorprogrammierung der Richtungsachse

Beschreibung

Maschinenhersteller

Mit MD $MC_ORI_DEF_WITH_G_CODEwird festgelegt, wie die programmierten Winkel `A2`, `B2`, `C2` definiert werden:

Definition erfolgt nach MD $MC_ORIENTATION_IS_EULER (Standard) oder Definition erfolgt nach G-Gruppe 50 (`ORIEULER`, `ORIRPY`, `ORIVIRT1`, `ORIVIRT2`).

Mit MD $MC_ORI_IPO_WITH_G_CODE wird festgelegt, welche Interpolationsart wirksam ist: `ORIWKS/ORIMKS` oder `ORIAXES/ORIVECT`.

Betriebsart JOG

Die Orientierungswinkel werden in dieser Betriebsart immer linear interpoliert. Beim kontinuierlichen und inkrementellen Verfahren über Verfahrtasten kann nur eine Orientierungsachse verfahren werden. Über die Handräder können die Orientierungsachsen gleichzeitig verfahren werden.

Für das Handverfahren von Orientierungsachsen wirkt der kanalspezifische Vorschub-Korrekturschalter bzw. der Eilgang-Korrekturschalter bei Eilgangüberlagerung.

Mit folgenden Maschinendaten ist eine separate Geschwindigkeitsvorgabe möglich:

$MC_JOG_VELO_RAPID_GEO

$MC_JOG_VELO_GEO

$MC_JOG_VELO_RAPID_ORI

$MC_JOG_VELO_ORI

Hinweis

SINUMERIK 840D mit "Transformationspaket Handling"

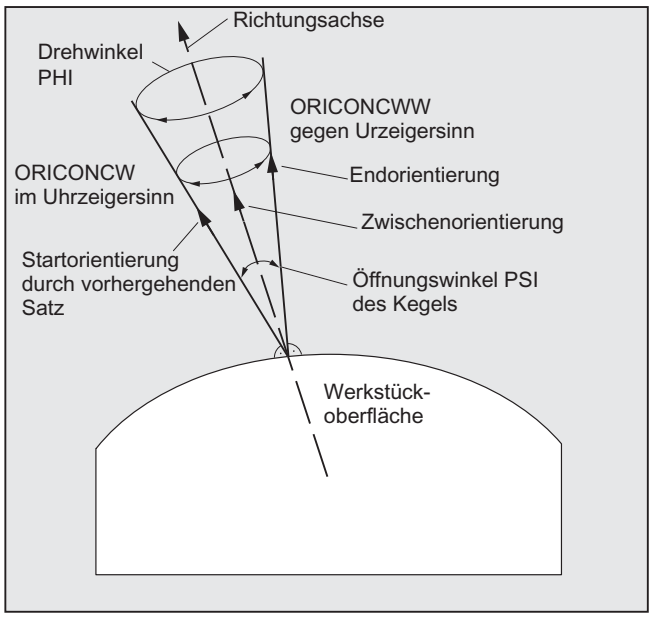
Mit der Funktion "Kartesisches Handverfahren" kann im JOG-Betrieb die Translation von Geometrieachsen in den Bezugssystemen MKS, WKS und TKS getrennt voneinander eingestellt werden.

Literatur:
/FB2/ Funktionshandbuch Erweiterungsfunktionen; Kinematische Transformation (M1)

6.2.8 Orientierungsprogrammierung entlang einer Kegelmantelfläche (ORIPLANE, ORICONCW, ORICONCCW, ORICONTO, ORICONIO)

Funktion

Mit der erweiterten Orientierung ist es möglich, Orientierungsänderungen entlang sich einer im Raum befindlichen Kegelmantelfläche auszuführen. Die Interpolation des Orientierungsvektors auf einer Kegelmantelfläche erfolgt mit den modalen Befehlen ORICONxx. Für die Interpolation in einer Ebene kann die Endorientierung mit ORIPLANE programmiert werden. Generell wird die Startorientierung durch die vorhergehenden Sätze festgelegt.



Programmierung

Die Endorientierung wird entweder durch Angabe der Winkelprogrammierung in Euler- oder RPY-Winkel mit A2, B2, C2 oder durch Programmierung der Rundachspositionen mit A, B, C festgelegt. Für die Orientierungsachsen entlang der Kegelmantelfläche sind weitere Programmierangaben erforderlich:

- Drehachse des Kegels als Vektor mit `A6, B6, C6`
- Öffnungswinkel `PSI` mit den Bezeichner `NUT`
- Zwischenorientierung im Kegelmantel mit `A7, B7, C7`

Hinweis

Programmierung des Richtungsvektor A6, B6, C6 für die Drehachse des Kegels

Die Programmierung einer Endorientierung ist nicht unbedingt erforderlich. Ist keine Endorientierung angegeben, dann wird ein voller Kegelmantel mit 360 Grad interpoliert.

Programmierung des Öffnungswinkel des Kegels mit NUT=winkel

Die Angabe einer Endorientierung ist zwingend erforderlich.

Ein vollständiger Kegelmantel mit 360 Grad kann auf diese Weise nicht interpoliert werden.

Programmierung der Zwischenorientierung A7, B7, C7 im Kegelmantel

Die Angabe einer Endorientierung ist zwingend erforderlich. Die Orientierungsänderung und Drehrichtung wird eindeutig durch die drei Vektoren Start-, End- und Zwischenorientierung festgelegt. Alle drei Vektoren müssen hierbei voneinander unterschiedlich sein. Ist die programmierte Zwischenorientierung parallel zur Start- oder Endorientierung, dann wird eine lineare Großkreisinterpolation der Orientierung in der Ebene, die von Start- und Endvektor aufgespannt wird, durchgeführt.

Erweiterte Orientierungsinterpolation auf einer Kegelmantelfläche

```
N... ORICONCW oder ORICONCCW
N... A6= B6= C6= A3= B3= C3=
oder
N... ORICONTO
N... G1 X Y Z A6= B6= C6=
oder
N... ORICONIO
N... G1 X Y Z A7= B7= C7=
N... PO[PHI]=(a2, a3, a4, a5)
N... PO[PSI]=(b2, b3, b4, b5)
```

Interpolation auf einen Kegelmantel mit

Richtungsvektor im/gegen Uhrzeigersinn des Kegels und Endorientierung oder

tangentialem Übergang und Angabe der Endorientierung

oder

Angabe der Endorientierung und einer Zwischenorientierung im Kegelmantel mit

Polynome für Drehwinkel und Polynome für Öffnungswinkel

Parameter

ORIPLANE	Interpolation in der Ebene (Großkreisinterpolation)
ORICONCW	Interpolation auf einer Kegelmantelfläche im Uhrzeigersinn
ORICONCCW	Interpolation auf einer Kegelmantelfläche gegen Uhrzeigersinn
ORICONTO	Interpolation auf einer Kegelmantelfläche tangentialer Übergang
A6= B6= C6=	Programmierung der Drehachse des Kegels (normierter Vektor)
NUT=winkel	Öffnungswinkel des Kegels in Grad
NUT=+179	Verfahrwinkel kleiner oder gleich 180 Grad
NUT=-181	Verfahrwinkel größer oder gleich 180 Grad
ORICONIO	Interpolation auf einer Kegelmantelfläche
A7= B7= C7=	Zwischenorientierung (Programmierung als normierter Vektor)
PHI	Drehwinkel der Orientierung um die Richtungsachse des Kegels
PSI	Öffnungswinkel des Kegels
mögliche Polynome PO[PHI]=(a2, a3, a4, a5) PO[PSI]=(b2, b3, b4, b5)	Außer den jeweiligen Winkeln sind auch Polynome maximal 5. Grades programmierbar

Beispiel unterschiedliche Orientierungsänderungen

```
…
N10 G1 X0 Y0 F5000
N20 TRAORI(1)                          ; Orientierungstransformation ein.
N30 ORIVECT                            ; Werkzeug-Orientierung als Vektor
                                         interpolieren.
…                                      ; Werkzeugorientierung in der Ebene.
N40 ORIPLANE                           ; Großkreisinterpolation auswählen.
N50 A3=0 B3=0 C3=1
N60 A3=0 B3=1 C3=1                     ; Orientierung in der Y/Z-Ebene um 45 Grad
                                         gedreht, am Satzende wird die Orientierung
                                         (0,1/√2,1/√2)erreicht.
…
N70 ORICONCW                           ; Orientierungsprogrammierung auf Kegelmantel:
N80 A6=0 B6=0 C6=1 A3=0 B3=0 C3=1      ; Der Orientierungsvektor wird auf einem
                                         Kegelmantel mit der Richtung (0,0,1) bis zur
                                         Orientierung (1/√2,0,1/√2) im Uhrzeigersinn
                                         interpoliert, der Drehwinkel beträgt hierbei
                                         270 Grad.
N90 A6=0 B6=0 C6=1                     ; Die Werkzeugorientierung durchläuft eine
                                         volle Umdrehung auf demselben Kegelmantel.
```

Beschreibung

Sollen Orientierungsänderungen auf einer beliebig im Raum liegenden Kegelmantelfläche beschrieben werden, dann muss der Vektor um den die Werkzeugorientierung gedreht werden soll, bekannt sein. Außerdem müssen die Start- und Endorientierung vorgegeben werden. Die Startorientierung ergibt sich aus den vorhergehenden Satz und die Endorientierung muss entweder programmiert oder durch andere Bedingungen festgelegt werden.

Programmierung in der Ebene ORIPLANE entspricht ORIVECT

Die Programmierung der Großkreisinterpolation zusammen mit Winkelpolynomen entspricht der Linear- und Polynominterpolation von Konturen. Die Werkzeugorientierung wird in einer Ebene interpoliert, die von der Start- und Endorientierung aufgespannt wird. Werden zusätzlich Polynome programmiert, dann kann der Orientierungsvektor auch aus der Ebene gekippt werden.

Programmierung von Kreisen in einer Ebene G2/G3, CIP und CT

Die erweiterte Orientierung entspricht der Interpolation von Kreisen in einer Ebene. Zu den entsprechenden Programmiermöglichkeiten von Kreisen mit Mittelpunktsangabe oder Radiusangabe wie G2/G3, Kreis über Zwischenpunkt CIP und Tangentialkreise CT siehe

Literatur: Programmierhandbuch Grundlagen, "Wegbefehle programmieren".

Orientierungsprogrammierung

Interpolation des Orientierungsvektors auf einer Kegelmantelfläche ORICONxx

Für die Interpolation von Orientierungen auf einer Kegelmantelfläche können vier verschiedene Interpolationsarten aus der G-Code Gruppe 51 ausgewählt werden:

1. Interpolation auf einen Kegelmantel im Uhrzeigersinn `ORICONCW` mit Angabe der Endorientierung und der Kegelrichtung oder des Öffnungswinkels. Der Richtungsvektor wird mit den Bezeichnern `A6, B6, C6` und der Öffnungswinkel des Kegels wird mit dem Bezeichner NUT= Wertebereich im Intervall 0 bis 180 Grad programmiert.
2. Interpolation auf einen Kegelmantel gegen Uhrzeigersinn `ORICONCWW` mit Angabe der Endorientierung und der Kegelrichtung oder des Öffnungswinkels. Der Richtungsvektor wird mit den Bezeichnern `A6, B6, C6` und der Öffnungswinkel des Kegels wird mit dem Bezeichner NUT= Wertebereich im Intervall 0 bis 180 Grad programmiert.
3. Interpolation auf einen Kegelmantel `ORICONIO` mit Angabe der Endorientierung und einer Zwischenorientierung, die mit den Bezeichnern `A7, B7, C7` programmiert wird.
4. Interpolation auf einen Kegelmantel `ORICONTO` mit tangentialem Übergang und Angabe der Endorientierung. Der Richtungsvektor wird mit den Bezeichnern `A6, B6, C6` programmiert.

6.2.9 Orientierungsvorgabe zweier Kontaktpunke (ORICURVE, PO[XH]=, PO[YH]=, PO[ZH]=)

Funktion

Programmierung der Orientierungsänderung durch die zweite Raumkurve ORICURVE

Eine weitere Möglichkeit der Programmierung von Orientierungsänderungen besteht darin, außer der Werkzeugspitze entlang einer Raumkurve auch die Bewegung eines zweiten Kontaktpunktes des Werkzeugs mit ORICURVE zu programmieren. Damit können Orientierungsänderungen des Werkzeugs, wie bei der Programmierung des Werkzeugvektors selber, eindeutig festgelegt werden.

Maschinenhersteller

Beachten Sie bitte die Hinweise des Maschinenherstellers zu über Maschinendatum einstellbare Achsbezeichner für die Programmierung der 2. Orientierungsbahn des Werkzeugs.

Programmierung

Bei dieser Interpolationsart können für die beiden Raumkurven Punkte mit G1 bzw. Polynome mit POLY programmiert werden. Kreise und Evolventen sind nicht zulässig. Zusätzlich kann eine Spline-Interpolation mit BSPLINE und die Funktion "Zusammenfassung kurzer Spline-Sätze" aktiviert werden.

Literatur:
/FB1/ Funktionshandbuch Grundfunktionen; Bahnsteuerbetrieb, Genauhalt, Look Ahead (B1), Kapitel: Zusammenfassung kurzer Spline-Sätze

Die anderen Splinearten ASPLINE und CSPLINE sowie die Aktivierung eines Kompressors mit COMPON, COMPCURV oder COMPCAD sind nicht zulässig.

Die Bewegung der zwei Kontaktpunkte des Werkzeugs kann bei der Programmierung der Orientierungspolynome für Koordinaten bis maximal 5.Grades vorgegeben werden.

Erweiterte Orientierungsinterpolation mit zusätzlicher Raumkurve und Polynome für Koordinaten

```
N... ORICURVE
N... PO[XH]=(xe, x2, x3, x4, x5)
N... PO[YH]=(ye, y2, y3, y4, y5)
N... PO[ZH]=(ze, z2, z3, z4, z5)
```

Angabe der Bewegung des zweiten Kontaktpunkts des Werkzeugs und zusätzliche Polynome der jeweiligen Koordinaten

Parameter

ORICURVE	Interpolation der Orientierung mit Vorgabe der Bewegung zweier Kontaktpunkte des Werkzeuges.
XH YH ZH	Bezeichner der Koordinaten des zweiten Kontaktpunktes des Werkzeugs der zusätzlichen Kontur als Raumkurve
mögliche Polynome PO[XH]=(xe, x2, x3, x4, x5) PO[YH]=(ye, y2, y3, y4, y5) PO[ZH]=(ze, z2, z3, z4, z5)	Außer den jeweiligen Endpunkten sind die Raumkurven zusätzlich mit Polynomen programmierbar.
xe, ye, ze	Endpunkte der Raumkurve
xi, yi, zi	Koeffizienten der Polynome maximal 5. Grades

Hinweis

Bezeichner XH YH ZH für die Programmierung einer 2. Orientierungsbahn

Die Bezeichner müssen so gewählt werden, dass kein Konflikt mit anderen Bezeichnern der Linearachsen

X Y Z Achsen

und Rundachsen wie

A2 B2 C2 Eulerwinkel bzw. RPY-Winkel

A3 B3 C3 Richtungsvektoren

A4 B4 C4 bzw. A5 B5 C5 Flächennormalenvektoren

A6 B6 C6 Drehvektoren bzw. A7 B7 C7 Zwischenpunktkoordinaten

oder anderen Interpolationsparameter entsteht.

6.3 Orientierungspolynome (PO[Winkel], PO[Koordinate])

Funktion

Unabhängig davon, welche Polynominterpolation der G-Code Gruppe 1 gerade aktiv ist, können zwei verschiedene Typen von Orientierungspolynomen bis maximal 5. Grades bei einer Drei- bis Fünf-Achs-Transformation programmiert werden.

1. Polynome für **Winkel:** Voreilwinkel LEAD, Seitwärtswinkel TILT in Bezug auf die Ebene, die von Start- und Endorientierung aufgespannt wird.
2. Polynome für **Koordinaten:** XH, YH, ZH der zweiten Raumkurve für die Werkzeugorientierung eines Bezugspunktes auf dem Werkzeug.

Bei einer Sechs-Achs-Transformation kann zur Werkzeugorientierung zusätzlich die Drehung des Drehvektors THT mit Polynomen bis maximal 5. Grades für Drehungen des Werkzeugs selbst programmiert werden.

Syntax

Orientierungspolynome vom Typ 1 für **Winkel**

```
N… PO[PHI]=(a2, a3, a4, a5)
N… PO[PSI]=(b2, b3, b4, b5)
```

Drei- bis Fünf-Achs-Transformation

Drei- bis Fünf-Achs-Transformation

Orientierungspolynome vom Typ 2 für **Koordinaten**

```
N… PO[XH]=(xe, x2, x3, x4, x5)
N… PO[YH]=(ye, y2, y3, y4, y5)
N… PO[ZH]=(ze, z2, z3, z4, z5)
```

Bezeichner für die Koordinaten der zweiten Orientierungsbahn für die Werkzeugorientierung

Zusätzlich kann in beiden Fällen ein Polynom für die **Drehung** bei Sechs-Achs-Transformationen mit

```
N… PO[THT]=(c2, c3, c4, c5)
```

oder

```
N… PO[THT]=(d2, d3, d4, d5)
```

Bahnrelative Interpolation der Drehung

absoluter, relative und tangentiale Interpolation zur Orientierungsänderung

des Orientierungsvektors programmiert werden. Dies ist dann möglich, wenn die Transformation einen Drehvektor mit einem durch den Drehwinkel THETA programmierbaren und interpolierbaren Offset unterstützt.

Bedeutung

`PO[PHI]`	Winkel in der Ebene zwischen Start- und Endorientierung
`PO[PSI]`	Winkel der die Auskippung der Orientierung aus der Ebene zwischen Start- und Endorientierung beschreibt
`PO[THT]`	Drehwinkel der durch Drehung des Drehvektors einer der mit THETA programmierten G-Codes der Gruppe 54
`PHI`	Voreilwinkel LEAD
`PSI`	Seitwärtswinkel TILT
`THETA`	Drehung um die Werkzeugrichtung in Z
`PO[XH]`	X-Koordinate des Bezugspunktes auf dem Werkzeug
`PO[YH]`	Y-Koordinate des Bezugspunktes auf dem Werkzeug
`PO[ZH]`	Z-Koordinate des Bezugspunktes auf dem Werkzeug

Beschreibung

Orientierungspolynomen können nicht programmiert werden

- wenn die Splineinterpolationen ASPLINE, BSPLINE, CSPLINE aktiv sind.
  Polynome vom Typ1 für Orientierungswinkel sind für jede Interpolationsart außer Spline d.h. bei Linearinterpolation mit Eilgang G00 bzw. mit Vorschub G01
  bei Polynominterpolation mit POLY und
  bei Kreis- bzw. Evolventeninterpolation mit G02, G03, CIP, CT, INVCW und INCCCW möglich.
  Polynome vom Typ2 für Orientierungskoordinaten sind dagegen nur möglich, wenn Linearinterpolation mit Eilgang G00 bzw. mit Vorschub G01 oder
  Polynominterpolation mit POLY aktiv ist.
- wenn die Orientierung mittels Achsinterpolation ORIAXES interpoliert wird. In diesem Fall können direkt Polynome mit PO[A] und PO[B] für die Orientierungsachsen A und B programmiert werden.

Orientierungspolynome vom Typ 1 mit ORIVECT, ORIPLANE und ORICONxx

Bei Großkreisinterpolation und Kegelmantelinterpolation mit ORIVECT, ORIPLANE und ORICONxx sind nur Orientierungspolynome vom Typ 1 möglich.

Orientierungspolynome vom Typ 2 mit ORICURVE

Ist die Interpolation mit zusätzlicher Raumkurve ORICURVE aktiv, werden die kartesischen Komponenten des Orientierungsvektors interpoliert und es sind nur Orientierungspolynome vom Typ 2 möglich.

6.4 Drehungen der Werkzeugorientierung (ORIROTA, ORIROTR, ORIROTT, ORIROTC, THETA)

Funktion

Soll bei Maschinentypen mit beweglichem Werkzeug auch die Orientierung des Werkzeugs veränderbar sein, so wird jeder Satz mit einer Endorientierung programmiert. Abhängig von der Maschinenkinematik können entweder die Orientierungsrichtung der Orientierungsachsen oder die Drehrichtung des Orientierungsvektors THETA programmiert werden. Für diese Drehvektoren sind verschiedene Interpolationsarten programmierbar:

- ORIROTA: Drehwinkel zu einer absolut vorgegebenen Drehrichtung.
- ORIROTR: Drehwinkel relativ zur Ebene zwischen Start- und Endorientierung.
- ORIROTT: Drehwinkel relativ zur Änderung des Orientierungsvektors.
- ORIROTC: Tangentialer Drehwinkel zur Bahntangente.

Syntax

Nur wenn die Interpolationsart `ORIROTA` aktiv ist, kann der Drehwinkel oder der Drehvektor auf die vier möglichen Arten wie folgt programmiert werden:

1. Direkt die Rundachspositionen `A, B, C`
2. Eulerwinkel (in Grad) über `A2, B2, C2`
3. RPY-Winkel (in Grad) über `A2, B2, C2`
4. Richtungsvektor über `A3, B3, C3` (Drehwinkel mittels `THETA=Wert`)

Falls `ORIOTR` oder `ORIOTT` aktiv sind, kann der Drehwinkel nur noch direkt mit `THETA` programmiert werden.

Eine Drehung kann auch allein in einem Satz programmiert werden, ohne dass eine Orientierungsänderung stattfindet. Dabei haben `ORIROTR` und `ORIROTT` keine Bedeutung. In diesem Fall wird der Drehwinkel immer in Bezug zur absoluten Richtung interpretiert (`ORIROTA`).

```
N... ORIROTA                          Interpolation des Drehvektors festlegen
N... ORIROTR
N... ORIROTT
N... ORIROTC
N... A3= B3= C3= THETA=Wert           Drehung des Orientierungsvektors festlegen

N... PO[THT]=(d2, d3, d4, d5)         Drehwinkel mit Polynom 5. Grades interpolieren
```

Bedeutung

`ORIROTA`	Drehwinkel zu einer absolut vorgegebenen Drehrichtung
`ORIROTR`	Drehwinkel relativ zur Ebene zwischen Start- und Endorientierung
`ORIROTT`	Drehwinkel als tangentialer Drehvektor zur Orientierungsänderung
`ORIROTC`	Drehwinkel als tangentialer Drehvektor zur Bahntangente
`THETA`	Drehung des Orientierungsvektors
`THETA=Wert`	Drehwinkel in Grad, der am Satzende erreicht wird
`THETA=Θe`	Drehwinkel mit Endwinkel Θ_e des Drehvektors
`THETA=AC(…)`	Satzweise auf Maßangabe absolut umschalten
`THETA=AC(…)`	Satzweise auf Kettenmaßangabe umschalten
`Θe`	Endwinkel des Drehvektors sowohl absolut mit G90 als auch relativ mit G91 (Kettenmaßangabe) ist aktiv
`PO[THT]=(....)`	Polynom für den Drehwinkel

Beispiel Drehungen der Orientierungen

Programmcode	**Kommentar**
`N10 TRAORI`	`; Orientierungstransformation aktivieren`
`N20 G1 X0 Y0 Z0 F5000`	`; Orientierung des Werkzeugs`
`N30 A3=0 B3=0 C3=1 THETA=0`	`; in Z-Richtung mit Drehwinkel 0`
`N40 A3=1 B3=0 C3=0 THETA=90`	`; in X-Richtung und Drehung um 90 Grad`
`N50 A3=0 B3=1 C3=0 PO[THT]=(180,90)`	`; Orientierung`
`N60 A3=0 B3=1 C3=0 THETA=IC(-90)`	`; in Y-Richtung und Drehung auf 180 Grad`
`N70 ORIROTT`	`; bleibt konstant und Drehung auf 90 Grad`
`N80 A3=1 B3=0 C3=0 THETA=30`	`; Drehwinkel relativ zur Orientierungsänderung`
	`; Drehvektor im Winkel 30 Grad zur X-Y Ebene`

Bei der Interpolation von Satz
`N40` wird der Drehwinkel vom Startwert 0 Grad zum Endwert 90 Grad linear interpoliert. Im Satz `N50` ändert sich der Drehwinkel von 90 Grad auf 180 Grad gemäß der Parabel $\theta(u) = +90u^2$. In `N60` kann auch eine Drehung ausgeführt werden, ohne dass eine Orientierungsänderung stattfindet.

Bei `N80` wird die Werkzeugorientierung von der Y–Richtung in X–Richtung gedreht. Dabei liegt die Orientierungsänderung in der X–Y Ebene und der Drehvektor bildet zu dieser Ebene einen Winkel von 30 Grad.

Beschreibung

ORIROTA

Der Drehwinkel `THETA` wird bezüglich einer absolut festgelegten Richtung im Raum interpoliert. Die Grunddrehrichtung erfolgt über Maschinendaten

ORIROTR

Der Drehwinkel `THETA` wird relativ zur Ebene, die von der Start- und Endorientierung aufgespannt wird, interpretiert.

ORIROTT

Der Drehwinkel `THETA` wird relativ zur Orientierungsänderung interpretiert. Für `THETA=0` wird der Drehvektor tangential zur Orientierungsänderung interpoliert und unterscheidet sich nur dann zu `ORIROTR`, wenn für die Orientierung mindestens ein Polynom für den "Kippwinkel PSI" programmiert wurde. Damit ergibt sich eine Orientierungsänderung, die nicht in der Ebene abläuft. Durch einen zusätzlich programmierten Drehwinkel `THETA` kann dann z. B. der Drehvektor so interpoliert werden, dass er immer einen bestimmten Wert zur Orientierungsänderung bildet.

ORIROTC

Der Drehvektor wird relativ zur Bahntangente mit einem durch den Winkel `THETA` programmierbaren Offset interpoliert. Für den Offsetwinkel kann dabei auch ein Polynom `PO[THT]=(c2, c3, c4, c5)` maximal 5. Grades programmiert werden.

6.5 Bahnrelative Orientierungen

6.5.1 Orientierungsarten relativ zur Bahn

Funktion

Mit dieser erweiterten Funktion wird die relative Orientierung nicht nur am Satzende, sondern über den gesamten Bahnverlauf erreicht. Es wird die im Vorgängersatz erreichte Orientierung mittels Großkreisinterpolation in die programmierte Endorientierung überführt. Grundsätzlich gibt es zwei Möglichkeiten die gewünschte Orientierung relativ zur Bahn zu programmieren:

1. Die Werkzeugorientierung als auch die Drehung des Werkzeugs wird mit ORIPATH, ORPATHTS relativ zur Bahn interpoliert.
2. Der Orientierungsvektor wird wie bisher üblich programmiert und interpoliert. Mit ORIROTC wird die Drehung des Orientierungsvektors relativ zur Bahntangente angestellt.

Syntax

Die Interpolationsart der Orientierung und der Drehung des Werkzeugs wird programmiert mit:

`N... ORIPATH`	Bahnrelative Orientierung
`N... ORIPATHS`	Bahnrelative Orientierung mit Glättung des Orientierungsverlaufs
`N... ORIROTC`	Bahnrelative Interpolation des Drehvektors

Ein durch eine Ecke im Bahnverlauf hervorgerufener Knick der Orientierung kann mit `ORIPATHS` geglättet werden. Die Richtung und Weglänge der Abhebebewegung wird durch den Vektor mit den Komponenten `A8=X, B8=Y C8=Z` programmiert.

Mit ORIPATH/ORIPATHS können verschiedene Bezüge zur Bahntangente über die drei Winkel

- LEAD= Angabe Vorwärtswinkel bezogen auf die Bahn und Oberfläche
- TILT= Angabe von Seitwärtswinkel bezogen auf die Bahn und Oberfläche
- THETA= Drehwinkel

für den gesamten Bahnverlauf programmiert werden. Zum Drehwinkel THETA können mit PO[THT]=(...) zusätzlich Polynome maximal 5. Grades programmiert werden.

Hinweis

Maschinenhersteller

Bitte beachten Sie die Angaben des Maschinenherstellers. Über projektierbare Maschinen- und Settingdaten können zur Bahnrelativen Orientierungsart weitere Einstellungen vorgenommen werden. Weitere Erläuterungen siehe

Literatur:
/FB3/ Funktionshandbuch Sonderfunktionen; 3- bis 5-Achs-Transformation (F2), Kapitel "Orientierung"

Bedeutung

Die Interpolation der Winkel LEAD und TILT ist über Maschinedatum unterschiedlich einstellbar:

- Der mit LEAD und TILT programmierte Bezug der Werkzeugorientierung wird über den ganzen Satz hinweg eingehalten.
- Vorwärtswinkel LEAD: Drehung um die Richtung senkrecht zur Tangente und Normalenvektor TILT: Drehung der Orientierung um den Normalenvektor.
- Vorwärtswinkel LEAD: Drehung um die Richtung senkrecht zur Tangente und Normalenvektor Seitwärtswinkel TILT: Drehung der Orientierung um die Richtung der Bahntangente.
- Drehwinkel THETA: Drehung des Werkzeugs um sich selbst mit einer zusätzlichen dritten Rundachse als Orientierungsachse bei Sechs-Achs-Transformation.

Hinweis

Bahnrelative Orientierung zusammen mit OSC, OSS, OSSE, OSD, OST unzulässig

Die bahnrelative Orientierungsinterpolation ORIPATH bzw. ORIPATHS und ORIOTC kann nicht zusammen mit der Glättung des Orientierungsverlaufs mit einen der G-Codes aus der Gruppe 34 programmiert werden. Hierfür muss OSOF aktiv sein.

6.5.2 Bahnrelative Drehung der Werkzeugorientierung (ORIPATH, ORIPATHS, Drehwinkel)

Funktion

Bei einer Sechs-Achs-Transformation kann zur Werkzeugorientierung beliebig im Raum auch das Werkzeug mit einer dritten Rundachse um sich selbst gedreht werden. Bei bahnrelativer Drehung der Werkzeugorientierung mit ORIPATH bzw. ORIPATHS kann die zusätzliche Drehung über den Drehwinkel THETA programmiert werden. Alternativ hierzu können die Winkel LEAD und TILT durch einen Vektor, der in der Ebene senkrecht zur Werkzeugrichtung liegt, programmiert werden.

Maschinenhersteller

Bitte beachten Sie die Angaben des Maschinenherstellers. Über Maschinendatum kann die Interpolation der Winkel LEAD und TILT unterschiedlich eingestellt werden.

Syntax

Drehung der Werkzeugorientierung und des Werkzeugs

Die Werkzeugorientierungsart relativ zur Bahn wird mit ORIPATH oder ORIPATHS aktiviert.

`N... ORIPATH`	Orientierungsart bezogen auf die Bahn aktivieren
`N... ORIPATHS`	Orientierungsart bezogen auf die Bahn mit Glättung des Orientierungsverlaufs aktivieren

Aktivierung der drei möglichen Winkel mit Drehwirkung:

`N... LEAD=`	Winkel für die programmierten Orientierung relativ zum Flächennormalenvektor
`N... TILT=`	Winkel für die programmierte Orientierung in der Ebene senkrecht zur Bahntangente relativ zum Flächennormalenvektor
`N... THETA=`	Drehwinkel relativ zur Orientierungsänderung um die Werkzeugrichtung der dritten Rundachse

Die Werte der Winkel am Satzende werden mit `LEAD=Wert`, `TILT=Wert` bzw. `THETA=Wert` programmiert. Zusätzlich zu den konstanten Winkeln können für alle drei Winkel Polynome maximal 5. Grades programmiert werden.

`N... PO[PHI]=(a2, a3, a4, a5)`	Polynom für den Voreilwinkel LEAD
`N... PO[PSI]=(b2, b3, b4, b5)`	Polynom für den Seitwärtswinkel TILT
`N... PO[THT]=(d2, d3, d4, d5)`	Polynom für den Drehwinkel THETA

Bei der Programmierung können die höheren Polynomkoeffizienten, die Null sind, weggelassen werden. Beispiel `PO[PHI]=a2` ergibt für den Voreilwinkel `LEAD` eine Parabel.

Bedeutung

Bahnrelative Werkzeugorientierung

`ORIPATH`	Werkzeugorientierung bezogen auf die Bahn
`ORIPATHS`	Werkzeugorientierung bezogen auf die Bahn Knick im Orientierungsverlauf wird geglättet
`LEAD`	Winkel relativ zum Flächennormalenvektor, in der von Bahntangente und Flächennormalenvektor aufgespannten Ebene
`TILT`	Drehung der Orientierung um die Z-Richtung bzw. Drehung um die Bahntangente
`THETA`	Drehung um die Werkzeugrichtung nach Z
`PO[PHI]`	Orientierungspolynom für den Voreilwinkel LEAD
`PO[PSI]`	Orientierungspolynom für den Seitwärtswinkel TILT
`PO[THT]`	Orientierungspolynom für den Drehwinkel THETA

Hinweis

Drehwinkel THETA

Für die Drehung des Werkzeugs mit dritter Rundachse als Orientierungsachse um sich selbst, ist eine Sechs-Achs-Transformation erforderlich.

6.5.3 Bahnrelative Interpolation der Werkzeugdrehung (ORIROTC, THETA)

Funktion

Interpolation mit Drehvektoren

Zur mit ORIROTC programmierten Drehung des Werkzeugs relativ zur Bahntangenten kann der Drehvektor auch mit einem durch den Drehwinkel THETA programmierbaren Offset interpoliert werden. Dabei kann für den Offsetwinkel mit PO[THT] ein Polynom bis maximal 5. Grades programmiert werden.

Syntax

`N... ORIROTC`	Drehung des Werkzeugs relativ zur Bahntangente anstellen
`N... A3= B3= C3= THETA=Wert`	Drehung des Orientierungsvektors festlegen
`N... A3= B3= C3= PO[THT]=(c2, c3, c4, c5)`	Offsetwinkel mit Polynom maximal 5. Grades interpolieren

Eine Drehung kann auch allein in einem Satz programmiert werden, ohne dass eine Orientierungsänderung stattfindet.

Bedeutung

Bahnrelative Interpolation der Drehung des Werkzeugs bei Sechs-Achs-Transformation

`ORIROTC`	tangentialer Drehvektor zur Bahntangente anstellen
`THETA=Wert`	Drehwinkel in Grad, der am Satzende erreicht wird
`THETA=θe`	Drehwinkel mit Endwinkel Θ_e des Drehvektors
`THETA=AC(…)`	Satzweise auf Maßangabe absolut umschalten
`THETA=IC(…)`	Satzweise auf Kettenmaßangabe umschalten
`PO[THT]=(c2, c3, c4, c5)`	Offsetwinkel mit Polynom 5. Grades interpolieren

Hinweis

Interpolation des Drehvektors ORIROTC

Soll gegen die Orientierungsrichtung des Werkzeugs auch die Drehung des Werkzeugs relativ zur Bahntangente angestellt werden, dann ist dies nur bei einer Sechs-Achs-Transformation möglich.

Bei aktiven ORIROTC

Der Drehvektor ORIROTA kann nicht programmiert werden. Im Falle einer Programmierung wird der ALARM 14128 "Absolutprogrammierung der Werkzeugdrehung bei aktivem ORIROTC" ausgegeben.

Orientierungsrichtung des Werkzeugs bei Drei- bis Fünf-Achs-Transformation

Die Orientierungsrichtung des Werkzeugs kann wie bei der Drei- bis Fünf-Achs-Transformation gewohnt über Eulerwinkel bzw. RPY-Winkel oder des Richtungsvektoren programmiert werden. Auch sind Orientierungsänderungen des Werkzeugs im Raum durch Programmierung der Großkreisinterpolation ORIVECT, der linearen Interpolation der Orientierungsachsen ORIAXES, alle Interpolationen auf einer Kegelmantelfläche ORICONxx sowie der Interpolation zusätzlich zur Raumkurve mit zwei Kontaktpunkten des Werkzeugs ORICURVE möglich.

`G....`	Angabe der Bewegungsart der Rundachsen
`X Y Z`	Angabe der Linearachsen
`ORIAXES`	Lineare Interpolation der Maschinen- oder Orientierungsachsen
`ORIVECT`	Großkreisinterpolation (identisch mit ORIPLANE)
`ORIMKS`	Drehung im Maschinenkoordinatensystem
`ORIWKS`	Drehung im Werkstückkoordinatensystem
	Beschreibung siehe Kap. Drehungen der Werkzeugorientierung
`A= B= C=`	Programmierung der Maschinenachsposition
`ORIEULER`	Orientierungsprogrammierung über Euler-Winkel
`ORIRPY`	Orientierungsprogrammierung über RPY-Winkel
`A2= B2= C2=`	Winkelprogrammierung virtueller Achsen

`ORIVIRT1` `ORIVIRT2`	Orientierungsprogrammierung über virtuelle Orientierungsachsen (Definition 1), Festlegung nach MD $MC_ORIAX_TURN_TAB_1 (Definition 2), Festlegung nach MD $MC_ORIAX_TURN_TAB_2
`A3= B3= C3=`	Richtungsvektorprogrammierung der Richtungsachse
`ORIPLANE`	Interpolation in der Ebene (Großkreisinterpolation)
`ORICONCW`	Interpolation auf einer Kegelmantelfläche im Uhrzeigersinn
`ORICONCCW`	Interpolation auf einer Kegelmantelfläche gegen Uhrzeigersinn
`ORICONTO`	Interpolation auf einer Kegelmantelfläche tangentialer Übergang
`A6= B6= C6=`	Programmierung der Drehachse des Kegels (normierter Vektor)
`NUT=winkel`	Öffnungswinkel des Kegels in Grad
`NUT=+179`	Verfahrwinkel kleiner oder gleich 180 Grad
`NUT=-181`	Verfahrwinkel größer oder gleich 180 Grad
`ORICONIO`	Interpolation auf einer Kegelmantelfläche
`A7= B7= C7=`	Zwischenorientierung (Programmierung als normierter Vektor)
`ORICURVE` `XH YH ZH z.B. mit Polynome PO[XH]=(xe, x2, x3, x4, x5)`	Interpolation der Orientierung mit Vorgabe der Bewegung zweier Kontaktpunkte des Werkzeuges. Außer den jeweiligen Endpunkten sind zusätzliche Raumkurven Polynome programmierbar.

Hinweis

Wird die Werkzeugorientierung mit aktiven ORIAXES über die Orientierungsachsen Interpoliert, dann wird die bahnrelative Anstellung des Drehwinkels nur am Satzende erfüllt.

6.5.4 Glättung des Orientierungsverlaufs (ORIPATHS A8=, B8=, C8=)

Funktion

Bei beschleunigungsstetigen Orientierungsänderungen an der Kontur sind Unterbrechungen der Bahnbewegungen, die besonders an einer Ecke der Kontur auftreten können unerwünscht. Der sich hieraus ergebene Knick im Orientierungsverlauf kann durch Einfügen eines eigenen Zwischensatzes geglättet werden. Die Orientierungsänderung erfolgt dann beschleunigungsstetig, wenn während der Umorientierung auch ORIPATHS aktiv ist. In dieser Phase kann eine Abhebebewegung des Werkzeugs durchgeführt werden.

Maschinenhersteller

Beachten Sie bitte die Hinweise des Maschinenherstellers zu gegebenenfalls vordefinierten Maschinendaten und Settingdaten mit denen diese Funktion aktiviert wird.

Über Maschinendatum ist einstellbar, wie der Abhebevektor interpretiert wird:

1. Im Werkzeugkoordinatensystem wird die Z-Koordinate durch die Werkzeugrichtung definiert.
2. Im Werkstückkoordinatensystem wird die Z-Koordinate durch die aktive Ebene definiert.

Weitere Erläuterungen zur Funktion "Bahnrelative Orientierung" siehe
Literatur: /FB3/ Funktionshandbuch Sonderfunktionen; 3- bis 5-Achstransformation (F2)

Syntax

Für stetige Werkzeugorientierungen bezogen auf die gesamte Bahn sind an einer Ecke der Kontur weitere Programmierangaben erforderlich. Die Richtung und die Weglänge dieser Bewegung wird durch den Vektor mit den Komponenten A8=X, B8=Y, C8=Z programmiert:

```
N... ORIPATHS A8=X B8=Y C8=Z
```

Bedeutung

`ORIPATHS`	Werkzeugorientierung bezogen auf die Bahn, ein Knick im Orientierungsverlauf wird geglättet.
`A8= B8= C8=`	Vektorkomponenten für Richtung und Weglänge
`X, Y, Z`	Abhebebewegung in Werkzeugrichtung

Hinweis

Programmierung des Richtungsvektors A8, B8, C8

Ist die Länge dieses Vektors gleich Null erfolgt keine Abhebebewegung.

ORIPATHS

Die bahnbezogene Werkzeugorientierung wird mit ORIPATHS aktiv. Anderenfalls wird die Orientierung mittels linearer Großkreisinterpolation von der Start- zur Endorientierung überführt.

6.6 Komprimierung der Orientierung (COMPON, COMPCURV, COMPCAD)

Funktion

NC-Programme, in denen eine Orientierungstransformation (TRAORI) aktiv und die Orientierung mittels Richtungsvektoren programmiert ist, können unter Einhaltung von vorgegeben Toleranzen komprimiert werden.

Hinweis

Die Orientierungsbewegung wird nur bei aktiver Großkreisinterpolation komprimiert und ist so vom G-Code für die Orientierungsinterpolation abhängig. Dieser ist ebenso wie die maximale Weglänge und eine zulässige Toleranz für jede Achse bzw. für den Bahnvorschub für die Kompressor-Funktion über Maschinendaten einstellbar. Bitte beachten Sie die Angaben des Maschinenherstellers.

Programmierung

Werkzeugorientierung

Falls eine Orientierungstransformation (TRAORI) aktiv ist, kann bei 5-Achs Maschinen die Werkzeugorientierung folgendermaßen (kinematikunabhängig) programmiert werden:

- Programmierung des Richtungs**vektors** über:

  ```
  A3=<...> B3=<...> C3=<...>
  ```

- Programmierung der Euler**winkel** bzw. RPY-**Winkel** über:

  ```
  A2=<...> B2=<...> C2=<...>
  ```

Drehung des Werkzeugs

Bei **6-Achs** Maschinen kann zusätzlich zur Werkzeugorientierung noch die Drehung des Werkzeugs programmiert werden.

Die Programmierung des Drehwinkels erfolgt mit:

```
THETA=<...>
```

Siehe " Drehungen der Werkzeugorientierung (Seite 343) ".

Hinweis

NC-Sätze, in denen zusätzlich eine Drehung programmiert ist, sind nur dann komprimierbar, falls sich der Drehwinkel **linear** ändert. D. h. für den Drehwinkel darf kein Polynom mit `PO[THT]=(...)` programmiert sein.

Allgemeine Form eines komprimierbaren NC-Satzes

Die allgemeine Form eines komprimierbaren NC-Satzes kann daher wie folgt aussehen:

```
N... X=<...> Y=<...> Z=<...> A3=<...> B3=<...> C3=<...> THETA=<...> F=<...>
```

bzw.

```
N... X=<...> Y=<...> Z=<...> A2=<...> B2=<...> C2=<...> THETA=<...> F=<...>
```

Hinweis

Die Positionswerte können direkt (z. B. X90) oder indirekt über Parameterzuweisungen (z. B. X=R1*(R2+R3)) angegeben werden.

Programmierung der Werkzeugorientierung durch Rundachspositionen

Die Werkzeugorientierung kann auch durch Rundachspositionen angegeben sein, z. B. in der Form:

```
N... X=<...> Y=<...> Z=<...> A=<...> B=<...> C=<...> THETA=<...> F=<...>
```

In diesem Fall wird die Komprimierung auf zwei unterschiedliche Arten durchgeführt, abhängig davon ob eine Großkreisinterpolation durchgeführt wird oder nicht. Wenn keine Großkreisinterpolation stattfindet, dann wird die komprimierte Orientierungsänderung durch axiale Polynome für die Rundachsen in üblicher Weise dargestellt.

Konturgenauigkeit

Abhängig vom eingestellten Kompressionsmodus (MD20482 $MC_COMPRESSOR_MODE) werden für die Geometrieachsen und Orientierungsachsen bei der Komprimierung entweder die projektierten achsspezifischen Toleranzen (MD33100 $MA_COMPRESS_POS_TOL) oder die folgenden über Settingdaten einstellbaren kanalspezifischen Toleranzen wirksam:

SD42475 $SC_COMPRESS_CONTUR_TOL (Maximale Konturabweichung)

SD42476 $SC_COMPRESS_ORI_TOL (Maximale Winkelabweichung für die Werkzeugorientierung)

SD42477 $SC_COMPRESS_ORI_ROT_TOL (Maximale Winkelabweichung für den Drehwinkel des Werkzeugs) (nur bei 6-Achs Maschinen verfügbar)

Literatur:
Funktionshandbuch Grundfunktionen; 3- bis 5-Achs-Transformation (F2),
Kapitel: "Komprimierung der Orientierung"

Aktivierung / Deaktivierung

Kompressor-Funktionen werden eingeschaltet durch die modalen G-Codes COMPON, COMPCURV bzw. COMPCAD.

Beendet wird die Kompressor-Funktion mit COMPOF.

Siehe " NC-Satz-Kompression (COMPON, COMPCURV, COMPCAD) (Seite 247) ".

Hinweis

Die Orientierungsbewegung wird nur komprimiert bei aktiver Großkreisinterpolation (d. h. die Änderung der Werkzeugorientierung erfolgt in der Ebene, die von Start- und Endorientierung aufgespannt wird).

Eine Großkreisinterpolation wird unter den folgenden Bedingungen durchgeführt:

- MD21104 $MC_ORI_IPO_WITH_G_CODE = 0,
  ORIWKS ist aktiv und
  Orientierung ist mittels Vektoren programmiert (mit A3, B3, C3 bzw. A2, B2, C2).
- MD21104 $MC_ORI_IPO_WITH_G_CODE = 1 und
  ORIVECT bzw. ORIPLANE ist aktiv.

  Die Werkzeugorientierung kann entweder als Richtungsvektor oder mit Rundachspositionen programmiert sein. Ist einer der G-Codes ORICONxx oder ORICURVE aktiv oder sind Polynome für die Orientierungswinkel (PO[PHI] und PO[PSI]) programmiert, wird keine Großkreisinterpolation durchgeführt.

Beispiel

Im nachfolgenden Programmbeispiel wird ein Kreis, der durch einen Polygonzug angenähert ist, komprimiert. Die Werkzeugorientierung bewegt sich dabei synchron dazu auf einem Kegelmantel. Obwohl die aufeinanderfolgenden programmierten Orientierungsänderungen unstetig verlaufen, generiert die Kompressor-Funktion einen glatten Verlauf der Orientierung.

Programmierung	**Kommentar**
DEF INT ANZAHL=60	
DEF REAL RADIUS=20	
DEF INT COUNTER	
DEF REAL WINKEL	
N10 G1 X0 Y0 F5000 G64	
$SC_COMPRESS_CONTUR_TOL=0.05	; Maximale Abweichung der Kontur = 0.05 mm
$SC_COMPRESS_ORI_TOL=5	; Maximale Abweichung der Orientierung = 5 Grad
TRAORI	
COMPCURV	
	; Es wird ein Kreis gefahren, der aus Polygonen gebildet wird. Die Orientierung bewegt sich dabei auf einem Kegel um die Z-Achse mit einem Öffnungswinkel von 45 Grad.
N100 X0 Y0 A3=0 B3=-1 C3=1	
N110 FOR COUNTER=0 TO ANZAHL	
N120 WINKEL=360*COUNTER/ANZAHL	
N130 X=RADIUS*cos(WINKEL) Y=RADIUS*sin(WINKEL)	
A3=sin(WINKEL) B3=-cos(WINKEL) C3=1	
N140 ENDFOR	

6.7 Glättung des Orientierungsverlaufs (ORISON, ORISOF)

Funktion

Mit der Funktion "Glättung des Orientierungsverlaufs (ORISON)" können Schwankungen der Orientierung über mehrere Sätze hinweg geglättet werden. Dadurch wird ein glatter Verlauf sowohl der Orientierung als auch der Kontur erzielt.

Voraussetzung

Die Funktion "Glättung des Orientierungsverlaufs (ORISON)" ist nur in Systemen mit 5/6-Achs-Transformation verfügbar.

Syntax

```
ORISON
...
ORISOF
```

Bedeutung

ORISON:	Glättung des Orientierungsverlaufs EIN	
	Wirksamkeit:	modal
ORISOF:	Glättung des Orientierungsverlaufs AUS	
	Wirksamkeit:	modal

Settingdaten

Die Glättung des Orientierungsverlaufs erfolgt unter Einhaltung:

- einer vorgegebenen maximalen Toleranz (maximale Winkelabweichung der Werkzeugorientierung in Grad)

  und

- eines vorgegebenen maximalen Bahnwegs.

Diese Vorgaben werden über Settingdaten definiert:

- SD42678 $SC_ORISON_TOL (Toleranz für die Glättung des Orientierungsverlaufs)
- SD42680 O$SC_ORISON_DIST (Bahnweg für die Glättung des Orientierungsverlaufs)

Beispiel

```
Programmcode                        Kommentar
...
TRAORI()                            ; Einschalten der Orientierungstransformation.
ORISON                              ; Einschalten der Orientierungsglättung.
$SC_ORISON_TOL=1.0                  ; Toleranz der Orientierungsglättung = 1,0 Grad.
G91
X10 A3=1 B3=0 C3=1
X10 A3=-1 B3=0 C3=1
X10 A3=1 B3=0 C3=1
X10 A3=-1 B3=0 C3=1
X10 A3=1 B3=0 C3=1
X10 A3=-1 B3=0 C3=1
X10 A3=1 B3=0 C3=1
X10 A3=-1 B3=0 C3=1
X10 A3=1 B3=0 C3=1
X10 A3=-1 B3=0 C3=1
...
ORISOF                              ; Ausschalten der Orientierungsglättung.
...
```

Die Orientierung wird um 90 Grad in der XZ–Ebene von -45 bis +45 Grad geschwenkt. Durch die Glättung des Orientierungsverlaufs erreicht die Orientierung nicht mehr die maximalen Winkelwerte von -45 bzw. +45 Grad.

Weitere Informationen

Anzahl der Sätze

Die Glättung des Orientierungsverlaufs erfolgt über eine projektierte Anzahl von Sätzen, die im Maschinendatum MD28590 $MC_MM_ORISON_BLOCKS hinterlegt ist.

Hinweis

Wird die Glättung des Orientierungsverlaufs mit `ORISON` aktiviert, ohne dass ausreichend Satzspeicher dafür projektiert wurde (MD28590 < 4), dann erfolgt eine Alarmmeldung und die Funktion kann nicht ausgeführt werden.

Maximale Satzweglänge

Der Orientierungsverlauf wird nur in solchen Sätzen geglättet, deren Verfahrweg kleiner ist als die projektierte maximale Satzweglänge (MD20178 $MC_ORISON_BLOCK_PATH_LIMIT). Sätze mit längeren Verfahrwegen unterbrechen die Glättung und werden wie programmiert abgefahren.

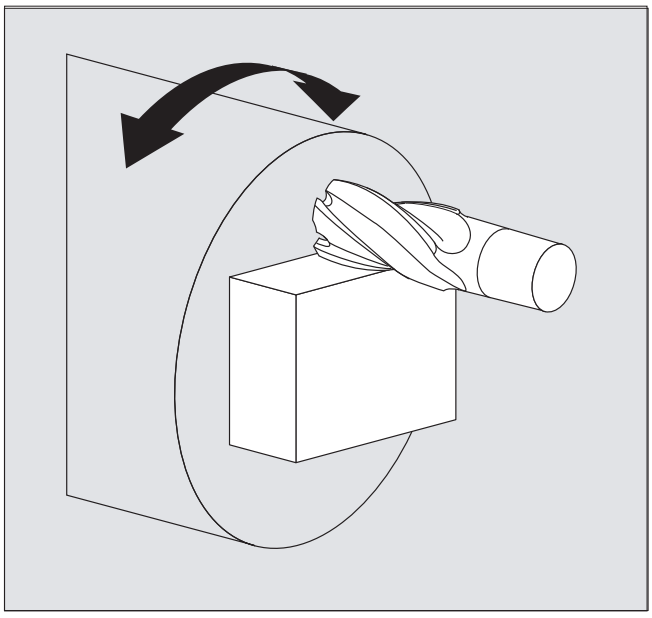
6.8 Kinematische Transformation

6.8.1 Fräsbearbeitung an Drehteilen (TRANSMIT)

Funktion

Die Funktion TRANSMIT ermöglicht folgende Leistungen:

- Stirnseitige Bearbeitung an Drehteilen in der Drehaufspannung (Bohrungen, Konturen).
- Für die Programmierung dieser Bearbeitungen kann ein kartesisches Koordinatensystem benutzt werden.
- Die Steuerung transformiert die programmierten Verfahrbewegungen des kartesischen Koordinatensystems auf die Verfahrbewegungen der realen Maschinenachsen (Standardfall):
  - Rundachse
  - Zustellachse senkrecht zur Drehachse
  - Längsachse parallel zur Drehachse
  - Die Linearachsen stehen senkrecht aufeinander.
- Werkzeugmittenversatz relativ zur Drehmitte ist zulässig.
- Die Geschwindigkeitsführung berücksichtigt die für die Drehbewegungen definierten Begrenzungen.



TRANSMIT Transformationstypen

Für TRANSMIT-Bearbeitungen gibt es zwei einstellbare Ausprägungen:

- `TRANSMIT` im Standardfall mit (TRAFO_TYPE_n = 256)
- `TRANSMIT` mit zusätzlicher Y-Linearachse (TRAFO_TYPE_n = 257)

Der erweiterte Transformationstyp 257 kann dazu verwendet werden, um z. B. Aufspannkorrekturen eines Werkzeugs mit realer Y-Achse zu kompensieren.

Syntax

`TRANSMIT` oder `TRANSMIT(n)`

`TRAFOOF`

Rundachse

Die Rundachse kann nicht programmiert werden, da sie von einer Geometrie-Achse belegt wird und somit als Kanalachse nicht direkt programmierbar ist.

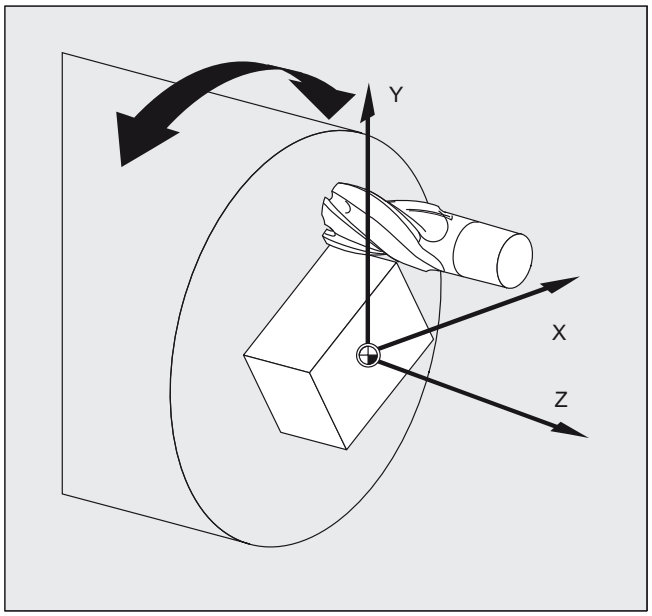
Bedeutung

`TRANSMIT`:	Aktiviert die erste vereinbarte TRANSMIT-Funktion. Diese Funktion wird auch als Polar-Transformation bezeichnet.
`TRANSMIT(n)`:	Aktiviert die n. vereinbarte TRANSMIT-Funktion; n darf maximal 2 sein (TRANSMIT(1) entspricht TRANSMIT).
`TRAFOOF`:	Schaltet eine aktive Transformation aus
`OFFN`:	Offset Kontur-normal: Abstand der stirnseitigen Bearbeitung von der programmierten Bezugskontur

Hinweis

Eine aktive Transformation `TRANSMIT` wird ebenfalls ausgeschaltet, wenn im jeweiligen Kanal eine der übrigen Transformationen aktiviert wird (z. B. `TRACYL`, `TRAANG`, `TRAORI`).

Beispiel



Programmcode	Kommentar
N10 T1 D1 G54 G17 G90 F5000 G94	; Werkzeuganwahl
N20 G0 X20 Z10 SPOS=45	; Anfahren der Ausgangsstellung
N30 TRANSMIT	; TRANSMIT-Funktion aktivieren
N40 ROT RPL=-45	; Frame einstellen
N50 ATRANS X-2 Y10	
N60 G1 X10 Y-10 G41 OFFN=1OFFN	; Vierkant schruppen; Aufmaß 1 mm
N70 X-10	
N80 Y10	
N90 X10	
N100 Y-10	
N110 G0 Z20 G40 OFFN=0	; Werkzeugwechsel
N120 T2 D1 X15 Y-15	
N130 Z10 G41	
N140 G1 X10 Y-10	; Vierkant schlichten
N150 X-10	
N160 Y10	
N170 X10	
N180 Y-10	
N190 Z20 G40	; Frame abwählen
N200 TRANS	
N210 TRAFOOF	
N220 G0 X20 Z10 SPOS=45	; Anfahren der Ausgangsstellung
N230 M30	

Beschreibung

Pol

Zum Durchfahren des Pols gibt es zwei Möglichkeiten:

- Verfahren der Linearachse allein
- Verfahren in den Pol mit Drehung der Rundachse im Pol und Fahren aus dem Pol

Die Auswahl erfolgt über die MD 24911 und 24951.

TRANSMIT mit zusätzlicher Y-Linearachse (Transformationstyp 257):

Diese Tranformationsvariante der Polar-Transformation nützt bei einer Maschine mit einer weiteren Linearachse die Redundanz aus, um eine verbesserte Werkzeugkorrektur durchzuführen. Für die zweite Linearachse gilt dann:

- ein kleinerer Arbeitsbereich und
- dass die zweite Linearachse für das Abfahren des Teileprogramms nicht genutzt werden soll.

Für das Teileprogramm und die Zuordnung der entsprechenden Achsen im BKS oder MKS werden bestimmte Maschinendateneinstellungen vorausgesetzt, siehe

Literatur

/FB2/ Funktionshandbuch Erweiterungsfunktionen; Kinematische Transformationen (M1)

6.8.2 Zylindermanteltransformation (TRACYL)

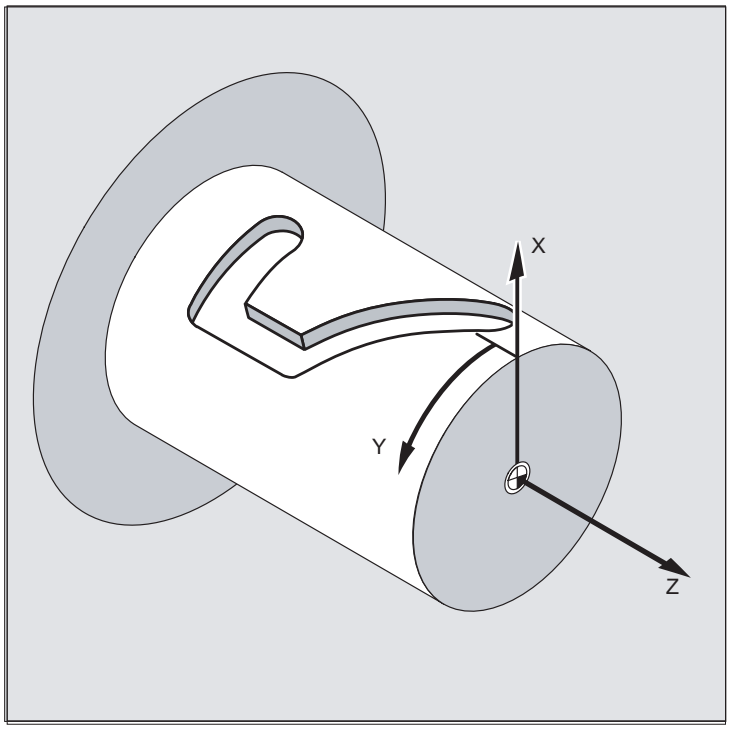
Funktion

Die Zylindermantelkurventransformation TRACYL ermöglicht folgende Leistungen:

Bearbeitung von

- Längsnuten an zylindrischen Körpern,
- Quernuten an zylindrischen Körpern,
- beliebig verlaufende Nuten an zylindrischen Körpern.

Der Verlauf der Nuten wird bezogen auf die abgewickelte, ebene Zylindermantelfläche programmiert.



TRACYL Transformationstypen

Die Zylindermantelkoordinatentransformation gibt es in drei Ausprägungen:

- `TRACYL` ohne Nutwandkorrektur: (TRAFO_TYPE_n=512)
- `TRACYL` mit Nutwandkorrektur: (TRAFO_TYPE_n=513)
- `TRACYL` mit zusätzlicher Linearachse und mit Nutwandkorrektur: (TRAFO_TYPE_n=514)
  Die Nutwandkorrektur wird mit `TRACYL` über den dritten Parameter parametriert.

Bei Zylindermantelkurventransformation mit Nutwandkorrektur sollte die für die Korrektur verwendete Achse auf Null (y=0) stehen, damit die Nut mittig zur programmierten Nutmittellinie gefertigt wird.

Achsnutzung

Folgende Achsen können nicht als Positionierachse bzw. Pendelachse verwendet werden:

- die Geometrieachse in Umfangsrichtung der Zylindermantelfläche (Y-Achse)
- die zusätzliche Linearachse bei Nutwandkorrektur (Z-Achse)

Syntax

`TRACYL(d)` oder `TRACYL(d, n)` oder

für Transformationstyp 514

```
TRACYL(d, n, Nutwandkorrektur)
TRAFOOF
```

Rundachse

Die Rundachse kann nicht programmiert werden, da sie von einer Geometrie-Achse belegt wird und somit als Kanalachse nicht direkt programmierbar ist.

Bedeutung

`TRACYL(d)`	Aktiviert die erste in den Kanalmaschinendaten vereinbarte TRACYL-Funktion. d Parameter für den Arbeitsdurchmesser.
`TRACYL (d, n)`	Aktiviert die n. in den Kanalmaschinendaten vereinbarte TRACYL-Funktion. n darf maximal 2 sein, TRACYL(d,1) entspricht TRACYL(d).
`D`	Wert für den Arbeitsdurchmesser. Der Arbeitsdurchmesser ist der doppelte Abstand zwischen Werkzeugspitze und Drehmitte. Dieser Durchmesser muss immer angegeben werden und größer als 1 sein.
`n`	Optionaler 2. Parameter für den TRACYL-Datensatz 1 (vorangewählt) oder 2.
`Nutwandkorrektur`	Optionaler 3. Parameter dessen Wert für TRACYL aus den Mode von Maschinendaten vorangewählt wird. Wertebereich: 0: Transformationstyp 514 ohne Nutwandkorrektur wie bisher 1: Transformationstyp 514 mit Nutwandkorrektur
`TRAFOOF`	Transformation aus (BKS und MKS sind wieder identisch).
`OFFN`	Offset Kontur-normal: Abstand der Nutwand von der programmierten Bezugskontur

Hinweis

Eine aktive Transformation `TRACYL` wird ebenfalls ausgeschaltet, wenn im jeweiligen Kanal eine der übrigen Transformationen aktiviert wird (z. B. `TRANSMIT`, `TRAANG`, `TRAORI`).

Beispiel: Definition des Werkzeugs

Folgendes Beispiel ist geeignet, die Parametrierung der Zylindertransformation `TRACYL` zu testen:

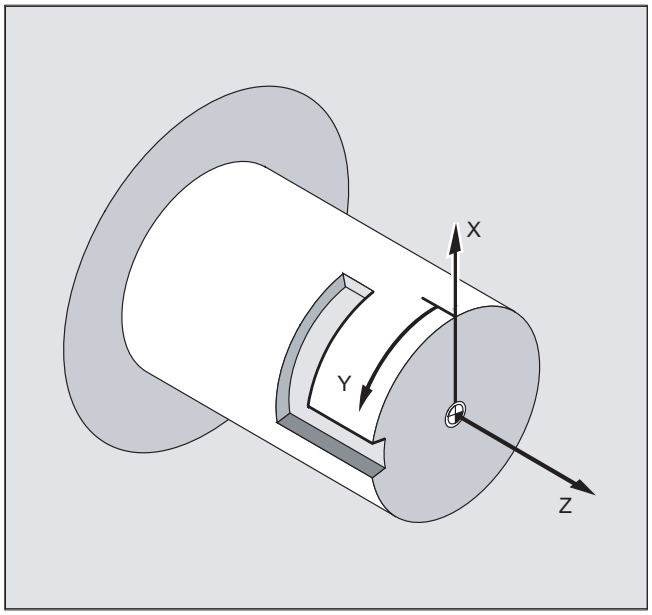
Programmcode	**Kommentar**	
`Werkzeugparameter`	`Bedeutung`	`Bemerkung`
`Nummer (DP)`		
`$TC_DP1[1,1]=120`	`Werkzeugtyp`	`Fräser`
`$TC_DP2[1,1]=0`	`Schneidenlage`	`nur für Drehwerkzeuge`

Programmcode	**Kommentar**	
`Geometrie`	`Längenkorrektur`	
`$TC_DP3[1,1]=8.`	`Längenkorrekturvektor`	`Verrechnung nach Typ`
`$TC_DP4[1,1]=9.`		`und Ebene`
`$TC_DP5[1,1]=7.`		

Programmcode	Kommentar	
Geometrie	Radius	
$TC_DP6[1,1]=6.	Radius	Werkzeugradius
$TC_DP7[1,1]=0	Nutbreite b für Nutsäge, Verrundungsradius für Fräswerkzeuge	
$TC_DP8[1,1]=0	Überstand k	nur für Nutsäge
$TC_DP9[1,1]=0		
$TC_DP10[1,1]=0		
$TC_DP11[1,1]=0	Winkel für kegelige Fräswerkzeuge	

Programmcode	Kommentar	
Verschleiß	Längen- und Radiuskorrektur	
$TC_DP12[1,1]=0	Die restlichen Parameter bis $TC_DP24=0	Basismaß/Adapter

Beispiel: Fertigen einer hakenförmigen Nut



Zylindermanteltransformation einschalten:

Programmcode	Kommentar
N10 T1 D1 G54 G90 F5000 G94	; Werkzeuganwahl, Aufspannkompensation
N20 SPOS=0	; Anfahren der Ausgangsstellung
N30 G0 X25 Y0 Z105 CC=200	
N40 TRACYL (40)	; Zylindermantelkurventransformation ;einschalten
N50 G19	; Ebenenanwahl

Hakenförmige Nut fertigen:

Programmcode	Kommentar
N60 G1 X20	; Werkzeug auf Nutgrund zustellen
N70 OFFN=12	; Nutwandabstand 12 mm relativ zur Nutmittenlinie festlegen
N80 G1 Z100 G42	; Anfahren der rechten Nutwand
N90 G1 Z50	; Nutabschnitt parallel zur Zylinderachse
N100 G1 Y10	; Nutabschnitt parallel zum Umfang
N110 OFFN=4 G42	; Anfahren der linken Nutwand; Nutwandabstand 4 mm relativ zur Nutmittenlinie festlegen
N120 G1 Y70	; Nutabschnitt parallel zum Umfang
N130 G1 Z100	; Nutabschnitt parallel zur Zylinderachse
N140 G1 Z105 G40	; Abfahren von der Nutwand
N150 G1 X25	; Freifahren
N160 TRAFOOF	
N170 G0 X25 Y0 Z105 CC=200	; Anfahren der Ausgangsstellung
N180 M30	

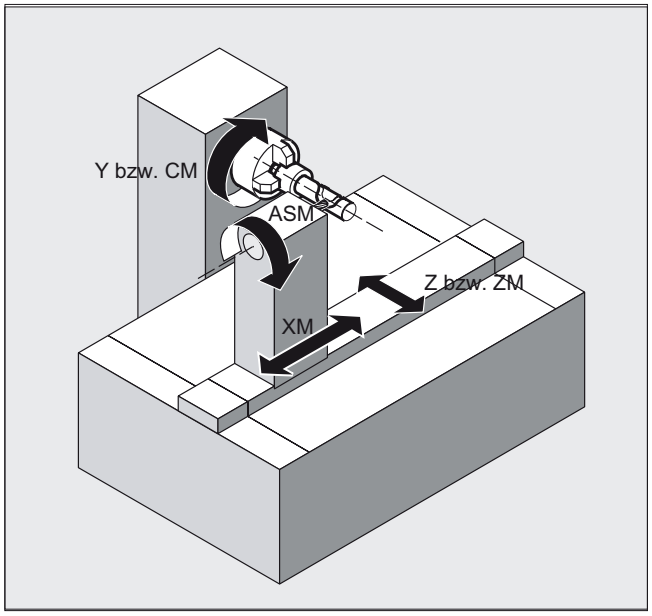
Beschreibung

Ohne Nutwandkorrektur (Transformationstyp 512):

Die Steuerung transformiert die programmierten Verfahrbewegungen des Zylinder-Koordinatensystems auf die Verfahrbewegungen der realen Maschinenachsen:

- Rundachse
- Zustellachse senkrecht zur Drehachse
- Längsachse parallel zur Drehachse

Die Linearachsen stehen senkrecht aufeinander. Die Zustellachse schneidet die Rundachse.

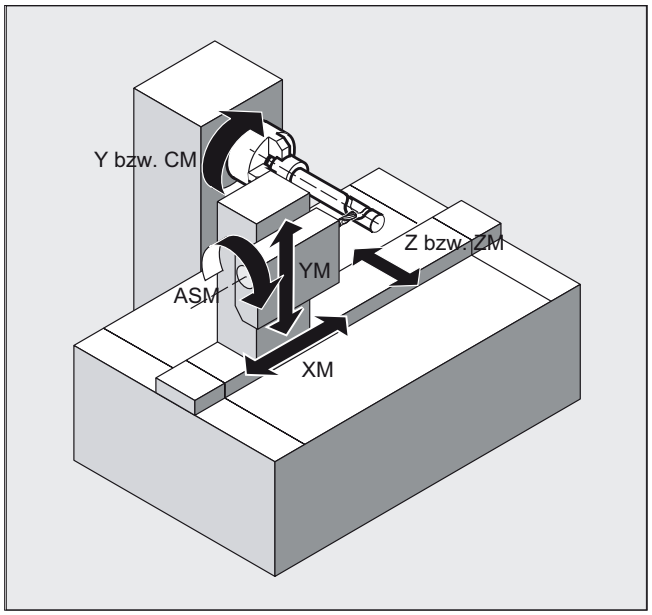


Mit Nutwandkorrektur (Transformationstyp 513):

Kinematik wie oben, aber zusätzlich –Längsachse parallel zur Umfangsrichtung

Die Linearachsen stehen senkrecht aufeinander.

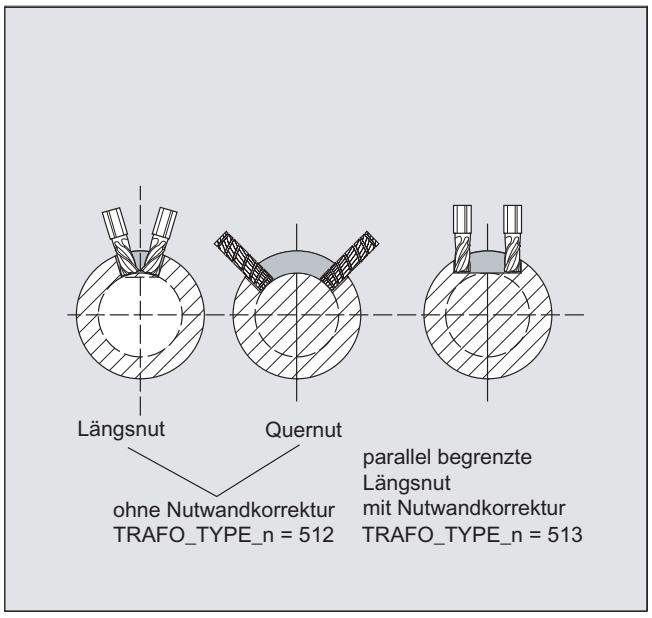
Die Geschwindigkeitsführung berücksichtigt die für die Drehbewegungen definierten Begrenzungen.



Nutquerschnitt

Bei Achskonfiguration 1 sind Nuten längs zur Rundachse nur dann parallel begrenzt, wenn die Nutbreite genau dem Werkzeugradius entspricht.

Nuten parallel zum Umfang (Quernuten) sind an Anfang und Ende nicht parallel.



Mit zusätzlicher Linearachse und mit Nutwandkorrektur (Transformationstyp 514):

Diese Transformationsvariante nützt bei einer Maschine mit einer weiteren Linearachse die Redundanz aus, um eine verbesserte Werkzeugkorrektur durchzuführen. Für die zweite Linearachse gilt dann:

- ein kleinerer Arbeitsbereich und
- dass die zweite Linearachse für das Abfahren des Teileprogramms nicht genutzt werden soll.

Für das Teileprogramm und die Zuordnung der entsprechenden Achsen im BKS oder MKS werden bestimmte Maschinendateneinstellungen vorausgesetzt, siehe

Literatur

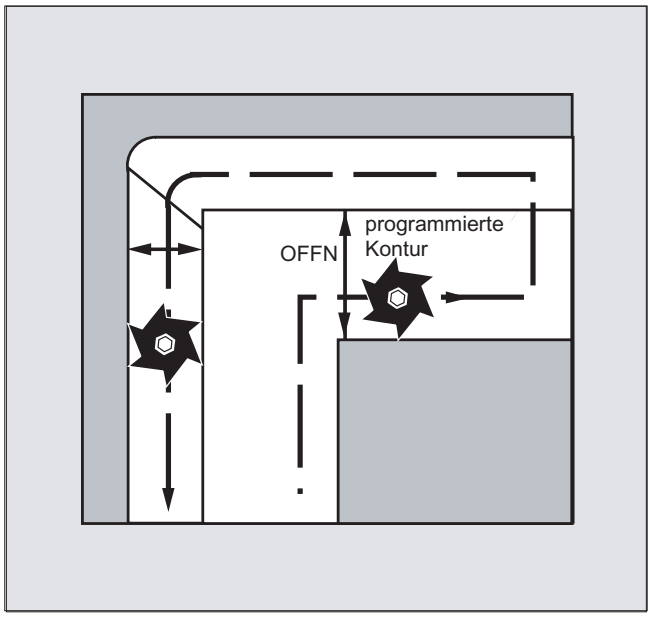
/FB2/ Funktionshandbuch Erweiterungsfunktionen; Kinematische Transformationen (M1)

Offset Kontur-normal OFFN (Transformationstyp 513)

Um mit `TRACYL` Nuten zu fräsen, wird im

- Teileprogramm die Nutmittenlinie,
- über `OFFN` **die halbe Nutbreite** programmiert.

`OFFN` wird erst mit angewählter Werkzeugradiuskorrektur wirksam, um eine Beschädigung der Nutwand zu vermeiden). **Ferner sollte OFFN>=Werkzeugradius sein, um eine Beschädigung der gegenüberliegenden Nutwand auszuschließen.**



Ein Teileprogramm zum Fräsen einer Nut besteht in der Regel aus folgenden Schritten:

1. Werkzeug anwählen
2. TRACYL anwählen
3. Passende Koordinatenverschiebung (FRAME) anwählen
4. Positionieren
5. OFFN programmieren
6. WRK anwählen
7. Anfahrsatz (Einfahren der WRK und Anfahren der Nutwand)
8. Kontur der Nutmittenlinie
9. WRK abwählen
10. Abfahrsatz (Ausfahren der WRK und Wegfahren von der Nutwand)
11. Positionieren
12. TRAFOOF
13. Ursprüngliche Koordinatenverschiebung (FRAME) wieder anwählen

Besonderheiten

- WRK-Anwahl:

  WRK wird nicht hinsichtlich der Nutwand, sondern relativ zur programmierten Nutmittenlinie programmiert. Damit das Werkzeug links von der Nutwand fährt, wird G42 eingegeben (anstatt G41). Sie vermeiden dies, wenn in OFFN die Nutbreite mit negativem Vorzeichen eingetragen wird.

- OFFN mit TRACYL wirkt sich anders aus als ohne TRACYL. Da OFFN auch ohne TRACYL bei aktiver WRK eingerechnet wird, sollte OFFN nach TRAFOOF wieder zu Null gesetzt werden.
- Eine Änderung von OFFN innerhalb des Teileprogramms ist möglich. Damit könnte die Nutmittenlinie aus der Mitte verschoben werden (siehe Bild).
- Führungsnuten:

  Mit TRACYL wird nicht dieselbe Nut bei Führungsnuten erzeugt als wäre diese mit einem Werkzeug gefertigt worden, dessen Durchmesser die Nutbreite aufweist. Es ist prinzipiell nicht möglich, mit einem kleineren zylindrischen Werkzeug dieselbe Nutwandgeometrie zu erzeugen wie mit einem größeren. TRACYL minimiert den Fehler. Um Genauigkeitsprobleme zu vermeiden, sollte der Werkzeugradius nur wenig kleiner als die halbe Nutbreite sein.

Hinweis

OFFN und WRK

Bei TRAFO_TYPE_n = 512 wirkt der Wert unter OFFN als Aufmass zur WRK.

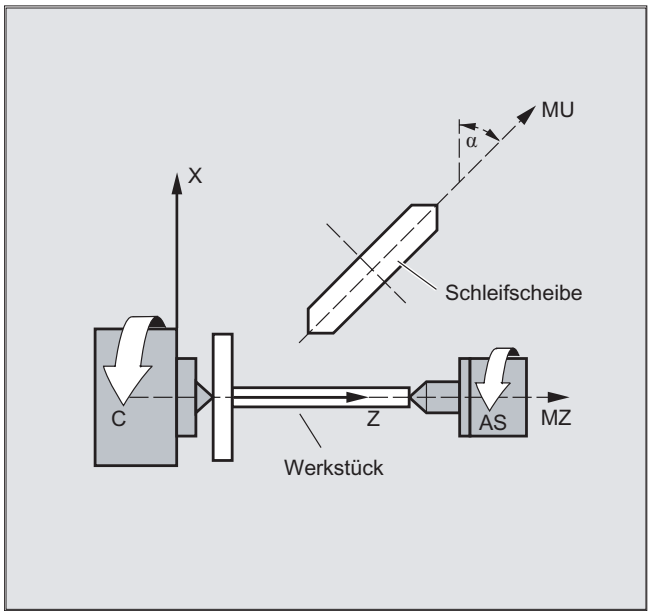
Bei TRAFO_TYPE_n = 513 wird im OFFN die halbe Nutbreite programmiert. Die Kontur wird mit OFFN-WRK abgefahren.

6.8.3 Schräge Achse (TRAANG)

Funktion

Die Funktion Schräge Achse ist für die Technologie Schleifen gedacht und ermöglicht folgende Leistungen:

- Bearbeitung mit schräger Zustellachse
- Für die Programmierung kann ein kartesisches Koordinatensystem verwendet werden.
- Die Steuerung transformiert die programmierten Verfahrbewegungen des kartesischen Koordinatensystems auf die Verfahrbewegungen der realen Maschinenachsen (Standardfall): schräge Zustellachse.



Syntax

`TRAANG(α)` oder `TRAANG(α, n)`
`TRAFOOF`

Bedeutung

`TRAANG( )`
oder
`TRAANG( ,n)`
	Transformation mit der Parametrierung der vorhergehenden Anwahl aktivieren.

`TRAANG(α)`	Aktiviert die erste vereinbarte Transformation Schräge Achse
`TRAANG(α,n)`	Aktiviert die n. vereinbarte Transformation Schräge Achse. n darf maximal 2 sein. TRAANG(α,1) entspricht TRAANG(α).

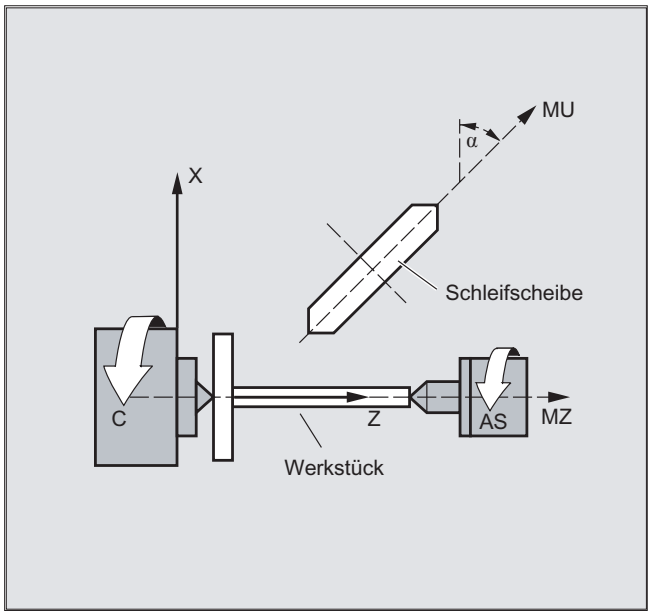
αA	Winkel der schrägstehenden Achse Zulässige Werte für α sind: -90 Grad < α < + 90 Grad
TRAFOOF	Transformation aus
n	Anzahl vereinbarte Transformationen

Winkel α weglassen oder Null

Wird der Winkel α weggelassen (z. B. `TRAANG()`, `TRAANG(, n)`), wird die Transformation mit der Parametrierung der vorhergehenden Anwahl aktiviert. Bei der ersten Anwahl gilt die Vorbelegung gemäß den Maschinendaten.

Ein Winkel α = 0 (z. B. `TRAANG(0)`, `TRAANG(0,n)`) ist eine gültige Parametrierung und entspricht nicht mehr dem Weglassen des Parameters bei älteren Versionen.

Beispiel



Programmcode	Kommentar
N10 G0 G90 Z0 MU=10 G54 F5000 ->	; Werkzeuganwahl, Aufspannkompensation,
-> G18 G64 T1 D1	Ebenenanwahl
N20 TRAANG(45)	; Transformation Schräge Achse einschalten
N30 G0 Z10 X5	; Anfahren der Ausgangsstellung
N40 WAITP(Z)	; Achsen zum Pendeln freigeben
N50 OSP[Z]=10 OSP2[Z]=5 OST1[Z]=-2 ->	; Pendeln, bis Maß erreicht
-> OST2[Z]=-2 FA[Z]=5000	(Pendeln siehe Kapitel "Pendeln")
N60 OS[Z]=1	
N70 POS[X]=4.5 FA[X]=50	
N80 OS[Z]=0	

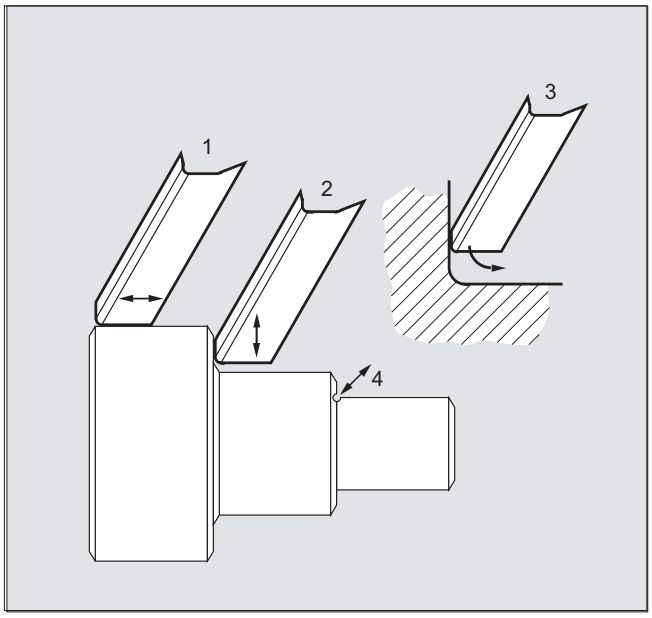
Programmcode	Kommentar
N90 WAITP(Z)	; Pendelachsen als Positionierachsen freigeben
N100 TRAFOOF	; Transformation ausschalten
N110 G0 Z10 MU=10	; Freifahren
N120 M30	;

-> in einem Satz programmieren

Beschreibung

Folgende Bearbeitungen sind möglich:

1. Längsschleifen
2. Planschleifen
3. Schleifen einer bestimmten Kontur
4. Schrägeinstechschleifen



Maschinenhersteller

Folgende Einstellungen werden über Maschinendatum festgelegt:

- der Winkel zwischen einer Maschinenachse und der schrägen Achse,
- die Lage des Werkzeugnullpunktes bezogen auf den Ursprung des bei der Funktion "Schräge Achse" vereinbarten Koordinatensystems,
- die Geschwindigkeitsreserve, die auf der parallelen Achse für die Ausgleichsbewegung bereitgehalten wird,
- die Achsbeschleunigungsreserve, die auf der parallelen Achse für die Ausgleichsbewegung bereitgehalten wird.

Achskonfiguration

Um im kartesischen Koordinatensystem programmieren zu können, muss der Steuerung der Zusammenhang zwischen diesem Koordinatensystem und den tatsächlich existierenden Maschinenachsen (MU, MZ) mitgeteilt werden:

- Benennung der Geometrieachsen
- Zuordnung der Geometrieachsen zu Kanalachsen
  - allgemeiner Fall (Schräge Achse nicht aktiv)
  - Schräge Achse aktiv
- Zuordnung der Kanalachsen zu den Maschinenachsnummern
- Kennzeichnung der Spindeln
- Zuweisung von Maschinenachsnamen

Das Vorgehen entspricht mit Ausnahme von "Schräge Achse aktiv" dem Vorgehen bei der normalen Achskonfiguration.

6.8.4 Schräge Achse programmieren (G05, G07)

Funktion

Im JOG–Betrieb kann die Schleifscheibe wahlweise kartesisch oder in Richtung der Schrägen Achse bewegt (Anzeige bleibt kartesisch) werden. Es bewegt sich nur die reale U–Achse, die Anzeige der Z–Achse wird aktualisiert.

REPOS–Verschiebungen müssen im Jog–Betrieb kartesisch zurückgefahren werden.

Das Überfahren der kartesischen Arbeitsfeldbegrenzung wird im JOG–Betrieb bei aktivem "PTP–Fahren" überwacht, die entsprechende Achse wird vorher gebremst. Ist "PTP–Fahren" nicht aktiv, kann die Achse exakt bis zur Arbeitsfeldbegrenzung gefahren werden.

Literatur

/FB2/ Funktionshandbuch Erweiterungsfunktionen; Kinematische Transformation (M1)

Syntax

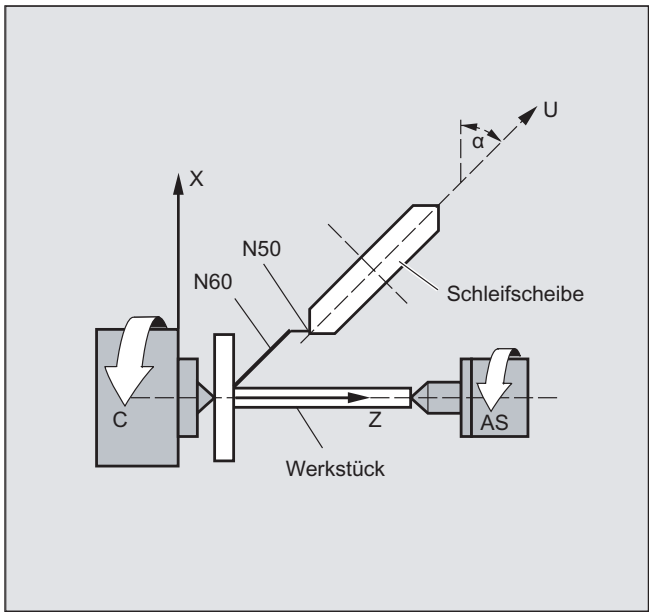
```
G07
G05
```

Die Befehle `G07/G05` dienen der Erleichterung der Programmierung der Schrägen Achse. Dabei können Positionen im kartesischen Koordinatensystem programmiert und angezeigt werden. Die Werkzeugkorrektur und Nullpunktverschiebung werden kartesisch eingerechnet. Nach der Programmierung des Winkels für die Schräge Achse im NC–Programm kann die Startposition angefahren werden (`G07`) und danach das Schrägeinstechen (`G05`) vollzogen werden.

Bedeutung

G07	Startposition anfahren
G05	Aktiviert Schrägeinstechen

Beispiel




```
Programmierung                          Kommentar
N.. G18                  ; Winkel für die Schräge Achse programmieren
N50 G07 X70 Z40 F4000    ; Startposition anfahren
N60 G05 X70 F100         ; Schräg einstechen
N70 ...                  ;
```

6.9 Kartesisches PTP-Fahren

Funktion

Mit dieser Funktion kann eine Position in einem kartesischen Koordinatensystem programmiert werden, die Bewegung der Maschine erfolgt aber in Maschinenkoordinaten. Die Funktion kann beispielsweise beim Wechseln der Gelenkstellung angewendet werden, wenn dabei die Bewegung durch eine Singularität führt.

Hinweis

Die Funktion ist nur in Verbindung mit einer aktiven Transformation sinnvoll. Weiterhin ist das "PTP-Fahren" nur in Verbindung mit G0 und G1 zulässig.

Syntax

```
N... TRAORI
N... STAT='B10' TU='B100' PTP
N... CP
```

PTP-Fahren bei generischer 5/6-Achs Transformation

Wird bei aktiver generischer 5/6-Achs-Transformation mit PTP ein Punkt-zu-Punkt Fahren im Maschinenkoordinatensystem `(ORIMKS)` aktiviert, dann kann die Werkzeugorientierung sowohl mit Rundachspositionen

```
N... G1 X Y Z A B C
```

als auch mit von der Kinematik unabhängigen Vektoren Euler- bzw. RPY-Winkel

`N... ORIEULER` oder `ORIRPY`

```
N... G1 X Y Z A2 B2 C2
```

oder den Richtungsvektoren

```
N... G1 X Y Z A3 B3 C3
```

programmiert werden. Dabei kann sowohl Rundachsinterpolation als auch Vektorinterpolation mit Großkreisinterpolation `ORIVECT` oder Interpolation des Orientierungsvektors auf einer Kegelmantelfläche `ORICONxx` aktiv sein.

Mehrdeutigkeiten der Orientierung mit Vektoren

Bei der Programmierung der Orientierung mit Vektoren gibt es eine Mehrdeutigkeit in den möglichen Rundachspositionen. Die anzufahrenden Rundachspositionen können dabei durch die Programmierung von `STAT = <...>` ausgewählt werden. Wird

`STAT = 0` programmiert wird (dies entspricht der Standardeinstellung),
werden die Positionen, die den kürzesten Abstand zu den Startpositionen haben angefahren. Wenn

`STAT = 1` programmiert wird,
werden die Positionen, die den längeren Abstand zu den Startpositionen haben angefahren.

Bedeutung

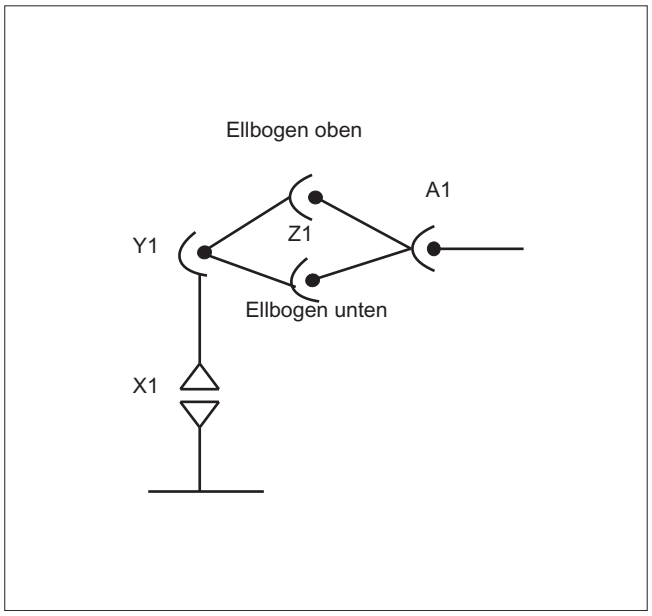
Die Befehle `PTP` und `CP` sind modal wirksam. `CP` ist die Standardeinstellung.

Während die Programmierung des STAT–Wertes modal gültig ist, wirkt die Programmierung von TU = <...> satzweise.

Ein weiterer Unterschied ist auch, dass die Programmierung eines STAT–Wertes sich nur bei Vektorinterpolation auswirkt, während die Programmierung von TU auch bei aktiver Rundachsinterpolation ausgewertet wird.

`PTP`	**P**oint **t**o **P**oint (Punkt zu Punkt Bewegung) Die Bewegung wird als Synchronachsbewegung ausgeführt; die langsamste an der Bewegung beteiligte Achse ist die dominierende Achse für die Geschwindigkeit.
`CP`	**c**ontinuous **p**ath (Bahnbewegung) Die Bewegung wird als kartesische Bahnbewegung ausgeführt.
`STAT=`	Stellung der Gelenke; Wert ist abhängig von der Transformation.
`TU=`	TURN-Information ist satzweise wirksam. Dadurch ist es möglich, Achswinkel zwischen -360 Grad und +360 Grad eindeutig anzufahren.

Beispiel




```
N10 G0 X0 Y-30 Z60 A-30 F10000
```
Ausgangsstellung
→ Ellbogen oben

```
N20 TRAORI(1)
```
Transformation ein

```
N30 X1000 Y0 Z400 A0
```

```
N40 X1000 Z500 A0 STAT='B10'
TU='B100' PTP
```
Umorientierung ohne Transformation
→ Ellbogen unten

```
N50 X1200 Z400 CP                    Transformation wieder aktiv
N60 X1000 Z500 A20
N70 M30
```

Beispiel PTP-Fahren bei generischer 5-Achs Transformation

Annahme: Es liegt eine rechtwinklige CA–Kinematik zu Grunde.

Programmcode	**Kommentar**
TRAORI	; Transformation CA-Kinematik ein
PTP	; PTP-Fahren einschalten
N10 A3 = 0 B3 = 0 C3 = 1	; Rundachspositionen C = 0 A = 0
N20 A3 = 1 B3 = 0 C3 = 1	; Rundachspositionen C = 90 A = 45
N30 A3 = 1 B3 = 0 C3 = 0	; Rundachspositionen C = 90 A = 90
N40 A3 = 1 B3 = 0 C3 = 1 STAT = 1	; Rundachspositionen C = 270 A = –45

Eindeutige Anfahrstellung der Rundachsposition auswählen:

Im Satz N40 fahren dabei die Rundachsen durch die Programmierung von `STAT = 1` den längeren Weg von ihrem Startpunkt (C=90, A=90) zum Endpunkt (C=270, A=–45), anstatt wie es bei `STAT = 0` der Fall wäre den kürzesten Weg zum Endpunkt (C=90, A=45).

Beschreibung

Die Umschaltung zwischen dem kartesischen Verfahren und dem Verfahren der Maschinenachsen erfolgt durch die Befehle `PTP` und `CP`.

PTP-Fahren bei generischer 5/6-Achs Transformation

Beim PTP–Fahren bleibt im Gegensatz zur 5/6–Achs Transformation die TCP allgemein nicht ortsfest, falls sich nur die Orientierung ändert. Es werden die transformierten Endpositionen aller Transformationsachsen (3 Linearachsen und bis zu 3 Rundachsen) linear angefahren, ohne dass dabei die Transformation im eigentlichen Sinn noch aktiv ist.

Das PTP–Fahren wird durch Programmierung des modalen G–Codes `CP` ausgeschaltet.

Die unterschiedlichen Transformationen sind enthalten in der Druckschrift:
/FB3/ Funktionshandbuch Sonderfunktionen; Transformationspaket Handling (TE4).

Programmierung der Stellung (STAT=)

Eine Maschinenstellung ist allein durch die Positionsangabe mit kartesischen Koordinaten und der Orientierung des Werkzeugs nicht eindeutig bestimmt. Je nachdem, um welche Kinematik es sich handelt, existieren bis zu 8 unterschiedliche bzw. unterscheidende Gelenkstellungen. Diese sind damit transformationsspezifisch. Um eine kartesische Position eindeutig in die Achswinkel umrechnen zu können, muss die Stellung der Gelenke mit dem Befehl `STAT=` angegeben werden. Der Befehl "`STAT`" enthält als Binärwert für jede der möglichen Stellungen ein Bit.

Die Stellungsbits, welche bei "`STAT`" zu programmieren sind, siehe:
/FB2/ Funktionshandbuch Erweiterungsfunktionen; Kinematische Transformation (M1), Kapitel "Kartesisches PTP-Fahren".

Programmierung der Achswinkel (TU=)

Um Achswinkel < ±360 Grad eindeutig anfahren zu können, muss diese Information mit dem Befehl "TU= " programmiert werden.

Die Achsen verfahren auf kürzestem Weg:

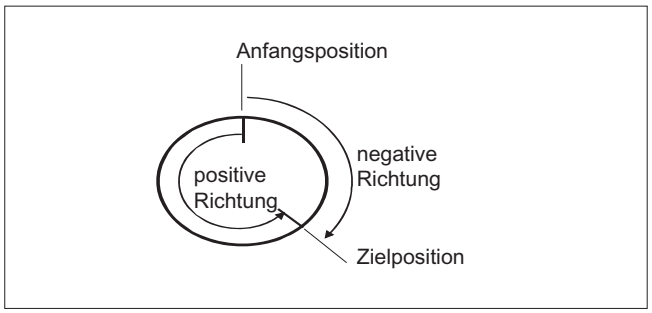
- wenn bei einer Position kein TU programmiert wird,
- bei Achsen, welche einen Verfahrbereich > ±360 Grad besitzen.

Beispiel:

Die im Bild angegebene Zielposition kann in negativer oder in positiver Richtung angefahren werden. Unter der Adresse A1 wird die Richtung programmiert.

A1=225°, TU=Bit 0, → positive Richtung

A1=−135°, TU=Bit 1, → negative Richtung



Beispiel Auswertung von TU für generische 5/6-Achs Transformation und Zielpositionen

Die Variable TU enthält für jede Achse, die in die Transformation eingeht, ein Bit, das die Verfahrrichtung anzeigt. Die Zuordnung der TU–Bits entspricht der Kanalachssicht der Rundachsen. Die TU–Information wird nur bei den bis zu 3 möglichen Rundachsen, die in die Transformation eingehen, ausgewertet:

Bit0: Achse 1, TU–Bit = 0 : 0 Grad <= Rundachswinkel < 360 Grad

Bit1: Achse 2, TU–Bit = 1: –360 Grad < Rundachswinkel < 0 Grad

Die Startposition einer Rundachse ist C = 0, durch die Programmierung von C = 270 fährt die Rundachse auf folgende Zielpositionen:

C = 270: TU–Bit 0, positive Drehrichtung

C = –90: TU–Bit 1, negative Drehrichtung

Weiteres Verhalten

Betriebsartenwechsel

Die Funktion "Kartesisches PTP-Fahren" ist nur in den Betriebsarten AUTO und MDA sinnvoll. Beim Wechsel der Betriebsart nach JOG bleibt die aktuelle Einstellung erhalten.

Wenn der G-Code `PTP` eingestellt ist, werden die Achsen im MKS verfahren. Wenn der G-Code `CP` eingestellt ist, werden die Achsen im WKS verfahren.

Power On/RESET

Nach Power On oder nach RESET ist die Einstellung abhängig vom Maschinendatum `$MC_GCODE_REST_VALUES[48]`. Standardmäßig ist die Verfahrart "`CP`" eingestellt.

REPOS

War während des Unterbrechungssatzes die Funktion "Kartesisches PTP-Fahren" eingestellt, wird auch mit `PTP` rückpositioniert.

Überlagerte Bewegungen

DRF-Verschiebung oder externe Nullpunktverschiebung sind beim kartesischen PTP-Fahren nur eingeschränkt möglich. Beim Wechsel von einer PTP- nach einer CP-Bewegung dürfen keine Überlagerungen im BKS vorhanden sind.

Überschleifen zwischen CP- und PTP-Bewegungen

Zwischen den Sätzen ist mit `G641` ein programmierbares Übergangsverschleifen möglich.

Die Größe des Verschleifbereiches ist der Bahnweg in mm oder Inch, ab dem bzw. zu dem der Satzübergang verschliffen wird. Die Größe ist wie folgt anzugeben:

- für G0-Sätze mit `ADISPOS`
- für alle anderen Wegbefehle mit `ADIS`

Die Bahnwegberechnung entspricht der Berücksichtigung der F-Adressen bei Nicht-G0-Sätzen. Der Vorschub wird auf die in `FGROUP(...)` angegebenen Achsen eingehalten.

Vorschubberechnung

Für CP-Sätze werden die kartesischen Achsen des Basiskoordinatensystems zur Berechnung verwendet.

Für PTP-Sätze werden die entsprechenden Achsen des Maschinenkoordinatensystems zur Berechnung verwendet.

6.9.1 PTP bei TRANSMIT

Funktion

Mit PTP bei TRANSMIT können G0- und G1-Sätze zeitoptimiert angefahren werden. Anstatt die Achsen des Basiskoordinatensystems linear zu verfahren (CP), werden die Maschinenachsen linear verfahren (PTP). Dadurch wirkt sich der Maschinenachsverlauf in Polnähe so aus, dass der Satzendpunkt erheblich schneller erreicht werden kann.

Das Teileprogramm wird weiterhin im kartesischen Werkstückkoordinatensystem geschrieben und alle Koordinatenverschiebungen, Drehungen und Frameprogrammierungen bleiben gültig. Die Simulation auf HMI, wird ebenfalls im kartesischen Werkstückkoordinatensystem angezeigt.

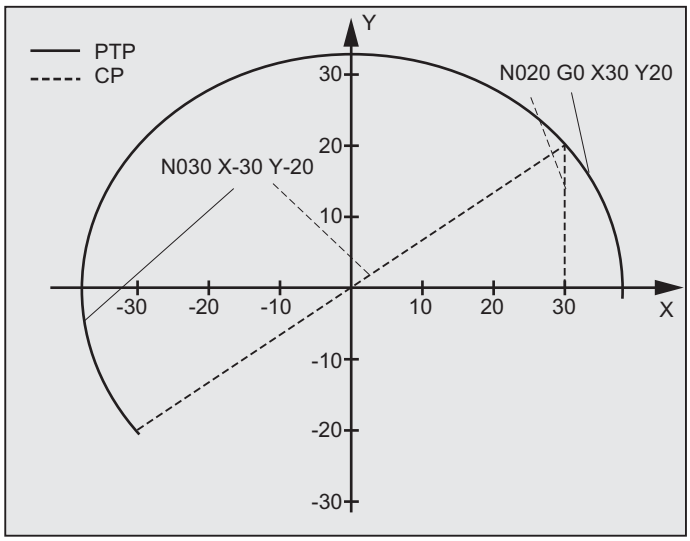
Syntax

```
N... TRANSMIT
N... PTPG0
N... G0 ...
...
N... G1 ...
```

Bedeutung

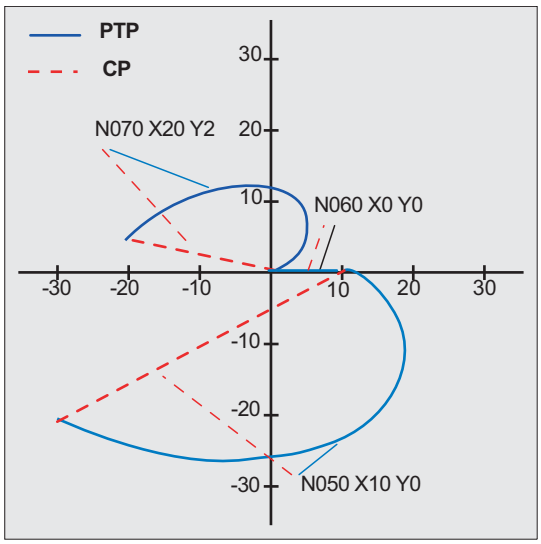
`TRANSMIT`	Aktiviert die erste vereinbarte TRANSMIT-Funktion (siehe Kapitel "Fräsbearbeitungen an Drehteilen: TRANSMIT")
`PTPG0`	**P**oint **t**o **P**oint **G0** (Punkt zu Punkt Bewegung automatisch zu jedem G0-Satz und danach wieder CP setzen) Da STAT und TU modal sind, gilt immer der zuletzt programmierte Wert.
`PTP`	**P**oint **t**o **P**oint (Punkt zu Punkt Bewegung) Für TRANSMIT bedeutet PTP, dass im Kartesischen auf Archimedischen Spiralen entweder um den Pol oder aus dem Pol herausgefahren wird. Die hieraus resultierenden Werkzeugbewegungen verlaufen deutlich anders als bei CP und sind in den jeweiligen Programmierbeispielen dargestellt.
`STAT=`	Auflösen der Mehrdeutigkeit hinsichtlich des Pols.
`TU=`	TU ist bei PTP bei TRANSMIT nicht relevant

Beispiel Umfahren des Poles mit PTP und TRANSMIT



Programmcode	**Kommentar**
N001 G0 X30 Z0 F10000 T1 D1 G90	; Ausgangsstellung Absolutmaß
N002 SPOS=0	
N003 TRANSMIT	; Transformation TRANSMIT
N010 PTPG0	; zu jedem G0-Satz automatisch PTP und danach wieder CP
N020 G0 X30 Y20	
N030 X-30 Y-20	
N120 G1 X30 Y20	
N110 X30 Y0	
M30	

Beispiel Herausfahren aus dem Pol mit PTP und TRANSMIT



Programmierung	Kommentar
N001 G0 X90 Z0 F10000 T1 D1 G90	; Ausgangsstellung
N002 SPOS=0	
N003 TRANSMIT	; Transformation TRANSMIT
N010 PTPG0	; zu jedem G0-Satz automatisch PTP und danach wieder CP
N020 G0 X90 Y60	
N030 X-90 Y-60	
N040 X-30 Y-20	
N050 X10 Y0	
N060 X0 Y0	
N070 X-20 Y2	
N170 G1 X0 Y0	
N160 X10 Y0	
N150 X-30 Y-20	
M30	

Beschreibung

PTP und PTPG0

PTPG0 wird bei allen Transformationen berücksichtigt, die PTP abarbeiten können. In allen anderen Fällen ist PTPG0 nicht relevant.

G0-Sätze werden im CP-Mode abgefahren.

Die Anwahl von PTP bzw. PTPG0 erfolgt im Teileprogramm oder durch die Abwahl von CP im Maschinendatum $MC_GCODE_RESET_VALUES[48].

⚠VORSICHT

Randbedingungen

Hinsichtlich der Werkzeugbewegungen und Kollision gelten mehrere Randbedingungen und bestimmte Funktionsausschlüsse wie:

Mit PTP darf keine Werkzeugradiuskorrektur (WRK) aktiv sein.

Mit PTPG0 wird bei aktiver Werkzeugradiuskorrektur (WRK) per CP gefahren.

Mit PTP ist Weiches An- und Abfahren (WAB) nicht möglich.

Mit PTPG0 wird bei Weichem An- und Abfahren (WAB) per CP gefahren.

Mit PTP sind Abspanzyklen (CONTPRON, CONTDCON) nicht möglich.

Mit PTPG0 wird in Abspanzyklen (CONTPRON, CONTDCON) per CP gefahren.

Fase (CHF, CHR) und Rundung (RND, RNDM) werden ignoriert.

Kompressor ist nicht verträglich mit PTP und wird automatisch in PTP-Sätzen abgewählt.

Eine Achsüberlagerung in der Interpolation darf sich während des PTP-Abschnittes nicht ändern.

Bei G643 wird automatisch nach Überschleifen mit axialer Genauigkeit G642 umgeschaltet.

Bei aktivem PTP können Achsen der Transformation nicht gleichzeitig Positionierachsen sein.

Literatur:

/FB2/ Funktionshandbuch Erweiterungsfunktionen; Kinematische Transformation (M1), Kapitel "Kartesisches PTP-Fahren"

PTP bei TRACON:

PTP kann auch mit TRACON genutzt werden, wenn die erste verkettete Transformation PTP unterstützt.

Bedeutung von STAT= und TU= bei TRANSMIT

Soll die Rundachse um 180 Grad drehen, bzw. die Kontur bei CP durch den Pol führen, können Rundachsen abhängig vom Maschinendatum $MC_TRANSMIT_POLE_SIDE_FIX_1/2 [48] um -/+ 180 Grad gedreht und im oder gegen den Uhrzeigersinn verfahren werden. Ebenso kann eingestellt werden, ob durch den Pol gefahren wird, oder um den Pol gedreht wird.

6.10 Randbedingungen bei der Anwahl einer Transformation

Funktion

Die Anwahl von Transformationen ist über Teileprogramm bzw. MDA möglich. Dabei ist zu beachten:

- Ein Bewegungszwischensatz wird nicht eingefügt (Fasen/Radien).
- Eine Spline-Satzfolge muss abgeschlossen sein; wenn nicht, erscheint eine Meldung.
- Werkzeugfeinkorrektur muss abgewählt sein (FTOCOF); wenn nicht, erscheint eine Meldung.
- Werkzeugradiuskorrektur muss abgewählt sein (G40); wenn nicht, erscheint eine Meldung.
- Eine aktivierte Werkzeuglängenkorrektur wird von der Steuerung in die Transformation übernommen.
- Der vor der Transformation wirksame aktuelle Frame wird von der Steuerung abgewählt.
- Eine aktive Arbeitsfeldbegrenzung wird für die von der Transformation betroffenen Achsen von der Steuerung abgewählt (entspricht WALIMOF).
- Schutzbereichsüberwachung wird abgewählt.
- Bahnsteuerbetrieb und Überschleifen werden unterbrochen.
- Alle in dem Maschinendatum angegebenen Achsen müssen satzbezogen synchronisiert sein.
- Getauschte Achsen werden zurückgetauscht; wenn nicht, erscheint eine Meldung.
- Bei abhängigen Achsen wird eine Meldung ausgegeben.

Werkzeugwechsel

Ein Werkzeugwechsel ist nur bei abgewählter Werkzeugradiuskorrektur zulässig.

Ein Wechsel der Werkzeuglängenkorrektur und eine An-/Abwahl der Werkzeugradiuskorrektur darf nicht im selben Satz programmiert sein.

Framewechsel

Alle Anweisungen, die sich nur auf das Basis-Koordinatensystem beziehen, sind erlaubt (FRAME, Werkzeugradiuskorrektur). Ein Framewechsel bei G91 (Kettenmaß) wird aber – anders als bei inaktiver Transformation – nicht gesondert behandelt. Das zu fahrende Inkrement wird im Werkstück-Koordinatensystem des neuen Frames ausgewertet – unabhängig davon, welches Frame im Vorgängersatz wirkte.

Ausschlüsse

Von der Transformation betroffene Achsen können nicht verwendet werden:

- als Preset-Achse (Alarm),
- für das Fixpunktanfahren (Alarm),
- zum Referieren (Alarm).

6.11 Transformation abwählen (TRAFOOF)

Funktion

Mit dem Befehl `TRAFOOF` werden alle aktiven Transformationen und Frames ausgeschaltet.

Hinweis

Danach benötigte Frames müssen durch erneute Programmierung aktiv geschaltet werden.

Dabei ist zu beachten:

Für die Abwahl der Transformation gelten dieselben Randbedingungen wie für die Anwahl (siehe Kapitel "Randbedingungen bei der Anwahl einer Transformation").

Syntax

```
TRAFOOF
```

Bedeutung

`TRAFOOF`	Befehl zum Ausschalten aller aktiven Transformationen/Frames

6.12 Verkettete Transformationen (TRACON, TRAFOOF)

Funktion

Jeweils **zwei** Transformationen können hintereinander geschaltet (verkettet) werden, so dass die Bewegungsanteile für die Achsen aus der ersten Transformation Eingangsdaten für die verkettete zweite Transformation sind. Die Bewegungsanteile aus der zweiten Transformation wirken auf die Maschinenachsen.

Die Kette darf **zwei** Transformationen umfassen.

Hinweis

Ein Werkzeug wird immer der ersten Transformation einer Kette zugeordnet. Die nachfolgende Transformation verhält sich dann so, als ob die aktive Werkzeuglänge Null wäre. Es sind nur die über Maschinendaten eingestellten Basislängen eines Werkzeuges (_BASE_TOOL_) für die erste Transformation der Kette wirksam.

Maschinenhersteller

Beachten Sie die Hinweise des Maschinenherstellers zu ggf. durch Maschinendaten vordefinierten Transformationen.

Transformationen und verkettete Transformationen sind Optionen. Über die Verfügbarkeit bestimmter Transformationen in der Kette in bestimmten Steuerungen gibt jeweils der aktuelle Katalog Auskunft.

Anwendungen

- Schleifen von Konturen, die als Mantellinie einer Zylinderabwicklung programmiert wurden (TRACYL) mit einer schräg stehenden Schleifscheibe z. B. Werkzeugschleifen.
- Feinbearbeitung einer mit TRANSMIT erzeugten nicht runden Kontur mit schräg stehender Schleifscheibe.

Syntax

`TRACON(trf,par)`	Eine verkettete Transformation wird eingeschaltet.
`TRAFOOF`	

Bedeutung

TRACON	Die verkettete Transformation wird eingeschaltet. Eine zuvor aktivierte andere Transformation wird durch TRACON() implizit ausgeschaltet.
TRAFOOF	Die zuletzt eingeschaltete (verkettete) Transformation wird ausgeschaltet.
trf	Nummer der verketteten Transformation: 0 oder 1 für erste/einzige verkettete Transformation. Ist an dieser Stelle nichts programmiert, ist das gleichbedeutend mit der Angabe des Wertes 0 oder 1, d. h. es wird die erste/einzige Transformation aktiviert. 2 für die zweite verkettete Transformation. (Werte ungleich 0 - 2 erzeugen einen Fehleralarm).
par	Ein oder mehrere durch Komma getrennte Parameter für die Transformationen in der Verkettung, die Parameter erwarten, z. B. Winkel der schrägen Achse. Bei nicht gesetzten Parametern werden die Voreinstellungen oder die zuletzt benutzten Parameter wirksam. Durch Kommasetzung muss dafür gesorgt werden, dass die angegeben Parameter in der Reihenfolge ausgewertet werden, in der sie erwartet werden, wenn für vorher stehende Parameter Voreinstellungen wirken sollen. Insbesondere muss bei Angabe mindestens eines Parameters vor diesem ein Komma stehen, auch wenn die Angabe von trf nicht notwendig ist, also beispielsweise TRACON(, 3.7).

Voraussetzung

Die **zweite** Transformation muss **"Schräge Achse"** (TRAANG) sein. Als erste Transformation sind möglich:

- Orientierungstransformationen (TRAORI), einschließlich Kardanischer Fräskopf
- TRANSMIT
- TRACYL
- TRAANG

Für die Benutzung des Einschaltbefehles für eine verkettete Transformation ist Voraussetzung, dass die einzelnen zu verkettenden Transformationen und die zu aktivierende verkettete Transformation durch Maschinendaten definiert sind.

Die in den Einzelbeschreibungen für die Transformationen angegebenen Randbedingungen und Sonderfälle sind auch bei der Benutzung innerhalb einer Verkettung zu beachten.

Informationen zur Projektierung der Maschinendaten der Transformationen finden Sie im:

/FB2/ Funktionshandbuch Erweiterungsfunktionen; Kinematische Transformationen (M1) und

/FB3/ Funktionshandbuch Sonderfunktionen; 3- bis 5-Achs-Transformationen (F2).

Werkzeugkorrekturen 7

7.1 Korrekturspeicher

Funktion

Aufbau des Korrekturspeichers

Jedes Datenfeld ist mit einer T- und D-Nummer aufrufbar (außer "Flache D-Nr.") und enthält neben den geometrischen Angaben für das Werkzeug noch weitere Einträge, z. B. den Werkzeugtyp.

Flache D-Nummern-Struktur

Die "flache D-Nummern-Struktur" wird verwendet, wenn die Werkzeugverwaltung außerhalb des NCK erfolgt. In diesem Fall werden die D-Nummern mit den zugehörigen Werkzeugkorrektursätzen ohne Zuordnung zu Werkzeugen angelegt.

Im Teileprogramm kann weiterhin T programmiert werden. Dieses T hat aber keinen Bezug zur programmierten D-Nummer.

Anwender-Schneidendaten

Über Maschinendatum können Anwender-Schneidendaten konfiguriert werden. Bitte beachten Sie die Angaben des Maschinenherstellers.

Werkzeugparameter

Hinweis

Einzelne Werte im Korrekturspeicher

Die einzelnen Werte des Korrekturspeichers P1 bis P25 sind über Systemvariable vom Programm les- und schreibbar. Alle übrigen Parameter sind reserviert.

Die Werkzeugparameter $TC_DP6 bis $TC_DP8, $TC_DP10 und $TC_DP11 sowie $TC_DP15 bis $TC_DP17, $TC_DP19 und $TC_DP20 haben abhängig vom Werkzeugtyp eine andere Bedeutung.

[1]Gilt auch bei Fräswerkzeugen für das 3D-Stirnfräsen
[2]Bei Nutsäge Werkzeugtyp
[3]reserviert: Wird von SINUMERIK 840D nicht benutzt

Werkzeugparameter Nummer (DP)	Bedeutung der Systemvariablen	Bemerkung
$TC_DP1	Werkzeugtyp	Übersicht siehe Liste
$TC_DP2	Schneidenlage	nur für Drehwerkzeuge
Geometrie	**Längenkorrektur**	
$TC_DP3	Länge 1	Verrechnung nach
$TC_DP4	Länge 2	Typ und Ebene
$TC_DP5	Länge 3	
Geometrie	**Radius**	
$TC_DP6[1] $TC_DP6[2]	Radius 1 / Länge 1 Durchmesser d	Fräs-/Dreh-/Schleifwerkz. Nutsäge
$TC_DP7[1] $TC_DP7[2]	Länge 2 / Eckenradius kegelige Fräser Nutbreite b Eckenradius	Fräswerkzeuge Nutsäge
$TC_DP8[1] $TC_DP8[2]	Verrundungsradius 1 für Fräswerkzeuge Überstand k	Fräswerkzeuge Nutsäge
$TC_DP9[1,3]	Verrundungsradius 2	reserviert
$TC_DP10[1]	Winkel 1 Stirnseite des Werkzeugs	kegelige Fräswerkzeuge
$TC_DP11[1]	Winkel 2 Werkzeug-Längsachse	kegelige Fräswerkzeuge
Verschleiß	**Längen- und Radiuskorrektur**	
$TC_DP12	Länge 1	
$TC_DP13	Länge 2	
$TC_DP14	Länge 3	
$TC_DP15[1] $TC_DP15[2]	Radius 1 / Länge 1 Durchmesser d	Fräs-/Dreh-/Schleifwerkz. Nutsäge
$TC_DP16[1] $TC_DP16[3]	Länge 2 / Eckenradius kegelige Fräser Nutbreite b Eckenradius	Fräswerkzeuge Nutsäge
$TC_DP17[1] $TC_DP17[2]	Verrundungsradius 1für Fräswerkzeuge Überstand k	Fräsen / 3D Stirnfräsen Nutsäge
$TC_DP18[1,3]	Verrundungsradius 2	reserviert
$TC_DP19[1]	Winkel 1 Stirnseite des Werkzeugs	kegelige Fräswerkzeuge
$TC_DP20[1]	Winkel 2 Werkzeug-Längsachse	kegelige Fräswerkzeuge
Basismaß/Adapter	**Längenkorrekturen**	
$TC_DP21	Länge 1	
$TC_DP22	Länge 2	
$TC_DP23	Länge 3	
Technologie		
$TC_DP24	Freiwinkel	nur für Drehwerkzeuge
$TC_DP25		reserviert

Anmerkungen

Für die geometrischen Größen (z. B. Länge 1 oder Radius) bestehen mehrere Eintragskomponenten. Diese werden zu einer resultierenden Größe additiv verrechnet (z. B. Gesamtlänge 1, Gesamtradius), die dann zur Wirkung kommt.

Nicht benötigte Korrekturen sind mit dem Wert Null zu belegen.

Werkzeugparameter $TC-DP1 bis $TC-DP23 mit Konturwerkzeugen

Hinweis

Die Werkzeugparameter, die in der Tabelle nicht aufgeführt sind wie z.B. $TC_DP7, werden nicht ausgewertet, d. h. ihr Inhalt ist bedeutungslos.

Werkzeugparameter Nummer (DP)	Bedeutung	Schneiden Dn		Bemerkung
$TC_DP1	Werkzeugtyp			400 bis 599
$TC_DP2	Schneidenlage			
Geometrie	**Längenkorrektur**			
$TC_DP3	Länge 1			
$TC_DP4	Länge 2			
$TC_DP5	Länge 3			
Geometrie	**Radius**			
$TC_DP6	Radius			
Geometrie	**Grenzwinkel**			
$TC_DP10	minimaler Grenzwinkel			
$TC_DP11	maximaler Grenzwinkel			
Verschleiß	**Längen- und Radiuskorrektur**			
$TC_DP12	Verschleiß Länge 1			
$TC_DP13	Verschleiß Länge 2			
$TC_DP14	Verschleiß Länge 3			
$TC_DP15	Verschleiß Radius			
Verschleiß	**Grenzwinkel**			
$TC_DP19	Verschleiß min. Grenzwinkel			
$TC_DP20	Verschleiß max. Grenzwinkel			
Basismaß/Adapter	**Längenkorrekturen**			
$TC_DP21	Länge 1			
$TC_DP22	Länge 2			
$TC_DP23	Länge 3			

Grundwert und Verschleißwert

Die resultierenden Größen ergeben sich jeweils als Summe aus Grundwert und Verschleißwert (z. B. $TC_DP6 + $TC_DP15 für den Radius). Zur Werkzeuglänge der ersten Schneide wird außerdem noch das Basismaß ($TC_DP21 – $TC_DP23) addiert. Zusätzlich wirken auf diese Werkzeuglänge alle anderen Größen, die auch bei einem herkömmlichen Werkzeug die effektive Werkzeuglänge beeinflussen können (Adapter, orientierbarer Werkzeugträger, Settingdaten).

Grenzwinkel 1 und 2

Die Grenzwinkel 1 bzw. 2 beziehen sich jeweils auf den Vektor vom Schneidenmittelpunkt zum Schneidenbezugspunkt und werden im Gegenuhrzeigersinn gezählt.

7.2 Additive Korrekturen

7.2.1 Additive Korrekturen anwählen (DL)

Funktion

Additive Korrekturen können als in der Bearbeitung programmierbare Prozesskorrekturen betrachtet werden. Sie beziehen sich auf die geometrischen Daten einer Schneide und sind somit Bestandteil der Werkzeugschneidendaten.

Die Daten einer additiven Korrektur werden über eine DL-Nummer angesprochen (DL: Location dependent; Korrekturen bezüglich des jeweiligen Einsatzorts) und über die Bedienoberfläche eingegeben.

Anwendung

Durch additive Korrekturen können einsatzortbedingte Maßfehler ausgeglichen werden.

Syntax

```
DL=<Nummer>
```

Bedeutung

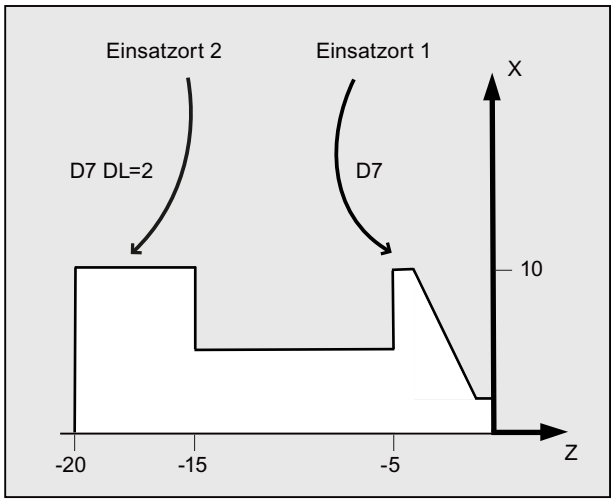
`DL`	Befehl zur Aktivierung einer additiven Korrektur
`<Nummer>`	Über den Parameter `<Nummer>` wird der zu aktivierende additive Werkzeugkorrekturdatensatz angegeben.

Hinweis

Die Festlegung von Anzahl und Aktivierung der additiven Korrekturen erfolgt über Maschinendaten (→ Angaben des Maschinenherstellers beachten!).

Beispiel

Die gleiche Schneide wird für 2 Lagersitze verwendet:



Programmcode	Kommentar
N110 T7 D7	; Der Revolver wird auf Platz 7 positioniert. D7 und DL=1 werden aktiviert und im nächsten Satz herausgefahren.
N120 G0 X10 Z1	
N130 G1 Z-6	
N140 G0 DL=2 Z-14	; Additiv zu D7 wird DL=2 aktiviert und im nächsten Satz herausgefahren.
N150 G1 Z-21	
N160 G0 X200 Z200	; Werkzeugwechselpunkt anfahren.
...	

7.2.2 Verschleiß- und Einrichtewerte festlegen ($TC_SCPxy[t,d], $TC_ECPxy[t,d])

Funktion

Verschleiß- und Einrichtewerte können über Systemvariablen gelesen und geschrieben werden. Dabei orientiert sich die Logik an der Logik der entsprechenden Systemvariablen für Werkzeuge und Schneiden.

Systemvariablen

Systemvariable	Bedeutung
$TC_SCPxy[<t>,<d>]	Verschleißwerte, die über xy dem jeweiligen Geometrieparameter zugeordnet sind, wobei x die Nummer des Verschleißwerts entspricht und y den Bezug zum Geometrieparameter herstellt.
$TC_ECPxy[<t>,<d>]	Einrichtewerte, die über xy dem jeweiligen Geometrieparameter zugeordnet sind, wobei x die Nummer des Einrichtewerts entspricht und y den Bezug zum Geometrieparameter herstellt.
<t>: T-Nummer des Werkzeugs <d>: D-Nummer der Schneide des Werkzeugs	

Hinweis

Die festgelegten Verschleiß- und Einrichtewerte werden zu den Geometrieparametern und den übrigen Korrekturparametern (D-Nummer) addiert.

Beispiel

Der Verschleißwert der Länge 1 wird für die Schneide <d> des Werkzeugs <t> auf den Wert 1.0 festgelegt.

Parameter: $TC_DP**3** (Länge 1, bei Drehwerkzeugen)

Verschleißwerte: $TC_SCP1**3** bis $TC_SCP6**3**

Einrichtewerte: $TC_ECP1**3** bis $TC_ECP6**3**

$TC_SCP4**3** [<t>,<d>] = 1.0

7.2.3 Additive Korrekturen löschen (DELDL)

Funktion

Mit dem Befehl DELDL werden additive Korrekturen für die Schneide eines Werkzeugs gelöscht (Freigabe von Speicher). Dabei werden sowohl die festgelegten Verschleißwerte als auch die Einrichtewerte gelöscht.

Syntax

```
DELDL[<t>,<d>]
DELDL[<t>]
DELDL
<Status>=DELDL[<t>,<d>]
```

Bedeutung

DELDL	Befehl zum Löschen additiver Korrekturen
<t>	T-Nummer des Werkzeugs
<d>	D-Nummer der Schneide des Werkzeugs
DELDL[<t>,<d>]	Es werden alle additiven Korrekturen der Schneide <d> des Werkzeugs <t> gelöscht.
DELDL[<t>]	Es werden alle additiven Korrekturen aller Schneiden des Werkzeugs <t> gelöscht.
DELDL	Es werden alle additiven Korrekturen aller Schneiden aller Werkzeuge der TO-Einheit gelöscht (für den Kanal, in dem der Befehl programmiert wird).
<Status>	Lösch-Status Wert: Bedeutung: 0 Das Löschen wurde erfolgreich durchgeführt. - Das Löschen wurde nicht durchgeführt (wenn die Parametrierung genau eine Schneide bezeichnet), oder das Löschen erfolgte nicht vollständig (wenn die Parametrierung mehrere Schneiden bezeichnet).

Hinweis

Verschleiß- und Einrichtewerte aktiver Werkzeuge können nicht gelöscht werden (verhält sich analog zum Löschverhalten von D bzw. Werkzeugdaten).

7.3 Werkzeugkorrektur - Sonderbehandlung

Funktion

Mit den Settingdaten SD42900 bis SD42960 lässt sich die Bewertung der Vorzeichen für Werkzeuglänge und Verschleiß steuern.

Das gilt ebenfalls für das Verhalten der Verschleißkomponenten beim Spiegeln von Geometrieachsen oder beim Wechsel der Bearbeitungsebene und auch zur Temperaturkompensation in Werkzeugrichtung.

Verschleißwerte

Wenn im Folgenden auf Verschleißwerte Bezug genommen wird, ist darunter jeweils die Summe aus den eigentlichen Verschleißwerten ($TC_DP12 bis $TC_DP20) und den Summenkorrekturen mit den Verschleißwerten ($SCPX3 bis $SCPX11) und Einrichtewerten ($ECPX3 bis $ECPX11) zu verstehen.

Näheres zu den Summenkorrekturen siehe:
Literatur:
Funktionshandbuch Werkzeugverwaltung

Settingdaten

Settingdatum	Bedeutung
SD42900 $SC_MIRROR_TOOL_LENGTH	Spiegeln von Werkzeuglängenkomponenten und Komponenten des Basismaßes.
SD42910 $SC_MIRROR_TOOL_WEAR	Spiegeln von Verschleißwerten der Werkzeuglängenkomponenten.
SD42920 $SC_WEAR_SIGN_CUTPOS	Vorzeichenbewertung der Verschleißkomponenten in Abhängigkeit von der Schneidenlage.
SD42930 $SC_WEAR_SIGN	Invertiert die Vorzeichen der Verschleißmaße.
SD42935 $SC_WEAR_TRANSFORM	Transformation der Verschleißwerte.
SD42940 $SC_TOOL_LENGTH_CONST	Zuordnung der Werkzeuglängenkomponenten zu den Geometrieachsen.
SD42950 $SC_TOOL_LENGTH_TYPE	Zuordnung der Werkzeuglängenkomponenten unabhängig vom Werkzeugtyp.
SD42960 $SC_TOOL_TEMP_COMP	Temperaturkompensationswert in Werkzeugrichtung. Ist auch bei vorhandener Werkzeugorientierung wirksam.

Literatur

Funktionshandbuch Grundfunktionen; Werkzeugkorrektur (W1)

Weitere Informationen

Wirksamwerden der veränderten Settingdaten

Die Neubewertung von Werkzeugkomponenten bei einer Änderung der beschriebenen Settingdaten wird erst wirksam, wenn das nächste Mal eine Werkzeugschneide angewählt wird. Ist ein Werkzeug bereits aktiv und die Bewertung der Daten dieses Werkzeugs soll verändert wirksam werden, muss dieses Werkzeug erneut angewählt werden.

Entsprechendes gilt für den Fall, dass sich die resultierende Werkzeuglänge ändert, weil der Spiegelungszustand einer Achse geändert wurde. Das Werkzeug muss nach dem Spiegelbefehl erneut angewählt werden, damit die geänderten Werkzeuglängenkomponenten wirksam werden.

Orientierbare Werkzeugträger und neue Settingdaten

Die Settingdaten SD42900 bis SD42940 wirken nicht auf die Komponenten eines eventuell aktiven orientierbaren Werkzeugträgers. Ein Werkzeug geht jedoch immer mit seiner gesamten resultierenden Länge (Werkzeuglänge + Verschleiß + Basismaß) in die Berechnung mit einem orientierbaren Werkzeugträger ein. Bei der Berechnung der resultierenden Gesamtlänge werden alle Änderungen berücksichtigt, die durch die Settingdaten verursacht wurden; d.h. Vektoren des orientierbaren Werkzeugträgers sind unabhängig von der Bearbeitungsebene.

Hinweis

Häufig wird es beim Einsatz orientierbarer Werkzeugträger sinnvoll sein, alle Werkzeuge für ein nicht gespiegeltes Grundsystem zu definieren, auch diejenigen, die nur bei Spiegelbearbeitung verwendet werden. Bei Bearbeitung mit gespiegelten Achsen wird dann der Werkzeugträger so gedreht, dass die tatsächliche Lage des Werkzeugs richtig beschrieben wird. Alle Werkzeuglängenkomponenten wirken dann automatisch in der richtigen Richtung, so dass sich eine Steuerung der Bewertung einzelner Komponenten über Settingdaten abhängig vom Spiegelungszustand einzelner Achsen erübrigt.

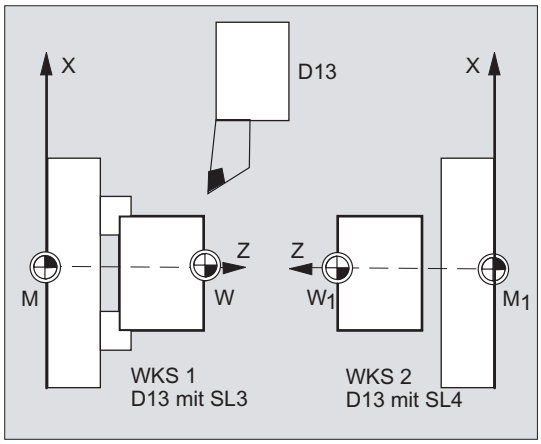
Weitere Anwendungsmöglichkeiten

Die Verwendung der Funktionalität orientierbarer Werkzeugträger kann auch dann sinnvoll sein, wenn an der Maschine physikalisch keine Möglichkeit vorgesehen ist, Werkzeuge zu drehen, Werkzeuge aber mit verschiedenen Orientierungen fest installiert sind. Die Werkzeugvermaßung kann dann einheitlich in einer Grundorientierung vorgenommen werden, und die für die Bearbeitung relevanten Maße ergeben sich durch Drehungen eines virtuellen Werkzeugträgers.

7.3.1 Werkzeuglängen spiegeln

Funktion

Mit gesetzten Settingdaten SD42900 $SC_MIRROR_TOOL_LENGTH und SD42910 $SC_MIRROR_TOOL_WEAR ungleich Null können Sie Werkzeuglängenkomponenten und Komponenten der Basismaße mit Verschleißwerten deren zugehörigen Achsen spiegeln.



SD42900 $SC_MIRROR_TOOL_LENGTH

Settingdatum **ungleich** Null:

Es werden die Werkzeuglängenkomponenten ($TC_DP3, $TC_DP4 und $TC_DP5) und die Komponenten der Basismaße ($TC_DP21, $TC_DP22 und $TC_DP23), deren zugehörige Achsen gespiegelt sind, ebenfalls gespiegelt - durch Vorzeicheninvertierung.

Die Verschleißwerte werden **nicht** mitgespiegelt. Sollen diese ebenfalls gespiegelt werden, muss das Settingdatum SD42910 $SC_MIRROR_TOOL_WEAR gesetzt sein.

SD42910 $SC_MIRROR_TOOL_WEAR

Settingdatum **ungleich** Null:

Es werden die Verschleißwerte der Werkzeuglängenkomponenten, deren zugehörige Achsen gespiegelt sind, ebenfalls gespiegelt - durch Vorzeicheninvertierung.

7.3.2 Vorzeichenbewertung Verschleiß

Funktion

Mit gesetzten Settingdaten SD42920 $SC_WEAR_SIGN_CUTPOS und SD42930 $SC_WEAR_SIGN ungleich Null können Sie die Vorzeichenbewertung der Verschleißkomponenten invertieren.

SD42920 $SC_WEAR_SIGN_CUTPOS

Settingdatum **ungleich** Null:

Bei Werkzeugen mit relevanter Schneidenlage (Dreh- und Schleifwerkzeuge, Werkzeugtypen 400) hängt die Vorzeichenbewertung der Verschleißkomponenten in der Bearbeitungsebene von der Schneidenlage ab. Bei Werkzeugtypen ohne relevanter Schneidenlage ist dieses Settingdatum bedeutungslos.

In folgender Tabelle sind die Maße durch ein X gekennzeichnet, deren Vorzeichen über das SD42920 (ungleich 0) invertiert wird:

Schneidenlage	Länge 1	Länge 2
1		
2		X
3	X	X
4	X	
5		
6		
7		X
8	X	
9		

Hinweis

Die Vorzeichenbewertung durch SD42920 und SD42910 sind voneinander unabhängig. Wenn z. B. das Vorzeichen einer Maßangabe durch beide Settingdaten geändert wird, bleibt das resultierende Vorzeichen unverändert.

SD42930 $SC_WEAR_SIGN

Settingdatum **ungleich** Null:

Das Vorzeichen aller Verschleißmaße wird invertiert. Es wirkt sowohl auf die Werkzeuglänge als auch auf die übrigen Größen wie Werkzeugradius, Verrundungsradius usw.

Wird ein positives Verschleißmaß eingegeben, wird somit das Werkzeug "kürzer" und "dünner", siehe Kapitel "Werkzeugkorrektur, Sonderbehandlung", Wirksamwerden der veränderten Settingdaten".

7.3.3 Koordinatensystem der aktiven Bearbeitung (TOWSTD, TOWMCS, TOWWCS, TOWBCS, TOWTCS, TOWKCS)

Funktion

Abhängig von der Kinematik der Maschine oder vom Vorhandensein eines orientierbaren Werkzeugträgers werden die in einem dieser Koordinatensysteme gemessenen Verschleißwerte in ein geeignetes Koordinatensystem überführt bzw. transformiert.

Koordinatensysteme der aktiven Bearbeitung

Aus den folgenden Koordinatensystemen können Offsets der Werkzeuglänge hervorgehen, welche die Werkzeuglängenkomponente Verschleiß über den entsprechenden G-Code der Gruppe 56 in ein aktives Werkzeug eingerechnet werden.

- Maschinenkoordinatensystem (MKS)
- Basiskoordinatensystem (BKS)
- Werkstückkoordinatensystem (WKS)
- Werkzeugkoordinatensystem (TCS)
- Werkzeugkoordinatensystem der kinematischen Transformation (KCS)

Syntax

```
TOWSTD
TOWMCS
TOWWCS
TOWBCS
TOWTCS
TOWKCS
```

Bedeutung

`TOWSTD`	Grundstellungswert für Korrekturen in der Werkzeuglänge Verschleißwert
`TOWMCS`	Korrekturen in der Werkzeuglänge im MKS
`TOWWCS`	Korrekturen in der Werkzeuglänge im WKS
`TOWBCS`	Korrekturen in der Werkzeuglänge im BKS
`TOWTCS`	Korrekturen der Werkzeuglänge am Werkzeugträgerbezugspunkt (orientierbarer Werkzeugträger)
`TOWKCS`	Korrekturen der Werkzeuglänge des Werkzeugkopfes (kinematischer Transformation)

Weitere Informationen

Unterscheidungsmerkmale

In der folgenden Tabelle sind die wichtigsten Unterscheidungsmerkmale dargestellt:

G-Code	Verschleißwert	Aktiver orientierbarer Werkzeugträger
TOWSTD	Grundstellungswert, Werkzeuglänge	Verschleißwerte unterliegen der Drehung.
TOWMCS	Verschleißwert Im MKS. TOWMCS ist mit TOWSTD identisch, wenn kein orientierbarer WZ-Träger aktiv ist.	Es dreht nur der Vektor der resultierenden Werkzeuglänge ohne Berücksichtigung des Verschleißes.
TOWWCS	Der Verschleißwert wird Im WKS auf das MKS umgerechnet.	Der Werkzeugvektor wird ohne Berücksichtigung des Verschleißes wie bei TOWMCS berechnet.
TOWBCS	Der Verschleißwert wird Im BKS auf das MKS umgerechnet.	Der Werkzeugvektor wird ohne Berücksichtigung des Verschleißes wie bei TOWMCS berechnet.
TOWTCS	Der Verschleißwert wird Im Werkzeugkoordinatensystem auf das MKS umgerechnet.	Der Werkzeugvektor wird ohne Berücksichtigung des Verschleißes wie bei TOWMCS berechnet.

TOWWCS , TOWBCS, TOWTCS: Der Verschleißvektor wird zum Werkzeugvektor addiert.

Lineare Transformation

Die Werkzeuglänge ist im MKS nur sinnvoll definierbar, wenn das MKS aus dem BKS durch eine lineare Transformation hervorgeht.

Nicht lineare Transformation

Ist z. B. mit TRANSMIT eine nicht lineare Transformation aktiv, dann wird bei Angabe des MKS als gewünschtes Koordinatensystem automatisch das BKS verwendet.

Keine kinematische Transformation und kein orientierbarer Werkzeugträger

Ist weder eine kinematische Transformation noch ein orientierbarer Werkzeugträger aktiv, dann fallen bis auf das WKS alle weiteren vier Koordinatensysteme zusammen. Damit unterscheidet sich nur das WKS von den übrigen. Da ausschließlich Werkzeuglängen zu bewerten sind, haben Translationen zwischen den Koordinatensystemen keine Bedeutung.

Literatur:

Weitere Informationen zur Werkzeugkorrektur siehe:
Funktionshandbuch Grundfunktionen; Werkzeugkorrektur (W1)

Einrechnung der Verschleißwerte

Das Settingdatum **SD42935 $SC_WEAR_TRANSFORM** legt fest, welche der drei Verschleißkomponenten:

- Verschleiß
- Summenkorrekturen fein
- Summenkorrekturen grob

einer Drehung durch eine Adaptertransformation oder einen orientierbaren Werkzeugträger unterworfen werden soll, wenn einer der folgenden G-Codes aktiv ist:

- `TOWSTD` Grundstellung

  für Korrekturen in der Werkzeuglänge
- `TOWMCS` Verschleißwerte

  im Maschinenkoordinatesystem (MKS)
- `TOWWCS` Verschleißwerte

  im Werkstückkoordinatesystem (WKS)
- `TOWBCS` Verschleißwerte (BKS)

  im Basiskoordinatesystem
- `TOWTCS` Verschleißwerte im Werkzeugkoordinatesystem an der Werkzeughalteraufnahme (T Werkzeugträgerbezug)
- `TOWKCS` Verschleißwerte im Koordinatesystem des Werkzeugkopfes bei kinetischer Transformation

---

Hinweis

Die Bewertung der einzelnen Verschleißkomponenten (Zuordnung zu den Geometrieachsen, Vorzeichenbewertung) wird beeinflusst durch:

- die aktive Ebene
- die Adaptertransformation
- folgende Settingdaten:
  - SD42910 $SC_MIRROW_TOOL_WEAR
  - SD42920 $SC_WEAR_SIGN_CUTPOS
  - SD42930 $SC_WEAR_SIGN
  - SD42940 $SC_TOOL_LENGTH_CONST
  - SD42950 $SC_TOOL_LENGTH_TYPE

7.3.4 Werkzeuglänge und Ebenenwechsel

Funktion

Mit gesetzten Settingdaten SD42940 $SC_TOOL_LENGTH_CONST ungleich Null können Sie Werkzeuglängenkomponenten wie Länge, Verschleiß und Basismaß zu den Geometrieachsen für Dreh- und Schleifwerkzeuge bei einen Ebenenwechsel zuordnen.

SD42940 $SC_TOOL_LENGTH_CONST

Settingdatum **ungleich** Null:

Die Zuordnung der Werkzeuglängenkomponenten (Länge, Verschleiß und Basismaß) zu den Geometrieachsen beim Wechsel der Bearbeitungsebene (`G17` - `G19`) wird nicht verändert.

Folgende Tabelle zeigt die Zuordnung der Werkzeuglängenkomponenten zu den Geometrieachsen für Dreh- und Schleifwerkzeuge (WZ-Typ 400 bis 599):

Inhalt	Länge 1	Länge 2	Länge 3
17	Y	X	Z
*)	X	Z	Y
19	Z	Y	X
-17	X	Y	Z
-18	Z	X	Y
-19	Y	Z	X

*) Jeder Wert ungleich 0, der nicht gleich einem der sechs aufgeführten Werte ist, wird wie der Wert 18 bewertet.

Folgende Tabelle zeigt die Zuordnung der Werkzeuglängenkomponenten zu den Geometrieachsen für alle anderen Werkzeuge (WZ-Typ < 400 bzw. > 599):

Bearbeitungsebene	Länge 1	Länge 2	Länge 3
*)	Z	Y	X
18	Y	X	Z
19	X	Z	Y
-17	Z	X	Y
-18	Y	Z	X
-19	X	Y	Z

*) Jeder Wert ungleich 0, der nicht gleich einem der sechs aufgeführten Werte ist, wird wie der Wert 17 bewertet.

Hinweis

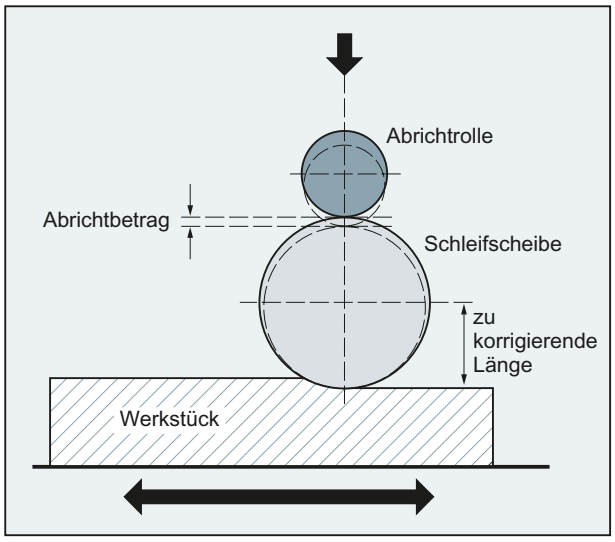
Bei der Darstellung in den Tabellen wird davon ausgegangen, dass die Geometrieachsen bis 3 mit X, Y, Z bezeichnet werden. Für die Zuordnung einer Korrektur zu einer Achse ist nicht der Achsbezeichner, sondern die Achsreihenfolge maßgebend.

7.4 Online-Werkzeugkorrektur (PUTFTOCF, FCTDEF, PUTFTOC, FTOCON, FTOCOF)

Funktion

Mit aktiver Funktion "Online-Werkzeugkorrektur" wird bei Schleifwerkzeugen eine Werkzeuglängenkorrektur, die sich aus der Bearbeitung ergibt, sofort eingerechnet.

Ein Anwendungsbeispiel ist das CD-Abrichten, bei dem die Schleifscheibe parallel zur Bearbeitung abgerichtet wird:



Die Werkzeuglängenkorrektur kann aus dem Bearbeitungskanal oder einem parallelen Kanal (Abrichterkanal) verändert werden.

Zum Schreiben der Online-WZK werden je nach gewünschtem Zeitpunkt des Abrichtvorgangs unterschiedliche Funktionen verwendet:

- Schreiben kontinuierlich satzweise (`PUTFTOCF`)

  Mit `PUTFTOCF` erfolgt der Abrichtvorgang zeitgleich mit der Bearbeitung.

  Die Werkzeugkorrektur wird im Bearbeitungskanal kontinuierlich nach einer Polynom-Funktion 1., 2. oder 3. Grades geändert, die vorher mit `FCTDEF` definiert werden muss.

  `PUTFTOCF` wirkt immer satzweise, d. h. im anschließenden Verfahrsatz.

- Schreiben kontinuierlich modal: `ID=1 DO FTOC` (siehe "Online-Werkzeugkorrektur (FTOC) (Seite 593)")

- Schreiben diskret (`PUTFTOC`)

  Mit `PUTFTOC` erfolgt der Abrichtvorgang nicht zeitgleich mit der Bearbeitung aus einem parallelen Kanal. Der mit `PUTFTOC` angegebene Korrekturwert wird im Zielkanal sofort wirksam.

Hinweis

Die Online-WZK kann nur bei Schleifwerkzeugen angewendet werden.

Syntax

Online-WZK im Zielkanal ein-/ausschalten:

```
FTOCON
...
FTOCOF
```

Online-WZK schreiben:

- Kontinuierlich satzweise:

```
FCTDEF(<Funktion>,<LLimit>,<ULimit>,<a0>,<a1>,<a2>,<a3>)
PUTFTOCF(<Funktion>,<Bezugswert>,<WZ-Parameter>,<Kanal>,<Spindel>)
...
```

- diskret:

```
PUTFTOC(<Korrekturwert>,<WZ-Parameter>,<Kanal>,<Spindel>)
...
```

Bedeutung

`FTOCON`: Online-WZK einschalten

`FTOCON` muss in dem Kanal programmiert werden, in dem die Online-WZK wirksam werden soll.

`FTOCOF`: Online-WZK abbrechen

Mit `FTOCOF` wird die Korrektur nicht weiter herausgefahren, in den schneidenspezifischen Korrekturdaten ist jedoch der komplette mit `PUTFTOC/PUTFTOCF` geschriebene Betrag korrigiert.

Hinweis:
Zum endgültigen Deaktivieren der Online-WZK muss nach `FTOCOF` noch eine An-/Abwahl des Werkzeugs (`T...`) erfolgen.

`FCTDEF`: Mit `FCTDEF` wird die Polynom-Funktion für `PUTFTOCF` definiert.

Parameter:

`<Funktion>`: Nummer der Polynom-Funktion
Typ: INT

`<LLimit>`: Unterer Grenzwert
Typ: REAL

`<ULimit>`: Oberer Grenzwert
Typ: REAL

`<a0> ... <a3>`: Koeffizienten der Polynom-Funktion
Typ: REAL

`PUTFTOCF`: Funktion "Online-WZK schreiben kontinuierlich satzweise" aufrufen

Parameter:

`<Funktion>`: Nummer der Polynom-Funktion

Typ: INT

Hinweis:
Muss mit der Angabe bei `FCTDEF` übereinstimmen.

`<Bezugswert>`: Variabler Bezugswert, von dem die Korrektur abgeleitet werden soll (z. B. sich verändernder Istwert).

Typ: VAR REAL

`<WZ-Parameter>`: Nummer des Verschleißparameters (Länge 1, 2 oder 3), in dem der Korrekturwert addiert werden soll.

Typ: INT

`<Kanal>`: Nummer des Kanals, in dem die Online-WZK wirksam werden soll.

Typ: INT

Hinweis:
Eine Angabe ist nur erforderlich, wenn die Korrektur nicht im aktiven Kanal wirksam werden soll.

`<Spindel>`: Nummer der Spindel, für die die Online-WZK wirksam werden soll.

Typ: INT

Hinweis:
Eine Angabe ist nur erforderlich, wenn statt dem aktiven, im Einsatz befindlichen Werkzeug eine nicht aktive Schleifscheibe korrigiert werden soll.

`PUTFTOC`: Funktion "Online-WZK schreiben diskret" aufrufen

Parameter:

`<Korrekturwert>`: Korrekturwert, der im Verschleißparameter addiert werden soll.

Typ: REAL

`<WZ-Parameter>`: siehe `PUTFTOCF`

`<Kanal>`: Nummer des Kanals, in dem die Online-WZK wirksam werden soll.

Typ: INT

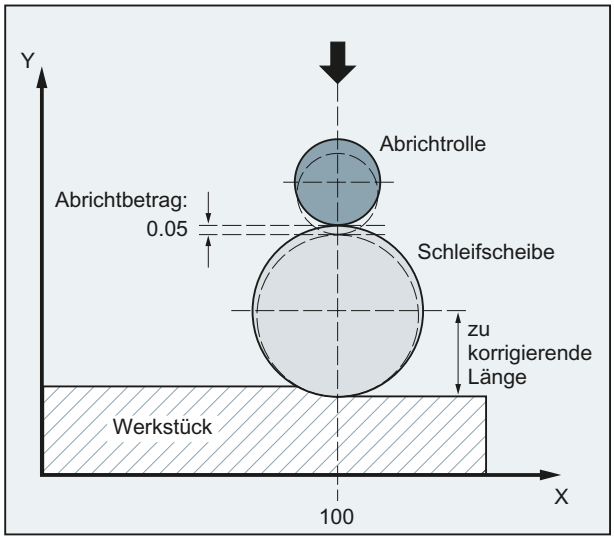
`<Spindel>`: siehe `PUTFTOCF`

Beispiel

Flachschleifmaschine mit:

- Y: Zustellachse für die Schleifscheibe
- V: Zustellachse für die Abrichtrolle
- Bearbeitungskanal: Kanal 1 mit den Achsen X, Z, Y
- Abrichtkanal: Kanal 2 mit Achse V

Nach Beginn der Schleifbewegung soll bei X100 die Schleifscheibe um den Betrag 0,05 abgerichtet werden. Der Abrichtbetrag soll mit "Online-WZK schreiben kontinuierlich" beim Schleifwerkzeug wirksam werden.



Bearbeitungsprogramm in Kanal 1:

Programmcode	Kommentar
…	
N110 G1 G18 F10 G90	; Grundstellung.
N120 T1 D1	; Aktuelles Werkzeug anwählen.
N130 S100 M3 X100	; Spindel ein, Fahren auf Ausgangsposition.
N140 INIT(2,"ABRICHT","S")	; Anwahl des Abrichtprogramms in Kanal 2.
N150 START(2)	; Starten des Abrichtprogramms in Kanal 2.
N160 X200	; Fahren auf Zielposition.
N170 FTOCON	; Online-Korrektur einschalten.
N… G1 X100	; Weitere Bearbeitung.
N… M30	

Abrichtprogramm in Kanal 2:

```
Programmcode                                  Kommentar
…
N40 FCTDEF(1,-1000,1000,-$AA_IW[V],1)    ; Funktion definieren: Gerade mit
                                           Steigung=1.
N50 PUTFTOCF(1,$AA_IW[V],3,1)            ; Online-WZK schreiben kontinuierlich:
                                           abgeleitet von der Bewegung der V-Achse
                                           wird die Länge 3 der aktuellen
                                           Schleifscheibe in Kanal 1 korrigiert.
N60 V-0.05 G1 F0.01 G91                  ; Zustellbewegung zum Abrichten, nur in
                                           diesem Satz ist PUTFTOCF wirksam.
…
N… M30
```

Weitere Informationen

Allgemeines zur Online-WZK

Beim kontinuierlichen Schreiben (je IPO-Takt) wird nach dem Einschalten der Auswertefunktion jede Änderung additiv im Verschleißspeicher verrechnet (um Sollwertsprünge zu vermeiden).

In jedem Fall gilt: Die Online-WZK kann in jedem Kanal für jede Spindel und die Länge 1, 2 **oder** 3 der Verschleißparameter wirken.

Die Zuordnung der Längen zu den Geometrieachsen erfolgt anhand der aktuellen Arbeitsebene.

Die Zuordnung der Spindel zum Werkzeug erfolgt durch die Werkzeugdaten bei GWPSON bzw. TMON, sofern es sich nicht um die aktive Schleifscheibe handelt.

Korrigiert wird immer der Verschleißparameter für die aktuelle Scheibenseite bzw. die linke Scheibenseite bei nicht aktiven Werkzeugen.

Hinweis

Bei identischer Korrektur für mehrere Scheibenseiten ist über Verkettungsvorschrift dafür zu sorgen, dass die Werte automatisch für die zweite Scheibenseite übernommen werden.

Werden für einen Bearbeitungskanal Online-Korrekturen vorgegeben, so dürfen die Verschleißwerte für das aktuelle Werkzeug in diesem Kanal nicht aus dem Bearbeitungsprogramm oder über Bedienung geändert werden.

Die Online-WZK wird auch für die konstante Scheibenumfangsgeschwindigkeit (SUG) sowie Werkzeugüberwachung (TMON) berücksichtigt.

7.5 Aktivierung 3D-Werkzeugkorrekturen (CUT3DC..., CUT3DF...)

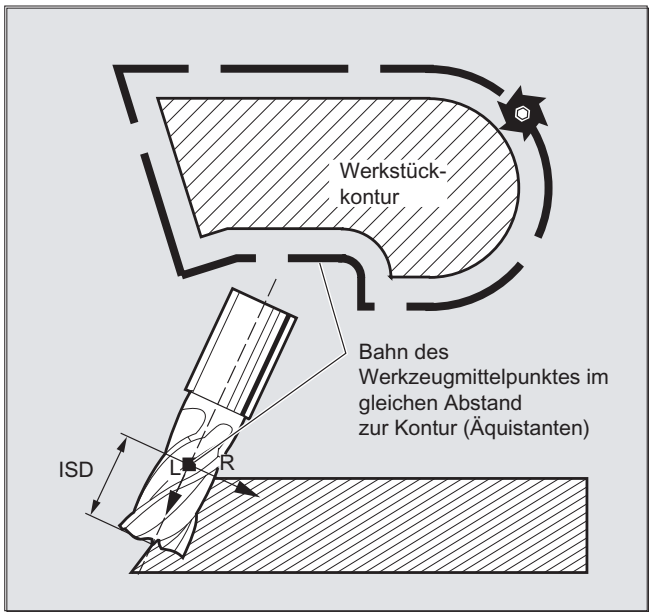
7.5.1 Aktivierung von 3D-Werkzeugkorrekturen (CUT3DC, CUT3DF, CUT3DFS, CUT3DFF, ISD)

Funktion

Bei der Werkzeugradiuskorrektur für zylindrische Werkzeuge wird die veränderliche Werkzeugorientierung berücksichtigt.

Für die Anwahl der 3D-Werkzeugradiuskorrektur gelten die gleichen Programmbefehle wie bei der 2D-Werkzeugradiuskorrektur. Mit G41/G42 wird die Korrektur links/rechts in Bewegungsrichtung angegeben. Das Anfahrverhalten ist immer NORM. Die 3D-Werkzeugradiuskorrektur wirkt nur bei angewählter 5-Achs-Transformation.

Die 3D-Werkzeugradiuskorrektur wird auch als 5D-Korrektur bezeichnet, da in diesem Fall 5 Freiheitsgrade für die Lage des Werkzeugs im Raum zur Verfügung stehen.



Unterschied zwischen 2 1/2D- und 3D-Werkzeugradiuskorrektur

Bei der 3D-Werkzeugradiuskorrektur ist die Werkzeug-Orientierung veränderlich. Bei der 2 1/2D-Werkzeugradiuskorrektur wird nur mit einem Werkzeug mit konstanter Orientierung gerechnet.

Syntax

```
CUT3DC
CUT3DFS
CUT3DFF
CUT3DF
ISD=<Wert>
```

Bedeutung

`CUT3DC`	Aktivierung der 3D-Radiuskorrektur für das Umfangsfräsen
`CUT3DFS`	D-Werkzeugkorrektur für das Stirnfräsen mit konstanter Orientierung. Die Werkzeugorientierung ist durch `G17` - `G19` festgelegt und wird durch Frames nicht beeinflusst.
`CUT3DFF`	D-Werkzeugkorrektur für das Stirnfräsen mit konstanter Orientierung. Die Werkzeugorientierung ist die durch `G17` - `G19` festgelegte und gegebenenfalls durch einen Frame gedrehte Richtung.
`CUT3DF`	D-Werkzeugkorrektur für das Stirnfräsen mit Orientierungsänderung (nur bei aktiver 5-Achs-Transformation).
`G40 X... Y... Z...`	Zum Ausschalten: Linearsatz `G0`/`G1` mit Geometrieachsen
`ISD`	Eintauchtiefe

Hinweis

Die Befehle sind modal wirksam und stehen in der gleichen Gruppe wie`CUT2D` und `CUT2DF`. Die Abwahl findet erst mit der nächsten Bewegung in der aktuellen Ebene statt. Dies gilt immer für `G40` und ist unabhängig vom `CUT`-Befehl.

Zwischensätze bei aktiver 3D-Werkzeugradiuskorrektur sind erlaubt. Es gelten die Festlegungen der 2 1/2D-Werkzeugradiuskorrektur.

Randbedingungen

- **G450/G451 und DISC**

  An Außenecken wird immer ein Kreissatz eingefügt. `G450/G451` haben keine Bedeutung. Der Befehl `DISC` wird nicht ausgewertet.

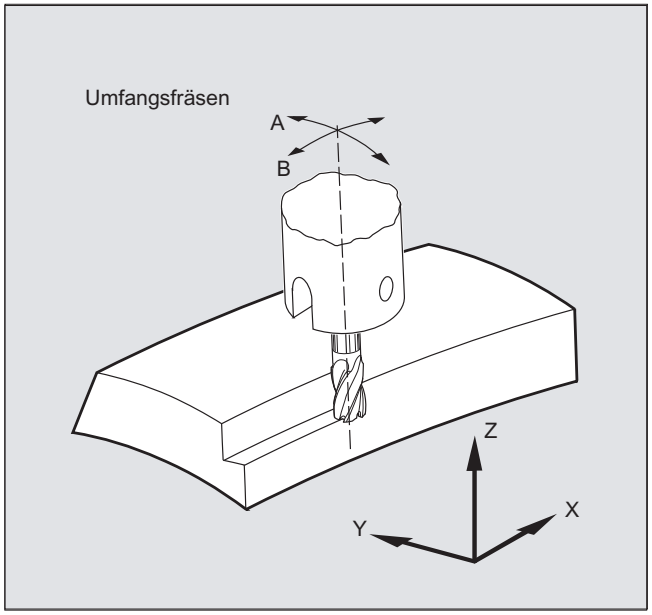
Beispiel

```
Programmcode                    Kommentar
N10 A0 B0 X0 Y0 Z0 F5000
N20 T1 D1                       ; Werkzeugaufruf, Werkzeugkorrekturwerte aufrufen.
N30 TRAORI(1)                   ; Transformationsanwahl
N40 CUT3DC                      ; 3D-Werkzeugradiuskorrektur-Anwahl
N50 G42 X10 Y10                 ; Werkzeugradiuskorrektur-Anwahl
N60 X60
N70 …
```

7.5.2 3D-Werkzeugkorrektur: Umfangfräsen, Stirnfräsen

Umfangsfräsen

Die hier benutzte Variante des Umfangsfräsens ist durch die Vorgabe einer Bahn (Leitlinie) und der zugehörigen Orientierung realisiert. Bei dieser Art der Bearbeitung ist auf der Bahn die Werkzeugform ohne Bedeutung. Entscheidend ist allein der Radius am Werkzeugeingriffspunkt.

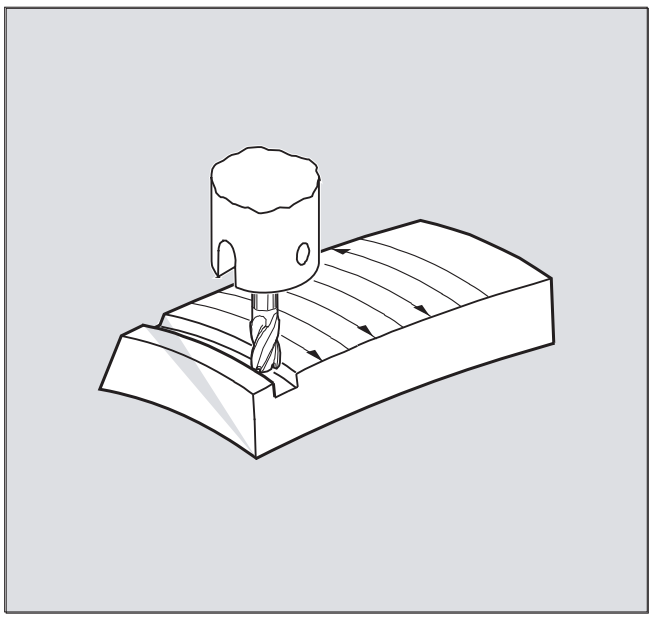


Hinweis

Die Funktion 3D-WRK beschränkt sich auf zylindrische Werkzeuge.

Stirnfräsen

Für diese Art des 3D-Fräsens benötigen Sie die zeilenweise Beschreibung der 3D-Bahnen auf der Werkstückoberfläche. Die Berechnungen werden unter Berücksichtigung der Werkzeugform und Werkzeugabmessungen - üblicherweise im CAM durchgeführt. Der Postprozessor schreibt in das Teileprogramm - neben den NC-Sätzen - die Werkzeugorientierungen (bei aktiver 5-Achstransformation) und den G-Code für die gewünschte 3D-Werkzeugkorrektur. Hierdurch hat der Maschinenbediener die Möglichkeit – abweichend von dem für die Berechnung der NC-Bahnen verwendeten Werkzeug -, geringfügig kleinere Werkzeuge einzusetzen.



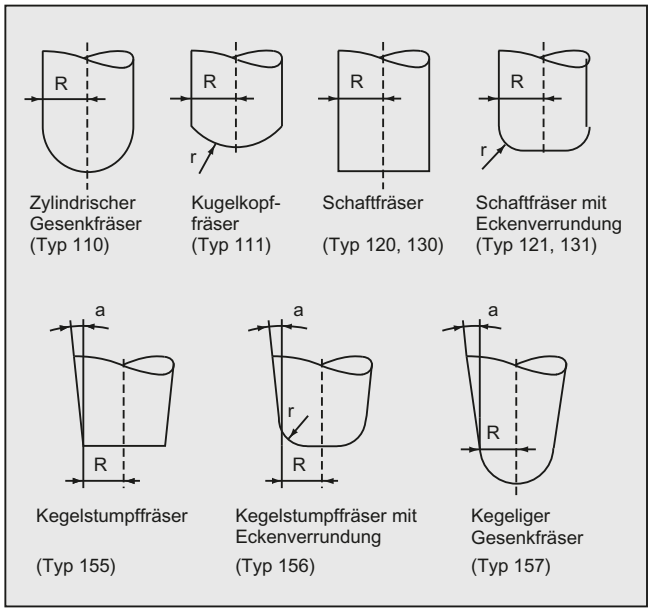
Beispiel:

NC-Sätze wurden mit Fräser 10 mm berechnet. Hier könnte auch mit Fräserdurchmesser 9,9 mm gefertigt werden, wobei dann mit verändertem Rauheitsprofil zu rechnen ist.

7.5.3 3D-Werkzeugkorrektur: Werkzeugformen und Werkzeugdaten für Stirnfräsen

Fräserformen, Werkzeugdaten

Im Folgenden sind die für Stirnfräsen möglichen Werkzeugformen und Grenzwerte der Werkzeugdaten zusammengestellt. Die Form des Werkzeugschafts wird nicht berücksichtigt. Die Werkzeugtypen 120 und 156 sind in ihrer Wirkung identisch.



Wird im NC-Programm eine andere als in der Abbildung gezeigte Typ-Nummer angegeben, verwendet das System automatisch den Werkzeugtyp 110 (Zylindrischer Gesenkfräser). Bei Verletzung der Grenzwerte für die Werkzeugdaten wird ein Alarm ausgegeben.

Fräsertyp	Typ-Nr.	R	r	a
Zylindrischer Gesenkfräser	110	> 0	-	-
Kugelkopffräser	111	> 0	> R	-
Schaftfräser, Winkelkopffräser	120, 130	> 0	-	-
Schaftfräser, Winkelkopffräser mit Eckenverrundung	121, 131	> r	> 0	-
Kugelstumpffräser	155	> 0	-	> 0
Kugelstumpffräser mit Eckenverrundung	156	> 0	> 0	> 0
Kegeliger Gesenkfräser	157	> 0	-	> 0

R = Schaftradius (Werkzeugradius)

r = Eckenradius

a = Winkel zwischen Werkzeuglängsachse und oberem Ende der Torusfläche

\- = wird nicht ausgewertet

Werkzeugdaten	Werkzeugparameter	
Werkzeugmaße	Geometrie	Verschleiß
R	$TC_DP6	$TC_DP15
r	$TC_DP7	$TC_DP16
a	$TC_DP11	$TC_DP20

Werkzeuglängenkorrektur

Als Bezugspunkt für die Längenkorrektur gilt die Werkzeugspitze (Schnittpunkt Längsachse/Oberfläche).

3D-Werkzeugkorrektur, Werkzeugwechsel

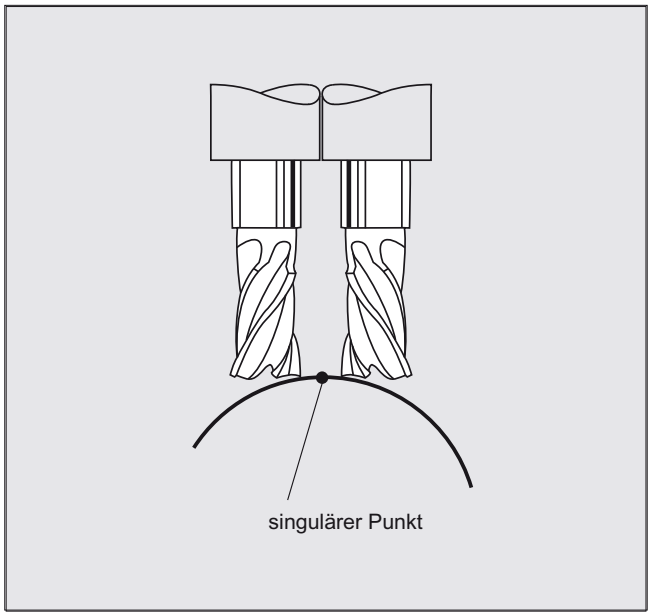
Ein neues Werkzeug mit veränderten Abmessungen (R, r, a) oder anderer Form darf nur mit Programmierung von G41 bzw. G42 angegeben werden (Übergang G40 nach G41 bzw. G42, erneute Programmierung von G41 bzw. G42). Alle anderen Werkzeugdaten, z. B. Werkzeuglängen, bleiben von dieser Regel unberücksichtigt, so dass solche Werkzeuge auch ohne erneutes G41 bzw. G42 eingewechselt werden können.

7.5.4 3D-Werkzeugkorrektur: Korrektur auf der Bahn, Bahnkrümmung, Eintauchtiefe (CUT3DC, ISD)

Funktion

Korrektur auf der Bahn

Beim Stirnfräsen muss der Fall betrachtet werden, dass der Berührpunkt auf der Werkzeugoberfläche springt. Wie in diesem Beispiel bei der Bearbeitung einer konvexen Fläche mit senkrecht stehendem Werkzeug. Die im Bild gezeigte Anwendung kann als Grenzfall betrachtet werden.



Dieser Grenzfall wird von der Steuerung überwacht, indem auf Basis der Winkelanstellungen zwischen Werkzeug und Flächennormalenvektoren sprunghafte Änderungen des Bearbeitungspunktes erkannt werden. An diesen Stellen fügt die Steuerung Linearsätze ein, so dass die Bewegung ausgeführt werden kann.

Für die Berechnung der Linearsätze sind in Maschinendaten für den Seitwärtswinkel zulässige Winkelbereiche hinterlegt. Falls die in Maschinendaten festgelegten Grenzwerte für zugelassene Winkelbereiche überschritten werden, meldet das System Alarm.

Bahnkrümmung

Die Bahnkrümmung wird nicht überwacht. Auch hier empfiehlt es sich, nur solche Werkzeuge zu verwenden, mit denen ohne Konturverletzung gearbeitet werden kann.

Eintauchtiefe (ISD)

Die Eintauchtiefe ISD wird nur bei aktiver 3D-Werkzeugradiuskorrektur ausgewertet.

Mit dem Programmbefehl `ISD` (Insertion Depth) wird die Eintauchtiefe des Werkzeugs beim Umfangsfräsen programmiert. Damit ist es möglich, die Lage des Bearbeitungspunkts auf der Mantelfläche des Werkzeugs zu verändern.

Syntax

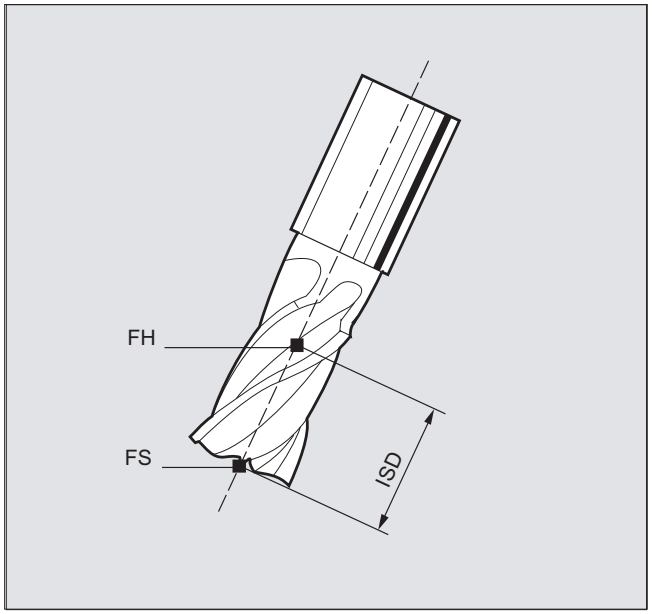
3D-Werkzeugkorrektur Umfangsfräsen
```
CUT3DC
ISD=<Wert>
```

Bedeutung

`CUT3DC`	3D-Werkzeugkorrektur für das Umfangsfräsen aktivieren, z. B. für Taschenfräsen mit schrägen Seitenwänden.
`ISD`	Mit dem Befehl `ISD` wird der Abstand (`<Wert>`) zwischen Fräserspitze (FS) und dem Fräserhilfspunkt (FH) angegeben.

Fräserhilfspunkt

Der Fräserhilfspunkt (FH) entsteht durch Projektion des programmierten Bearbeitungspunkts auf die Werkzeugachse.



Weitere Informationen

Taschenfräsen mit schrägen Seitenwänden für Umfangsfräsen mit CUT3DC

Bei dieser 3D-Werkzeugradiuskorrektur wird eine Abweichung des Fräserradius kompensiert, indem in Richtung der Flächennormalen der zu bearbeitenden Fläche zugestellt wird. Dabei bleibt die Ebene, in der die Stirnseite des Fräsers liegt unverändert, wenn die Eintauchtiefe `ISD` gleich geblieben ist. Ein Fräser mit z. B. kleinerem Radius gegenüber einem Normwerkzeug würde dann den Taschenboden, der auch die Begrenzungsfläche darstellt, nicht erreicht werden. Für eine automatische Zustellung des Werkzeugs muss der Steuerung diese Begrenzungsfläche bekannt sein, siehe Kapitel "3D-Umfangsfräsen mit Begrenzungsflächen".

Weitere Informationen zur Kollisionsüberwachung siehe:
Literatur:
Programmierhandbuch Grundlagen; Kapitel "Werkzeugkorrekturen".

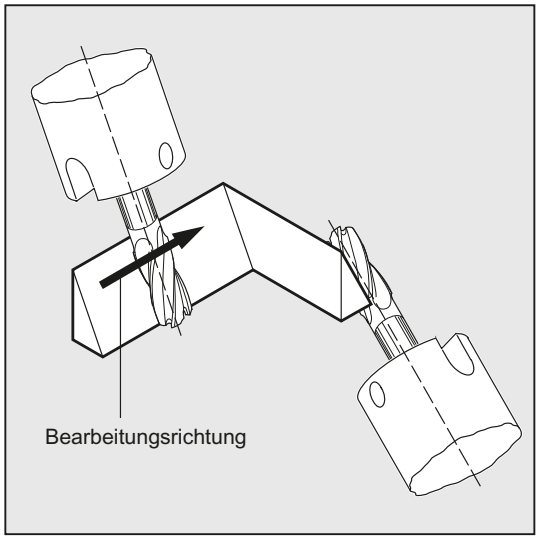
7.5.5 3D-Werkzeugkorrektur: Innenecken/Außenecken und Schnittpunktverfahren (G450/G451)

Funktion

Innenecken/Außenecken

Außen- und Innenecken werden getrennt behandelt. Die Bezeichnung Innen- oder Außenecke ist abhängig von der Werkzeugorientierung.

Bei Orientierungsänderungen an einer Ecke kann der Fall auftreten, dass sich der Eckentyp während der Bearbeitung ändert. Tritt dieser Fall auf, wird die Bearbeitung mit einer Fehlermeldung abgebrochen.



Syntax

```
G450
G451
```

Bedeutung

G450	Übergangskreis (Werkzeug umfährt Werkstückecken auf einer Kreisbahn)
G451	Schnittpunkt der Äquidistanten (Werkzeug schneidet in der Werkstückecke frei)

Weitere Informationen

Schnittpunktverfahren für 3D-Korrektur

Bei 3D-Umfangsfräsen wird jetzt an Außenecken der G-Code G450/G451 ausgewertet, d. h. es kann der Schnittpunkt der Offset-Kurven angefahren werden. Bis SW 4 wurde an Außenecken immer ein Kreis eingefügt. Das verfügbare Schnittpunktverfahren ist bei typischen CAD-erzeugten 3D-Programmen besonders vorteilhaft. Diese bestehen häufig aus kurzen Geradensätzen (zur Approximation glatter Kurven), bei denen die Übergänge zwischen benachbarten Sätzen nahezu tangential sind.

Bei Werkzeugradiuskorrektur an der Außenseite der Kontur wurden bislang grundsätzlich Kreise zum Umfahren der Außenecken eingefügt. Da diese Sätze bei nahezu tangentialen Übergängen sehr kurz werden, ergeben sich unerwünschte Geschwindigkeitseinbrüche.

In diesen Fällen werden analog zur 2 ½ D-Radiuskorrektur die beiden beteiligen Kurven verlängert, der Schnittpunkt der beiden verlängerten Kurven wird angefahren.

Der Schnittpunkt wird bestimmt, indem die Offsetkurven der beiden beteiligten Sätze verlängert werden und deren Schnittpunkt in der Ebene senkrecht zur Werkzeugorientierung an der Ecke bestimmt wird. Existiert kein derartiger Schnittpunkt, wird die Ecke wie bisher behandelt, d. h. es wird ein Kreis eingefügt.

Weitere Informationen zum Schnittpunktverfahren siehe:
Literatur:
Funktionshandbuch Sonderfunktionen; 3D-Werkzeugradiuskorrektur (W5)

7.5.6 3D-Werkzeugkorrektur: 3D-Umfangsfräsen mit Begrenzungsflächen

Anpassungen von 3D-Umfangsfräsen an Gegebenheiten von CAD-Programmen

Von CAD-Sytemen generierte NC-Programme approximieren in der Regel die Mittelpunktsbahn eines Normwerkzeuges mit einer großen Anzahl kurzer Linearsätze. Damit diese so erzeugten Sätze vieler Teilekonturen die ursprüngliche Originalkontur möglichst genau nachbilden, ist es notwendig im Teileprogramm gewisse Anpassungen vorzunehmen.

Wichtige Informationen, die für eine optimale Korrektur erforderlich wären, aber im Teilprogramm nicht mehr zur Verfügung stehen, müssen durch geeignete Maßnahmen ersetzt werden. Nachfolgend werden typische Methoden dargestellt, um kritische Übergänge entweder direkt im Teileprogramm oder bei Ermittlung der realen Kontur (z. B. durch Zustellung des Werkzeugs) auszugleichen.

Anwendungen

Zusätzlich zu den typischen Anwendungsfällen, bei denen anstelle des Normwerkzeugs ein reales Werkzeug die Mittelpunktsbahn beschreibt, werden auch zylindrische Werkzeuge mit 3D-Werkzeugkorrektur behandelt. Hierbei bezieht sich die programmierte Bahn auf die Kontur an der Bearbeitungsfläche. Die hierfür zutreffende Begrenzungsfläche ist werkzeugunabhängig. Es wird wie bei der herkömmlichen Werkzeugradiuskorrektur der Gesamtradius zur Berechnung des senkrechten Offsets zur Begrenzungsfläche herangezogen.

7.5.7 3D-Werkzeugkorrektur: Berücksichtigung einer Begrenzungsfläche (CUT3DCC, CUT3DCCD)

Funktion

3D-Umfangsfräsen mit realen Werkzeugen

Beim 3D-Umfangsfräsen mit kontinuierlicher oder konstanter Veränderung der Werkzeugorientierung wird häufig die Werkzeugmittelpunktsbahn für ein definiertes Normwerkzeug programmiert. Da in der Praxis oft nicht die passenden Normwerkzeuge zur Verfügung stehen, kann ein von einem Normwerkzeug nicht allzu stark abweichendes Werkzeug eingesetzt werden.

Mit `CUT3DCCD` wird für ein reales Differenzwerkzeug eine Begrenzungsfläche berücksichtigt, die das programmierte Normwerkzeug beschreiben würde. Das NC-Programm beschreibt die Mittelpunksbahn des Normwerkzeuges.

Mit `CUT3DCC` wird bei Verwendung von zylindrischen Werkzeugen eine Begrenzungsfläche berücksichtigt, die das programmierte Normwerkzeug erreicht hätte. Das NC-Programm beschreibt die Kontur auf der Bearbeitungsfläche.

Syntax

```
CUT3DCCD
CUT3DCC
```

Bedeutung

CUT3DCCD	Aktivierung der 3D-Werkzeugkorrektur für das Umfangsfräsen mit Begrenzungsflächen mit Differenzwerkzeug auf der Werkzeugsmittelpunktbahn: Zustellung zur Begrenzungsfläche.
CUT3DCC	Aktivierung der 3D-Werkzeugkorrektur für das Umfangsfräsen mit Begrenzungsflächen mit 3D-Radiuskorrektur: Kontur an der Bearbeitungsfläche

Hinweis

Werkzugradiuskorrektur mit G41, G42

Für die Werkzeugradiuskorrektur mit G41, G42 bei aktivem CUT3DCCD oder CUT3DCC muss die Option "Orientierungstransformation" vorhanden sein.

Normwerkzeuge mit Eckenverrundung

Die Eckenverrundung des Normwerkzeugs wird durch den Werkzeugparameter $TC_DP7 beschrieben. Aus den Werkzeugparameter $TC_DP16 ergibt sich die Abweichung der Eckenverrundung des realen Werkzeugs gegenüber dem Normwerkzeug.

Beispiel

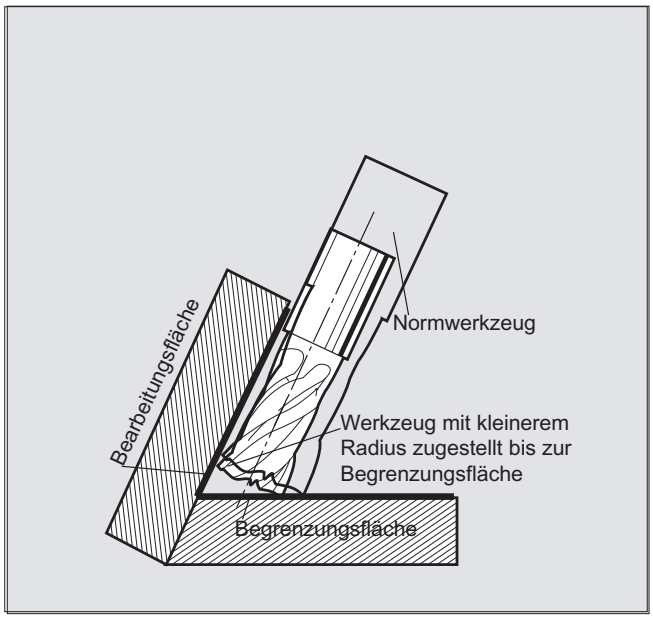
Werkzeugabmessungen für einen Torusfräser mit verringertem Radius gegenüber dem Normwerkzeug.

Werkzeugtyp	R = Schaftradius	r = Eckenradius
Normwerkzeug mit Eckenverrundung	R = $TC_DP6	r = $TC_DP7
Reales Werkzeug mit Eckenverrundung: Werkzeugtypen 121 und 131 Torusfräser (Schaftfräser)	R' = $TC_DP6 + $TC_DP15 + OFFN	r' = $TC_DP7 + $TC_DP16
In diesem Beispiel sind sowohl $TC_DP15 + OFFN als auch $TC_DP16 negativ. Der Werkzeugtyp ($TC_DP1) wird ausgewertet.		
Zugelassen sind nur Fräsertypen mit zylindrischen Schaft (Zylinder- oder Schaftfräser) sowie Torusfräser (Typ 121 und 131) und im Grenzfall der zylindrische Gesenkfräser (Typ 110).	Bei diesen zugelassenen Fräsertypen ist der Eckenradius r gleich dem Schaftradius R. Alle anderen zugelassenen Werkzeugtypen werden als Zylinderfräser interpretiert und ein eventuell angegebenes Maß für die Eckenverrundung wird nicht ausgewertet.	
Zugelassen sind alle Werkzeugtypen der Nummern 1 – 399 mit Ausnahme der Nummern 111 und 155 bis 157.		

Weitere Informationen

Werkzeugmittelpunktsbahn mit Zustellung bis zur Begrenzungsfläche CUT3DCCD

Wird ein Werkzeug verwendet, welches im Vergleich zum passenden Normwerkzeug einen kleineren Radius aufweist, dann wird ein in Längsrichtung zugestellter Fräser soweit weiter geführt, bis dieser den Taschenboden wieder berührt. Damit wird die Ecke, die von der Bearbeitungs- und der Begrenzungsfläche gebildet wird so weit ausgeräumt, wie dies das Werkzeug zulässt. Es handelt sich dabei um eine gemischte Bearbeitungsweise aus Umfangs- und Stirnfräsen. Analog zu einem Werkzeug mit verringertem Radius, wird beim Werkzeug mit vergrößerten Radius, in die entgegengesetzte Richtung entsprechend zugestellt.



Gegenüber allen anderen Werkzeugkorrekturen der G-Code Gruppe 22 hat ein für `CUT3DCCD` angegebener Werkzeugparameter `$TC_DP6` keine Bedeutung für den Werkzeugradius und beeinflusst die resultierende Korrektur nicht.

Der Korrekturoffset ergibt sich aus der Summe von:

- Verschleißwert des Werkzeugradius (Werkzeugparameter `$TC_DP15`)
- und einem zur Berechnung des senkrechten Offsets zur Begrenzungsfläche programmierten Werkzeugoffset `OFFN`.

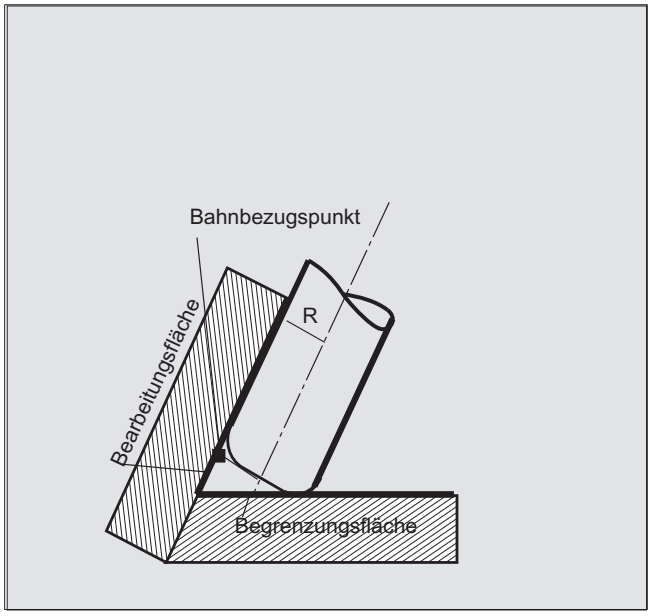
Ob die zu bearbeitende Fläche links oder rechts von der Bahn liegt, kann aus dem erzeugten Teileprogramm nicht entnommen werden. Es wird deshalb von einem positiven Radius und einem negativen Verschleißwert des Originalwerkzeuges ausgegangen. Ein negativer Verschleißwert beschreibt immer ein Werkzeug mit verringertem Durchmesser.

Verwendung von zylindrischen Werkzeugen

Bei der Verwendung von zylindrischen Werkzeugen ist eine Zustellung nur dann erforderlich, wenn die Bearbeitungsfläche und die Begrenzungsfläche einen spitzen Winkel (kleiner als 90 Grad) bilden. Werden Torusfräser (Zylinder mit Eckverrundung) verwendet, dann erfordert dies sowohl bei spitzen als auch bei stumpfen Winkeln eine Zustellung in Längsrichtung des Werkzeugs.

3D-Radiuskorrektur mit CUT3DCC, Kontur an der Bearbeitungsfläche

Ist `CUT3DCC` mit einem Torusfräser aktiv, so bezieht sich die programmierte Bahn auf einen fiktiven Zylinderfäser gleichen Durchmessers. Der hieraus resultierende Bahnbezugspunkt ist bei Verwendung eines Torusfräser im folgenden Bild dargestellt.



Es ist zulässig, dass der Winkel zwischen Bearbeitungs- und Begrenzungsfläche auch innerhalb eines Satzes von einem spitzen in einem stumpfen Winkel oder umgekehrt übergeht.

Gegenüber dem Normwerkzeug darf das verwendete reale Werkzeug sowohl größer als auch kleiner sein. Dabei darf der resultierende Eckenradius nicht negativ werden und das Vorzeichen des resultierenden Werkzeugradius muss erhalten bleiben.

Bei `CUT3DCC` bezieht sich das NC-Teileprogramm auf die Kontur an der Bearbeitungsfläche. Es wird hierbei wie bei der herkömmlichen Werkzeugradienkorrektur der Gesamtradius herangezogen, der sich zusammensetzt aus der Summe von:

- Werkzeugradius (Werkzeugparameter `$TC_DP6`)
- Verschleißwert (Werkzeugparameter `$TC_DP15`)
- und einem zur Berechnung des senkrechten Offsets zur Begrenzungsfläche programmierten Werkzeugoffset `OFFN`.

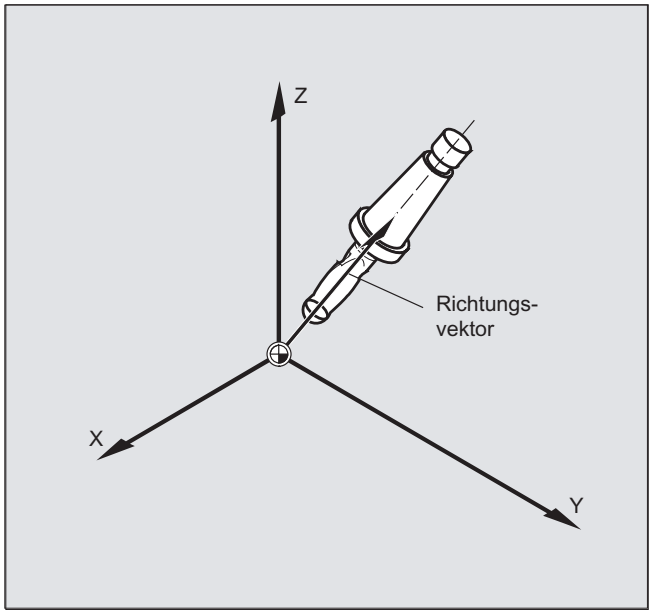
Die Lage der Begrenzungsfläche wird bestimmt aus der Differenz der beiden Werte:

- Abmessungen des Normwerkzeugs
- Werkzeugradius (WZ-Parameter `$TC_DP6`)

7.6 Werkzeugorientierung (ORIC, ORID, OSOF, OSC, OSS, OSSE, ORIS, OSD, OST)

Funktion

Unter Werkzeugorientierung versteht man die geometrische Ausrichtung des Werkzeugs im Raum. Bei einer 5-Achs-Bearbeitungsmaschine ist die Werkzeugorientierung über Programmbefehle einstellbar.



Mit OSD und OST aktivierte Überschleifbewegungen der Orientierung werden je nach Interpolationsart für die Werkzeugorientierung unterschiedlich gebildet.

Bei aktiver Vektorinterpolation wird der geglättete Orientierungsverlauf auch mittels Vektorinterpolation interpoliert. Dagegen wird bei aktiver Rundachsinterpolation die Orientierung direkt mittels Rundachsbewegungen geglättet.

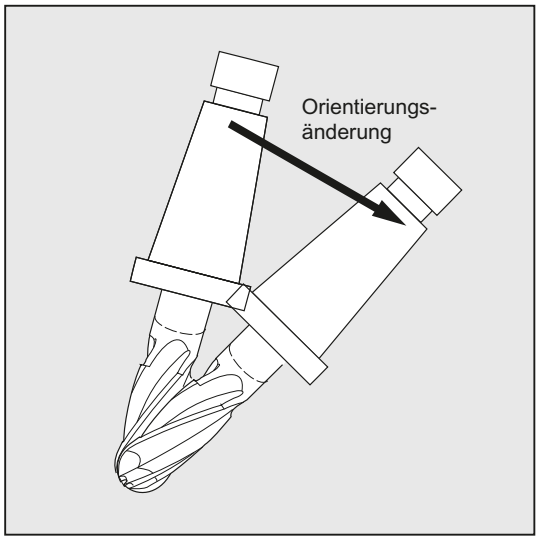
Programmierung

Programmierung der Orientierungsänderung:

Eine Orientierungsänderung des Werkzeugs kann programmiert werden durch:

- direkte Programmierung der Rundachsen A, B, C (Rundachsinterpolation)
- Euler- oder RPY-Winkel
- Richtungsvektor (Vektorinterpolation durch Angabe von A3 oder B3 oder C3)
- LEAD/TILT (Stirnfräsen)

Das Bezugskoordinatensystem ist entweder das Maschinenkoordinatensystem (ORIMKS) oder das aktuelle Werkstückkoordinatensystem (ORIWKS).



Programmierung der Werkzeugorientierung:

Befehl	Bedeutung
ORIC:	Orientierung und Bahnbewegung parallel
ORID:	Orientierung und Bahnbewegung nacheinander
OSOF:	keine Orientierungsglättung
OSC:	Orientierung konstant
OSS:	Orientierungsglättung nur am Satzanfang
OSSE:	Orientierungsglättung am Satzanfang und -ende
ORIS:	Geschwindigkeit der Orientierungsänderung bei eingeschalteter Orientierungsglättung in Grad pro mm (gilt für OSS und OSSE)
OSD:	Überschleifen der Orientierung durch Vorgabe der Überschleiflänge mit dem Settingdatum: SD42674 $SC_ORI_SMOOTH_DIST
OST:	Überschleifen der Orientierung durch Vorgabe der Winkeltoleranz in Grad bei Vektorinterpolation mit dem Settingdatum: SD42676 $SC_ORI_SMOOTH_TOL Bei Rundachsinterpolation wird die vorgegebene Toleranz als maximale Abweichung der Orientierungsachsen angenommen.

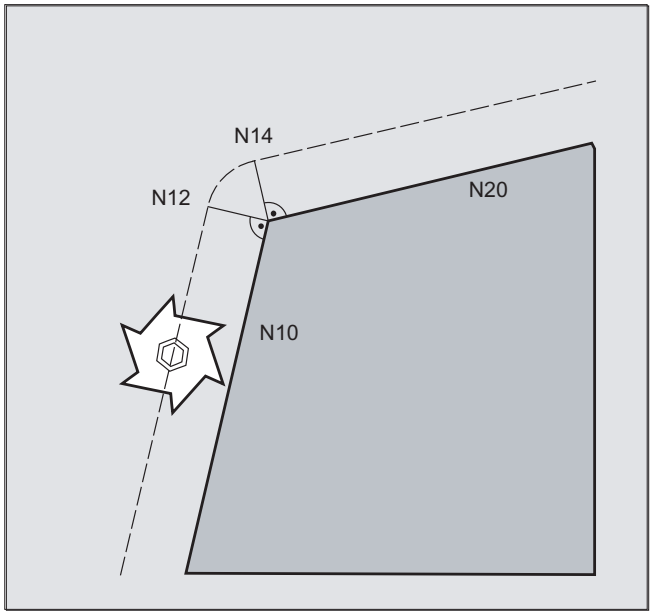
Hinweis

Alle Befehle zum Überschleifen der Werkzeugorientierung (OSOF, OSC, OSS, OSSE, OSD und OST) sind in der G-Funktionsgruppe 34 zusammengefasst. Sie sind modal wirksam, d. h. es kann immer nur einer dieser Befehle wirken.

Beispiele

Beispiel 1: ORIC

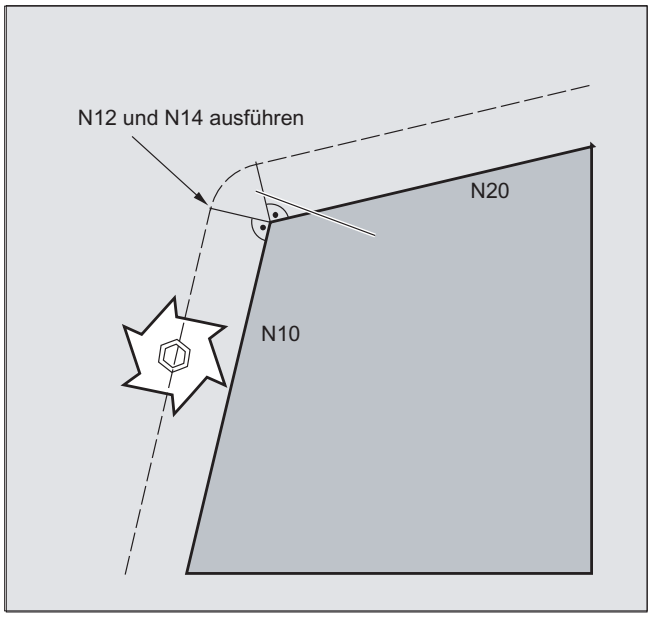
Sind zwischen den Verfahrsätzen `N10` und`N20`zwei oder mehrere Sätze mit Orientierungsänderungen (z. B. `A2=... B2=... C2=...`) programmiert und `ORIC` ist aktiv, so wird der eingefügte Kreissatz entsprechend dem Betrag der Winkeländerungen auf diese Zwischensätze aufgeteilt.



Programmcode	**Kommentar**
ORIC	
N8 A2=… B2=… C2=…	
N10 X… Y… Z…	
N12 C2=… B2=…	; Der Kreissatz, der an der Außenecke eingefügt wird,
N14 C2=… B2=…	verteilt sich auf N12 und N14, entsprechend der Orientierungsänderung. Kreisbewegung und Orientierungsänderung werden hierbei parallel ausgeführt.
N20 X =…Y=… Z=… G1 F200	

Beispiel 2: ORID

Ist `ORID` aktiv, so werden alle Sätze zwischen den beiden Verfahrsätzen am Ende des ersten Verfahrsatzes ausgeführt. Der Kreissatz mit konstanter Orientierung wird unmittelbar vor dem zweiten Verfahrsatz ausgeführt.



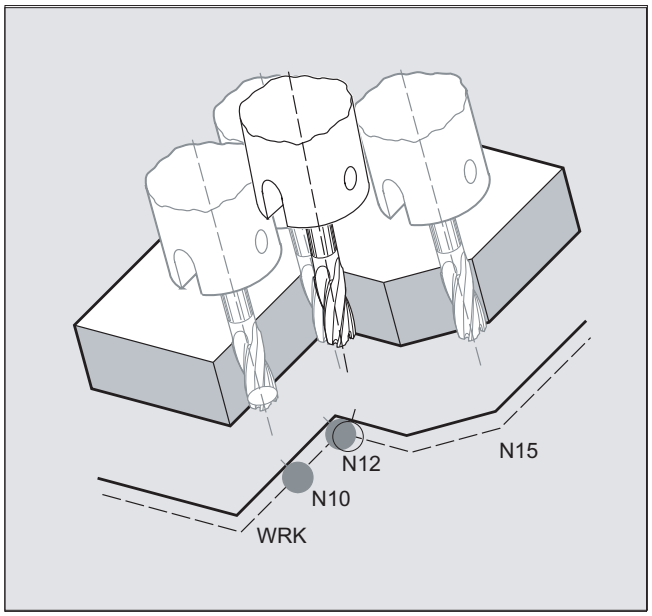
Programmcode	Kommentar
ORID	
N8 A2=… B2=… C2=…	
N10 X… Y… Z…	
N12 A2=… B2=… C2=…	; Der Satz N12 und N14 wird am Ende von N10 ausgeführt. Danach wird der Kreissatz mit der aktuellen Orientierung ausgefahren.
N14 M20	; Hilfsfunktionen etc.
N20 X… Y… Z…	

Hinweis

Für die Art der Orientierungsänderung an einer Außenecke ist der Programmbefehl maßgebend, welcher im ersten Verfahrsatz einer Außenecke aktiv ist.

Ohne Orientierungsänderung: Wird die Orientierung an der Satzgrenze nicht verändert, so ist der Werkzeugquerschnitt ein Kreis, der die beiden Konturen berührt.

Beispiel 3: Änderung der Orientierung an einer Innenecke



Programmcode

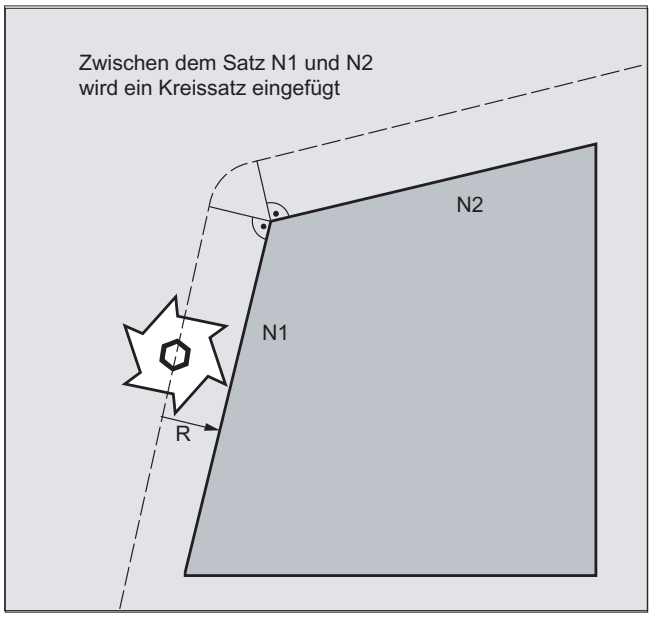
```
ORIC
N10 X …Y… Z… G1 F500
N12 X …Y… Z… A2=… B2=… C2=…
N15 X …Y… Z… A2=… B2=… C2=…
```

Weitere Informationen

Verhalten an Außenecken

An einer Außenecke wird immer ein Kreissatz mit dem Radius des Fräsers eingefügt.

Mit den Programmbefehlen `ORIC` bzw. `ORID` kann festgelegt werden, ob Orientierungsänderungen, die zwischen Satz `N1` und `N2` programmiert wurden, vor Beginn des eingefügten Kreissatzes oder gleichzeitig mit diesem ausgeführt werden.



Ist an Außenecken eine Orientierungsänderung notwendig, so kann diese wahlweise parallel zur Interpolation oder getrennt mit der Bahnbewegung erfolgen.

Bei ORID werden zunächst die eingefügten Sätze ohne Bahnbewegung ausgeführt. Der Kreissatz wird unmittelbar vor dem zweiten der beiden Verfahrsätze eingefügt, durch welche die Ecke gebildet wird.

Sind an einer Außenecke mehrere Orientierungssätze eingefügt und ORIC ist angewählt, so wird die Kreisbewegung entsprechend den Beträgen der Orientierungsänderungen der einzelnen eingefügten Sätze auf diese verteilt.

Überschleifen der Orientierung mit OSD bzw. OST

Beim Überschleifen mit G642 kann die maximale Abweichung für die Konturachsen und die Orientierungsachsen nicht sehr unterschiedlich sein. Die kleinere Toleranz von beiden bestimmt die Form der Überschleifbewegung bzw. Winkeltoleranz, den Orientierungsverlauf relativ stark zu glätten, ohne dabei größere Konturabweichungen hinnehmen zu müssen.

Durch Aktivierung von OSD bzw. OST ist es möglich, mit einer vorgegebenen Überschleiflänge bzw. Winkeltoleranz sehr geringe Abweichungen des Orientierungsverlaufs ohne gravierende Konturabweichungen "großzügig" zu glätten.

Hinweis

Im Unterschied zum Überschleifen der Kontur (und dem Orientierungsverlauf) mit G642 wird beim Überschleifen der Orientierung mit OSD bzw. OST kein eigener Satz gebildet, sondern die Überschleifbewegung wird direkt in die programmierten Originalsätze eingefügt.

Mit OSD bzw. OST können keine Satzübergänge überschliffen werden bei denen ein Wechsel der Interpolationsart für die Werkzeugorientierung (Vektor → Rundachse, Rundachse → Vektor) stattfindet. Diese Satzübergänge können gegebenenfalls mit den herkömmlichen Überschleiffunktionen G641, G642 bzw. G643 überschliffen werden.

7.7 Freie D-Nummernvergabe, Schneidennummer

7.7.1 Freie D-Nummernvergabe, Schneidennummer (Adresse CE)

D-Nummer

Die D-Nummern können als Korrekturnummern verwendet werden. Zusätzlich kann über die Adresse CE die Nummer der Schneide adressiert werden. Über die Systemvariable $TC_DPCE kann die Schneidenummer beschrieben werden.

Voreinstellung: Korrekturnr. == Schneidennr.

Über Maschinendaten werden die maximale Anzahl der D-Nummern (Schneidennummern) und die maximale Schneidenanzahl pro Werkzeug festgelegt (→ Maschinenhersteller). Die folgenden Befehle sind nur sinnvoll, wenn die maximale Schneidennummer (MD18105) größer als die Anzahl der Schneiden pro Werkzeug (MD18106) festgelegt wurde. Beachten Sie die Angaben des Maschinenherstellers.

Hinweis

Neben der relativen D-Nummernvergabe können die D-Nummern auch als "flache" bzw. "absolute" D-Nummern (1-32000) ohne Bezug zu einer T-Nummer vergeben werden (innerhalb der Funktion "Flache D-Nummernstruktur").

Literatur

Funktionshandbuch Grundfunktionen; Werkzeugkorrektur (W1)

7.7.2 Freie D-Nummernvergabe: D-Nummern prüfen (CHKDNO)

Funktion

Mit dem Befehl CKKDNO prüfen Sie, ob die vorhandenen D-Nummern eindeutig vergeben worden sind. Die D-Nummern aller innerhalb einer TO-Einheit definierten Werkzeuge dürfen nur einmal auftreten. Ersatzwerkzeuge werden dabei nicht berücksichtigt.

Syntax

```
state=CHKDNO(Tno1,Tno2,Dno)
```

Bedeutung

state	=TRUE:	Die D-Nummern wurden für den überprüften Bereich eindeutig vergeben.
	=FALSE:	Es erfolgte eine D-Nummernkollision oder die Parametrierung ist ungültig. Über Tno1, Tno2 und Dno werden die Parameter übergeben, die zur Kollision führten. Diese Daten können im Teileprogramm ausgewertet werden.
CHKDNO(Tno1,Tno2)	Es werden alle D-Nummern der genannten Werkzeuge geprüft.	
CHKDNO(Tno1)	Es werden alle D-Nummern von Tno1 gegen alle anderen Werkzeuge geprüft.	
CHKDNO	Es werden alle D-Nummern aller Werkzeuge gegen alle anderen Werkzeuge geprüft.	

7.7.3 Freie D-Nummernvergabe: D-Nummern umbenennen (GETDNO, SETDNO)

Funktion

D-Nummern müssen eindeutig vergeben werden. Zwei verschiedene Schneiden eines Werkzeuges können nicht dieselbe D-Nummer haben.

GETDNO

Dieser Befehl liefert die D-Nummer einer bestimmten Schneide (ce) eines Werkzeuges mit der T-Nummer t. Existiert keine D-Nummer zu den eingegebenen Parametern, wird d=0 gesetzt. Ist die D-Nummer ungültig wird ein Wert größer 32000 zurückgegeben.

SETDNO

Mit diesem Befehl weisen Sie den Wert d der D-Nummer einer Schneide ce des Werkzeuges t zu. Über state wird das Ergebnis dieser Anweisung zurückgegeben (TRUE oder FALSE). Existiert kein Datensatz zu den eingegebenen Parametern wird FALSE zurückgegeben. Syntaxfehler erzeugen einen Alarm. Die D-Nummer kann nicht explizit auf 0 gesetzt werden.

Syntax

```
d = GETDNO (t,ce)
state = SETDNO (t,ce,d)
```

Bedeutung

`d`	D-Nummer der Schneide des Werkzeuges
`t`	T-Nummer des Werkzeuges
`ce`	Schneidennummer (CE-Nummer) des Werkzeuges
`state`	Gibt an, ob der Befehl fehlerfrei ausgeführt werden konnte (TRUE oder FALSE).

Beispiel Umbenennen einer D-Nummer

Programmierung	**Kommentar**
`$TC_DP2[1,2] = 120`	`;`
`$TC_DP3[1,2] = 5.5`	`;`
`$TC_DPCE[1,2] = 3`	`;  Schneidennummer CE`
`...`	`;`
`N10 def int DNrAlt, DNrNeu = 17`	`;`
`N20 DNrAlt = GETDNO(1,3)`	`;`
`N30 SETDNO(1,3,DNrNeu)`	`;`

Damit wird der Schneide CE=3 der neue D-Wert 17 zugewiesen. Jetzt werden die Daten dieser Schneide über die D-Nummer 17 angesprochen; sowohl über die Systemvariablen als auch in der Programmierung mit der NC-Adresse.

7.7.4 Freie D-Nummernvergabe: T-Nummer zur vorgegebenen D-Nummer ermitteln (GETACTTD)

Funktion

Mit dem Befehl `GETACTTD` ermitteln Sie zu einer absoluten D-Nummer die dazugehörige T-Nummer. Es erfolgt keine Prüfung auf Eindeutigkeit. Gibt es mehrere gleiche D-Nummern innerhalb einer TO-Einheit, wird die T-Nummer des ersten gefundenen Werkzeugs zurückgegeben. Bei Verwendung "flacher" D-Nummern ist die Verwendung des Befehls nicht sinnvoll, da hier immer der Wert "1" zurückgegeben wird (keine T-Nummer in der Datenhaltung).

Syntax

```
status=GETACTTD(Tnr,Dnr)
```

Bedeutung

`Dnr`	D-Nummer, für die die T-Nummer gesucht werden soll.	
`Tnr`	Gefundene T-Nummer	
`status`	Wert:	Bedeutung:
	0	Die T-Nummer wurde gefunden. Tnr erhält den Wert der T-Nummer.
	-1	Zur angegebenen D-Nummer existiert keine T-Nummer; Tnr=0.
	-2	Die D-Nummer ist nicht absolut. Tnr erhält den Wert des ersten gefundenen Werkzeugs, das die D-Nummer mit dem Wert Dnr enthält.
	-5	Die Funktion konnte aus einem anderen Grund nicht ausgeführt werden.

7.7.5 Freie D-Nummernvergabe: D-Nummern ungültig setzen (DZERO)

Funktion

Der Befehl DZERO dient zur Unterstützung während dem Umrüsten. So gekennzeichnete Korrekturdatensätze werden nicht mehr vom Befehl CHKDNO geprüft. Um sie wieder zugänglich zu machen, muss die D-Nummer wieder mit SETDNO gesetzt werden.

Syntax

```
DZERO
```

Bedeutung

DZERO	Kennzeichnet alle D-Nummern der TO-Einheit als ungültig.

7.8 Werkzeugträgerkinematik

Voraussetzungen

Ein Werkzeugträger kann ein Werkzeug nur dann in alle möglichen Raumrichtungen orientieren, wenn

- zwei Drehachsen v_1 und v_2 vorhanden sind.
- die Drehachsen aufeinander senkrecht stehen.
- die Werkzeuglängsachse senkrecht auf der zweiten Drehachse v_2 steht.

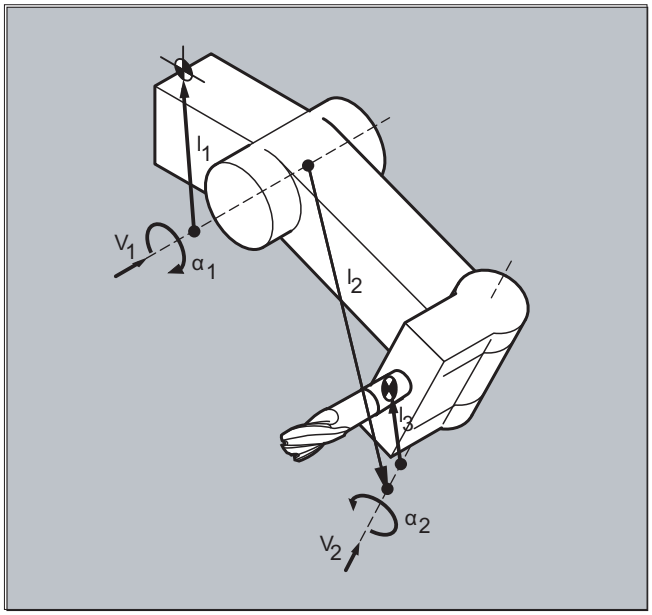
Zusätzlich gilt bei Maschinen, bei denen alle möglichen Orientierungen einstellbar sein müssen, folgende Forderung:

- die Werkzeugorientierung muss senkrecht auf der ersten Drehachse v_1 stehen.

Funktion

Die Werkzeugträgerkinematik mit maximal zwei Drehachsen v_1 oder v_2 wird über die 17 Systemvariablen `$TC_CARR1[m]` bis `$TC_CARR17[m]` beschrieben. Die Beschreibung des Werkzeugträgers besteht aus:

- dem vektoriellen Abstand von der ersten Drehachse zum Bezugspunkt des Werkzeugträgers i_1, dem vektoriellen Abstand von erster zu zweiter Drehachse i_2, dem vektoriellen Abstand von zweiter Drehachse zum Bezugspunkt des Werkzeugs i_3.
- den Richtungsvektoren beider Drehachsen v_1, v_2.
- den Drehwinkeln α_1, $\alpha_2$um die beiden Achsen. Die Drehwinkel werden mit Blickrichtung in Richtung der Drehachsvektoren im Uhrzeigersinn positiv gezählt.



Für Maschinen mit **aufgelöster Kinematik** (sowohl Werkzeug als auch Werkstück sind drehbar) wurden die Systemvariablen um die Einträge

- `$TC_CARR18[m]` bis `$TC_CARR23[m]` erweitert.

Parameter

Funktion der Systemvariablen für orientierbare Werkzeugträger			
Bezeichnung	**x-Komponente**	**y-Komponente**	**z-Komponente**
l_1 Offsetvector	$TC_CARR1[m]	$TC_CARR2[m]	$TC_CARR3[m]
l_2 Offsetvector	$TC_CARR4[m]	$TC_CARR5[m]	$TC_CARR6[m]
v_1 Drehachse	$TC_CARR7[m]	$TC_CARR8[m]	$TC_CARR9[m]
v_2 Drehachse	$TC_CARR10[m]	$TC_CARR11[m]	$TC_CARR12[m]
α_1 Drehwinkel α_2 Drehwinkel	$TC_CARR13[m] $TC_CARR14[m]		
l_3 Offsetvector	$TC_CARR15[m]	$TC_CARR16[m]	$TC_CARR17[m]

Erweiterungen der Systemvariablen für orientierbare Werkzeugträger			
Bezeichnung	**x-Komponente**	**y-Komponente**	**z-Komponente**
l_4 Offsetvector	$TC_CARR18[m]	$TC_CARR19[m]	$TC_CARR20[m]
Achsbezeichner Drehachse v_1 Drehachse v_2	Achsbezeichner der Drehachsen $v_1$und v_2 (Vorbelegung ist Null) $TC_CARR21[m] $TC_CARR22[m]		
Kinematiktyp	$TC_CARR23[m]		
Tool **P**art **M**ixed mode	Kinematiktyp-T ->	Kinematiktyp-P ->	Kinematiktyp-M
	Nur das Werkzeug ist drehbar (Vorbelegung)	Nur das Werkstück ist drehbar	Werkstück & Werkzeug sind drehbar
Offset der Drehachse v_1 Drehachse v_2	Winkel in Grad der Drehachsen $v_1$und v_2 bei Einnahme der Grundstellung $TC_CARR24[m] $TC_CARR25[m]		
Winkeloffset der Drehachse v_1 Drehachse v_2	Offset der Hirth-Verzahnung in Grad der Drehachsen $v_1$und v_2 $TC_CARR26[m] $TC_CARR27[m]		
Winkelinkrem. v_1 Drehachse v_2 Drehachse	Inkrement der Hirth-Verzahnung in Grad der Drehachsen $v_1$und v_2 $TC_CARR28[m] $TC_CARR29[m]		
Min.-Position Drehachse v_1 Drehachse v_2	Software-Limit für Minimalposition der Drehachsen $v_1$und v_2 $TC_CARR30[m] $TC_CARR31[m]		
Max.-Position Drehachse v_1 Drehachse v_2	Software-Limits für Maximalposition der Drehachsen $v_1$und v_2 $TC_CARR32[m] $TC_CARR33[m]		
Werkzeugträger Name	Anstelle einer Zahl kann ein Werkzeugträger einem Namen bekommen. $TC_CARR34[m]		
Anwender: Achsname 1 Achsname 2 Kennung	Beabsichtigte Verwendung innerhalb der Messzyklen vom Anwender. $TC_CARR35[m] $TC_CARR36[m] $TC_CARR37[m]		
Position	$TC_CARR38[m]	$TC_CARR39[m]	$TC_CARR40[m]

Erweiterungen der Systemvariablen für orientierbare Werkzeugträger			
Feinver-schiebung	Parameter, die **zu den Werten in den Basisparameternaddiert** werden können.		
l_1 Offsetvector	$TC_CARR41[m]	$TC_CARR42[m]	$TC_CARR43[m]
l_2 Offsetvector	$TC_CARR44[m]	$TC_CARR45[m]	$TC_CARR46[m]
l_3 Offsetvector	$TC_CARR55[m]	$TC_CARR56[m]	$TC_CARR57[m]
l_4 Offsetvector	$TC_CARR58[m]	$TC_CARR59[m]	$TC_CARR60[m]
v_1 Drehachse	$TC_CARR64[m]		
v_2 Drehachse	$TC_CARR65[m]		

Hinweis

Erklärungen zu den Parametern

Mit "m" wird jeweils die Nummer des zu beschreibenden Werkzeugträgers angegeben.

`$TC_CARR47` bis `$TC_CARR54` sowie `$TC_CARR61` bis `$TC_CARR63` sind nicht definiert und führen beim Versuch hierauf lesend oder schreiben zuzugreifen, zu einem Alarm.

Die Anfangs- bzw. Endpunkte der Abstandsvektoren auf den Achsen können frei gewählt werden. Die Drehwinke $\alpha 1$, $\alpha 2$ um die beiden Achsen werden im Grundzustand des Werkzeugträgers mit 0° definiert. Die Kinematik eines Werkzeugträgers kann so auf beliebig viele Möglichkeiten beschrieben werden.

Werkzeugträger mit nur einer oder keiner Drehachse können durch Nullsetzen der Richtungsvektoren einer oder beider Drehachsen beschrieben werden.
Bei einem Werkzeugträger ohne Drehachse wirken die Abstandsvektoren wie zusätzliche Werkzeugkorrekturen, deren Komponenten beim Umschalten der Bearbeitungsebenen (`G17` bis `G19`)° nicht beeinflusst werden.

Erweiterungen der Parameter

Parameter der Drehachsen

Die Systemvariablen wurden um die Einträge $TC_CARR24[m] bis $TC_CARR33[m] erweitert und wie folgt beschrieben:

Den **Offset** der Drehachsen v_1, v_2	Veränderung der Position der Drehachse v_1 oder v_2 bei Grundstellung des orientierbaren Werkzeugträgers.
Den **Winkeloffset/ Winkelinkrement** Drehachsen v_1, v_2	Offset oder Winkelinkrement der Hirth-Verzahnung der Drehachsen v_1 und v_2. Programmierter oder berechneter Winkel wird auf den nächstliegenden Wert gerundet, der sich bei ganzzahligem n aus phi = s + n * d ergibt.
Minimal- und Maximalposition Drehachsen v_1, v_2	Der Minimalposition/Maximalposition der Drehachse Grenzwinkel (Software-Limit) der Drehachse v1 und v2.

Parameter für den Anwender

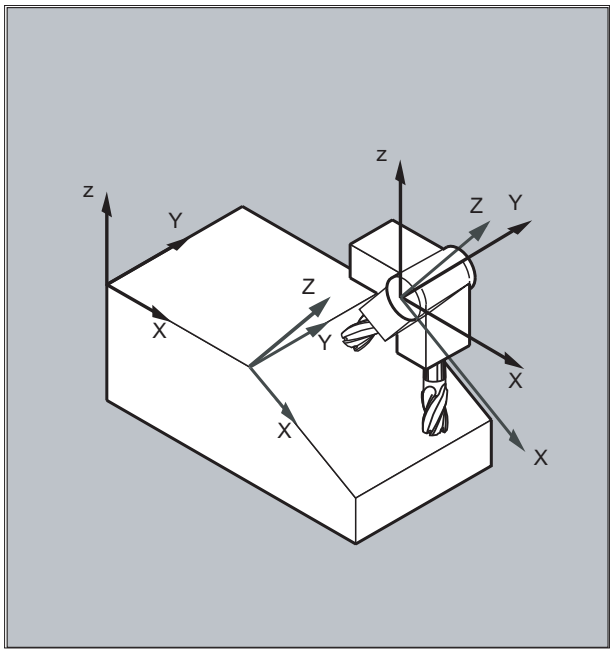
$TC_CARR34 bis $TC_CARR40 enthalten Parameter, die den Anwender zur freien Verfügung stehen und bis zum SW 6.4 standardmäßig innerhalb der NCK nicht weiter ausgewertet werden oder keine Bedeutung haben.

Parameter der Feinverschiebung

$TC_CARR41 bis $TC_CARR65 enthalten Feinverschiebungsparameter, die zu den Werten in den Basisparametern addiert werden können. Der einem Basisparameter zugeordnete Feinverschiebungswert ergibt sich, wenn zur Parameternummer der Wert 40 addiert wird.

Beispiel

Der im folgenden Beispiel verwendete Werkzeugträger lässt sich durch eine Drehung um die Y-Achse vollständig beschreiben.



Programmcode	**Kommentar**
N10 $TC_CARR8[1]=1	; Definition der Y-Komponente der ersten Drehachse des Werkzeugträgers 1.
N20 $TC_DP1[1,1]=120	; Definition eines Schaftfräsers.
N30 $TC_DP3[1,1]=20	; Definition eines Schaftfräsers mit Länge 20 mm.
N40 $TC_DP6[1,1]=5	; Definition eines Schaftfräsers mit Radius 5 mm.
N50 ROT Y37	; Framedefinition mit Drehung von 37° um die Y-Achse.
N60 X0 Y0 Z0 F10000	; Ausgangsposition anfahren.
N70 G42 CUT2DF TCOFR TCARR=1 T1 D1 X10	; Radiuskorrektur, Werkzeuglängenkorrektur im gedrehten Frame einstellen, Werkzeugträger 1, Werkzeug 1 anwählen.

Programmcode	Kommentar
N80 X40	; Bearbeitung unter einer Drehung von 37° durchführen.
N90 Y40	
N100 X0	
N110 Y0	
N120 M30	

Weitere Informationen

Aufgelöste Kinematik

Für Maschinen mit aufgelöster Kinematik (sowohl Werkzeug als auch Werkstück sind drehbar) wurden die Systemvariablen um die Einträge $TC_CARR18[m] bis $TC_CARR23[m] erweitert und wie folgt beschrieben:

Der drehbare Werkzeugtisch bestehend aus:

- dem vektoriellen Abstand der zweiten Drehachse V_2 zum Bezugspunkt eines drehbaren Werkzeugtisches I_4 der dritten Drehachse.

Die Rundachsen bestehend aus:

- den beiden Kanalbezeichnern für den Bezug der Drehachsen V_1und V_2, auf deren Position gegebenenfalls bei der Bestimmung der Orientierung des orientierbaren Werkzeugträgers zugegriffen wird.

Der Kinematiktyp mit einem der Werte T, P oder M:

- Kinematiktyp T: Nur das Werkzeug ist drehbar.
- Kinematiktyp P: Nur das Werkstück ist drehbar.
- Kinematiktyp M: Werkzeug und Werkstück sind drehbar.

Löschen der Werkzeugträgerdaten

Mit $TC_CARR1[0]=0 können die Daten aller Werkzeugträgerdatensätze gelöscht werden.

Der Kinematiktyp $TC_CARR23[T]=T muss mit einen der drei zulässigen Groß- oder Kleinbuchstaben (T,P,M) belegt werden und sollte aus diesen Grund nicht gelöscht werden.

Ändern der Werkzeugträgerdaten

Jeder der beschriebenen Werte kann durch Zuweisung eines neuen Wertes im Teileprogramm verändert werden. Jedes andere Zeichen als T, P oder M führt bei dem Versuch, den orientierbaren Werkzeugträger zu aktivieren, zu einen Alarm.

Lesen der Werkzeugträgerdaten

Jeder der beschriebenen Werte kann durch Zuweisung an eine Variable im Teileprogramm gelesen werden.

Feinverschiebungen

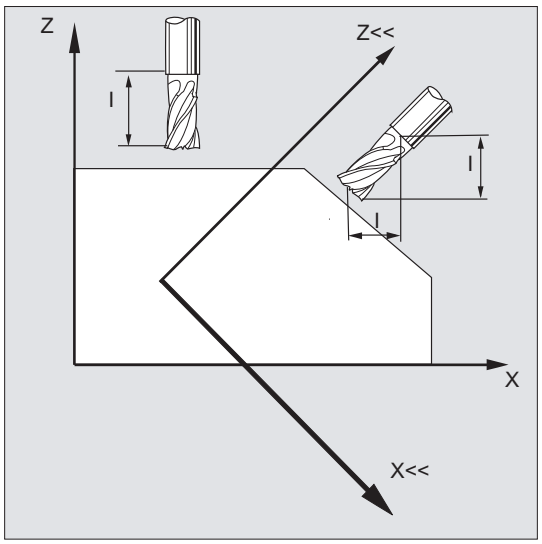
Ein unzulässiger Feinverschiebungswert wird erst erkannt, wenn ein orientierbarer Werkzeugträger aktiviert wird, der solch einen Wert enthält und gleichzeitig das Settingdatum SD42974 $SC_TOCARR_FINE_CORRECTION = TRUE ist.

Der Betrag der zulässigen Feinverschiebung wird über Maschinendaten auf einen maximal zulässigen Wert begrenzt.

7.9 Werkzeuglängenkorrektur für orientierbare Werkzeugträger (TCARR, TCOABS, TCOFR, TCOFRX, TCOFRY, TCOFRZ)

Funktion

Mit veränderter Raumorientierung des Werkzeugs ändern sich auch dessen Werkzeuglängenkomponenten.



Nach Umrüsten, z. B. durch manuelle Einstellung oder Wechsel des Werkzeugträgers mit fester räumlicher Ausrichtung, müssen daher die Werkzeuglängenkomponenten neu ermittelt werden. Dies erfolgt mit den Wegbefehlen TCOABS und TCOFR.

Bei einem orientierbaren Werkzeugträger eines aktiven Frames kann bei Werkzeuganwahl mit TCOFRZ, TCOFRY und TCOFRX die Richtung, in die das Werkzeug zeigen soll, bestimmt werden.

Syntax

```
TCARR=[<m>]
TCOABS
TCOFR
TCOFRZ
TCOFRY
TCOFRX
```

Bedeutung

TCARR=[<m>]:	Werkzeugträger mit der Nummer "m" anfordern
TCOABS:	Werkzeuglängenkomponenten aus der aktuellen Werkzeugträgerorientierung berechnen
TCOFR:	Werkzeuglängenkomponenten aus der Orientierung des aktiven Frames bestimmen
TCOFRZ:	Orientierbarer Werkzeugträger aus aktiven Frame, dessen Werkzeug in Z-Richtung zeigt
TCOFRY:	Orientierbarer Werkzeugträger aus aktiven Frame, dessen Werkzeug in Y-Richtung zeigt
TCOFRX:	Orientierbarer Werkzeugträger aus aktiven Frame, dessen Werkzeug in X-Richtung zeigt

Weitere Informationen

Werkzeuglängenkorrektur aus Trägerorientierung (TCOABS)

TCOABS berechnet die Werkzeuglängenkorrektur aus den aktuellen Orientierungswinkeln des Werkzeugträgers; abgelegt in den Systemvariablen $TC_CARR13 und $TC_CARR14.

Zur Definition der Werkzeugträgerkinematik mit Systemvariablen siehe
" Werkzeugträgerkinematik (Seite 434) ".

Zur Neuberechnung der Werkzeuglängenkorrektur bei Frame-Wechsel muss das Werkzeug nochmals angewählt werden.

Werkzeugrichtung aus aktiven Frame

Der orientierbare Werkzeugträger kann so eingestellt werden, dass das Werkzeug in folgende Richtungen zeigt:

- mit TCOFR bzw. TCOFRZ in Z-Richtung
- mit TCOFRY in Y-Richtung
- mit TCOFRX in X-Richtung

Ein Umschalten zwischen TCOFR und TCOABS bewirkt eine Neuberechnung der Werkzeuglängenkorrektur.

Werkzeugträger anfordern (TCARR)

Mit TCARR werden mit der Werkzeugträgernummer m dessen Geometriedaten angefordert (Korrekturspeicher).

Mit m=0 wird der aktive Werkzeugträger abgewählt.

Die Geometriedaten des Werkzeugträgers werden erst nach Aufruf eines Werkzeugs aktiv. Das angewählte Werkzeug bleibt über den Wechsel eines Werkzeugträgers hinaus aktiv.

Die aktuellen Geometriedaten des Werkzeugträgers können auch im Teileprogramm über die entsprechenden Systemvariablen definiert werden.

Neuberechnung der Werkzeuglängenkorrektur (TCOABS) bei Frame-Wechsel

Zur Neuberechnung der Werkzeuglängenkorrektur bei Frame-Wechsel muss das Werkzeug nochmals angewählt werden.

Hinweis

Die Werkzeugorientierung muss dem aktiven Frame manuell angepasst werden.

Bei der Berechnung der Werkzeuglängenkorrektur werden in einem Zwischenschritt auch die Drehwinkel des Werkzeugträgers berechnet. Da bei Werkzeugträgern mit zwei Drehachsen im Allgemeinen zwei Drehwinkelpaare existieren, mit denen die Werkzeugorientierung dem aktiven Frame angepasst werden kann, müssen die in den Systemvariablen abgelegten Drehwinkelwerte zumindest annähernd den mechanisch eingestellten Drehwinkeln entsprechen.

Hinweis

Werkzeugorientierung

Die Steuerung kann die über die Frame-Orientierung berechneten Verdrehwinkel nicht auf die Einstellbarkeit an der Maschine überprüfen.

Sind die Drehachsen des Werkzeugträgers konstruktiv so angeordnet, dass die durch die Frame-Orientierung berechnete Werkzeugorientierung nicht erreicht werden kann, wird ein Alarm ausgegeben.

Die Kombination von Werkzeugfeinkorrektur und den Funktionalitäten zur Werkzeuglängenkorrektur bei beweglichen Werkzeugträgern ist nicht zulässig. Beim Versuch beide Funktionen gleichzeitig aufzurufen, erfolgt eine Fehlermeldung.

Mit `TOFRAME` ist es möglich, einen Frame aufgrund der Orientierungsrichtung des angewählten Werkzeugträgers zu definieren. Genauere Informationen siehe Kapitel "Frames".

Bei aktiver Orientierungstransformation (3-, 4-, 5-Achstransformation) kann ein Werkzeugträger mit von der Null-Lage abweichender Orientierung angewählt werden, ohne dass dabei ein Alarm ausgegeben wird.

Übergabeparameter von Standard- und Messzyklen

Für die Übergabeparameter von Standard- und Messzyklen gelten definierte Wertebereiche.

Bei Winkelwerten ist der Wertebereich wie folgt festgelegt:

- Drehung um 1. Geometrieachse: -180 Grad bis +180 Grad
- Drehung um 2. Geometrieachse: -90 Grad bis +90 Grad
- Drehung um 3. Geometrieachse: -180 Grad bis +180 Grad

Siehe Kapitel Frames, "Programmierbare Drehung (ROT, AROT, RPL)".

Hinweis

Bei der Übergabe von Winkelwerten an einen Standard- oder Messzyklus ist zu beachten:

Werte kleiner als die Rechenfeinheit der NC sind auf Null zu runden!

Die Rechenfeinheit der NC für Winkelpositionen ist festgelegt im Maschinendatum:

MD10210 $MN_INT_INCR_PER_DEG

7.10 Online-Werkzeuglängenkorrektur (TOFFON, TOFFOF)

Funktion

Über die Systemvariable $AA_TOFF[<n>] können die effektiven Werkzeuglängen entsprechend der drei Werkzeugrichtungen dreidimensional in Echtzeit überlagert werden.

Als Index <n> werden die drei Geometrieachsbezeichner verwendet. Damit ist die Anzahl der aktiven Korrekturrichtungen durch die zur selben Zeit aktiven Geometrieachsen festgelegt.

Alle Korrekturen können gleichzeitig aktiv sein.

Die Funktion Online-Werkzeuglängenkorrektur ist anwendbar bei:

- Orientierungstransformation TRAORI
- Orientierbare Werkzeugträger TCARR

Hinweis

Die Online-Werkzeuglängenkorrektur ist eine **Option,** die vorher frei geschaltet werden muss. Nur in Verbindung mit einer aktiven Orientierungstransformation oder einem aktiven orientierbaren Werkzeugträger ist diese Funktion sinnvoll.

Syntax

```
TRAORI
TOFFON(<Korrekturrichtung>[,<Offsetwert>])
WHEN TRUE DO $AA_TOFF[<Korrekturrichtung>]              ; In Synchronaktionen.
...
TOFFOF(<Korrekturrichtung>)
```

Weitere Erläuterungen zur Programmierung der Online-Werkzeuglängenkorrektur in Bewegungssynchronaktionen siehe "Online-Werkzeuglängenkorrektur ($AA_TOFF) (Seite 596)".

Bedeutung

`TOFFON`: Online-Werkzeuglängenkorrektur **aktivieren**

- `<Korrekturrichtung>`: Werkzeugrichtung (`X`, `Y`, `Z`), in der die Online-Werkzeuglängenkorrektur wirksam sein soll.
- `<Offsetwert>`: Bei der Aktivierung kann für die entsprechende Korrekturrichtung ein Offsetwert angegeben werden, der sofort herausgefahren wird.

`TOFFOF`: Online-Werkzeuglängenkorrektur **zurücksetzen**

Die Korrekturwerte in der angegebenen Korrekturrichtung werden zurückgesetzt und es wird ein Vorlaufstopp ausgelöst.

Beispiele

Beispiel 1: Anwahl der Werkzeuglängenkorrektur

Programmcode	Kommentar
MD21190 $MC_TOFF_MODE =1	; Absolute Werte werden angefahren.
MD21194 $MC_TOFF_VELO[0] =1000	
MD21196 $MC_TOFF_VELO[1] =1000	
MD21194 $MC_TOFF_VELO[2] =1000	
MD21196 $MC_TOFF_ACCEL[0] =1	
MD21196 $MC_TOFF_ACCEL[1] =1	
MD21196 $MC_TOFF_ACCEL[2] =1	
N5 DEF REAL XOFFSET	
N10 TRAORI(1)	; Transformation ein.
N20 TOFFON(Z)	; Aktivierung der Online-WZL-Korrektur für die Z-Werkzeugrichtung.
N30 WHEN TRUE DO $AA_TOFF[Z]=10 G4 F5	; Für die Z-Werkzeugrichtung wird eine WZL-Korrektur von 10 interpoliert.
...	
N100 XOFFSET=$AA_TOFF_VAL[X]	; Aktuelle Korrektur in X-Richtung zuweisen.
N120 TOFFON(X,-XOFFSET) G4 F5	; Für die X-Werkzeugrichtung wird die WZL-Korrektur wieder zu 0 zurückgefahren.

Beispiel 2: Abwahl der Werkzeuglängenkorrektur

Programmcode	Kommentar
N10 TRAORI(1)	; Transformation ein.
N20 TOFFON(X)	; Aktivierung der Online-WZL-Korrektur für die X-Werkzeugrichtung.
N30 WHEN TRUE DO $AA_TOFF[X]=10 G4 F5	; Für die X-Werkzeugrichtung wird eine WZL-Korrektur von 10 interpoliert.
...	
N80 TOFFOF(X)	; Positionsoffset der X-Werkzeugrichtung wird gelöscht: ...$AA_TOFF[X]=0 Es wird keine Achse verfahren. Zur aktuellen Position im WKS wird der Positionsoffset entsprechend der aktuellen Orientierung hinzugerechnet.

Weitere Informationen

Satzaufbereitung

Bei der Satzaufbereitung im Vorlauf wird der im Hauptlauf wirksame aktuelle Werkzeuglängenoffset mit berücksichtigt. Um die maximal zulässigen Achsgeschwindigkeiten weitgehend ausnutzen zu können, ist es erforderlich, die Satzaufbereitung mit einem Vorlaufstopp `STOPRE` anzuhalten, während ein Werkzeugoffset aufgebaut wird.

Der Werkzeugoffset ist zum Vorlaufzeitpunkt auch immer dann bekannt, wenn die Werkzeuglängenkorrekturen nach Programmstart nicht mehr verändert werden, oder wenn nach einer Veränderung der Werkzeuglängenkorrekturen mehr Sätze abgearbeitet wurden als der IPO-Buffer zwischen Vorlauf und Hauptlauf aufnehmen kann.

Variable $AA_TOFF_PREP_DIFF

Das Maß für die Differenz zwischen der aktuellen im Interpolator wirksamen Korrektur und der Korrektur, die zum Zeitpunkt der Satzaufbereitung wirksam war, kann in der Variablen$AA_TOFF_PREP_DIFF[<n>] abgefragt werden.

Maschinendaten und Settingdaten einstellen

Für die Online-Werkzeuglängenkorrektur stehen folgende Systemdaten zur Verfügung:

- MD20610 $MC_ADD_MOVE_ACCEL_RESERVE (Beschleunigungsreserve für überlagerte Bewegung)
- MD21190 $MC_TOFF_MODE

  Inhalt der Systemvariable$AA_TOFF[<n>] wird als absoluter Wert herausgefahren oder aufintegriert.
- MD21194 $MC_TOFF_VELO (Geschwindigkeit der Online-Werkzeuglängenkorrektur)
- MD21196 $MC_TOFF_ACCEL (Beschleunigung der Online-Werkzeuglängenkorrektur)
- Settingdatum zur Vorgabe von Grenzwerten:
  SD42970 $SC_TOFF_LIMIT (Obergrenze des Werkzeuglängenkorrekturwertes)

Literatur:

Funktionshandbuch Sonderfunktionen; F2: Mehrachstransformationen

7.11 Schneidendaten-Modifikation bei drehbaren Werkzeugen (CUTMOD)

Funktion

Mit der Funktion "Schneidendaten-Modifikation bei drehbaren Werkzeugen" können die veränderten geometrischen Verhältnisse, die sich bei der Drehung von Werkzeugen (vorwiegend Drehwerkzeuge, aber auch Bohr- und Fräswerkzeuge) relativ zum bearbeiten Werkstück ergeben, bei der Werkzeugkorrektur berücksichtigt werden.

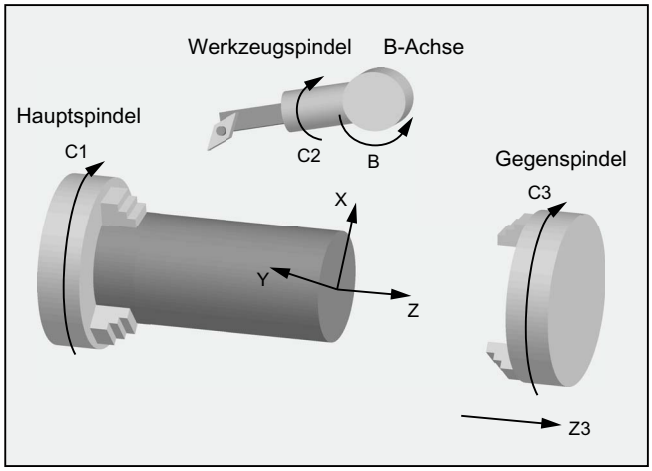


Bild 7-1 Drehbares Werkzeug bei einer Drehmaschine

Die aktuelle Drehung des Werkzeugs wird dabei immer aus einem aktuell aktiven orientierbaren Werkzeugträger (siehe " Werkzeuglängenkorrektur für orientierbare Werkzeugträger (Seite 439) ") ermittelt.

Die Funktion wird aktiviert mit dem Befehl `CUTMOD`.

Syntax

```
CUTMOD=<Wert>
```

Bedeutung

CUTMOD Befehl zum Einschalten der Funktion "Schneidendaten-Modifikation bei drehbaren Werkzeugen"

<Wert> Dem CUTMOD-Befehl können folgende Werte zugewiesen werden:

0 Die Funktion ist deaktiviert.

Die von den Systemvariablen $P_AD... gelieferten Werte sind gleich den korrespondierenden Werkzeugparametern.

> 0 Die Funktion wird aktiviert, falls ein orientierbarer Werkzeugträger mit der angegebenen Nummer aktiv ist, d. h. die Aktivierung ist an einen bestimmten orientierbaren Werkzeugträger gebunden.

Die von den Systemvariablen $P_AD... gelieferten Werte sind gegenüber den korrespondierenden Werkzeugparametern abhängig von der aktiven Drehung gegebenenfalls modifiziert.

Die Deaktivierung des bezeichneten orientierbaren Werkzeugträgers deaktiviert die Funktion temporär, die Aktivierung eines anderen orientierbaren Werkzeugträgers deaktiviert sie permanent. Im ersten Fall wird die Funktion deshalb bei erneuter Anwahl des gleichen orientierbaren Werkzeugträgers wieder aktiviert, im zweiten Fall ist eine erneute Anwahl notwendig, auch dann wenn zu einem späteren Zeitpunkt der orientierbare Werkzeugträger mit der angegebenen Nummer erneut aktiviert wird.

Die Funktion wird durch Reset nicht beeinflusst.

-1 Die Funktion wird immer aktiviert, falls ein orientierbarer Werkzeugträger aktiv ist.

Beim Wechsel des Werkzeugträgers oder bei dessen Abwahl und einer späteren erneuten Anwahl muss CUTMOD nicht erneut gesetzt werden.

-2 Die Funktion wird immer aktiviert, falls ein orientierbarer Werkzeugträger aktiv ist, dessen Nummer gleich der des aktuell aktiven orientierbaren Werkzeugträgers ist.

Ist kein orientierbarer Werkzeugträger aktiv, ist das gleichbedeutend mit CUTMOD=0. Ist ein orientierbarer Werkzeugträger aktiv, ist das gleichbedeutend mit der unmittelbaren Angabe der aktuellen Werkzeugträgernummer.

< -2 Werte kleiner -2 werden ignoriert, d. h. dieser Fall wird so behandelt, als wäre CUTMOD nicht programmiert.

Hinweis:
Dieser Wertebereich sollte nicht verwendet werden, weil er für eventuelle spätere Erweiterungen reserviert ist.

Hinweis

SD42984 $SC_CUTDIRMOD

Die über den Befehl CUTMOD aktivierbare Funktion ersetzt die über das Settingdatum SD42984 $SC_CUTDIRMOD aktivierbare Funktion. Diese Funktion steht jedoch weiterhin unverändert zur Verfügung. Da es aber nicht sinnvoll ist, beide Funktionen parallel zu nutzen, kann sie nur aktiviert werden, wenn CUTMOD gleich Null ist.

Beispiel

Das folgende Beispiel bezieht sich auf ein Werkzeug mit der Schneidenlage 3 und einem orientierbaren Werkzeugträger, der das Werkzeug um die B-Achse drehen kann.

Die Zahlenwerte in den Kommentaren geben jeweils die Satzendpositionen in Maschinenkoordinaten (MKS) in der Reihenfolge X, Y, Z an.

```
Programmcode                                              Kommentar
N10 $TC_DP1[1,1]=500
N20 $TC_DP2[1,1]=3                                        ; Schneidenlage
N30 $TC_DP3[1,1]=12
N40 $TC_DP4[1,1]=1
N50 $TC_DP6[1,1]=6
N60 $TC_DP10[1,1]=110                                     ; Halterwinkel
N70 $TC_DP11[1,1]=3                                       ; Schnittrichtung
N80 $TC_DP24[1,1]=25                                      ; Freiwinkel

N90 $TC_CARR7[2]=0 $TC_CARR8[2]=1 $TC_CARR9[2]=0          ; B-Achse
N100 $TC_CARR10[2]=0 $TC_CARR11[2]=0 $TC_CARR12[2]=1      ; C-Achse
N110 $TC_CARR13[2]=0
N120 $TC_CARR14[2]=0
N130 $TC_CARR21[2]=X
N140 $TC_CARR22[2]=X
N150 $TC_CARR23[2]="M"

N160 TCOABS CUTMOD=0
N170 G18 T1 D1 TCARR=2                                      X           Y           Z
N180 X0 Y0 Z0 F10000                                      ; 12.000      0.000       1.000

N190 $TC_CARR13[2]=30
N200 TCARR=2
N210 X0 Y0 Z0                                             ; 10.892      0.000       -5.134
N220 G42 Z-10                                             ; 8.696       0.000       -17.330
N230 Z-20                                                 ; 8.696       0.000       -21.330
N240 X10                                                  ; 12.696      0.000       -21.330
N250 G40 X20 Z0                                           ; 30.892      0.000       -5.134

N260 CUTMOD=2 X0 Y0 Z0                                    ; 8.696       0.000       -7.330
N270 G42 Z-10                                             ; 8.696       0.000       -17.330
N280 Z-20                                                 ; 8.696       0.000       -21.330
N290 X10                                                  ; 12.696      0.000       -21.330
N300 G40 X20 Z0                                           ; 28.696      0.000       -7.330

N310 M30
```

Erläuterungen:

In Satz N180 wird zunächst das Werkzeug bei CUTMOD=0 und nicht gedrehtem orientierbaren Werkzeugträger angewählt. Da alle Offsetvektoren des orientierbaren Werkzeugträgers 0 sind, wird die Position angefahren, die den in $TC_DP3[1,1] und $TC_DP4[1,1] angegebenen Werkzeuglängen entspricht.

In Satz N200 wird der orientierbare Werkzeugträger mit einer Drehung von 30° um die B-Achse aktiviert. Da die Schneidenlage wegen CUTMOD=0 nicht modifiziert wird, ist nach wie vor der alte Schneidenbezugspunkt maßgebend. Deshalb wird in Satz N210 die Position angefahren, die den alten Schneidenbezugspunkt im Nullpunkt beibehält (d. h. der Vektor (1, 12) wird in der Z/X-Ebene um 30° gedreht).

In Satz N260 ist im Unterschied zu Satz N200 CUTMOD=2 wirksam. Aufgrund der Drehung des orientierbaren Werkzeugträgers wird die modifizierte Schneidenlage 8. Daraus folgen auch abweichende Achspositionen.

In den Sätzen N220 bzw. N270 wird jeweils die Werkzeugradiuskorrektur (WRK) aktiviert. Die unterschiedliche Schneidenlage in beiden Programmstücken hat auf die Endpositionen der Sätze, in denen die WRK aktiv ist, keinen Einfluss, die entsprechenden Positionen sind deshalb identisch. Erst in den Abwahlsätzen N260 bzw. N300 wirken sich die unterschiedlichen Schneidenlagen wieder aus.

Weitere Informationen

Wirksamkeit der modifizierten Schneidendaten

Die modifizierte Schneidenlage und der modifizierte Schneidenbezugspunkt werden bei Programmierung auch für ein bereits aktives Werkzeug sofort wirksam. Eine Werkzeugneuanwahl ist dazu nicht notwendig.

Einfluss der aktiven Arbeitsebene

Für die Bestimmung von modifizierter Schneidenlage, Schnittrichtung und Halter- bzw. Freiwinkel ist die Betrachtung der Schneide in der jeweils aktiven Ebene (G17 - G19) maßgebend.

Enthält jedoch das Settingdatum SD42940 $SC_TOOL_LENGTH_CONST (Wechsel der Werkzeuglängenkomponenten bei Ebenenwechsel) einen gültigen Wert ungleich Null (plus oder minus 17, 18 oder 19), so bestimmt dessen Inhalt die Ebene, in der die relevanten Größen betrachtet werden.

Systemvariablen

Folgende Systemvariablen stehen zur Verfügung:

Systemvariablen	Bedeutung
$P_CUTMOD_ANG / $AC_CUTMOD_ANG	Liefert den (nicht gerundeten) Winkel in der aktiven Bearbeitungsebene, der für die Modifikation der Schneidendaten (Schneidenlage, Schnittrichtung, Freiwinkel und Halterwinkel) bei den mit CUTMOD bzw. $SC_CUTDIRMOD aktivierten Funktionen zugrunde gelegt wurde. $P_CUTMOD_ANG bezieht sich auf den aktuellen Zustand im Vorlauf, $AC_CUTMOD_ANG auf den aktuellen Hauptlaufsatz.
$P_CUTMOD / $AC_CUTMOD	Liest den aktuell gültigen Wert, der zuletzt mit dem Befehl CUTMOD programmiert wurde (Nummer des Werkzeugträgers, für den die Schneidendaten-Modifikation aktiviert werden soll). War der letzte programmierte CUTMOD-Wert = -2 (Aktivierung mit dem aktuell aktiven orientierbaren Werkzeugträger), dann wird in $P_CUTMOD nicht der Wert -2, sondern die Nummer des zum Zeitpunkt der Programmierung aktiven orientierbaren Werkzeugträgers zurückgeliefert. $P_CUTMOD bezieht sich auf den aktuellen Zustand im Vorlauf, $AC_CUTMOD auf den aktuellen Hauptlaufsatz.
$P_CUT_INV / $AC_CUT_INV	Liefert den Wert TRUE, wenn das Werkzeug so gedreht ist, dass die Spindeldrehrichtung invertiert werden muss. Dazu müssen in dem Satz, auf den sich die jeweilige Leseoperation bezieht, die folgenden vier Bedingungen erfüllt sein: 1. Es ist ein Dreh- oder Schleifwerkzeug aktiv (Werkzeugtypen 400 bis 599 und / oder SD42950 $SC_TOOL_LENGTH_TYPE = 2). 2. Die Schneidenbeeinflussung wurde mit dem Sprachbefehl CUTMOD aktiviert. 3. Es ist ein orientierbarer Werkzeugträger aktiv, der durch den numerischen Wert von CUTMOD bezeichnet wurde. 4. Der orientierbare Werkzeugträger dreht das Werkzeug um eine Achse in der Bearbeitungsebene (typischerweise die C-Achse) so, dass die resultierende Normale der Werkzeugschneide gegenüber der Ausgangslage um mehr als 90° (typischerweise 180°) gedreht ist. Ist mindestens eine der genannten vier Bedingungen nicht erfüllt, ist der Inhalt der Variablen FALSE. Für Werkzeuge, deren Schneidenlage nicht definiert ist, ist der Wert der Variablen immer FALSE. $P_CUT_INV bezieht sich auf den aktuellen Zustand im Vorlauf und $AC_CUT_INV auf den aktuellen Hauptlaufsatz.

Alle Hauptlaufvariablen ($AC_CUTMOD_ANG, $AC_CUTMOD und $AC_CUT_INV) können in Synchronaktionen gelesen werden. Ein Lesezugriff aus dem Vorlauf generiert einen Vorlaufstopp.

Modifizierte Schneidendaten:

Falls eine Werkzeugdrehung aktiv ist, werden die modifizierten Daten in den folgenden Systemvariablen zur Verfügung gestellt:

Systemvariable	Bedeutung
$P_AD[2]	Schneidenlage
$P_AD[10]	Halterwinkel
$P_AD[11]	Schnittrichtung
$P_AD[24]	Freiwinkel

Hinweis

Die Daten sind gegenüber den korrespondierenden Werkzeugparametern ($TC_DP2[..., ...] usw.) immer dann modifiziert, wenn die Funktion "Schneidendaten-Modifikation bei drehbaren Werkzeugen" mit dem Befehl `CUTMOD` aktiviert wurde und ein orientierbarer Werkzeugträger aktiv ist, der eine Werkzeugdrehung bewirkt.

Literatur

Weitere Informationen zur Funktion "Schneidendaten-Modifikation bei drehbaren Werkzeugen" siehe:

Funktionshandbuch Grundfunktionen; Werkzeugkorrektur (W1)

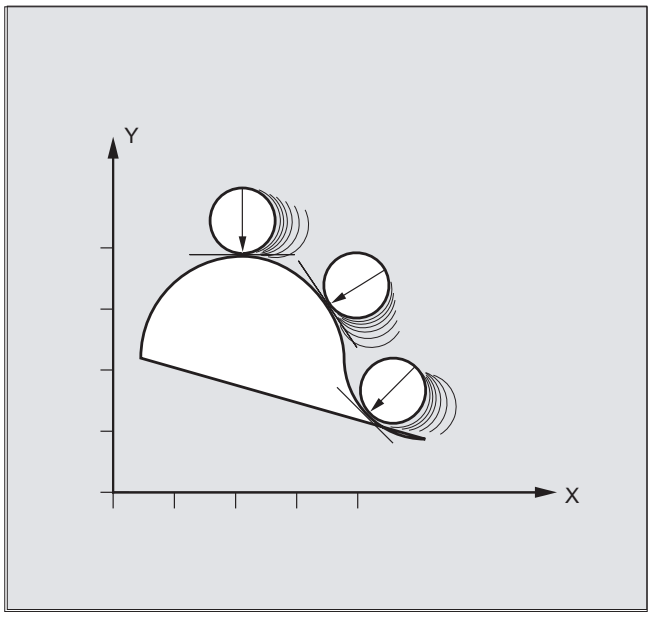
Bahnverhalten

8

8.1 Tangentialsteuerung (TANG, TANGON, TANGOF, TLIFT, TANGDEL)

Funktion

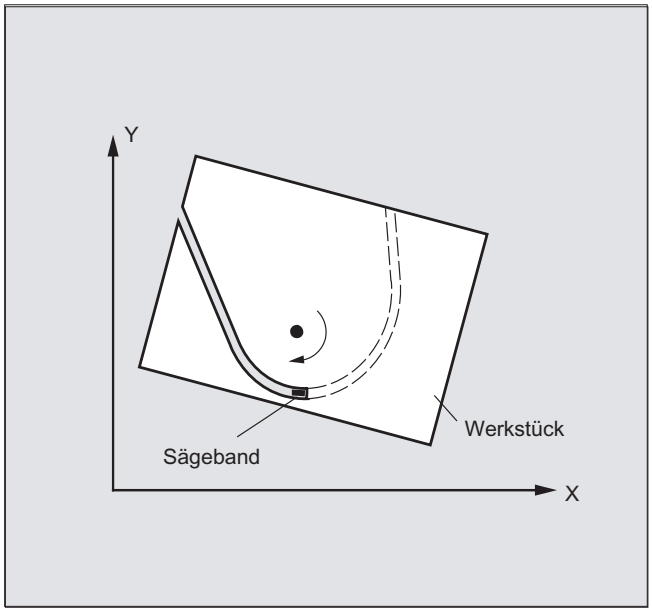
Die Folgeachse wird gemäß der Tangente an der durch die Leitachsen festgelegten Bahn nachgeführt. Dadurch kann ein Werkzeug parallel zur Kontur ausgerichtet werden. Durch den in der TANGON-Anweisung programmierten Winkel kann das Werkzeug relativ zur Tangente angestellt werden.



Anwendung

Die Tangentialsteuerung kann z. B. eingesetzt werden bei:

- Tangentiellem Anstellen eines drehbaren Werkzeugs beim Nibbeln
- Nachführen der Werkstückausrichtung bei einer Bandsäge (siehe folgende Abbildung)
- Anstellen eines Abrichtwerkzeugs an eine Schleifscheibe
- Anstellen eines Schneidrädchens zur Glas- oder Papierverarbeitung
- Tangentialer Zuführung eines Drahtes beim 5-achsigen Schweißen



Syntax

Tangentiale Nachführung definieren:
`TANG(<FAchse>,<LAchse1>,<LAchse2>,<Koppelfaktor>,<KS>,<Opt>)`

Tangentialsteuerung einschalten:
`TANGON(<FAchse>,<Winkel>,<Dist>,<Winkeltoleranz>)`

Tangentialsteuerung ausschalten:
`TANGOF(<FAchse>)`

Funktion "Zwischensatz an Konturecken einfügen" einschalten:
`TLIFT(<FAchse>)`

Die `TLIFT`-Anweisung wird im Anschluss an die Achsenzuordnung mit `TANG(…)` angegeben.

Funktion "Zwischensatz an Konturecken einfügen" ausschalten:

`TANG(...)`-Anweisung wiederholen ohne folgendes `TLIFT(<FAchse>)`.

Definition einer Tangentialen Nachführung löschen:
`TANGDEL(<FAchse>)`

Eine bestehende anwenderdefinierte Tangentiale Nachführung muss gelöscht werden, wenn eine neue Tangentiale Nachführung mit der gleichen Folgeachse im Vorbereitungsaufruf `TANG` definiert werden soll. Ein Löschen ist nur möglich, wenn die Kopplung mit `TANGOF(<FAchse>)` ausgeschaltet ist.

Bedeutung

`TANG`:	Vorbereitende Anweisung für die Definition einer tangentialen Nachführung
`TANGON`:	Tangentialsteuerung für die angegebene Folgeachse einschalten

`TANGOF`: Tangentialsteuerung für die angegebene Folgeachse ausschalten

`TLIFT`: Funktion "Zwischensatz an Konturecken einfügen" einschalten

`TANGDEL`: Definition einer Tangentialen Nachführung löschen

`<FAchse>`: Folgeachse: Tangential nachgeführte Zusatzrundachse

`<LAchse1>,<LAchse2>`: Leitachsen: Bahnachsen, aus denen die Tangente für die Nachführung bestimmt wird

`<Koppelfaktor>`: Koppelfaktor: Zusammenhang zwischen Winkeländerung der Tangente und der nachgeführten Achse

Voreinstellung: 1

Hinweis:
Ein Koppelfaktor von 1 muss nicht explizit programmiert werden.

`<KS>`: Kennbuchstabe für Koordinatensystem

`"B"`: Basiskoordinatensystem (Voreinstellung)

Hinweis:
<`<KS>` = `"B"` muss nicht explizit programmiert werden.

`"W"`: Werstückkoordinatensystem (nicht verfügbar)

`<Opt>`: Optimierung

`"S"`: Standard (Voreinstellung)

Hinweis:
<`<Opt>` = `"S"` muss nicht explizit programmiert werden.

`"P"`: Automatische Anpassung des Zeitverlaufs der tangentialen Achse und der Kontur

Hinweis:
Mit **<**`<Opt>` = `"P"` wird die Dynamik der Folgeachse bei der Geschwindigkeitsbegrenzung der Leitachsen mitberücksichtigt. Diese Einstellung ist vor allem beim Einsatz von kinematischen Transformationen zu empfehlen.

`<Winkel>`: Offsetwinkel der Folgeachse

`<Dist>`: Überschleifweg der Folgeachse (erforderlich bei `<Opt>` **=** `"P"`)

`<Winkeltoleranz>`: Winkeltoleranz der Folgeachse (optional; Auswertung nur bei `<Opt>` **=** `"P"`)

Hinweis:
Die Parameter **<**`<Dist>` und `<Winkeltoleranz>` begrenzen gezielt den Fehler zwischen der nachgeführten Achse und der Tangente der Leitachsen.

Beispiele

Beispiel 1: Tangentiale Nachführung definieren und einschalten

Programmcode	Kommentar
`N10 TANG(C,X,Y,1,"B","P")`	`; Definition einer tangentialen Nachführung: Rundachse C soll den Geometrieachsen X und Y folgen.`
`N20 TANGON(C,90)`	`; Die C-Achse ist Folgeachse. Sie wird bei jeder Bewegung der Bahnachsen in eine 90°-Position zur Bahn-Tangente gedreht.`
`...`	

Hinweis

Vereinfachte Programmierung

`TANG(C,X,Y,1,"B","P")` kann vereinfacht programmiert werden als `TANG(C,X,Y,,,"P")`.

Beispiel 2: Ebenenwechsel

Programmcode	Kommentar
`N10 TANG(A,X,Y,1)`	`; 1.Definition der Tangentialen Nachführung.`
`N20 TANGON(A)`	`; Aktivierung der Kopplung.`
`N30 X10 Y20`	`; Radius`
`...`	
`N80 TANGOF(A)`	`; Ausschalten der 1.Kopplung.`
`N90 TANGDEL(A)`	`; Löschen der 1.Definition.`
`...`	
`TANG(A,X,Z)`	`; 2.Definition der Tangentialen Nachführung.`
`TANGON(A)`	`; Aktivierung der neuen Kopplung.`
`...`	
`N200 M30`	

Beispiel 3: Geometrieachsumschaltung und TANGDEL

Es wird kein Alarm erzeugt.

```
Programmcode                    Kommentar
N10 GEOAX(2,Y1)                 ; Y1 ist Geometrieachse 2.
N20 TANG(A,X,Y)                 ; 1.Definition der Tangentialen Nachführung.
N30 TANGON(A,90)                ; Aktivierung der Nachführung mit Y1
N40 G2 F8000 X0 Y0 I0 J50
N50 TANGOF(A)                   ; Deaktivierung der Nachführung mit Y1.
N60 TANGDEL(A)                  ; Löschen der 1.Definition.
N70 GEOAX(2,Y2)                 ; Y2 ist neue Geometrieachse 2.
N80 TANG(A,X,Y)                 ; 2.Definition der Tangentialen Nachführung.
N90 TANGON(A,90)                ; Aktivierung der Nachführung mit Y2.
...
```

Beispiel 4: Tangentiale Nachführung mit automatischer Optimierung

Y1 ist Geometrieachse 2.

```
Programmcode                                Kommentar
...
N80 G0 C0
N100 F=50000
N110 G1 X1000 Y500
N120 TRAORI
N130 G642                                   ; Überschleifen unter Einhaltung der
                                              maximal erlaubten Bahnabweichung.
N171 TRANS X50 Y50
N180 TANG(C,X,Y,1,,"P")                     ; Definition Tangentiale Nachführung mit
                                              automatischer Optimierung der
                                              Bahngeschwindigkeit.
N190 TANGON(C,0,5.0,2.0)                    ; Tangentiale Nachführung mit
                                              automatischer Optimierung einschalten:
                                              Überschleifweg 5 mm, Winkeltoleranz
                                              2 Grad.
N210 G1 X1310 Y500
N215 G1 X1420 Y500
N220 G3 X1500 Y580 I=AC(1420) J=AC(580)
N230 G1 X1500 Y760
N240 G3 X1360 Y900 I=AC(1360) J=AC(760)
N250 G1 X1000 Y900
N280 TANGOF(C)
N290 TRAFOOF
N300 M02
```

Weitere Informationen

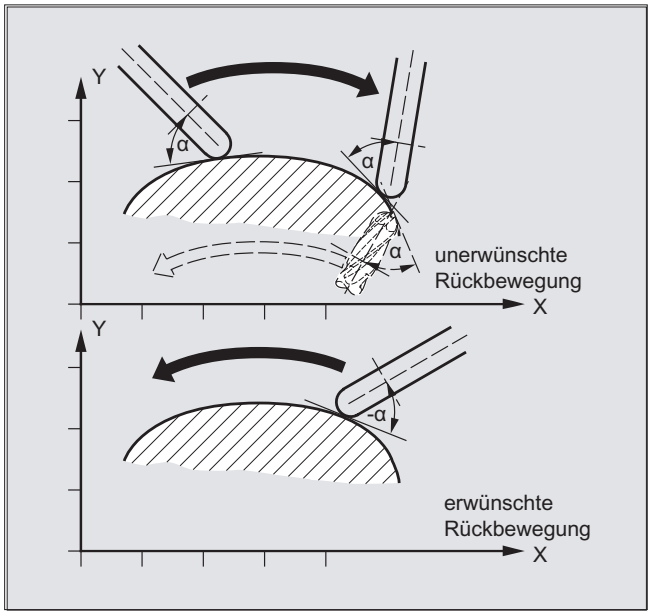
Folge- und Leitachse definieren

Die Definition von Folge- und Leitachsen erfolgt mit TANG.

Ein Koppelfaktor gibt den Zusammenhang zwischen einer Winkeländerung der Tangente und der nachgeführten Achse an. Sein Wert beträgt in der Regel 1 (Voreinstellung).

Grenzwinkel durch Arbeitsfeldbegrenzung

Bei hin- und hergeführten Bahnbewegungen springt die Tangente im Umkehrpunkt der Bahn um 180° um, entsprechend ändert sich die Ausrichtung der Folgeachse. In der Regel ist dieses Verhalten nicht sinnvoll: Die Rückbewegung soll im gleichen negativen Offsetwinkel wie die Hinbewegung abgefahren werden:



Dazu muss das Arbeitsfeld der Folgeachse begrenzt werden (G25, G26). Die Arbeitsfeldbegrenzung muss zum Zeitpunkt der Bahnumkehr aktiv sein (WALIMON). Liegt der Offsetwinkel außerhalb der Arbeitsfeldbegrenzung, wird versucht, mit negativem Offsetwinkel wieder in den zulässigen Arbeitsbereich zu kommen.

Zwischensatz an Konturecken einfügen (TLIFT)

An einer Ecke der Kontur ändert sich die Tangente und damit die Sollposition der nachgeführten Achse sprunghaft. Die Achse versucht normalerweise, diesen Sprung mit ihrer maximal möglichen Geschwindigkeit auszugleichen. Dabei ergibt sich jedoch über eine gewisse Strecke auf der Kontur nach der Ecke eine Abweichung zur gewünschten tangentiellen Anstellung. Wenn dies aus technologischen Gründen nicht tolerierbar ist, kann mit der Anweisung TLIFT die Steuerung dazu veranlasst werden, an der Ecke anzuhalten und in einem automatisch erzeugten Zwischensatz die nachgeführte Achse in die neue Tangentenrichtung zu drehen.

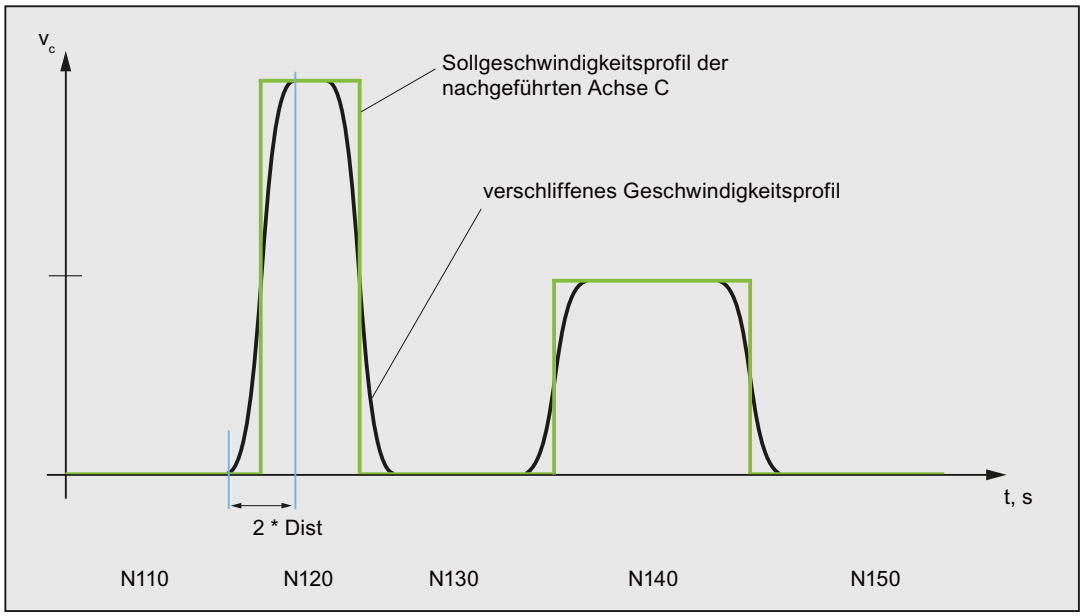
Die Drehung erfolgt mit der programmierten Bahnachse, wenn die nachgeführte Achse einmal als Bahnachse gefahren wurde. Durch die Funktion TFGREF[<Achse>]=0.001 kann hier eine maximale Achsgeschwindigkeit der nachgeführten Achse erreicht werden.

Wurde die nachgeführte Achse bisher nicht als Bahnachse verfahren, so wird diese Achse als Positionierachse verfahren. Die Geschwindigkeit ist dann abhängig von der im Maschinendatum hinterlegten Positioniergeschwindigkeit.

Die Drehung erfolgt mit der maximalen Geschwindigkeit der nachgeführten Achse.

Optimierungsmöglichkeit

Ist die automatische Optimierung angewählt (<Opt>="P") und sind für die Folgeachse die Parameter Überschleifweg (<Dist>) und Winkeltoleranz (<Winkeltoleranz>) angegeben, dann werden beim Tangentialen Nachführen Geschwindigkeitssprünge der Folgeachse infolge von Sprüngen in der Leitachskontur überschliffen bzw. geglättet. Dabei wird die Folgeachse vorausschauend geführt (siehe Diagramm), um die Abweichung möglichst klein zu halten.



Winkeländerung definieren

Die Winkeländerung, ab der ein automatischer Zwischensatz eingeführt wird, wird über das folgende Maschinendatum definiert:

MD37400 $MA_EPS_TLIFT_TANG_STEP (Tangentenwinkel für Eckenerkennung)

Einfluss auf Transformationen

Die Position der nachgeführten Rundachse kann Eingangswert für eine Transformation sein.

Explizite Positionierung der Folgeachse

Wird eine ihren Leitachsen nachgeführte Folgeachse explizit positioniert, so wirkt die Positionsangabe additiv zum programmierten Offsetwinkel.

Zulässig sind alle Wegvorgaben (Bahn- und Positionierachsbewegungen).

Status der Kopplung

Im NC-Teileprogramm kann der Status der Kopplung mit der Systemvariablen $AA_COUP_ACT[<Achse>] abgefragt werden:

Wert	Bedeutung
0	Keine Kopplung aktiv
1,2,3	Tangentiales Nachführen aktiv

8.2 Vorschubverlauf (FNORM, FLIN, FCUB, FPO)

Funktion

Zur flexibleren Vorgabe des Vorschubverlaufs wird die Vorschubprogrammierung nach DIN 66025 um lineare und kubische Verläufe erweitert.

Die kubischen Verläufe können direkt oder als interpolierende Splines programmiert werden. Hierdurch lassen sich - abhängig von der Krümmung des zu bearbeitenden Werkstücks - kontinuierlich glatte Geschwindigkeitsverläufe programmieren.

Diese Geschwindigkeitsverläufe ermöglichen ruckfreie Beschleunigungsänderungen und hierdurch Fertigung gleichmäßiger Werkstückoberflächen.

Syntax

```
F… FNORM
F… FLIN
F… FCUB
F=FPO(…,…,…)
```

Bedeutung

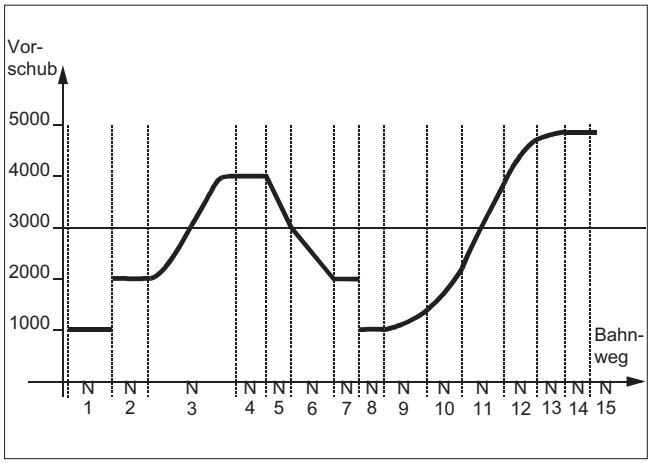
`FNORM`	Grundeinstellung. Der Vorschubwert wird über den Bahnweg des Satzes vorgegeben und gilt danach als modaler Wert.
`FLIN`	Bahngeschwindigkeitsprofil **linear:** Der Vorschubwert wird vom aktuellen Wert am Satzanfang bis zum Satzende über den Bahnweg linear eingefahren und gilt danach als modaler Wert. Dieses Verhalten kann mit G93 und G94 kombiniert werden.
`FCUB`	Bahngeschwindigkeitsprofil **kubisch:** Die satzweise programmierten F-Werte werden - bezogen auf den Satzendpunkt - durch einen Spline verbunden. Der Spline beginnt und endet tangential zur vorhergehenden bzw. nachfolgenden Vorschubangabe und wirkt mit G93 und G94. Fehlt in einem Satz die F-Adresse, so wird hierfür der zuletzt programmierte F-Wert verwendet.
`F=FPO…`	Bahngeschwindigkeitsprofil **über Polynom:** Die F-Adresse bezeichnet den Vorschubverlauf über ein Polynom vom aktuellen Wert bis zum Satzende. Der Endwert gilt danach als modaler Wert.

Vorschuboptimierung bei gekrümmten Bahnstücken

Vorschub-Polynom `F=FPO` und Vorschubspline `FCUB` sollten immer mit konstanter Schnittgeschwindigkeit `CFC` abgefahren werden. Hierdurch lässt sich ein beschleunigungsstetiges Sollvorschubprofil erzeugen.

Beispiel: Verschiedene Vorschubprofile

In diesem Beispiel finden Sie die Programmierung und grafische Darstellung verschiedener Vorschubprofile.

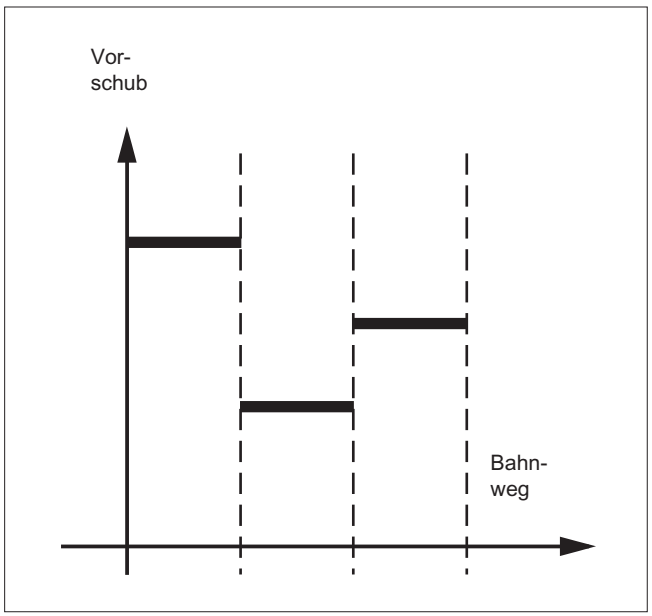


Programmcode	**Kommentar**
N1 F1000 FNORM G1 X8 G91 G64	; Konstantes Vorschubprofil, Kettenmaßangabe
N2 F2000 X7	; Sprunghafte Sollgeschwindigkeitsänderung
N3 F=FPO(4000, 6000, -4000)	; Vorschubprofil über Polynom mit Vorschub 4000 am Satzende
N4 X6	; Polynomvorschub 4000 gilt als modaler Wert
N5 F3000 FLIN X5	; Lineares Vorschubprofil
N6 F2000 X8	; Lineares Vorschubprofil
N7 X5	Linearer Vorschub gilt als modaler Wert
N8 F1000 FNORM X5	; Konstantes Vorschubprofil mit sprunghafter Beschleunigungsänderung
N9 F1400 FCUB X8	; Alle folgenden satzweise programmierten F-Werte werden mit Splines verbunden
N10 F2200 X6	
N11 F3900 X7	
N12 F4600 X7	
N13 F4900 X5	; Splineprofil ausschalten
N14 FNORM X5	
N15 X20	

FNORM

Die Vorschubadresse F bezeichnet den Bahnvorschub als konstanten Wert nach DIN 66025.

Mehr Informationen hierzu finden Sie im Programmierhandbuch "Grundlagen".

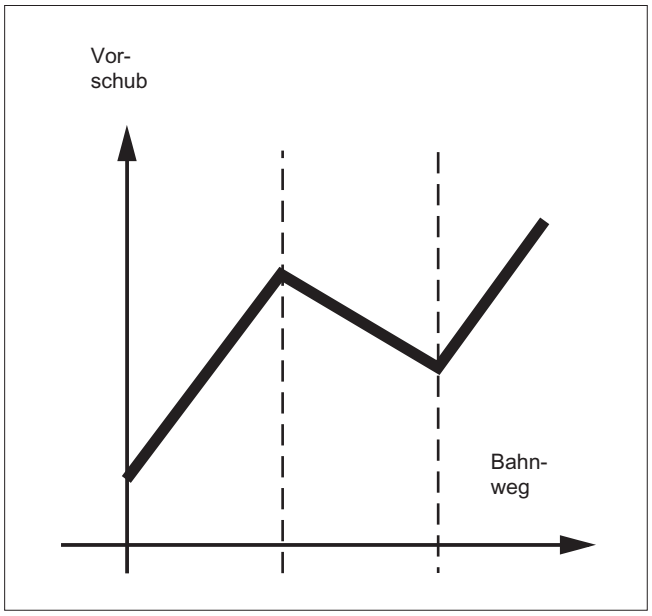


FLIN

Der Vorschubverlauf wird vom aktuellen Vorschubwert zum programmierten F-Wert linear bis Satzende eingefahren.

Beispiel:

```
N30 F1400 FLIN X50
```

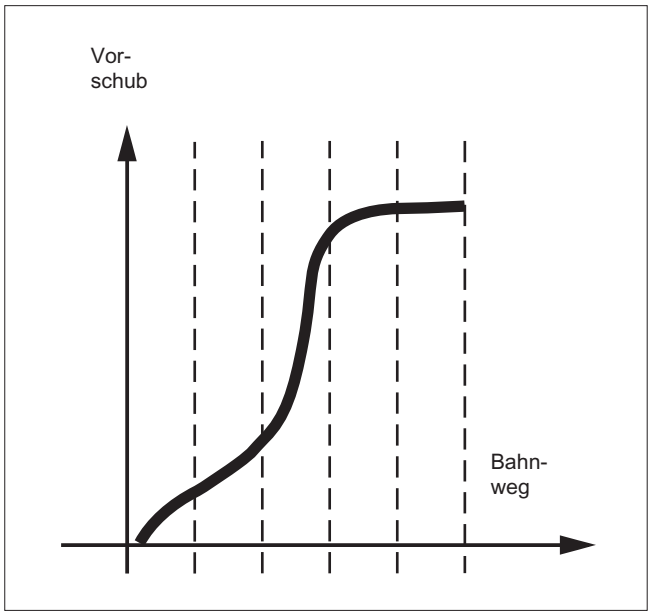


FCUB

Der Vorschub wird vom aktuellen Vorschubwert zum programmierten F-Wert bis Satzende im kubischen Verlauf eingefahren. Die Steuerung verbindet alle mit aktivem FCUB satzweise programmierten Vorschubwerte durch Splines. Die Vorschubwerte dienen hier als Stützpunkte zur Berechnung der Splineinterpolation.

Beispiel:

```
N50 F1400 FCUB X50
N60 F2000 X47
N70 F3800 X52
```



F=FPO(…,…,…)

Der Vorschubverlauf wird über ein Polynom direkt programmiert. Die Angabe der Polynomkoeffizienten erfolgt analog zur Polynominterpolation.

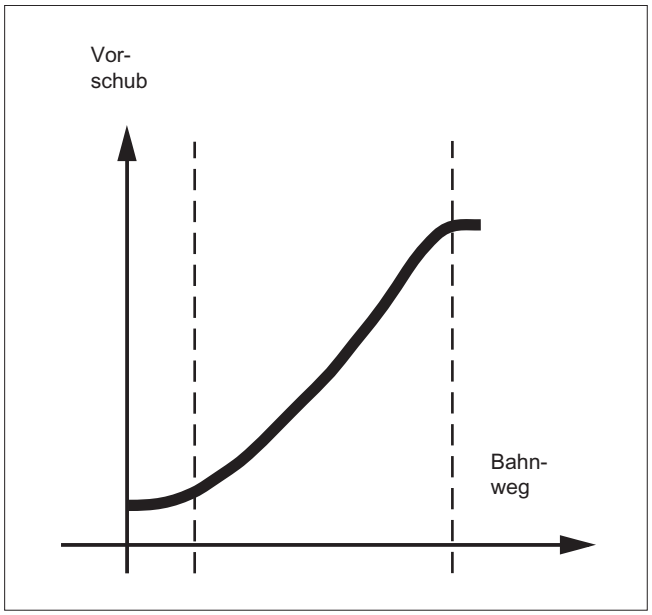
Beispiel:

```
F=FPO(endfeed, quadf, cubf)
```

endfeed, quadf und cubf sind vorher definierte Variable.

endfeed:	Vorschub am Satzende
quadf:	Quadratischer Polynomkoeffizient
cubf:	Kubischer Polynomkoeffizient

Bei aktivem `FCUB` schließt der Spline am Satzanfang und Satzende tangential an den über `FPO` festgelegten Verlauf an.



Randbedingungen

Unabhängig vom programmierten Vorschubverlauf gelten die Funktionen zur Programmierung des Bahnfahrverhaltens.

Der programmierbare Vorschubverlauf gilt grundsätzlich absolut - unabhängig von `G90` oder `G91`.

Der Vorschubverlauf FLIN und FCUB wirkt mit

`G93` und `G94`.

`FLIN` und `FCUB` **wirkt nicht bei**

`G95`, `G96/G961` und `G97/G971`.

Aktiver Kompressor COMPON

Bei aktivem Kompressor `COMPON` gilt bei Zusammenfassung mehrerer Sätze zu einem Splinesegment:

FNORM:

Für das Splinesegment gilt das F-Wort des letzten zugehörigen Satzes.

FLIN:

Für das Splinesegment gilt das F-Wort des letzten zugehörigen Satzes.
Der programmierte F-Wert gilt zum Ende des Segments und wird dann linear angefahren.

FCUB:

Der erzeugte Vorschubspline weicht maximal um den im Maschinendatum `$MC_COMPRESS_VELO_TOL` definierten Wert von den programmierten Endpunkten ab.

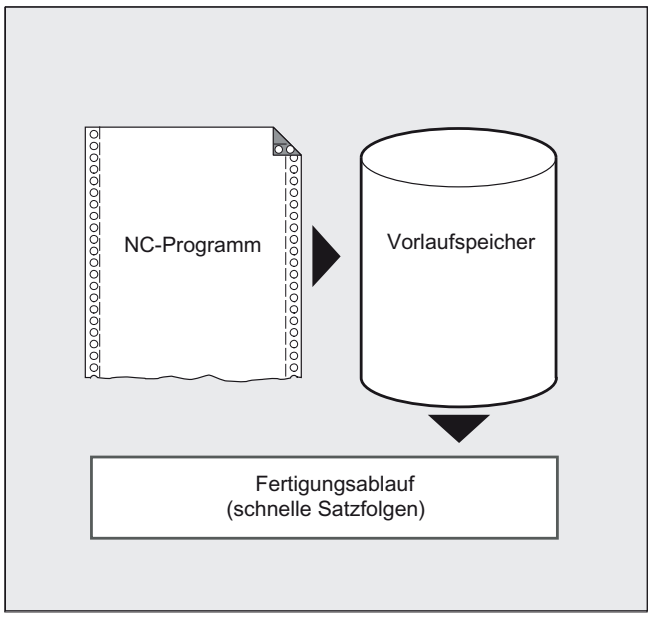
F=FPO(...,...,...)

Diese Sätze werden nicht komprimiert.

8.3 Programmablauf mit Vorlaufspeicher (STOPFIFO, STARTFIFO, FIFOCTRL, STOPRE)

Funktion

Je nach Ausbaustufe verfügt die Steuerung über eine bestimmte Menge sog. Vorlaufspeicher, die fertig aufbereitete Sätze vor der Abarbeitung speichern und im Fertigungsablauf als schnelle Satzfolgen ausgeben. Hierdurch lassen sich kurze Wege mit hohen Geschwindigkeiten abfahren. Soweit die Restzeit der Steuerung es zulässt, wird der Vorlaufspeicher grundsätzlich gefüllt.



Bearbeitungsabschnitt kennzeichnen

Der Bearbeitungsabschnitt, der im Vorlaufspeicher zwischengespeichert werden soll, wird im Teileprogramm am Anfang mit STOPFIFO und am Ende mit STARTFIFO gekennzeichnet. Die Abarbeitung der aufbereiteten und zwischengespeicherten Sätze beginnt erst nach dem Befehl STARTFIFO oder wenn der Vorlaufspeicher voll ist.

Automatische Vorlaufspeichersteuerung

Die automatische Vorlaufspeichersteuerung wird mit dem Befehl FIFOCTRL aufgerufen. FIFOCTRL wirkt zunächst genauso wie STOPFIFO. Bei jeder Programmierung wird gewartet, bis der Vorlaufspeicher voll ist, dann beginnt die Abarbeitung. Unterschiedlich ist dagegen das Verhalten beim Leerlaufen des Vorlaufspeichers: mit FIFOCTRL wird ab einem Füllstand von 2/3 die Bahngeschwindigkeit zunehmend reduziert, um ein komplettes Leerlaufen und ein Abbremsen bis zum Stillstand zu verhindern.

Vorlaufstopp

Die Satzaufbereitung und -zwischenspeicherung wird angehalten, wenn im Satz der Befehl STOPRE programmiert ist. Der nachfolgende Satz wird erst dann ausgeführt, wenn alle vorher aufbereiteten und gespeicherten Sätze vollständig abgearbeitet sind. Der vorherige Satz wird im Genauhalt angehalten (wie G9).

Syntax

Tabelle 8- 1 Bearbeitungsabschnitt kennzeichnen:

```
STOPFIFO
...
STARTFIFO
```

Tabelle 8- 2 Automatische Vorlaufspeichersteuerung:

```
...
FIFOCTRL
...
```

Tabelle 8- 3 Vorlaufstopp:

```
...
STOPRE
...
```

Hinweis

Die Befehle STOPFIFO, STARTFIFO, FIFOCTRL und STOPRE müssen im eigenen Satz programmiert werden.

Bedeutung

STOPFIFO:	STOPFIFO kennzeichnet den Beginn eines Bearbeitungsabschnitts, der im Vorlaufspeicher zwischengespeichert werden soll. Mit STOPFIFO wird die Bearbeitung angehalten und der Vorlaufspeicher gefüllt, bis: • STARTFIFO oder STOPRE erkannt wird oder • der Vorlaufspeicher voll ist oder • das Programmende erreicht ist.
STARTFIFO:	Mit STARTFIFO startet die schnelle Abarbeitung des Bearbeitungsabschnitts, parallel dazu erfolgt das Auffüllen des Vorlaufspeichers
FIFOCTRL:	Einschalten der automatischen Vorlaufspeichersteuerung
STOPRE:	Vorlauf stoppen

Hinweis

Das Auffüllen des Vorlaufspeichers wird nicht ausgeführt bzw. unterbrochen, wenn der Bearbeitungsabschnitt Befehle enthält, die einen ungepufferten Betrieb erzwingen (Referenzpunktfahren, Messfunktionen, …).

Hinweis

Beim Zugriff auf Zustandsdaten der Maschine ($SA...) erzeugt die Steuerung internen Vorlaufstopp.

⚠ VORSICHT

Bei eingeschalteter Werkzeugkorrektur und bei Spline-Interpolationen sollte kein `STOPRE` programmiert werden, da sonst zusammengehörige Satzfolgen unterbrochen werden.

Beispiel: Vorlauf stoppen

```
Programmcode                                    Kommentar
...
N30 MEAW=1 G1 F1000 X100 Y100 Z50               ; Messsatz mit Messtaster des ersten
                                                  Messeingangs und Geradeninterpolation.
N40 STOPRE                                      ; Vorlaufstopp.
...
```

8.4 Bedingt unterbrechbare Programmabschnitte (DELAYFSTON, DELAYFSTOF)

Funktion

Bedingt unterbrechbare Teileprogrammabschnitte werden Stop-Delay-Bereiche genannt Innerhalb bestimmter Programmabschnitte soll nicht **angehalten** werden und auch der **Vorschub** nicht verändert werden. Im Wesentlichen sollen kurze Programmabschnitte, die z. B. zur Herstellung eines Gewindes dienen, vor fast allen Stopp-Ereignissen geschützt werden. Ein etwaiger Stopp wirkt erst, nachdem der Programmabschnitt zu Ende bearbeitet worden ist.

Syntax

```
DELAYFSTON
DELAYFSTOF
```

Die Befehle stehen allein in einer Teileprogrammzeile.

Beide Befehle sind nur in Teileprogrammen, nicht jedoch in Synchronaktionen zulässig.

Bedeutung

`DELAYFSTON`	Beginn eines Bereiches definieren, in dem "sanfte" Stopps verzögert werden, bis das Ende des Stop-Delay-Bereiches erreicht wird.
`DELAYFSTOF`	Ende eines Stop-Delay-Bereiches definieren

Hinweis

Bei Maschinendatum MD11550 $MN_STOP_MODE_MASK Bit 0 = 0 (Default) wird ein Stop-Delay-Bereich implizit definiert, wenn `G331/G332` aktiv ist und eine Bahnbewegung bzw. `G4` programmiert ist.

Beispiel: Stopp-Ereignisse

Im Stop-Delay-Bereich wird eine Veränderung des **Vorschubs und Vorschubsperre** ignoriert. Sie wirken erst nach dem Stop-Delay-Bereich.

Die Stopp-Ereignisse werden unterschieden in:

"Sanfte" Stopp-Ereignisse	Reaktion: delayed
"Harte" Stopp-Ereignisse	Reaktion: immediate

Auswahl einiger Stopp-Ereignisse, die zumindest kurzfristig stoppen:

Ereignisname	Reaktion	Unterbrechungsparameter
RESET	immediate	NST: DB21,… DBX7.7 und DB11, … DBX20.7
PROG_END	Alarm 16954	NC-Prog: M30
INTERRUPT	delayed	NST: FC-9 und ASUP DB10, ... DBB1
SINGLEBLOCKSTOP	delayed	Einzelsatzbetrieb im Stopp-Delay-Bereich eingeschaltet: NC stoppt am Ende des 1.Satzes außerhalb des Stopp-Delay Bereichs. Einzelsatz bereits vor Stopp-Delay-Bereich angewählt: NST: "NC-Stop an der Satzgrenze" DB21, ... DBX7.2
STOPPROG	delayed	NST: DB21,… DBX7.3 und DB11, … DBX20.5
PROG_STOP	Alarm 16954	NC-Prog: M0 und M1
WAITM	Alarm 16954	NC-Prog: WAITM
WAITE	Alarm 16954	NC-Prog: WAITE
STOP_ALARM	immediat	Alarm: Alarmprojektierung STOPBYALARM
RETREAT_MOVE_THREAD	Alarm 16954	NC-Prog: Alarm 16954 bei LFON (Stopp & Fastlift im G33 nicht möglich)
WAITMC	Alarm 16954	NC-Prog: WAITMC
NEWCONF_PREP_STOP	Alarm 16954	NC-Prog: NEWCONF
SYSTEM_SHUTDOWN	immediate	System-Shutdown bei 840Di
ESR	delayed	Erweitertes Stillsetzen und Rückziehen
EXT_ZERO_POINT	delayed	Externe Nullpunktverschiebung
STOPRUN	Alarm 16955	BTSS: PI "_N_FINDST" STOPRUN

Erklärung der Reaktionen

immediate ("hartes" Stopp-Ereignis)	Stoppt sofort auch im Stopp-Delay-Bereich
delayed ("sanftes" Stopp-Ereignis)	Stopp (auch ein kurzfristiger) erfolgt erst nach dem Stopp-Delay-Bereich.
Alarm 16954	Programm wird abgebrochen, da im Stopp-Delay-Bereich unerlaubte Programmbefehle verwendet worden sind.
Alarm 16955	Programm wird fortgesetzt, im Stopp-Delay-Bereich hat eine unerlaubte Aktion stattgefunden hat.
Alarm 16957	Der Programmbereich (Stopp-Delay-Bereich), der durch DELAYFSTON und DELAYFSTOF geklammert ist, konnte nicht aktiviert werden. Damit wirkt jeder Stopp im Bereich sofort und wird nicht verzögert.

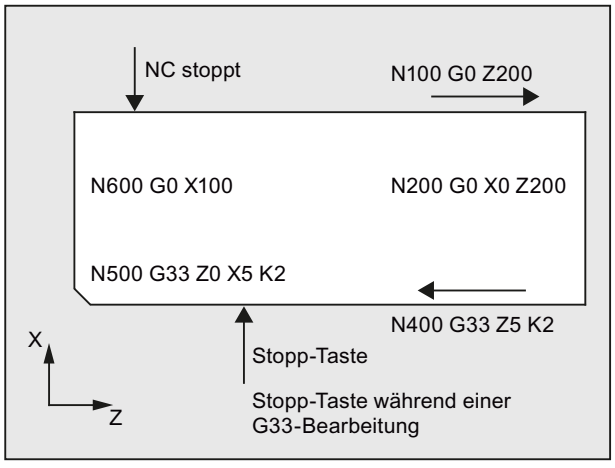
Eine Zusammenfassung weiterer Reaktionen auf Stopp-Ereignisse siehe:
Literatur:
Funktionshandbuch Grundfunktionen; BAG, Kanal, Programmbetrieb, (K1), Kapitel "Beeinflussung und Auswirkung auf Stopp-Ereignisse"

Beispiel: Verschachtelung von Stopp-Delay-Bereichen in zwei Programmebenen

```
Programmcode                    Kommentar
N10010 DELAYFSTON()             ; Sätze mit N10xxx Programmebene 1.
N10020 R1 = R1 + 1
N10030 G4 F1                    ; Stop-Delay-Bereich beginnt.
...
N10040 Unterprogramm2
...
...                             ; Interpretation des Unterprogramms 2.
N20010 DELAYFSTON()             ; Unwirksam, wiederholter Beginn, 2. Ebene.
...
N20020 DELAYFSTOF()             ; Unwirksam, Ende in anderer Ebene.
N20030 RET
N10050 DELAYFSTOF()             ; Stop-Delay-Bereichs-Ende in gleicher Ebene.
...
N10060 R2 = R2 + 2
N10070 G4 F1                    ; Stop-Delay-Bereich endet. Stopps wirken ab jetzt
                                  unmittelbar.
```

Beispiel: Programmauszug

In einer Schleife wird folgender Programmblock wiederholt:



Im Bild ist erkennbar, dass der Anwender im Stop-Delay-Bereich "Stopp" drückt, und die NC beginnt den Bremsvorgang außerhalb des Stop-Delay-Bereichs, d. h. im Satz `N100`. Damit kommt die NC im vorderen Bereich von `N100` zum Halten.

Programmcode	Kommentar
...	
N99 MY_LOOP:	
N100 G0 Z200	
N200 G0 X0 Z200	
N300 DELAYFSTON()	
N400 G33 Z5 K2 M3 S1000	
N500 G33 Z0 X5 K3	
N600 G0 X100	
N700 DELAYFSTOF()	
N800 GOTOB MY_LOOP	

Details über Satzsuchlauf vom Typ SERUPRO und Vorschübe in Verbindung mit `G331/G332` Vorschub bei Gewindebohren ohne Ausgleichsfutter siehe:
Literatur:
Funktionshandbuch Grundfunktionen; BAG, Kanal, Programmbetrieb (K1)
Funktionshandbuch Grundfunktionen; Vorschübe (V1)

Vorteile des Stopp-Delay-Bereiches

Ein Programmabschnitt wird ohne Geschwindigkeitseinbruch bearbeitet.

Bricht der Anwender, nachdem gestoppt ist, das Programm mit RESET ab, so ist der abgebrochene Programmsatz nach dem geschützten Bereich. Dieser Programmsatz eignet sich dann als Suchziel für einen nachfolgenden Suchlauf.

Solange ein Stopp-Delay-Bereich bearbeitet wird werden folgende Hauptlaufachsen nicht gestoppt:

- Kommandoachsen und
- Positionierachsen die mit `POSA` verfahren

Der Teileprogrammbefehl `G4` ist im Stopp-Delay-Bereich zulässig, dagegen sind andere Teileprogrammbefehle, die zu einem vorübergehenden Stopp führen (z. B `WAITM`) nicht zulässig.

`G4` macht, wie eine Bahnbewegung, den Stopp-Delay-Bereich wirksam bzw. hält seine Wirksamkeit aufrecht.

Beispiel: Vorschub Eingriffe

Wird der Override vor dem Stopp-Delay-Bereich auf 6% gesenkt, so wird der Override im Stopp-Delay-Bereich wirksam.

Wird der Override im Stopp-Delay-Bereich von 100% auf 6% abgesenkt, so wird der Stopp-Delay-Bereich mit 100% zu Ende gefahren und danach wird mit 6% weitergefahren.

Die Vorschubsperre wirkt im Stopp-Delay-Bereich nicht, erst nach dem Verlassen des Stopp-Delay-Bereichs wird angehalten.

Überlappung/Schachtelung:

Überschneiden sich zwei Stopp-Delay-Bereiche, einer aus den Sprachbefehlen und der andere aus dem Maschinendatum MD 11550: STOP_MODE_MASK, so wird der größtmögliche Stopp-Delay-Bereich gebildet.

Folgende Punkte regeln das Zusammenspiel der Sprachbefehle DELAYFSTON und DELAYFSTOF mit Verschachtelungen und dem Unterprogrammende:

1. Mit dem Ende des Unterprogramms, in dem DELAYFSTON gerufen wurde, wird implizit DELAYFSTOF aktiviert.
2. DELAYFSTON Stopp-Delay-Bereich bleibt ohne Wirkung.
3. Ruft Unterprogramm1 in einem Stopp-Delay-Bereich Unterprogramm2, so ist Unterprogramm2 komplett ein Stopp-Delay-Bereich. Insbesondere ist DELAYFSTOF in Unterprogramm2 wirkungslos.

Hinweis

REPOSA ist ein Unterprogrammende und DELAYFSTON wird in jedem Fall abgewählt.

Trifft ein "hartes" Stopp-Ereignis auf den "Stopp-Delay-Bereich", so wird der "Stopp-Delay-Bereich" komplett abgewählt! Das heißt, tritt in diesem Programmabschnitt ein weiterer beliebiger Stopp auf, so wird sofort angehalten. Erst eine Neuprogrammierung (erneutes DELAYFSTON) lässt einen neuen Stopp-Delay-Bereich beginnen.

Wird die Stopp-Taste vor dem Stopp-Delay-Bereich gedrückt und der NCK muss zum Bremsen in den Stopp-Delay-Bereich einfahren, so stoppt der NCK im Stopp-Delay-Bereich und der Stopp-Delay-Bereich bleibt abgewählt!

Wird ein Stopp-Delay-Bereich mit Override 0% betreten, so wird der Stopp-Delay-Bereich **nicht** akzeptiert!

Dies gilt für alle "sanften" Stopp-Ereignisse.

Mit STOPALL kann im Stopp-Delay-Bereich gebremst werden. Mit einem STOPALL werden aber alle anderen Stopp-Ereignisse sofort aktiv, die bislang verzögert worden sind.

Systemvariablen

Ein Stopp-Delay-Bereich kann mit $P_DELAYFST im Teileprogramm erkannt werden. Ist Bit 0 der Systemvariablen auf 1 gesetzt, so befindet sich die Teileprogrammbearbeitung zu diesem Zeitpunkt in einem Stopp-Delay-Bereich.

Ein Stopp-Delay-Bereich kann mit $AC_DELAYFST in Synchronaktionen erkannt werden. Ist Bit 0 der Systemvariablen auf 1 gesetzt, so befindet sich die Teileprogrammbearbeitung zu diesem Zeitpunkt in einem Stopp-Delay-Bereich.

Kompatibilität

Die Vorbesetzung des Maschinendatums MD 11550: STOP_MODE_MASK Bit 0 = 0 bewirkt impliziten Stopp-Delay-Bereich während der G-Code-Gruppe G331/G332 und wenn eine Bahnbewegung bzw. G4 programmiert ist.

Bit 0 = 1 ermöglicht Stopp während der G-Code-Gruppe G331/G332 und wenn eine Bahnbewegung bzw. G4 programmiert ist (Verhalten bis SW 6). Zur Definition eines Stopp-Delay-Bereiches müssen die Befehle DELAYFSTON/DELAYFSTOF benutzt werden.

8.5 Programmstelle für SERUPRO verhindern (IPTRLOCK, IPTRUNLOCK)

Funktion

Für bestimmte komplizierte mechanische Situationen an der Maschine ist es erforderlich, den Satzsuchlauf SERUPRO zu verhindern.

Mit einem programmierbaren Unterbrechungszeiger besteht eine Eingriffsmöglichkeit, beim "Suchen auf der Unterbrechungsstelle", vor der suchunfähigen Stelle aufzusetzen.

Es können auch suchunfähige Bereiche in Teileprogrammbereichen definiert werden, in denen die NCK noch nicht wieder einsteigen kann. Mit dem Programmabbruch vermerkt der NCK den zuletzt verarbeiteten Satz, auf den über die Bedienoberfläche HMI gesucht werden kann.

Syntax

```
IPTRLOCK
IPTRUNLOCK
```

Die Befehle stehen allein in einer Teleprogrammzeile und ermöglichen einen programmierbaren Unterbrechungszeiger

Bedeutung

`IPTRLOCK`	Beginn des suchunfähigen Programmabschnitts
`IPTRUNLOCK`	Ende des suchunfähigen Programmabschnitts

Beide Befehle sind nur in Teileprogrammen, **nicht** jedoch in Synchronaktionen zulässig.

Beispiel

Verschachtelung suchunfähiger Programmabschnitte in zwei Programmebenen mit impliziten IPTRUNLOCK. Das implizite IPTRUNLOCK in Unterprogramm 1 beendet den suchunfähigen Bereich.

```
Programmcode                    Kommentar
N10010 IPTRLOCK()
N10020 R1 = R1 + 1
N10030 G4 F1                    ; Haltesatz, der suchunfähige Programmabschnitt
                                  beginnt.
...
N10040 Unterprogramm2
...                             ; Interpretation des Unterprogramms 2.
N20010 IPTRLOCK ()              ; Unwirksam, wiederholter Beginn.
...
N20020 IPTRUNLOCK ()            ; Unwirksam, Ende in anderer Ebene.
N20030 RET
...
N10060 R2 = R2 + 2
N10070 RET                      ; Ende des suchunfähigen Programmabschnitts.
N100 G4 F2                      ; Hauptprogramm wird fortgesetzt.
```

Eine Unterbrechung auf 100 liefert dann wieder der Unterbrechungszeiger.

Suchunfähige Bereiche erfassen und suchen

Die suchunfähigen Programmabschnitte werden mit dem Sprachbefehlen IPTRLOCK und IPTRUNLOCK gekennzeichnet.

Der Befehl IPTRLOCK friert den Unterbrechungszeiger auf ein im Hauptlauf ausführbaren Einzelsatz (SBL1) ein. Dieser Satz wird im Folgenden als Haltesatz bezeichnet. Tritt nach IPTRLOCK ein Programmabbruch ein, so kann auf der Bedienoberfläche HMI nach diesen sogenannten Haltesatz gesucht werden.

Auf den aktuellen Satz wieder aufsetzen

Der Unterbrechungszeiger wird mit IPTRUNLOCK für den nachfolgenden Programmabschnitt auf den aktuellen Satz zum Unterbrechungspunkt gesetzt werden.

Nach einem gefundenen Suchziel kann mit dem selben Haltesatz ein neues Suchziel wiederholt werden.

Ein vom Benutzer editierter Unterbrechungszeiger, muss über HMI wieder entfernt werden.

Regeln bei Schachtelung

Folgende Punkte regeln das Zusammenspiel der Sprachbefehle `IPTRLOCK` und `IPTRUNLOCK` mit Verschachtelungen und dem Unterprogrammende:

1. Mit dem Ende des Unterprogramms, in dem `IPTRLOCK` gerufen wurde, wird implizit `IPTRUNLOCK` aktiviert.
2. `IPTRLOCK` in einem suchunfähigen Bereich bleibt ohne Wirkung.
3. Ruft Unterprogramm1 in einem suchunfähigen Bereich Unterprogramm2, so bleibt Unterprogramm2 komplett suchunfähig. Insbesondere ist `IPTRUNLOCK` in Unterprogramm2 wirkungslos.

Weitere Informationen hierzu siehe
/FB1/ Funktionshandbuch Grundfunktionen; BAG, Kanal, Programmbetrieb (K1).

Systemvariable

Ein suchunfähiger Bereich kann mit `$P_IPTRLOCK` im Teileprogramm erkannt werden.

Automatischer Unterbrechungszeiger

Die Funktion automatischer Unterbrechungszeiger legt automatisch eine vorher festgelegte Kopplungsart als suchunfähig fest. Mittels Maschinendatum wird für

- Elektronisches Getriebe bei `EGON`
- Axiale Leitwertkopplung bei `LEADON`

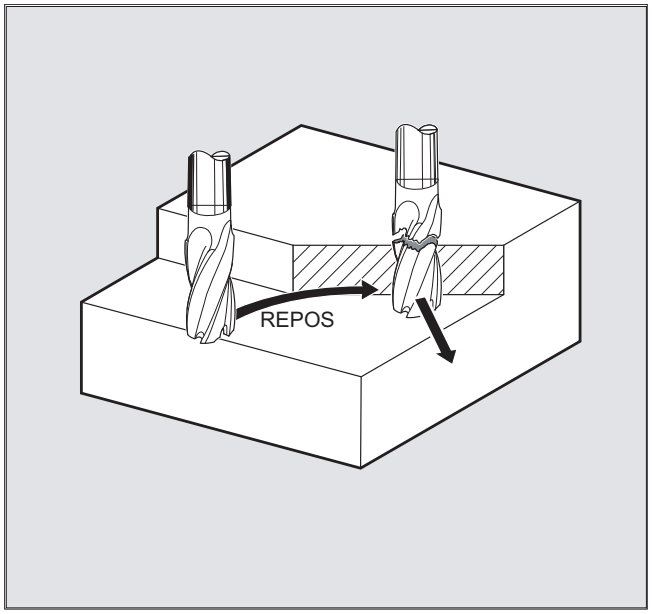
der automatische Unterbrechungszeiger aktiviert. Überschneiden sich der programmierte und der über Maschinendatum aktivierbare automatische Unterbrechungszeiger, so wird der größtmögliche suchunfähige Bereich gebildet.

8.6 Wiederanfahren an Kontur (REPOSA, REPOSL, REPOSQ, REPOSQA, REPOSH, REPOSHA, DISR, DISPR, RMI, RMB, RME, RMN)

Funktion

Wenn Sie während der Bearbeitung das laufende Programm unterbrechen und das Werkzeug freifahren – zum Beispiel wegen Werkzeugbruch oder weil Sie etwas nachmessen wollen – können Sie die Kontur an einem wählbaren Punkt programmgesteuert wiederanfahren.

Der REPOS-Befehl wirkt wie ein Unterprogramm-Rücksprung (z. B. über M17). Nachfolgende Sätze in der Interruptroutine werden nicht mehr ausgeführt.



Zur Unterbrechung des Programmlaufs siehe auch Abschnitt "Flexible NC-Programmierung" Kapitel "Interruptroutine" in diesem Programmierhandbuch.

Syntax

```
REPOSA RMI DISPR=…
REPOSA RMB
REPOSA RME
REPOSA RMN
REPOSL RMI DISPR=…
REPOSL RMB
REPOSL RME
REPOSL RMN
REPOSQ RMI DISPR=… DISR=…
REPOSQ RMB DISR=…
REPOSQ RME DISR=…
REPOSQA DISR=…
REPOSH RMI DISPR=… DISR=…
REPOSH RMB DISR=…
REPOSH RME DISR=…
REPOSHA DISR=…
```

Bedeutung

Anfahrweg

`REPOSA`	Anfahren auf einer Geraden mit allen Achsen
`REPOSL`	Anfahren auf einer Geraden
`REPOSQ DISR=…`	Anfahren auf einem Viertelkreis mit Radius DISR
`REPOSQA DISR=…`	Anfahren auf allen Achsen auf einem Viertelkreis mit Radius DISR
`REPOSH DISR=…`	Anfahren auf einem Halbkreis mit Durchmesser DISR
`REPOSHA DISR=…`	Anfahren auf allen Achsen auf einem Halbkreis mit Radius DISR

Wiederanfahrpunkt

`RMI`	Unterbrechungspunkt anfahren
`RMI DISPR=…`	Eintrittspunkt im Abstand DISPR in mm/inch vor Unterbrechungspunkt
`RMB`	Satzanfangspunkt anfahren
`RME`	Satzendpunkt anfahren
`RME DISPR=…`	Satzendpunkt anfahren im Abstand DISPR vor Endpunkt
`RMN`	An den nächstliegenden Bahnpunkt anfahren
`A0 B0 C0`	Achsen, in denen angefahren werden soll

Beispiel: Anfahren auf einer Geraden anfahren, REPOSA, REPOSL

Das Werkzeug fährt den Wiederanfahrpunkt direkt auf einer Geraden an.

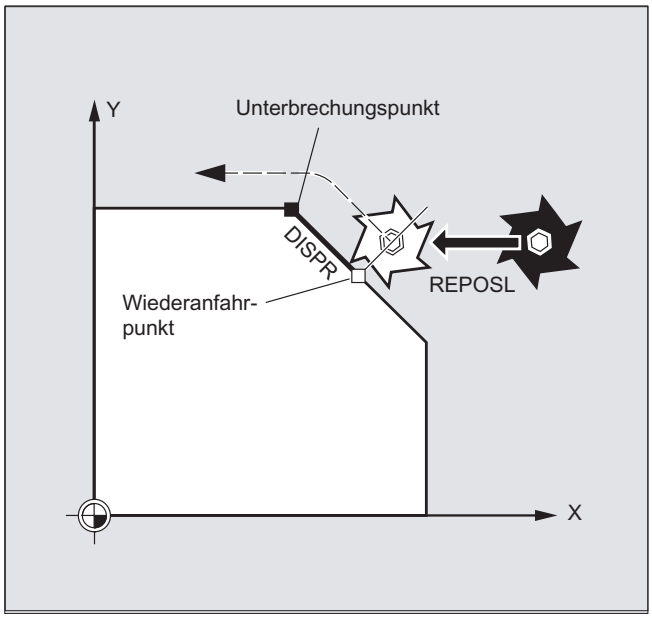
Mit REPOSA werden automatisch alle Achsen verfahren. Bei REPOSL können Sie die zu verfahrenden Achsen angeben.

Beispiel:

```
REPOSL RMI DISPR=6 F400
```

oder

```
REPOSA RMI DISPR=6 F400
```

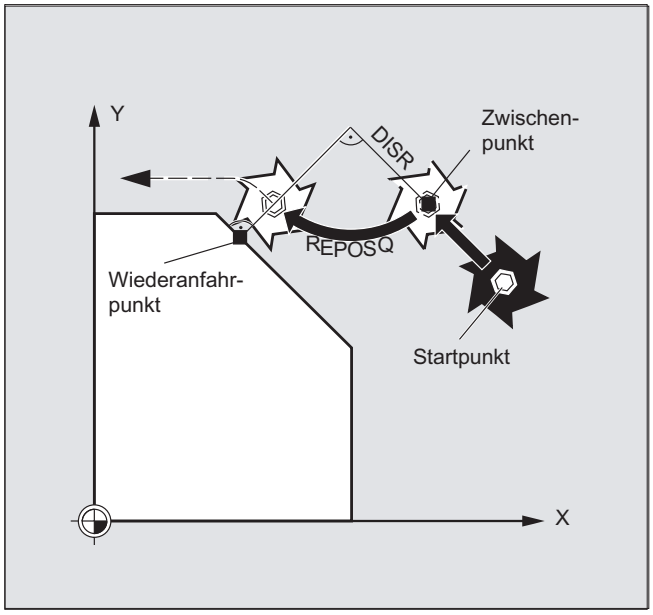


Beispiel: Anfahren im Viertelkreis anfahren, REPOSQ, REPOSQA

Das Werkzeug fährt den Wiederanfahrpunkt auf einem Viertelkreis mit Radius `DISR=...` an. Den notwendigen Zwischenpunkt zwischen Start- und Wiederanfahrpunkt berechnet die Steuerung automatisch.

Beispiel:

```
REPOSQ RMI DISR=10 F400
```

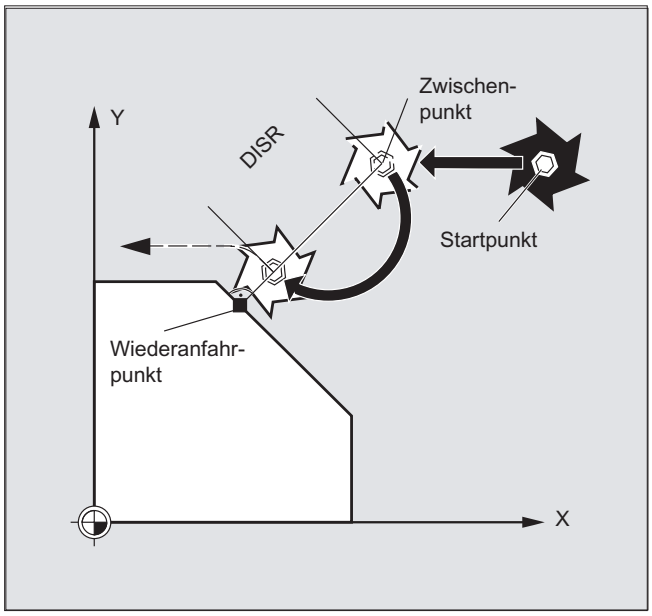


Beispiel: Werkzeug im Halbkreis anfahren, REPOSH, REPOSHA

Das Werkzeug fährt den Wiederanfahrpunkt auf einem Halbkreis mit Durchmesser `DISR=...` an. Den notwendigen Zwischenpunkt zwischen Start- und Wiederanfahrpunkt berechnet die Steuerung automatisch.

Beispiel:

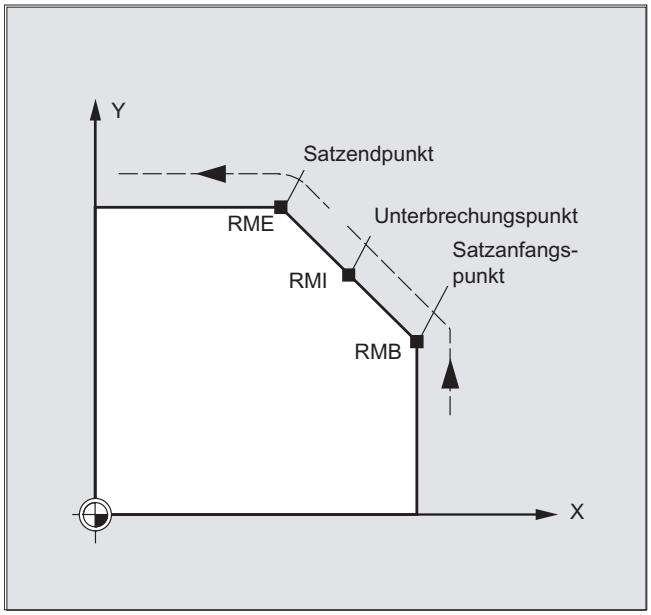
```
REPOSH RMI DISR=20 F400
```



Wiederanfahrpunkt festlegen (nicht für SERUPRO Anfahren mit RMN)

Bezogen auf den NC-Satz, in dem der Programm-Ablauf unterbrochen wurde, können Sie zwischen drei Wiederanfahrpunkten wählen:

- RMI, Unterbrechungspunkt
- RMB, Satzanfangspunkt bzw. letzter Endpunkt
- RME, Satzendpunkt



Mit `RMI DISPR=...` bzw. mit `RME DISPR=...` können Sie einen Wiederanfahrpunkt festlegen, der vor dem Unterbrechungspunkt bzw. vor dem Satzendpunkt liegt.

Mit `DISPR=...` beschreiben Sie den Konturweg in mm/inch, um den der Wiederanfahrpunkt **vor** dem Unterbrechungs- bzw. Endpunkt liegt. Dieser Punkt kann - auch für größere Werte - maximal im Satzanfangspunkt liegen.

Wird kein `DISPR=...` programmiert, gilt `DISPR=0` und damit der Unterbrechungspunkt (bei `RMI`) bzw. der Satzendpunkt (bei `RME`).

Vorzeichen von DISPR

Das Vorzeichen von DISPR wird ausgewertet. Bei positivem Vorzeichen ist das Verhalten wie bisher.

Bei negativem Vorzeichen wird hinter dem Unterbrechungspunkt bzw. bei RMB hinter dem Startpunkt wieder aufgesetzt.

Der Abstand Unterbrechungspunkt-Aufsetzpunkt ergibt sich aus dem Betrag von DISPR. Dieser Punkt kann auch für betragsmäßig größere Werte maximal im Satzendpunkt liegen.

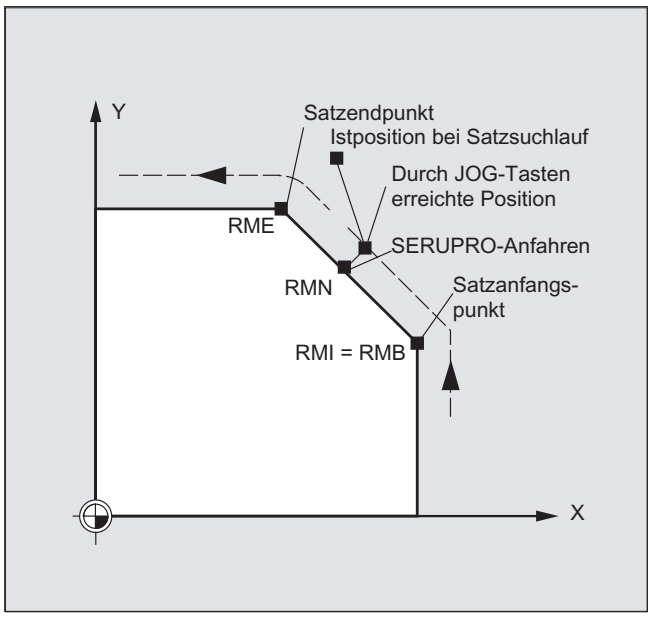
Anwendungsbeispiel:

Durch einen Sensor wird die Annäherung an eine Spannpratze erkannt. Es wird ein ASUP ausgelöst, mit dem die Spannpratze umfahren wird.

Anschließend wird mit negativem DISPR auf einen Punkt hinter der Spannpratze repositioniert und das Programm fortgesetzt.

SERUPRO-Anfahren mit RMN

Wird bei der Bearbeitung an einer beliebigen Stelle ein Abbruch erzwungen, dann wird mit SERUPRO-Anfahren unter RMN der kürzeste Weg von der Abbruchstelle angefahren, um anschließend nur den Restweg abzuarbeiten. Dazu startet der Anwender ein SERUPRO-Vorgang auf den Unterbrechungssatz und positioniert mit den JOG-Tasten vor die schadhafte Stelle des Zielsatzes.



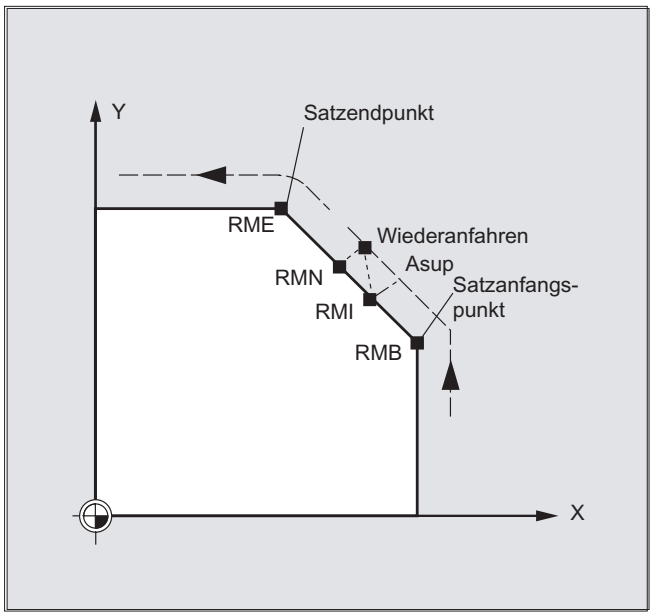
Hinweis

SERUPRO

Für SERUPRO ist RMI und RMB identisch. RMN ist nicht nur auf SERUPRO beschränkt, sondern allgemein gültig.

Anfahren vom nächstliegenden Bahnpunkt RMN

Zum Interpretationszeitpunkt von `REPOSA` wird nach einer Unterbrechung der Wiederanfahrsatz mit `RMN` nicht noch einmal komplett begonnen, sondern nur der Restweg abgearbeitet. Es wird der nächstliegende Bahnpunkt des unterbrochenen Satzes angefahren.



Status für den gültigen REPOS-Mode

Der gültige REPOS-Mode des unterbrochenen Satzes kann über Synchronaktionen mit der Variablen `$AC_REPOS_PATH_MODE` gelesen werden:

0: Anfahren nicht definiert

1 `RMB`: Anfahren auf den Beginn

2 `RMI`: Anfahren auf den Unterbrechungspunkt

3 `RME`: Anfahren auf den Satzendpunkt

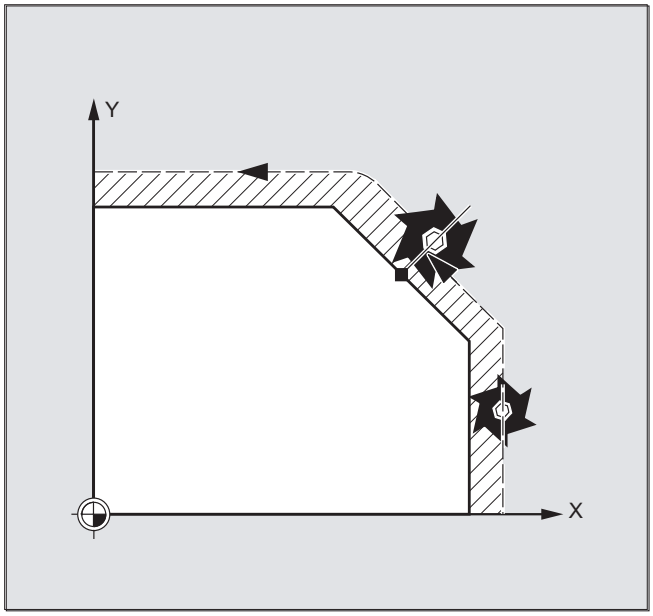
4 `RMN`: Anfahren auf den nächstliegenden Bahnpunkt des unterbrochenen Satzes.

Anfahren mit neuem Werkzeug

Falls Sie den Programmablauf wegen Werkzeugbruch gestoppt haben:

Mit Programmierung der neuen D-Nummer wird das Programm ab Wiederanfahrpunkt mit den geänderten Werkzeugkorrekturwerten fortgesetzt.

Bei geänderten Werkzeugkorrekturwerten kann der Unterbrechungspunkt möglicherweise nicht mehr angefahren werden. In diesem Fall wird der dem Unterbrechungspunkt nächstgelegene Punkt auf der neuen Kontur angefahren (gegebenenfalls um DISPR modifiziert).



Kontur anfahren

Die Bewegung, mit der das Werkzeug wieder an die Kontur heranfährt, ist programmierbar. Die Adressen der zu verfahrenden Achsen geben Sie mit Wert Null an.

Mit den Befehlen `REPOSA`, `REPOSQA` und `REPOSHA` werden automatisch alle Achsen repositioniert. Es ist keine Achsangabe notwendig.

Bei Programmierung von `REPOSL`, `REPOSQ` und `REPOSH` fahren alle Geometrieachsen automatisch, also auch ohne Angabe im Befehl, an. Alle anderen Achsen müssen im Befehl angegeben werden.

Für die Kreisbewegungen REPOSH und REPOSQ gilt:

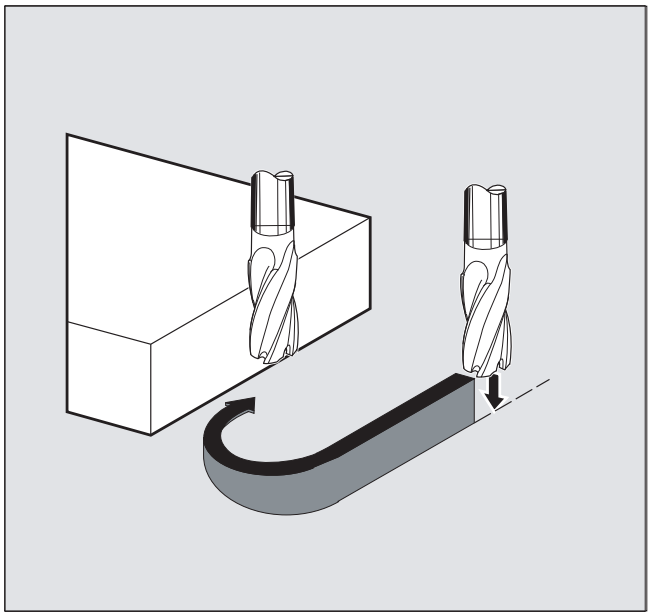
Der Kreis wird in der angegebenen Arbeitsebene `G17` bis `G19` gefahren.

Falls Sie im Anfahrsatz die dritte Geometrieachse (Zustellrichtung) angeben, wird der Wiederanfahrpunkt für den Fall, dass Werkzeugposition und programmierte Position in Zustellrichtung nicht übereinstimmen, auf einer Schraubenlinie angefahren.

In folgenden Fällen wird automatisch auf

lineares Anfahren `REPOSL` umgeschaltet:

- Sie haben keinen Wert für `DISR` angegeben.
- Es gibt keine definierte Anfahrrichtung (Programmunterbrechung in einem Satz ohne Verfahrinformation).
- Bei Anfahrrichtung senkrecht zur aktuellen Arbeitsebene.



8.7 Beeinflussung der Bewegungsführung

8.7.1 Prozentuale Ruckkorrektur (JERKLIM)

Funktion

Mit dem NC-Befehl `JERKLIM` kann der per Maschinendatum eingestellte maximal mögliche Ruck einer Achse bei Bahnbewegung in kritischen Programmabschnitten reduziert oder überhöht werden.

Voraussetzung

Der Beschleunigungsmodus SOFT muss aktiv sein.

Wirksamkeit

Die Funktion wirkt:

- in den AUTOMATIK-Betriebsarten.
- nur auf Bahnachsen.

Syntax

```
JERKLIM[<Achse>]=<Wert>
```

Bedeutung

`JERKLIM`:	Befehl zur Ruckkorrektur
`<Achse>`:	Maschinenachse, deren Ruckgrenzwert angepasst werden soll.
`<Wert>`:	Prozentualer Korrekturwert, bezogen auf den projektierten maximalen Achsruck bei Bahnbewegung (MD32431 $MA_MAX_AX_JERK). Wertebereich: 1 ... 200 Der Wert 100 bewirkt keine Beeinflussung des Rucks.

Hinweis

Das Verhalten von JERKLIM bei Teileprogrammende und Kanal-Reset wird projektiert mit Bit 0 im Maschinendatum MD32320 $MA_DYN_LIMIT_RESET_MASK:

- Bit 0 = 0:
  Der programmierte Wert für JERKLIM wird mit Kanal-Reset/M30 auf 100 % zurückgesetzt.
- Bit 0 = 1:
  Der programmierte Wert für JERKLIM bleibt über Kanal-Reset/M30 hinaus erhalten.

Beispiel

```
Programmcode                    Kommentar
...
N60 JERKLIM[X]=75               ; Der Achsschlitten in X-Richtung soll nur mit maximal
                                  75% des für die Achse zulässigen Rucks
                                  beschleunigt/verzögert werden.
...
```

8.7.2 Prozentuale Geschwindigkeitskorrektur (VELOLIM)

Funktion

Mit dem NC-Befehl `VELOLIM` kann die per Maschinendatum eingestellte maximal mögliche Geschwindigkeit einer Achse/Spindel im Achsbetrieb bzw. die maximal mögliche getriebestufenabhängige Drehzahl einer Spindel im Spindelbetrieb (Drehzahlsteuerbetrieb M3, M4, M5 und Positionierbetrieb SPOS, SPOSA, M19) in kritischen Programmabschnitten reduziert werden, z. B. um die Maschinenbeanspruchung zu verringern oder die Bearbeitungsgüte zu verbessern.

Wirksamkeit

Die Funktion wirkt:

- in den AUTOMATIK-Betriebsarten.
- auf Bahn- und Positionierachsen.
- auf Spindeln im Spindel-/Achsbetrieb

Syntax

```
VELOLIM[<Achse/Spindel>]=<Wert>
```

Bedeutung

VELOLIM:	Befehl zur Geschwindigkeitskorrektur
<Achse/Spindel>:	Maschinenachse oder Spindel, deren Geschwindigkeits- oder Drehzahlgrenzwert angepasst werden soll. **VELOLIM für Spindeln** Über Maschinendatum (MD30455 $MA_MISC_FUNCTION_MASK, Bit 6) kann für die Programmierung im Teileprogramm eingestellt werden, ob VELOLIM unabhängig von der aktuellen Verwendung als Spindel oder Achse wirkt (Bit 6 = 1) oder getrennt für jede Betriebsart programmierbar sein soll (Bit 6 = 0). Ist eine getrennte Wirkung projektiert, dann wird die Auswahl über den Bezeichner bei der Programmierung getroffen: • Spindelbezeichner S<n> für Spindelbetriebsarten • Achsbezeichner, z. B. "C", für den Achsbetrieb
<Wert>:	Prozentualer Korrekturwert Der Korrekturwert bezieht sich: • bei Achsen / Spindeln im Achsbetrieb (wenn MD30455 Bit 6 = 0): auf die projektierte maximale Achsgeschwindigkeit (MD32000 $MA_MAX_AX_VELO). • bei Spindeln im Spindel- oder Achsbetrieb (wenn MD30455 Bit 6 = 1): auf die Maximaldrehzahl der aktiven Getriebestufe (MD35130 $MA_GEAR_STEP_MAX_VELO_LIMIT[<n>]) Wertebereich: 1 ... 100 Der Wert 100 bewirkt keine Beeinflussung der Geschwindigkeit bzw. Drehzahl.

Hinweis

Verhalten bei Teileprogrammende und Kanal-Reset

Das Verhalten von VELOLIM bei Teileprogrammende und Kanal-Reset wird projektiert mit Bit 0 im Maschinendatum MD32320 $MA_DYN_LIMIT_RESET_MASK:

- Bit 0 = 0:
  Der programmierte Wert für VELOLIM wird mit Kanal-Reset/M30 auf 100 % zurückgesetzt.
- Bit 0 = 1:
  Der programmierte Wert für VELOLIM bleibt über Kanal-Reset/M30 hinaus erhalten.

Hinweis

VELOLIM für Spindeln in Synchronaktionen

Bei der Programmierung von VELOLIM in Synchronaktionen wird nicht zwischen Spindel- und Achsbetrieb unterschieden. Unabhängig von dem bei der Programmierung verwendeten Bezeichner werden die Drehzahl im Spindelbetrieb und die Geschwindigkeit im Achsbetrieb gleichermaßen limitiert.

Diagnose

Diagnose von VELOLIM im Spindelbetrieb

Eine aktive Drehzahlbegrenzung durch VELOLIM (kleiner 100 %) kann im Spindelbetrieb durch das Lesen der Systemvariablen $AC_SMAXVELO und $AC_SMAXVELO_INFO erkannt werden.

Im Falle einer Begrenzung liefert $AC_SMAXVELO das durch VELOLIM erzeugte Drehzahllimit. Die Variable $AC_SMAXVELO_INFO gibt in diesem Fall den Wert "16" als Kennung für die Begrenzungsursache VELOLIM zurück.

Beispiele

Beispiel 1: Geschwindigkeitsbegrenzung Maschinenachse

```
Programmcode                    Kommentar
...
N70 VELOLIM[X]=80               ; Der Achsschlitten in X-Richtung soll nur mit maximal
                                  80 % der für die Achse zulässigen Geschwindigkeit
                                  verfahren werden.
...
```

Beispiel 2: Drehzahlbegrenzung Spindel

```
Programmcode                    Kommentar
N05 VELOLIM[S1]=90              ; Begrenzung der Maximaldrehzahl von Spindel 1 auf 90%
                                  von 1000 U/min.
...
N50 VELOLIM[C]=45               ; Begrenzung der Drehzahl auf 45% von 1000 U/min, C sei
                                  der Achsbezeichner von S1.
...
```

Projektierungsdaten für Spindel 1 (AX5):

MD35130 $MA_GEAR_STEP_MAX_VELO_LIMIT[1,AX5]=1000	; Maximaldrehzahl der Getriebstufe 1 = 1000 U/min
MD30455 $MA_MISC_FUNCTION_MASK[AX5] = 64	; Bit 6 = 1: Die Programmierung von VELOLIM wirkt gemeinsam für Spindel- und Achsbetrieb unabhängig vom programmierten Bezeichner.

8.7.3 Programmbeispiel für JERKLIM und VELOLIM

Das folgende Programm stellt ein Anwendungsbeispiel für die prozentuale Ruck- und Geschwindigkeitsbegrenzung dar:

```
Programmcode                          Kommentar
N1000 G0 X0 Y0 F10000 SOFT G64
N1100 G1 X20 RNDM=5 ACC[X]=20
ACC[Y]=30
N1200 G1 Y20 VELOLIM[X]=5             ; Der Achsschlitten in X-Richtung soll nur
                                        mit max. 5% der für die Achse zulässigen
                                        Geschwindigkeit verfahren werden.
JERKLIM[Y]=200                        ; Der Achsschlitten in Y-Richtung kann mit
                                        max. 200% des für die Achse zulässigen
                                        Rucks beschleunigt/verzögert werden.
N1300 G1 X0 JERKLIM[X]=2              ; Der Achsschlitten in X-Richtung soll nur
                                        mit max. 2% des für die Achse zulässigen
                                        Rucks beschleunigt/verzögert werden.
N1400 G1 Y0
M30
```

8.8 Programmierbare Kontur-/Orientierungtoleranz (CTOL, OTOL, ATOL)

Funktion

Mit den Befehlen CTOL, OTOL und ATOL können die über Maschinen- und Settingdaten festgelegten Bearbeitungstoleranzen für die Kompressor-Funktionen (COMPON, COMPCURV, COMPCAD), die Überschleifarten G642, G643, G645, OST und die Orientierungsglättung ORISON im NC-Programm angepasst werden.

Die programmierten Werte gelten, bis sie neu programmiert oder durch Zuweisung eines negativen Werts gelöscht werden. Sie werden ferner gelöscht bei Programmende, Kanal-Reset, BAG-Reset, NCK-Reset (Warmstart) und Power On (Kaltstart). Nach dem Löschen gelten wieder die Werte aus den Maschinen- und Settingdaten.

Syntax

```
CTOL=<Wert>
OTOL=<Wert>
ATOL[<Achse>]=<Wert>
```

Bedeutung

CTOL Befehl zum Programmieren der **Konturtoleranz**

CTOL ist gültig für:

- alle Kompressor-Funktionen
- alle Überschleifarten außer G641 und G644

<Wert>: Der Wert für die Konturtoleranz ist eine Längenangabe.

Typ: REAL

Einheit: Inch/mm (abhängig von der aktuellen Einstellung der Maßangabe)

OTOL Befehl zum Programmieren der **Orientierungstoleranz**

OTOL ist gültig für:

- alle Kompressor-Funktionen
- Orientierungsglättung ORISON
- alle Überschleifarten außer G641, G644, OSD

<Wert>: Der Wert für die Orientierungstoleranz ist eine Winkelangabe.

Typ: REAL

Einheit: Grad

`ATOL` Befehl zum Programmieren einer **achsspezifischen Toleranz**

`ATOL` ist gültig für:

- alle Kompressor-Funktionen
- Orientierungsglättung ORISON
- alle Überschleifarten außer G641, G644, OSD

`<Achse>`: Name der Achse, für die eine Achstoleranz programmiert werden soll

`<Wert>`: Der Wert für die Achstoleranz ist je nach Achstyp (Linear- oder Rundachse) eine Längen- oder Winkelangabe.

Typ: REAL

Einheit: für Linearachsen: Inch/mm (abhängig von der aktuellen Einstellung der Maßangabe)

für Rundachsen: Grad

Hinweis

`CTOL` und `OTOL` haben Vorrang vor `ATOL`.

Randbedingungen

Skalierende Frames

Skalierende Frames wirken auf die programmierten Toleranzen in gleicher Weise wie auf die Achspositionen, d. h. die relative Toleranz bleibt gleich.

Beispiel

```
Programmcode                    Kommentar
COMPCAD G645 G1 F10000          ; Kompressor-Funktion COMPCAD aktivieren.
X... Y... Z...                  ; Hier wirken die Maschinen-und Settingdaten.
X... Y... Z...
X... Y... Z...
CTOL=0.02                       ; Ab hier wirkt eine Konturtoleranz von 0,02 mm.
X... Y... Z...
X... Y... Z...
X... Y... Z...
ASCALE X0.25 Y0.25 Z0.25        ; Ab hier wirkt eine Konturtoleranz von 0,005 mm.
X... Y... Z...
X... Y... Z...
X... Y... Z...
```

Programmcode	Kommentar
CTOL=-1	; Ab hier wirken wieder Maschinen- und Settingdaten.
X... Y... Z...	
X... Y... Z...	
X... Y... Z...	

Weitere Informationen

Toleranzwerte lesen

Für weitergehende Anwendungsfälle oder zur Diagnose sind die aktuell gültigen Toleranzen für die Kompressor-Funktionen (COMPON, COMPCURV, COMPCAD), die Überschleifarten G642, G643, G645, OST und die Orientierungsglättung ORISON unabhängig von der Art des Zustandekommens über Systemvariablen lesbar.

- In Synchronaktionen oder mit Vorlauf-Stopp im Teileprogramm über die Systemvariablen:

$AC_CTOL	Konturtoleranz, die bei der Aufbereitung des aktuellen Hauptlaufsatzes wirksam war Falls keine Konturtoleranz wirksam ist, liefert `$AC_CTOL` die Wurzel aus der Summe der Quadrate der Toleranzen der Geometrieachsen.
$AC_OTOL	Orientierungstoleranz, die bei der Aufbereitung des aktuellen Hauptlaufsatzes wirksam war Falls keine Orientierungstoleranz wirksam ist, liefert `$AC_OTOL` während einer aktiven Orientierungstransformation die Wurzel aus der Summe der Quadrate der Toleranzen der Orientierungsachsen, ansonsten den Wert "-1".
$AA_ATOL[<Achse>]	Achstoleranz, die bei der Aufbereitung des aktuellen Hauptlaufsatzes wirksam war Falls eine Konturtoleranz aktiv ist, liefert `$AA_ATOL[<Geometrieachse>]` die Konturtoleranz geteilt durch die Wurzel aus der Anzahl der Geometrieachsen. Falls eine Orientierungstoleranz und eine Orientierungstransformation aktiv sind, liefert `$AA_ATOL[<Orientierungsachse>]` die Orientierungstoleranz geteilt durch die Wurzel aus der Anzahl der Orientierungsachsen.

Hinweis

Wenn keine Toleranzwerte programmiert wurden, dann sind die $A-Variablen nicht differenziert genug, um die möglicherweise verschiedenen Toleranzen der einzelnen Funktionen zu unterscheiden, da sie ja nur einen Wert nennen können.

Solche Fälle können auftreten, wenn die Maschinen- und Settingdaten unterschiedliche Toleranzen für Kompressor-Funktionen, Überschleifen und Orientierungsglättung einstellen. Die Variablen liefern dann den größten Wert, der bei den gerade aktiven Funktionen auftritt.

Ist z. B. eine Kompressor-Funktion mit Orientierungstoleranz 0,1° und eine Orientierungsglättung ORISON mit 1° aktiv, liefert die Variable $AC_OTOL den Wert "1". Wird die Orientierungsglättung ausgeschaltet, liest man nur noch den Wert "0.1".

- Ohne Vorlauf-Stopp im Teileprogramm über die Systemvariablen:

$P_CTOL	Programmierte Konturtoleranz
$P_OTOL	Programmierte Orientierungstoleranz
$PA_ATOL	Programmierte Achstoleranz

Hinweis

Wenn keine Toleranzwerte programmiert sind, dann liefern die $P-Variablen den Wert "-1".

8.9 Toleranz bei G0-Bewegungen (STOLF)

G0-Toleranzfaktor

G0-Bewegungen (Eilgang, Zustellbewegungen) können im Unterschied zur Werkstückbearbeitung mit größerer Toleranz verfahren werden. Dies hat den Vorteil, dass sich die Abfahrzeiten für G0-Bewegungen verkürzen.

Die Einstellung der Toleranzen bei G0-Bewegungen erfolgt durch Projektierung des G0-Toleranzfaktors (MD20560 $MC_G0_TOLERANCE_FACTOR).

Der G0-Toleranzfaktor wird nur wirksam, wenn:

- eine der folgenden Funktionen aktiv ist:
  - Kompressorfunktionen: COMPON, COMPCURV und COMPCAD
  - Überschleiffunktionen: G642 und G645
  - Orientierungsüberschleifen: OST
  - Orientierungsglättung: ORISON
  - Glättung bei bahnrelativer Orientierung: ORIPATH
- mehrere (≥ 2) G0-Sätze aufeinanderfolgen.

  Bei einem einzelnen G0-Satz wird der G0-Toleranzfaktor nicht wirksam, da **beim Übergang** von einer Nicht-G0-Bewegung zu einer G0-Bewegung (und umgekehrt) grundsätzlich die "**kleinere Toleranz**" (Werkstückbearbeitungstoleranz) wirkt!

Funktion

Durch Programmierung von `STOLF` im Teileprogramm kann der projektierte G0-Toleranzfaktor (MD20560) temporär überschrieben werden. Der Wert im MD20560 wird dabei nicht verändert. Nach Reset bzw. Teileprogrammende wird der projektierte Toleranzfaktor wieder wirksam.

Syntax

```
STOLF=<Toleranzfaktor>
```

Bedeutung

`STOLF`:	Befehl zur Programmierung des G0-Toleranzfaktors
`<Toleranzfaktor>`:	G0-Toleranzfaktor Der Faktor kann sowohl größer 1 als auch kleiner 1 sein. Normalerweise werden jedoch für G0-Bewegungen größere Toleranzen einstellbar sein. Bei `STOLF=1.0` (entspricht dem projektierten Standardwert) sind für G0-Bewegungen dieselben Toleranzen wirksam wie für Nicht-G0-Bewegungen.

Systemvariablen

Der im Teileprogramm bzw. im aktuellen IPO-Satz wirksame G0-Toleranzfaktor ist über Systemvariablen lesbar.

- In Synchronaktionen oder mit Vorlauf-Stopp im Teileprogramm über die Systemvariable:

  $AC_STOLF Aktiver G0-Toleranzfaktor
  G0-Toleranzfaktor, der bei der Aufbereitung des aktuellen Hauptlaufsatzes wirksam war.

- Ohne Vorlauf-Stopp im Teileprogramm über die Systemvariable:

  $P_STOLF Programmierter G0-Toleranzfaktor

Ist im aktiven Teileprogramm kein Wert mit `STOLF` programmiert, dann liefern diese beiden Systemvariablen den durch MD20560 $MC_G0_TOLERANCE_FACTOR eingestellten Wert.

Ist in einem Satz kein Eilgang (G0) aktiv, dann liefern diese Systemvariablen immer den Wert 1.

Beispiel

```
Programmcode                    Kommentar
COMPCAD G645 G1 F10000          ; Kompressor-Funktion COMPCAD
X... Y... Z...                  ; Hier wirken die Maschinen- und Settingdaten.
X... Y... Z...
X... Y... Z...
G0 X... Y... Z...
G0 X... Y... Z...               ; Hier wirkt das Maschinendatum $MC_G0_TOLERANCE_FACTOR
                                  (z.B. =3), also eine Überschleiftoleranz von
                                  $MC_G0_TOLERANCE_FACTOR*$MA_COMPRESS_POS_TOL.
CTOL=0.02
STOLF=4
G1 X... Y... Z...               ; Ab hier wirkt eine Konturtoleranz von 0,02mm.
X... Y... Z...
X... Y... Z...
G0 X... Y... Z...
X... Y... Z...                  ; Ab hier wirkt ein G0-Toleranzfaktor von 4, also eine
                                  Konturtoleranz von 0,08mm.
```

Achskopplungen

9

9.1 Mitschleppen (TRAILON, TRAILOF)

Funktion

Beim Bewegen einer definierten Leitachse fahren ihr zugeordnete Mitschleppachsen (= Folgeachsen) unter Berücksichtigung eines Koppelfaktors die von der Leitachse abgeleiteten Verfahrwege ab.

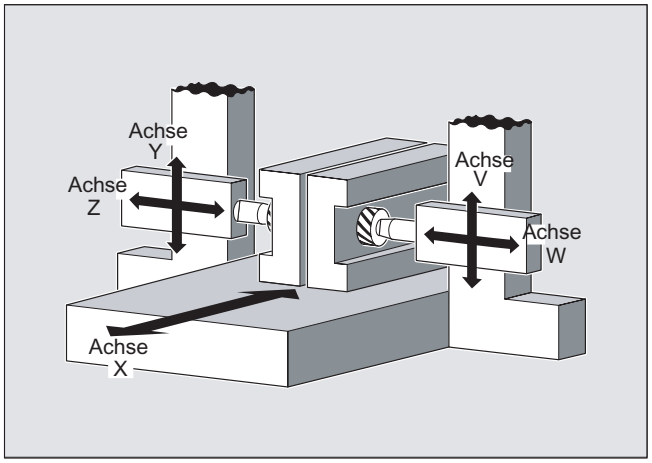
Leitachse und Folgeachsen bilden zusammen einen Mitschleppverband.

Anwendungsbereiche

- Verfahren einer Achse durch eine simulierte Achse. Die Leitachse ist eine simulierte Achse und die Mitschleppachse eine reale Achse. Damit kann die reale Achse mit Berücksichtigung des Koppelfaktors verfahren werden.
- Zweiseitenbearbeitung mit 2 Mitschleppverbänden:

  1. Leitachse Y, Mitschleppachse V

  2. Leitachse Z, Mitschleppachse W



Syntax

```
TRAILON(<Folgeachse>,<Leitachse>,<Koppelfaktor>)
TRAILOF(<Folgeachse>,<Leitachse>,<Leitachse 2>)
TRAILOF(<Folgeachse>)
```

Bedeutung

`TRAILON`	Befehl zum Einschalten und Definieren eines Mitschleppverbandes Wirksamkeit: modal
`<Folgeachse>`	Parameter 1: Achsbezeichnung der Mitschleppachse **Hinweis:** Eine Mitschleppachse kann auch Leitachse für weitere Mitschleppachsen sein. Auf diese Weise können unterschiedliche Mitschleppverbände aufgebaut werden.
`<Leitachse>`	Parameter 2: Achsbezeichnung der Leitachse
`<Koppelfaktor>`	Parameter 3: Koppelfaktor Der Koppelfaktor gibt das gewünschte Verhältnis der Wege von Mitschleppachse und Leitachse an: <Koppelfaktor> = Weg der Mitschleppachse / Weg der Leitachse Typ: REAL Voreinstellung: 1 Die Eingabe eines negativen Wertes bewirkt eine entgegengesetzte Verfahrbewegung der Leit- und Mitschleppachse. Wird der Koppelfaktor bei der Programmierung nicht angegeben, so gilt automatisch der Koppelfaktor 1.

`TRAILOF`	Befehl zum Ausschalten eines Mitschleppverbandes
	Wirksamkeit: modal
	`TRAILOF` mit 2 Parametern schaltet nur die Kopplung zur angegebenen Leitachse aus:
	`TRAILOF(<Folgeachse>,<Leitachse>)`
	Besitzt eine Mitschleppachse 2 Leitachsen, kann zum Ausschalten der beiden Kopplungen `TRAILOF` mit 3 Parametern aufgerufen werden:
	`TRAILOF(<Folgeachse>,<Leitachse>,<Leitachse 2>)`
	Das gleiche Ergebnis liefert die Programmierung von `TRAILOF` ohne Angabe einer Leitachse:
	`TRAILOF(<Folgeachse>)`

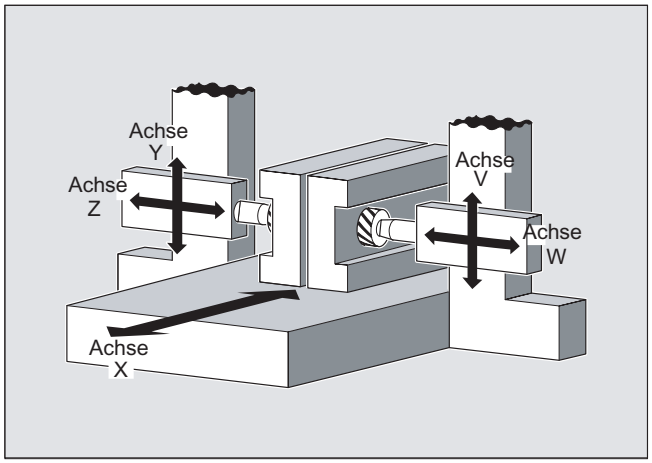
Hinweis

Das Mitschleppen erfolgt immer im Basiskoordinatensystem (BKS).

Die Anzahl der gleichzeitig aktivierbaren Mitschleppverbände wird nur begrenzt durch die Kombinationsmöglichkeiten der an der Maschine vorhandenen Achsen.

Beispiel

Das Werkstück soll zweiseitig mit der dargestellten Achskonstellation bearbeitet werden. Dazu bilden Sie 2 Mitschleppverbände.



Programmcode	Kommentar
…	
N100 TRAILON(V,Y)	; Einschalten des 1. Mitschleppverbandes
N110 TRAILON(W,Z,-1)	; Einschalten des 2. Mitschleppverbandes. Koppelfaktor negativ: Mitschleppachse fährt jeweils in entgegengesetzter Richtung wie Leitachse.
N120 G0 Z10	; Zustellung der Z- und W-Achse in entgegengesetzter Achsrichtung.
N130 G0 Y20	; Zustellung der Y- und V-Achse in gleicher Achsrichtung.
…	
N200 G1 Y22 V25 F200	; Überlagerung einer abhängigen und unabhängigen Bewegung der Mitschleppachse V.
…	
TRAILOF(V,Y)	; Ausschalten des 1. Mitschleppverbandes.
TRAILOF(W,Z)	; Ausschalten des 2. Mitschleppverbandes.

Weitere Informationen

Achstypen

Ein Mitschleppverband kann aus beliebigen Kombinationen von Linear- und Rundachsen bestehen. Als Leitachse kann dabei auch eine simulierte Achse definiert werden.

Mitschleppachsen

Einer Mitschleppachse können gleichzeitig maximal 2 Leitachsen zugeordnet werden. Die Zuordnung erfolgt in unterschiedlichen Mitschleppverbänden.

Eine Mitschleppachse kann mit allen zur Verfügung stehenden Bewegungsbefehlen programmiert werden (G0, G1, G2, G3, …). Zusätzlich zu den unabhängig definierten Wegen fährt die Mitschleppachse die mit den Koppelfaktoren aus ihren Leitachsen abgeleiteten Wege.

Dynamikbegrenzung

Die Dynamikbegrenzung ist abhängig von der Art der Aktivierung des Mitschleppverbandes:

- Aktivierung im Teileprogramm

  Erfolgt die Aktivierung im Teileprogramm und sind alle Leitachsen als Programmachsen im aktivierenden Kanal, wird beim Verfahren der Leitachsen die Dynamik aller Mitschleppachsen so berücksichtigt, dass keine Mitschleppachse überlastet wird.

  Erfolgt die Aktivierung im Teileprogramm mit Leitachsen, die nicht als Programmachsen im aktivierenden Kanal aktiv sind ($AA_TYP ≠ 1), wird beim Verfahren der Leitachsen die Dynamik der Mitschleppachse nicht berücksichtigt. Dadurch kann es bei Mitschleppachsen mit einer geringeren als der für die Kopplung benötigten Dynamik zu einer Überlastung kommen.

- Aktivierung in Synchronaktion

  Erfolgt die Aktivierung in einer Synchronaktion, wird beim Verfahren der Leitachsen die Dynamik der Mitschleppachsen nicht berücksichtigt. Dadurch kann es bei Mitschleppachsen mit einer geringeren als der für die Kopplung benötigten Dynamik zu einer Überlastung kommen.

⚠VORSICHT

Wird ein Mitschleppverband

- in Synchronaktionen
- im Teileprogramm mit Leitachsen, die nicht Programmachsen im Kanal der Mitschleppachse sind,

aktiviert, dann liegt es in der besonderen Verantwortung des Anwenders/Maschinenherstellers, geeignete Maßnahmen vorzusehen, damit es durch die Verfahrbewegungen der Leitachse nicht zu einer Überlastung der Mitschleppachsen kommt.

Kopplungsstatus

Der Kopplungsstatus einer Achse kann im Teileprogramm abgefragt werden mit der Systemvariablen:

$AA_COUP_ACT[<Achse>]

Wert	Bedeutung
0	Keine Kopplung aktiv
8	Mitschleppen aktiv

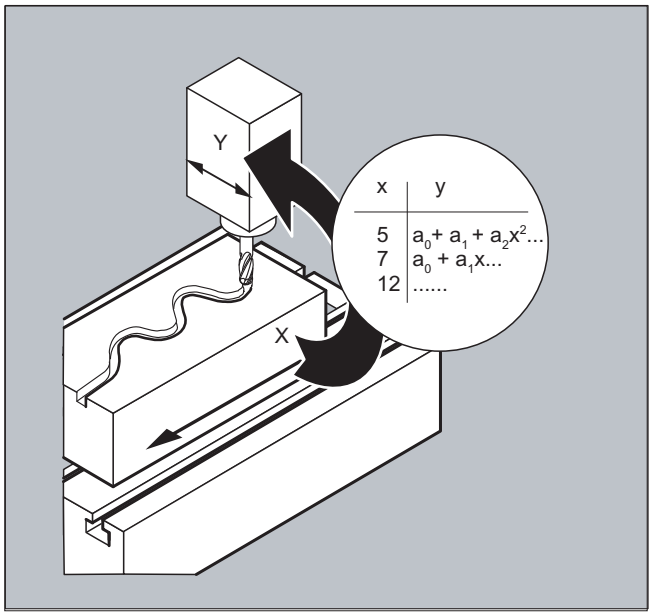
9.2 Kurventabellen (CTAB)

Funktion

Mit Hilfe von Kurventabellen können Positions- und Geschwindigkeitsbeziehungen zwischen zwei Achsen (Leit- und Folgeachse) programmiert werden. Die Kurventabellendefinition erfolgt im Teileprogramm.

Anwendung

Kurventabellen ersetzen mechanische Kurvenscheiben. Die Kurventabelle bildet dabei die Grundlage für die axiale Leitwertkopplung, indem sie den funktionellen Zusammenhang zwischen Leit- und Folgewert schafft: Die Steuerung berechnet bei entsprechender Programmierung aus einander zugeordneten Positionen von Leit- und Folgeachse ein Polynom, das der Kurvenscheibe entspricht.

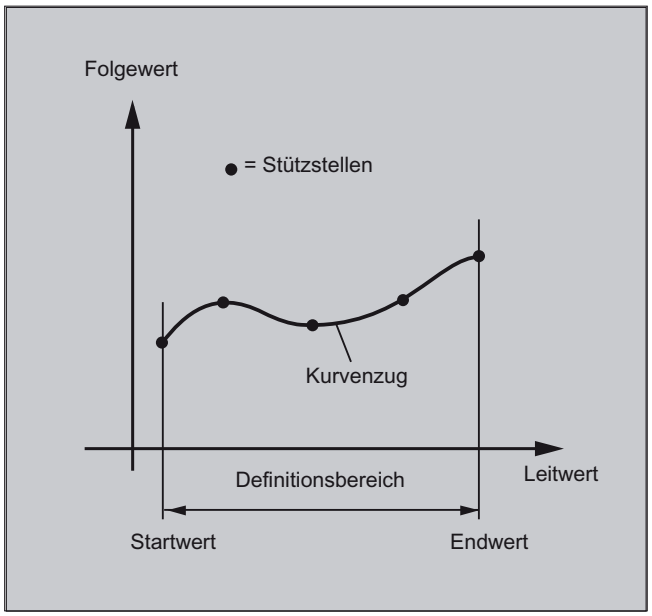


9.2.1 Kurventabellen definieren (CTABDEF, CATBEND)

Funktion

Eine Kurventabelle stellt ein Teileprogramm oder einen Teileprogrammabschnitt dar, welcher durch Voranstellen von `CTABDEF` und den abschließenden Befehl `CTABEND` gekennzeichnet ist.

Innerhalb dieses Teileprogrammabschnitts werden durch Bewegungsanweisungen einzelnen Positionen der Leitachse eindeutige Folgeachspositionen zugeordnet, die als Stützstellen für die Berechnung eines Kurvenzugs in Form eines Polynoms bis zu maximal 5. Grades dienen.



Voraussetzung

Für die Definition von Kurventabellen muss durch entsprechende MD-Projektierung Speicherplatz reserviert sein (→ Maschinenhersteller!).

Syntax

```
CTABDEF(<Folgeachse>,<Leitachse>,<n>,<Periodizität>[,<Speicherort>])
...
CTABEND
```

Bedeutung

`CTABDEF( )`	Beginn der Kurventabellendefinition
`CTABEND`	Ende der Kurventabellendefinition
`<Folgeachse>`	Achse, deren Bewegung über die Kurventabelle berechnet werden soll
`<Leitachse>`	Achse, die die Leitwerte zur Berechnung der Folgeachsbewegung liefert
`<n>`	Nummer (**ID**) der Kurventabelle Die Nummer einer Kurventabelle ist eindeutig und unabhängig vom Speicherort. Es können keine Tabellen mit der gleichen Nummer im statischen und dynamischen NC-Speicher liegen.
`<Periodizität>`	Tabellenperiodizität `0` Tabelle ist nicht periodisch (wird nur einmal abgearbeitet, auch bei Rundachsen) `1` Tabelle ist periodisch bezüglich Leitachse `2` Tabelle ist periodisch bezüglich Leitachse und Folgeachse
`<Speicherort>`	Angabe des Speicherorts (optional) `"SRAM"` Die Kurventabelle wird im **statischen** NC-Speicher angelegt. `"DRAM"` Die Kurventabelle wird im **dynamischen** NC-Speicher angelegt. **Hinweis:** Wenn für diesen Parameter kein Wert programmiert wird, dann wird der mit MD20905 $MC_CTAB_DEFAULT_MEMORY_TYPE eingestellte Standard-Speicherort verwendet.

Hinweis

Überschreiben

Eine Kurventabelle wird überschrieben, sobald bei einer neuen Tabellendefinition deren Nummer (`<n>`) benutzt wird (Ausnahme: die Kurventabelle ist in einer Achskopplung aktiv oder mit `CTABLOCK` gesperrt). **Beim Überschreiben wird keine entsprechende Warnung ausgegeben!**

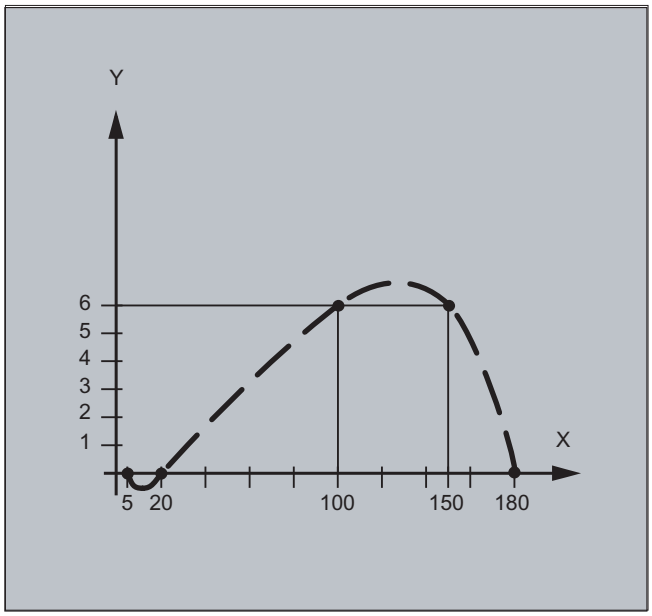
Beispiele

Beispiel 1: Programmabschnitt als Kurventabellendefinition

Ein Programmabschnitt soll unverändert zur Definition einer Kurventabelle benutzt werden. Der darin auftretende Befehl zum Vorlaufstopp STOPRE kann stehen bleiben und wird sofort wieder aktiv, sobald der Programmabschnitt nicht mehr zur Tabellendefinition benutzt wird und CTABDEF und CTABEND entfernt wurden.

```
Programmcode                    Kommentar
…
CTABDEF(Y,X,1,1)                ; Definition einer Kurventabelle.
…
IF NOT ($P_CTABDEF)
STOPRE
ENDIF
…
CTABEND
```

Beispiel 2: Definition einer nichtperiodischen Kurventabelle




```
Programmcode                Kommentar
N100 CTABDEF(Y,X,3,0)       ; Beginn der Definition einer nichtperiodischen
                              Kurventabelle mit der Nummer 3.
N110 X0 Y0                  ; 1.Bewegungsanweisung, legt Startwerte und 1. Stützstelle
                              fest:
                              Leitwert: 0, Folgewert: 0
N120 X20 Y0                 ; 2.Stützstelle:
                              Leitwert: 0…20, Folgewert: Startwert…0
```

```
Programmcode            Kommentar
N130 X100 Y6            ; 3.Stützstelle:
                          Leitwert: 20…100, Folgewert: 0…6
N140 X150 Y6            ; 4.Stützstelle:
                          Leitwert: 100…150, Folgewert: 6…6
N150 X180 Y0            ; 5.Stützstelle:
                          Leitwert: 150…180, Folgewert: 6…0
N200 CTABEND            ; Ende der Definition. Die Kurventabelle wird in ihrer
                          internen Darstellung als Polynom maximal 5.Grades erzeugt.
                          Die Berechnung des Kurvenzugs mit den angegebenen
                          Stützstellen ist abhängig von der modal gewählten
                          Interpolationsart (Kreis-, Linear-, Spline-Interpolation).
                          Der Teileprogrammzustand vor Beginn der Definition wird
                          wiederhergestellt.
```

Beispiel 3: Definition einer periodischen Kurventabelle

Definition einer periodischen Kurventabelle mit Nummer 2, Leitwertbereich von 0 bis 360, Folgeachsbewegung von 0 nach 45 und zurück nach 0:

```
Programmcode                                    Kommentar
N10 DEF REAL DEPPOS
N20 DEF REAL GRADIENT
N30 CTABDEF(Y,X,2,1)                            ; Beginn der Definition.
N40 G1 X=0 Y=0
N50 POLY
N60 PO[X]=(45.0)
N70 PO[X]=(90.0) PO[Y]=(45.0,135.0,-90)
N80 PO[X]=(270.0)
N90 PO[X]=(315.0) PO[Y]=(0.0,-135.0,90)
N100 PO[X]=(360.0)
N110 CTABEND                                    ; Ende der Definition.
;Test der Kurve durch Kopplung von Y an X:
N120 G1 F1000 X0
N130 LEADON(Y,X,2)
N140 X360
N150 X0
N160 LEADOF(Y,X)
N170 DEPPOS=CTAB(75.0,2,GRADIENT)               ; Lesen der Tabellenfunktion beim
                                                  Leitwert 75.0.
N180 G0 X75 Y=DEPPOS                            ; Positionieren von Leit- und
                                                  Folgeachse.
;Nach Einschalten der Kopplung ist kein Synchronisieren der Folgeachse nötig.
N190 LEADON(Y,X,2)
N200 G1 X110 F1000
N210 LEADOF(Y,X)
N220 M30
```

Weitere Informationen

Start- und Endwert der Kurventabelle

Als Startwert für den Beginn des Definitionsbereichs der Kurventabelle gilt die erste Angabe von zusammengehörigen Achspositionen (die erste Bewegungsanweisung) innerhalb der Kurventabellendefinition. Der Endwert des Definitionsbereichs der Kurventabelle wird entsprechend durch den letzten Verfahrbefehl bestimmt.

Verfügbarer Sprachumfang

Innerhalb der Definition der Kurventabelle steht der gesamte NC-Sprachumfang zur Verfügung.

Hinweis

Folgende Angaben sind In Kurventabellendefinitionen nicht zulässig:

- Vorlaufstopp
- Sprünge in der Leitachsenbewegung (z. B. beim Wechsel von Transformationen)
- Bewegungsanweisung allein für die Folgeachse
- Bewegungsumkehr der Leitachse, d. h. Position der Leitachse muss immer eindeutig sein
- `CTABDEF`- und `CTABEND`-Anweisung in unterschiedlichen Programmebenen.

Wirksamkeit von modalen Anweisungen

Sämtliche modal wirksamen Anweisungen, die innerhalb der Kurventabellendefinition getroffen werden, sind mit Abschluss der Tabellendefinition ungültig. Das Teileprogramm, in dem die Tabellendefinition erfolgt, befindet sich damit vor und nach der Tabellendefinition im gleichen Zustand.

Zuweisungen an R-Parameter

Zuweisungen an R-Parameter innerhalb der Tabellendefinition werden nach `CTABEND` zurückgesetzt.

Beispiel:

```
Programmcode                        Kommentar
...
R10=5 R11=20                        ; R10=5
...
CTABDEF
G1 X=10 Y=20 F1000
R10=R11+5                           ; R10=25
X=R10
CTABEND
...                                 ; R10=5
```

Aktivierung von ASPLINE, BSPLINE, CSPLINE

Wird innerhalb einer Kurventabellendefinition CTABDEF ... CTABEND ein ASPLINE, BSPLINE oder CSPLINE aktiviert, so sollte vor dieser Spline-Aktivierung mindestens ein Startpunkt programmiert werden. Eine sofortige Aktivierung nach CTABDEF sollte vermieden werden, da sonst der Spline von der aktuellen Achsposition vor der Kurventabellendefinition abhängt.

Beispiel:

Programmcode

```
...
CTABDEF(Y,X,1,0)
X0 Y0
ASPLINE
X=5 Y=10
X10 Y40
...
CTABEND
```

Wiederholte Verwendung von Kurventabellen

Der über die Kurventabelle berechnete funktionelle Zusammenhang von Leit- und Folgeachse bleibt unter der gewählten Tabellennummer über das Teileprogrammende und über POWER OFF hinaus erhalten, falls die Tabelle im statischen NC-Speicher (SRAM) abgelegt ist.

Eine Tabelle, die im dynamischen Speicher (DRAM) angelegt wurde, wird bei POWER ON gelöscht und muss eventuell noch einmal erzeugt werden.

Die einmal erstellte Kurventabelle lässt sich auf beliebige Achskombinationen von Leit- und Folgeachse anwenden und ist unabhängig davon, welche Achsen zur Erstellung der Kurventabelle benutzt wurden.

Überschreiben von Kurventabellen

Eine Kurventabelle wird überschrieben, sobald bei einer erneuten Tabellendefinition deren Nummer benutzt wird.

Ausnahme: Eine Kurventabelle ist in einer Achskopplung aktiv oder mit CTABLOCK gesperrt.

Hinweis

Beim Überschreiben von Kurventabellen wird keine entsprechende Warnung ausgegeben!

Kurventabellendefinition aktiv?

Mit der Systemvariablen $P_CTABDEF kann aus dem Teileprogramm heraus jederzeit abgefragt werden, ob eine Kurventabellendefinition aktiv ist.

Aufheben der Kurventabellendefinition

Der Teileprogrammabschnitt ist nach Ausklammern der Anweisungen zur Kurventabellendefinition wieder als reales Teileprogramm verwendbar.

Laden von Kurventabellen über "Abarbeiten von Extern"

Beim externen Abarbeiten von Kurventabellen muss die Größe des Nachladebuffers (DRAM) über MD18360 $MN_MM_EXT_PROG_BUFFER_SIZE so gewählt werden, dass die gesamte Kurventabellendefinition gleichzeitig im Nachladebuffer abgelegt werden kann. Die Teileprogrammbearbeitung wird anderenfalls mit einem Alarm abgebrochen.

Sprünge der Folgeachse

Abhängig von der Einstellung im Maschinendatum:
MD20900 $MC_CTAB_ENABLE_NO_LEADMOTION
können Sprünge der Folgeachse bei fehlender Bewegung der Leitachse toleriert werden.

9.2.2 Vorhandensein einer Kurventabelle prüfen (CTABEXISTS)

Funktion

Mit dem Befehl `CTABEXISTS` kann geprüft werden, ob eine bestimmte Kurventabellennummer im NC-Speicher vorhanden ist.

Syntax

```
CTABEXISTS(<n>)
```

Bedeutung

`CTABEXISTS`	Prüft, ob die Kurventabelle mit Nummer `<n>` im statischen oder dynamischen NC-Speicher vorhanden ist 0 Tabelle existiert nicht 1 Tabelle existiert
`<n>`	Nummer (**ID**) der Kurventabelle

9.2.3 Kurventabellen löschen (CTABDEL)

Funktion

Mit `CTABDEL` können Kurventabellen gelöscht werden.

Hinweis

Kurventabellen, die in einer Achskopplung aktiv sind, können nicht gelöscht werden.

Syntax

```
CTABDEL(<n>)
CTABDEL(<n>,<m>)
CTABDEL(<n>,<m>,<Speicherort>)
CTABDEL()
CTABDEL(,,<Speicherort>)
```

Bedeutung

`CTABDEL`	Befehl zum Löschen von Kurventabellen
`<n>`	Nummer (**ID**) der zu löschenden Kurventabelle Beim Löschen eines Kurventabellenbereichs `CTABDEL(<n>,<m>)` wird mit `<n>` die Nummer der ersten Kurventabelle des Bereichs angegeben.
`<m>`	Beim Löschen eines Kurventabellenbereichs `CTABDEL(<n>,<m>)` wird mit `<m>` die Nummer der letzten Kurventabelle des Bereichs angegeben. `<m>` muss größer `<n>` sein!
`<Speicherort>`	Angabe des Speicherorts (optional) Beim Löschen **ohne** Speicherort-Angabe werden die angegebenen Kurventabellen im statischen und dynamischen NC-Speicher gelöscht. Beim Löschen **mit** Speicherort-Angabe werden von den angegebenen Kurventabellen nur diejenigen gelöscht, die im angegebenen Speicher liegen. Die übrigen bleiben bestehen. `"SRAM"` Löschen im **statischen** NC-Speicher `"DRAM"` Löschen im **dynamischen** NC-Speicher

Wird `CTABDEL` ohne Angabe der zu löschenden Kurventabelle programmiert, dann werden **alle** Kurventabellen bzw. alle Kurventabellen im angegebenen Speicher gelöscht:

`CTABDEL()`	Löscht alle Kurventabellen im statischen und dynamischen NC-Speicher
`CTABDEL(,,"SRAM")`	Löscht alle Kurventabellen im statischen NC-Speicher
`CTABDEL(,,"DRAM")`	Löscht alle Kurventabellen im dynamischen NC-Speicher

Hinweis

Wenn beim Mehrfachlöschen `CTABDEL(<n>,<m>)` oder `CTABDEL()` wenigstens eine der zu löschenden Kurventabellen in einer Kopplung aktiv ist, dann wird der Löschbefehl nicht ausgeführt, d. h. **keine** der adressierten Kurventabellen wird gelöscht.

9.2.4 Kurventabellen gegen Löschen und Überschreiben sperren (CTABLOCK, CTABUNLOCK)

Funktion

Kurventabellen können durch Setzen von Sperren vor unbeabsichtigtem Löschen und Überschreiben geschützt werden. Eine gesetzte Sperre kann jederzeit auch wieder aufgehoben werden.

Syntax

Sperre setzen:

```
CTABLOCK(<n>)
CTABLOCK(<n>,<m>)
CTABLOCK(<n>,<m>,<Speicherort>)
CTABLOCK()
CTABLOCK(,,<Speicherort>)
```

Sperre aufheben:

```
CTABUNLOCK(<n>)
CTABUNLOCK(<n>,<m>)
CTABUNLOCK(<n>,<m>,<Speicherort>)
CTABUNLOCK()
CTABUNLOCK(,,<Speicherort>)
```

Bedeutung

CTABLOCK	Befehl zum **Setzen** einer Sperre gegen Löschen/Überschreiben
CTABUNLOCK	Befehl zum **Aufheben** einer Sperre gegen Löschen/Überschreiben CTABUNLOCK gibt die mit CTABLOCK gesperrten Kurventabellen wieder frei. Kurventabellen, die in einer aktiven Kopplung wirken, bleiben weiterhin gesperrt und können nicht gelöscht werden. Die Sperre mit CTABLOCK ist aufgehoben, sobald die Sperrung durch die aktive Kopplung mit Deaktivierung der Kopplung aufgehoben wird. Damit kann diese Tabelle gelöscht werden. Ein nochmaliger CTABUNLOCK-Aufruf ist nicht notwendig.
<n>	Nummer (**ID**) der zu sperrenden/entsperrenden Kurventabelle Beim Sperren/Entsperren eines Kurventabellenbereichs CTABLOCK(<n>,<m>)/CTABUNLOCK(<n>,<m>) wird mit <n> die Nummer der ersten Kurventabelle des Bereichs angegeben.

<m>	Beim Sperren/Entsperren eines Kurventabellenbereichs CTABLOCK(<n>,<m>)/CTABUNLOCK(<n>,<m>) wird mit <m> die Nummer der letzten Kurventabelle des Bereichs angegeben. <m> muss größer <n> sein!
<Speicherort>	Angabe des Speicherorts (optional) Beim Setzen/Aufheben einer Sperre **ohne** Speicherort-Angabe werden die angegebenen Kurventabellen im statischen und dynamischen NC-Speicher gesperrt/entsperrt. Beim Setzen/Aufheben einer Sperre **mit** Speicherort-Angabe werden von den angegebenen Kurventabellen nur diejenigen gesperrt/entsperrt, die im angegebenen Speicher liegen. Die übrigen werden nicht gesperrt/entsperrt. "SRAM" Sperre setzen/aufheben im **statischen** NC-Speicher "DRAM" Sperre setzen/aufheben im **dynamischen** NC-Speicher

Wird CTABLOCK/CTABUNLOCK ohne Angabe der zu sperrenden/entsperrenden Kurventabelle programmiert, dann werden **alle** Kurventabellen bzw. alle Kurventabellen im angegebenen Speicher gesperrt/entsperrt:

CTABLOCK()	Sperrt alle Kurventabellen im statischen und dynamischen NC-Speicher
CTABLOCK(,,"SRAM")	Sperrt alle Kurventabellen im statischen NC-Speicher
CTABLOCK(,,"DRAM")	Sperrt alle Kurventabellen im dynamischen NC-Speicher
CTABUNLOCK()	Entsperrt alle Kurventabellen im statischen und dynamischen NC-Speicher
CTABUNLOCK(,,"SRAM")	Entsperrt alle Kurventabellen im statischen NC-Speicher
CTABUNLOCK(,,"DRAM")	Entsperrt alle Kurventabellen im dynamischen NC-Speicher

9.2.5 Kurventabellen: Tabelleneigenschaften ermitteln (CTABID, CTABISLOCK, CTABMEMTYP, CTABPERIOD)

Funktion

Mit diesen Befehlen können wichtige Eigenschaften einer Kurventabelle (Tabellennummer, Sperrzustand, Speicherort, Periodizität) abgefragt werden.

Syntax

```
CTABID(<p>)
CTABID(<p>,<Speicherort>)
CTABISLOCK(<n>)
CTABMEMTYP(<n>)
TABPERIOD(<n>)
```

Bedeutung

`CTABID` Liefert die **Tabellennummer**, die im angegebenen Speicher als die `<p>`-te Kurventabelle eingetragen ist.

Beispiel:

`CTABID(1,"SRAM")` liefert die Nummer der ersten Kurventabelle im statischen NC-Speicher. Die erste Kurventabelle entspricht dabei der Kurventabelle mit der höchsten Tabellennummer.

Hinweis:
Wird zwischen aufeinander folgenden Aufrufen von `CTABID` die Reihenfolge der Kurventabellen im Speicher geändert, z. B. durch Löschen von Kurventabellen mit `CTABDEL`, kann `CTABID(<p>,...)` mit derselben Nummer `<p>` eine andere Kurventabelle liefern als vorher.

`CTABISLOCK` Gibt den **Sperrzustand** der Kurventabelle mit Nummer `<n>` zurück:

0 Tabelle ist nicht gesperrt
1 Tabelle ist gesperrt durch `CTABLOCK`
2 Tabelle ist gesperrt durch aktive Kopplung
3 Tabelle ist gesperrt durch `CTABLOCK` und aktive Kopplung
-1 Tabelle existiert nicht

`CTABMEMTYP` Liefert den **Speicherort** der Kurventabelle mit Nummer `<n>`:

0 Tabelle im statischen NC-Speicher
1 Tabelle im dynamischen NC-Speicher
-1 Tabelle existiert nicht

`CTABPERIOD` Liefert die **Periodizität** der Kurventabelle mit Nummer `<n>`:

0 Tabelle ist nicht periodisch
1 Tabelle ist periodisch in der Leitachse
2 Tabelle ist periodisch in der Leit- und Folgeachse
-1 Tabelle existiert nicht

`<p>` Eintragsnummer im Speicher

`<n>` Nummer (**ID**) der Kurventabelle

`<Speicherort>` Angabe des Speicherorts (optional)

`"SRAM"` **Statischer** NC-Speicher
`"DRAM"` **Dynamischer** NC-Speicher

Hinweis:
Wenn für diesen Parameter kein Wert programmiert wird, dann wird der mit MD20905 $MC_CTAB_DEFAULT_MEMORY_TYPE eingestellte Standard-Speicherort verwendet.

9.2.6 Kurventabellenwerte lesen (CTABTSV, CTABTEV, CTABTSP, CTABTEP, CTABSSV, CTABSEV, CTAB, CTABINV, CTABTMIN, CTABTMAX)

Funktion

Folgende Kurventabellenwerte können im Teileprogramm gelesen werden:

- Folgeachs- und Leitachswerte am Anfang und Ende einer Kurventabelle
- Folgeachswerte am Anfang und Ende eines Kurvensegments
- Folgeachswert zu einem Leitachswert
- Leitachswert zu einem Folgeachswert
- Minimal- und Maximalwert der Folgeachse
  - im gesamten Definitionsbereich der Kurventabelle

    oder
  - in einem definierten Kurventabellenintervall

Syntax

```
CTABTSV(<n>,<Gradient>[,<Folgeachse>])
CTABTEV(<n>,<Gradient>[,<Folgeachse>])
CTABTSP(<n>,<Gradient>[,<Leitachse>])
CTABTEP(<n>,<Gradient>[,<Leitachse>])
CTABSSV(<Leitwert>,<n>,<Gradient>[,<Folgeachse>])
CTABSEV(<Leitwert>,<n>,<Gradient>[,<Folgeachse>])
CTAB(<Leitwert>,<n>,<Gradient>[,<Folgeachse>,<Leitachse>]
CTABINV(<Folgewert>,<Näherungswert>,<n>,<Gradient>[,<Folgeachse>,<Le
itachse>]
CTABTMIN(<n>[,<Folgeachse>])
CTABTMAX(<n>[,<Folgeachse>])
CTABTMIN(<n>,<a>,<b>[,<Folgeachse>,<Leitachse>])
CTABTMAX(<n>,<a>,<b>[,<Folgeachse>,<Leitachse>])
```

Bedeutung

CTABTSV:	Folgeachswert am **Anfang** der Kurventabelle Nr. <n> lesen
CTABTEV:	Folgeachswert am **Ende** der Kurventabelle Nr. <n> lesen
CTABTSP:	Leitachswert am **Anfang** der Kurventabelle Nr. <n> lesen
CTABTEP:	Leitachswert am **Ende** der Kurventabelle Nr. <n> lesen
CTABSSV:	Folgeachswert am **Anfang** des zum angegebenen Leitachswert (<Leitwert>) gehörenden Kurvensegments lesen
CTABSEV:	Folgeachswert am **Ende** des zum angegebenen Leitachswert (<Leitwert>) gehörenden Kurvensegments lesen
CTAB:	Folgeachswert zum angegebenen Leitachswert (<Leitwert>) lesen
CTABINV:	Leitachswert zum angegebenen Folgeachswert (<Folgewert>) lesen
CTABTMIN:	**Minimalwert** der Folgeachse bestimmen: • im gesamten Definitionsbereich der Kurventabelle oder • in einem definierten Intervall <a> ... <b>
CTABTMAX:	**Maximalwert** der Folgeachse bestimmen: • im gesamten Definitionsbereich der Kurventabelle oder • in einem definierten Intervall <a> ... <b>
<n>:	Nummer (**ID**) der Kurventabelle
<Gradient>:	Im Parameter <Gradient> wird die **Steigung** der Kurventabellenfunktion an der ermittelten Position zurückgegeben
<Folgeachse>:	Achse, deren Bewegung über die Kurventabelle berechnet werden soll (optional)
<Leitachse>:	Achse, die die Leitwerte zur Berechnung der Folgeachsbewegung liefert (optional)
<Folgewert>:	Folgeachswert zum Lesen des zugehörigen Leitachswerts bei CTABINV
<Leitwert>:	Leitachswert: • zum Lesen des zugehörigen Folgeachswerts bei CTAB oder • für die Auswahl des Kurvensegments bei CTABSSV/CTABSEV
<Näherungswert>:	Die Zuordnung eines Leitachswerts zu einem Folgeachswert bei CTABINV muss nicht immer eindeutig sein. CTABINV benötigt daher als Parameter einen Näherungswert für den erwarteten Leitachswert.
<a>:	**Untere** Grenze des Leitwertintervalls bei CTABTMIN/CTABTMAX
<b>:	**Obere** Grenze des Leitwertintervalls bei CTABTMIN/CTABTMAX **Hinweis:** Das Leitwertintervall <a> ... <b> muss innerhalb des Definitionsbereichs der Kurventabelle liegen.

Beispiele

Beispiel 1:

Folgeachs- und Leitachswerte am Anfang und Ende der Kurventabelle sowie Minimal- und Maximalwert der Folgeachse im gesamten Definitionsbereich der Kurventabelle bestimmen.

```
Programmcode                                Kommentar
N10 DEF REAL STARTPOS
N20 DEF REAL ENDPOS
N30 DEF REAL STARTPARA
N40 DEF REAL ENDPARA
N50 DEF REAL MINVAL
N60 DEF REAL MAXVAL
N70 DEF REAL GRADIENT
...
N100 CTABDEF(Y,X,1,0)                       ; Beginn der Tabellendefinition
N110 X0 Y10                                 ; Startposition 1.Tabellensegment
N120 X30 Y40                                ; Endposition 1.Tabellensegment = Startposition
                                              2.Tabellensegment
N130 X60 Y5                                 ; Endposition 2.Tabellensegment = ...
N140 X70 Y30
N150 X80 Y20
N160 CTABEND                                ; Ende der Tabellendefinition.
...
N200 STARTPOS=CTABTSV(1,GRADIENT)           ; Folgeachswert am Kurventabellenanfang = 10
N210 ENDPOS=CTABTEV(1,GRADIENT)             ; Folgeachswert am Kurventabellenende = 20
N220 STARTPARA=CTABTSP(1,GRADIENT)          ; Leitachswert am Kurventabellenanfang = 0
N230 ENDPARA=CTABTEP(1,GRADIENT)            ; Leitachswert am Kurventabellenende = 80
N240 MINVAL=CTABTMIN(1)                     ; Minimalwert der Folgeachse bei Y=5
N250 MAXVAL=CTABTMAX(1)                     ; Maximalwert der Folgeachse bei Y=40
```

Beispiel 2:

Bestimmung der Folgeachswerte am Anfang und Ende des zum Leitachswert X=30 gehörenden Kurvensegments.

```
Programmcode                                Kommentar
N10 DEF REAL STARTPOS
N20 DEF REAL ENDPOS
N30 DEF REAL GRADIENT
...
N100 CTABDEF(Y,X,1,0)                       ; Beginn der Tabellendefinition.
N110 X0 Y0                                  ; Startposition 1.Tabellensegment
N120 X20 Y10                                ; Endposition 1.Tabellensegment = Startposition
                                              2.Tabellensegment
N130 X40 Y40                                  Endposition 2.Tabellensegment = ...
```

```
Programmcode                                    Kommentar
N140 X60 Y10
N150 X80 Y0
N160 CTABEND                                    ; Ende der Tabellendefinition.
...
N200 STARTPOS=CTABSSV(30.0,1,GRADIENT)          ; Startposition Y im 2.Segment = 10
N210 ENDPOS=CTABSEV(30.0,1,GRADIENT)            ; Endposition Y im 2.Segment = 40
```

Weitere Informationen

Verwendung in Synchronaktionen

Alle Befehle zum Lesen von Kurventabellenwerten können auch in Synchronaktionen verwendet werden (siehe auch Kapitel "Bewegungssynchronaktionen").

Bei Verwendung der Befehle CTABINV, CTABTMIN und CTABTMAX ist darauf zu achten, dass:

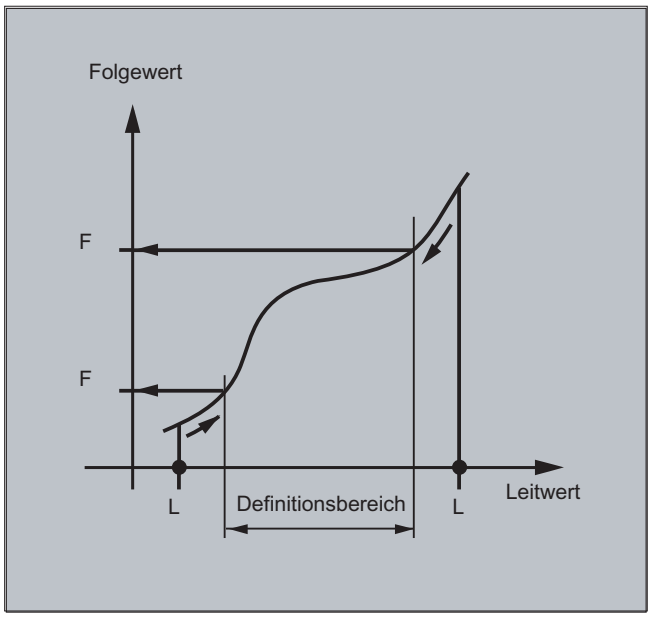
- zum Ausführungszeitpunkt ausreichend NC-Leistung verfügbar ist

  oder

- vor dem Aufruf die Anzahl der Segmente der Kurventabelle abgefragt wird, um gegebenenfalls die betreffende Tabelle unterteilen zu können

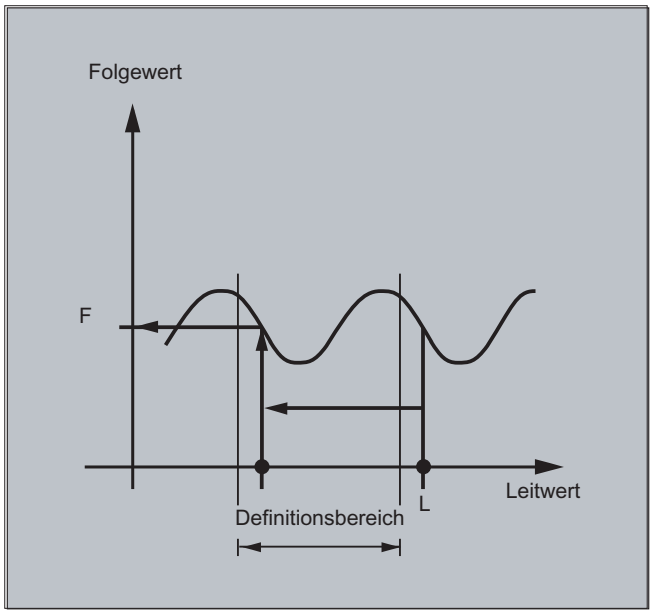
CTAB bei nichtperiodischen Kurventabellen

Liegt der angegebene <Leitwert> außerhalb des Definitionsbereichs, wird als Folgewert die obere bzw. untere Grenze ausgegeben:



CTAB bei periodischen Kurventabellen

Liegt der angegebene <Leitwert> außerhalb des Definitionsbereichs, wird der Leitwert Modulo des Definitionsbereichs bewertet und der entsprechende Folgewert ausgegeben:



Näherungswert für CTABINV

Der Befehl CTABINV benötigt einen Näherungswert für den erwarteten Leitwert. CTABINV gibt den Leitwert zurück, der dem Näherungswert am nächsten liegt. Der Näherungswert kann z. B. der Leitwert aus dem vorherigen Interpolationstakt sein.

Steigung der Kurventabellenfunktion

Die Ausgabe der Steigung (<Gradient>) ermöglicht es, die Geschwindigkeit der Leit- oder Folgeachse an der entsprechenden Position zu berechnen.

Angabe der Leit- oder Folgeachse

Die optionale Angabe der Leit- und/oder Folgeachse ist wichtig, falls Leit- und Folgeachse in verschiedenen Längeneinheiten projektiert sind.

CTABSSV, CTABSEV

Die Befehle CTABSSV und CTABSEV sind in folgenden Fällen **nicht** dazu geeignet, programmierte Segmente abzufragen:

- Kreise oder Evolventen sind programmiert.
- Fasen bzw. Runden mit CHF/RND ist aktiv.
- Überschleifen mit G643 ist aktiv.
- NC-Satz-Kompression mit COMPON/COMPCURV/COMPCAD ist aktiv.

9.2.7 Kurventabellen: Ressourcennutzung prüfen (CTABNO, CTABNOMEM, CTABFNO, CTABSEGID, CTABSEG, CTABFSEG, CTABMSEG, CTABPOLID, CTABPOL, CTABFPOL, CTABMPOL)

Funktion

Mit diesen Befehlen hat der Programmierer die Möglichkeit, sich aktuell über die Belegung der Ressourcen für Kurventabellen, Tabellensegmente und Polynome zu informieren.

Syntax

```
CTABNO
CTABNOMEM(<Speicherort>)
CTABFNO(<Speicherort>)
CTABSEGID(<n>,<Speicherort>)
CTABSEG(<Speicherort>,<Segmentart>)
CTABFSEG(<Speicherort>,<Segmentart>)
CTABMSEG(<Speicherort>,<Segmentart>)
CTABPOLID(<n>)
CTABPOL(<Speicherort>)
CTABFPOL(<Speicherort>)
CTABMPOL(<Speicherort>)
```

Bedeutung

`CTABNO`	Gesamtanzahl der **definierten** Kurventabellen bestimmen (im statischen und dynamischen NC-Speicher)
`CTABNOMEM`	Anzahl der **definierten** Kurventabellen im angegebenen `<Speicherort>` bestimmen
`CTABFNO`	Anzahl der **noch möglichen** Kurventabellen im angegebenen `<Speicherort>` bestimmen
`CTABSEGID`	Anzahl der Kurvensegmente der angegebenen `<Segmentart>` bestimmen, die von der Kurventabelle mit Nummer `<n>` verwendet werden
`CTABSEG`	Anzahl der **verwendeten** Kurvensegmente der angegebenen `<Segmentart>` im angegebenen `<Speicherort>` bestimmen
`CTABFSEG`	Anzahl der **noch möglichen** Kurvensegmente der angegebenen `<Segmentart>` im angegebenen `<Speicherort>` bestimmen
`CTABMSEG`	Anzahl der **maximal möglichen** Kurvensegmente der angegebenen `<Segmentart>` im angegebenen `<Speicherort>` bestimmen
`CTABPOLID`	Anzahl der Kurvenpolynome bestimmen, die von der Kurventabelle mit Nummer `<n>` verwendet werden
`CTABPOL`	Anzahl der **verwendeten** Kurvenpolynome im angegebenen `<Speicherort>` bestimmen
`CTABFPOL`	Anzahl der **noch möglichen** Kurvenpolynome im angegebenen `<Speicherort>` bestimmen

CTABMPOL	Anzahl der **maximal möglichen** Kurvenpolynome im angegebenen <Speicherort> bestimmen
<n>	Nummer (**ID**) der Kurventabelle
<Speicherort>	Angabe des Speicherorts (optional) "SRAM" **Statischer** NC-Speicher "DRAM" **Dynamischer** NC-Speicher **Hinweis:** Wenn für diesen Parameter kein Wert programmiert wird, dann wird der mit MD20905 $MC_CTAB_DEFAULT_MEMORY_TYPE eingestellte Standard-Speicherort verwendet.
<Segmentart>	Angabe der Segmentart (optional) "L" Lineare Segmente "P" Polynomsegmente **Hinweis:** Wenn für diesen Parameter kein Wert programmiert wird, dann wird die Summe aus Linear- und Polynom-Segmenten ausgegeben.

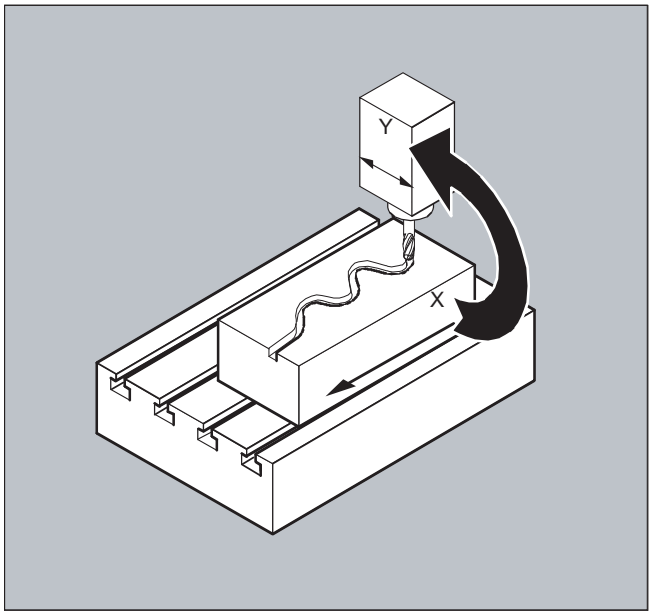
9.3 Axiale Leitwertkopplung (LEADON, LEADOF)

Hinweis

Diese Funktion steht für SINUMERIK 828D nicht zur Verfügung!

Funktion

Bei der axialen Leitwertkopplung werden eine Leit- und eine Folgeachse synchron verfahren. Dabei ist die jeweilige Position der Folgeachse über eine Kurventabelle bzw. ein daraus berechnetes Polynom eindeutig einer - ggf. simulierten - Position der Leitachse zugeordnet.



Leitachse heißt diejenige Achse, die die Eingangswerte für die Kurventabelle liefert.
Folgeachse heißt die Achse, die die über die Kurventabelle errechneten Positionen einnimmt.

Ist- und Sollwertkopplung

Als Leitwerte, also Ausgangswerte zur Positionsermittlung der Folgeachse können verwendet werden:

- Istwerte der Leitachsposition: Istwertkopplung
- Sollwerte der Leitachsposition: Sollwertkopplung

Die Leitwertkopplung gilt immer im Basiskoordinatensystem.

Zur Erstellung von Kurventabellen siehe Kapitel "Kurventabellen".
Zur Leitwertkopplung siehe /FB/, M3, Mitschleppen und Leitwertkopplung.

Syntax

```
LEADON(FAchse,LAchse,n)
LEADOF(FAchse,LAchse)
```

oder Ausschalten ohne Angabe der Leitachse:

```
LEADOF(FAchse)
```

Die Leitwertkopplung kann sowohl vom Teileprogramm als auch während der Bewegung aus Synchronaktionen, siehe Kapitel "Bewegungssynchronaktionen" heraus ein- und ausgeschaltet werden.

Bedeutung

`LEADON`	Leitwertkopplung einschalten
`LEADOF`	Leitwertkopplung ausschalten
`FAchse`	Folgeachse
`LAchse`	Leitachse
`n`	Kurventabellen-Nummer
`$SA_LEAD_TYPE`	Umschaltung zwischen Soll- und Istwertkopplung

Leitwertkopplung ausschalten, LEADOF

Mit dem Ausschalten der Leitwertkopplung wird die Folgeachse wieder zur normalen Kommandoachse!

Axiale Leitwertkopplung und verschiedene Betriebszustände, RESET

Abhängig von der Einstellung im Maschinendatum werden Leitwertkopplungen mit RESET ausgeschaltet.

Beispiel Leitwertkopplung aus Synchronaktion

Bei einer Pressenanlage soll eine herkömmliche mechanische Kopplung zwischen einer Leitachse (Stempelwelle) und Achsen eines Transfersystems aus Transferachsen und Hilfsachsen durch ein elektronisches Koppelsystem ersetzt werden.

Es demonstriert, wie bei einer Pressenanlage ein mechanisches Transfersystem durch ein elektronisches Transfersystem ersetzt wird. Die Kopplungs- und Entkopplungsvorgänge sind als **statische Synchronaktionen** realisiert.

Von der Leitachse LW (Stempelwelle) werden Transferachsen und Hilfsachsen als Folgeachsen über Kurventabellen definiert gesteuert.

Folgeachsen

X Vorschub- bzw. Längsachse
YL Schließ- bzw. Querachse
ZL Hubachse
U Walzenvorschub, Hilfsachse
V Richtkopf, Hilfsachse
W Befettung, Hilfsachse

Aktionen

Als Aktionen treten in den Synchronaktionen z. B. auf:

- Einkoppeln, `LEADON(Folgeachse, Leitachse, Kurventabellen-Nummer)`
- Auskoppeln, `LEADOF(Folgeachse, Leitachse)`
- Istwertsetzen, `PRESETON(Achse, Wert)`
- Merker setzen, `$AC_MARKER[i] = Wert`
- Kopplungsart: reeller/virtueller Leitwert
- Anfahren von Achspositionen, `POS[Achse] = Wert`

Bedingungen

Als Bedingungen werden digitale schnelle Eingänge, Echtzeitvariablen `$AC_MARKER` und Positionsvergleiche, mit dem logischen Operator AND verknüpft, ausgewertet.

Hinweis

Im folgenden Beispiel wurden Zeilenwechsel, Einrückungen und **Fett**satz ausschließlich dafür verwendet, die Lesbarkeit der Programmierung zu erhöhen. Für die Steuerung ist alles unter einer Zeilennummer stehende einzeilig.

Kommentar

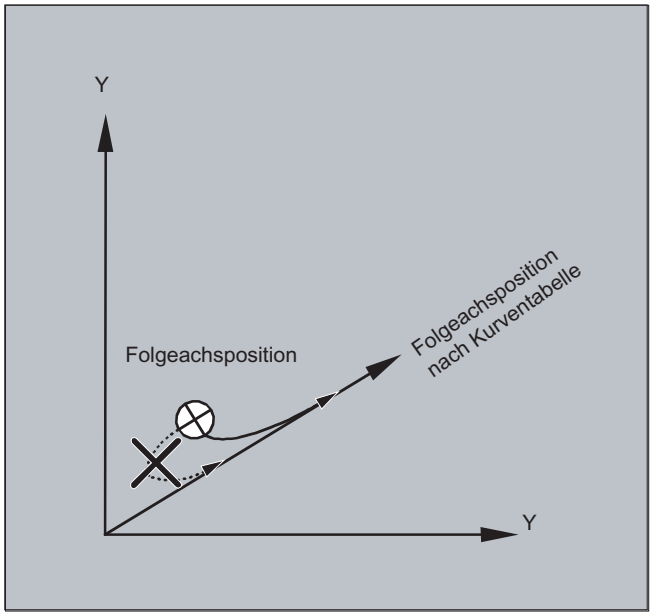
```
Programmcode                                                    Kommentar
                                                                ;  Definiert sämtliche statische Synchronaktionen.
                                                                ;  ****Marker rücksetzen
N2 $AC_MARKER[0]=0 $AC_MARKER[1]=0 $AC_MARKER[2]=0 $AC_MARKER[3]=0 $AC_MARKER[4]=0 $AC_MARKER[5]=0
$AC_MARKER[6]=0 $AC_MARKER[7]=0
                                                                ;  **** E1 0=>1 Kopplung Transfer EIN
N10 IDS=1 EVERY ($A_IN[1]==1) AND ($A_IN[16]==1) AND ($AC_MARKER[0]==0)
DO LEADON(X,LW,1) LEADON(YL,LW,2) LEADON(ZL,LW,3) $AC_MARKER[0]=1
                                                                ;  **** E1 0=>1 Kopplung Walzenvorschub EIN
N20 IDS=11 EVERY ($A_IN[1]==1) AND ($A_IN[5]==0) AND ($AC_MARKER[5]==0)
DO LEADON(U,LW,4) PRESETON(U,0) $AC_MARKER[5]=1
                                                                ;  **** E1 0->1 Kopplung Richtkopf EIN
N21 IDS=12 EVERY ($A_IN[1]==1) AND ($A_IN[5]==0) AND ($AC_MARKER[6]==0)
DO LEADON(V,LW,4) PRESETON(V,0) $AC_MARKER[6]=1
                                                                ;  **** E1 0->1 Kopplung Befettung EIN
N22 IDS=13 EVERY ($A_IN[1]==1) AND ($A_IN[5]==0) AND ($AC_MARKER[7]==0)
DO LEADON(W,LW,4) PRESETON(W,0) $AC_MARKER[7]=1
                                                                ;  **** E2 0=>1 Kopplung AUS
N30 IDS=3 EVERY ($A_IN[2]==1)
DO LEADOF(X,LW) LEADOF(YL,LW) LEADOF(ZL,LW) LEADOF(U,LW) LEADOF(V,LW) LEADOF(W,LW) $AC_MARKER[0]=0
$AC_MARKER[1]=0 $AC_MARKER[3]=0 $AC_MARKER[4]=0 $AC_MARKER[5]=0 $AC_MARKER[6]=0 $AC_MARKER[7]=0
....
N110 G04 F01
N120 M30
```

Beschreibung

Die Leitwertkopplung erfordert die Synchronisation von Leit- und Folgeachse. Diese Synchronisation kann nur erreicht werden, wenn die Folgeachse bei Einschalten der Leitwertkopplung innerhalb des Toleranzbereiches des aus der Kurventabelle berechneten Kurvenzugs steht.

Der Toleranzbereich für die Stellung der Folgeachse ist über Maschinendatum `MD 37200: COUPLE_POS_POL_COARSE A_LEAD_TYPE` definiert.

Befindet sich die Folgeachse mit dem Einschalten der Leitwertkopplung noch nicht an der entsprechenden Position, wird der Synchronlauf automatisch hergestellt, sobald sich der berechnete Positionssollwert für die Folgeachse der tatsächlichen Folgeachsposition nähert. Die Folgeachse wird dabei während des Synchronisationsvorganges in die Richtung verfahren, die durch die Sollgeschwindigkeit der Folgeachse (berechnet aus Leitachsgeschwindigkeit und nach Kurventabelle CTAB) definiert ist.

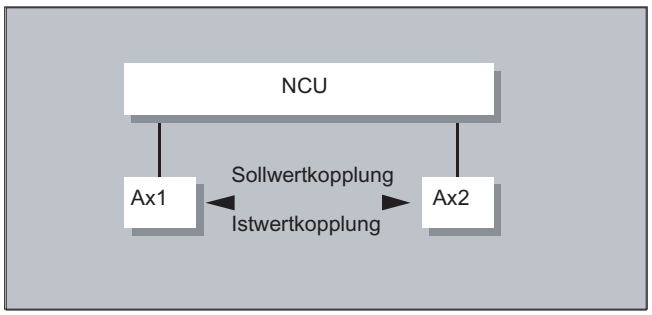


Kein Synchronlauf

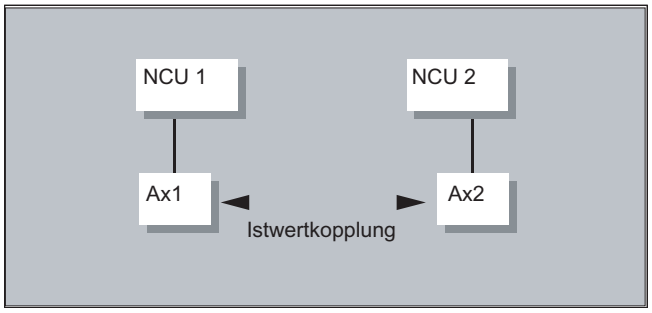
Entfernt sich die berechnete Folgeachssollposition mit Einschalten der Leitwertkopplung von der aktuellen Folgeachsposition, wird kein Synchronlauf hergestellt.

Ist- und Sollwertkupplung

Die Sollwertkopplung liefert im Vergleich zur Istwertkopplung einen besseren Synchronlauf zwischen Leit- und Folgeachse und ist deshalb standardmäßig voreingestellt.



Sollwertkopplung ist nur möglich, wenn Leit- und Folgeachse von derselben NCU interpoliert werden. Bei einer externen Leitachse kann die Folgeachse nur über Istwerte an die Leitachse gekoppelt werden.



Eine **Umschaltung** ist über das Settingdatum `$SA_LEAD_TYPE` möglich.

Das Umschalten zwischen Ist- und Sollwertkopplung sollte immer bei Stillstand der Folgeachse erfolgen. Denn nur im Stillstand wird nach dem Umschalten neu synchronisiert.

Anwendungsbeispiel

Das Lesen der Istwerte kann bei großen Maschinenerschütterungen nicht fehlerfrei erfolgen. Beim Einsatz der Leitwertkopplung im Pressentransfer kann es daher in den Arbeitsschritten mit größten Erschütterungen notwendig werden, von Istwertkopplung auf Sollwertkopplung umzuschalten.

Leitwertsimulation bei Sollwertkopplung

Über Maschinendatum lässt sich der Interpolator für die Leitachse vom Servo trennen. Damit können bei Sollwertkopplung Sollwerte ohne tatsächliche Bewegung der Leitachse erzeugt werden.

Die über Sollwertkopplung erzeugten Leitwerte sind zur Benutzung z. B. in Synchronaktionen aus folgenden Variablen lesbar:

```
- $AA_LEAD_P                                    Leitwert Position
- $AA_LEAD_V                                    Leitwert Geschwindigkeit
```

Leitwerte erzeugen

Leitwerte können wahlweise mit anderen selbst programmierten Verfahren erzeugt werden. Die so erzeugten Leitwerte werden in die Variable

```
- $AA_LEAD_SP                                   Leitwert Position
- $AA_LEAD_SV                                   Leitwert Geschwindigkeit
```

geschrieben und aus ihnen gelesen. Zur Benutzung dieser Variablen muss das Settingdatum `$SA_LEAD_TYPE = 2` gesetzt werden.

Status der Kopplung

Im NC-Teileprogramm können Sie den Kopplungsstatus mit folgender Systemvariablen abfragen:

`$AA_COUP_ACT[Achse]`
0: Keine Kopplung aktiv
16: Leitwertkopplung aktiv

Status-Verwaltung bei Synchronaktionen

Schalt- und Koppelvorgänge werden über Echtzeitvariablen:

`$AC_MARKER[i] = n`
verwaltet mit:
i Merker-Nummer
n Statuswert

9.4 Elektronisches Getriebe (EG)

Funktion

Mit Hilfe der Funktion "Elektronisches Getriebe" ist es möglich, die Bewegung einer **Folgeachse** nach linearem Bewegungssatz abhängig von bis zu fünf **Leitachsen** zu steuern. Die Zusammenhänge zwischen den Leitachsen und der Folgeachse sind je Leitachse durch den Koppelfaktor definiert.

Der berechnete Folgeachs-Bewegungsanteil wird aus den einzelnen Leitachsen-Bewegungsanteilen multipliziert mit den jeweiligen Koppelfaktoren durch Addition gebildet. Bei der Aktivierung eines EG-Achsverbundes kann die Synchronisation der Folgeachse auf eine definierte Position veranlasst werden. Ein Getriebeverband kann aus dem Teileprogramm:

- definiert,
- eingeschaltet,
- ausgeschaltet,
- gelöscht

werden.

Die Folgeachsbewegung kann wahlweise abgeleitet werden aus den

- Sollwerten der Leitachsen sowie den
- Istwerten der Leitachsen.

Als Erweiterung können auch nichtlineare Zusammenhänge zwischen den Leitachsen und der Folgeachse über **Kurventabellen** (siehe Kapitel Bahnverhalten) realisiert werden. Elektronische Getriebe können kaskadiert werden, d. h. die Folgeachse eines Elektronischen Getriebes kann Leitachse für ein weiteres Elektronisches Getriebe sein.

9.4.1 Elektronisches Getriebe definieren (EGDEF)

Funktion

Ein EG-Achsverband wird durch die Angabe der Folgeachse und mindestens einer, jedoch höchstens fünf Leitachsen mit dem jeweiligen Kopplungstyp festgelegt.

Voraussetzung

Voraussetzung für eine EG-Achsverband-Definition:

Für die Folgeachse darf noch keine Achskopplung definiert sein (ggf. muss eine bestehende vorher mit `EGDEL` gelöscht werden).

Syntax

```
EGDEF(Folgeachse,Leitachse1,Kopplungstyp1,Leitachse2,Kopplungstyp2,...)
```

Bedeutung

`EGDEF`	Definition eines elektronischen Getriebes
`Folgeachse`	Achse, die von Leitachsen beeinflusst wird
`Leitachse1 ,...,Leitachse5`	Achsen, die die Folgeachse beeinflussen
`Kopplungstyp1 ,...,Kopplungstyp5`	Kopplungstyp Der Kopplungstyp muss nicht für alle Leitachsen gleich sein und ist daher für jede Leitachse einzeln anzugeben.

Wert:	Bedeutung:
0	Die Folgeachse wird beeinflusst vom **Istwert** der entsprechenden Leitachse.
1	Die Folgeachse wird beeinflusst vom **Sollwert** der entsprechenden Leitachse.

Hinweis

Die Koppelfaktoren werden bei der Definition des EG-Kopplungsverbandes mit Null vorbesetzt.

Hinweis

`EGDEF` löst Vorlaufstopp aus. Die Getriebedefinition mit `EGDEF` ist auch dann unverändert zu verwenden, wenn bei Systemen eine oder mehrere Leitachsen über **Kurventabelle** auf die Folgeachse einwirken.

Beispiel

Programmcode	**Kommentar**
`EGDEF(C,B,1,Z,1,Y,1)`	`; Definition eines EG-Achsverbandes. Die Leitachsen B, Z, Y beeinflussen die Folgeachse C über den Sollwert.`

9.4.2 Elektronisches Getriebe einschalten (EGON, EGONSYN, EGONSYNE)

Funktion

Für das Einschalten eines EG-Achsverbandes existieren 3 Varianten.

Syntax

Variante 1:

Der EG-Achsverband wird ohneSynchronisation selektiv eingeschaltet mit:
`EGON(FA,"Satzwechselmodus",LA1,Z1,N1,LA2,Z2,N2,...,LA5,Z5,N5)`

Variante 2:

Der EG-Achsverband wird mitSynchronisation selektiv eingeschaltet mit:
`EGONSYN(FA,"Satzwechselmodus",SynPosFA,[,LAi,SynPosLAi,Zi,Ni])`

Variante 3:

Der EG-Achsverband wird mitSynchronisation selektiv eingeschaltet und der Anfahrmodus vorgegeben mit:
`EGONSYNE(FA,"Satzwechselmodus",SynPosFA,Anfahrmodus[,LAi,SynPosLAi,Zi,Ni])`

Bedeutung

Variante 1:

`FA`	Folgeachse	
`Satzwechselmodus`	Folgende Modi können benutzt werden:	
	`"NOC"`	Satzwechsel erfolgt sofort
	`"FINE"`	Satzwechsel erfolgt bei "Synchronlauf fein"
	`"COARSE"`	Satzwechsel erfolgt bei "Synchronlauf grob"
	`"IPOSTOP"`	Satzwechsel erfolgt bei sollwertseitigem Synchronlauf
`LA1, ... LA5`	Leitachsen	
`Z1, ... Z5`	Zähler für den Koppelfaktor i	
`N1, ... N5`	Nenner für den Koppelfaktor i	
	Koppelfaktor i = Zähler i/Nenner i	

Es dürfen nur die Leitachsen programmiert werden, die zuvor mit `EGDEF` spezifiziert worden sind. Es muss mindestens eine Leitachse programmiert werden.

Variante 2:

`FA`	Folgeachse
`Satzwechselmodus`	Folgende Modi können benutzt werden:
	`"NOC"` Satzwechsel erfolgt sofort
	`"FINE"` Satzwechsel erfolgt bei "Synchronlauf fein"
	`"COARSE"` Satzwechsel erfolgt bei "Synchronlauf grob"
	`"IPOSTOP"` Satzwechsel erfolgt bei sollwertseitigem Synchronlauf
`[,LAi,SynPosLAi,Zi,Ni]`	(Eckige Klammern nicht schreiben)
	Mind. 1, max. 5 Folgen von:
`LA1, ... LA5`	Leitachsen
`SynPosLAi`	Synchronposition für die i. Leitachse
`Z1, ... Z5`	Zähler für den Koppelfaktor i
`N1, ... N5`	Nenner für den Koppelfaktor i
	Koppelfaktor i = Zähler i/Nenner i

Es dürfen nur Leitachsen programmiert werden, die zuvor mit `EGDEF` spezifiziert worden sind. Durch die programmierten "Synchronpositionen" für die Folgeachse (`SynPosFA`) und für die Leitachsen (`SynPosLA`) werden Positionen definiert, in denen der Koppelverband als *synchron* gilt. Sofern sich das elektronische Getriebe beim Einschalten nicht in synchronem Zustand befindet, fährt die Folgeachse auf ihre definierte Synchronposition.

Variante 3:

Die Parameter entsprechen denen der Variante 2 zuzüglich:

`Anfahrmodus`	Folgende Modi können benutzt werden:
	`"NTGT"` Nächste Zahnlücke zeitoptimiert anfahren
	`"NTGP"` Nächste Zahnlücke wegoptimiert anfahren
	`"ACN"` Rundachse in negativer Drehrichtung verfahren absolut
	`"ACP"` Rundachse in positiver Drehrichtung verfahren absolut
	`"DCT"` Zeitoptimiert zur programmierten Synchronposition
	`"DCP"` Wegoptimiert zur programmierten Synchronposition

Die Variante 3 hat nur Auswirkungen auf Modulo-Folgeachsen, die an Modulo-Leitachsen gekoppelt sind. Zeitoptimierung berücksichtigt die Geschwindigkeitsgrenzen der Folgeachse.

Weitere Informationen

Beschreibung der Einschaltvarianten

Variante 1:

Die Positionen der Leitachsen sowie der Folgeachse zum Zeitpunkt des Einschaltens werden gespeichert als "Synchronpositionen". Die "Synchronpositionen" können mit den Systemvariablen `$AA_EG_SYN` gelesen werden.

Variante 2:

Wenn Moduloachsen im Koppelverband sind, werden ihre Positionswerte modulo reduziert. Damit ist gewährleistet, dass die nächstmögliche Synchronposition angefahren wird (sog. *relative Synchronisation*: z. B. die nächste Zahnlücke). Wenn für die Folgeachse nicht "Freigabe Folgeachsüberlagerung" Nahtstellensignal DB(30 +Achsnummer), DBX 26 Bit 4 gegeben ist, wird nicht auf die Synchronposition gefahren. Stattdessen wird das Programm beim EGONSYN-Satz angehalten und es wird der selbstlöschende Alarm 16771 gemeldet, solange bis das o.g. Signal gesetzt wird.

Variante 3:

Der Zahnabstand (Grad) ergibt sich aus: 360 * Zi/Ni. Für den Fall, dass die Folgeachse zum Aufrufzeitpunkt steht, liefert wegoptimiert das gleiche Verhalten wie zeitoptimiert.

Bei bereits fahrender Folgeachse wird mit `NTGP` unabhängig von der aktuellen Geschwindigkeit der Folgeachse auf die nächste Zahnlücke synchronisiert. Bei bereits fahrender Folgeachse wird mit `NTGT` abhängig von der aktuellen Geschwindigkeit der Folgeachse auf die nächste Zahnlücke synchronisiert. Die Achse wird dazu ggf. auch abgebremst.

Kurventabellen

Wird für eine der Leitachsen eine **Kurventabelle** verwendet, so muss:

Ni	der Nenner des Koppelfaktors linearer Kopplungen auf 0 gesetzt werden. (Nenner 0 wäre für lineare Kopplungen unzulässig). Nenner Null ist für die Steuerung das Kennzeichen, dass
Zi	als Nummer der zu verwendenden Kurventabelle interpretiert werden soll. Die Kurventabelle mit der angegebenen Nummer muss zum Einschaltzeitpunkt bereits definiert sein.
LAi	Die Angabe der Leitachse entspricht der Leitachsangabe bei Kopplung über Koppelfaktor (lineare Kopplung).

Weitere Hinweise über die Nutzung von Kurventabellen und das Kaskadieren von Elektronischen Getrieben und deren Synchronisierung finden Sie in:
Literatur:
Funktionshandbuch Sonderfunktionen; Achskopplungen und ESR (M3), Kapitel "Mitschleppen und Leitwertkopplung".

Verhalten des Elektronischen Getriebes bei Power On, RESET, Betriebsartenwechsel, Suchlauf

- Nach Power On ist **keine** Kopplung aktiv.
- Aktive Kopplungen bleiben über RESET und Betriebsartenwechsel erhalten.
- Bei Satzsuchlauf werden Befehle zum Schalten, Löschen, Definieren des Elektronischen Getriebes nicht ausgeführt und nicht aufgesammelt, sondern übergangen.

Systemvariablen des Elektronischen Getriebes

Mit Hilfe der Systemvariablen des Elektronischen Getriebes kann das Teileprogramm aktuelle Zustände eines EG-Achsverbandes ermitteln und ggf. darauf reagieren.

Die Systemvariablen des Elektronischen Getriebes sind wie folgt gekennzeichnet:

$AA_EG_ ...

oder

$VA_EG_ ...

Literatur:
Handbuch der Systemvariablen

9.4.3 Elektronisches Getriebe ausschalten (EGOFS, EGOFC)

Funktion

Für das Ausschalten eines aktiven EG-Achsverbandes existieren 3 Varianten.

Programmierung

Variante 1:

Syntax	Bedeutung
`EGOFS(Folgeachse)`	Das elektronische Getriebe wird ausgeschaltet. Die Folgeachse wird zum Stillstand abgebremst. Der Aufruf löst Vorlaufstopp aus.

Variante 2:

Syntax	Bedeutung
`EGOFS(Folgeachse,Leitachse1,…,Leitachse5)`	Diese Parametrierung des Befehls erlaubt **selektiv** den Einfluss einzelner Leitachsen auf die Bewegung der Folgeachse zu beseitigen.

Es muss wenigstens eine Leitachse angegeben werden. Der Einfluss der angegebenen Leitachsen auf die Folgeachse wird gezielt ausgeschaltet. Der Aufruf löst Vorlaufstopp aus. Verbleiben noch aktive Leitachsen, so läuft die Folgeachse unter deren Einfluss weiter. Sind alle Leitachseneinflüsse auf diese Weise ausgeschaltet, so wird die Folgeachse zum Stillstand abgebremst.

Variante 3:

Syntax	Bedeutung
`EGOFC(Folgespindel)`	Das elektronische Getriebe wird ausgeschaltet. Die Folgespindel läuft mit der zum Ausschaltzeitpunkt aktuellen Drehzahl/Geschwindigkeit weiter. Der Aufruf löst Vorlaufstopp aus.

Hinweis

Diese Variante ist nur für Spindeln erlaubt.

9.4.4 Definition eines Elektronischen Getriebes löschen (EGDEL)

Funktion

Ein EG-Achsverband muss ausgeschaltet sein, bevor seine Definition gelöscht werden kann.

Programmierung

Syntax	Bedeutung
`EGDEL(Folgeachse)`	Die Kopplungsdefinition des Achsverbandes wird gelöscht. Es wird bis zum Erreichen der maximalen Anzahl gleichzeitig aktivierter Achsverbände wieder möglich, weitere Achsverbände mit `EGDEF` neu zu definieren. Der Aufruf löst Vorlaufstopp aus.

9.4.5 Umdrehungsvorschub (G95) / Elektronisches Getriebe (FPR)

Funktion

Mit dem FPR-Befehl kann auch die Folgeachse eines Elektronischen Getriebes als vorschubbestimmende Achse des Umdrehungsvorschubes angegeben werden. Für diesen Fall gilt folgendes Verhalten:

- Der Vorschub ist abhängig von der Sollgeschwindigkeit der Folgeachse des Elektronischen Getriebes.
- Die Sollgeschwindigkeit wird berechnet aus den Geschwindigkeiten der Leitspindeln und Modulo-Leitachsen (die nicht Bahnachsen sind) und deren zugeordneten Koppelfaktoren.
- Geschwindigkeitsanteile von linearen bzw. nicht Modulo-Leitachsen und überlagerte Bewegungen der Folgeachse werden nicht berücksichtigt.

9.5 Synchronspindel

Funktion

Im Synchronbetrieb gibt es eine Leitspindel (LS) und eine Folgespindel (FS), das sog. **Synchronspindelpaar**. Die Folgespindel folgt bei aktiver Kopplung (Synchronbetrieb) den Bewegungen der Leitspindel entsprechend dem festgelegten Funktionszusammenhang.

Die Synchronspindelpaare lassen sich für jede Maschine sowohl mit Hilfe von kanalspezifischen Maschinendaten fest projektieren oder über das CNC-Teileprogramm anwendungsspezifisch definieren. Je NC-Kanal sind bis zu 2 Synchronspindelpaare gleichzeitig betreibbar.

Die Kopplung kann aus dem Teileprogramm

- definiert bzw. geändert
- eingeschaltet
- ausgeschaltet
- gelöscht

werden.

Darüber hinaus kann abhängig vom Softwarestand

- auf die Synchronlaufbedingung gewartet
- das Satzwechselverhalten verändert
- die Kopplungsart entweder Sollwertkopplung oder Istwertkopplung ausgewählt oder der Winkelversatz zwischen Leit- und Folgespindel vorgegeben
- beim Einschalten der Kopplung eine vorhergehende Programmierung der Folgespindel übernommen
- entweder eine gemessene oder eine bereits bekannte Synchronlaufabweichung korrigiert

werden.

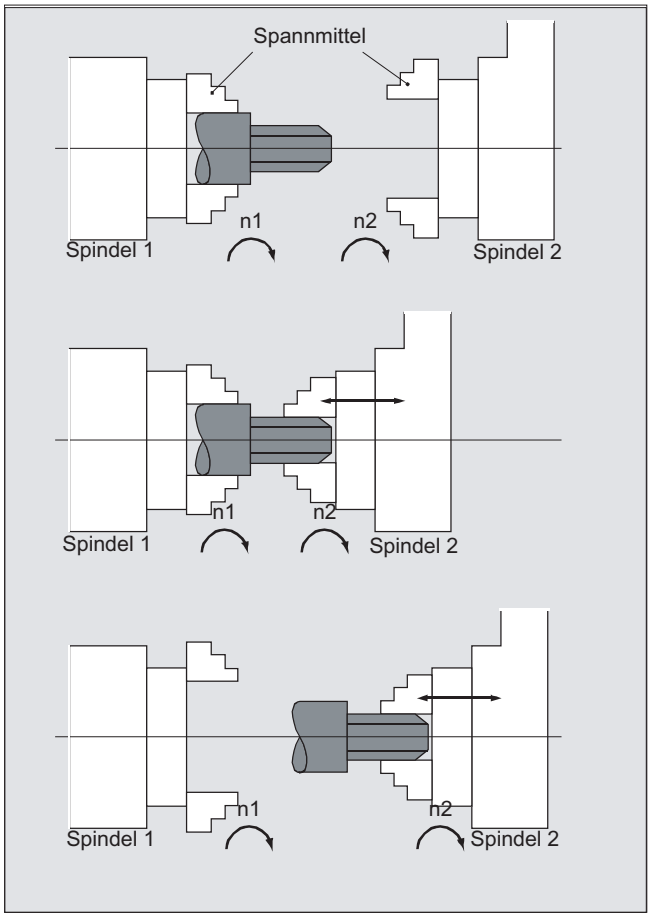
9.5.1 Synchronspindel: Programmierung (COUPDEF, COUPDEL, COUPON, COUPONC, COUPOF, COUPOFS, COUPRES, WAITC)

Funktion

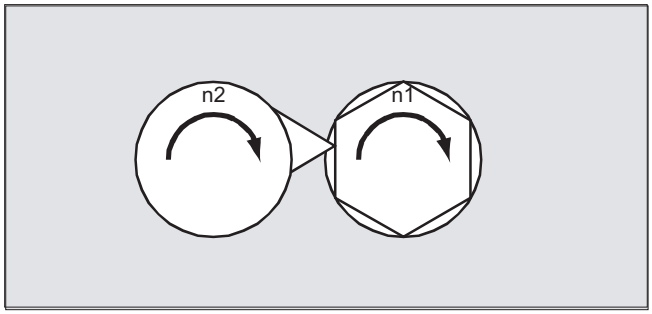
Die Funktion Synchronspindel ermöglicht ein synchrones Verfahren zweier Spindeln (Folgespindel FS und Leitspindel LS), z. B. zur fliegenden Werkstückübergabe.

Die Funktion bietet folgende Modi:

- Drehzahlsynchronität ($n_{FS} = n_{LS}$)
- Lagesynchronität ($\phi_{FS} = \phi_{LS}$)
- Lagesynchronität mit Winkelversatz ($\phi_{FS} = \phi_{LS} + \Delta\phi$)



Durch Vorgabe eines Übersetzungsverhältnisses ungleich 1 zwischen Leit- und Folgespindel ist auch eine Mehrkantbearbeitung (Polygondrehen) möglich.



Syntax

```
COUPDEF(<FS>,<LS>,<ÜFS>,<ÜLS>,<Satzwechsel>,<Koppelart>)
COUPON(<FS>,<LS>,<POSFS>)
COUPONC(<FS>,<LS>)
COUPOF(<FS>,<LS>,<POSFS>,<POSLS>)
COUPOFS(<FS>,<LS>)
COUPOFS(<FS>,<LS>,<POSFS>)
COUPRES(<FS>,<LS>)
COUPDEL(<FS>,<LS>)
WAITC(<FS>,<Satzwechsel>,<LS>,<Satzwechsel>)
```

Hinweis

Verkürzte Schreibweise

Bei den Anweisungen COUPOF, COUPOFS, COUPRES und COUPDEL ist eine verkürzte Schreibweise ohne Angabe der Leitspindel möglich.

Bedeutung

COUPDEF:	Kopplung anwenderspezifisch definieren/ändern
COUPON:	Kopplung einschalten. Ausgehend von der aktuellen Drehzahl synchronisiert sich die Folgespindel auf die Leitspindel
COUPONC:	Kopplung beim Einschalten mit vorhergehender Programmierung von M3 S... oder M4 S... übernehmen. Eine Differenzdrehzahl der Folgespindel wird sofort übernommen.
COUPOF:	Kopplung ausschalten. • mit sofortigem Satzwechsel: COUPOF(<S2>,<S1>) • Satzwechsel erst nach Überfahren der Ausschaltposition(en) <POSFS> bzw. <POSLS>: COUPOF(<S2>,<S1>,<POSFS>) COUPOF(<S2>,<S1>,<POSFS>,<POSLS>)

COUPOFS:	Ausschalten einer Kopplung mit Stopp der Folgespindel. Satzwechsel schnellstmöglich mit sofortigen Satzwechsel: COUPOFS(<S2>,<S1>) Satzwechsel erst nach Überfahren der Ausschaltposition: COUPOFS(<S2>,<S1>,<POSFS>)
COUPRES:	Kopplungsparameter zurücksetzen auf projektierte MD und SD
COUPDEL:	Anwenderdefinierte Kopplung löschen
WAITC:	Warten Synchronlaufbedingung (NOC werden auf IPO bei Satzwechsel aufgehoben)
<FS>:	Bezeichnung der Folgespindel

Optionale Parameter:

<LS>:	Bezeichnung der Leitspindel Angabe mit Spindelnummer: z. B. S2, S1
<ÜFS>, <ÜLS>:	Übersetzungsverhältnis zwischen FS und LS. <ÜFS> = Zähler, <ÜLS> = Nenner Voreinstellung: <ÜFS> / <ÜLS> = 1.0 ; Angabe des Nenners optional
<Satzwechsel>:	Satzwechselverhalten Der Satzwechsel erfolgt: "NOC" sofort "FINE" mit Erreichen von "Synchronlauf fein" "COARSE" mit Erreichen von "Synchronlauf grob" "IPOSTOP" mit Erreichen von IPOSTOP, d. h. nach sollwertseitigem Synchronlauf (Voreinstellung) Das Satzwechselverhalten ist modal wirksam.
<Koppelart>:	Kopplungsart: Kopplung zwischen FS und LS "DV" Sollwertkopplung (Voreinstellung) "AV" Istwertkopplung "VV" Geschwindigkeitskopplung Die Kopplungsart ist modal wirksam.
<POSFS>:	Winkelversatz zwischen Leit- und Folgespindel Wertebereich: 0°… 359,999°
<POSFS>,<POSLS>:	Ausschaltpositionen von Folge- und Leitspindel "Der Satzwechsel wird nach überfahren der POS_{FS}, POS_{LS} freigegeben" Wertebereich: 0°… 359,999°

Beispiele

Beispiel 1: Arbeiten mit Leit- und Folgespindel

Programmierung	**Kommentar**
	; Leitspindel = Masterspindel = Spindel 1
	; Folgespindel = Spindel 2
N05 M3 S3000 M2=4 S2=500	; Leitspindel dreht mit 3000 U/min, Folgespindel dreht mit 500 U/min.
N10 COUPDEF(S2,S1,1,1,"NOC","Dv")	; Definition der Kopplung (kann auch projektiert werden).
...	
N70 SPCON	; Leitspindel in Lageregelung nehmen (Sollwertk.).
N75 SPCON(2)	; Folgespindel in Lageregelung nehmen.
N80 COUPON(S2,S1,45)	; Fliegend auf Offsetposition = 45 Grad einkoppeln.
...	
N200 FA[S2]=100	; Positioniergeschwindigkeit = 100 grd/min
N205 SPOS[2]=IC(-90)	; 90 Grad überlagert in negative Richtung fahren.
N210 WAITC(S2,"Fine")	; Warten auf Synchronlauf "fein".
N212 G1 X... Y... F...	; Bearbeitung
...	
N215 SPOS[2]=IC(180)	; 180 Grad überlagert in positive Richtung fahren.
N220 G4 S50	; Verweilzeit = 50 Umdrehungen der Masterspindel
N225 FA[S2]=0	; Projektierte Geschw. (MD) aktivieren.
N230 SPOS[2]=IC(-7200)	; 20 Umdrehungen. Mit projektierter Geschwindigkeit in negative Richtung fahren.
...	
N350 COUPOF(S2,S1)	; Fliegend auskoppeln, S=S2=3000
N355 SPOSA[2]=0	; FS bei Null Grad stoppen.
N360 G0 X0 Y0	
N365 WAITS(2)	; Warten auf Spindel 2.
N370 M5	; FS stoppen.
N375 M30	

Beispiel 2: Programmierung einer Differenzdrehzahl

```
Programmierung                  Kommentar
                                ; Leitspindel = Masterspindel = Spindel 1
                                ; Folgespindel = Spindel 2
N01 M3 S500                     ; Leitspindel dreht mit 500 U/min.
N02 M2=3 S2=300                 ; Folgespindel dreht mit 300 U/min.
...
N10 G4 F1                       ; Verweilzeit der Masterspindel.
N15 COUPDEF (S2,S1,-1)          ; Koppelfaktor mit Übersetzungsverhältnis -1:1
N20 COUPON (S2,S1)              ; Kopplung aktivieren. Die Drehzahl der Folgespindel ergibt sich aus
                                  der Drehzahl der Leitspindel und dem Koppelfaktor.
...
N26 M2=3 S2=100                 ; Programmierung einer Differenzdrehzahl.
```

Beispiel 3: Beispiele der Übernahme einer Bewegung zur Differenzdrehzahl

1. Kopplung bei vorhergehender Programmierung der Folgespindel mit COUPON einschalten

```
Programmierung                  Kommentar
                                ; Leitspindel = Masterspindel = Spindel 1
                                ; Folgespindel = Spindel 2
N05 M3 S100 M2=3 S2=200         ; Leitspindel dreht mit 100 U/min, Folgespindel mit 200 U/min.
N10 G4 F5                       ; Verweilzeit = 5 Sekunden der Masterspindel
N15 COUPDEF(S2,S1,1)            ; Übersetzungsverhältnis FS zu LS ist 1,0 (Voreinstellung).
N20 COUPON(S2,S1)               ; Fliegend auf Leitspindel einkoppeln.
N10 G4 F5                       ; Folgespindel dreht mit 100 U/min.
```

2. Kopplung bei vorhergehender Programmierung der Folgespindel mit COUPONC einschalten

```
Programmierung                  Kommentar
                                ; Leitspindel = Masterspindel = Spindel 1
                                ; Folgespindel = Spindel 2
N05 M3 S100 M2=3 S2=200         ; Leitspindel dreht mit 100 U/min, Folgespindel mit 200 U/min.
N10 G4 F5                       ; Verweilzeit = 5 Sekunden der Masterspindel
N15 COUPDEF(S2,S1,1)            ; Übersetzungsverhältnis FS zu LS ist 1,0 (Voreinstellung).
N20 COUPONC(S2,S1)              ; Fliegend auf Leitspindel einkoppeln und vorhergehende Drehzahl zu S2
                                  übernehmen.
N10 G4 F5                       ; S2 dreht mit 100U/min + 200U/min = 300U/min
```

3. Kopplung bei stehender Folgespindel mit COUPON einschalten

```
Programmierung                  Kommentar
                                ; Leitspindel = Masterspindel = Spindel 1
                                ; Folgespindel = Spindel 2
N05 SPOS=10 SPOS[2]=20          ; Folgespindel S2 im Positionierbetrieb.
N15 COUPDEF(S2,S1,1)            ; Übersetzungsverhältnis FS zu LS ist 1,0 (Voreinstellung).
N20 COUPON(S2,S1)               ; Fliegend auf Leitspindel einkoppeln.
N10 G4 F1                       ; Kopplung wird geschlossen, S2 bleibt auf 20 Grad stehen.
```

4. Kopplung bei stehender Folgespindel mit COUPONC einschalten

Hinweis

Positionier- oder Achsbetrieb

Befindet sich die Folgespindel vor dem Einkoppeln im Positionier- oder Achsbetrieb, dann verhält sich die Folgespindel bei COUPON(<FS>,<LS>) und COUPONC(<FS>,<LS>) gleich.

ACHTUNG

Leitspindel und Achsbetrieb

Befindet sich die Leitspindel vor der Definition der Kopplung im Achsbetrieb, wirkt auch nach dem Einschalten der Kopplung der Geschwindigkeitsgrenzwert aus Maschinendatum:

MD32000 $MA_MAX_AX_VELO (maximale Achsgeschwindigkeit)

Zur Vermeidung dieses Verhaltens muss die Achse vor der Definition der Kopplung in den Spindelbetrieb (M3 S... oder M4 S...) geschaltet werden.

Weitere Informationen

Synchronspindelpaar festlegen

Projektierte Kopplung:

Bei der projektierten Kopplung werden Leit- und Folgespindel über Maschinendatum festgelegt. Die projektierten Spindeln können im Teileprogramm nicht verändert werden. Die Parametrierung der Kopplung kann mit COUPDEF im Teileprogramm erfolgen (Voraussetzung: kein Schreibschutz festgelegt).

Anwenderdefinierte Kopplung:

Mit COUPDEF kann eine Kopplung im Teileprogramm neu definiert oder verändert werden. Ist bereits eine Kopplung aktiv, muss diese vor der Definition einer neuen Kopplung zuerst mit COUPDEL gelöscht werden.

Kopplung definieren: COUPDEF

Eine Kopplung wird vollständig definiert durch:

COUPDEF(<FS>,<LS>,<ÜFS>,<ÜLS>, Satzwechselverhalten, Koppelart)

Folgespindel (FS) und Leitspindel (LS)

Mit den Achsnamen für die Folge- (FS) und Leitspindel (LS) wird die Kopplung eindeutig bestimmt. Die Achsnamen müssen mit jeder Anweisung `COUPDEF` programmiert werden. Die anderen Kopplungsparameter sind modal wirksam und müssen nur programmiert werden, wenn sie geändert werden.

Beispiel:

```
COUPDEF(S2,S1)
```

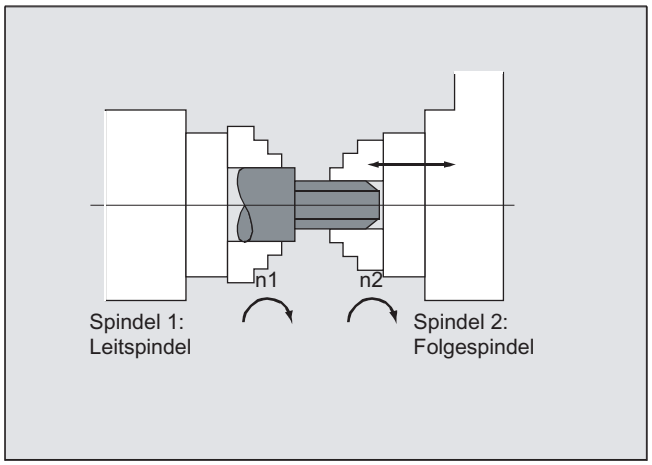
Übersetzungsverhältnis ÜFS / ÜLS

Das Übersetzungsverhältnis wird als Drehzahlverhältnis zwischen Folgespindel (Zähler) und Leitspindel (Nenner) angegeben. Der Zähler muss programmiert werden. Wird kein Nenner programmiert, wird Nenner = 1.0 gesetzt.

Beispiel:

Folgespindel S2 und Leitspindel S1, Übersetzungsverhältnis = 1 / 4 = 0.25.

```
COUPDEF(S2,S1,1.0, 4.0)
```



Hinweis

Das Übersetzungsverhältnis kann auch bei eingeschalteter Kopplung und drehenden Spindeln verändert werden.

Satzwechselverhalten NOC, FINE, COARSE, IPOSTOP

Bei der Programmierung des Satzwechselverhaltens ist folgende verkürzte Schreibweise möglich:

- `"NO"`: sofort (Voreinstellung)
- `"FI"`: mit Erreichen von "Synchronlauf fein"
- `"CO"`: mit Erreichen von "Synchronlauf grob"
- `"IP"`: mit Erreichen von IPOSTOP, d. h. nach sollwertseitigem Synchronlauf

Kopplungsart DV, AV

⚠ VORSICHT

Die Kopplungsart darf nur bei ausgeschalteter Kopplung verändert werden!

Synchronbetrieb einschalten COUPON, POSFS

- Einschalten der Kopplung mit beliebigem Winkelbezug zwischen LS und FS:
  - `COUPON(S2,S1)`
  - `COUPON(S2,S1,<POSFS>)`
  - `COUPON(S2)`
- Einschalten der Kopplung mit Winkelversatz <`POSFS>`

  Zur positionssynchronen Kopplung für profilierte Werkstücke.
  `<POSFS>`bezieht sich auf die 0°-Position der Leitspindel in positiver Drehrichtung
  Wertebereich `<POSFS>`: 0°… 359,999°
  - `COUPON(S2,S1,30)`

  Auf diese Weise können Sie auch bei schon aktivierter Kopplung den Winkelversatz ändern.

Positionieren der Folgespindel

Bei eingeschalteter Synchronspindelkopplung lassen sich auch Folgespindeln unabhängig von der durch die Leitspindel ausgelösten Bewegung im Bereich ±180° positionieren.

Positionierung SPOS

Die Folgespindel kann mit `SPOS=...` interpoliert werden.

Beispiel:

```
SPOS[2]=IC(-90)
```

Weitere Informationen zu `SPOS` finden sich in:
Literatur:
Programmierhandbuch Grundlagen

Diffferenzdrehzahl M3 S... oder M4 S...

Eine Differenzdrehzahl entsteht durch vorzeichenbehaftete Überlagerung zweier Drehzahlquellen und wird zur Folgespindel z. B. mit `S<n>=...` oder `M<n>=3`, `M<n>=4` im Drehzahlsteuerbetrieb während einer aktiven Synchronspindelkopplung erneut programmiert. Dabei wird dieser Drehzahlanteil über den Koppelfaktor von der Leitspindel abgeleitet und der Folgespindel vorzeichenrichtig dazu addiert.

Hinweis

Mit der Drehrichtung `M3` oder `M4` muss auch die Drehzahl `S...` neu programmiert werden, weil sonst die fehlende Programmierung durch einen Alarm gemeldet wird.

Weitere Informationen zur Differenzdrehzahl siehe:
Literatur:
Funktionshandbuch Erweiterungsfunktionen; Synchronspindel (S3)

Differenzdrehzahl bei COUPONC

Übernahme einer Bewegung zur Differenzdrehzahl

Durch das Einschalten einer Synchronspindelkopplung mit `COUPONC` wird eine aktuell wirksame Drehzahl der Folgespindel (`M3 S...` oder `M4 S...`) überlagert.

Hinweis

Freigabe der Überlagerung

Eine Überlagern einer Spindeldrehzahl (`M3 S...` oder `M4 S...`) durch Synchronspindelkopplung `COUPONC` wird nur wirksam, wenn die Überlagerung freigegeben ist.

Dynamikeinschränkung der Leitspindel

Die Dynamik der Leitspindel muss so weit eingeschränkt werden, dass bei einer Überlagerung der Folgespindel deren Dynamikgrenzwerte nicht überschritten werden.

Geschwindigkeit, Beschleunigung: FA, ACC, OVRA, VELOLIMA

Axiale Geschwindigkeit und Beschleunigung einer Folgespindeln sind programmierbar mit:

- `FA[SPI(S<n>)]` bzw. `FA[S<n>]` (axiale Geschwindigkeit)
- `ACC[SPI(S<n>)]` bzw. `ACC[S<n>]` (axiale Beschleunigung)
- `OVRA[SPI(S<n>)]` bzw. `OVRA[S<n>]` (axialer Override)
- `VELOLIMA[SPI(S<n>)]` bzw. `VELOLIMA[S<n>]` (axiale Geschwindigkeitsüberhöhung bzw. -reduktion)

Mit `<n>` = 1, 2, 3, ... (Spindelnummer der Folgespindeln)

Literatur:
Programmierhandbuch Grundlagen

Hinweis

Beschleunigungsanteil JERKLIMA[S<n>]

Die Programmierung einer axialen Geschwindigkeitsüberhöhung bzw. -reduktion ist bei Spindeln aktuell nicht wirksam.

Weitere Informationen zur Projektierung der axialen Dynamik finden sich in:
Literatur:
Funktionshandbuch Erweiterungsfunktionen; Rundachsen (R2)

Programmierbares Satzwechselverhalten WAITC

Mit `WAITC` kann das Satzwechselverhalten, z. B. nach Änderung von Kopplungsparametern oder Positioniervorgängen, mit unterschiedlichen Synchronlaufbedingungen (grob, fein, `IPOSTOP`) vorgegeben werden. Sind keine Synchronlaufbedingungen angegeben, gilt das bei der Definition `COUPDEF` angegebene Satzwechselverhalten.

Beispiel:

Warten auf das Erreichen der Synchronlaufbedingung entsprechend `COUPDEF`

```
WAITC( )
```

Warten auf das Erreichen der Synchronlaufbedingung `FINE` bei Folgespindel S2 und `COARSE` bei Folgespindel S4:

```
WAITC(S2,"FINE",S4,"COARSE")
```

Kopplung ausschalten COUPOF

Mit COUPOF kann das Ausschaltverhalten der Kopplung vorgegeben werden:

- Ausschalten der Kopplung mit sofortigem Satzwechsel:
  - COUPOF(S2,S1) (mit Angabe der Leitspindel)
  - COUPOF(S2) (ohne Angabe der Leitspindel)
- Ausschalten der Kopplung nach Überfahren von Ausschaltpositionen. Der Satzwechsel erfolgt nach dem Überfahren der Ausschaltpositionen.
  - COUPOF(S2,S1,150) (Ausschaltposition FS: 150°)
  - COUPOF(S2,S1,150,30) (Ausschaltposition FS: 150°, LS: 30°)

Kopplung ausschalten mit Stopp der Folgespindel COUPOFS

Mit COUPOFS kann das Ausschaltverhalten der Kopplung mit Stopp der Folgespindel vorgegeben werden:

- Ausschalten der Kopplung mit Stopp der Folgespindel und sofortigem Satzwechsel:
  - COUPOFS(S2,S1) (mit Angabe der Leitspindel)
  - COUPOFS(S2) (ohne Angabe der Leitspindel)
- Ausschalten der Kopplung nach Überfahren von Ausschaltpositionen mit Stopp der Folgespindel. Der Satzwechsel erfolgt nach dem Überfahren der Ausschaltpositionen.
  - COUPOFS(S2,S1,150) (Ausschaltposition FS: 150°)

Kopplungen löschen COUPDEL

Mit COUPDEL wird die Kopplung gelöscht:

- COUPDEL(S2,S1) (mit Angabe der Leitspindel)
- COUPDEL(S2) (ohne Angabe der Leitspindel)

Kopplungsparameter zurücksetzen COUPRES

Mit COUPRES werden die in den Maschinen- und Settingdaten parametrierten Werte der Kopplung aktiviert:

- COUPRES(S2,S1) (mit Angabe der Leitspindel)
- COUPRES(S2) (ohne Angabe der Leitspindel)

Systemvariablen

Aktueller Kopplungszustand der Folgespindel

Der aktuelle Kopplungszustand einer Folgespindel kann über folgende Systemvariable gelesen werden:

`$AA_COUP_ACT[<FS>]`

Wert	Bedeutung
0	keine Kopplung aktiv
4	Synchronspindelkopplung aktiv
Hinweis Andere Werte der Systemvariablen beziehen sich auf den Achsbetrieb **Literatur:** Listenhandbuch Systemvariablen	

Aktueller Winkelversatz

Der aktuelle Winkelversatz einer Folgespindel bezüglich der Leitspindel kann über folgende Systemvariable gelesen werden:

- `$AA_COUP_OFFS[<FS>]` (Sollwertseitiger Winkelversatz)
- `$VA_COUP_OFFS[<FS>]` (Istwertseitiger Winkelversatz)

Hinweis

Nach Wegnahme der Reglerfreigabe bei eingeschalteter Kopplung und Nachführbetrieb stellt sich nach erneuter Erteilung der Reglerfreigabe ein anderer Positionsoffset ein als der ursprünglich programmierte Wert. In diesem Fall kann der veränderte Positionsoffset gelesen werden und ggf. im Teileprogramm korrigiert werden.

9.6 Master-/Slave-Verband (MASLDEF, MASLDEL, MASLON, MASLOF, MASLOFS)

Funktion

Die Master-/Slave-Kopplung vor SW 6.4 gestattet das Einkoppeln der Slave-Achsen auf ihre Masterachse nur im Stillstand der beteiligten Achsen.

Die Erweiterung des SW-Standes 6.5 erlaubt das Koppeln und Trennen von **drehenden**, drehzahlgesteuerten Spindeln und die dynamische Projektierung.

Syntax

`MASLON(Slv1,Slv2,..., )`	
`MASLOF(Slv1,Slv2,..., )`	
`MASLDEF(Slv1,Slv2,..., Masterachse)`	Erweiterung für dynamische Projektierung
`MASLDEL(Slv1,Slv2,..., )`	Erweiterung für dynamische Projektierung
`MASLOFS(Slv1, Slv2, ..., )`	Erweiterung für Slave-Spindel

Hinweis

Bei `MASLOF/MASLOFS` entfällt der implizite Vorlaufstopp. Bedingt durch den fehlenden Vorlaufstopp liefern die $P-Systemvariablen für die Slave-Achsen bis zum Zeitpunkt erneuter Programmierung keine aktualisierten Werte.

Bedeutung

Allgemein

`MASLON`	Eine temporäre Kopplung einschalten.
`MASLOF`	Eine aktive Kopplung trennen. Bei Spindeln sind die Erweiterungen zu beachten.

Erweiterung dynamische Projektierung

`MASLDEF`	Kopplung anwenderdefiniert über Maschinendaten oder auch aus dem Teileprogramm heraus anlegen/ändern.
`MASLOFS`	Die Kopplung analog zu `MASLOF` trennen und die Slave-Spindel automatisch abbremsen.

MASLDEL	Master/Slave-Achsverband trennen und Definition des Verbandes löschen.
Slv1, Slv2, ...	Slave-Achsen, die von einer Masterachse geführt werden.
Masterachse	Achse, die in einem Master/Slave-Verband definierte Slave-Achsen führt.

Beispiele

Beispiel 1: Dynamische Projektierung einer Master/Slave-Kopplung

Dynamische Projektierung einer Master/Slave-Kopplung aus dem Teileprogramm heraus:

Die nach einer Achscontainerdrehung relevante Achse soll zur Masterachse werden.

```
Programmcode          Kommentar
MASLDEF(AUX,S3)       ; S3 Master für AUX
MASLON(AUX)           ; Kopplung ein für AUX
M3=3 S3=4000          ; Drehrichtung rechts
MASLDEL(AUX)          ; Projektierung löschen und Trennen der Kopplung
AXCTSWE(CT1)          ; Containerdrehung
```

Beispiele

Beispiel 2: Istwertkopplung einer Slave-Achse

Istwertkopplung einer Slave-Achse auf den gleichen Wert der Master-Achse durch PRESETON.

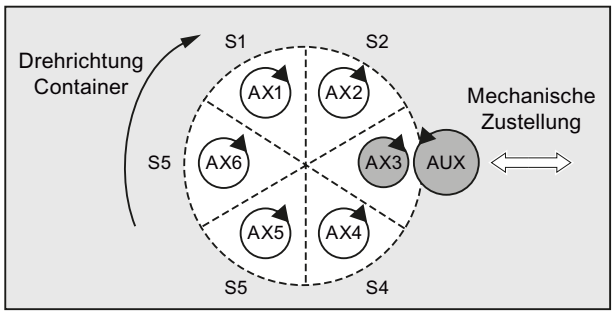
Bei einer Permanenten Master/Slave-Kopplung soll an der SLAVE Achse der Istwert durch PRESETON verändert werden.

```
Programmcode                                    Kommentar
N37262 $MA_MS_COUPLING_ALWAYS_ACTIVE[AX2]=0     ; Permanente Kopplung kurz
                                                  ausschalten.
N37263 NEWCONF
N37264 STOPRE
MASLOF(Y1)                                      ; Temporäre Kopplung aus.
N5 PRESETON(Y1,0,Z1,0,B1,0,C1,0,U1,0)           ; Istwert setzen der nicht
                                                  referierten Slave Achsen,
                                                  da diese mit Power On
                                                  aktiviert sind.
N37262 $MA_MS_COUPLING_ALWAYS_ACTIVE[AX2]=1     ; Permanente Kopplung
                                                  aktivieren.
N37263 NEWCONF
```

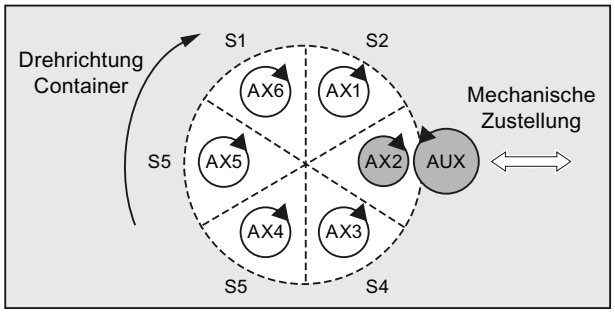
Beispiel 3: Kopplungssequenz Lage 3/Container CT1

Damit die Kopplung nach der Containerdrehung mit einer anderen Spindel geschlossen werden kann, muss vorher die alte Kopplung getrennt, die Projektierung gelöscht und die neue Kopplung projektiert werden.

Ausgangssituation:



Nach Drehung um einen Slot:



Literatur:

Funktionshandbuch Erweiterungsfunktionen; Mehrere Bedientafelfronten und NCUs (B3), Kapitel: "Achscontainer"

Weitere Informationen

Allgemein

MASLOF	Bei Spindeln im Drehzahlsteuerbetrieb wird diese Anweisung unmittelbar ausgeführt. Die zu diesem Zeitpunkt drehenden Slave-Spindeln behalten ihre Drehzahlen bis zur erneuten Drehzahlprogrammierung bei.

Erweiterung dynamische Projektierung

MASLDEF	Definition eines Master-/Slave-Verbandes aus dem Teileprogramm heraus. Vorher erfolgte die Definition ausschließlich über Maschinendaten.
MASLDEL	Die Anweisung hebt die Zuordnung der Slave-Achsen zur Masterachse auf und trennt gleichzeitig, analog zu MASLOF die Kopplung auf. Die in den Maschinendaten vereinbarten Master-/Slave-Definitionen bleiben erhalten.
MASLOFS	MASLOFS kann dazu verwendet werden, um Slave-Spindeln beim Trennen der Kopplung automatisch abzubremsen. Bei Achsen und Spindeln im Positionierbetrieb wird die Kopplung nur im Stillstand geschlossen und getrennt.

Hinweis

Für die Slave-Achse kann der Istwert durch PRESETON auf den gleichen Wert der Master Achse synchronisiert werden. Dazu muss die dauerhafte Master-/Slave-Kopplung kurzfristig ausgeschaltet werden um den Istwert der nicht referierten Slave-Achse mit Power On auf den Wert der Master-Achse zu setzen. Danach wird die dauerhafte Kopplung wieder hergestellt.

Die dauerhafte Master-/Slave-Kopplung wird mit der MD-Einstellung MD37262 $MA_MS_COUPLING_ALWAYS_ACTIVE = 1 aktiviert und hat keine Auswirkung auf die Sprachbefehle der temporären Kopplung.

Koppelverhalten bei Spindeln

Bei Spindeln im Drehzahlsteuerbetrieb wird das Koppelverhalten von MASLON, MASLOF, MASLOFS und MASLDEL explizit über das Maschinendatum MD37263 $MA_MS_SPIND_COUPLING_MODE festgelegt.

In der Standardeinstellung mit MD37263 = 0 erfolgt das Einkoppeln und Trennen der Slave-Achsen ausschließlich im Stillstand der beteiligten Achsen. MASLOFS entspricht dem MASLOF.

Bei MD37263 = 1 wird die Koppelanweisung unmittelbar, und damit auch in der Bewegung ausgeführt. Die Kopplung wird bei MASLON sofort geschlossen und bei MASLOFS oder MASLOF sofort getrennt. Die zu diesem Zeitpunkt drehenden Slave-Spindeln werden bei MASLOFS automatisch abgebremst und behalten bei MASLOF ihre Drehzahlen bis zur erneuten Drehzahlprogrammierung bei.

Bewegungssynchronaktionen 10

10.1 Grundlagen

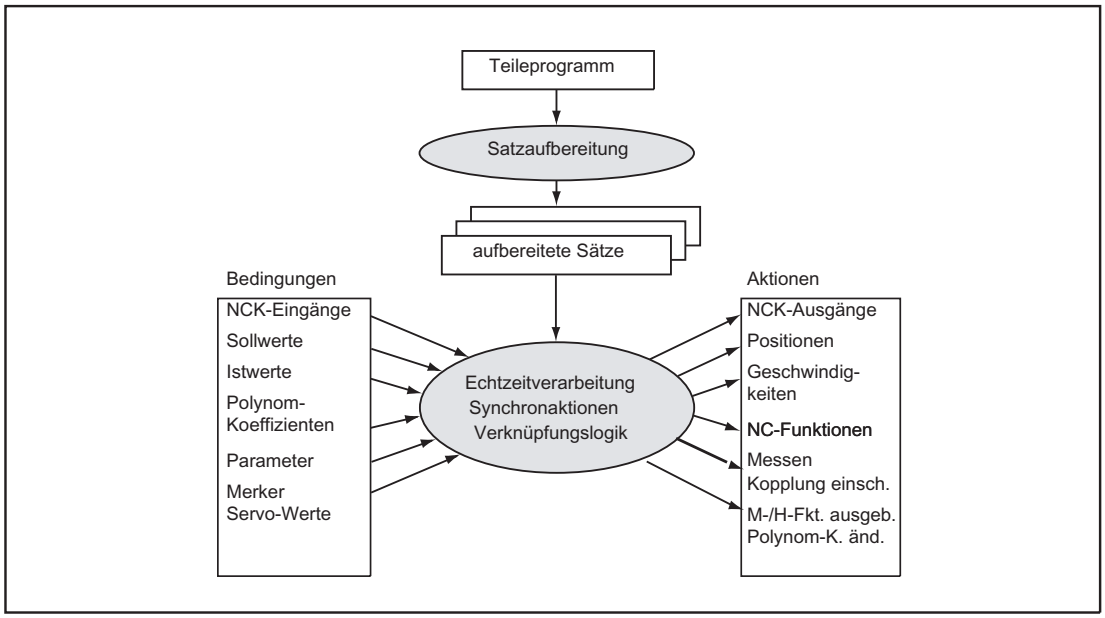
Funktion

Synchronaktionen bieten die Möglichkeit, synchron zu Bearbeitungssätzen Aktionen auszuführen.

Der Ausführungszeitpunkt der Aktionen kann über Bedingungen definiert werden. Die Bedingungen werden im Interpolationstakt überwacht. Die Aktionen stellen somit eine Reaktion auf Echtzeitereignisse dar, ihre Ausführung ist nicht an Satzgrenzen gebunden.

Zusätzlich enthält eine Synchronaktion Angaben zu ihrer Lebensdauer und zur Abfragehäufigkeit für die programmierten Hauptlaufvariablen und damit zur Ausführungshäufigkeit der zu startenden Aktionen. Dadurch kann eine Aktion nur einmal oder aber zyklisch (jeweils im Interpolationstakt) angestoßen werden.

Mögliche Anwendungen

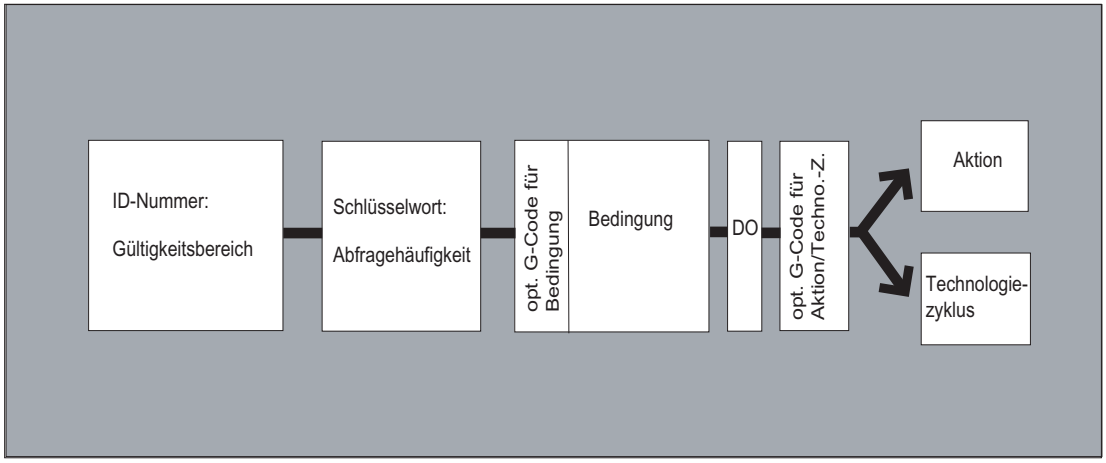


- Optimierung laufzeitkritischer Anwendungen (z. B. Werkzeugwechsel)
- Schnelle Reaktion auf externe Ereignisse
- AC-Regelungen programmieren
- Sicherheitsfunktionen einrichten
- ....

Programmierung

Eine Synchronaktion steht allein im Satz und wirkt ab dem nächsten ausführbaren Satz einer Maschinenfunktion (z. B. Verfahrbewegung mit G0, G1, G2, G3).

Synchronaktionen bestehen aus bis zu 5 Befehlselementen mit unterschiedlichen Aufgaben:



Syntax:

```
DO <Aktion1> <Aktion2> …
<SCHLÜSSELWORT> <Bedingung> DO <Aktion1> <Aktion2> …
ID=<n> <SCHLÜSSELWORT> <Bedingung> DO <Aktion1> <Aktion2> …
IDS=<n> <SCHLÜSSELWORT> <Bedingung> DO <Aktion1> <Aktion2> …
```

Bedeutung:

DO	Anweisung zur Auslösung der programmierten Aktion(en) Wirkt nur bei erfüllter <Bedingung> (sofern programmiert). → Siehe " Aktionen "
<Aktion1> <Aktion2> …	Zu startende Aktion(en) Beispiele: • Variable zuweisen • Technologiezyklus starten
<SCHLÜSSELWORT>	Über das Schlüsselwort (WHEN, WHENEVER, FROM oder EVERY) wird die zyklische Prüfung der <Bedingung> einer Synchronaktion definiert. → Siehe " Zyklische Prüfung der Bedingung "

<Bedingung>	Verknüpfungslogik für Hauptlaufvariablen Die Bedingung wird im IPO-Takt geprüft.
ID=<n> bzw. IDS=<n>	Identifikationsnummer Mit der Identifikationsnummer werden der Gültigkeitsbereich und die Position innerhalb der Bearbeitungsreihenfolge festgelegt. → Siehe " Gültigkeitsbereich und Bearbeitungsreihenfolge "

Koordinierung von Synchronaktionen/Technologiezyklen

Zur Koordinierung von Synchronaktionen/Technologiezyklen stehen folgende Befehle zur Verfügung:

Befehl	Bedeutung
CANCEL(<n>)	Synchronaktionen löschen → Siehe " Synchronaktion löschen "
LOCK(<n>)	Synchronaktionen sperren
UNLOCK(<n>)	Synchronaktionen freischalten
RESET	Technologiezyklus zurücksetzen
	Bezüglich LOCK, UNLOCK und RESET: → siehe " Sperren, Freischalten, Zurücksetzen "

Beispiel

Programmcode	Kommentar
`WHEN $AA_IW[Q1]>5 DO M172 H510`	`; Wenn der Istwert der Achse Q1 5 mm übersteigt, werden die Hilfsfunktionen M172 und H510 an die PLC-Nahtstelle ausgegeben.`

10.1.1 Gültigkeitsbereich und Bearbeitungsreihenfolge (ID, IDS)

Funktion

Gültigkeitsbereich

Der Gültigkeitsbereich einer Synchronaktion wird festgelegt durch die Kennung ID bzw. IDS:

Keine Modal-ID:	Satzweise wirksame Synchronaktion im Automatik-Betrieb
ID:	Modal wirksame Synchronaktion im Automatik-Betrieb bis Programmende
IDS:	Statische Synchronaktion, modal wirksam in jeder Betriebsart, auch über Programmende

Anwendungen

- AC-Schleifen im JOG-Betrieb
- Verknüpfungslogik für Safety Integrated
- Überwachungsfunktionen, Reaktionen auf Maschinenzustände in allen Betriebsarten

Bearbeitungsreihenfolge

Modal und statisch wirksame Synchronaktionen werden in der Reihenfolge ihrer `ID`- bzw. `IDS`-Nummer (`ID=<n>` bzw. `IDS=<n>`) im Interpolationstakt bearbeitet.

Satzweise wirksame Synchronaktionen (ohne ID-Nummer) werden nach der Abarbeitung der modal wirksamen Synchronaktionen in der programmierten Reihenfolge bearbeitet.

Hinweis

Über Maschinendateneinstellungen können modal wirksame Synchronaktionen gegenüber Änderungen oder Löschungen geschützt werden (→ Maschinenhersteller!).

Programmierung

Syntax	Bedeutung
keine Modal-ID	Die Synchronaktion ist nur im **Automatik-Betrieb** wirksam. Sie gilt nur für den folgenden ausführbaren Satz (Satz mit Bewegungsanweisung oder sonstiger Maschinenaktion), ist also **satzweise** wirksam. Beispiel: `WHEN $A_IN[3]==TRUE DO $A_OUTA[4]=10`
`ID=<n> ...`	Die Synchronaktion wirkt in den folgenden Sätzen **modal** und kann durch `CANCEL(<n>)` ausgeschaltet oder durch Programmierung einer neuen Synchronaktion mit gleicher ID überschrieben werden. Die im `M30`-Satz aktiven Synchronaktionen verzögern das Programmende. ID-Synchronaktionen wirken nur im **Automatik-Betrieb**. Wertebereich für **<n>**: 1 ... 255 Beispiel: `ID=2 EVERY $A_IN[1]==1 DO POS[X]=0`
`IDS=<n>`	Die statischen Synchronaktionen wirken **modal in allen Betriebsarten**. Sie bleiben auch über das Programmende aktiv und können direkt nach Power On mit einem ASUP aktiviert werden. Dadurch ist es möglich, Aktionen zu aktivieren, die unabhängig von der gewählten Betriebsart in der NC ablaufen sollen. Wertebereich für **<n>**: 1 ... 255 Beispiel: `IDS=1 EVERY $A_IN[1]==1 DO POS[X]=100`

10.1.2 Zyklische Prüfung der Bedingung (WHEN, WHENEVER, FROM, EVERY)

Funktion

Über ein Schlüsselwort wird die zyklische Prüfung der Bedingung einer Synchronaktion definiert. Ist kein Schlüsselwort programmiert, werden die Aktionen der Synchronaktion in jedem IPO-Takt ausgeführt.

Schlüsselworte

kein Schlüsselwort	Die Ausführung der Aktion ist an keine Bedingung geknüpft. Die Aktion wird zyklisch in jedem Interpolationstakt ausgeführt.
`WHEN`	Die Bedingung wird so lange in jedem Interpolationstakt abgefragt, bis sie einmal erfüllt ist; die zugehörige Aktion wird genau dann einmalig ausgeführt.
`WHENEVER`	Die Bedingung wird in jedem Interpolationstakt zyklisch überprüft. Die zugehörige Aktion wird in jedem Interpolationstakt ausgeführt, solange die Bedingung erfüllt ist.
`FROM`	Die Bedingung wird in jedem Interpolationstakt überprüft, bis sie einmal erfüllt ist. Die Aktion wird daraufhin so lange ausgeführt, wie die Synchronaktion aktiv ist, d. h. auch dann, wenn die Bedingung nicht mehr erfüllt ist.
`EVERY`	Die Bedingung wird in jedem Interpolationstakt abgefragt. Die Aktion wird immer dann einmalig ausgeführt, wenn die Bedingung erfüllt ist. Flankensteuerung: Die Aktion wird wieder ausgeführt, wenn die Bedingung vom Zustand FALSE auf den Zustand TRUE wechselt.

Hauptlaufvariablen

Die verwendeten Variablen werden im Interpolationstakt ausgewertet. Hauptlaufvariablen in Synchronaktionen lösen keinen Vorlaufstopp aus.

Auswertung:

Treten in einem Teileprogramm Hauptlaufvariablen auf (z.B. Istwert, Stellung eines digitalen Ein- oder Ausgangs usw.), wird der Vorlauf angehalten, bis der vorhergehende Satz abgearbeitet ist und die Werte der Hauptlaufvariablen vorliegen.

Beispiele

Beispiel 1: Kein Schlüsselwort

Programmcode	Kommentar
`DO $A_OUTA[1]=$AA_IN[X]`	`; Istwertausgabe auf Analogausgang.`

Beispiel 2: WHENEVER

Programmcode	Kommentar
`WHENEVER $AA_IM[X] > 10.5*SIN(45) DO …`	`; Vergleich mit im Vorlauf berechnetem Ausdruck.`
`WHENEVER $AA_IM[X] > $AA_IM[X1] DO …`	`; Vergleich mit weiterer Hauptlaufvariable.`
`WHENEVER ($A_IN[1]==1) OR ($A_IN[3]==0) DO ...`	`; Zwei miteinander verknüpfte Vergleiche.`

Beispiel 3: EVERY

Programmcode	Kommentar
`ID=1 EVERY $AA_IM[B]>75 DO POS[U]=IC(10) FA[U]=900`	`; Immer wenn der Istwert der Achse B im MKS den Wert 75 überschreitet, soll die U-Achse mit axialem Vorschub um 10 weiterpositionieren.`

Weitere Informationen

Bedingung

Die Bedingung stellt einen logischen Ausdruck dar, der über Boole'sche Operatoren beliebig aufgebaut sein kann. Boole'sche Ausdrücke sollen immer in Klammern angegeben werden.

Die Bedingung wird im Interpolationstakt überprüft.

Vor der Bedingung kann ein G-Code angegeben werden. Damit kann erreicht werden, dass unabhängig vom gerade aktiven Teileprogrammzustand für die Auswertung der Bedingung und die auszuführende Aktion/Technologiezyklus definierte Einstellungen bestehen. Die Abkopplung der Synchronaktionen vom Programmumfeld ist erforderlich, weil Synchronaktionen zu beliebigen Zeitpunkten aufgrund erfüllter Auslösebedingungen ihre Aktionen in definiertem Ausgangszustand ausführen sollen.

Anwendungsfälle

Festlegung der Maßsysteme für Bedingungsauswertung und Aktion durch G-Codes `G70`, `G71`, `G700`, `G710`.

Ein angegebener G-Code bei der Bedingung gilt für die Auswertung der Bedingung und für die Aktion, wenn bei der Aktion kein eigener G-Code angegeben ist.

Pro Bedingungssteil darf nur **ein G-Code** der G-Code-Gruppe programmiert werden.

Mögliche Bedingungen

- Vergleich von Hauplaufvariablen (analoge/digitale Ein-/Ausgänge, u.a.)
- Boole'sche Verknüpfung zwischen Vergleichsergebnissen
- Berechnung von Echtzeitausdrücken
- Zeit/Entfernung vom Satzanfang
- Entfernung vom Satzende
- Messwerte, Messergebnisse
- Servo-Werte
- Geschwindigkeiten, Achsstatus

10.1.3 Aktionen (DO)

Funktion

In Synchronaktionen können eine oder mehrere Aktionen programmiert werden. Sämtliche in einem Satz programmierte Aktionen werden im gleichen Interpolationstakt aktiv.

Syntax

```
DO <Aktion1> <Aktion2> ...
```

Bedeutung

`DO`	Löst bei erfüllter Bedingung eine Aktion oder einen Technologiezyklus aus.
`<Aktion>`	Bei erfüllter Bedingung gestartete Aktion wie z. B. Variable zuweisen, Achskopplung einschalten, NCK-Ausgänge setzen, M-, S- und H-Funktionen ausgeben, programmierten G-Code vorgeben, ...

G-Codes sind in Synchronaktionen für die Aktionen/Technologiezyklen programmierbar. Dieser G-Code gibt für alle Aktionen im Satz und Technologiezyklen ggf. einen anderen G-Code als den bei der Bedingung gesetzten vor. Sind Technologiezyklen im Aktionsteil, so gilt der G-Code auch nach Abschluss des Technologiezyklus' für alle darauf folgenden Aktionen bis zum nächsten G-Code modal weiter.

Pro Aktionsteil darf nur **ein G-Code** der G-Code-Gruppe (`G70`, `G71`, `G700`, `G710`) programmiert werden.

Beispiel: Synchronaktion mit zwei Aktionen

Programmcode	Kommentar
`WHEN $AA_IM[Y]>=35.7 DO M135 $AC_PARAM=50`	`; Falls die Bedingung erfüllt ist, wird M135 an PLC ausgegeben und der Override auf 50% gesetzt.`

10.2 Operatoren für Bedingungen und Aktionen

Vergleiche (==, <>, <, >, <=, >=)	In Bedingungen können Variablen oder Teilausdrücke verglichen werden. Das Ergebnis ist immer vom Datentyp BOOL. Zulässig sind alle bekannten Vergleichsoperatoren.
Boole'sche Operatoren (NOT, AND, OR, XOR)	Variablen, Konstanten oder Vergleiche können mit den bekannten Bool'schen Operatoren miteinander verknüpft werden.
Bitweise Operatoren (B_NOT, B_AND, B_OR, B_XOR)	Möglich sind die bitweisen Operatoren B_NOT, B_AND, B_OR, B_XOR.
Grundrechenarten (+, -, *, /, DIV, MOD)	Hauptlaufvariablen können durch die Grundrechenarten miteinander oder mit Konstanten verknüpft werden.
Mathematische Funktionen (SIN, COS, TAN, ASIN, ACOS, ABS, TRUNC, ROUND, LN, EXP, ATAN2, POT, SQRT, CTAB, CTABINV).	Auf Variablen vom Datentyp REAL können mathematische Funktionen angewendet werden.
Indizierung	Indizierung ist mit Hauptlaufausdrücken möglich.

Beispiel

- **Grundrechenarten verknüpft**

Es gilt Punkt-vor-Strich Rechnung, die Klammerung von Ausdrücken ist zulässig. Die Operatoren DIV und MOD sind auch für den Datentyp REAL zulässig

Programmierung	**Kommentar**
DO $AC_PARAM[3] = $A_INA[1]-$AA_IM[Z1]	; ;Subtraktion zweier ;Hauptlaufvariablen
WHENEVER $AA_IM[x2] < $AA_IM[x1]-1.9 DO $A_OUT[5] = 1	; ;Subtraktion einer Konstanten von Variablen
DO $AC_PARAM[3] = $INA[1]-4*SIN(45.7 $P_EP[Y])*R4	;Konstanter Ausdruck, im Vorlauf berechnet

- **Mathematische Funktionen**

Programmierung	**Kommentar**
DO $AC_PARAM[3] = COS($AC_PARAM[1])	; ;

- **Echtzeitausdrücke**

Programmierung	Kommentar
ID=1 WHENEVER ($AA_IM[Y]>30) AND ($AA_IM[Y]<40) DO $AA_OVR[S1]=80	; Auswahl eines Positions-Fenster
ID=67 DO $A_OUT[1]=$A_IN[2] XOR $AN_MARKER[1]	; 2 boole'sche Signale auswerten
ID=89 DO $A_OUT[4]=$A_IN[1] OR ($AA_IM[Y]>10)	; Ergebnis eines Vergleichs ausgeben

- **Hauptlaufvariable indiziert**

Programmierung	Kommentar
WHEN…DO $AC_PARAM[$AC_MARKER[1]] = 3	;
Unzulässig ist	;
$AC_PARAM[1] = $P_EP[$AC_MARKER]	;

10.3 Hauptlaufvariablen für Synchronaktionen

10.3.1 Systemvariablen

Funktion

Mit Hilfe von Systemvariablen können Daten der NC gelesen und geschrieben werden. Systemvariable werden in Vorlauf- und Hauptlaufvariable unterschieden. Vorlaufvariable werden immer zum Vorlaufzeitpunkt ausgeführt. Hauptlaufvariable ermitteln ihren Wert immer bezüglich des aktuellen Hauptlaufzustandes.

Bennenung

Der Name von Systemvariablen beginnt meist mit einem $–Zeichen:

Vorlaufvariablen:

$M...	Maschinendaten
$S...	Settingdaten, Schutzbereiche
$T...	Werkzeugverwaltungsdaten
$P...	Programmierte Werte, Vorlaufdaten
$C...	Zyklenvariablen der ISO-Hüllzyklen
$O...	Optionsdaten
R ...	R-Parameter

Hauptlaufvariablen:

$$A...	Aktuelle Hauptlaufdaten
$$V...	Servo-Daten
$R...	R-Parameter

Ein 2. Buchstabe beschreibt die Zugriffsmöglichkeit auf die Variable:

N...	NCK-globaler Wert (allgemein gültiger Wert)
C...	Kanalspezifischer Wert
A...	Achsspezifischer Wert

Der 2. Buchstabe wird meistens nur für Hauptlaufvariablen verwendet. Vorlaufvariablen wie z. B. $P_ werden meist ohne 2. Buchstaben ausgeführt.

Dem Prefix ($ gefolgt von einem oder zwei Buchstaben) folgt immer ein Unterstrich und der nachfolgende Variablenname (meistens als englische Bezeichnung oder Abkürzung).

Datentypen

Hauptlaufvariablen können folgende Datentypen haben:

INT	Integer für ganzzahlige Werte mit Vorzeichen
REAL	Real für gebrochenrationale Zahlen
BOOL	Boolean TRUE und FALSE
CHAR	ASCII-Zeichen
STRING	Zeichenkette mit alphanumerischen Zeichen
AXIS	Achsadressen und Spindeln

Vorlaufvariablen können zusätzlich folgenden Datentyp haben:

FRAME	Koordinatentransformationen

Variablen-Felder

Systemvariablen können als 1- bis 3-dimensionale Felder angelegt werden.

Folgende Datentypen werden unterstützt: BOOL, CHAR, INT, REAL, STRING, AXIS

Der Datentyp der Indizes kann vom Typ INT und AXIS sein, wobei diese beliebig sortiert werden können.

STRING-Variablen können nur 2-dimensional angelegt werden.

Beispiele für Felddefinitionen:

```
DEF BOOL $AA_NEWVAR[x,y,2]
DEF CHAR $AC_NEWVAR[2,2,2]
DEF INT $AC_NEWVAR[2,10,3]
DEF REAL $AA_VECTOR[x,y,z]
DEF STRING $AC_NEWSTRING[3,3]
DEF AXIS $AA_NEWAX[x,3,y]
```

Hinweis

Die Anzeige von 3-dimensionalen Systemvariablen ist uneingeschränkt möglich, wenn es zu der Systemvariablen eine BTSS-Variable gibt.

10.3.2 Implizite Typwandlung

Funktion

Bei Wertzuweisungen und Parameterübergaben können Variablen unterschiedlicher Datentypen zugewiesen oder übergeben werden.

Die implizite Typwandlung löst eine interne Typenkonvertierung von Werten aus.

Mögliche Typkonvertierungen

nach	REAL	INT	BOOL	CHAR	STRING	AXIS	FRAME
von							
REAL	ja	ja*	ja[1])	–	–	–	–
INT	ja	ja	ja[1])	–	–	–	–
BOOL	ja	ja	ja	–	–	–	–

Erklärungen

* Bei Typumwandlung von REAL nach INT wird bei gebrochenem Wert >= 0.5 aufgerundet, ansonsten wird abgerundet (vgl. Funktion ROUND).
Bei Werteüberschreitungen wird ein Alarm ausgelöst.

1) Wert <> 0 entspricht TRUE, Wert == 0 entspricht FALSE

Ergebnisse

```
Typwandlung von REAL oder INTEGER nach BOOL
Ergebnis BOOL = TRUE          wenn der Wert von REAL oder INTEGER ungleich Null ist
Ergebnis BOOL = FALSE         wenn der Wert von REAL oder INTEGER gleich Null ist
Typwandlung von BOOL nach REAL oder INTEGER
Ergebnis REAL TRUE            wenn der Wert von BOOL = TRUE (1) ist
Ergebnis INTEGER = TRUE       wenn der Wert von BOOL = TRUE (1) ist
Typwandlung von BOOL nach REAL oder INTEGER
Ergebnis REAL FALSE)          wenn der Wert von BOOL = FALSE (0) ist
Ergebnis INTEGER = FALSE      wenn der Wert von BOOL = FALSE (0) ist
```

Beispiele impliziter Typwandlungen

```
Typwandlung von INTEGER nach BOOL
$AC_MARKER[1] = 561
ID=1 WHEN $A_IN[1] == TRUE DO $A_OUT[0]=$AC_MARKER[1]

Typwandlung von REAL nach BOOL
R401 = 100.542
WHEN $A_IN[0] == TRUE DO $A_OUT[2]=$R401

Typwandlung von BOOL nach INTEGER
ID=1 WHEN $A_IN[2] == TRUE DO $AC_MARKER[4] = $A_OUT[1]]

Typwandlung von BOOL nach REAL
R401 = 100.542
WHEN $A_IN[3] == TRUE DO $R10 = $A_OUT[3]
```

10.3.3 GUD-Variablen

Synchronaktionsfähige GUD-Variablen

Neben spezifischen Systemvariablen können in Synchronaktionen auch vordefinierte globale Synchronaktions-Anwendervariablen (Synchronaktions-GUD) verwendet werden. Die Anzahl der dem Anwender zur Verfügung stehenden Synchronaktions-GUD wird Datentyp- und Zugriffs-spezifisch über folgende Maschinendaten parametriert:

- MD18660 $MM_NUM_SYNACT_GUD_**REAL**[<x>] = <Anzahl>
- MD18661 $MM_NUM_SYNACT_GUD_**INT**[<x>] = <Anzahl>
- MD18662 $MM_NUM_SYNACT_GUD_**BOOL**[<x>] = <Anzahl>
- MD18663 $MM_NUM_SYNACT_GUD_**AXIS**[<x>] = <Anzahl>
- MD18664 $MM_NUM_SYNACT_GUD_**CHAR**[<x>] = <Anzahl>
- MD18665 $MM_NUM_SYNACT_GUD_**STRING**[<x>] = <Anzahl>

Über den Index <x> wird der Datenbaustein (Zugriffrechte), über den Wert <Anzahl> die Anzahl von Synchronaktions-GUD des jeweiligen Datentyps (REAL, INT, ...) angegeben. Im jeweiligen Datenbaustein wird daraufhin für jeden Datentyp ein 1-dimensionale Feldvariable mit folgendem Namens-Schema angelegt: SYG_<Datentyp><Zugriffsrecht>[<Index>]:

Index <x>		Datentyp (MD18660 ... MD18665)					
	Baustein	**REAL**	**INT**	**BOOL**	**AXIS**	**CHAR**	**STRING**
0	SGUD	SYG_RS[i]	SYG_IS[i]	SYG_BS[i]	SYG_AS[i]	SYG_CS[i]	SYG_SS[i]
1	MGUD	SYG_RM[i]	SYG_IM[i]	SYG_BM[i]	SYG_AM[i]	SYG_CM[i]	SYG_SM[i]
2	UGUD	SYG_RU[i]	SYG_IU[i]	SYG_BU[i]	SYG_AU[i]	SYG_CU[i]	SYG_SU[i]
3	GUD4	SYG_R4[i]	SYG_I4[i]	SYG_B4[i]	SYG_A4[i]	SYG_C4[i]	SYG_S4[i]
4	GUD5	SYG_R5[i]	SYG_I5[i]	SYG_B5[i]	SYG_A5[i]	SYG_C5[i]	SYG_S5[i]
5	GUD6	SYG_R6[i]	SYG_I6[i]	SYG_B6[i]	SYG_A6[i]	SYG_C6[i]	SYG_S6[i]
6	GUD7	SYG_R7[i]	SYG_I7[i]	SYG_B7[i]	SYG_A7[i]	SYG_C7[i]	SYG_S7[i]
7	GUD8	SYG_R8[i]	SYG_I8[i]	SYG_B8[i]	SYG_A8[i]	SYG_C8[i]	SYG_S8[i]
8	GUD9	SYG_R9[i]	SYG_I9[i]	SYG_B9[i]	SYG_A9[i]	SYG_C9[i]	SYG_S9[i]
mit i = 0 bis (<Anzahl> - 1) Baustein: _N_DEF_DIR/_N_ ... _DEF, z.B.für SGUD ⇒ _N_DEF_DIR/_N_**SGUD**_DEF							

Eigenschaften

Synchronaktions-GUD haben folgende Eigenschaften:

- Synchronaktions-GUD können in Synchronaktionen und Teileprogrammen / Zyklen gelesen und geschrieben werden
- Auf Synchronaktions-GUD kann über BTSS zugegriffen werden
- Synchronaktions-GUD werden auf der Bedienoberfläche des HMI im Bedienbereich "Parameter" angezeigt
- Synchronaktions-GUD können auf HMI im Wizard, in der Variablenansicht und im Variablenprotokoll verwendet werden
- Die Feldgröße bei Synchronaktions-GUD vom Typ STRING ist feste auf 32 (31 Zeichen + \0) definiert.
- Auch wenn manuell keine Definitionsdateien für globale Anwenderdaten (GUD) angelegt wurden, können über Maschinendaten definierte Synchronaktions-GUD im jeweiligen GUD-Baustein von HMI aus gelesen werden.

ACHTUNG

Anwendervariablen (GUD, PUD, LUD) können mit dem gleichen Namen wie Synchronaktions-GUD nur definiert werden (`DEF ... SYG_xy`), wenn keine Synchronaktions-GUD mit gleichem Namen parametriert sind (MD18660 - MD18665) . Diese vom Anwender definierten GUD können **nicht** in Synchronaktionen verwendet werden.

Zugriffsrechte

Die in einer GUD-Definitionsdatei definierten Zugriffsrechte bleiben weiterhin gültig und beziehen sich nur auf die in dieser GUD-Definitionsdatei definierten GUD-Variablen.

Löschverhalten

Wird der Inhalt einer bestimmten GUD-Definitionsdatei neu aktiviert, wird zunächst der alte GUD-Datenbaustein im aktiven Filesystem gelöscht. Die projektierten Synchronaktions-GUD werden dabei ebenfalls zurückgesetzt. Dieser Vorgang ist auch über HMI im Bedienbereich "Dienste" > "Anwenderdaten (GUD) definieren und aktivieren" möglich.

10.3.4 Default-Achsbezeichner (NO_AXIS)

Funktion

Variablen oder Parameter vom Typ AXIS, die nicht mit einem Wert initialisiert wurden, können mit definierten Default-Achsbezeichnern versehen werden. Undefinierte Achsvariablen werden mit diesem Default-Wert initialisiert.

Nicht-initialisierte gültige Achsnamen werden über eine Abfrage der Variable "NO_AXIS" in Synchronaktionen erkannt. Diesen nicht-initialisierten Achsbezeichner wird der über ein Maschinendatum projektierte Default-Achsbezeichner zugewiesen.

Maschinenhersteller

Über Maschinendaten muss mindestens ein gültiger vorhandener Achsbezeichner definiert und vorbelegt sein. Es können aber auch alle vorhandenen gültigen Achsbezeichner vorbelegt werden. Bitte beachten Sie die Angaben des Maschinenherstellers.

Hinweis

Neu angelegte Variable bekommen nun automatisch bei der Definition dem im Maschinendatum hinterlegten Wert für Default-Achsnamen zugewiesen.

Weitere Informationen zu einer über Maschinendatum gültigen Definition siehe:
Literatur:
/FBSY/ Funktionshandbuch Synchronaktionen

Syntax

```
PROC UP(AXIS PAR1=NO_AXIS, AXIS PAR2=NO_AXIS)
IF PAR1 <>NO_AXIS...
```

Bedeutung

`PROC`	Unterprogrammdefinition
`UP`	Unterprogrammname zur Erkennung
`PARn`	Parameter n
`NO_AXIS`	Initialisierung des Formalparameters mit Default-Achsbezeichner

Beispiel: Definition einer Achsvariablen im Hauptprogramm

```
Programmcode
DEF AXIS AXVAR
UP( , AXVAR)
```

10.3.5 Synchronaktions-Marker ($AC_MARKER[n])

Funktion

Die Feld-Variable $AC_MARKER[n] kann in Synchronaktionen gelesen, geschrieben werden. Diese Variablen können entweder im Speicher des aktiven oder passiven Filesystems liegen.

Synchronaktions-Variable: Datentyp INT

$AC_MARKER[n]	Kanalspezifischer Merker/Zähler vom Datentyp INTEGER
$MC_MM_NUM_AC_MARKER	Maschinendatum zum Einstellen der Anzahl kanalspezifischen Merker für Bewegungssynchronaktionen
n	Feldindex der Variablen 0-n

Beispiel für Merkervariable lesen und schreiben

```
Programmcode
WHEN ... DO $AC_MARKER[0] = 2
WHEN ... DO $AC_MARKER[0] = 3
WHENEVER $AC_MARKER[0] == 3 DO $AC_OVR=50
```

10.3.6 Synchronaktions-Parameter ($AC_PARAM[n])

Funktion

Synchronaktions-Parameter $AC_PARAM[n] dienen für Berechnungen und als Zwischenspeicher in Synchronaktionen. Diese Variablen können entweder im Speicher vom aktiven oder passiven Filesystem liegen.

Synchronisations-Variable: Datentyp REAL

Die Parameter sind unter gleichem Namen einmal pro Kanal vorhanden.

`$AC_PARAM[n]`	Rechenvariable für Bewegungssynchronaktionen (REAL)
`$MC_MM_NUM_AC_PARAM`	Maschinendatum zum Einstellen der Anzahl der Parameter für Bewegungssynchronaktionen bis maximal 20000.
`n`	Feldindex des Parameters 0n

Beispiel für Synchronaktions-Parameter $AC_PARAM[n]

Programmcode

```
$AC_PARAM[0]=1.5
$AC_MARKER[0]=1
ID=1 WHEN $AA_IW[X]>100 DO $AC_PARAM[1]=$AA_IW[X]
ID=2 WHEN $AA_IW[X]>100 DO $AC_MARKER[1]=$AC_MARKER[2]
```

10.3.7 Rechenparameter ($R[n])

Funktion

Diese statische Feld-Variable dient zu Berechnungen im Teileprogramm und Synchronaktionen.

Syntax

Programmierung im Teileprogramm:

```
REAL R[n]
REAL Rn
```

Programmierung in Synchronaktionen:

```
REAL $R[n]
REAL $Rn
```

Rechenparameter

Die Verwendung von Rechenparametern ermöglicht:

- Abspeicherung von Werten, die über Programmende, NC-Reset und Power On hinweg erhalten werden sollen.
- Anzeige abgespeicherter Werte im R-Parameter-Bild.

Beispiele

Programmcode	Kommentar
`WHEN $AA_IM[X]>=40.5 DO $R10=$AA_MM[Y]`	`; Verwendung von R10 in Synchronaktion.`
`G01 X500 Y70 F1000`	
`STOPRE`	`; Vorlaufstopp`
`IF R10>20`	`; Auswertung der Rechenvariable.`

Programmcode

```
SYG_AS[2]=X
SYG_IS[1]=1
WHEN $AA_IM[SGY_AS[2]]>10 DO $R3=$AA_EG_DENOM[SYG_AS[1]],SYG_AS[2]]
WHEN $AA_IM[SGY_AS[2]]>12 DO $AA_SCTRACE[SYG_AS[2]]=1
SYG_AS[1]=X
SYG_IS[0]=1
WHEN $AA_IM[SGY_AS[1]]>10 DO $R3=$$MA_POSCTRL_GAIN[SYG_IS[0]],SYG_AS[1]]
WHEN $AA_IM[SGY_AS[1]]>10 DO $R3=$$MA_POSCTRL_GAIN[SYG_AS[1]]
WHEN $AA_IM[SGY_AS[1]]>15 DO $$MA_POSCTRL_GAIN[SYG_AS[0]], SYG_AS[1]]=$R3
```

10.3.8 NC-Maschinen- und NC-Settingdaten lesen und schreiben

Funktion

Das Lesen und Schreiben von NC-Maschinen-/Settingdaten ist auch aus Synchronaktionen möglich. Beim Lesen und Schreiben von Maschinendaten-Feldelementen kann bei der Programmierung ein Index weggelassen werden. Geschieht dies im Teileprogramm, so wird beim Lesen das **erste** Feldelement gelesen und beim Schreiben werden alle Elemente des Feldes mit dem Wert beschrieben.

In Synchronaktionen wird in diesem Fall nur das **erste** Element gelesen oder geschrieben.

Festlegung

MD, SD mit

`$`: Lesen des Wertes zum Interpretationszeitpunkt der Synchronaktionen

`$$`: Lesen des Wertes im Hauptlauf

MD- und SD Werte zum Vorlaufzeitpunkt lesen

Sie werden aus der Synchronaktion mit den $-Zeichen adressiert und zum Vorlaufzeitpunkt ausgewertet.

```
ID=2 WHENEVER $AA_IM[z]<$SA_OSCILL_REVERSE_POS2[Z]-6 DO $AA_OVR[X]=0
;Hier wird der als unveränderlich angenommene Umkehrbereich 2 für Pendeln
angesprochen
```

MD- und SD Werte zum Hauptlaufzeitpunkt lesen

Sie werden aus der Synchronaktion mit den $$-Zeichen adressiert und zum Hauptlaufzeitpunkt ausgewertet.

```
ID=1 WHENEVER $AA_IM[z]<$$SA_OSCILL_REVERSE_POS2[Z]-6 DO $AA_OVR[X]=0
;Hier wird davon ausgegangen, dass die Umkehrposition durch Bedienung während der
Bearbeitung verändert werden könnte.
```

MD- und SD zum Hauptlaufzeitpunkt schreiben

Das aktuell eingestellte Zugriffsrecht muss den Schreibzugriff zulassen. Die Wirksamkeit für alle MD und SD ist in der **Literatur:** /LIS/, Listen (Buch 1) angegeben.

Zu schreibende `MD` und `SD` sind eingeleitet mit `$$` zu adressieren.

Beispiel

Programmcode	**Kommentar**
`ID=1 WHEN $AA_IW[X]>10 DO`	`;` Veränderung der Schaltposition
`$$SN_SW_CAM_PLUS_POS_TAB_1[0]=20`	von SW-Nocken. Hinweis: Die Schaltpositionen müssen 2-3 IPO-Takte vor Erreichen der Position verändert werden.
`$$SN_SW_CAM_MINUS_POS_TAB_1[0]=30`	

10.3.9 Timer-Variablen ($AC_Timer[n])

Funktion

Die Systemvariable $AC_TIMER[n] ermöglicht das Starten von Aktionen nach definierten Wartezeiten.

Timer-Variable: Datentyp REAL

`$AC_TIMER[n]`	Kanalspezifischer Timer vom Datentyp REAL
`s`	Einheit in Sekunden
`n`	Index der Timer-Variable

Timer setzen

Das Hochzählen einer Timer-Variable wird gestartet durch Wertzuweisung:
`$AC_TIMER[n] = value`

n:	Nummer der Zeitvariablen
value:	Startwert (i. d. R "0")

Timer anhalten

Das Hochzählen einer Timer-Variable wird gestoppt durch Zuweisung eines negativen Werts:
`$AC_TIMER[n]=-1`

Timer lesen

Der aktuelle Zeitwert kann bei laufender oder gestoppter Timer-Variablen gelesen werden. Nach dem Stoppen der Timer-Variablen durch Zuweisung von -1 bleibt der zuletzt aktuelle Zeitwert stehen und kann weiterhin gelesen werden.

Beispiel

Ausgabe eines Ist-Werts über Analogausgang 500 ms nach Erkennen eines digitalen Eingangs:

Programmcode	**Kommentar**
`WHEN $A_IN[1]==1 DO $AC_TIMER[1]=0`	`; Timer rücksetzen und starten`
`WHEN $AC_TIMER[1]>=0.5 DO $A_OUTA[3]=$AA_IM[X] $AC_TIMER[1]=-1`	

10.3.10 FIFO-Variablen ($AC_FIFO1[n] ... $AC_FIFO10[n])

Funktion

Zur Abspeicherung zusammengehöriger Datenfolgen stehen 10 FIFO-Variable (Umlaufspeicher) zur Verfügung.
Datentyp: REAL

Anwendung:

- zyklisches Messen
- Durchlaufbearbeitung

Auf jedes Element kann lesend und schreibend zugegriffen werden.

FIFO-Variable

Die Anzahl der verfügbaren FIFO-Variablen wird per Maschinendatum MD28260 $MC_NUM_AC_FIFO festgelegt.

Die Anzahl der in eine FIFO-Variable einschreibbaren Werte wird durch das Maschinendatum MD28264 $MC_LEN_AC_FIFOdefiniert. Alle FIFO-Variablen haben gleiche Länge.

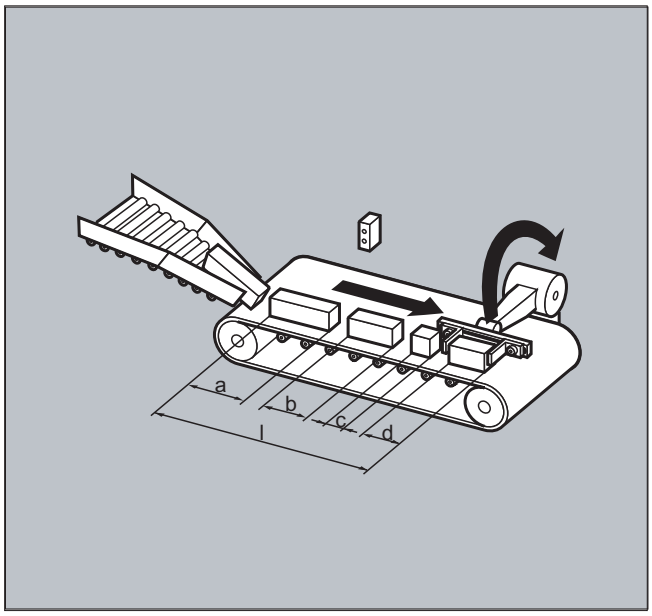
Die Summe aller FIFO-Elemente wird nur gebildet, wenn in MD28266 $MC_MODE_AC_FIFO Bit0 gesetzt ist.

Die Indizes **0 bis 5** haben Sonderbedeutung:

Index	Bedeutung	
0	Beim Schreiben:	Neuer Wert wird in den FIFO abgelegt.
	Beim Lesen:	Ältestes Element wird gelesen und aus FIFO entfernt.
1	Zugriff auf das älteste gespeicherte Element	
2	Zugriff auf das jüngste gespeicherte Element	
3	Summe aller FIFO-Elemente	
4	Anzahl der im FIFO verfügbaren Elemente Auf jedes Element des FIFO kann lesend und schreibend zugegriffen werden. Das Rücksetzen der FIFO-Variablen erfolgt durch Rücksetzen der Element-Anzahl, z. B. für die erste FIFO-Variable: $AC_FIFO1[4] = 0	
5	Aktueller Schreibindex relativ zum FIFO-Anfang	
6 bis n_{max}	Zugriff auf n-tes FIFO-Element	

Beispiel: Umlaufspeicher

Während eines Produktionsablaufs wird ein Förderband zum Transport von Produkten mit unterschiedlichen Längen (a, b, c, d) benutzt. Auf dem Förderband mit der Transportlänge werden daher abhängig von den jeweiligen Produktlängen unterschiedliche Anzahlen von Produkten gleichzeitig befördert. Bei gleich bleibender Fördergeschwindigkeit muss damit eine Anpassung der Produktentnahme vom Band an die variablen Ankunftszeiten der Produkte erfolgen.



Programmcode	Kommentar
`DEF REAL ZWI=2.5`	`; Konstanter Abstand zwischen aufgelegten Produkten.`
`DEF REAL GESAMT=270`	`; Abstand zwischen Längenmess- und Entnahmeposition.`
`EVERY $A_IN[1]==1 DO $AC_FIFO1[4]=0`	`; Bei Prozessbeginn, FIFO zurücksetzen.`
`EVERY $A_IN[2]==1 DO $AC_TIMER[0]=0`	`; Unterbricht ein Produkt die Lichtschranke, Zeitmessung starten.`
`EVERY $A_IN[2]==0 DO $AC_FIFO1[0]=$AC_TIMER[0]*$AA_VACTM[B]`	`; Wird Lichtschranke frei, aus gemessener Zeit und Transportgeschwindigkeit die Produktlänge berechnen und in FIFO speichern.`
`EVERY $AC_FIFO1[3]+$AC_FIFO1[4]*ZWI>=GESAMT DO POS[Y]=-30 $R1=$AC_FIFO1[0]`	`; Sobald Summe aller Produktlängen und Zwischenabstände größer/gleich der Länge zwischen Auflege- und Entnahmeposition ist, Produkt an der Entnahmeposition vom Transportband nehmen, zugehörige Produktlänge aus FIFO auslesen.`

10.3.11 Auskunft über Satztypen im Interpolator ($AC_BLOCKTYPE, $AC_BLOCKTYPEINFO, $AC_SPLITBLOCK)

Funktion

Für Synchronaktionen stehen die folgenden Systemvariablen zur Verfügung, um Auskunft über einen im Hauptlauf gerade aktuellen Satz zu erhalten:

- $AC_BLOCKTYPE
- $AC_BLOCKTYPEINFO
- $AC_SPLITBLOCK

Blocktype- und Blocktypeinfo-Variablen

$AC_BLOCKTYPE		**$AC_BLOCKTYPEINFO**				
Wert:		**Wert:**				
0	**ungleich 0**	**T**	**H**	**Z**	**E**	**Bedeutung:**
Origi-nalsatz	**Zwischensatz**					**Auslöser für Zwischensatz:**
	1	1	0	0	0	Intern generierter Satz, keine weitere Auskunft
	2	2	0	0	1	Fasen/Runden: Gerade
	2	2	0	0	2	Fasen/Runden: Kreis
	3	3	0	0	1	WAB: Anfahren mit Gerade
	3	3	0	0	2	WAB: Anfahren mit Viertelkreis
	3	3	0	0	3	WAB: Anfahren mit Halbkreis
						Werkzeugkorrektur:
	4	4	0	0	1	Anfahrsatz nach STOPRE
	4	4	0	0	2	Verbindungssätze bei nicht gefundenem Schnittpunkt
	4	4	0	0	3	Punktförmiger Kreis an Innenecken (nur bei TRACYL)
	4	4	0	0	4	Umfahrungskreis (bzw. Kegelschnitt) an Außenecken
	4	4	0	0	5	Anfahrsätze bei Korrekturunterdrückung
	4	4	0	0	6	Anfahrsätze bei erneuter WRK-Aktivierung
	4	4	0	0	7	Satzaufspaltung wegen zu hoher Krümmung
	4	4	0	0	8	Ausgleichssätze beim 3D-Stirnfräsen (WerkzeugVektor || Flächenvektor)
						Überschleifen durch:
	5	5	0	0	1	G641
	5	5	0	0	2	G642

$AC_BLOCKTYPE		$AC_BLOCKTYPEINFO				
Wert:		Wert:				
0	ungleich 0	T	H	Z	E	Bedeutung:
Origi-nalsatz	Zwischensatz					Auslöser für Zwischensatz:
	5	5	0	0	3	G643
	5	5	0	0	4	G644
						TLIFT-Satz mit:
	6	6	0	0	1	linearer Bewegung der Tangentialachse und ohne Abhebebewegung
	6	6	0	0	2	Nichtlinearer Bewegung der Tangentialachse (Polynom) und ohne Abhebebewegung
	6	6	0	0	3	Abhebebewegung, Tangentialachsbewegung und Abhebebewegung starten gleichzeitig
	6	6	0	0	4	Abhebebewegung, Tangentialachse startet erst, wenn bestimmte Abhebeposition erreicht wird.
						Wegaufteilung:
	7	7	0	0	1	programmierte Wegaufteilung ohne das Stanzen oder Nibbeln aktiv ist
	7	7	0	0	2	programmierte Wegaufteilung mit aktivem Stanzen oder Nibbeln
	7	7	0	0	3	automatisch intern generierte Wegaufteilung
						Compile Zyklen:
	8	ID-Applikation				ID der Compile-Zyklen-Applikation, die den Satz erzeugt hat

T: Tausenderstelle
H: Hunderterstelle
Z: Zehnerstelle
E: Einerstelle

Hinweis

$AC_BLOCKTYPEINFO enthält in der Tausenderstelle (T) immer auch den Wert für Blocktype für den Fall, dass ein Zwischensatz vorliegt. In $AC_BLOCKTYPE ungleich 0 wird die Tausenderstelle nicht übernommen.

$AC_SPLITBLOCK	
Wert:	**Bedeutung:**
0	Unveränderter programmierter Satz, (ein durch den Kompressor generierter Satz wird auch als programmierter Satz behandelt)
1	Es liegt ein intern generierter Satz oder ein verkürzter Originalsatz vor
3	Es liegt der letzte Satz in einer Kette von intern generierten Sätzen oder verkürzten Originalsätzen vor

Beispiel: Zählen von Überschleifsätzen

```
Programmcode                Kommentar

$AC_MARKER[0]=0
$AC_MARKER[1]=0
$AC_MARKER[2]=0
...
                            ; Definition von Synchronaktionen, mit denen
                              Überschleifsätze gezählt werden.
                            ; Alle Überschleifsätze zählen in $AC_MARKER[0]:
ID=1 WHENEVER ($AC_TIMEC==0) AND ($AC_BLOCKTYPE==5) DO $AC_MARKER[0]=$AC_MARKER[0]+1
...
                            ; Mit G641 erzeugte Überschleifsätze zählen in $AC_MARKER[1]:
ID=2 WHENEVER ($AC_TIMEC==0) AND ($AC_BLOCKTYPEINFO==5001) DO $AC_MARKER[1]=$AC_MARKER[1]+1
                            ; Mit G642 erzeugte Überschleifsätze zählen in $AC_MARKER[2]:
ID=3 WHENEVER ($AC_TIMEC==0) AND ($AC_BLOCKTYPEINFO==5002) DO $AC_MARKER[2]=$AC_MARKER[2]+1
...
```

10.4 Aktionen in Synchronaktionen

10.4.1 Übersicht möglicher Aktionen in Synchronaktionen

Aktionen in Synchronaktionen bestehen aus Wertzuweisungen, Funktions- oder Parameteraufrufen, Schlüsselwörter oder Technologiezyklen. Über Operatoren sind komplexe Ausführungen möglich.

Mögliche Anwendungen sind:

- Berechnungen komplexer Ausdrücke im IPO-Takt
- Achsbewegungen und Spindelsteuerungen
- Settingdaten aus Synchronaktionen online verändern und auswerten (z. B. Positionen und Zeiten von Softwarenocken an PLC oder NC-Peripherie ausgeben)
- Hilfsfunktionsausgabe an PLC
- Zusätzliche Sicherheitsfunktionen einrichten
- Überlagerte Bewegung, Online-Werkzeugkorrektur und Abstandsregelung einstellen
- Aktionen in allen Betriebsarten ausführen
- Synchronaktionen von PLC aus beeinflussen
- Technologiezyklen ausführen
- Ausgabe von digitalen und analogen Signalen
- Performanceerfassung von Synchronaktionen am Interpolationstakt und die Rechenzeit des Lagereglers für eine Auslastungsbewertung erfassen
- Diagnose-Möglichkeiten in der Bedienoberfläche

Synchronaktion	Beschreibung
DO $V…= DO $A...=	zuweisen (Servo-Werte) Variable zuweisen (Hauptlaufvariable)
DO $AC…[n]= DO $AC_MARKER[n]= DO $AC_PARAM[n]=	Spezielle Hauptlaufvariable Synchronaktions-Marker lesen oder schreiben Synchronaktions-Parameter lesen oder schreiben
DO $R[n]=	Rechenvariable lesen oder schreiben
DO $MD...= DO $$SD...=	Lesen des MD-Wertes zum Interpolationszeitpunkt Schreiben des SD-Wertes im Hauptlauf
DO $AC_TIMER[n]=Startwert	Timer
DO $AC_FIFO1[n] …FIFO10[n]=	FIFO-Variable
DO $AC_BLOCKTYPE= DO $AC_BLOCKTYPEINFO= DO $AC_SPLITBLOCK=	Aktuellen Satz interpretieren (Hauptlaufvariable)
DO M-, S und H z. B. M07	Ausgabe von M-, S- und H-Hilfsfunktionen
DO RDISABLE	Einlesesperre setzen
DO STOPREOF	Vorlaufstopp aufheben
DO DELDTG	Schnelles Restweglöschen ohne Vorlaufstopp
FTCDEF(Polyn., LL, UL , Koeffiz.) DO SYNFCT(Polyn., Output, Input)	Definition von Polynomen Aktivierung von Synchronfunktionen: AC-Regelung
DO FTOC	Online-Werkzeugkorrektur
DO G70/G71/G700/G710	Maßsystem für Positionieraufgaben festlegen (Maßangabe in Inch oder metrisch)
DO POS[Achse]= / DO MOV[Achse]= DO SPOS[Spindel]=	Kommandoachsen starten/positionieren/stoppen Spindeln starten/positionieren/stoppen
DO MOV[Achse]=Wert	Endlosbewegungen einer Kommandoachse starten/stoppen
DO POS[Achse]= FA [Achse]=	Axialen Vorschub FA
ID=1 ... DO POS[Achse]= FA [Achse]= ID=2 ... DO POS[Achse]= $AA_IM[Achse] FA [Achse]=	Positionieren aus Synchronaktionen
DO PRESETON(Achse, Wert)	Istwertsetzen (Preset aus Synchronaktionen)
ID=1 EVERY $A_IN[1]=1 DO M3 S… ID=2 EVERY $A_IN[2]=1 DO SPOS=	Spindeln starten/positionieren/stoppen
DO TRAILON(FA,LA,Koppelfaktor) DO LEADON(FA,LA,NRCTAB,OVW)	Mitschleppen einschalten Leitwertkopplung einschalten
DO MEAWA(Achse)= DO MEAC(Achse)=	Axiales Messen einschalten Kontinuierliches Messen einschalten
DO [Feld n, m]=SET(Wert, Wert, ...) DO [Feld n, m]=REP(Wert, Wert, ...)	Initialisierung von Feldvariablen mit Wertelisten Initialisierung von Feldvariablen mit gleichen Werten
DO SETM(Marker Nr.) DO CLEARM(Marker Nr.)	Wartemarken setzen Wartemarken löschen
DO SETAL(Alarm Nr.)	Zyklenalarm setzen (zusätzliche Sicherheitsfunktion)

Synchronaktion	Beschreibung
DO FXS[Achse]= DO FXST[Achse]= DO FXSW[Achse]= DO FOCON[Achse]= DO FOCOF[Achse]=	Fahren auf Festanschlag anwählen Klemmmoment verändern Überwachungsfenster verändern Fahren mit begrenztem Moment/Kraft aktivieren (modal) FOC Fahren mit begrenztem Moment/Kraft deaktivieren (Synchronaktion wirkt satzbezogen)
ID=2 EVERY $AC_BLOCKTYPE==0 DO $R1=$AC_TANEB	Winkel zwischen Bahntangente im Endpunkt des aktuellen Satzes und der Bahntangente im Startpunkt des programmierten Folgesatzes
DO $AA_OVR= DO $AC_OVR= DO $AA_PLC_OVR DO $AC_PLC_OVR DO $AA_TOTAL_OVR DO $AC_TOTAL_OVR	Axial Override Bahnoverride der von der PLC vorgegebene achsiale Override der von der PLC vorgegebene Bahnoverride resultierender axial Override resultierender Bahnoverride
$AN_IPO_ACT_LOAD= $AN_IPO_MAX_LOAD= $AN_IPO_MIN_LOAD= $AN_IPO_LOAD_PERCENT= $AN_SYNC_ACT_LOAD= $AN_SYNC_MAX_LOAD= $AN_SYNC_TO_IPO=	aktuelle IPO-Rechenzeit längste IPO-Rechenzeit kürzeste IPO-Rechenzeit aktuelle IPO-Rechenzeit im Verhältnis zum IPO-Takt aktuelle Rechenzeit für Synchronaktion über alle Kanäle längste Rechenzeit für Synchronaktion über alle Kanäle prozentualer Anteil der ges. Synchronaktion
DO TECCYCLE	Technologiezyklus ausführen
DO LOCK(n, n, ...) DO UNLOCK(n, n, ...) DO RESET(n, n, ...)	Sperren Freischalten RESET eines Technologiezyklus
CANCEL(n, n, ...)	Modale Synchronaktionen mit der Bezeichnung ID(S) im Teileprogramm löschen

10.4.2 Ausgabe von Hilfsfunktionen

Funktion

Ausgabezeitpunkt

Die Ausgabe von Hilfsfunktionen erfolgt in der Synchronaktion unmittelbar zum Ausgabezeitpunkt der Aktion. Der über Maschinendatum definierte Ausgabezeitpunkt für Hilfsfunktionen ist unwirksam.

Der Ausgabezeitpunkt ist dann gegeben, wenn die Bedingung erfüllt ist.

Beispiel:

Kühlmittel einschalten bei bestimmter Achsposition:

```
WHEN $AA_IM[X]>=15 DO M07 POS[X]=20 FA[X]=250
```

Erlaubte Schlüsselworte in satzweise wirksamen Synchronaktionen (ohne Modal-ID)

Hilfsfunktionen in satzweise wirksamen Synchronaktionen (ohne Modal-ID) können nur mit den Schlüsselwörtern `WHEN` oder `EVERY` programmiert werden.

Hinweis

Folgende Hilfsfunktionen sind in Synchronaktion heraus nicht erlaubt:

- `M0`, `M1`, `M2`, `M17`, `M30`: Programm-Halt/-Ende (`M2`, `M17`, `M30` möglich bei Technologiezyklus)
- `M6` bzw. über Maschinendatum eingestellte M-Funktionen für den Werkzeugwechsel

---

Beispiel

Programmcode	Kommentar
`WHEN $AA_IW[Q1]>5 DO M172 H510`	`; Wenn Istwert der Q1-Achse 5 mm übersteigt, Hilfsfunktionen M172 und H510 an PLC ausgeben.`

10.4.3 Einlesesperre setzen (RDISABLE)

Funktion

Mit RDISABLE wird bei erfüllter Bedingung die weitere Satzbearbeitung im Hauptprogramm angehalten. Programmierte Bewegungssynchronaktionen werden weiterbearbeitet, nachfolgende Sätze weiter aufbereitet.

Im Bahnsteuerbetrieb wird am Anfang eines Satzes mit RDISABLE in Synchronaktionen immer Genauhalt ausgelöst, unabhängig davon, ob RDISABLE wirksam wird oder nicht.

Beispiel

Abhängig von externen Eingängen Programm im Interpolationstakt starten.

```
Programmcode                                    Kommentar
...
WHENEVER $A_INA[2]<7000 DO RDISABLE             ; Falls am Eingang 2 die Spannung
                                                  von 7V
                                                  unterschritten wird,
                                                  Programmfortsetzung ;anhalten
                                                  (1000= 1V).
N10 G1 X10                                      ; wenn Bedingung erfüllt ist, wirkt
                                                  Einlesesperre am Ende von N10
N20 G1 X10 Y20
...
```

10.4.4 Vorlaufstopp aufheben (STOPREOF)

Funktion

Bei explizit programmiertem Vorlaufstopp STOPRE oder durch eine aktive Synchronaktion implizit aktiviertem Vorlaufstopp hebt STOPREOF, sobald die Bedingung erfüllt ist, den Vorlaufstopp nach dem nächsten Bearbeitungssatz auf.

Hinweis

STOPREOF muss mit dem Schlüsselwort `WHEN` und satzweise (ohne ID-Nummer) programmiert werden.

Beispiel

Schnelle Programmverzweigung am Satzende.

Programmcode	Kommentar
`WHEN $AC_DTEB<5 DO STOPREOF`	`; Wenn die Entfernung zum Satzende 5 mm ;unterschreitet, Vorlaufstopp aufheben.`
`G01 X100`	`; Nach Ausführung der Linearinterpolation, wird Vorlaufstopp aufgehoben.`
`IF $A_INA[7]>500 GOTOF MARKE1=X100`	`; Wenn am Eingang 7 die Spannung von 5V ;überschritten wird, zu Label 1 springen.`

10.4.5 Restweglöschen (DELDTG)

Funktion

In Abhängigkeit von einer Bedingung kann Restweglöschen für die Bahn und für die angegebenen Achsen ausgelöst werden.

Zur Verfügung steht:

- Schnelles, vorbereitetes Restweglöschen
- Restweglöschen ohne Vorbereitung

Vorbereitetes Restweglöschen mit DELDTG erlaubt eine sehr schnelle Reaktion auf das Auslöseereignis und wird daher bei zeitkritischen Anwendungen verwendet, z. B. wenn

- die Zeit zwischen Restweglöschen und Start des Folgesatzes sehr kurz sein soll.
- die Bedingung für das Restweglöschen mit sehr hoher Wahrscheinlichkeit erfüllt wird.

Hinweis

Die dem DELDTG in Klammer nachgestellte Achsbezeichnung ist nur für **eine** Positionierachse gültig.

Syntax

Restweglöschen für die Bahn
```
DO DELDTG
```
axiales Restweglöschen
```
DO DELDTG(Achse1) DELDTG(Achse2) ...
```

Beispiel Schnelles Restweglöschen Bahn

Programmcode	**Kommentar**
`WHEN $A_IN[1]==1 DO DELDTG`	
`N100 G01 X100 Y100 F1000`	`; wenn Eingang gesetzt ist, wird die Bewegung abgebrochen`
`N110 G01 X…`	
`IF $AA_DELT>50…`	

Beispiel Schnelles axiales Restweglöschen

```
Programmcode                                        Kommentar
Abbruch einer Positionierbewegung:
ID=1 WHEN $A_IN[1]==1 DO MOV[V]=3 FA[V]=700     ;  Achse starten
WHEN $A_IN[2]==1 DO DELDTG(V)                   ;  Restweglöschen, Achse anhalten
                                                   erfolgt mit MOV=0
Abhängig von Eingangsspannung den Restweg
löschen:
WHEN $A_INA[5]>8000 DO DELDTG(X1)               ;  Sobald am Eingang 5 Spannung von
                                                   8V überschritten wird, Restweg
                                                   von Achse X1
                                                   löschen. Bahnbewegung läuft
                                                   weiter.
POS[X1]=100 FA[X1]=10 G1 Z100 F1000
```

Weitere Informationen

Am Ende des Bewegungssatzes, in dem vorbereitetes Restweglöschen ausgelöst wurde, wird implizit Vorlaufstopp aktiviert.

Bahnsteuerbetrieb bzw. Positionierachsbewegungen werden damit am Ende des Satzes mit schnellem Restweglöschen unterbrochen bzw. gestoppt.

Hinweis

Vorbereitetes Restweglöschen:

- kann bei aktiver Werkzeugradiuskorrektur nicht eingesetzt werden.
- darf nur in satzweise wirksamen Synchronaktionen (ohne ID-Nummer) Aktion programmiert werden.

10.4.6 Polynomdefinition (FCTDEF)

Funktion

Mit FCTDEF können Polynome 3. Grades in der Form $y=a_0+a_1x+a_2x^2+a_3x^3$ definiert werden. Diese Polynome werden von der Online-Werkzeugkorrektur FTOC und der Auswertefunktion SYNFCT benutzt.

Syntax

```
FCTDEF(Polynom-Nr.,LLIMIT,ULIMIT,a0,a1,a2,a3)
```

Bedeutung

`Polynom-Nr.`	Nummer des Polynoms 3. Ordnung
`LLIMIT`	untere Grenze für Funktionswert
`ULIMIT`	obere Grenze für Funktionswert
`a0, a1, a2, a3`	Polynomkoeffizienten

Diese Werte sind auch über Systemvariable zugreifbar

`$AC_FCTLL[n]`	Untergrenze für Funktionswert
`$AC_FCTUL[n]`	Obergrenze für Funktionswert
`$AC_FCT0[n]`	a_0
`$AC_FCT1[n]`	a_1
`$AC_FCT2[n]`	a_2
`$AC_FCT3[n]`	a_3

Hinweis

Systemvariablen schreiben

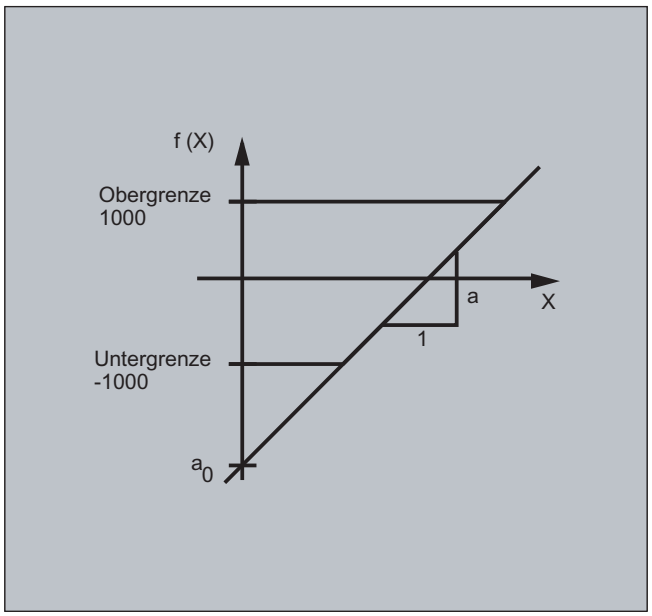
- Die Systemvariablen können vom Teileprogramm oder aus einer Synchronaktion heraus geschrieben werden. Beim Schreiben vom Teileprogramm aus muss durch Programmierung von STOPRE für satzsynchrones Schreiben gesorgt werden.
- Die Systemvariablen `$AC_FCTLL[n]`, `$AC_FCTUL[n]`, `$AC_FCT0[n]` bis `$AC_FCTn[n]` sind aus Synchronaktionen änderbar

Beim Schreiben aus Synchronaktionen sind die Polynomkoeffizienten und Funktionswertgrenzen sofort wirksam.

Beispiel Polynom für Geradenabschnitt

Mit Obergrenze 1000, Untergrenze -1000, dem Ordinatenabschnitt `a0=$AA_IM[X]` und der Geradensteigung 1 lautet die Polynomdefinition:

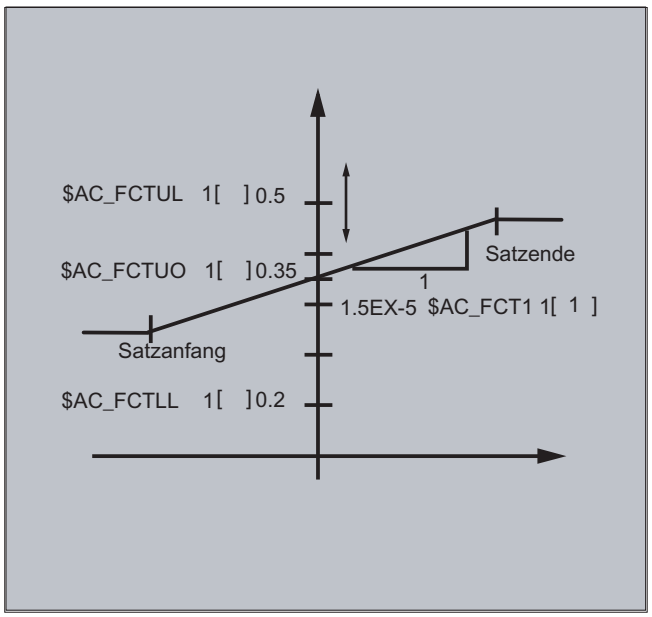
```
FCTDEF(1, -1000,1000,$AA_IM[X],1)
```



Beispiel Laserleistungssteuerung

Eine der möglichen Anwendungen von Polynomdefinition ist die Laserleistungssteuerung.

Laserleistungssteuerung heißt:
Beeinflussung eines Analogausgangs in Abhängigkeit z. B. der Bahngeschwindigkeit.



Programmcode	Kommentar
`$AC_FCTLL[1]=0.2`	`;` Definition der Polynomkoeffizienten
`$AC_FCTUL[1]=0.5`	
`$AC_FCT0[1]=0.35`	
`$AC_FCT1[1]=1.5EX-5`	
`STOPRE`	
`ID=1 DO $AC_FCTUL[1]=$A_INA[2]*0.1 +0.35`	`;` Obergrenze online verändern.
`ID=2 DO SYNFCT(1,$A_OUTA[1],$AC_VACTW)`	`;` in Abhängigkeit von der Bahngeschwindigkeit (in $AC_VACTW hinterlegt) wird die Laserleistungssteuerung über den Analog-Ausgang 1 gesteuert

Hinweis

Die Benutzung des oben definierten Polynoms erfolgt mit SYNFCT.

10.4.7 Synchronfunktion (SYNFCT)

Funktion

SYNFCT berechnet den Ausgangswert eines Polynoms 3. Grades gewichtet mit der Eingangs-Variablen. Das Ergebnis steht in der Ausgangs-Variablen und wird nach oben und unten begrenzt.

Die Auswertefunktion findet Anwendung

- bei AC-Regelung (Adaptive Control),
- bei Laserleistungssteuerung,
- bei Positionsaufschaltung.

Syntax

```
SYNFCT(Polynom-Nr., Hauptlaufvariable-Ausgang, Hauptlaufvariable-Eingang)
```

Bedeutung

Als Ausgangs-Variable können Variable gewählt werden, die

- mit additiver Beeinflussung
- mit multiplikativer Beeinflussung
- als Positionsoffset
- direkt

in den Bearbeitungsvorgang eingehen.

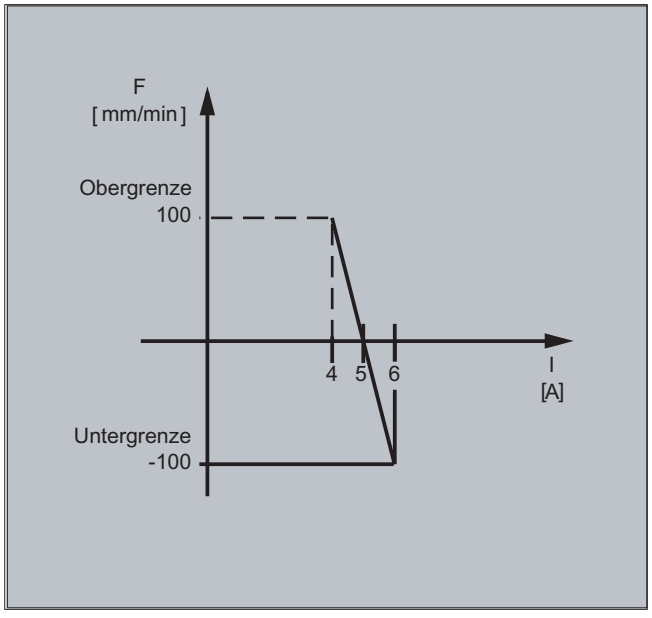
`DO SYNFCT`	Aktivierung der Auswertefunktion
`Polynom-Nr.`	Mit FCTDEF definiertes Polynom (siehe Unterkapitel "Polynomdefinition")
`Hauptlaufvariable-Ausgang`	Hauptlaufvariable schreiben
`Hauptlaufvariable-Eingang`	Hauptlaufvariable lesen

Beispiel AC-Regelung (additiv)

Additive Beeinflussung des programmierten Vorschubs

Ein programmierter Vorschub soll additiv über den Strom der X-Achse (Zustellachse) geregelt werden:

Der Vorschub soll um +/- 100 mm/min variieren, wobei der Strom um +/-1A um den Arbeitspunkt bei 5A schwankt.



1. Polynomdefinition

Bestimmung der Koeffizienten

$y = f(x) = a_0 + a_1x + a_2x^2 + a_3x^3$

a_1 = -100mm/1 min A

a_0 = -(-100)*5 =500

$a_2 = a_3 = 0$ (kein quadratisches und kubisches Glied)

Obergrenze = 100

Untergrenze = -100

Daraus folgt:

```
FCTDEF(1,-100,100,500,-100,0,0)
```

2. AC-Regelung einschalten

```
ID=1 DO SYNFCT(1,$AC_VC,$AA_LOAD[x])
```

;Über `$AA_LOAD[x]` aktuelle Achslast (% max. Antriebsstrom) lesen,
;mit oben definiertem Polynom Bahnvorschubkorrektur berechnen.

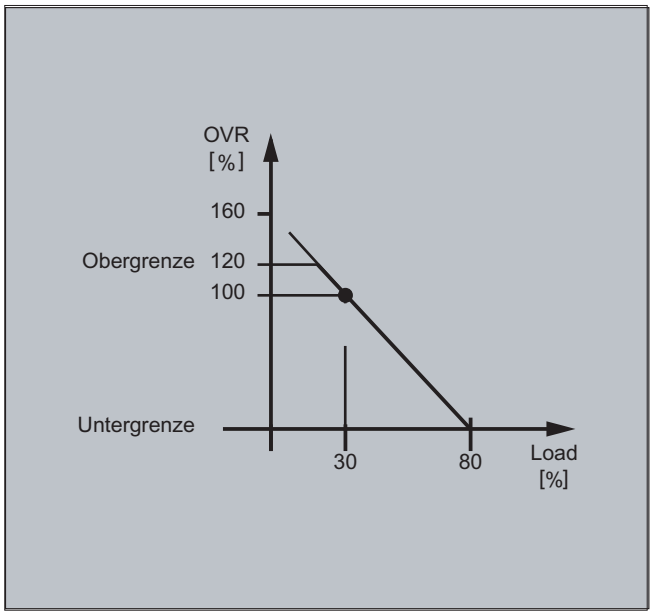
Beispiel AC-Regelung (multiplikativ)

Programmierten Vorschub multiplikativ beeinflussen

Der programmierter Vorschub soll multiplikativ beeinflusst werden, wobei der Vorschub – abhängig von der Belastung des Antriebes –bestimmte Grenzen nicht überschreiten soll:

- Bei Antriebslast von 80% soll der Vorschub stoppen: Override = 0.
- Bei Antriebslast von 30% darf mit programmiertem Vorschub verfahren werden: Override = 100%.

Vorschubgeschwindigkeit darf um maximal 20% überschritten werden:
Max. Override = 120%.



1. Polynomdefinition

Bestimmung der Koeffizienten

$y = f(x) = a_0 + a_1x + a_2x^2 + a_3x^3$

$a_1 = -100\%/(80-30)\% = -2$

$a_0 = 100 + (2*30) = 160$

$a_2 = a_3 = 0$ (kein quadratisches und kubisches Glied)

Obergrenze = 120

Untergrenze = 0

Daraus folgt:

```
FCTDEF(2,0,120,160,-2,0,0)
```

2. AC-Regelung einschalten

```
ID=1 DO SYNFCT(2,$AC_OVR,$AA_LOAD[x])
```

;Über `$AA_LOAD[x]` aktuelle Achslast (% max. Antriebsstrom) lesen,
;mit oben definiertem Polynom Vorschuboverride berechnen

10.4.8 Abstandsregelung mit begrenzter Korrektur ($AA_OFF_MODE)

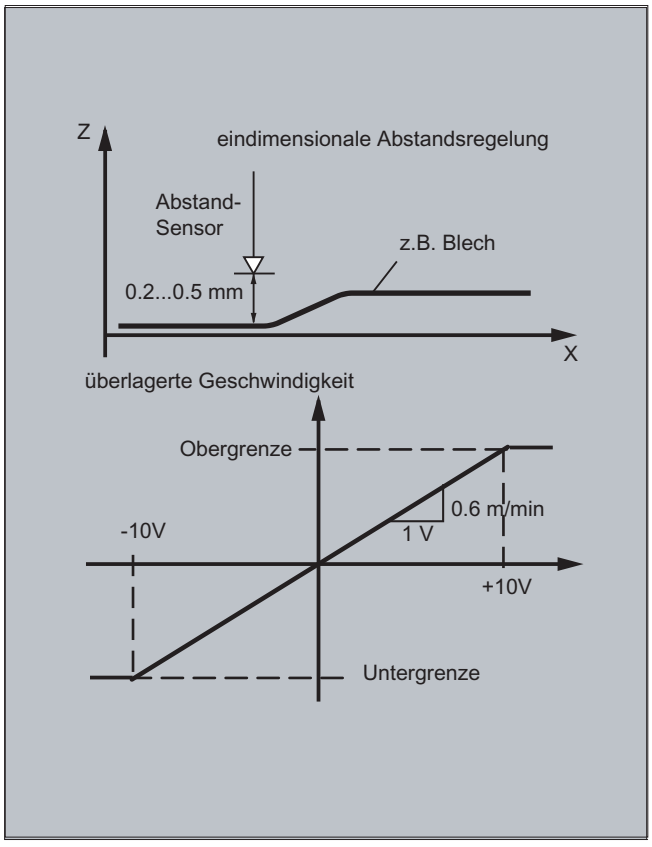
Hinweis

Diese Funktion steht für SINUMERIK 828D nicht zur Verfügung!

Funktion

Die integrierende Berechnung der Abstandswerte erfolgt mit Grenzbereichsprüfung:

$AA_OFF_MODE = 1



ACHTUNG
Die Kreisverstärkung des überlagerten Regelkreises ist abhängig von der Einstellung des IPO-Taktes. Abhilfe: MD für IPO-Takt lesen und einrechnen.

Hinweis

Begrenzung der Geschwindigkeit des überlagerten Interpolators durch MD32020 JOG_VELO bei Ipo-Takt 12 ms.

Formel für Geschwindigkeit:

$$\frac{0.120\text{mm}}{0.6\text{ms}} / \text{mV} = 0.6 \frac{\text{m}}{\text{min}} / \text{V}$$

Beispiel

Unterprogramm "AON": Abstandsregelung Ein

```
PROC AON
$AA_OFF_LIMIT[Z]=1                          ; Grenzwert festlegen.
FCTDEF(1, -10, +10, 0, 0.6, 0.12)           ; Polynomdefinition
ID=1 DO SYNFCT(1,$AA_OFF[Z],$A_INA[3])      ; Abstandsregelung aktiv.
ID=2 WHENEVER $AA_OFF_LIMIT[Z]<>0           ; Bei Überschreitung des
 DO $AA_OVR[X] = 0                            Grenzbereiches Achse X sperren.
RET
ENDPROC
```

Unterprogramm "AOFF": Abstandsregelung Aus

```
PROC AOFF
CANCEL(1)                 ; Synchronaktion Abstandsregel. löschen
CANCEL(2)                 ; Grenzbereichsprüfung löschen
RET
ENDPROC
```

Hauptprogramm "MAIN"

```
AON                       ; Abstandsregelung Ein
...
G1 X100 F1000
AOFF                      ; Abstandsregelung Aus
M30
```

Weitere Informationen

Positionsoffset im Basiskoordinatensystem

Mit der Systemvariable $AA_OFF[Achse]kann jeder Achse im Kanal eine Bewegung überlagert werden. Sie wirkt als Positionsoffset im Basiskoordinatensystem.

Der so programmierte Positionsoffset wird der entsprechenden Achse sofort überlagert, unabhängig davon, ob die Achse programmiert verfahren wird oder nicht.

Hauptlaufvariable-Ausgang begrenzen:

Es ist möglich, den absolut zu korrigierenden Wert (Hauptlaufvariable-Ausgang) auf den im Settingdatum SD43350 $SA_AA_OFF_LIMIThinterlegten Wert zu begrenzen.

Über das Maschinendatum MD36750 $MA_AA_OFF_MODE wird die Art der Überlagerung des Abstandes festgelegt:

Wert	Bedeutung
0	Proportionale Bewertung
1	Integrierende Bewertung

Mit der Systemvariable $AA_OFF_LIMIT[Achse] kann richtungsabhängig abgefragt werden, ob sich der Korrekturwert im Grenzbereich befindet. Diese Systemvariable kann aus Synchronaktionen abgefragt werden und beim Erreichen eines Grenzwerts z. B. die Achse stoppen oder Alarm setzen.

0: Korrekturwert nicht im Grenzbereich
1 Limit des Korrekturwertes in positiver Richtung erreicht
-1: Limit des Korrekturwertes in negativer Richtung erreicht

10.4.9 Online-Werkzeugkorrektur (FTOC)

Funktion

FTOC ermöglicht eine überlagerte Bewegung für eine Geometrieachse nach einem mit FCTDEF programmierten Polynom in Abhängigkeit von einem Bezugswert, der z. B. der Istwert einer Achse sein kann.

Der Koeffizient a_0 der Funktionsdefinition FCTDEF(...) wird bei FTOC ausgewertet. Ober- und Untergrenze sind abhängig von a_0.

Mit FTOC können modale Online-Werkzeugkorrekturen oder Abstandsregelungen als Synchronaktionen programmiert werden.

Die Funktion findet Anwendung bei der Bearbeitung des Werkstücks und Abrichten der Schleifscheibe im gleichen Kanal oder in unterschiedlichen Kanälen (Bearbeitungs- und Abrichtkanal).

Die Randbedingungen und Festlegungen zum Abrichten von Schleifscheiben gelten für FTOC analog zur Online-Werkzeugkorrektur mit PUTFTOCF (siehe "Online-Werkzeugkorrektur (PUTFTOCF, FCTDEF, PUTFTOC, FTOCON, FTOCOF) (Seite 404)").

Syntax

```
FCTDEF(<Funktion>,<LLimit>,<ULimit>,<a0>,<a1>,<a2>,<a3>)
FTOC(<Funktion>,<Bezugswert>,<WZ-Parameter>,<Kanal>,<Spindel>)
...
```

Bedeutung

FCTDEF: Mit FCTDEF wird die Polynom-Funktion für FTOC definiert.

Parameter:

<Funktion>: Nummer der Polynom-Funktion
Typ: INT
Wertebereich: 1 ... 3

<LLimit>: Unterer Grenzwert
Typ: REAL

<ULimit>: Oberer Grenzwert
Typ: REAL

<a0> ... <a3>: Koeffizienten der Polynom-Funktion
Typ: REAL

`DO FTOC`: Funktion "Online-WZK schreiben kontinuierlich modal" ausführen

Parameter:

`<Funktion>`: Nummer der Polynom-Funktion

Typ: INT

Wertebereich: 1 ... 3

Hinweis:
Muss mit der Angabe bei `FCTDEF` übereinstimmen.

`<Bezugswert>`: Hauptlaufvariable, zu der ein Funktionswert über die mit `FCTDEF` definierte Polynom-Funktion berechnet werden soll.

Typ: VAR REAL

`<WZ-Parameter>`: Nummer des Verschleißparameters (Länge 1, 2 oder 3), in dem der Korrekturwert addiert werden soll.

Typ: INT

`<Kanal>`: Nummer des Kanals, in dem die Online-WZK wirksam werden soll.

Typ: INT

Hinweis:
Eine Angabe ist nur erforderlich, wenn die Korrektur nicht im aktiven Kanal wirksam werden soll.

`<Spindel>`: Nummer der Spindel, für die die Online-WZK wirksam werden soll.

Typ: INT

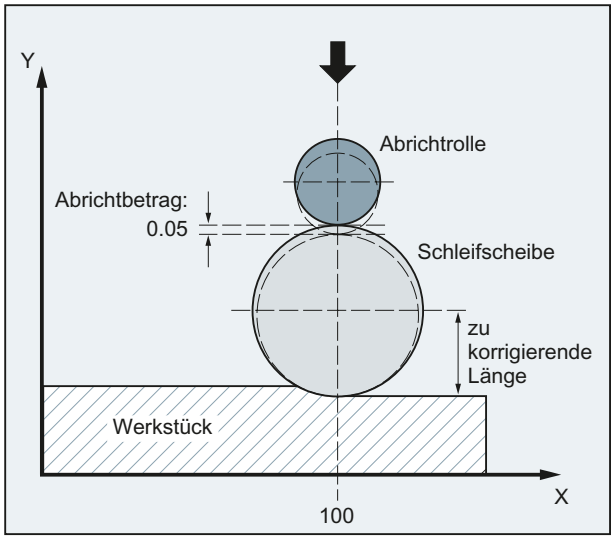
Hinweis:
Eine Angabe ist nur erforderlich, wenn statt dem aktiven, im Einsatz befindlichen Werkzeug eine nicht aktive Schleifscheibe korrigiert werden soll.

Hinweis

Im Zielkanal muss `FTOCON` eingeschaltet sein.

Beispiel

Die Länge der aktiven, im Eingriff befindlichen Schleifscheibe soll korrigiert werden.



Programmcode	**Kommentar**
`FCTDEF(1,-1000,1000,-$AA_IW[V],1)`	`; Funktion definieren.`
`ID=1 DO FTOC(1,$AA_IW[V],3,1)`	`; Online-Werkzeugkorrektur anwählen: Istwert der V-Achse ist Eingangswert für Polynom 1. Ergebnis wird im Kanal 1 als Korrekturwert zur Länge 3 der aktiven Schleifscheibe addiert.`
`WAITM(1,1,2)`	`; Synchronisation mit Bearbeitungskanal.`
`G1 V-0.05 F0.01 G91`	`; Zustellbewegung zum Abrichten.`
`G1 V-0.05 F0.02`	
`...`	
`CANCEL(1)`	`; Online-Korrektur abwählen.`
`...`	

10.4.10 Online-Werkzeuglängenkorrektur ($AA_TOFF)

Funktion

Über die Systemvariable $AA_TOFF[ ] können die effektiven Werkzeuglängen entsprechend der drei Werkzeugrichtungen dreidimensional in Echtzeit überlagert werden.

Als Index werden die drei Geometrieachsbezeichner verwendet. Damit ist die Anzahl der aktiven Korrekturrichtungen durch die zur selben Zeit aktiven Geometrieachsen festgelegt.

Alle Korrekturen können gleichzeitig aktiv sein.

Syntax

```
N... TRAORI
N... TOFFON(X,<Offsetwert>)
N... WHEN TRUE DO $AA_TOFF[X]
N... TOFFON(Y,<Offsetwert>)
N... WHEN TRUE DO $AA_TOFF[Y]
N... TOFFON(Z,<Offsetwert>)
N... WHEN TRUE DO $AA_TOFF[Z]
```

Bedeutung

`TOFFON`:	Online-Werkzeuglängenkorrektur **aktivieren**	
	`X, Y, Z`:	Werkzeugrichtung, in der die Online-Werkzeuglängenkorrektur wirksam sein soll.
	`<Offsetwert>`:	Bei der Aktivierung kann für die entsprechende Korrekturrichtung ein Offsetwert angegeben werden, der sofort herausgefahren wird.
`TOFFOF`:	Online-Werkzeuglängenkorrektur **zurücksetzen** Die Korrekturwerte in der angegebenen Korrekturrichtung werden zurückgesetzt und es wird ein Vorlaufstopp ausgelöst.	
`$AA_TOFF[X]=<Wert>`:	Überlagerung in X-Richtung	
`$AA_TOFF[Y]=<Wert>`:	Überlagerung in Y-Richtung	
`$AA_TOFF[Z]=<Wert>`:	Überlagerung in Z-Richtung	

Beispiele

Beispiel 1: Anwahl der Werkzeuglängenkorrektur

Programmcode	Kommentar
N10 TRAORI(1)	; Transformation ein.
N20 TOFFON(Z)	; Aktivierung der Online-WZL-Korrektur für die Z-Werkzeugrichtung.
N30 WHEN TRUE DO $AA_TOFF[Z]=10 G4 F5	; Für die Z-Werkzeugrichtung wird eine WZL-Korrektur von 10 interpoliert.
N40 TOFFON(X)	; Aktivierung der Online-WZL-Korrektur für die X-Werkzeugrichtung.
N50 ID=1 DO $AA_TOFF[X]=$AA_IW[X2] G4 F5	; Für die X-Werkzeugrichtung wird eine Korrektur abhängig von der Position der Achse X2 ausgeführt.
...	
	; Aktuelle Korrektur in X-Richtung zuweisen. Für die X-Werkzeugrichtung wird die WZL-Korrektur wieder zu 0 zurückgefahren:
N100 XOFFSET=$AA_TOFF_VAL[X] N120 TOFFON(X,-XOFFSET) G4 F5	

Beispiel 2: Abwahl der Werkzeuglängenkorrektur

Programmcode	Kommentar
N10 TRAORI(1)	; Transformation ein.
N20 TOFFON(X)	; Aktivierung der Online-WZL-Korrektur für die X-Werkzeugrichtung.
N30 WHEN TRUE DO $AA_TOFF[X]=10 G4 F5	; Für die X-Werkzeugrichtung wird eine WZL-Korrektur von 10 interpoliert.
...	
N80 TOFFOF(X)	; Positionsoffset der X-Werkzeugrichtung wird gelöscht: ...$AA_TOFF[X]=0 Es wird keine Achse verfahren. Zur aktuellen Position im WKS wird der Positionsoffset entsprechend der aktuellen Orientierung hinzugerechnet.

10.4.11 Positionierbewegungen

Funktion

Achsen können vollkommen asynchron zum Teileprogramm aus Synchronaktionen heraus positioniert werden. Die Programmierung von Positionierachsen aus Synchronaktionen empfiehlt sich für zyklische Abläufe oder Vorgänge, die stark ereignisgesteuert sind. Aus Synchronaktionen heraus programmierte Achsen heißen **Kommandoachsen**.

Programmierung

Literatur:
/PG/ Programmierhandbuch Grundlagen; Kapitel "Wegangaben"
/FBSY/ Funktionsbeschreibung Synchronaktionen; "Starten von Kommandoachsen"

Parameter

Das Maßsystem für Positionieraufgaben in Synchronaktionen wird mit den G-Codes `G70`/`G71`/`G700`/`G710` festgelegt.

Durch Programmieren von G-Funktionen in der Synchronaktion kann die INCH/METRIC-Bewertung für die Synchronaktion unabhängig vom Teileprogrammkontext festgelegt werden.

10.4.12 Achse positionieren (POS)

Funktion

Die Positionierachsbewegung hat im Gegensatz zur Programmierung aus dem Teileprogramm keinen Einfluss auf die Abarbeitung des Teileprogramms.

Syntax

```
POS[Achse] = Wert
```

Bedeutung

`DO POS`	Kommandoachse starten/positionieren
`Achse`	Name der Achse, die verfahren werden soll
`Wert`	Angabe des zu verfahrenden Wertes (je nach Verfahrmodus)

Beispiele

Beispiel 1:

Programmcode	**Kommentar**
`ID=1 EVERY $AA_IM[B]>75 DO POS[U]=100`	`; Achse U abhängig vom Verfahrmodus inkrementell um 100 (inch/mm) bzw. auf Position 100 (inch/mm) vom Steuerungsnullpunkt verfahren.`
	`; Achse U um aus Hauptlaufvariablen berechneten Weg verfahren:`
`ID=1 EVERY $AA_IM[B]>75 DO POS[U]=$AA_MW[V]-$AA_IM[W]+13.5`	

Beispiel 2:

Programmumgebung beeinflusst Positionierweg der Positionierachse (keine G-Funktion im Aktionsteil der Synchronaktion):

Programmcode	**Kommentar**
`N100 R1=0`	
`N110 G0 X0 Z0`	
`N120 WAITP(X)`	
`N130 ID=1 WHENEVER $R==1 DO POS[X]=10`	
`N140 R1=1`	
`N150 G71 Z10 F10`	`; Z=10mm X=10mm`
`N160 G70 Z10 F10`	`; Z=254mm X=254mm`
`N170 G71 Z10 F10`	`; Z=10mm X=10mm`
`N180 M30`	

`G71` im Aktionsteil der Synchronaktion bestimmt den Positionierweg der Positionierachse eindeutig (metrisch), unabhängig von der Programmumgebung:

```
Programmcode                                          Kommentar
N100 R1=0
N110 G0 X0 Z0
N120 WAITP(X)
N130 ID=1 WHENEVER $R==1 DO G71 POS[X]=10
N140 R1=1
N150 G71 Z10 F10                                      ; Z=10mm X=10mm
N160 G70 Z10 F10                                      ; Z=254mm X=10mm (X positioniert immer auf 10mm)
N170 G71 Z10 F10                                      ; Z=10mm X=10mm
N180 M30
```

Soll die Achsbewegung nicht mit Satzanfang gestartet werden, kann der Override für die Achse aus einer Synchronaktion bis zum gewünschten Startzeitpunkt auf 0 gehalten werden:

```
Programmcode                                          Kommentar
WHENEVER $A_IN[1]==0 DO $AA_OVR[W]=0 G01 X10 Y25 F750 POS[W]=1500 FA=1000
                                                      ; Die Positionierachse wird so lange angehalten,
                                                        solange der digitale Eingang 1=0.
```

10.4.13 Position im vorgegebenen Referenzbereich (POSRANGE)

Funktion

Mit der Funktion POSRANGE( ) kann ermittelt werden, ob sich die aktuelle interpolierte Sollposition einer Achse, in einem Fenster um eine vorgegebene Referenzposition befindet. Die Positionsangaben können sich auf vorgebbare Koordinatensysteme beziehen.

Bei Abfrage der Achs-Istposition einer Modoluachse wird die Modulo-Korrektur berücksichtigt.

Hinweis

Die Funktion kann nur aus der Synchronaktion aufgerufen werden. Beim Aufruf aus dem Teileprogramm erfolgt der Alarm 14091 %1 Satz %2 Funktion nicht zulässig, Index: %3 mit dem Index 5 aufgerufen.

Syntax

```
BOOL POSRANGE(Achse, Refpos, Winlimit,[Coord])
```

Bedeutung

`BOOL POSRANGE`	Aktuelle Position der Kommandoachse ist im Fenster der vorgegebenen Referenzposition.
`AXIS <Achse>`	Achsbezeichner der Maschinen-, Kanal oder Geometrie-Achse
`REAL Refpos`	Referenzposition im Coord -Koordinatensystem
`REAL Winlimit`	Betrag, der die Grenze für das Positionsfenster ergibt
`INT Coord`	Optional ist das MKS aktiv. Möglich sind: 0 für MKS (Maschinenkoordinatensystem) 1 für BKS (Basiskoordinatensystem) 2 für ENS (Einstellbares Nullpunktsystem) 3 für WKS (Werkstückkoordinatensystem)

Funktionswert

Aktuelle Sollposition je nach Positionsangabe im vorgegebenen Koordinatensystem

`Funktionswert: TRUE`	wenn Refpos(Coord) - abs(Winlimit)
`Funktionswert: FALSE`	≤ Actpos(Coord) ≤ Refpos(Coord) + abs(Winlimit) sonst

10.4.14 Achse starten/stoppen (MOV)

Funktion

Mit MOV[Achse]=Wert kann eine Kommandoachse ohne Angabe einer Endposition gestartet werden. Die jeweilige Achse wird in die programmierte Richtung verfahren, bis durch einen neuen Bewegungs- oder Positionierbefehl eine andere Bewegung vorgegeben wird oder die Achse mit einem Stoppbefehl angehalten wird.

Syntax

```
MOV[Achse] = Wert
```

Bedeutung

`DO MOV`	Kommandoachsbewegung starten
`Achse`	Name der Achse, die gestartet werden soll
`Wert`	Startbefehl für Verfahr-/Stoppbewegung Das Vorzeichen bestimmt die Bewegungsrichtung
	Datentyp des Wertes ist INTEGER.
`Wert >0 (üblicherweise +1)`	positive Richtung
`Wert <0 (üblicherweise -1)`	negative Richtung
`Wert ==0`	Achsbewegung stoppen

Hinweis

Wird eine Teilungsachse mit `MOV[Achse] = 0` gestoppt, wird die Achse an nächster Teilungsposition angehalten.

Beispiel

Programmcode	**Kommentar**
`... DO MOV[U]=0`	`; Achse U wird gestoppt`

10.4.15 Achstausch (RELEASE, GET)

Funktion

Für einen Werkzeugwechsel können die betreffenden Kommandoachsen als Aktion einer Synchronaktion mit GET(Achse) angefordert werden. Der diesem Kanal zugeordnete Achstyp und das damit zu diesem Zeitpunkt verbundene Interpolationsrecht kann über die Systemvariable $AA_AXCHANGE_TYP abgefragt werden. Abhängig vom eigentlichen Zustand und vom Kanal der das aktuelle Interpolationsrecht dieser Achse besitzt, sind unterschiedliche Abläufe möglich.

Ist der Werkzeugwechsel vollzogen, dann kann diese Kommandoachse als Aktion einer Synchronaktion mit RELEASE(Achse) für den Kanal freigegeben werden.

Maschinenhersteller

Die betreffende Achse muss dem Kanal über Maschinendaten zugeordnet sein. Bitte beachten Sie die Angaben des Maschinenherstellers

Syntax

`GET(Achse[,Achse{,...}])` Achse anfordern

`RELAESE(Achse[,Achse{,...}])` Achse freigeben

Bedeutung

`DO RELEASE`	Achse als neutrale Achs freigeben
`DO GET`	Achse für Achstausch holen
`Achse`	Name der Achse, die gestartet werden soll

Beispiel Programmablauf für einen Achstausch zweier Kanäle

Die Achse Z ist im 1.Kanal und im 2. Kanal bekannt.

Programmablauf im 1. Kanal:

```
Programmcode                              Kommentar
WHEN TRUE DO RELEASE(Z)                   ; Z-Achse wird zur neutralen Achse
WHENEVER($AA_TYP[Z]==1) DO RDISABLE       ; Einlesesperre solange Z-Achse Programmachse
N110 G4 F0.1
WHEN TRUE DO GET(Z)                       ; Z-Achse wird wieder NC-Programm-Achse
WHENEVER($AA_TYP[Z]<>1) DO RDISABLE       ; Einlesesperre bis Z-Achse Programmachse ist
N120 G4 F0.1
WHEN TRUE DO RELEASE(Z)                   ; Z-Achse wird zur neutralen Achse
WHENEVER($AA_TYP[Z]==1) DO RDISABLE       ; Einlesesperre solange Z-Achse Programmachse
N130 G4 F0.1                              ;
N140 START(2)                             ; den 2. Kanal starten
```

Programmablauf im 2. Kanal:

```
Programmcode                                    Kommentar
WHEN TRUE DO GET(Z)                             ; ;Z-Achse in den 2. Kanal holen
WHENEVER($AA_TYP[Z]==0) DO RDISABLE             ; ;Einlesesperre solange Z-Achse in
                                                  anderem
                                                  ;Kanal
N210 G4 F0.1
WHEN TRUE DO GET(Z)                             ; ;Z-Achse wird NC-Programm-Achse
WHENEVER($AA_TYP[Z]<>1) DO RDISABLE             ; ;Einlesesperre bis Z-Achse
                                                  Programmachse ist
N220 G4 F0.1
WHEN TRUE DO RELEASE(Z)                         ; ;Z-Achse im 2. Kanal neutrale
                                                  Achse
WHENEVER($AA_TYP[Z]==1) DO RDISABLE             ; ;Einlesesperre solange Z-Achse
                                                  Programmachse
N230 G4 F0.1
N250 WAITM(10, 1, 2)                            ; mit Kanal 1 synchronisieren
```

Weiter Programmablauf im 1. Kanal:

```
Programmcode                                    Kommentar
N150 WAIM(10, 1, 2)                             ; mit Kanal 2 synchronisieren
WHEN TRUE DO GET(Z)                             ; Z-Achse in diesen Kanal holen
WHENEVER($AA_TYP[Z]==0) DO RDISABLE             ; Einlesesperre solange Z-Achse in
                                                  anderem
                                                  Kanal
N160 G4 F0.1
N199 WAITE(2)
N999 M30                                        ; warte auf Programmende im Kanal 2
```

Beispiel Achstausch im Technologiezyklus

Die Achse U ($MA_AUTO_GET_TYPE=2) ist im 1.Kanal und im 2. Kanal bekannt und aktuell hat der Kanal 1 das Interpolationsrecht. Im Kanal 2 wird folgender Technologiezyklus gestartet:

```
Programmcode                                    Kommentar
GET(U)                                          ; U-Achse in Kanal holen
POS[U]=100                                      ; U-Achse soll auf Position 100
                                                  verfahren
                                                  werden
```

Die Zeile der Kommandoachsbewegung POS[U] wird erst ausgeführt, wenn die U-Achse in den Kanal 2 geholt wurde.

Ablauf

Die zum Aktivierungszeitpunkt der Aktion `GET(Achse)` angeforderte Achse kann bezüglich des Achstyps für einen Achstausch mit der Systemvariable ($AA_AXCHANGE_TYP[<Achse>] gelesen werden:

- 0: Achse dem NC-Programm zugeordnet
- 1: Achse der PLC zugeordnet oder als Kommandoachse oder Pendelachse aktiv
- 2: ein anderer Kanal hat Interpolationsrecht
- 3: Achse ist neutrale Achse
- 4: neutrale Achse ist vom PLC kontrolliert
- 5: ein anderer Kanal hat Interpolationsrecht, Achse ist angefordert für das NC-Programm
- 6: ein anderer Kanal hat Interpolationsrecht, Achse ist angefordert als neutrale Achse
- 7: Achse der PLC oder als Kommandoachse oder Pendelachse aktiv, Achse ist angefordert für das NC-Programm
- 8: Achse der PLC oder als Kommandoachse oder Pendelachse aktiv, Achse ist angefordert als neutrale Achse

Randbedingungen

Die betreffende Achse muss dem Kanal über Maschinendaten zugeordnet sein.

Eine ausschließlich von der PLC kontrollierte Achse kann nicht dem NC–Programm zugeordnet werden.

Literatur:
/FB2/ Funktionshandbuch Erweiterungsfunktionen; Positionierachsen (P2)

Achse aus einem anderen Kanal mit der Aktion GET anfordern

Hat zum Aktivierungszeitpunkt der Aktion `GET` ein **anderer Kanal das Schreibrecht** (Interpolationsrecht) für die Achse ($AA_AXCHANGE_TYP[<Achse>] == 2), so wird die Achse mittels Achstausch von diesem Kanal angefordert ($AA_AXCHANGE_TYP[<Achse>]==6) und sobald als möglich dem anfordernden Kanal zugeordnet.

Sie nimmt dann den Zustand neutrale Achse an ($AA_AXCHANGE_TYP[<Achse>]==3).

Ein Reorganisieren im anfordernden Kanal findet nicht statt.

Zuordnung **als NC-Programm Achse mit Reorganisieren:**

Wurde die Achse bereits zum Aktivierungszeitpunkt der Aktion `GET` als neutrale Achse angefordert ($AA_AXCHANGE_TYP[<Achse>]==6), so wird die Achse für das NC–Programm angefordert ($AA_AXCHANGE_TYP[<Achse>]==5) und sobald als möglich dem NC–Programm des Kanals zugeordnet ($AA_AXCHANGE_TYP[<Achse>]==0).

Achse bereits dem angeforderten Kanal zugeordnet

Zuordnung **als NC-Programm Achse mit Reorganisieren:**

Ist die angeforderte Achse zum Aktivierungszeitpunkt bereits dem anfordernden Kanal zugeordnet, und im Zustand neutrale Achse – nicht von der PLC kontrolliert – ($AA_AXCHANGE_TYP[<Achse>]==3), so wird sie dem NC–Programm zugeordnet ($AA_AXCHANGE_TYP[<Achse>]==0).

Achse im Zustand als neutrale Achse ist vom PLC kontrolliert

Ist die Achse im Zustand neutrale Achse von PLC kontrolliert ($AA_AXCHANGE_TYP[<Achse>]==4), so wird die Achse als neutrale Achse angefordert ($AA_AXCHANGE_TYP[<Achse>] == 8), dabei wird die Achse abhängig vom Bit 0 im Maschinendatum MD 10722: AXCHANGE_MASK für einen automatischen Achstausch zwischen Kanälen gesperrt (Bit 0 == 0). Dies entspricht ($AA_AXCHANGE_STAT[<Achse>] == 1).

Achse ist als neutrale Kommandoachse bzw. Pendelachse aktiv oder der PLC zugeordnet

Ist die Achse als Kommandoachse bzw. Pendelachse aktiv oder der PLC zum Verfahren zugeordnet, PLC–Achse == konkurrierende Positionierachse, ($AA_AXCHANGE_TYP[<Achse>]==1),so wird die Achse als neutrale Achse angefordert ($AA_AXCHANGE_TYP[<Achse>] == 8), dabei wird die Achse abhängig vom Bit 0 im Maschinendatum MD 10722: AXCHANGE_MASK für einen automatischen Achstausch zwischen Kanälen gesperrt (Bit 0 == 0). Dies entspricht ($AA_AXCHANGE_STAT[<Achse>] ==1).

Eine erneute GET–Aktion fordert die Achse dann für das NC–Programm an ($AA_AXCHANGE_TYP[<Achse>] wird == 7).

Achse ist bereits dem NC-Programm zugeordnet

Ist die Achse bereits dem NC–Programm des Kanals zugeordnet ($AA_AXCHANGE_TYP[<Achse>]==0) oder ist diese Zuordnung angefordert, z.B. Achstausch vom NC–Programm ausgelöst ($AA_AXCHANGE_TYP[<Achse>]==5 bzw. $AA_AXCHANGE_TYP[<Achse>] == 7), so ergibt sich keine Zustandsänderung.

10.4.16 Axialer Vorschub (FA)

Funktion

Der axiale Vorschub für Kommandoachsen ist modal wirksam.

Syntax

```
FA[<Achse>]=<Wert>
```

Beispiel

Programmcode	Kommentar
`ID=1 EVERY $AA_IM[B]>75 DO POS[U]=100 FA[U]=990`	`; Vorschubwert fest vorgeben.`
	`; Vorschubwert aus Hauptlaufvariablen bilden:`
`ID=1 EVERY $AA_IM[B]>75 DO POS[U]=100 FA[U]=$AA_VACTM[W]+100`	

10.4.17 SW-Endschalter

Funktion

Die mit G25/G26 programmierte Arbeitsfeldbegrenzung wird in Abhängigkeit vom Settingdatum $SA_WORKAREA_PLUS_ENABLE für die Kommandoachsen berücksichtigt.

Ein- und Ausschalten der Arbeitsfeldbegrenzung über die G-Funktionen WALIMON/WALIMOF im Teileprogramm wirkt nicht auf Kommandoachsen.

10.4.18 Achskoordinierung

Funktion

Typischerweise wird eine Achse entweder aus dem Teileprogramm oder als Positionierachse aus der Synchronaktion bewegt.

Soll dieselbe Achse jedoch wechselweise aus dem Teileprogramm als Bahn- oder Positionierachse und aus Synchronaktionen verfahren werden, so erfolgt eine koordinierte Übergabe zwischen beiden Achsbewegungen.

Wird eine Kommandoachse anschließend aus dem Teileprogramm verfahren, so erfordert dies eine Reorganisation der Vorverarbeitung. Dies wiederum bedingt eine Unterbrechung der Teileprogrammbearbeitung, vergleichbar einem Vorlaufstopp.

Beispiel X-Achse wahlweise aus Teileprogramm und Synchronaktionen fahren

Programmcode	Kommentar
N10 G01 **X100** Y200 F1000	; X-Achse im Teileprogramm programmiert
…	
N20 ID=1 WHEN $A_IN[1]==1 DO **POS[X]=150** FA[X]=200	; Positionieren aus Synchronaktion starten, wenn ;digitaler Eingang ansteht
…	
CANCEL(1)	; Synchronaktion abwählen
…	
N100 G01 **X240** Y200 F1000	; X wird Bahnachse; vor Bewegung tritt Wartezeit aufgrund von Achsübergabe auf, falls digitaler Eingang 1 war und X aus Synchronaktion positioniert wurde.

Beispiel Verfahrbefehl für dieselbe Achse verändern

Programmcode	Kommentar
ID=1 EVERY $A_IN[1]>=1 DO POS[V]=100 FA[V]=560	; Positionieren aus Synchronaktion starten, wenn digitaler Eingang >= 1
ID=2 EVERY $A_IN[2]>=1 DO POS[V]=$AA_IM[V] FA[V]=790	; Achse läuft nach, der 2. Eingang wird gesetzt, d. h. Endposition und Vorschub für die Achse V wird bei zwei gleichzeitig aktiven Synchronaktionen fließend bei laufender Bewegung nachgeführt.

10.4.19 Istwertsetzen (PRESETON)

Funktion

Bei der Ausführung von PRESETON (Achse,Wert) wird die aktuelle Achsposition nicht verändert, es wird ihr ein neuer Wert zugewiesen.

PRESETON aus Synchronaktionen ist möglich für:

- Modulo-Rundachsen, die aus dem Teileprogramm gestartet wurden
- alle Kommandoachsen, die aus der Synchronaktion gestartet wurden

Syntax

```
DO PRESETON(Achse, Wert)
```

Bedeutung

`DO PRESETON`	Istwertsetzen in Synchronaktionen
`Achse`	Achse, deren Steuerungsnullpunkt verändert werden soll
`Wert`	Wert, um den der Steuerungsnullpunkt verändert wird

Einschränkungen für Achsen

`PRESETON` ist nicht möglich für Achsen, die an der Transformation beteiligt sind.

Ein- und dieselbe Achse kann nur zeitlich versetzt aus dem Teileprogramm oder einer Synchronaktion heraus bewegt werden, daher können bei der Programmierung einer Achse aus dem Teileprogramm Wartezeiten auftreten, falls diese Achse vorher in einer Synchronaktion programmiert war.

Wird die gleiche Achse wechselweise benutzt, so erfolgt eine koordinierte Übergabe zwischen beiden Achsbewegungen. Die Teileprogrammbearbeitung muss dazu unterbrochen werden.

Beispiel

Steuerungsnullpunkt einer Achse verschieben

Programmcode	**Kommentar**
`WHEN $AA_IM[a] >= 89.5 DO PRESETON(a4,10.5)`	`; Steuerungsnullpunkt der Achse a um 10.5 Längeneinheiten (inch bzw. mm) in positive Achsrichtung verschieben`

10.4.20 Spindelbewegungen

Funktion

Spindeln können vollkommen asynchron zum Teileprogramm aus Synchronaktionen heraus positioniert werden. Diese Art der Programmierung empfiehlt sich für zyklische Abläufe oder Vorgänge, die stark ereignisgesteuert sind.

Werden durch gleichzeitig aktive Synchronaktionen für eine Spindel konkurrierende Befehle vorgegeben, gilt der zeitlich letzte Spindelbefehl.

Beispiel Spindel starten/stoppen/positionieren

Programmcode	**Kommentar**
`ID=1 EVERY $A_IN[1]==1 DO M3 S1000`	`; Drehrichtung und Drehzahl einstellen`
`ID=2 EVERY $A_IN[2]==1 DO SPOS=270`	`; Spindel positionieren`

Beispiel Drehrichtung, Drehzahl einstellen/ Spindel positionieren

Programmcode	**Kommentar**
`ID=1 EVERY $A_IN[1]==1 DO M3 S300`	`; Drehrichtung und Drehzahl einstellen`
`ID=2 EVERY $A_IN[2]==1 DO M4 S500`	`; neue Drehrichtung und neue Drehzahl vorgeben`
`ID=3 EVERY $A_IN[3]==1 DO S1000`	`; neue Drehzahl vorgeben`
`ID=4 EVERY ($A_IN[4]==1) AND ($A_IN[1]==0) DO SPOS=0`	`; Spindel positionieren`

10.4.21 Mitschleppen (TRAILON, TRAILOF)

Funktion

Beim Einschalten der Kopplung aus der Synchronaktion kann die Leitachse in Bewegung sein. Die Folgeachse wird in diesem Fall auf die Sollgeschwindigkeit beschleunigt. Die Position der Leitachse zum Synchronisationszeitpunkt der Geschwindigkeiten ist Startposition für das Mitschleppen. Die Funktionalität des Mitschleppens wird im Kapitel "Bahnfahrverhalten" beschrieben.

Syntax

Mitschleppen einschalten

```
DO TRAILON(Folgeachse, Leitachse, Koppelfaktor)
```

Mitschleppen ausschalten

```
DO TRAILOF (Folgeachse, Leitachse, Leitachse 2)
```

Bedeutung

Asynchrones Mitschleppen aktivieren:

```
... DO TRAILON(FA, LA, Kf)
```

mit:
FA: Folgeachse
LA: Leitachse
Kf: Koppelfaktor

Asynchrones Mitschleppen deaktivieren:

```
... DO TRAILOF(FA, LA, LA2)
```

mit:
FA: Folgeachse
LA: Leitachse, optional
LA2: Leitachse 2, optional

```
... DO TRAILOF(FA)
```

Alle Kopplungen zur Folgeachse werden ausgeschaltet.

Beispiel

Programmcode	Kommentar
`$A_IN[1]==0 DO TRAILON(Y,V,1)`	`; Einschalten des 1. Mitschleppverbandes, wenn der digitale Eingang 1 ist`
`$A_IN[2]==0 DO TRAILON(Z,W,-1)`	`; Einschalten des 2. Mitschleppverbandes`
`G0 Z10`	`; Zustellung der Z- und W-Achse in entgegengesetzter ;Achsrichtung`
`G0 Y20`	`; Zustellung der Y- und V-Achse in gleicher Achsrichtung`
`...`	
`G1 Y22 V25`	`; Überlagerung einer abhängigen und unabhängigen ;Bewegung der Mitschleppachse "V"`
`...`	
`TRAILOF(Y,V)`	`; Ausschalten des 1. Mitschleppverbandes`
`TRAILOF(Z,W)`	`; Ausschalten des 2. Mitschleppverbandes`

Beispiel Konfliktvermeidung mit TRAILOF

Um eine gekoppelte Achse wieder für den Zugriff als Kanalachse frei zu schalten, muss vorher die Funktion `TRAILOF` aufgerufen werden. Es muss sichergestellt werden, dass `TRAILOF` ausgeführt ist, bevor der Kanal die betreffende Achse anfordert. Dies ist im folgenden Beispiel nicht der Fall

```
…
N50 WHEN TRUE DO TRAILOF(Y,X)
N60 Y100
…
```

In diesem Fall wird die Achse nicht rechtzeitig freigeben, da die satzweise wirksame Synchronaktion mit `TRAILOF` synchron mit `N60` aktiv wird, siehe Kapitel Bewegungssynchronaktion, "Struktur, allgemeine Grundlagen.
Zur Vermeidung von Konfliktsituationen sollte in folgender Weise verfahren werden

```
…
N50 WHEN TRUE DO TRAILOF(Y,X)
N55 WAITP(Y)
N60 Y100
```

10.4.22 Leitwertkopplung (LEADON, LEADOF)

Hinweis

Diese Funktion steht für SINUMERIK 828D nicht zur Verfügung!

Funktion

Die axiale Leitwertkopplung ist ohne Einschränkung in Synchronaktionen programmierbar. Das Ändern einer Kurventabelle bei bestehender Kopplung ohne einer vorherigen Neusynchronisation ist optional nur in Synchronaktionen möglich.

Syntax

Leitwertkopplung einschalten

```
DO LEADON(Folgeachse, Leitachse, Kurvtab. Nr., OVW)
```

Leitwertkopplung ausschalten

```
DO LEADOF(Folgeachse, Leitachse, Leitachse 2)
```

Bedeutung

```
Axiale Leitwertkopplung einschalten:
...DO LEADON(FA, LA, NR, OVW)
```

mit:
FA: Folgeachse
LA: Leitachse
NR: Nummer der gespeicherten Kurventabelle
OVW: Überschreiben einer bestehenden Kopplung mit geänderter Kurventabelle erlauben

```
Axiale Leitwertkopplung ausschalten:
...DO LEADOF(FA, LA)
```

mit:
FA: Folgeachse
LA: Leitachse, optional

```
... DO LEADOF(FA)
```

verkürzte Form ohne Angabe der Leitachse

Zugriff per Synchronaktionen freischalten RELEASE

Um eine zu koppelnde Achse für den Zugriff per Synchronaktion frei zu schalten, muss vorher die Funktion RELEASE für die zu koppelnde Folgeachse aufgerufen werden.

Beispiel:

```
RELEASE(XKAN)
ID=1 every SR1==1 to LEADON(CACH,XKAN,1)
```

OVW=0 (Defaultwert)

Einer bestehenden Kopplung kann ohne Neusynchronisation keine neue Kurventabelle vorgegeben werden. Eine Änderung der Kurventabelle erfordert das vorige Ausschalten der bestehenden Kopplung und ein erneutes Einschalten mit der geänderten Kurventabellennummmer. Dies bewirkt eine Neusynchronisation der Kopplung.

Ändern der Kurventabelle bei bestehender Kopplung mit OVW=1

Mit `OVW=1` kann einer bestehenden Kopplung eine neue Kurventabelle vorgegeben werden. Es erfolgt keine Neusynchronisation. Die Folgeachse versucht schnellst möglich dem durch die neue Kurventabelle vorgegebenen Positionswerte zu folgen.

Beispiel Fliegendes Trennen

Ein Strangmaterial, das sich stetig durch einen Arbeitsbereich einer Trennvorrichtung bewegt, soll in gleichlange Stücke zerteilt werden.

X-Achse: Achse in der sich das Strangmaterial bewegt. WKS
X1-Achse: Maschinenachse des Strangmaterials, MKS
Y-Achse: Achse, in der die Trennvorrichtung mit dem Strangmaterial "mitfährt"

Es wird angenommen, dass die Zustellung des Trennwerkzeuges und seine Steuerung durch PLC kontrolliert werden. Zur Feststellung der Synchronität zwischen Strangmaterial und Trennwerkzeug können die Signale der PLC-Nahtstelle ausgewertet werden.

Aktionen
Kopplung einschalten, `LEADON`
Kopplung ausschalten, `LEADOF`
Istwertsetzen, `PRESETON`

Programmcode	**Kommentar**
N100 R3=1500	; Länge eines abzutrennenden Teiles
N200 R2=100000 R13=R2/300	
N300 R4=100000	
N400 R6=30	; Startposition Y Achse
N500 R1=1	; Startbedingung für Bandachse
N600 LEADOF(Y,X)	; löschen einer evtl. bestehenden Kopplung
N700 CTABDEF(Y,X,1,0)	; Tabellendefinition
N800 X=30 Y=30	; Wertepaare
N900 X=R13 Y=R13	
N1000 X=2*R13 Y=30	
N1100 CTABEND	; Ende der Tabelledefinition
N1200 PRESETON(X1,0)	; PRESET zu Beginn
N1300 Y=R6 G0	; Startposition Y Achse, Achse ist linear
N1400 ID=1 WHENEVER $AA_IW[X]>$R3 DO PESETON(X1,0)	; PRESET nach Länge R3, neuer Beginn nach Abtrennen
N1500 RELEASE(Y)	
N1800 ID=6 EVERY $AA_IM[X]<10 DO LEADON(Y,X,1)	; Y über Tabelle 1 an X ankoppeln bei X <10
N1900 ID=10 EVERY $AA_IM[X]>$R3-30 DO EADOF(Y,X)	; > 30 vor gefahrener Trennlänge abkoppeln
N2000 WAITP(X)	
N2100 ID=7 WHEN $R1==1 DO MOV[X]=1 FA[X]=$R4	; Strangachse stetig in Bewegung setzen
N2200 M30	

10.4.23 Messen (MEAWA, MEAC)

Funktion

Im Vergleich zur Verwendung in Bewegungssätzen des Teileprogramms kann die Messfunktion aus Synchronaktionen beliebig ein- und ausgeschaltet werden.

Weitere Information zum Messen, siehe Spezielle Wegbefehle "Erweiterte Messfunktion

Syntax

Axiales Messen ohne Restweglöschen

```
MEAWA[Achse] = (Modus, Triggerereignis_1, ..._4)
```

Kontinuierliches Messen ohne Restweglöschen

```
MEAC[Achse] = (Modus, Messpeicher, Triggerereignis_1, ..._4))
```

Bedeutung

Programmcode	**Kommentar**		
DO MEAWA	;	Axiales Messen einschalten	
DO MEAC	;	Kontinuierliches Messen einschalten	
Achse	;	Name der Achse, für die gemessen wird	
Modus	;	Angabe der **Zehner**dekade 0: aktives Messsystem Anzahl der Messsysteme (je nach Modus) 1: 1. Messsystem 2: 2. Messsystem 3: beide Messsysteme	Angabe der **Einer**dekade 0: Messauftrag abbrechen bis zu 4 aktivierbare Triggerereignisse 1: gleichzeitig 2: nacheinander 3: wie 2 jedoch keine Überwachung von Triggerereignis1 beim Start
Triggerereignis_1 bis _4	;	: steigende Flanke Messtaster 1 -1: fallende Flanke Messtaster 1 optional 2: steigende Flanke Messtaster 2 optional -2: fallende Flanke Messtaster 2 optional	
Messspeicher	;	Nummer des FIFO-Umlaufspeichers	

10.4.24 Initialisierung von Feld-Variablen (SET, REP)

Funktion

In Synchronaktionen können Feld-Variablen initialisiert oder mit bestimmten Werten beschrieben werden.

Hinweis

Es sind nur Variablen möglich, die in Synchronaktionen beschreibbar sind. Maschinendaten lassen sich damit nicht initialisieren. Achsvariablen können nicht mit dem Wert NO_AXIS angegeben werden.

Syntax

```
DO FELD[n,m]=SET(<Wert1>,<Wert2>,...)
DO FELD[n,m]=REP(<Wert>)
```

Bedeutung

`FELD[n,m]`	Programmierte Feldindizes Die Initialisierung beginnt bei den programmierten Feldindizes. Bei 2-dimensionalen Feldern wird zuerst der 2. Index inkrementiert. Bei Achsindizes wird dieser nicht durchlaufen.
`SET(<Wert1>,<Wert2>,...)`	Initialisierung mit Wertelisten Das Feld wird von den programmierten Feldindizes an mit den Parametern von `SET` beschrieben. Es werden so viele Feldelemente zugewiesen, wie Werte programmiert sind. Werden mehr Werte programmiert als restliche Feldelemente vorhanden sind, wird ein Alarm ausgelöst
`REP(<Wert>)`	Initialisierung mit gleichen Werten Das Feld wird von den programmierten Feldindizes an bis zum Feldende wiederholt mit dem Parameter (`<Wert>`) von `REP` beschrieben.

Beispiel

Programmcode	**Kommentar**
`WHEN TRUE DO SYG_IS[0]=REP(0)`	`; Ergebnis:`
`WHEN TRUE DO SYG_IS[1]=SET(3,4,5)`	`; SYG_IS[0]=0`
	`SYG_IS[1]=3`
	`SYG_IS[2]=4`
	`SYG_IS[3]=5`
	`SYG_IS[4]=0`

10.4.25 Wartemarken setzen/löschen (SETM, CLEARM)

Funktion

In Synchronaktionen können Wartemarken gesetzt bzw. gelöscht werden, um z. B. Kanäle untereinander zu koordinieren.

Syntax

```
DO SETM(<Marker-Nummer>)
DO CLEARM(<Marker-Nummer>)
```

Bedeutung

`SETM`	Befehl zum Setzen der Wartemarke für den Kanal Der Befehl `SETM` kann im Teileprogramm und im Aktionsteil einer Synchronaktion geschrieben werden. Er setzt die Marke (`<Marker-Nummer>`) für den Kanal, in dem der Befehl läuft.
`CLEARM`	Befehl zum Löschen der Wartemarke für den Kanal Der Befehl `CLEARM` kann im Teileprogramm und im Aktionsteil einer Synchronaktion geschrieben werden. Er löscht die Marke (`<Marker-Nummer>`) für den Kanal, in dem der Befehl läuft.
`<Marker-Nummer>`	Wartemarke

10.4.26 Fehlerreaktionen (SETAL)

Funktion

Mit Synchronaktionen können Fehlerreaktionen programmiert werden. Dabei werden Zustandsvariablen abgefragt und entsprechende Aktionen ausgelöst.

Mögliche Reaktionen auf Fehlerzustände sind:

- Achse stoppen (Override=0)
- Alarm setzen

  Mit `SETAL` können Zyklen-Alarme aus Synchronaktionen gesetzt werden.
- Ausgang setzen
- Sämtliche in Synchronaktionen mögliche Aktionen

Syntax

Zyklen-Alarm setzen:

```
DO SETAL(<Alarm-Nummer>)
```

Bedeutung

`SETAL`	Befehl zum Setzen eines Zyklen-Alarms
`<Alarm-Nummer>`	Nummer des Alarms Zyklen-Alarmbereich für den Anwender: 65000 bis 69999

Beispiel

Programmcode	**Kommentar**
`ID=67 WHENEVER ($AA_IM[X1]-$AA_IM[X2])<4.567 DO $AA_OVR[X2]=0`	`; Wenn Sicherheitsabstand zwischen Achsen X1 und X2 zu klein, Achse X2 anhalten.`
`ID=67 WHENEVER ($AA_IM[X1]-$AA_IM[X2])<4.567 DO SETAL(65000)`	`; Wenn Sicherheitsabstand zwischen Achsen X1 und X2 zu klein, Alarm 65000 setzen.`

10.4.27 Fahren auf Festanschlag (FXS, FXST, FXSW, FOCON, FOCOF)

Funktion

Die Befehle für die Funktion "Fahren auf Festanschlag" werden mit den Teileprogrammbefehlen `FXS`, `FXST` und `FXSW` in Synchronaktionen/Technologiezyklen programmiert.

Die Aktivierung kann ohne Bewegung erfolgen, das Moment wird sofort begrenzt. Sobald die Achse sollwertseitig bewegt wird, wird auf Anschlag überwacht.

Fahren mit begrenztem Moment/Kraft (FOC)

Die Funktion gestattet es, über Synchronaktionen zu jeder Zeit Moment/Kraft zu ändern und kann modal oder satzbezogen aktiviert werden.

Syntax

```
FXS[<Achse>]
FXST[<Achse>]
FXSW[<Achse>]
FOCON[<Achse>]
FOCOF[<Achse>]
```

Bedeutung

`FXS`	Anwahl nur in Systemen mit digitalen Antrieben (VSA, HSA, HLA)
`FXST`	Veränderung des Klemmoments FXST
`FXSW`	Veränderung des Überwachungsfensters FXSW
`FOCON`	Aktivierung der modal wirksamen Moment/Kraft-Begrenzung
`FOCOF`	Abschalten der Moment/Kraft-Begrenzung
<Achse>	Achsbezeichner Zulässig sind: • Geometrieachs-Bezeichner • Kanalachs-Bezeichner • Maschinenachs-Bezeichner

Hinweis

Eine Anwahl darf nur einmal erfolgen.

Beispiele

Beispiel 1: Fahren auf Festanschlag (FXS), ausgelöst durch eine Sychronaktion

```
Programmcode                                                    Kommentar
Y-Achse:                                                        ; Statische Synchronaktionen
aktivieren:
N10 IDS=1 WHENEVER (($R1==1) AND $AA_FXS[y]==0)) D $R1=0 FXS[Y]=1 FXST[Y]=10 FA[Y]=200 POS[Y]=150
                                                                ; Durch das Setzen von $R1=1 wird für die
                                                                  Achse Y FXS aktiviert, das wirksame
                                                                  Moment auf 10% reduziert und eine
                                                                  Fahrbewegung in Richtung des Anschlags
                                                                  gestartet.
N11 IDS=2 WHENEVER ($AA_FXS[Y]==4) DO FXST[Y]=30                ; Sobald der Anschlag erkannt wurde
                                                                  ($AA_FXS[Y]==4), wird das Moment auf 30%
                                                                  heraufgesetzt.
N12 IDS=3 WHENEVER ($AA_FXS[Y]==1) DO FXST[Y]=$R0               ; Nach Erreichen des Anschlags wird das
                                                                  Moment abhängig von R0 gesteuert.
N13 IDS=4 WHENEVER (($R3==1) AND $AA_FXS[Y]==1)) DO FXS[Y]=0 FA[Y]=1000 POS[Y]=0
                                                                ; Abwahl in Abhängigkeit von R3 und
                                                                  zurückfahren.
N20 FXS[Y]=0 G0 G90 X0 Y0                                       ; Normaler Programmablauf:
N30 RELEASE(Y)                                                  ; Achse Y für die Bewegung in
                                                                  Synchronaktion freigeben.
N40 G1 F1000 X100                                               ; Bewegung einer anderen Achse.
N50 ...
N60 GET(Y)                                                      ; Achse Y wieder in den Bahnverbund
                                                                  aufnehmen
```

Beispiel 2: Aktivierung Momenten/Kraft-Begrenzung (FOC)

```
Programmcode                    Kommentar
N10 FOCON[X]                    ; Modale Aktivierung der Begrenzung.
N20 X100 Y200 FXST[X]=15        ; X fährt mit reduziertem Moment (15%).
N30 FXST[X]=75 X20              ; Änderung des Moment auf 75%, X fährt mit diesem
                                  begrenzten Moment.
N40 FOCOF[X]                    ; Abschalten der Momentenbegrenzung.
```

Weitere Informationen

Mehrfache Anwahl

Wird durch eine fehlerhafte Programmierung die Funktion nach der Aktivierung (`FXS[<Achse>]=1`) nochmals aufgerufen, wird folgender Alarm ausgelöst:

Alarm 20092 "Fahren auf Festanschlag noch aktiv"

Eine Programmierung, die in der Bedingung entweder $AA_FXS[ ] oder einen eigenen Merker (hier R1) abfragt, vermeidet eine mehrfache Aktivierung der Funktion "Teilprogrammfragment":

```
Programmcode
N10 R1=0
N20 IDS=1 WHENEVER ($R1==0 AND
$AA_IW[AX3] > 7) DO R1=1 FXST[AX1]=12
```

Satzbezogene Synchronaktionen

Durch die Programmierung einer satzbezogenen Synchronaktion kann Fahren auf Festanschlag während einer Anfahrbewegung zugeschaltet werden.

Beispiel:

Programmcode	Kommentar
N10 G0 G90 X0 Y0	
N20 WHEN $AA_IW[X] > 17 DO FXS[X]=1	; Erreicht X eine Position größer 17mm wird FXS aktiviert.
N30 G1 F200 X100 Y110	

Statische und satzbezogene Synchronaktionen

In statischen und satzbezogenen Synchronaktionen können die gleichen Befehle `FXS`, `FXST` und `FXSW` verwendet werden wie im normalen Teileprogrammablauf. Die Werte, die zugewiesen werden, können durch eine Berechnung entstanden sein.

10.4.28 Bestimmung des Bahntangentenwinkels in Synchronaktionen

Funktion

Die in Synchronaktionen lesbare Systemvariable $AC_TANEB (**T**angent **AN**gel at **E**nd of **B**lock) ermittelt den Winkel zwischen der Bahntangente im Endpunkt des aktuellen Satzes und der Bahntangente im Startpunkt des programmierten Folgesatzes.

Parameter

Der Tangentenwinkel wird stets positiv im Bereich 0.0 bis 180.0 Grad ausgegeben. Existiert kein Nachfolgesatz im Hauptlauf, so wird der Winkel -180.0 Grad ausgegeben.

Die Systemvariable `$AC_TANEB` sollte nicht für Sätze, die vom System erzeugt werden (Zwischensätze) gelesen werden. Zur Unterscheidung, ob es sich um einen programmierten Satz (Hauptsatz) handelt, dient die Systemvariable `$AC_BLOCKTYPE`.

Beispiel

```
ID=2 EVERY $AC_BLOCKTYPE==0 DO $SR1 = $AC_TANEB
```

10.4.29 Bestimmung des aktuellen Override

Funktion

Der aktuelle Override

(NC-Anteil) kann mit den Systemvariablen:

`$AA_OVR` Axial Override

`$AC_OVR` Bahnoverride

in Synchronaktionen gelesen und geschrieben werden.

Der von der PLC vorgegebene Override wird für Synchronaktionen in den Systemvariablen:

`$AA_PLC_OVR` Axial Override

`$AC_PLC_OVR` Bahnoverride

zum Lesen bereitgestellt.

Der resultierende Override

wird für Synchronaktionen in den Systemvariablen:

`$AA_TOTAL_OVR` Axial Override

`$AC_TOTAL_OVR` Bahnoverride

zum Lesen bereitgestellt.

Der resultierende Override errechnet sich als:

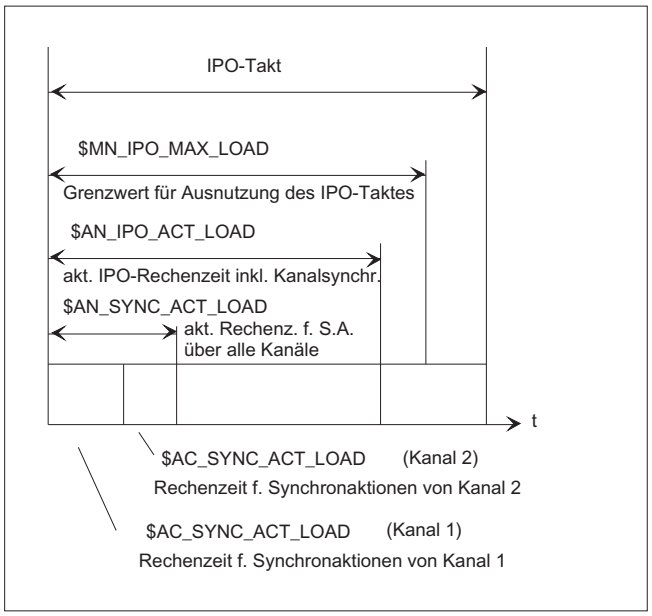
`$AA_OVR * $AA_PLC_OVR` bzw.

`$AC_OVR * $AC_PLC_OVR`

10.4.30 Auslastungsauswertung über Zeitbedarf der Synchronaktionen

Funktion

In einem Interpolationstakt müssen sowohl Synchronaktionen interpretiert als auch Bewegungen usw. von der NC berechnet werden. Mit den im Folgenden vorgestellten Systemvariablen können sich Synchronaktionen über die aktuellen Zeitanteile der Synchronaktionen am Interpolationstakt und über die Rechenzeit der Lageregler informieren.



Bedeutung

Die Variablen haben nur gültige Werte, wenn das Maschinendatum `$MN_IPO_MAX_LOAD` größer als 0 ist. Andernfalls geben die Variablen sowohl für SINUMERIK powerline als auch für solution line Systeme immer die Nettorechenzeit an, bei der die durch HMI erzeugten Unterbrechungen nicht mehr berücksichtigt werden. Die Nettorechenzeit ergibt sich aus:

- Synchronaktionszeit,
- Lageregelungszeit und der
- restlichen IPO-Rechenzeit ohne HMI bedingte Unterbrechungen

Die Systemvariablen enthalten immer die Werte des **vorhergehenden** IPO-Taktes

$AN_IPO_ACT_LOAD	aktuelle IPO-Rechenzeit (inkl. Synchronaktionen aller Kanäle)
$AN_IPO_MAX_LOAD	längste IPO-Rechenzeit (inkl. Synchronaktionen aller Kanäle)
$AN_IPO_MIN_LOAD	kürzeste IPO-Rechenzeit (inkl. Synchronaktionen aller Kanäle)
$AN_IPO_LOAD_PERCENT	aktuelle IPO-Rechenzeit im Verhältnis zum IPO-Takt (%).
$AN_SYNC_ACT_LOAD	aktuelle Rechenzeit für Synchronaktionen über alle Kanäle
$AN_SYNC_MAX_LOAD	längste Rechenzeit für Synchronaktionen über alle Kanäle
$AN_SYNC_TO_IPO	prozentualer Anteil der ges. Synchronaktionen an der gesamten IPO-Rechenzeit (über alle Kanäle)

$AC_SYNC_ACT_LOAD	aktuelle Rechenzeit für Synchronaktionen im Kanal
$AC_SYNC_MAX_LOAD	längste Rechenzeit für Synchronaktionen im Kanal
$AC_SYNC_AVERAGE_LOAD	durchschnittliche Rechenzeit für Synchronaktionen im Kanal

$AN_SERVO_ACT_LOAD	aktuelle Rechenzeit des Lagereglers
$AN_SERVO_MAX_LOAD	längste Rechenzeit des Lagereglers
$AN_SERVO_MIN_LOAD	kürzeste Rechenzeit des Lagereglers

Variable der Überlastmitteilung:

Über das Maschinendatum $MN_IPO_MAX_LOAD wird eingestellt, ab welcher IPO-Netto-Rechenzeit (in % vom IPO-Takt) die Systemvariable $AN_IPO_LOAD_LIMIT auf TRUE gesetzt werden soll. Unterschreitet die aktuelle Last diese Grenze wieder, so wird die Variable wieder auf FALSE gesetzt. Ist das Maschinendatum 0, so ist die gesamte Diagnosefunktion deaktiviert.

Durch die Auswertung von $AN_IPO_LOAD_LIMIT kann der Anwender eine eigene Strategie festlegen, um einen Ebenenüberlauf zu vermeiden.

10.5 Technologiezyklen

Funktion

Als Aktion in Synchronaktionen können auch Programme aufgerufen werden, die jedoch nur aus Funktionen aufgebaut sein dürfen, welche auch als Aktionen in Synchronaktionen zulässig sind. So aufgebaute Programme heißen Technologiezyklen.

Technologiezyklen werden als Unterprogramme in der Steuerung abgelegt.

In einem Kanal können parallel mehrere Technologiezyklen oder Aktionen bearbeitet werden.

Programmierung

Für die Programmierung von Technologiezyklen gelten folgende Regeln:

- Das Programmende wird mit `M02`/`M17`/`M30`/`RET` programmiert.
- Innerhalb einer Programmebene können alle in `ICYCOF` angegebenen Aktionen ohne Wartezyklen in einem Takt abgearbeitet werden.
- Es können bis zu 8 Technologiezyklen pro Synchronaktion hintereinander abgefragt werden.
- Technologiezyklen sind auch in satzweise wirksamen Synchronaktionen möglich.
- Es können sowohl `IF`-Kontrollstrukturen als auch Sprunganweisungen `GOTO`, `GOTOF` und `GOTOB` programmiert werden.
- Für Sätze mit `DEF`- und `DEFINE`-Anweisungen gilt:
  - `DEF`- und `DEFINE`-Anweisungen werden in Technologiezyklen überlesen.
  - Sie führen bei nicht korrekter oder unvollständiger Syntax zur Alarmmeldung.
  - Sie können, ohne selbst angelegt zu werden, ohne Alarmmeldung überlesen werden.
  - Sie werden mit Wertzuweisungen als Teileprogrammzyklus vollständig berücksichtigt.

Parameterübergabe

Eine Parameterübergabe an Technologiezyklen ist möglich. Berücksichtigt werden sowohl einfache Datentypen, die als Formal-Parameter "Call by Value" übergeben werden, als auch Standardeinstellungen, die beim Aufruf von Technologiezyklen wirksam werden. Dies sind:

- Programmierte Standardwerte, wenn kein Übergabeparameter programmiert ist.
- Standardparameter mit Initialwerte versehen.
- Nicht initialisierte Aktualparameter mit einem Standardwert übergeben.

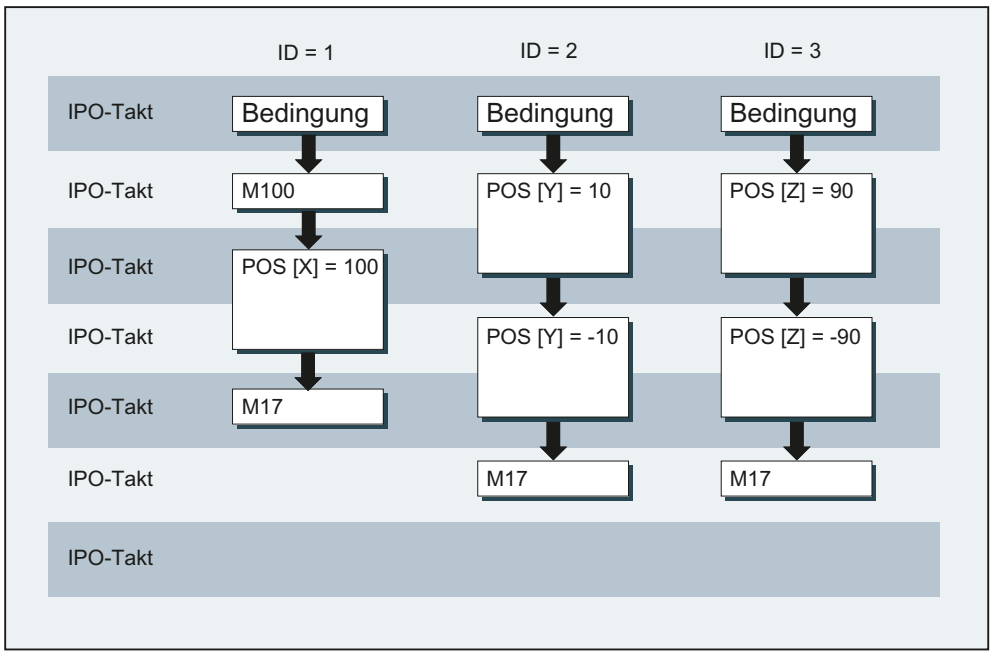
Ablauf

Technologiezyklen werden gestartet, sobald ihre Bedingungen erfüllt sind. Jede Zeile eines Technologiezyklus wird in einem separaten IPO-Takt abgearbeitet. Bei Positionierachsen werden zur Ausführung mehrere IPO-Takte benötigt. Andere Funktionen werden eintaktig ausgeführt. Im Technologiezyklus erfolgt die Abarbeitung der Sätze sequenziell.

Werden im gleichen Interpolationstakt Aktionen aufgerufen, die sich gegenseitig ausschließen, so wird diejenige Aktion aktiv, die von der Synchronaktion mit der höheren ID-Nummer aufgerufen wird.

Beispiele

Beispiel 1: Durch Setzen digitaler Eingänge werden Achs-Programme gestartet



Hauptprogramm:

Programmcode	**Kommentar**
`ID=1 EVERY $A_IN[1]==1 DO ACHSE_X`	`; Wenn Eingang 1 auf 1, starte Achsprogramm ACHSE_X.`
`ID=2 EVERY $A_IN[2]==1 DO ACHSE_Y`	`; Wenn Eingang 2 auf 1, starte Achsprogramm ACHSE_Y.`
`ID=3 EVERY $A_IN[3]==1 DO ACHSE_Z`	`; Wenn Eingang 3 auf 1, starte Achsprogramm ACHSE_Z.`
`M30`	

Achsprogramm ACHSE_X:

```
Programmcode
M100
POS[X]=100 FA[X]=300
M17
```

Achsprogramm ACHSE_Y:

```
Programmcode
POS[Y]=10 FA[Y]=200
POS[Y]=-10
M17
```

Achsprogramm ACHSE_Z:

```
Programmcode
POS[Z]=90 FA[Z]=250
POS[Z]=-90
M17
```

Beispiel 2: Verschiedene Programmsequenzen im Technologiezyklus

```
Programmcode
PROC CYCLE
N10 DEF REAL WERT=12.3
N15 DEFINE ABC AS G01
```

Beide Sätze werden ohne Alarm und ohne Anlegen der Variablen bzw. des Makros überlesen.

```
Programmcode
PROC CYCLE
N10 DEF REAL
N15 DEFINE ABC G01
```

Beide Sätze führen weiterhin zum NC-Alarm, weil die Syntax nicht korrekt geschrieben ist.

```
Programmcode
PROC CYCLE
N10 DEF AXIS ACHSE1=XX2
```

Ist die Achse XX2 nicht bekannt, wird Alarm 12080 ausgegeben. Andernfalls wird der Satz ohne Alarm und ohne Anlegen der Variablen überlesen.

```
Programmcode
PROC CYCLE
N10 DEF AXIS ACHSE1
N15 G01 X100 F1000
N20 DEF REAL WERT1
```

Der Satz `N20` führt immer zum Alarm 14500, da nach dem 1. Programmbefehl keine Definitionsanweisung folgen darf.

10.5.1 Kontext-Variable ($P_TECCYCLE)

Funktion

Mit Hilfe der Variablen $P_TECCYCLE können Programme in Synchronaktionsprogramme und Vorlaufprogramme unterteilt werden. Dadurch ist es möglich, syntaktisch korrekt geschriebene Sätze oder Programmsequenzen alternativ auch als Teileprogrammzyklus abzuarbeiten.

Kontext-Variable interpretieren

Die Systemvariable $P_TECCYCLE ermöglicht es, kontext-spezifische Interpretationen von Programmteilen in Technologiezyklen zu steuern:

```
IF $P_TECCYCLE==TRUE
...                         ; Programmsequenz für Technologiezyklus in der
                              Synchronaktion.
ELSE
...                         ; Programmsequenz für Teileprogrammzyklus.
ENDIF
```

Hinweis

Ein Satz mit fehlerhafter oder unerlaubter Programmsyntax sowie nicht bekannte Wertzuweisungen führen auch im Teileprogrammzyklus zu einer Alarmmeldung.

Beispiel

Programmsequenz mit Abfrage von $P_TECCYCLE im Technologiezyklus:

Programmcode	**Kommentar**
PROC CYCLE	
N10 DEF REAL WERT1	; Wird im Technologiezyklus überlesen.
N15 G01 X100 F1000	
N20 IF $P_TECCYCLE==TRUE	
...	; Programmsequenz für Technologiezyklus (ohne Variable WERT1).
N30 ELSE	
...	; Programmsequenz für Teileprogrammzyklus (Variable WERT1 ist vorhanden).
N40 ENDIF	

10.5.2 Call-by-Value-Parameter

Funktion

Technologiezyklen können mit Call-by-Value-Parametern definiert werden. Als Parameter sind einfache Datentypen wie INT, REAL, CHAR, STRING, AXIS und BOOL möglich.

Hinweis

Formal-Parameter, die Call-by-Value übergeben werden, können keine Felder sein.

Die Aktualparameter können auch aus Defaultparameter bestehen (siehe "Default-Parameter-Initialisierung (Seite 630)").

Syntax

```
ID=1 WHEN $AA_IW[X]>50 DO TEC(IVAL,RVAL,,SVAL,AVAL)
```

Bei nicht initialisierten Aktualparametern wird ein Defaultwert übergeben:

```
ID=1 WHEN $AA_IW[X]>50 DO TEC(IVAL,RVAL,,SYG_SS[0],AVAL)
```

10.5.3 Default-Parameter-Initialisierung

Funktion

Default-Parameter können in der PROC-Anweisung auch mit einem Initialwert versehen werden.

Syntax

Im Technologiezyklus Default-Parameter zuweisen:

```
PROC TEC (INT IVAL=1, REAL RVAL=1.0, CHAR CVAL='A', STRING[10] SVAL="ABC", AXIS
AVAL=X, BOOL BVAL=TRUE)
```

Wenn ein Aktualparameter aus einem Defaultparameter besteht, wird der Initialwert aus der PROC-Anweisung übergeben. Dies gilt sowohl im Teileprogramm, als auch in Synchronaktionen.

Beispiel

Programmcode	**Kommentar**
`TEC (IVAL, RVAL, SVAL, AVAL)`	`;  bei CVAL und BVAL gilt der Initialwert`

10.5.4 Steuerung der Abarbeitung von Technologiezyklen (ICYCOF, ICYCON)

Funktion

Zur Steuerung der zeitlichen Abarbeitung von Technologiezyklen dienen die Sprachbefehle ICYCOF und ICYCON.

Mit ICYCOF werden alle Sätze eines Technologiezykluses nur in einem Interpolationstakt abgearbeitet. Alle Aktionen, deren Ausführung mehrere Takte benötigen, führen bei ICYCOF zu parallelen Bearbeitungsprozessen.

Anwendung

Bei ICYCON können Kommandoachs-Bewegungen dazu führen, dass sich die Abarbeitung von einem Technologiezyklus verzögert. Ist dies unerwünscht dann können mit ICYCOF alle Aktionen ohne Wartezeiten in einem Interpolationstakt abgearbeitet werden.

Syntax

Für die zyklische Abarbeitung von Technologiezyklen gilt:

`ICYCON` jeder Satz von einem Technologiezyklus wird nach ICYCON in einem separaten IPO-Takt abgearbeitet

`ICYCOF` alle folgenden Sätze eines Technologiezykluses werden nach ICYCOF in einem IPO-Takt abgearbeitet

Hinweis

Die beiden Sprachbefehle ICYCON und ICYCOF wirken nur innerhalb der Programmebene. Im Teileprogramm werden beide Befehle ohne Reaktion einfach überlesen.

Beispiel für Abarbeitungsmode ICYCOF

Programmcode	**Kommentar**
IPO-Takt	; PROC TECHNOCYC
1.	; $R1=1
2.25	; POS[X]=100
26.	; ICYCOF
26.	; $R1=2
26.	; $R2=$R1+1
26.	; POS[X]=110
26.	; $R3=3
26.	; RET

10.5.5 Kaskadierungen von Technologiezyklen

Funktion

Es können bis zu 8 Technologiezyklen in Reihe geschaltet abgearbeitet werden. Damit sind in einer Synchronaktion mehrere Technologiezyklen programmierbar.

Syntax

```
ID=1 WHEN $AA_IW[X]>50 DO TEC1($R1) TEC2 TEC3(X)
```

Bearbeitungsreihenfolge

Die Technologiezyklen werden der Reihe nach (in Kaskade) von links nach rechts gemäß der oben angegebenen Programmierung abgearbeitet. Sollte ein Zyklus im Mode ICYCON abgearbeitet werden, so verzögert dieser alle nachfolgenden Bearbeitungen. Ein auftretender Alarm bricht alle nachfolgenden Aktionen ab.

10.5.6 Technologiezyklen in satzweisen Synchronaktionen

Funktion

Technologiezyklen sind auch in satzweisen Synchronaktion möglich.

Ist die Abarbeitungszeit eines Technologiezyklus länger als die Bearbeitungszeit des zugehörigen Satzes, so wird der Technologiezyklus beim Satzwechsel abgebrochen.

Hinweis

Ein Technologiezyklus verhindert nicht den Satzwechsel.

10.5.7 Kontrollstrukturen (IF)

Funktion

Für Verzweigungen in der Ablaufreihenfolge von Technologiezyklen können IF-Kontrollstrukturen in Synchronaktionen verwendet werden.

Syntax

```
IF <Bedingung>
$R1=1
[ELSE] optional
$R1=0
ENDIF
```

10.5.8 Sprunganweisungen (GOTO, GOTOF, GOTOB)

Funktion

In Technologiezyklen sind die Sprunganweisungen GOTO, GOTOF, GOTOB möglich. Die angegeben Labels müssen im Unterprogramm vorhanden sein, damit kein Alarm abgesetzt wird.

Hinweis

Labels und Satznummern dürfen nur Konstanten sein.

Syntax

Unbedingte Sprünge

`GOTO` Label, Satznummer

`GOTOF` Label, Satznummer

`GOTOB` Label, Satznummer

Sprunganweisungen und Sprungziele

`GOTO`	Springe erst vorwärts und anschließend rückwärts
`GOTOF`	Springe vorwärts
`GOTOB`	Springe rückwärts
`Label:`	Sprungmarke
`Satznummer`	Sprungziel zu diesen Satz
`N100`	Satznummer ist Nebensatz
`:100`	Satznummer ist Hauptsatz

10.5.9 Sperren, Freischalten, Zurücksetzen (LOCK, UNLOCK, RESET)

Funktion

Der Ablauf eines Technologiezyklus kann durch eine andere modale Synchronaktion gesperrt, wieder freigegeben oder zurückgesetzt werden.

Syntax

```
LOCK(<n1>,<n2>,...)
UNLOCK(<n1>,<n2>,...)
RESET(<n1>,<n2>,...)
```

Bedeutung

`LOCK`	Befehl zum Sperren von Synchronaktionen Die aktive Aktion wird unterbrochen.
`UNLOCK`	Befehl zum Freischalten von Synchronaktionen
`RESET`	Befehl zum Zurücksetzen von Technologiezyklen
`<n1>,<n2>,...`	Identifikationsnummern der Synchronaktionen bzw. Technologiezyklen, die gesperrt, freigeschaltet oder zurückgesetzt werden sollen.

Verriegelung von Synchronaktionen

Modale Synchronaktionen mit den ID-Nummern **<n> = 1 ... 64** können von der PLC verriegelt werden. Die zugehörige Bedingung wird damit nicht mehr ausgewertet und die Ausführung der zugehörigen Funktion im NCK gesperrt.

Mit einem Signal der PLC-Nahtstelle lassen sich pauschal alle Synchronaktionen sperren.

Hinweis

Eine programmierte Synchronaktion ist standardmäßig aktiv und kann gegen Überschreiben/Sperren über Maschinendatum gesichert werden.

Vom Maschinenhersteller festgelegte Synchronaktionen sollen vom Endkunden nicht beeinflusst werden können.

Beispiele

Beispiel 1: Synchronaktionen sperren (LOCK)

Programmcode

```
N100 ID=1 WHENEVER $A_IN[1]==1 DO M130
...
N200 ID=2 WHENEVER $A_IN[2]==1 DO LOCK(1)
```

Beispiel 2: Synchronaktionen freischalten (UNLOCK)

Programmcode

```
N100 ID=1 WHENEVER $A_IN[1]==1 DO M130
...
N200 ID=2 WHENEVER $A_IN[2]==1 DO LOCK(1)
...
N250 ID=3 WHENEVER $A_IN[3]==1 DO UNLOCK(1)
```

Beispiel 3: Technologiezyklus unterbrechen (RESET)

Programmcode

```
N100 ID=1 WHENEVER $A_IN[1]==1 DO M130
...
N200 ID=2 WHENEVER $A_IN[2]==1 DO RESET(1)
```

10.6 Synchronaktion löschen (CANCEL)

Funktion

Mit dem Befehl CANCEL kann eine modal oder statisch wirksame Synchronaktion aus dem Teileprogramm heraus abgebrochen (gelöscht) werden.

Wird eine Synchronaktion abgebrochen, währenddessen die daraus aktivierte Positionierachsbewegung noch aktiv ist, wird die Positionierachsbewegung abgeschlossen. Ist dies nicht erwünscht, kann die Achsbewegung mit axialem Restweglöschen vor dem CANCEL-Befehl abgebremst werden.

Syntax

```
CANCEL(<n1>,<n2>,...)
```

Bedeutung

CANCEL:	Befehl zum Löschen von programmierten Synchronaktionen
<n1>,<n2>,...:	Identifikationsnummern der zu löschenden Synchronaktionen **Hinweis:** Ohne Angabe von Identifikationsnummern werden **alle** modalen/statischen Synchronaktionen gelöscht.

Beispiele

Beispiel 1: Synchronaktion abbrechen

Programmcode	Kommentar
N100 ID=2 WHENEVER $A_IN[1]==1 DO M130	
...	
N200 CANCEL(2)	; Löscht die modale Synchronaktion Nr. 2.

Beispiel 2: Restweglöschen vor Abbrechen der Synchronaktion

Programmcode	Kommentar
N100 ID=17 EVERY $A_IN[3]==1 DO POS[X]=15 FA[X]=1500	; Positionierachsbewegung starten.
...	
N190 WHEN ... DO DELDTG(X)	; Positionierachsbewegung beenden.
N200 CANCEL(17)	; Löscht die modale Synchronaktion Nr. 17.

10.7 Steuerungsverhalten in bestimmten Betriebszuständen

POWER ON

Mit POWER ON sind grundsätzlich keine Synchronaktionen aktiv. Statische Synchronaktionen können mit einem von PLC gestarteten asynchronen Unterprogramm (`ASUP`) aktiviert werden.

Betriebsartenwechsel

Mit dem Schlüsselwort `IDS` aktivierte Synchronaktionen bleiben über Betriebsartenwechsel hinaus aktiv. Alle übrigen Synchronaktionen werden bei Betriebsartenwechsel inaktiv (z. B. Achse positionieren) und mit dem Repositionieren und Zurückschalten in den Automatik-Betrieb wieder aktiv.

RESET

Mit NC-RESET werden alle satzweise wirksamen und modalen Synchronaktionen beendet. Statische Synchronaktionen bleiben aktiv. Aus ihnen können neue Aktionen gestartet werden. Ist bei RESET eine Kommandoachsbewegung aktiv, so wird diese abgebrochen. Bereits ausgeführte Synchronaktionen vom WHEN-Typ werden nach RESET nicht mehr bearbeitet.

Verhalten nach RESET		
Synchronaktion/ Technologiezyklus	**modal / satzweise**	**statisch (IDS)**
	Aktive Aktion wird abgebrochen, Synchronaktionen werden gelöscht	Aktive Aktion wird abgebrochen, Technologiezyklus wird zurückgesetzt.
Achse/ positionierende Spindel	Bewegung wird abgebrochen.	Bewegung wird abgebrochen.
drehzahlgeregelte Spindel	$MA_SPIND_ACTIVE_AFTER_RESET==1: Spindel bleibt aktiv $MA_SPIND_ACTIVE_AFTER_RESET==0: Spindel stoppt.	
Leitwertkopplung	$MC_RESET_MODE_MASK, Bit13 == 1: Leitwertkopplung bleibt aktiv $MC_RESET_MODE_MASK, Bit13 == 0: Leitwertkopplung wird aufgelöst	
Messvorgänge	Aus Synchronaktionen gestartete Messvorgänge werden abgebrochen.	Aus statischen Synchronaktionen gestartete Messvorgänge werden abgebrochen.

NC-Stopp

Statische Synchronaktionen bleiben bei NC-Stopp aktiv. Aus statischen Synchronaktionen gestartete Bewegungen werden nicht abgebrochen. Zum aktiven Satz gehörige **programmlokale** Synchronaktionen bleiben aktiv, daraus gestartete Bewegungen werden abgebrochen.

Programmende

Programmende und Synchronaktion beeinflussen sich nicht gegenseitig. Laufende Synchronaktionen werden auch nach Programmende abgeschlossen. Im M30-Satz aktive Synchronaktionen bleiben im M30-Satz aktiv. Ist dies unerwünscht, muss die Synchronaktion vor Programmende mit `CANCEL` abgebrochen werden.

Verhalten nach Programmende		
Synchronaktion/ Technologiezyklus	**modal / satzweise → werden abgebrochen**	**statisch (IDS) → bleiben erhalten**
Achse/ positionierende Spindel	M30 wird verzögert, bis die Achse/Spindel steht.	Bewegung läuft weiter.
drehzahlgeregelte Spindel	Programmende: $MA_SPIND_ACTIVE_AFTER_RESET==1: Spindel bleibt aktiv $MA_SPIND_ACTIVE_AFTER_RESET==0: Spindel stoppt Bei Betriebsartenwechsel bleibt Spindel aktiv.	Spindel bleibt aktiv.
Leitwertkopplung	$MC_RESET_MODE_MASK, Bit13 == 1: Leitwertkopplung bleibt aktiv $MC_RESET_MODE_MASK, Bit13 == 0: Leitwertkopplung wird aufgelöst	Aus statischer Synchronaktion gestartete Kopplung bleibt erhalten.
Messvorgänge	Aus Synchronaktionen gestartete Messvorgänge werden abgebrochen.	Aus statischen Synchronaktionen gestartete Messvorgänge bleiben aktiv.

Satzsuchlauf

Während Satzsuchlauf werden Synchronaktionen aufgesammelt und bei NC-Start ausgewertet, die zugehörigen Aktionen gegebenenfalls gestartet. Statische Synchronaktionen wirken auch während des Satzsuchlaufs. Werden bei Satzsuchlauf mit `FCTDEF` programmierte Polynomkoeffizienten gefunden, werden diese direkt wirksam.

Programmunterbrechung durch asynchrones Unterprogramm ASUP

ASUP-Anfang:

Modale und statische Bewegungssynchronaktionen bleiben erhalten und sind auch im asynchronen Unterprogramm wirksam.

ASUP-Ende:

Wird das asynchrone Unterprogramm nicht mit REPOS fortgesetzt, wirken die im asynchronen Unterpogramm geänderten modalen und statischen Bewegungssynchronaktionen im Hauptprogramm weiter.

Repositionierung (REPOS)

Nach Repositionierung (REPOS) werden die im unterbrochenen Satz wirksamen Synchronaktionen wieder aktiv. Aus dem asynchronen Unterprogramm heraus geänderte modale Synchronaktionen sind nach REPOS bei Bearbeitung des Restsatzes nicht wirksam.

Mit `FCTDEF` programmierte Polynomkoeffizienten werden von asynchronen Unterprogrammen und REPOS nicht beeinflusst. Unabhängig, wo sie programmiert wurden, sind sie im asynchronen Unterprogramm und im Hauptprogramm auch nach Ausführung von REPOS jederzeit einsetzbar.

Verhalten bei Alarmen

Über Synchronaktionen gestartete Achs- und Spindelbewegungen werden abgebremst, wenn ein Alarm mit Bewegungsstopp aktiv ist. Alle weiteren Aktionen (wie z. B. Ausgang setzen) werden weiter ausgeführt.

Löst eine Synchronaktion selbst einen Alarm aus, dann kommt es zum Bearbeitungsabbruch und die nachfolgenden Aktionen dieser Synchronaktion werden nicht mehr ausgeführt. Ist die Synchronaktion modal wirksam, wird sie im nächsten Interpolationstaktakt nicht weiter bearbeitet. Der Alarm wird also nur einmal abgesetzt. Alle weiteren Synchronaktionen werden weiter bearbeitet.

Alarme, die Interpreterstopp als Alarmreaktion haben, wirken erst nach Abarbeiten der vordekodierten Sätze.

Löst ein Technologiezyklus einen Alarm mit Bewegungsstopp aus, so wird der Technologiezyklus nicht weiter bearbeitet.

Pendeln

11

11.1 Asynchrones Pendeln (OS, OSP1, OSP2, OST1, OST2, OSCTRL, OSNSC, OSE, OSB)

Funktion

Eine Pendelachse fährt zwischen den zwei Umkehrpunkten 1 und 2 mit gegebenem Vorschub hin und her, bis die Pendelbewegung abgeschaltet wird.

Andere Achsen können während der Pendelbewegung beliebig interpoliert werden. Über eine Bahnbewegung oder mit einer Positionierachse kann eine kontinuierliche Zustellung erreicht werden. Dabei besteht jedoch **kein Zusammenhang** zwischen der Pendel- und der Zustellbewegung.

Eigenschaften des asynchronen Pendelns

- Das asynchrone Pendeln ist achsspezifisch über Satzgrenzen hinweg wirksam.
- Über das Teileprogramm ist ein satzsynchrones Einschalten der Pendelbewegung gewährleistet.
- Eine gemeinsame Interpolation von mehreren Achsen und eine Überlagerung von Pendelstrecken sind nicht möglich.

Programmierung

Über die folgenden Befehle ist ein der Abarbeitung des NC-Programms entsprechendes Einschalten und Beeinflussen des asynchronen Pendelns vom Teileprogramm her möglich.

Die programmierten Werte werden satzsynchron im Hauptlauf in die entsprechenden Settingdaten eingetragen und bleiben bis zur nächsten Änderung wirksam.

Syntax

```
OSP1[<Achse>]=<Wert> OSP2[<Achse>]=<Wert>
OST1[<Achse>]=<Wert> OST2[<Achse>]=<Wert>
FA[<Achse>]=<Wert>
OSCTRL[<Achse>]=(<Setzoption>,<Rücksetzoption>)
OSNSC[<Achse>]=<Wert>
OSE[<Achse>]=<Wert>
OSB[<Achse>]=<Wert>
OS[<Achse>]=1
OS[<Achse>]=0
```

Bedeutung

`<Achse>` Name der Pendelachse

`OS` Pendeln ein-/ausschalten

Wert: 1 Pendeln **ein**schalten
0 Pendeln **aus**schalten

`OSP1` Position von Umkehrpunkt 1 festlegen

`OSP2` Position von Umkehrpunkt 2 festlegen

Hinweis:
Falls ein inkrementelles Verfahren aktiv ist, so wird die Position inkrementell zur letzten im NC-Programm programmierten entsprechenden Umkehrposition berechnet.

`OST1` Haltezeit im Umkehrpunkt 1 in [s] festlegen

`OST2` Haltezeit im Umkehrpunkt 2 in [s] festlegen

`<Wert>`: -2 Interpolation wird ohne Warten auf Genauhalt fortgesetzt
-1 Warten auf Genauhalt grob
0 Warten auf Genauhalt fein
>0 Warten auf Genauhalt fein und anschließend Abwarten der angegebenen Haltezeit

Hinweis:
Die Einheit für die Haltezeit ist identisch mit der über `G4` programmierten Haltezeit.

`FA` Vorschubgeschwindigkeit festlegen

Als Vorschubgeschwindigkeit gilt die definierte Vorschubgeschwindigkeit der Positionierachse. Ist keine Vorschubgeschwindigkeit definiert, gilt der im Maschinendatum hinterlegte Wert.

`OSCTRL`

Setz- und Rücksetzoptionen angeben

Die Optionswerte 0 - 3 verschlüsseln das Verhalten an den Umkehrpunkten beim Ausschalten. Es kann eine der Varianten 0 - 3 ausgewählt werden. Die übrigen Einstellungen sind nach Bedarf kombinierbar mit der gewählten Variante. Mehrere Optionen werden durch Pluszeichen (+) aneinandergefügt.

`<Wert>`:

0 Beim Abschalten der Pendelbewegung im nächsten Umkehrpunkt stoppen (Voreinstellung)

Hinweis:
Nur durch Rücksetzen der Werte 1 und 2 möglich.

1 Beim Abschalten der Pendelbewegung in Umkehrpunkt 1 stoppen

2 Beim Abschalten der Pendelbewegung in Umkehrpunkt 2 stoppen

3 Beim Abschalten der Pendelbewegung keinen Umkehrpunkt anfahren, falls keine Ausfeuerungshübe programmiert sind

4 Nach dem Ausfeuern Endposition anfahren

8 Wird die Pendelbewegung durch Restweglöschen abgebrochen, sollen anschließend Ausfeuerungshübe abgearbeitet und ggf. die Endposition angefahren werden.

16 Wird die Pendelbewegung durch Restweglöschen abgebrochen, soll wie beim Abschalten die entsprechende Umkehrposition angefahren werden.

32 Geänderter Vorschub ist erst ab dem nächsten Umkehrpunkt aktiv

64 FA gleich 0, FA = 0: Wegüberlagerung ist aktiv

FA ungleich 0, FA <> 0: Geschwindigkeitsüberlagerung ist aktiv

128 Bei Rundachse DC (kürzester Weg)

256 Ausfeuerhub wird als Doppelhub ausgeführt.(Standard)
1=Ausfeuerhub wird als Einzelhub ausgeführt.

512 Zuerst Startposition anfahren

OSNSC	Anzahl der Ausfeuerungshübe festlegen
OSE	Endposition (im WKS) festlegen, die nach Ausschalten des Pendelns angefahren werden soll **Hinweis:** Bei Programmierung von OSE wird für OSCTRL implizit Option 4 wirksam.
OSB	Startposition (im WKS) festlegen, die vor Einschalten des Pendelns angefahren werden soll Die Startposition wird vor Umkehrpunkt 1 angefahren. Stimmt die Startposition mit der Umkehrposition 1 überein, so wird als nächstes die Umkehrposition 2 angefahren. Beim Erreichen der Startposition wirkt keine Haltezeit, auch wenn die Startposition mit der Umkehrposition 1 übereinstimmt, stattdessen wird auf Genauhalt fein gewartet. Eine eingestellte Genauhaltbedingung wird eingehalten. **Hinweis:** Damit die Startposition angefahren wird, muss im Settingdatum SD43770 $SA_OSCILL_CTRL_MASK Bit 9 gesetzt sein.

Beispiele

Beispiel 1: Pendelachse soll zwischen zwei Umkehrpunkten pendeln

Die Pendelachse Z soll zwischen Position 10 und 100 pendeln. Umkehrpunkt 1 soll mit Genauhalt fein, Umkehrpunkt 2 mit Genauhalt grob angefahren werden. Der Vorschub für die Pendelachse soll 250 betragen. Am Ende der Bearbeitung sollen 3 Ausfeuerungshübe erfolgen und die Pendelachse soll die Endposition 200 ansteuern. Der Vorschub für die Zustellachse soll 1 betragen, das Ende der Zustellung in X-Richtung soll bei Position 15 erreicht sein.

```
Programmcode                              Kommentar
WAITP(X,Y,Z)                              ; Ausgangsstellung.
G0 X100 Y100 Z100                         ; Umschalten in Positionierachsbetrieb.
WAITP(X,Z)
OSP1[Z]=10 OSP2[Z]=100                    ; Umkehrpunkt 1, Umkehrpunkt 2.
OSE[Z]=200                                ; Endposition.
OST1[Z]=0 OST2[Z]=-1                      ; Haltezeit an U1: Genauhalt fein
                                          ; Haltezeit an U2: Genauhalt grob
FA[Z]=250 FA[X]=1                         ; Vorschub Pendelachse, Vorschub
                                            Zustellachse.
OSCTRL[Z]=(4,0)                           ; Setzoptionen.
OSNSC[Z]=3                                ; 3 Ausfeuerhübe.
OS[Z]=1                                   ; Pendeln starten.
WHEN $A_IN[3]==TRUE DO DELDTG(X)          ; Restweglöschen.
POS[X]=15                                 ; Ausgangsstellung X-Achse
POS[X]=50                                   Endstellung X-Achse.
OS[Z]=0                                   ; Pendeln stoppen.
M30
```

Hinweis

Der Befehlsfolge `OSP1[Z]=...` bis `OSNCS[Z]=...` kann auch in einem Satz programmiert werden.

Beispiel 2: Pendeln mit Online-Änderung der Umkehrposition

Die für das asynchrone Pendeln erforderlichen Settingdaten können im Teileprogramm eingestellt werden.

Werden im Teileprogramm die Settingdaten direkt beschrieben, so wird die Änderung schon zum Vorlaufzeitpunkt wirksam. Synchrones Verhalten kann über einen Vorlaufstopp (`STOPRE`) erreicht werden.

Programmcode	**Kommentar**
`$SA_OSCILL_REVERSE_POS1[Z]=-10`	
`$SA_OSCILL_REVERSE_POS2[Z]=10`	
`G0 X0 Z0`	
`WAITP(Z)`	
`ID=1 WHENEVER $AA_IM[Z] < $$AA_OSCILL_REVERSE_POS1[Z] DO $AA_OVR[X]=0`	`; Wenn der Istwert der`
`ID=2 WHENEVER $AA_IM[Z] < $$AA_OSCILL_REVERSE_POS2[Z] DO $AA_OVR[X]=0`	`Pendelachse den Umkehrpunkt überschritten hat, wird die Zustellachse angehalten.`
`OS[Z]=1 FA[X]=1000 POS[X]=40`	`; Pendeln einschalten.`
`OS[Z]=0`	`; Pendeln ausschalten.`
`M30`	

Weitere Informationen

Pendelachse

Für die Pendelachse gilt:

- Jede Achse kann als Pendelachse benutzt werden.
- Gleichzeitig können mehrere Pendelachsen aktiv sein (maximal: Anzahl der Positionierachsen).
- Für die Pendelachse ist immer - unabhängig vom im Programm aktuell gültigen G-Befehl - Linearinterpolation `G1` aktiv.

Die Pendelachse kann:

- Eingangsachse für die dynamische Transformation sein
- Führungsachse bei Gantry- und Mitschleppachsen sein
- verfahren werden:
  - ohne Ruckbegrenzung (BRISK)

    oder
  - mit Ruckbegrenzung (SOFT)

    oder
  - mit geknickter Beschleunigungskennlinie (wie Positionierachsen)

Pendelumkehrpunkte

Bei der Festlegung der Pendelpositionen sind die aktuellen Verschiebungen zu beachten:

- Absolute Angabe

  `OSP1[Z]=<Wert>`

  Position Umkehrpunkt = Summe der Verschiebungen + programmierter Wert
- Relative Angabe

  `OSP1[Z]=IC(<Wert>)`

  Position Umkehrpunkt = Umkehrpunkt 1 + programmierter Wert

Beispiel:

Programmcode

```
N10 OSP1[Z]=100 OSP2[Z]=110
...
...
N40 OSP1[Z]=IC(3)
```

WAITP

Soll mit einer Geometrieachse gependelt werden, so muss diese mit WAITP zum Pendeln freigegeben werden.

Nach beendetem Pendeln wird mit WAITP die Pendelachse wieder als Positionierachse eingetragen und kann wieder normal verwendet werden.

Pendeln mit Bewegungssynchronaktionen und Haltezeiten

Nach Ablauf der eingestellten Haltezeiten findet beim Pendeln der interne Satzwechsel statt (sichtbar an den neuen Restwegen der Achsen). Beim Satzwechsel wird die Ausschaltfunktion überprüft. Dabei wird nach der eingestellten Steuereinstellung für den Bewegungsablauf (OSCTRL) die Ausschaltfunktion festgelegt. *Dieses Zeitverhalten ist durch den Vorschuboverride beeinflussbar.*

Unter Umständen wird danach noch ein Pendelhub ausgeführt, bevor die Ausfeuerungshübe gestartet oder die Endposition angefahren wird. *Es entsteht dabei der Eindruck, es verändert sich das Ausschaltverhalten. Dies ist aber nicht der Fall.*

11.2 Über Synchronaktionen gesteuertes Pendeln (OSCILL)

Funktion

Bei dieser Art des Pendelns ist nur an den Umkehrpunkten bzw. innerhalb definierter Umkehrbereiche eine Zustellbewegung zugelassen.

Je nach Anforderung kann die Pendelbewegung während der Zustellung

- fortgeführt oder
- angehalten werden, bis die Zustellung vollständig ausgeführt ist.

Syntax

1. **Parameter für das Pendeln festlegen**
2. **Bewegungssynchronaktionen definieren**
3. **Achsen zuordnen, Zustellung festlegen**

Bedeutung

`OSP1[<Pendelachse>]=`	Position des Umkehrpunkts 1
`OSP2[<Pendelachse>]=`	Position des Umkehrpunkts 2
`OST1[<Pendelachse>]=`	Haltezeit in Umkehrpunkt 1 in Sekunden
`OST2[<Pendelachse>]=`	Haltezeit in Umkehrpunkt 2 in Sekunden
`FA[<Pendelachse>]=`	Vorschub der Pendelachse
`OSCTRL[<Pendelachse>]=`	Setz- bzw. Rücksetzoptionen
`OSNSC[<Pendelachse>]=`	Anzahl der Ausfeuerungshübe
`OSE[<Pendelachse>]=`	Endposition
`WAITP(<Pendelachse>)`	Achse für das Pendeln freigeben

Achszuordnung, Zustellung

```
OSCILL[<Pendelachse>]=(<Zustellachse 1>,<Zustellachse 2>,<Zustellachse 3>)
POSP[<Zustellachse>]=(<Endpos>,<Teillänge>,<Modus>)
```

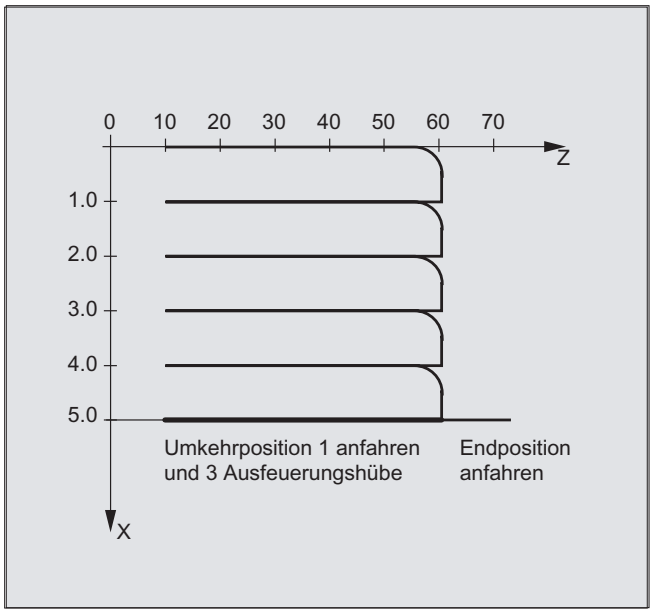
`OSCILL`:	Zustellachse(n) der Pendelachse zuordnen
`POSP`:	Gesamt- und Teilzustellungen festlegen (siehe Kapitel Datei- und Programmverwaltung)
`Endpos`:	Endposition für die Zustellachse, nachdem alle Teilzustellungen abgefahren sind.
`Teillänge`:	Größe der Teilzustellung am Umkehrpunkt/Umkehrbereich
`Modus`:	Aufteilung der Gesamtzustellung in Teilzustellungen
	= zwei gleich große Restschritte (Voreinstellung);
	= alle Teilzustellungen gleich groß

Bewegungssynchronaktionen

`WHEN… … DO`	wenn…, dann…
`WHENEVER … DO`	immer wenn…, dann…

Beispiel

Im Umkehrpunkt 1 soll keine Zustellung erfolgen. Beim Umkehrpunkt 2 soll die Zustellung bereits im Abstand ii2 vor dem Umkehrpunkt 2 erfolgen und die Pendelachse im Umkehrpunkt nicht auf das Beenden der Teilzustellung warten. Die Achse Z ist Pendelachse und die Achse X Zustellachse.



1. Parameter für das Pendeln

Programmcode	Kommentar
`DEF INT ii2`	`; Variable für Umkehrbereich 2 definieren`
`OSP1[Z]=10 OSP2[Z]=60`	`; Umkehrpunkt 1 und 2 definieren`
`OST1[Z]=0 OST2[Z]=0`	`; Umkehrpunkt 1: Genauhalt fein`
	`Umkehrpunkt 2: Genauhalt fein`
`FA[Z]=150 FA[X]=0.5`	`; Vorschub Pendelachse Z, Vorschub Zustellachse X`
`OSCTRL[Z]=(2+8+16,1)`	`; Pendelbewegung abschalten im Umkehrpunkt 2; nach RWL Ausfeuern und Endposition anfahren; nach RWL entsprechende Umkehrposition anfahren`
`OSNC[Z]=3`	`; Ausfeuerungshübe`
`OSE[Z]=70`	`; Endposition = 70`
`ii2=2`	`; Umkehrbereich einstellen`
`WAITP(Z)`	`; Erlaube Pendeln für Z-Achse`

2. Bewegungssynchronaktion

Programmcode	Kommentar
WHENEVER $AA_IM[Z]<$SA_OSCILL_REVERSE_POS2[Z] DO -> $AA_OVR[X]=0 $AC_MARKER[0]=0	; Immer wenn die aktuelle Position der Pendelachse Z im MKS kleiner als der Beginn des Umkehrbereichs 2 ist, dann setze den axialen Override der Zustellachse X auf 0% und den Merker mit dem Index 0 auf den Wert 0.
WHENEVER $AA_IM[Z]>=$SA_OSCILL_REVERSE_POS2[Z] DO $AA_OVR[Z]=0	; Immer wenn die aktuelle Position der Pendelachse Z im MKS größer gleich der Umkehrposition 2 ist, dann setze den axialen Override der Pendelachse Z auf 0%.
WHENEVER $AA_DTEPW[X]==0 DO $AC_MARKER[0]=1	; Immer wenn der Restweg der Teilzustellung gleich 0 ist, dann setze den Merker mit dem Index 0 auf den Wert 1.
WHENEVER $AC_MARKER[0]==1 DO $AA_OVR[X]=0 $AA_OVR[Z]=100	; Immer wenn der Merker mit dem Index 0 gleich 1 ist, dann setze den axialen Override der Zustellachse X auf 0%. Damit wird eine zu frühe Zustellung verhindert (Pendelachse Z hat den Umkehrbereich 2 noch nicht wieder verlassen, die Zustellachse X ist aber bereit für eine erneute Zustellung). Setze den axialen Override der Pendelachse Z von 0% (Aktion der 2. Synchronaktion) zum Verfahren wieder auf 100%.

-> muss in einem Satz programmiert werden

3. Pendeln starten

Programmcode	Kommentar
OSCILL[Z]=(X) POSP[X]=(5,1,1)	; Starten der Achsen Der Pendelachse Z wird die Achse X als Zustellachse zugewiesen. Die Achse X soll bis Endposition 5 in Schritten von 1 fahren.
M30	; Programmende

Beschreibung

1. **Pendelparameter festlegen**
   Vor dem Bewegungssatz, der die Zuordnung von Zustell- und Pendelachse sowie die Festlegung der Zustellung enthält, sind die Parameter für das Pendeln festzulegen (siehe "Asynchrones Pendeln").
2. **Bewegungssynchronaktionen festlegen**
   Über Synchronbedingungen erfolgt:
   Zustellung unterdrücken, bis sich die Pendelachse innerhalb eines Umkehrbereichs (ii1, ii2) oder an einem Umkehrpunkt (U1, U2) befindet.
   Pendelbewegung während der Zustellung im Umkehrpunkt **anhalten**.
   Pendelbewegung nach beendeter Teilzustellung wieder **starten**.
   Start der nächsten Teilzustellung festlegen.
3. **Pendeln- und Zustellachse zuordnen** sowie **Gesamt- und Teilzustellung** festlegen.

Pendelparameter festlegen

Zuordnung von Pendel- und Zustellachse: OSCILL

```
OSCILL[Pendelachse] = (Zustellachse1, Zustellachse2, Zustellachse3)
```

Mit dem Befehl `OSCILL` erfolgen die Achszuordnungen und der Start der Pendelbewegung.

Maximal können einer Pendelachse 3 Zustellachsen zugewiesen werden.

Hinweis

Vor dem Start des Pendelns müssen die Synchronbedingungen für das Verhalten der Achsen festgelegt sein.

Zustellungen festlegen: POSP

```
POSP[Zustellachse] = (Endpos, Teillänge, Modus)
```

Mit dem Befehl `POSP` werden der Steuerung mitgeteilt:

- Gesamtzustellung (über die Endposition)
- Die Größe der jeweiligen Teilzustellung am Umkehrpunkt bzw. im Umkehrbereich
- Das Teilzustellverhalten bei Erreichen der Endposition (über den Modus)

Modus = 0	Für die beiden letzten Teilzustellungen erfolgt eine Aufteilung des verbleibenden Weges bis zum Zielpunkt auf 2 gleich große Restschritte (Voreinstellung).
Modus = 1	Alle Teilzustellungen sind gleich groß. Sie werden aus der Gesamtzustellung berechnet.

Bewegungssynchronaktionen festlegen

Die im folgenden ausgeführten Bewegungssynchronaktionen werden ganz allgemein zum Pendeln verwendet.

Sie finden Beispiellösungen für die Lösung von einzelnen Anforderungen, die Ihnen als Bausteine für die Erstellung von anwenderspezifischen Pendelbewegungen dienen.

Hinweis

Im Einzelfall können die Synchronbedingungen auch anders programmiert werden.

Schlüsselwörter

`WHEN … DO …`	wenn…, dann…
`WHENEVER … DO`	immer wenn…, dann…

Funktionen

Mit den im folgenden detailliert beschriebenen Sprachmitteln können Sie folgende Funktionen

realisieren:

1. Zustellung im Umkehrpunkt.
2. Zustellung im Umkehrbereich.
3. Zustellung in beiden Umkehrpunkten.
4. Anhalten der Pendelbewegung im Umkehrpunkt.
5. Pendelbewegung wieder starten.
6. Teilzustellung nicht zu früh starten.

Für alle hier beispielhaft dargestellten Synchronaktionen gelten die Annahmen:

- Umkehrpunkt 1 < Umkehrpunkt 2
- Z = Pendelachse
- X = Zustellachse

Hinweis

Für nähere Erläuterungen, siehe Kapitel Bewegungssynchronaktionen.

Pendeln- und Zustellachse zuordnen sowie Gesamt- und Teilzustellung festlegen

Zustellung im Umkehrbereich

Die Zustellbewegung soll innerhalb eines Umkehrbereichs beginnen, bevor der Umkehrpunkt erreicht ist.

Diese Synchronaktionen verhindern die Zustellbewegung, bis sich die Pendelachse in einem Umkehrbereich befindet.

Unter den gegebenen Annahmen (siehe oben) ergeben sich folgende Anweisungen:

Umkehrbereich 1:

```
WHENEVER
$AA_IM[Z]>$SA_OSCILL_RESE
RVE_POS1[Z]+ii1 DO
$AA_OVR[X] = 0
```

Immer wenn die aktuelle Position der Pendelachse im MKS größer als der Beginn des Umkehrbereichs 1 ist, dann setze den axialen Override der Zustellachse auf 0%.

Umkehrbereich 2:

```
WHENEVER
$AA_IM[Z]<$SA_OSCILL_RESE
RVE_POS2[Z]+ii2 DO
$AA_OVR[X] = 0
```

Immer wenn die aktuelle Position der Pendelachse im KS kleiner als der Beginn des Umkehrbereichs 2 ist, dann setze den axialen Override der Zustellachse auf 0%.

Zustellung im Umkehrpunkt

Solange die Pendelachse den Umkehrpunkt nicht erreicht hat, findet keine Bewegung der Zustellachse statt.

Unter den gegebenen Annahmen (siehe oben) ergeben sich folgende Anweisungen:

Umkehrbereich 1:

```
WHENEVER
$AA_IM[Z]<>$SA_OSCILL_RES
ERVE_POS1[Z] DO
$AA_OVR[X] = 0 →
→ $AA_OVR[Z] = 100
```

Immer wenn die aktuelle Position der Pendelachse Z im MKS größer oder kleiner als die Position des Umkehrpunkts 1 ist, dann setze den axialen Override der Zustellachse X auf 0% und den axialen Override der Pendelachse Z auf 100%.

Umkehrbereich 2:

Für Umkehrpunkt 2:

```
WHENEVER
$AA_IM[Z]<>$SA_OSCILL_RES
ERVE_POS2[Z] DO
$AA_OVR[X] = 0 →
→ $AA_OVR[Z] = 100
```

Immer wenn die aktuelle Position der Pendelachse Zu im MKS größer oder kleiner als die Position des Umkehrpunkts 2 ist, dann setze den axialen Override der Zustellachse X auf 0% und den axialen Override der Pendelachse Z auf 100%.

Anhalten der Pendelbewegung im Umkehrpunkt

Die Pendelachse wird am Umkehrpunkt angehalten, gleichzeitig beginnt die Zustellbewegung. Die Pendelbewegung wird fortgesetzt, wenn die Zustellbewegung vollständig ausgeführt ist.

Gleichzeitig kann diese Synchronaktion dazu benutzt werden, die Zustellbewegung zu starten, falls diese durch eine vorhergehende Synchronaktion, die noch wirksam ist, gestoppt wurde.

Unter den gegebenen Annahmen (siehe oben) ergeben sich folgende Anweisungen:

Umkehrbereich 1:

```
WHENEVER
$SA_IM[Z]==$SA_OSCILL_RES
ERVE_POS1[Z] DO
$AA_OVR[X] = 0 →
→ $AA_OVR[Z] = 100
```

Immer wenn die aktuelle Position der Pendelachse im MKS gleich der Umkehrposition 1 ist, dann setze den axialen Ovberrie der Pendelachse auf 0% und den axialen Override der Zustellachse auf 100%.

Umkehrbereich 2:

```
WHENEVER
$SA_IM[Z]==$SA_OSCILL_RES
ERVE_POS2[Z] DO
$AA_OVR[X] = 0 →
→ $AA_OVR[Z] = 100
```

Immer wenn die aktuelle Position der Pendelachse Zu im MKS gleich der Umkehrposition 2 ist, dann setze den axialen Override der PendelachseX auf 0% und den axialen Override der Zustellachse auf 100%.

Online-Auswertung des Umkehrpunktes

Steht auf der rechten Seite des Vergleichs eine mit `$$` gekennzeichnete Hauptlaufvariable, so werden die beiden Variablen im IPO-Takt laufend ausgewertet und miteinander verglichen.

Hinweis

Mehr Informationen hierzu siehe Kapitel "Bewegungssynchronaktionen".

Pendelbewegung wieder starten

Diese Synchronaktion wird dazu benutzt, die Bewegung der Pendelachse fortzusetzen, wenn die Teilzustellbewegung abgeschlossen ist.

Unter den gegebenen Annahmen (siehe oben) ergeben sich folgende Anweisungen:

```
WHENEVER $AA_DTEPW[X]==0
DO $AA_OVR[Z] = 100
```

Immer wenn der REstweg für die Teilzustellung der Zustelalchse X im WKS gleich Null ist, dann setze den axialen Override der Pendelachse auf 100%.

Nächste Teilzustellung

Nach erfolgter Zustellung muss ein zu frühes Starten der nächsten Teilzustellung verhindert werden.

Dazu wird ein kanalspezifischer Merker (`$AC_MARKER[Index]`) verwendet, der am Ende der Teilzustellung (Teilrestweg ≡ 0) gesetzt wird und beim Verlassen des Umkehrbereichs gelöscht wird. Dann wird mit einer Synchronaktion die nächste Zustellbewegung verhindert.

Unter den gegebenen Annahmen (siehe oben) ergeben sich z. B. für Umkehrpunkt 1 folgende Anweisungen:

1. Marker setzen:

```
WHENEVER $AA_DTEPW[X]==0
DO $AC_MARKER[1] = 1
```

Immer wenn der Restweg für die Teilzustellung der Zustellachse X im WKS gleich Null ist, dann setze den Merker mit Index 1 auf 1.

2. Marker löschen

```
WHENEVER $AA_IM[Z]<>
$SA_OSCILL_RESERVE_POS1[Z
] DO $AC_MARKER[1]= 0
```

Immer wenn die aktuelle Position der Pendelachse Z ium MKS größer oder kleiner als die Position des Umkehrpunkts 1 ist, dann setze den Merker 1 auf 0.

3. Zustellung verhindern

```
WHENEVER $AC_MARKER[1]==1
DO $AA_OVR[X] = 0
```

Immer wenn der Merker 1 gleich ist, dann setze den axialen Override der Zustellachse X auf 0%.

Stanzen und Nibbeln 12

12.1 Aktivierung, Deaktivierung

12.1.1 Stanzen und Nibbeln ein oder aus (SPOF, SON, PON, SONS, PONS, PDELAYON, PDELAYOF, PUNCHACC)

Funktion

Stanzen bzw. Nibbeln aktivieren/deaktivieren

Mit PON und SON wird die Stanz- bzw. Nibbelfunktion aktiviert. SPOF beendet alle stanz- und nibbelspezifischen Funktionen. Die modal wirksamen Befehle PON und SON schließen sich gegenseitig aus, d. h. PON deaktiviert SON und umgekehrt.

Stanzen/Nibbeln mit Vorspann

Die Funktionen SONS und PONS schalten ebenfalls die Stanz- bzw. Nibbelfunktionen ein.

Im Gegensatz zu der bei SON/PON wirksamen Hubsteuerung auf Interpolationsebene erfolgt bei diesen Funktionen die signaltechnische Steuerung der Hubauslösung auf Servoebene. Hierdurch kann mit höheren Hubfrequenzen und damit höherer Stanzleistung gearbeitet werden.

Während der Signalauswertung im Vorspann sind alle Funktionen verriegelt, die zur Positionsänderung der Nibbel- oder Stanzachsen führen (z. B. Handradfahren, Änderungen von Frames über PLC, Messfunktionen).

Stanzen mit Verzögerung

PDELAYON bewirkt eine verzögerte Ausgabe des Stanzhubs. Der modal wirksame Befehl hat vorbereitende Funktion und steht damit in der Regel vor PON. Nach PDELAYOF wird normal weitergestanzt.

Hinweis

Die Verzögerungszeit wird eingestellt im Settingdatum SD42400 $SC_PUNCH_DWELLTIME.

Wegabhängige Beschleunigung

Mit PUNCHACC kann eine Beschleunigungskennlinie festgelegt werden, die je nach Lochabstand unterschiedliche Beschleunigungen definiert.

Zweites Stanz-Interface

Maschinen, die abwechselnd ein zweites Stand-Interface (zweite Stanzeinheit oder ein vergleichbares Medium) nutzen sollen, können auf ein zweites Paar der schnellen digitalen Ein- und Ausgänge der Steuerung (I/O-Paar) umgeschaltet werden. Für beide Stand-Interfaces ist die volle Stanz-/Nibbel-Funktionalität nutzbar. Die Umschaltung zwischen erstem und zweitem Stanz-Interface erfolgt über die Befehle `SPIF1` und `SPIF2`.

Hinweis

Voraussetzung:Über Maschinendaten muss ein zweites I/O-Paar für die Stanzfunktionalität definiert sein (→ siehe Angaben des Maschinenherstellers!).

Syntax

```
PON G... X... Y... Z...
SON G... X... Y... Z...
SONS G... X... Y... Z...
PONS G... X... Y... Z...
PDELAYON
PDELAYOF
PUNCHACC(<Smin>,<Amin>,<Smax>,<Amax>)
SPIF1/SPIF2
SPOF
```

Bedeutung

`PON`	Stanzen aktivieren
`SON`	Nibbeln aktivieren
`PONS`	Stanzen mit Vorspann aktivieren
`SONS`	Nibbeln mit Vorspann aktivieren
`SPOF`	Stanzen/Nibbeln deaktivieren
`PDELAYON`	Stanzen mit Verzögerung aktivieren
`PDELAYOF`	Stanzen mit Verzögerung deaktivieren
`PUNCHACC`	Wegabhängige Beschleunigung aktivieren Parameter: `<Smin>` Kleinster Lochabstand `<Amin>` Anfangsbeschleunigung `<Amin>` kann größer als `<Amax>` sein. `<Smax>` Größter Lochabstand `<Amax>` Endbeschleunigung `<Amax>` kann größer als `<Amin>` sein.
`SPIF1`	**Erstes** Stanz-Interface aktivieren Die Hubsteuerung erfolgt über das erste Paar der schnellen I/O.

`SPIF2`

Zweites Stanz-Interface aktivieren

Die Hubsteuerung erfolgt über das zweite Paar der schnellen I/O.

Hinweis:
Nach RESET oder Steuerungshochlauf ist immer das erste Stanz-Interface aktiv. Wird nur ein Stanz-Interface benutzt, so muss dieses nicht programmiert werden.

Beispiele

Beispiel 1: Nibbeln aktivieren

```
Programmcode                    Kommentar
...
N70 X50 SPOF                    ; Positionieren ohne Stanzauslösung.
N80 X100 SON                    ; Nibbeln aktivieren, Auslösung eines Hubs vor der
                                  Bewegung (X=50) und am Ende der programmierten
                                  Bewegung (X=100).
...
```

Beispiel 2: Stanzen mit Verzögerung

```
Programmcode                    Kommentar
...
N170 PDELAYON X100 SPOF         ; Positionieren ohne Stanzauslösung, Aktivierung der
                                  verzögerten Stanzauslösung.
N180 X800 PON                   ; Stanzen aktivieren. Nach Erreichen der Endposition
                                  wird Stanzhub verzögert ausgegeben.
N190 PDELAYOF X700              ; Stanzen mit Verzögerung deaktivieren, normale
                                  Stanzauslösung am Ende der programmierten Bewegung.
...
```

Beispiel 3: Stanzen mit zwei Stand-Interfaces

```
Programmcode                    Kommentar
...
N170 SPIF1 X100 PON             ; Am Ende des Satzes erfolgt eine Hubauslösung auf
                                  dem ersten schnellen Ausgang. Das Signal "Hub
                                  aktiv" wird auf dem ersten Eingang überwacht.
N180 X800 SPIF2                 ; Die zweite Hubauslösung erfolgt auf dem zweiten
                                  schnellen Ausgang. Das Signal "Hub aktiv" wird auf
                                  dem zweiten Eingang überwacht.
N190 SPIF1 X700                 ; Die Hubsteuerung für alle weiteren Hübe erfolgt mit
                                  dem ersten Interface.
...
```

Weitere Informationen

Stanzen und Nibbeln mit Vorspann (PONS/SONS)

Stanzen und Nibbeln mit Vorspann ist nicht gleichzeitig in mehreren Kanälen möglich. PONS bzw. SONS kann nur jeweils in einem Kanal aktiviert werden.

Wegabhängige Beschleunigung (PUNCHACC)

Beispiel:
```
PUNCHACC(2,50,10,100)
```

Lochabstände unter 2mm:

Es wird mit einer Beschleunigung von 50% der Maximalbeschleunigung verfahren.

Lochabstände von 2mm bis 10mm:

Die Beschleunigung wird proportional zum Abstand auf 100% gesteigert.

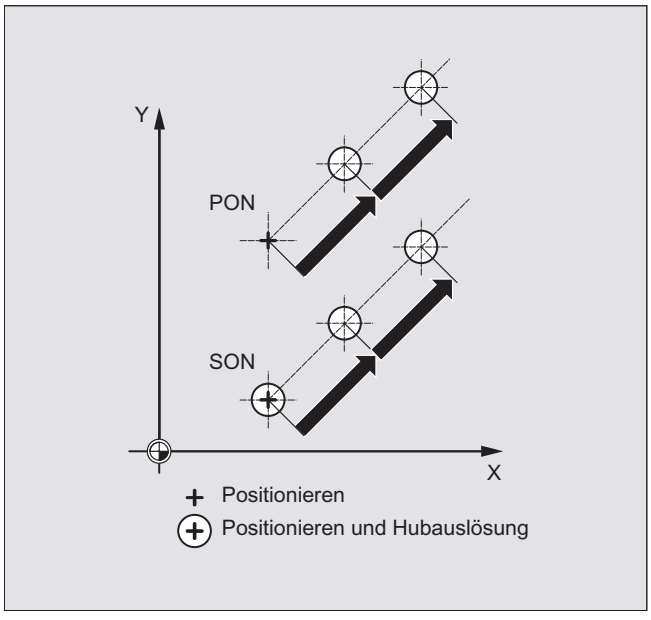
Lochabstände größer als 10mm:

Verfahren mit einer Beschleunigung von 100%.

Auslösung des ersten Hubs

Die Auslösung des ersten Hubs nach Aktivierung der Funktion erfolgt beim Nibbeln und Stanzen zeitlich unterschiedlich:

- PON/PONS:
  - Alle Hübe - auch der des ersten Satzes nach Aktivierung - erfolgen im Satzende.
- SON/SONS:
  - Der erste Hub nach Aktivierung des Nibbelns erfolgt bereits im Satzanfang.
  - Alle weiteren Hübe werden jeweils im Satzende ausgelöst.



Stanzen und Nibbeln auf der Stelle

Eine Hubauslösung erfolgt nur dann, wenn der Satz eine Verfahrinformation für die Stanz- oder Nibbelachsen (Achsen der aktiven Ebene) enthält.

Um dennoch einen Hub an gleicher Stelle auszulösen, wird eine der Stanz-/Nibbelachsen mit Verfahrweg 0 programmiert.

Arbeiten mit drehbaren Werkzeugen

Hinweis

Um drehbare Werkzeuge tangential an die programmierte Bahn anzustellen, verwenden Sie die Tangentialsteuerung.

Verwendung von M-Befehlen

Mit Hilfe der Makrotechnik ist es nach wie vor möglich, spezielle M-Funktionen statt der Sprachbefehle zu benutzen (Kompatibilität). Dabei gelten die folgenden Entsprechungen zu älteren Systemen:

```
M20, M23    ≙  SPOF
M22         ≙  SON
M25         ≙  PON
M26         ≙  PDELAYON
```

Beispiel für Makrodatei:

```
Programmcode                    Kommentar
DEFINE M25 AS PON               ; Stanzen ein
DEFINE M125 AS PONS             ; Stanzen mit Vorspann ein
DEFINE M22 AS SON               ; Nibbeln ein
DEFINE M122 AS SONS             ; Nibbeln mit Vorspann ein
DEFINE M26 AS PDELAYON          ; Stanzen mit Verzögerung ein
DEFINE M20 AS SPOF              ; Stanzen, Nibbeln aus
DEFINE M23 AS SPOF              ; Stanzen, Nibbeln aus
```

Programmierbeispiel:

```
Programmcode                    Kommentar
...
N100 X100 M20                   ; Positionieren ohne Stanzauslösung.
N110 X120 M22                   ; Nibbeln aktivieren, vor und nach Bewegung
                                  Hubauslösung.
N120 X150 Y150 M25              ; Stanzen aktivieren, Hubauslösung am Ende der
                                  Bewegung.
...
```

12.2 Automatische Wegaufbereitung

Funktion

Unterteilung in Teilstrecken

Bei aktiviertem Stanzen bzw. Nibbeln bewirken sowohl SPP als auch SPN eine Aufteilung der für die Bahnachsen programmierten Gesamtverfahrstrecke in eine Anzahl von gleichlangen Teilstrecken (äquidistante Wegaufteilung). Intern entspricht jede Teilstrecke einem Satz.

Anzahl der Hübe

Beim Stanzen erfolgt der erste Hub am Endpunkt der ersten Teilstrecke, beim Nibbeln dagegen am Startpunkt der ersten Teilstrecke. Über die Gesamtfahrstrecke ergeben sich damit folgende Zahlen:

Stanzen: Anzahl der Hübe = Anzahl der Teilstrecken

Nibbeln: Anzahl der Hübe = Anzahl der Teilstrecken + 1

Hilfsfunktionen

Hilfsfunktionen werden im ersten der erzeugten Sätze ausgeführt.

Syntax

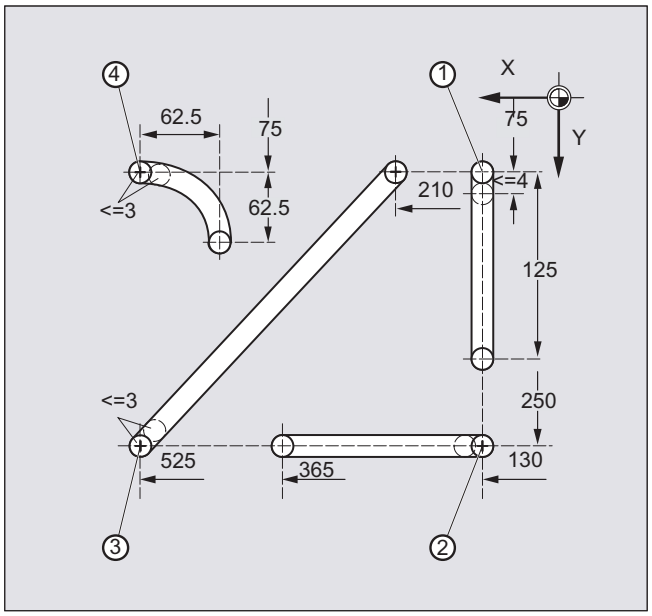
```
SPP=
SPN=
```

Bedeutung

SPP	Größe der Teilstrecke (maximaler Hubabstand); modal wirksam
SPN	Anzahl der Teilstrecken pro Satz; satzweise wirksam

Beispiel 1

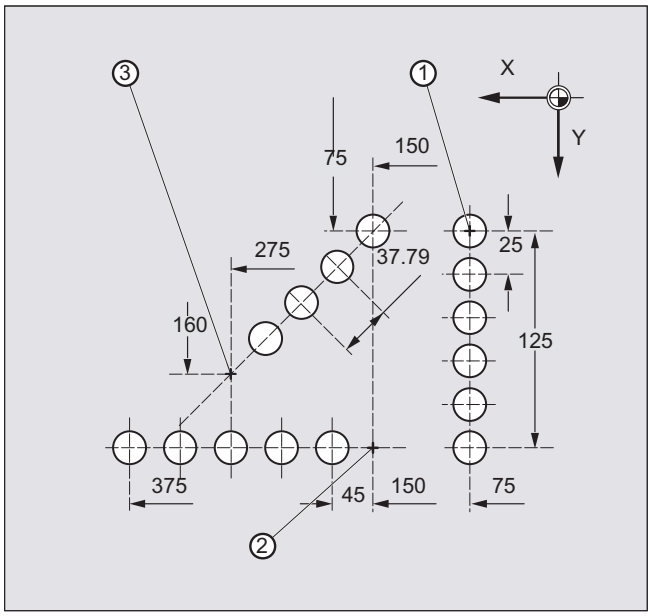
Die programmierten Nibbelstrecken sollen automatisch in gleichgroße Teilstrecken aufgeteilt werden.



Programmcode	Kommentar
N100 G90 X130 Y75 F60 SPOF	; Positionieren auf Startpunkt 1
N110 G91 Y125 SPP=4 SON	; Nibbeln ein; maximale Teilstreckenlänge für automatische Wegaufteilung: 4 mm
N120 G90 Y250 SPOF	; Nibbeln aus; Positionieren auf Startpunkt 2
N130 X365 SON	; Nibbeln ein; maximale Teilstreckenlänge für automatische Wegaufteilung: 4 mm
N140 X525 SPOF	; Nibbeln aus; Positionieren auf Startpunkt 3
N150 X210 Y75 SPP=3 SON	; Nibbeln ein; maximale Teilstreckenlänge für automatische Wegaufteilung: 3 mm
N160 X525 SPOF	; Nibbeln aus; Positionieren auf Startpunkt 4
N170 G02 X-62.5 Y62.5 I J62.5 SPP=3 SON	; Nibbeln ein; maximale Teilstreckenlänge für automatische Wegaufteilung: 3 mm
N180 G00 G90 Y300 SPOF	; Nibbeln aus

Beispiel 2

Für die einzelnen Lochreihen soll eine automatische Wegaufteilung erfolgen. Für die Aufteilung wird jeweils die maximale Teilstreckenlänge (SPP-Wert) angegeben.



Programmcode	**Kommentar**
N100 G90 X75 Y75 F60 PON	; Positionieren auf Startpunkt 1; Stanzen ein Einzelloch stanzen
N110 G91 Y125 SPP=25	; Maximale Teilstreckenlänge für automatische Wegaufteilung: 25 mm
N120 G90 X150 SPOF	; Stanzen aus; Positionieren auf Startpunkt 2
N130 X375 SPP=45 PON	; Stanzen ein; maximale Teilstreckenlänge für automatische Wegaufteilung: 45 mm
N140 X275 Y160 SPOF	; Stanzen aus; Positionieren auf Startpunkt 3
N150 X150 Y75 SPP=40 PON	; Stanzen ein; anstelle der programmierten Teilstreckenlänge von 40 mm wird die ;berechnete Teilstreckenlänge von 37,79 mm verwendet.
N160 G00 Y300 SPOF	; Stanzen aus; Positionieren

12.2.1 Wegaufteilung bei Bahnachsen

Länge der Teilstrecke SPP

Mit `SPP` geben Sie den maximalen Hubabstand und damit die maximale Länge der Teilstrecken an, in die die Gesamtverfahrstrecke aufgeteilt werden soll. Das Ausschalten des Befehls erfolgt mit `SPOF` oder `SPP=0`.

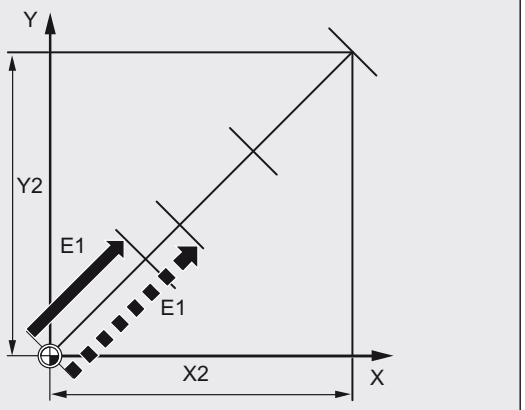
Beispiel:

```
N10 SON X0 Y0
N20 SPP=2 X10
```

Die Gesamtverfahrstrecke von 10 mm wird in 5 Teilstrecken von je 2 mm (`SPP=2`) aufgeteilt.

Hinweis

Die Wegaufteilung mit `SPP` erfolgt immer äquidistant: alle Teilstrecken sind gleich lang. Das heißt, die programmierte Teilstreckengröße (Wert von `SPP`) ist nur dann gültig, wenn der Quotient aus Gesamtverfahrstrecke und SPP-Wert ganzzahlig ist. Ist das nicht der Fall, so wird die Größe der Teilstrecke intern so reduziert, dass sich ein ganzzahliger Quotient ergibt.



X2/Y2 Programmierter Verfahrweg (Nibbel- oder Stanzsatz)
E1 Programmierte Teilstreckenlänge
E1' Automatisch abgerundete Teilstreckenlänge

Beispiel:

```
N10 G1 G91 SON X10 Y10
N20 SPP=3.5 X15 Y15
```

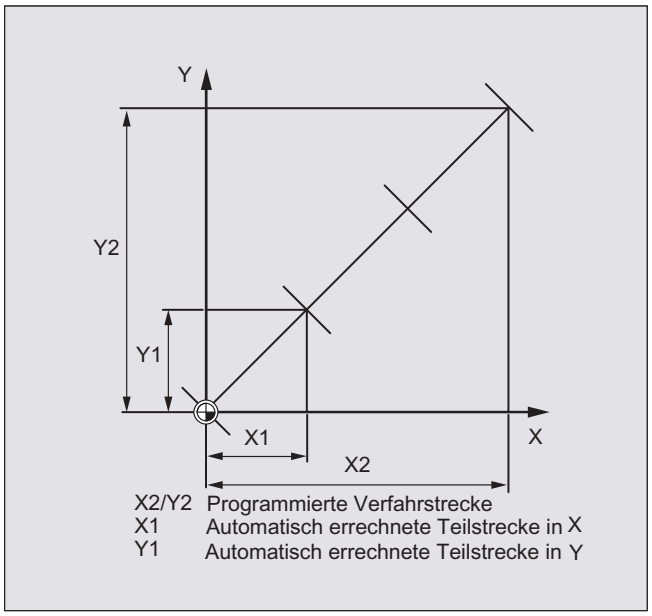
Bei der Gesamtverfahrstrecke von 15 mm und einer Teilstreckenlänge von 3,5 mm ergibt sich ein nicht ganzzahliger Quotient (4.28). Somit erfolgt eine Reduktion des SPP-Werts bis zum nächstmöglichen ganzzahligen Quotienten. In diesem Fall ergibt sich eine Teilstreckenlänge von 3 mm.

Anzahl der Teilstrecken SPN

Mit SPN definieren Sie die Anzahl der Teilstrecken, die aus der Gesamtverfahrstrecke erzeugt werden soll. Die Länge der Teilstrecken wird automatisch berechnet. Da SPN satzweise wirksam ist, muss vorher Stanzen oder Nibbeln mit PON oder SON aktiviert werden.

SPP und SPN im gleichen Satz

Programmieren Sie in einem Satz sowohl die Teilstreckenlänge (SPP) als auch Anzahl der Teilstrecken (SPN), so gilt für diesen Satz SPN, für alle weiteren SPP. Wurde SPP schon vor SPN aktiviert, so wird es nach dem Satz mit SPN wieder wirksam.



Hinweis

Sofern Stanzen/Nibbeln grundsätzlich in der Steuerung verfügbar ist, ist die Programmierung der automatischen Wegaufteilung mit SPN bzw. SPP auch unabhängig von dieser Technologie aktivierbar.

12.2.2 Wegaufteilung bei Einzelachsen

Sind neben den Bahnachsen auch Einzelachsen als Stanz-Nibbel-Achse definiert, so können auch sie der automatischen Wegaufteilung unterliegen.

Verhalten der Einzelachse bei SPP

Die programmierte Länge der Teilstrecke (`SPP`) bezieht sich grundsätzlich auf die Bahnachsen. Daher wird in einem Satz, in dem neben der Einzelachsbewegung und dem SPP-Wert keine Bahnachse programmiert ist, der SPP-Wert ignoriert.

Sind sowohl Einzel- als auch Bahnachse im Satz programmiert, so richtet sich das Verhalten der Einzelachse nach der Einstellung des entsprechenden Maschinendatums.

1. Standardeinstellung

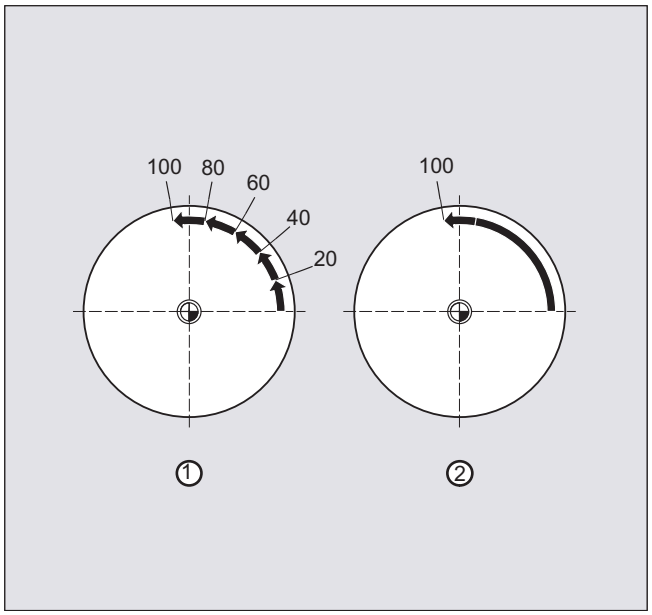
   Der Weg der Einzelachse wird gleichmäßig auf die durch `SPP` erzeugten Zwischensätze verteilt.

Beispiel:

```
N10 G1 SON X10 A0
N20 SPP=3 X25 A100
```

Durch die Hubstrecke von 3 mm werden bei der Gesamtverfahrstrecke der X-Achse (Bahnachse) von 15 mm 5 Sätze erzeugt.

Die A-Achse dreht sich damit in jedem Satz um 20°.



1. Einzelachse ohne Wegaufteilung

   Die Einzelachse verfährt ihren Gesamtweg im ersten der erzeugten Sätze.

2. Unterschiedliche Wegaufteilung

   Das Verhalten der Einzelachse ist abhängig von der Interpolation der Bahnachsen:

- Kreisinterpolation: Wegaufteilung
- Linearinterpolation: keine Wegaufteilung

Verhalten bei SPN

Die programmierte Anzahl von Teilstrecken gilt auch, wenn nicht gleichzeitig eine Bahnachse programmiert ist.

Voraussetzung: Einzelachse ist als Stanz-Nibbel-Achse definiert.

Schleifen

13

13.1 Schleifenspezifische Werkzeugüberwachung im Teileprogramm (TMON, TMOF)

Funktion

Mit dem Befehl TMON können Sie für Schleifwerkzeuge (Typ 400 - 499) die Geometrie- und Drehzahlüberwachung im NC-Teileprogramm aktivieren. Die Überwachung bleibt aktiv, bis sie im Teileprogramm durch den Befehl TMOF abgeschaltet wird.

Hinweis

Bitte beachten Sie die Angaben des Maschinenherstellers!

Voraussetzung

Die schleifspezifischen Werkzeug-Parameter $TC_TPG1 bis $TC_TPG9 müssen gesetzt sein.

Syntax

```
TMON(<T-Nr.>)
TMOF(<T-Nr.>)
```

Bedeutung

TMON	Befehl zum **Ein**schalten der schleifspezifischen Werkzeugüberwachung
TMOF	Befehl zum **Aus**schalten der schleifspezifischen Werkzeugüberwachung
<T-Nr.>	Angabe der T-Nummer **Hinweis:** Nur notwendig, wenn das Werkzeug mit dieser T-Nummer nicht aktiv ist.
TMOF(0)	Überwachung für alle Werkzeuge ausschalten

Weitere Informationen

Schleifspezifische Werkzeug-Parameter

Parameter	Bedeutung	Datentyp
$TC_TPG1	Spindelnummer	INT
$TC_TPG2	Verkettungsvorschrift Die Parameter werden automatisch für die linke und rechte Scheibenseite identisch gehalten.	INT
$TC_TPG3	Minimaler Scheibenradius	REAL
$TC_TPG4	Minimale Scheibenbreite	REAL
$TC_TPG5	Aktuelle Scheibenbreite	REAL
$TC_TPG6	Maximale Drehzahl	REAL
$TC_TPG7	Maximale Umfangsgeschwindigkeit	REAL
$TC_TPG8	Winkel der schrägen Scheibe	REAL
$TC_TPG9	Parameter-Nummer für Radiusberechnung	INT

Literatur:
Funktionshandbuch Grundfunktionen; Werkzeugkorrektur (W1)

Werkzeugüberwachung einschalten über Werkzeuganwahl

In Abhängigkeit von einem Maschinendatum kann für die Schleifwerkzeuge (Typ 400 - 499) die Werkzeugüberwachung implizit mit der Werkzeuganwahl eingeschaltet werden.

Zu jedem Zeitpunkt kann für jede Spindel nur **eine** Überwachung aktiv sein.

Geometrieüberwachung

Überwacht werden der aktuelle Scheibenradius und die aktuelle Breite.

Die Überwachung des Drehzahlsollwerts auf den Drehzahlgrenzwert erfolgt zyklisch unter Berücksichtigung des Spindel-Overrides.

Als Drehzahlgrenzwert gilt der kleinere Wert, der sich bei Vergleich von maximaler Drehzahl mit der berechneten Drehzahl aus maximaler Scheibenumfangsgeschwindigkeit und aktuellem Scheibenradius ergibt.

Arbeiten ohne T- und D-Nummer

Per Maschinendatum kann eine Standard-**T**-Nummer und Standard-**D**-Nummer eingestellt werden,

die nicht mehr programmiert werden muss und nach Power On / Reset wirksam wird.

Beispiel: Arbeiten mit derselben Schleifscheibe

Über das Maschinendatum kann eingestellt werden, dass das aktive Werkzeug bei Reset erhalten bleibt (siehe " Freie D-Nummernvergabe, Schneidennummer (Seite 429) ").

Weitere Funktionen 14

14.1 Achsfunktionen (AXNAME, AX, SPI, AXTOSPI, ISAXIS, AXSTRING, MODAXVAL)

Funktion

AXNAME wird z. B. bei der Erstellung allgemeingültiger Zyklen verwendet, wenn die Namen der Achsen nicht bekannt sind.

AX wird für die indirekte Programmierung von Geometrie- und Synchronachsen verwendet. Der Achsbezeichner wird dabei in einer Variablen vom Typ AXIS hinterlegt oder von einem Befehl wie AXNAME oder SPI geliefert.

SPI wird verwendet, wenn Achsfunktionen für eine Spindel, z. B. Synchronspindel, programmiert werden.

AXTOSPI wird verwendet, um einen Achsbezeichner in einen Spindelindex zu wandeln (Umkehrfunktion zu SPI).

AXSTRING wird verwendet, um einen Achsbezeichner (Datentyp AXIS) in einen String zu wandeln (Umkehrfunktion zu AXNAME).

ISAXIS wird in allgemeingültigen Zyklen verwendet, um sicherzustellen, dass eine bestimmte Geometrieachse vorhanden ist und damit ein nachfolgender Aufruf von $P_AXNX nicht mit Fehler abgebrochen wird.

MODAXVAL wird verwendet, um bei Modulo-Rundachsen die Modulo-Position zu ermitteln.

Syntax

```
AXNAME("String")
AX[AXNAME("String")]
SPI(n)
```

AXTOSPI(A) oder AXTOSPI(B) oder AXTOSPI(C)

```
AXSTRING(SPI(n))
ISAXIS(<Geometrieachsnummer>)
<Modulo-Position>=MODAXVAL(<Achse>,<Achsposition>)
```

Bedeutung

AXNAME	Konvertiert einen Eingangsstring in Achsbezeichner; der Eingangsstring muss gültigen Achsnamen enthalten.
AX	Variabler Achsbezeichner
SPI	Konvertiert Spindelnummer in Achsbezeichner; der Übergabeparameter muss eine gültige Spindelnummer enthalten.
n	Spindelnummer
AXTOSPI	Wandelt einen Achsbezeichner in einen Spindelindex vom Typ Integer um. AXTOSPI entspricht der Umkehrfunktion zu SPI.
X, Y, Z	Achsbezeichner vom Typ AXIS als Variable oder Konstante
AXSTRING	Es wird der String mit zugeordneter Spindelnummer ausgegeben.
ISAXIS	Prüft, ob die angegebene Geometrieachse vorhanden ist.
MODAXVAL	Ermittelt bei Modulo-Rundachsen die Modulo-Position; diese entspricht dem Modulo-Rest bezogen auf den parametrierten Modulo-Bereich (beträgt in der Standardeinstellung 0 bis 360 Grad; über MD30340 MODULO_RANGE_START und MD30330 $MA_MODULO_RANGE können Beginn und Größe des Modulo-Bereichs verändert werden).

Hinweis

SPI-Erweiterungen

Die Achsfunktion SPI(n) ist auch für das Lesen und Schreiben von Framekomponenten einsetzbar. Damit können Frames z. B. mit der Syntax `$P_PFRAME[SPI(1),TR]=2.22` geschrieben werden.

Durch die zusätzliche Programmierung von Achspositionen über die Adresse `AX[SPI(1)]=<Achsposition>` kann eine Achse verfahren werden. Voraussetzung dafür ist, dass sich die Spindel im Positionier- oder Achsbetrieb befindet.

Beispiele

Beispiel 1: AXNAME, AX, ISAXIS

```
Programmcode                                    Kommentar
OVRA[AXNAME("Planachse")]=10                    ; Override für Planachse
AX[AXNAME("Planachse")]=50.2                    ; Endposition für Planachse
OVRA[SPI(1)]=70                                 ; Override für Spindel 1
AX[SPI(1)]=180                                  ; Endposition für Spindel 1
IF ISAXIS(1)==FALSE GOTOF WEITER                ; Abszisse vorhanden?
AX[$P_AXN1]=100                                 ; Abszisse verfahren
WEITER:
```

Beispiel 2: AXSTRING

Bei der Programmierung mit AXSTRING[SPI(n)] wird nicht mehr der Achsindex der Achse, der die Spindel zugeordnet ist, als Spindelnummer ausgegeben, sondern es wird der String `"Sn"` ausgegeben.

Programmcode	Kommentar
`AXSTRING[SPI(2)]`	`; Es wird der String "S2" ausgegeben.`

Beispiel 3: MODAXVAL

Die Modulo-Position der Modulo-Rundachse A soll ermittelt werden.

Ausgangswert für die Berechnung ist die Achsposition 372.55.

Der parametrierte Modulo-Bereich beträgt 0 bis 360 Grad:

MD30340 MODULO_RANGE_START = 0

MD30330 $MA_MODULO_RANGE = 360

Programmcode	Kommentar
`R10=MODAXVAL(A,372.55)`	`; Berechnete Modulo-Position R10 = 12.55.`

Beispiel 4: MODAXVAL

Wenn sich der programmierte Achsbezeichner nicht auf eine Modulo-Rundachse bezieht, dann wird der zu wandelnde Wert (<Achsposition>) unverändert zurückgegeben.

Programmcode	Kommentar
`R11=MODAXVAL(X,372.55)`	`; X ist Linearachse; R11 = 372.55.`

14.2 Umschaltbare Geometrieachsen (GEOAX)

Funktion

Mit der Funktion "Umschaltbare Geometrieachsen" lässt sich der über Maschinendaten konfigurierte Geometrieachsverbund vom Teileprogramm aus verändern. Dabei kann eine als synchrone Zusatzachse definierte Kanalachse eine beliebige Geometrieachse ersetzen.

Syntax

```
GEOAX(<n>,<Kanalachse>,<n>,<Kanalachse>,<n>,<Kanalachse>)
GEOAX()
```

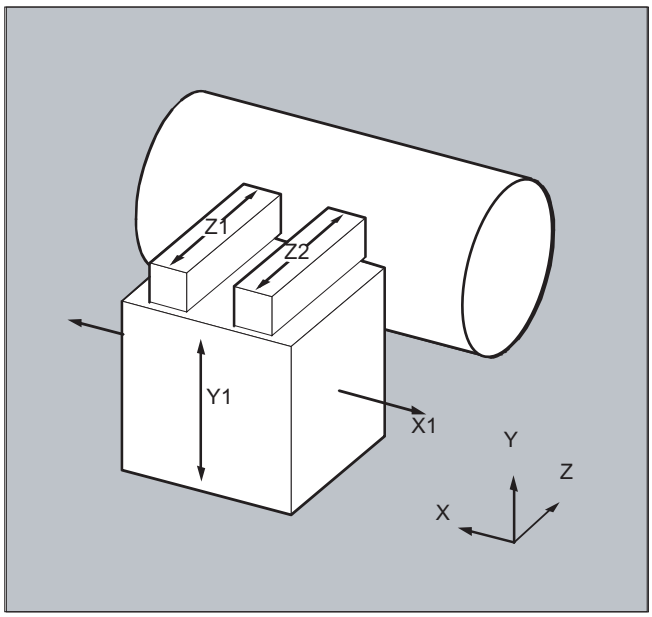
Bedeutung

`GEOAX(...)`	Befehl zum Umschalten der Geometrieachsen **Hinweis:** `GEOAX()` ohne Parameterangabe ruft die Grundkonfiguration der Geometrieachsen auf.
`<n>`	Mit diesem Parameter wird die Nummer der Geometrieachse angegeben, der die nachfolgend angegebene Kanalachse zugeordnet werden soll. Wertebereich: 1, 2 oder 3 **Hinweis:** Mit **<n>=0** kann die nachfolgend angegebene Kanalachse aus dem Geometrieachsverbund ersatzlos entfernt werden.
`<Kanalachse>`	Mit diesem Parameter wird der Name der Kanalachse angegeben, die in den Geometrieachsverbund aufgenommen werden soll.

Beispiele

Beispiel 1: Zwei Achsen wechselweise als Geometrieachse schalten

Ein Werkzeugschlitten kann über die Kanalachsen X1, Y1, Z1, Z2 verfahren werden:



Die Geometrieachsen sind so projektiert, dass nach dem Einschalten zunächst Z1 als 3. Geometrieachse unter dem Geometrieachsnamen "Z" wirksam ist und zusammen mit X1 und Y1 den Geometrieachsverbund bildet.

Im Teileprogramm sollen nun die Achsen Z1 und Z2 wechselweise als Geometrieachse Z zum Einsatz kommen:

```
Programmcode                    Kommentar
...
N100 GEOAX(3,Z2)                ; Als 3. Geometrieachse (Z) fungiert Kanalachse Z2.
N110 G1 ...
N120 GEOAX(3,Z1)                ; Als 3. Geometrieachse (Z) fungiert Kanalachse Z1.
...
```

Beispiel 2: Umschalten der Geometrieachsen bei 6 Kanalachsen

Eine Maschine besitzt 6 Kanalachsen mit den Namen XX, YY, ZZ, U, V, W.

Die Grundeinstellung der Geometrieachskonfiguration über Maschinendaten ist:

Kanalachse XX = 1. Geometrieachse (X-Achse)

Kanalachse YY = 2. Geometrieachse (Y-Achse)

Kanalachse ZZ = 3. Geometrieachse (Z-Achse)

Programmcode	**Kommentar**
N10 GEOAX()	; Grundkonfiguration der Geometrieachsen ist wirksam.
N20 G0 X0 Y0 Z0 U0 V0 W0	; Alle Achsen im Eilgang auf Position 0.
N30 GEOAX(1,U,2,V,3,W)	; Kanalachse U wird zur ersten (X), V zur zweiten (Y) und W zur dritten Geometrieachse (Z).
N40 GEOAX(1,XX,3,ZZ)	; Kanalachse XX wird zur ersten (X), ZZ zur dritten Geometrieachse (Z). Kanalachse V bleibt zweite Geometrieachse (Y).
N50 G17 G2 X20 I10 F1000	; Vollkreis in der X/Y-Ebene. Es fahren die Kanalachsen XX und V.
N60 GEOAX(2,W)	; Kanalachse W wird zweite Geometrieachse (Y).
N80 G17 G2 X20 I10 F1000	; Vollkreis in der X/Y-Ebene. Es fahren die Kanalachsen XX und W.
N90 GEOAX()	; Zurücksetzen auf Grundzustand.
N100 GEOAX(1,U,2,V,3,W)	; Kanalachse U wird zur ersten (X), V zur zweiten (Y) und W zur dritten Geometrieachse (Z).
N110 G1 X10 Y10 Z10 XX=25	; Kanalachsen U, V, W fahren jeweils auf die Position 10. XX als Zusatzachse fährt auf Position 25.
N120 GEOAX(0,V)	; V wird aus Geometrieachsverbund herausgenommen. U und W sind weiterhin erste (X) und dritte Geometrieachse (Z). Die zweite Geometrieachse (Y) bleibt unbelegt.
N130 GEOAX(1,U,2,V,3,W)	; Kanalachse U bleibt erste (X), V wird zur zweiten (Y), W bleibt dritte Geometrieachse (Z).
N140 GEOAX(3,V)	; V wird zur dritten Geometrieachse (Z), wobei W überschrieben und damit aus dem Geometrieachsverbund herausgenommen wird. Die zweite Geometrieachse (Y) ist nach wie vor unbelegt.

Hinweis

Achskonfiguration

Die Zuordnung zwischen den Geometrieachsen, Zusatzachsen, Kanalachsen und Maschinenachsen, sowie die Festlegung der Namen der einzelnen Achstypen wird über folgende Maschinendaten getroffen:

MD20050 $MC_AXCONF_GEOAX_ASIGN_TAB (Zuordnung Geometrieachse zu Kanalachse)

MD20060 $MC_AXCONF_GEOAX_NAME_TAB (Geometrieachsname im Kanal)

MD20070 $MC_AXCONF_MACHAX_USED (Maschinenachsnummer gültig im Kanal)

MD20080 $MC_AXCONF_CHANAX_NAME_TAB (Kanalachsname im Kanal)

MD10000 $MN_AXCONF_MACHAX_NAME_TAB (Maschinenachsname)

MD35000 $MA_SPIND_ASSIGN_TO_MACHAX (Zuordnung Spindel zu Maschinenachse)

Literatur:
Funktionshandbuch Grundfunktionen; Achsen, Koordinatensysteme Frames (K2)

Einschränkungen

- Die Umschaltung der Geometrieachsen ist nicht möglich bei:
  - aktiver Transformation
  - aktiver Spline-Interpolation
  - aktiver Werkzeugradiuskorrektur
  - aktiver Werkzeugfeinkorrektur
- Weisen Geometrieachse und Kanalachse gleiche Namen auf, ist kein Wechsel der jeweiligen Geometrieachse möglich.
- Keine der an der Umschaltung beteiligten Achsen darf an einer Aktion beteiligt sein, die über die Satzgrenzen hinweg andauern kann, wie es z. B. bei Positionierachsen vom Typ A oder bei Folgeachsen möglich ist.
- Mit dem Befehl `GEOAX` können nur bereits beim Einschalten vorhandene Geometrieachsen ersetzt werden (also keine neuen definiert werden).
- Ein Achstausch mit `GEOAX` während der Aufbereitung der Konturtabelle (`CONTPRON`, `CONTDCON`) führt zum Alarm.

Randbedingungen

Achszustand nach dem Ersetzen

Eine durch die Umschaltung im Geometrieachsverbund ersetzte Achse ist nach dem Umschaltvorgang über ihren Kanalachsnamen als Zusatzachse programmierbar.

Frames, Schutzbereiche, Arbeitsfeldbegrenzungen

Mit dem Umschalten der Geometrieachsen werden alle Frames, Schutzbereiche und Arbeitsfeldbegrenzungen gelöscht.

Polarkoordinaten

Ein Tausch der Geometrieachsen mit `GEOAX` setzt analog einem Ebenenwechsel mit `G17-G19` die modalen Polarkoordinaten auf den Wert 0.

DRF, NPV

Eine eventuelle Handrad-Verschiebung (DRF) oder eine externe Nullpunktverschiebung (NPV) bleibt nach der Umschaltung wirksam.

Grundkonfiguration der Geometrieachsen

Der Befehl `GEOAX()` ruft die Grundkonfiguration des Geometrieachsverbunds auf.

Nach POWER ON und bei Umschalten in die Betriebsart "Referenzpunktfahren" wird automatisch auf die Grundkonfiguration zurückgeschaltet.

Werkzeuglängenkorrektur

Eine aktive Werkzeuglängenkorrektur ist auch nach dem Umschaltvorgang wirksam. Sie gilt jedoch für die neu aufgenommenen bzw. positionsgetauschten Geometrieachsen als noch nicht herausgefahren. Beim ersten Bewegungsbefehl für diese Geometrieachsen besteht der resultierende Verfahrweg dementsprechend aus der Summe von Werkzeuglängenkorrektur und programmiertem Verfahrweg.

Geometrieachsen, die bei einer Umschaltung ihre Position im Achsverband beibehalten, behalten auch ihren Status bezüglich der Werkzeuglängenkorrektur.

Geometrieachskonfiguration bei aktiver Transformation

Die in einer aktiven Transformation geltende Geometrieachskonfiguration (festgelegt über Maschinendaten) ist über die Funktion "Umschaltbare Geometrieachsen" nicht veränderbar.

Besteht die Notwendigkeit, im Zusammenhang mit Transformationen die Geometrieachskonfiguration zu ändern, so ist dies nur über eine weitere Transformation möglich.

Eine über `GEOAX` veränderte Geometrieachskonfiguration wird durch Aktivierung einer Transformation gelöscht.

Widersprechen sich Einstellungen der Maschinendaten für die Transformation und für die Umschaltung von Geometrieachsen, so haben die Einstellungen in der Transformation Vorrang.

Beispiel:

Eine Transformation sei aktiv. Laut Maschinendaten soll die Transformation bei einem RESET erhalten bleiben, gleichzeitig soll bei einem RESET jedoch die Grundkonfiguration der Geometrieachsen hergestellt werden. In diesem Fall bleibt die Geometrieachskonfiguration erhalten, die mit der Transformation festgelegt wurde.

14.3 Achscontainer (AXCTSWE, AXCTSWED)

Funktion

Bei Rundtaktmaschinen/Mehrspindelmaschinen bewegen sich die Werkstück-tragenden Achsen von einer Bearbeitungseinheit zur nächsten. Weil die Bearbeitungseinheiten verschiedenen NCU-Kanälen unterstehen, müssen bei einem Stations- / Lagewechsel die Werkstück-tragenden Achsen dem entsprechenden NCU-Kanal dynamisch neu zugeordnet werden. Diesem Zweck dienen Achscontainer.

Zu einem Zeitpunkt ist immer nur eine Werkstück-Aufspann-Achse/-Spindel an der lokalen Bearbeitungseinheit aktiv. Der Achscontainer stellt die Verbindungsmöglichkeiten zu allen Aufspann-Achsen/-Spindeln zusammen, von denen immer nur genau eine für die Bearbeitungseinheitaktiviert ist.

Der Wechsel der über einen Achscontainer definiert benutzbaren Achsen erfolgt durch Verschiebung der Einträge im Achscontainer ("Achscontainer-Drehung") um eine über Settingdatum vorgebbare Schrittweite (Anzahl Slots).

Der Aufruf zur Achscontainer-Drehung aus dem Teileprogramm erfolgt mit dem Befehl `AXCTSWE` bzw. `AXCTSWED`.

Syntax

```
AXCTSWE(<Achscontainer>)
AXCTSWED(<Achscontainer>)
```

Bedeutung

`AXCTSWE`	Befehl zum Drehen eines Achscontainers Wenn in der Steuerung die Freigaben aller Kanäle für die Achsen des Containers eingetroffen sind, erfolgt die Containerdrehung mit der im SD41700 $SN_AXCT_SWWIDTH[<Containernummer>] hinterlegten Container-spezifischen Schrittweite.
`AXCTSWED`	Befehl zum Drehen eines Achscontainers unter alleiniger Wirkung des aktiven Kanals (Befehlsvariante für die Inbetriebnahme!) **Hinweis:** Die im Container eingetragenen Achsen werden nur freigegeben, wenn sich die übrigen Kanäle, die Achsen im Container haben, im Reset-Zustand befinden.

`<Achscontainer>`	Bezeichner des Achscontainers, der weitergeschaltet werden soll. Mögliche Angaben sind:	
	`CT<Containernummer>`	An die Buchstabenkombination CT wird die Nummer des Achscontainers angehängt. Beispiel: `CT3`
	`<Containername>`	Mit MD12750 $MN_AXCT_NAME_TAB eingestellter individueller Name des Achscontainers. Beispiel: `A_CONT3`

Weitere Informationen

Achscontainer

Über Achscontainer können zugeordnet werden.

- lokale Achsen und/oder
- Link-Achsen

Achscontainer mit Link-Achsen sind ein NCU-übergreifendes Betriebsmittel (NCU global), das durch die Steuerung koordiniert wird. Achscontainer, in denen ausschließlich lokale Achsen verwaltet werden, sind möglich.

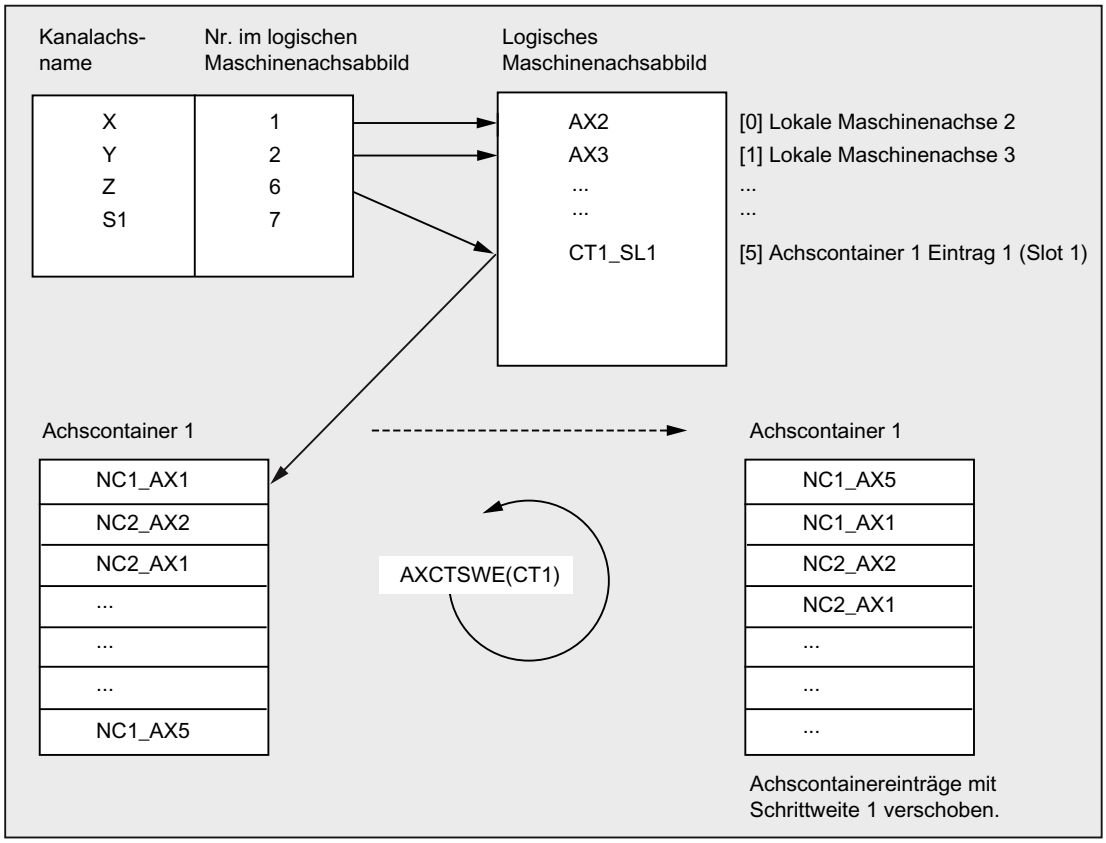
Literatur:
Detaillierte Hinweise zur Projektierung von Achscontainern siehe:
Funktionshandbuch Erweiterungsfunktionen; Mehrere Bedientafeln an mehreren NCUs, Dezentrale Systeme (B3)

Freigabekriterien

AXCTSWE( )

Jeder Kanal, dessen Achsen im angegebenen Container eingetragen sind, gibt die Freigabe für eine Containerdrehung(enable), wenn er mit der Bearbeitung der Lage/Station fertig ist. Wenn in der Steuerung die Freigaben **aller** Kanäle für die Achsen des Containers eingetroffen sind, erfolgt die Containerdrehung mit der im SD41700 $SN_AXCT_SWWIDTH[<Containernummer>] hinterlegten Container-spezifischen Schrittweite.

Beispiel:



Nach der Achscontainer-Drehung um 1 ist der Kanalachse Z statt der Achse AX1 auf NCU1 die Achse AX5 auf NCU1 zugeordnet.

AXCTSWED( )

Die Befehlsvariante `AXCTSWED()` kann zur Vereinfachung der Inbetriebnahme eingesetzt werden. Der Achscontainer dreht sich unter alleiniger Wirkung des aktiven Kanals um die im SD41700 $SN_AXCT_SWWIDTH[<Containernummer>] hinterlegten Container-spezifischen Schrittweite. Der Aufruf darf nur verwendet werden, wenn sich die übrigen Kanäle, die Achsen im Container haben, im **Reset**-Zustand befinden.

Wirksamkeit

Von der neuen Achszuordnung nach einer Achscontainerdrehung sind alle NCUs betroffen, deren Kanäle über das logische Maschinenachsabbild auf den gedrehten Achscontainer verweisen.

Achscontainer-Drehung mit impliziten GET/GETD

Bei der Freigabe einer Achscontainer-Drehung werden alle dem Kanal zugeordneten Achscontainer-Achsen mittels GET bzw. GETD dem Kanal zugeordnet. Eine Abgabe der Achsen ist erst nach der Achscontainer-Drehung wieder erlaubt.

Hinweis

Dieses Verhalten kann über Maschinendatum eingestellt werden. Bitte beachten Sie die Angaben des Maschinenherstellers.

Hinweis

Die Achscontainer-Drehung mit impliziten GET / GETD kann für eine Achse im Zustand Hauptlauf-Achse (z. B. für eine PLC-Achse) **nicht** angewendet werden, da diese Achse dann diesen Zustand zur Achscontainer-Drehung verlassen müsste.

14.4 Warten auf gültige Achsposition (WAITENC)

Funktion

Mit dem Sprachbefehl WAITENC kann im NC-Programm gewartet werden, bis für die mit MD34800 $MA_WAIT_ENC_VALID = 1 projektierten Achsen synchronisierte bzw. restaurierte Achspositionen zur Verfügung stehen.

Im Wartezustand kann eine Unterbrechung erfolgen, z. B. durch Start eines ASUPs oder durch Betriebsartenwechsel nach JOG. Mit der Programmfortsetzung wird der Wartezustand ggf. wieder eingenommen.

Hinweis

Der Wartezustand wird in der Bedienoberfläche durch den Haltezustand "Warten auf Mess-System" angezeigt.

Syntax

WAITENC kann im Programmteil eines beliebigen NC-Programms programmiert werden.

Die Programmierung muss in einem eigenen Satz erfolgen:

```
...
WAITENC
...
```

Beispiel

WAITENC wird z. B. im ereignisgesteuerten Anwenderprogramm .../_N_CMA_DIR/_N_PROG_EVENT_SPF verwendet, wie das folgende Anwendungsbeispiel zeigt.

Anwendungsbeispiel:Werkzeugrückzug nach POWER OFF mit Orientierungstransformation

Eine Bearbeitung mit Werkzeugorientierung wurde durch Spannungsausfall abgebrochen. Beim anschließenden Hochlauf wird das ereignisgesteuerte Anwenderprogramm .../_N_CMA_DIR/_N_PROG_EVENT_SPF aufgerufen.

Im ereignisgesteuerten Anwenderprogramm wird mit WAITENC auf synchronisierte bzw. restaurierte Achspositionen gewartet, um danach einen Frame berechnen zu können, der das WKS in Werkzeugrichtung ausrichtet.

```
Programmcode                          Kommentar
...
IF $P_PROG_EVENT == 4                 ; Hochlauf.
    IF $P_TRAFO <> 0                  ; Transformation wurde angewählt.
        WAITENC                       ; Warten auf gültige Achspositionen der Orientierungsachsen.
        TOROTZ                        ; Z-Achse des WKS in Richtung der Werkzeugachse drehen.
    ENDIF
    M17
ENDIF
...
```

Danach kann das Werkzeug in der Betriebsart JOG durch eine Rückzugsbewegung in Richtung der Werkzeugachse freigefahren werden.

14.5 Vorhandenen NC-Sprachumfang prüfen (STRINGIS)

Funktion

Mit der Funktion `STRINGIS(...)` kann geprüft werden, ob der angegebene String als Element der NC-Programmiersprache im aktuellen Sprachumfang zur Verfügung steht.

Definition

```
INT STRINGIS(STRING <Name>)
```

Syntax

```
STRINGIS(<Name>)
```

Bedeutung

`STRINGIS`:	Funktion mit Rückgabewert
`<Name>`:	Name des zu prüfenden Elementes der NC-Programmiersprache
Rückgabewert:	Das Format des Rückgabewertes ist yxx (Dezimal).

Elemente der NC-Programmiersprache

Folgende Elemente der NC-Programmiersprache können geprüft werden:

- G-Codes aller existierenden G-Funktionsgruppen z.B. `G0`, `INVCW`, `POLY`, `ROT`, `KONT`, `SOFT`, `CUT2D`, `CDON`, `RMB`, `SPATH`
- DIN- oder NC-Adressen wie z.B. `ADIS`, `RNDM`, `SPN`, `SR`, `MEAS`
- Funktionen z.B. `TANG(...)` oder `GETMDACT`
- Prozeduren z.B. `SBLOF`.
- Schlüsselworte z.B. `ACN`, `DEFINE` oder `SETMS`
- Systemdaten z.B. Maschinendaten `$M`... , Settingdaten `$S`... oder Optionsdaten `$O`...
- Systemvariable `$A`... , `$V`... , `$P`...
- Rechenparameter `R`...
- Zyklennamen von aktivierten Zyklen
- GUD- und LUD-Variablen
- Makro-Namen
- Label-Namen

Rückgabewert

Der Rückgabewert ist nur in den erst 3 Dezimalstellen relevant. Das Format des Rückgabewertes ist yxx, mit y = Basisinformation und xx = Detailinformation.

Rückgabewert	Bedeutung	
000	Der String 'name' ist im vorliegenden System nicht bekannt [1)]	
100	Der String 'name' ist ein Element der NC-Programmiersprache, aber aktuell nicht programmierbar (Option/Funktion ist inaktiv)	
2xx	Der String 'name' ist ein programmierbares Element der NC-Programmiersprache (Option/Funktion ist aktiv). Die Detailinformation xx enthält weitere Informationen über die Art des Elements:	
	xx	Bedeutung
	01	DIN–Adresse oder NC–Adresse [2)]
	02	G–Code (z.B. `G04`, `INVCW`)
	03	Funktion mit Rückgabewert
	04	Funktion ohne Rückgabewert
	05	Schlüsselwort (z.B. `DEFINE`)
	06	Maschinen- (`$M`...), Setting- (`$S`...) oder Optionsdatum (`$O`...)
	07	Systemparameter, z.B. Systemvariable (`$`...) oder Rechenparameter (`R`...)
	08	Zyklus (Der Zyklus muss im NCK geladen und die Zyklenprogramme aktiv sein [3)])
	09	GUD–Variable (Die GUD-Variable muss der in GUD–Definitionsdateien definierten und die GUD-Variablen aktiviert sein)
	10	Makroname (Das Makro muss in der Makro–Definitionsdateien definierten und Makros aktivierten sein) [4)]
	11	LUD–Variable des aktuellen Teileprogramms
	12	ISO G–Code (ISO Sprachmodus muss aktiv ist)
400	Der String 'name' ist eine NC–Adresse, die nicht als xx == 01 oder xx == 10 erkannt wurde und die nicht G oder R ist [2)]	
y00	Keine spezifische Zuordnung möglich	

1) Steuerungs-abhängig ist unter Umständen nur eine Untermenge der Siemens NC-Sprachbefehle bekannt, z.B. SINUMERIK 802D sl. Auf diesen Steuerungen wird für Strings, die prinzipiell Siemens NC-Sprachbefehle sind, der Wert 0 zurückgegeben. Dieses Verhalten kann über MD10711 $MN_NC_LANGUAGE_CONFIGURATION verändert werden. Bei MD10711 = 1 wird dann für Siemens NC-Sprachbefehle immer der Wert 100 zurückgegeben.

2) NC–Adressen sind folgende Buchstaben: A, B, C, E, I, J, K, Q, U, V, W, X, Y, Z. Diese NC–Adressen können auch mit einer Adresserweiterung programmiert werden. Die Adresserweiterung kann bei der Prüfung mit STRINGIS angegeben werden. Beispiel: `201 == STRINGIS("A1")`.
Die Buchstaben: D, F, H, L, M, N, O, P, S, T sind NC–Adressen oder Hilfsfunktionen die anwenderdefiniert verwendet werden. Für sie wird immer der Wert 400 zurückgegeben. Beispiel: 400 == STRINGIS("D"). Diese NC–Adressen können bei der Prüfung mit STRINGIS nicht mit Adresserweiterung angegeben werden.
Beispiel: `000 == STRINGIS("M02")`, aber `400 == STRINGIS("M")`.

3) Namen von Zyklenparametern können mit STRINGIS nicht geprüft werden.
4) Als Makro definierte Adress z.B. G, H, M, L werden als Makro identifiziert

Beispiele

In den folgenden Beispielen wird angenommen, dass die als String angegebenen NC-Sprachelement, sofern nicht besonders vermerkt, in der Steuerung prinzipiell programmierbar sind.

1. Der String "T" ist als Hilfsfunktion definiert:
   ```
   400 == STRINGIS("T")
   000 == STRINGIS ("T3")
   ```
2. Der String "X" ist als Achse definiert:
   ```
   201 == STRINGIS("X")
   201 == STRINGIS("X1")
   ```
3. Der String "A2" ist als NC–Adresse mit Erweiterung definiert:
   ```
   201 == STRINGIS("A")
   201 == STRINGIS("A2")
   ```
4. Der String "INVCW" ist als benamter G–Code definiert:
   ```
   202 == STRINGIS("INVCW")
   ```
5. Der String "$MC_GCODE_RESET_VALUES" ist als Maschinendatum definiert:
   ```
   206 == STRINGIS("$MC_GCODE_RESET_VALUES")
   ```
6. Der String "GETMDACT" ist eine NC–Sprachfunktion:
   ```
   203 == STRINGIS("GETMDACT ")
   ```
7. Der String "DEFINE" ist ein Schlüsselwort:
   ```
   205 == STRINGIS("DEFINE")
   ```
8. Der String "$TC_DP3" ist ein Systemparameter (Werkzeuglängenkomponente):
   ```
   207 == STRINGIS("$TC_DP3")
   ```
9. Der String "$TC_TP4" ist ein Systemparameter (Werkzeuggröße):
   ```
   207 == STRINGIS("$TC_TP4")
   ```
10. Der String "$TC_MPP4" ist ein Systemparameter (Magazinplatzzustand):
    - Die Werzeugmagazin–Verwaltung ist aktiv: `207 == STRINGIS("$TC_MPP4") ;`
    - Die Werzeugmagazin–Verwaltung ist nicht aktiv: `000 == STRINGIS("$TC_MPP4")`

    Siehe auch unten Absatz: Werkzeugmagazin-Verwaltung.
11. Der String "MACHINERY_NAME" ist als GUD-Variable definiert:
    ```
    209 == STRINGIS("MACHINERY_NAME")
    ```
12. Der String "LONGMACRO" ist als Makro definiert:
    ```
    210 == STRINGIS("LONGMACRO")
    ```
13. Der String "MYVAR" ist als LUD-Variable definiert:
    ```
    211 == STRINGIS("MYVAR")
    ```
14. Der String "XYZ" ist kein im NCK bekannter Befehl, GUD-Variable, Makro- oder Zyklen-Name:
    ```
    000 == STRINGIS("XYZ")
    ```

Werkzeugmagazin-Verwaltung

Ist die Funktion Werkzeugmagazin-Verwaltung nicht aktiv, liefert STRINGIS für die Systemparameter der Werkzeugmagazin-Verwaltung , unabhängig vom Maschinendatum

- MD10711 $MN_NC_LANGUAGE_CONFIGURATION

immer den Wert 000.

ISO Modus

Ist die Funktion "ISO Modus" aktiv:

- MD18800 $MN_MM_EXTERN_LANGUAGE (Aktivierung externer NC-Sprachen)
- MD10880 $MN_ MM_EXTERN_CNC_SYSTEM (zu adaptierendes Steuerungssystems)

überprüft STRINGIS den angegebenen String zuerst als SINUMERIK G-Code. Ist der String kein SINUMERIK G-Code wird er anschließend als ISO G-Code überprüft.

Programmierte Umschaltungen (`G290` (SINUMERIK Mode), `G291` (ISO Mode)) haben auf STRINGIS keine Auswirkung.

Beispiel

Die für die Funktion STRINGIS(...) relevanten Maschinendaten haben folgende Werte:

- MD10711 $MN_NC_LANGUAGE_CONFIGURATION = 2 (Es werden nur die NC-Sprachbefehle als bekannt angesehen, deren Optionen gesetzt sind)
- MD19410 $ON_TRAFO_TYPE_MASK = 'H0' (Option: Transformationen)
- MD10700 $MN_PREPROCESSING_LEVEL='H43' (Vorverarbeitung für Zyklen aktiv)

Das folgende Beispielprogramm wird ohne Fehlermeldung abgearbeitet:

```
Programmcode                              Kommentar
N1 R1=STRINGIS("TRACYL")                  ;   R1 == 0, da TRACYL wegen der fehlenden
                                          ;   Transformations-Option als "nicht bekannt"
                                          ;   erkannt wird
N2 IF STRINGIS("TRACYL") == 204           ;
N3 TRACYL(1,2,3)                          ;   N3 wird übersprungen
N4 ELSE
N5 G00                                    ;   und stattdessen N5 ausgeführt
N6 ENDIF
N7 M30
```

14.6 Funktionsaufruf ISVAR und Maschinendaten Array-Index lesen

Funktion

Der ISVAR-Befehl ist eine Funktion im Sinne der NC-Sprache mit einem:

- Funktionswert vom Typ BOOL
- Übergabeparameter vom Typ STRING

Der ISVAR-Befehl liefert TRUE, wenn der Übergabeparameter eine in der NC bekannte Variable enthält (Maschinendatum, Settingdatum, Systemvariable, allgemeine Variablen wie GUD's).

Syntax

```
ISVAR(<Variablenbezeichner>)
ISVAR(<Bezeichner>,[<Wert>,<Wert>])
```

Bedeutung

`<Variablenbezeichner>`	Übergabeparameter vom Typ String kann entweder dimensionslos, eindimensional oder zweidimensional sein.
`<Bezeichner>`	Bezeichner mit einer der NC bekannten Variable mit oder ohne Array-Index als Maschinendatum, Settingdatum, Systemvariable oder allgemeine Variable. **Erweiterung:** Bei allgemeinen und kanalspezifischen Maschinendaten wird das erste Element des Array auch bei fehlenden Index gelesen
`<Wert>`	Funktionswert vom Typ BOOL

Prüfungen

Entsprechend dem Übergabeparameter werden folgende Prüfungen durchgeführt:

- Ist der Bezeichner vorhanden
- Handelt es sich um ein- oder zweidimensionales Feld
- Ist ein Array-Index erlaubt

Nur wenn alle diese Prüfungen positiv sind, wird TRUE zurückgeliefert. Wird nur eine Prüfung nicht erfüllt oder ist ein Syntaxfehler aufgetreten, dann wird dies mit FALSE quittiert. Axialen Variablen werden als Index für die Achsnamen akzeptiert, jedoch nicht näher geprüft.

Erweiterung: Maschinendaten und Settingdaten Array ohne Index lesen.

Bei fehlenden Index von **allgemeinen und kanalspezifischen** Maschinendaten wird der Alarm 12400 "Kanal % 1 Satz % 2 Feld % 3 Element nicht vorhanden" **nicht mehr** ausgegeben.

Weiterhin muss **mindestens der Achsindex** bei **achsspezifischen** Maschinendaten programmiert werden. Anderenfalls wird der Alarm 12400 abgesetzt.

Beispiel: Funktionaufruf ISVAR

Programmcode	Kommentar
`DEF INT VAR1`	
`DEF BOOL IS_VAR=FALSE`	`; Übergabeparameter ist allgemeine Variable`
`N10 IS_VAR=ISVAR("VAR1")`	`; IS_VAR ist in diesem Fall TRUE`
`DEF REAL VARARRAY[10,10]`	
`DEF BOOL IS_VAR=FALSE`	`; verschiedene Syntaxvarianten`
`N20 IS_VAR=ISVAR("VARARRAY[,]")`	`; IS_VAR ist TRUE mit einen zweidimensionalen Array`
`N30 IS_VAR=ISVAR("VARARRAY")`	`; IS_VAR ist TRUE, Variable existiert`
`N40 IS_VAR=ISVAR("VARARRAY[8,11]")`	`; IS_VAR ist FALSE, Arrayindex ist nicht erlaubt`
`N50 IS_VAR=ISVAR("VARARRAY[8,8")`	`; IS_VAR ist FALSE, Syntaxfehler für fehlende "]"`
`N60 IS_VAR=ISVAR("VARARRAY[,8]")`	`; IS_VAR ist TRUE, Arrayindex ist erlaubt`
`N70 IS_VAR=ISVAR("VARARRAY[8,]")`	`; IS_VAR ist TRUE`
`DEF BOOL IS_VAR=FALSE`	`; Übergabeparameter ist ein Maschinendatum`
`N100 IS_VAR=ISVAR("$MC_GCODE_RESET_VALUES[1]"`	`; IS_VAR ist TRUE`
`DEF BOOL IS_VAR=FALSE`	`; Übergabeparameter ist eine Systemvariable`
`N10 IS_VAR=ISVAR("$P_EP")`	`; IS_VAR ist in diesem Fall TRUE`
`N10 IS_VAR=ISVAR("$P_EP[X]")`	`; IS_VAR ist in diesem Fall TRUE`

Beispiel: Maschinendaten Array mit und ohne Index lesen

Das erste Element wird gelesen bei

```
R1=$MC_EXTERN_GCODE_RESET_VALUES
```

dies entspricht wie bisher

```
R1=$MC_EXTERN_GCODE_RESET_VALUES[0]
```

oder gelesen wird das erste Element

```
R1=$MA_POSTCTRL_GAIN[X1]
```

dies entspricht wie bisher

```
R1=$MA_POSTCTRL_GAIN[0, X1]
```

Gelesen wird auch das erste Element in Synchronaktionen bei

```
WHEN TRUE DO $R1 = $MC_EXTERN_GCODE_RESET_VALUES
```

dies entspricht wie bisher

```
WHEN TRUE DO $R1 = $MC_EXTERN_GCODE_RESET_VALUES[0]
```

und wurde bisher mit Alarm 12400 **nicht gelesen.**

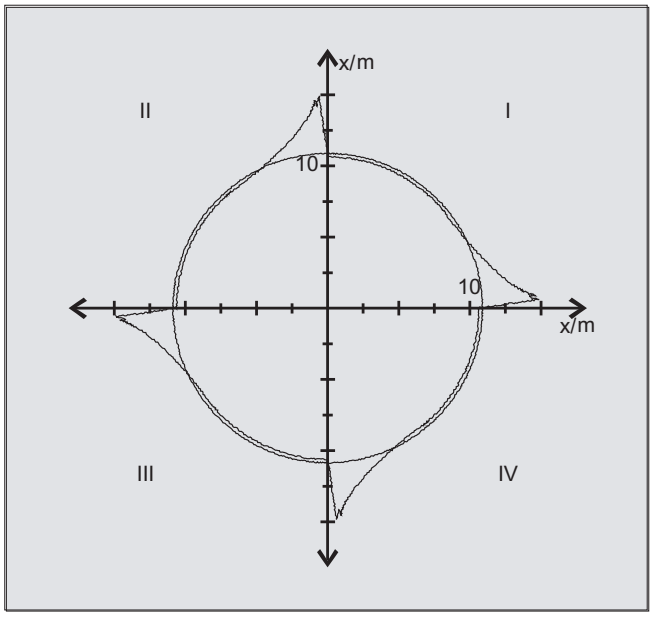
Der Alarm 12400 wird weiterhin ausgegeben bei

```
R1=$MA_POSTCTRL_GAIN
```

14.7 Kompensationskennlinien einlernen (QECLRNON, QECLRNOF)

Funktion .Fehler! Textmarke nicht definiert..

Die Quadrantenfehlerkompensation (QFK) reduziert die Konturfehler, die bei Umkehr der Fahrtrichtung durch mechanische Nichtlinearitäten (z. B. Reibung, Lose) oder Torsion entstehen. Die optimalen Kompensationsdaten können aufgrund eines neuronalen Netzes von der Steuerung während einer Lernphase adaptiert und so die Kompensationskennlinien automatisch ermittelt werden. Das Lernen kann für bis zu 4 Achsen gleichzeitig erfolgen.



Syntax

```
QECLRNON
QECLRNOF
```

Lernvorgang aktivieren: QECLRNON

Der eigentliche Lernvorgang wird im NC-Programm mit dem Befehl `QECLRNON` unter Angabe der Achsen aktiviert:

```
QECLRNON (X1, Y1, Z1, Q)
```

Nur wenn dieser Befehl aktiv ist, werden die Kennlinien verändert.

Lernen ausschalten: QECLRNOF

Nachdem die Lernbewegungen der gewünschten Achsen abgeschlossen sind, wird der Lernvorgang mit `QECLRNOF` für alle Achsen gleichzeitig ausgeschaltet.

Bedeutung

`QECLRNON (Achse.1,…4)`	Funktion "Quadrantenfehlerkompensation lernen" einschalten
`QECLRNO`	Funktion "Quadrantenfehlerkompensation lernen" ausschalten
`QECLRN.SPF`	Lernzyklus
`QECDAT.MPF`	Muster-NC-Programm für Belegen der Systemvariablen und für die Parametrierung des Lernzyklus
`QECTEST.MPF`	Muster-NC-Programm für Kreisformtest

Beschreibung

Die zum Lernen erforderlichen Verfahrbewegungen der Achsen werden mit Hilfe eines NC-Programms generiert. In diesem liegen die Lernbewegungen in Form eines Lernzyklus vor.

Erstmaliges Einlernen

Zum erstmaligen Einlernen bei der Inbetriebnahme sind auf der Diskette des PLC-Grundprogramms Muster-NC-Programme zum Einlernen für die Lernbewegungen sowie für die Belegung der QFK-Systemvariablen enthalten:

Nachlernen

Eine Nachoptimierung der bereits gelernten Kennlinien ist mit "Nachlernen" möglich. Dabei wird auf den bisherigen im Anwenderspeicher befindlichen Daten aufgesetzt. Zum Nachlernen passen Sie die Muster-NC-Programme an Ihre Anforderungen an.

Die Parameter des Lernzyklus (z. B. QECLRN.SPF) sind gegebenenfalls für das "Nachlernen" abzuändern:

- "Lernmodus" = 1 setzen
- "Anzahl der Lerndurchläufe" ggf. reduzieren
- "Abschnittsweises Lernen" ggf. aktivieren und zugehörige Bereichsgrenzen festlegen

14.8 Fenster aus dem Teileprogramm interaktiv aufrufen (MMC)

Funktion

Über den Befehl MMC können aus dem Teileprogramm auf dem HMI anwenderdefinierte Dialogfenster (Dialogbilder) angezeigt werden.

Das Aussehen der Dialogfenster wird durch rein textuelle Projektierung festgelegt (COM-Datei im Zyklenverzeichnis), die HMI -System-Software bleibt dabei unverändert.

Anwenderdefinierte Dialogfenster können nicht zeitgleich in verschiedenen Kanälen aufgerufen werden.

Syntax

```
MMC(CYCLES,PICTURE_ON,T_SK.COM,BILD,MGUD.DEF,BILD_3.AWB,TEST_1,A1","
S")
```

Bedeutung

MMC	Aus dem Teileprogramm interaktiv Dialogfenster am HMI aufrufen.
CYCLES	Bedienbereich, in dem die projektierten Anwenderdialoge ausgeführt werden.
PICTURE_ON bzw. PICTURE_OFF	Befehl: Bildanwahl bzw. Bildabwahl
T_SK.COM	Com-Datei: Name der Dialogbild-Datei (Anwenderzyklen). Hier ist das Aussehen der Dialogbilder festgelegt. Im Dialogbild können Anwendervariablen und/oder Kommentartexte angezeigt werden.
BILD	Dialogbildname: Die einzelnen Bilder werden durch den Dialogbildnamen ausgewählt.
MGUD.DEF	Anwenderdatendefinitionsdatei, auf die beim Lesen/Schreiben von Variablen zugegriffen wird.
BILD_3.AWB	Grafikdatei
TEST_1	Anzeigezeit oder Quittungsvariable
A1	Textvariablen...",
"S"	Quittungsmodus: synchron, Quittung über den Softkey "OK"

Literatur

Detaillierte Hinweise zur Programmierung des MMC-Befehls (inkl. Programmierbeispiele) siehe Inbetriebnahmehandbuch.

14.9 Programmlaufzeit / Werkstückzähler

14.9.1 Programmlaufzeit / Werkstückzähler (Übersicht)

Zur Unterstützung des Werkzeugmaschinenbedieners werden Informationen zur Programmlaufzeit und Werkstückzahl bereitgestellt.

Diese Informationen können als Systemvariablen im NC- und/oder PLC-Programm bearbeitet werden. Gleichzeitig stehen diese Informationen für die Anzeige auf der Bedienoberfläche zur Verfügung.

14.9.2 Programmlaufzeit

Funktion

Die Funktion "Programmlaufzeit" stellt NC-interne Timer zur Überwachung technologischer Prozesse zur Verfügung, die über NC- und Kanal-spezifische Systemvariablen im Teileprogramm und in Synchronaktionen gelesen werden können.

Der Trigger zur Laufzeitmessung ($AC_PROG_NET_TIME_TRIGGER) ist die einzige schreibbare Systemvariable der Funktion und dient zur selektiven Messung von Programmabschnitten. D. h. durch Beschreiben des Triggers im NC-Programm kann die Zeitmessung ein- und wieder ausgeschaltet werden.

Systemvariable	Bedeutung	Aktivität
NC-spezifisch		
$AN_SETUP_TIME	Zeit seit dem letzten Steuerungshochlauf mit Standardwerten ("Kaltstart") in Minuten Wird bei jedem Steuerungshochlauf mit Standardwerten automatisch auf "0" zurückgesetzt.	• immer aktiv
$AN_POWERON_TIME	Zeit seit dem letzten Normalhochlauf der Steuerung ("Warmstart") in Minuten Wird bei jedem Normalhochlauf der Steuerung automatisch auf "0" zurückgesetzt.	

Systemvariable	Bedeutung	Aktivität
Kanal-spezifisch		
$AC_OPERATING_TIME	Gesamtlaufzeit von NC-Programmen in der Betriebsart Automatik in Sekunden Der Wert wird mit jedem Steuerungshochlauf automatisch auf "0" zurückgesetzt.	• Aktivierung über MD27860 • nur Betriebsart AUTOMATIK
$AC_CYCLE_TIME	Laufzeit des angewählten NC-Programms in Sekunden Der Wert wird mit dem Start eines neuen NC-Programms automatisch auf "0" zurückgesetzt.	
$AC_CUTTING_TIME	Bearbeitungszeit in Sekunden Gemessen wird die Laufzeit der Bahnachsen (mindestens eine ist aktiv) ohne aktiven Eilgang in allen NC-Programmen zwischen NC-Start und Programmende / NC-Reset. Die Messung wird zusätzlich bei aktiver Verweilzeit unterbrochen. Der Wert wird bei jedem Steuerungshochlauf mit Standardwerten automatisch auf "0" zurückgesetzt.	
$AC_ACT_PROG_NET_TIME	Aktuelle Netto-Laufzeit des aktuellen NC-Programms in Sekunden Wird mit dem Start eines NC-Programms automatisch auf "0" zurückgesetzt.	• immer aktiv • nur Betriebsart AUTOMATIK
$AC_OLD_PROG_NET_TIME	Netto-Laufzeit des gerade korrekt mit `M30` beendeten Programms in Sekunden	
$AC_OLD_PROG_NET_TIME_COUNT	Änderungen auf $AC_OLD_PROG_NET_TIME Nach POWER ON steht $AC_OLD_PROG_NET_TIME_COUNT auf "0". $AC_OLD_PROG_NET_TIME_COUNT wird immer dann erhöht, wenn die Steuerung $AC_OLD_PROG_NET_TIME neu geschrieben hat.	

Systemvariable	Bedeutung		Aktivität
$AC_PROG_NET_TIME_TRIGGER	Trigger zur Laufzeitmessung:		• nur Betriebsart AUTOMATIK
	0	Neutraler Zustand Der Trigger ist nicht aktiv.	
	1	Beenden Beendet die Messung und kopiert den Wert aus $AC_ACT_PROG_NET_TIME in $AC_OLD_PROG_NET_TIME. $AC_ACT_PROG_NET_TIME wird auf "0" gesetzt und läuft danach weiter.	
	2	Start Startet die Messung und setzt dabei $AC_ACT_PROG_NET_TIME auf "0". $AC_OLD_PROG_NET_TIME wird nicht verändert.	
	3	Stopp Stoppt die Messung. Verändert $AC_OLD_PROG_NET_TIME nicht und hält $AC_ACT_PROG_NET_TIME bis zum Fortsetzen konstant.	
	4	Fortsetzen Fortsetzen der Messung, d. h. eine vorher gestoppte Messung wird wieder aufgenommen. $AC_ACT_PROG_NET_TIME läuft weiter. $AC_OLD_PROG_NET_TIME wird nicht verändert.	
Durch POWER ON werden alle Systemvariablen auf "0" zurückgesetzt!			
Literatur: Eine ausführliche Beschreibung der aufgelisteten Systemvariablen findet sich in: Funktionshandbuch Grundfunktionen; BAG, Kanal, Programmbetrieb, Reset-Verhalten (K1), Kapitel: Programmlaufzeit			

Hinweis

Maschinenhersteller

Das Einschalten der aktivierbaren Timer erfolgt über das Maschinendatum MD27860 $MC_PROCESSTIMER_MODE.

Das Verhalten der aktiven Zeitmessungen bei bestimmten Funktionen (z. B. GOTOS, Override = 0%, aktiver Probelaufvorschub, Programmtest, ASUP, PROG_EVENT, …) wird konfiguriert über die Maschinendaten MD27850 $MC_PROG_NET_TIMER_MODE und MD27860 $MC_PROCESSTIMER_MODE.

Literatur:
Funktionshandbuch Grundfunktionen; BAG, Kanal, Programmbetrieb, Reset-Verhalten (K1), Kapitel: Programmlaufzeit

Hinweis

Restzeit für ein Werkstück

Wenn nacheinander gleiche Werkstücke produziert werden, kann aus den Timerwerten:

- Bearbeitungszeit für das zuletzt produzierte Werkstück (siehe $AC_OLD_PROG_NET_TIME)

und

- aktuelle Bearbeitungszeit (siehe $AC_ACT_PROG_NET_TIME)

die verbleibende Restzeit für ein Werkstück ermittelt werden.

Die Restzeit wird zusätzlich zur aktuellen Bearbeitungszeit auf der Bedienoberfläche anzeigt.

ACHTUNG

Verwendung von STOPRE

Die Systemvariablen $AC_OLD_PROG_NET_TIME und $AC_OLD_PROG_NET_TIME_COUNT erzeugen keinen impliziten Vorlaufstopp. Bei der Verwendung im Teileprogramm ist das unkritisch, wenn der Wert der Systemvariablen aus dem vorangegangen Programmlauf stammt. Wenn aber der Trigger zur Laufzeitmessung ($AC_PROG_NET_TIME_TRIGGER) hochfrequent geschrieben wird und sich dadurch $AC_OLD_PROG_NET_TIME sehr oft ändert, dann sollte im Teileprogramm ein explizites `STOPRE` verwendet werden.

Randbedingungen

- **Satzsuchlauf**

  Bei Satzsuchlauf werden keine Programmlaufzeiten ermittelt.

- **REPOS**

  Die Zeitdauer eines REPOS-Vorgangs wird der aktuellen Bearbeitungszeit ($AC_ACT_PROG_NET_TIME) angerechnet.

Beispiele

Beispiel 1: Zeitdauer von "mySubProgrammA" messen

Programmcode

```
...
N50 DO $AC_PROG_NET_TIME_TRIGGER=2
N60 FOR ii= 0 TO 300
N70 mySubProgrammA
N80 DO $AC_PROG_NET_TIME_TRIGGER=1
N95 ENDFOR
N97 mySubProgrammB
N98 M30
```

Nachdem das Programm die Zeile `N80` verarbeitet hat, steht in $AC_OLD_PROG_NET_TIME die Nettolaufzeit von "mySubProgrammA".

Der Wert von $AC_OLD_PROG_NET_TIME:

- beibt über `M30` hinaus erhalten.
- wird nach jedem Schleifendurchlauf aktualisiert.

Beispiel 2: Zeitdauer von "mySubProgrammA" und "mySubProgrammC" messen

Programmcode

```
...
N10 DO $AC_PROG_NET_TIME_TRIGGER=2
N20 mySubProgrammA
N30 DO $AC_PROG_NET_TIME_TRIGGER=3
N40 mySubProgrammB
N50 DO $AC_PROG_NET_TIME_TRIGGER=4
N60 mySubProgrammC
N70 DO $AC_PROG_NET_TIME_TRIGGER=1
N80 mySubProgrammD
N90 M30
```

14.9.3 Werkstückzähler

Funktion

Die Funktion "Werkstückzähler" stellt diverse Zähler zur Verfügung, die insbesondere für die steuerungsinterne Zählung von Werkstücken verwendet werden können.

Die Zähler existieren als kanalspezifische Systemvariablen mit Schreib- und Lese-Zugriff im Wertebereich von 0 bis 999 999 999.

Systemvariable	Bedeutung
$AC_REQUIRED_PARTS	Anzahl der zu fertigenden Werkstücke (Soll-Werkstückzahl) In diesem Zähler kann die Anzahl der Werkstücke definiert werden, bei dessen Erreichen die Ist-Werkstückzahl ($AC_ACTUAL_PARTS) auf "0" zurückgesetzt wird.
$AC_TOTAL_PARTS	Anzahl der insgesamt gefertigten Werkstücke (Ist-Werkstückzahl Gesamt) Dieser Zähler gibt die Anzahl aller ab Startzeitpunkt gefertigten Werkstücke an. Der Wert wird nur bei einem Steuerungshochlauf mit Standardwerten automatisch auf "0" zurückgesetzt.
$AC_ACTUAL_PARTS	Anzahl der gefertigten Werkstücke (Ist-Werkstückzahl) In diesem Zähler wird die Anzahl aller ab Startzeitpunkt gefertigten Werkstücke registriert. Bei einem Erreichen der Soll-Werkstückzahl ($AC_REQUIRED_PARTS) wird der Zähler automatisch auf "0" zurückgesetzt ($AC_REQUIRED_PARTS > 0 vorausgesetzt).
$AC_SPECIAL_PARTS	Anzahl der vom Anwender gezählten Werkstücke Dieser Zähler erlaubt dem Anwender eine Werkstückzählung nach eigener Definition. Definiert werden kann eine Alarmausgabe bei Erreichen der Soll-Werkstückzahl ($AC_REQUIRED_PARTS). Eine Nullung des Zählers muss der Anwender selbst vornehmen.

Hinweis

Alle Werkstückzähler werden bei einem Steuerungshochlauf mit Standardwerten auf "0" gesetzt und können unabhängig von ihrer Aktivierung gelesen und geschrieben werden.

Hinweis

Über kanalspezifische Maschinendaten kann auf die Zähler-Aktivierung, den Zeitpunkt der Nullung und den Zählalgorithmus Einfluss genommen werden.

Hinweis

Werkstückzählung mit anwenderdefiniertem M-Befehl

Über Maschinendaten kann eingestellt werden, dass die Zählimpulse für die verschiedenen Werkstückzähler statt über das Programmende `M2`/`M30` über anwenderdefinierte M-Befehle ausgelöst werden.

Literatur

Weitere Informationen zur Funktion "Werkstückzähler" siehe:

- Funktionshandbuch Grundfunktionen; BAG, Kanal, Programmbetrieb, Reset-Verhalten (K1), Kapitel: Werkstückzähler

14.10 Alarme (SETAL)

Funktion

In einem NC-Programm können Alarme gesetzt werden. Diese werden in der Bedienoberfläche in einem besonderen Feld dargestellt. Mit einem Alarm ist jeweils eine Reaktion der Steuerung entsprechend der Alarmkategorie verbunden.

Literatur:
Weiterführende Informationen zu den Alarmreaktionen siehe Inbetriebnahmehandbuch.

Syntax

```
SETAL(<Alarmnummer>)
SETAL(<Alarmnummer>,<Zeichenkette>)
```

Bedeutung

`SETAL`	Schlüsselwort zur Programmierung eines Alarms. `SETAL` muss in einem eigenen NC-Satz programmiert werden.
`<Alarmnummer>`	Variable vom Typ INT. Enthält die Alarmnummer. Der gültige Bereich für Alarmnummern liegt zwischen 60000 und 69999, wovon 60000 bis 64999 für SIEMENS-Zyklen reserviert sind und 65000 bis 69999 für den Anwender zur Verfügung stehen.
`<Zeichenkette>`	Bei der Programmierung von Anwenderzyklenalarmen kann zusätzlich eine Zeichenkette mit bis zu 4 Parametern angegeben werden. In diesen Parametern können variable Anwendertexte definiert werden. Es stehen aber auch folgende vordefinierte Parameter zur Verfügung:

Parameter	Bedeutung
%1	Kanalnummer
%2	Satznummer, Label
%3	Textindex für Zyklenalarme
%4	zusätzlicher Alarmparameter

Hinweis

Alarmtexte müssen in der Bedienoberfläche projektiert werden.

Beispiel

Programmcode	Kommentar
...	
N100 SETAL(65000)	; Alarm Nr. 65000 setzen
...	

Eigene Abspanprogramme 15

15.1 Unterstützende Funktionen für das Abspanen

Funktionen

Für das Abspanen werden Ihnen fertige Bearbeitungszyklen angeboten. Darüber hinaus haben Sie die Möglichkeit, mit den nachfolgend aufgeführten Funktionen eigene Abspanprogramme zu erstellen:

- Konturtabelle erstellen (CONTPRON)
- Codierte Konturtabelle erstellen (CONTDCON)
- Konturaufbereitung ausschalten (EXECUTE)
- Schnittpunkt zwischen zwei Konturelementen ermitteln (INTERSEC)

  (Nur für Tabellen, die durch CONTPRON erstellt wurden.)
- Konturelemente einer Tabelle satzweise abarbeiten (EXECTAB)

  (Nur für Tabellen, die durch CONTPRON erstellt wurden.)
- Kreisdaten berechnen (CALCDAT)

Hinweis

Sie können diese Funktionen nicht nur zum Abspanen, sondern universell einsetzen.

Voraussetzungen

Vor dem Aufruf der Funktionen CONTPRON oder CONTDCON müssen:

- ein Startpunkt angefahren werden, der eine kollisionsfreie Bearbeitung erlaubt.
- die Schneidenradiuskorrektur mit `G40` ausgeschaltet sein.

15.2 Konturtabelle erstellen (CONTPRON)

Funktion

Mit dem Befehl CONTPRON wird die Konturaufbereitung eingeschaltet. Die nachfolgend aufgerufenen NC-Sätze werden nicht abgearbeitet, sondern in einzelne Bewegungen aufgeteilt und in der Konturtabelle abgelegt. Jedem Konturelement entspricht eine Tabellenzeile im zweidimensionalen Feld der Konturtabelle. Die Anzahl der ermittelten Hinterschnitte wird zurückgeliefert.

Syntax

Konturaufbereitung einschalten:
```
CONTPRON(<Konturtabelle>,<Bearbeitungsart>,<Hinterschnitte>,
<Bearbeitungsrichtung>)
```
Konturaufbereitung ausschalten und in den normalen Abarbeitungsmodus zurückkehren:
```
EXECUTE(<FEHLER>)
```
Siehe " Konturaufbereitung ausschalten (EXECUTE) "

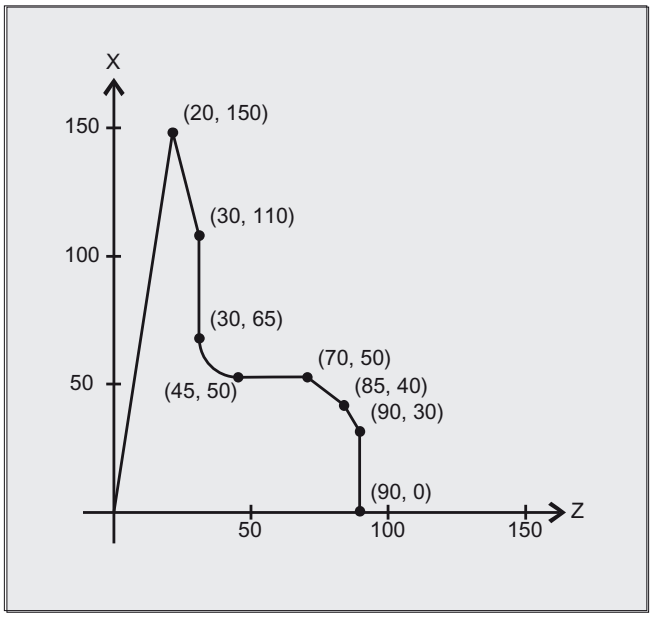
Bedeutung

CONTPRON	Befehl zum Einschalten der Konturaufbereitung zur Erstellung einer Konturtabelle		
<Konturtabelle>	Name der Konturtabelle		
<Bearbeitungsart>	Parameter für die Bearbeitungsart		
	Typ:	CHAR	
	Wert:	"G"	Längsdrehen: Innenbearbeitung
		"L"	Längsdrehen: Außenbearbeitung
		"N"	Plandrehen: Innenbearbeitung
		"P"	Plandrehen: Außenbearbeitung
<Hinterschnitte>	Ergebnisvariable für die Anzahl auftretender Hinterschnittelemente		
	Typ:	INT	
<Bearbeitungsrichtung>	Parameter für die Bearbeitungsrichtung		
	Typ:	INT	
	Wert:	0	Konturaufbereitung vorwärts (Standardwert)
		1	Konturaufbereitung in beiden Richtungen

Beispiel 1

Erstellen einer Konturtabelle mit:

- Namen "KTAB"
- max. 30 Konturelementen (Kreise, Geraden)
- einer Variablen für die Anzahl auftretender Hinterschnittelemente
- einer Variablen für Fehlermeldungen



NC-Programm:

Programmcode	**Kommentar**
N10 DEF REAL KTAB[30,11]	; Konturtabelle mit Namen KTAB und max. 30 Konturelementen, Parameterwert 11 (Spaltenzahl der Tabelle) ist eine feste Größe.
N20 DEF INT ANZHINT	; Variable für die Anzahl der Hinterschnittelemente mit Namen ANZHINT.
N30 DEF INT FEHLER	; Variable für die Fehlerrückmeldung (0=kein Fehler, 1=Fehler).
N40 G18	
N50 CONTPRON(KTAB,"G",ANZHINT)	; Konturaufbereitung einschalten.
N60 G1 X150 Z20	; N60 bis N120: Konturbeschreibung
N70 X110 Z30	
N80 X50 RND=15	
N90 Z70	
N100 X40 Z85	
N110 X30 Z90	
N120 X0	

Programmcode	Kommentar
N130 EXECUTE(FEHLER)	; Füllen der Konturtabelle beenden, Umschalten auf normalen Programmbetrieb.
N140 …	; Weitere Bearbeitung der Tabelle.

Konturtabelle KTAB:

Index Zeile	Spalte									
(0)	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
7	7	11	0	0	20	150	0	82.40535663	0	0
0	2	11	20	150	30	110	-1111	104.0362435	0	0
1	3	11	30	110	30	65	0	90	0	0
2	4	13	30	65	45	50	0	180	45	65
3	5	11	45	50	70	50	0	0	0	0
4	6	11	70	50	85	40	0	146.3099325	0	0
5	7	11	85	40	90	30	0	116.5650512	0	0
6	0	11	90	30	90	0	0	90	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

Erläuterung der Spalteninhalte:

(0) Zeiger auf nächstes Konturelement (auf die Zeilennummer desselben)

(1) Zeiger auf vorhergehendes Konturelement

(2) Codierung des Konturmodus für die Bewegung

Mögliche Werte für X = abc

a = 10^2 G90 = 0 G91 = 1

b = 10^1 G70 = 0 G71 = 1

c = 10^0 G0 = 0 G1 = 1 G2 = 2 G3 = 3

(3), (4) Anfangspunkt der Konturelemente

(3) = Abszisse, (4) = Ordinate in der aktuellen Ebene

(5), (6) Endpunkt der Konturelemente

(5) = Abszisse, (6) = Ordinate in der aktuellen Ebene

(7) Max-/min-Anzeiger: kennzeichnet lokale Maxima und Minima in der Kontur

(8) Maximaler Wert zwischen Konturelement und Abszisse (bei Längsbearbeitung) bzw. Ordinate (bei Planbearbeitung). Der Winkel ist abhängig von der programmierten Bearbeitungsart.

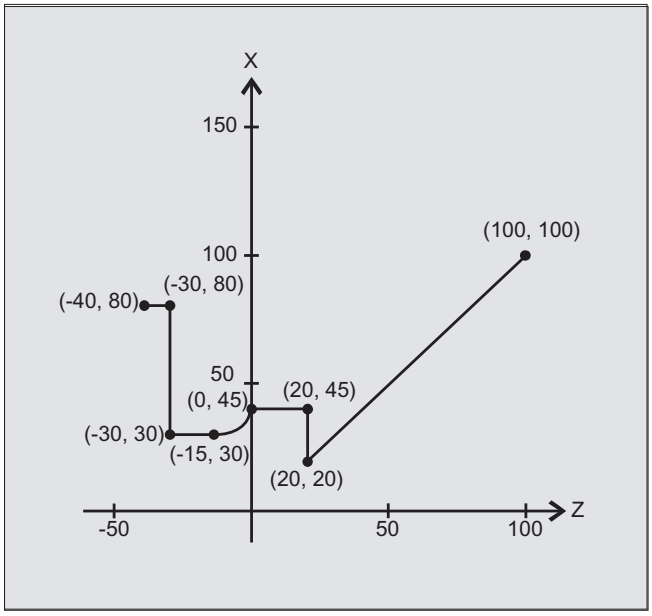
(9), (10) Mittelpunktskoordinaten des Konturelements, wenn es ein Kreissatz ist.

(9) = Abszisse, (10) = Ordinate

Beispiel 2

Erstellen einer Konturtabelle mit

- Namen KTAB
- max. 92 Konturelementen (Kreise, Geraden)
- Betriebsart: Längsdrehen, Außenbearbeitung
- Aufbereitung vorwärts und rückwärts



NC-Programm:

Programmcode	**Kommentar**
N10 DEF REAL KTAB[92,11]	; Konturtabelle mit Namen KTAB und max. 92 Konturelementen, Parameterwert 11 ist eine feste Größe.
N20 DEF CHAR BT="L"	; Betriebsart für CONTPRON: Längsdrehen, Außenbearbeitung
N30 DEF INT HE=0	; Anzahl der Hinterschnittelemente=0
N40 DEF INT MODE=1	; Aufbereitung vorwärts und rückwärts
N50 DEF INT ERR=0	; Fehlerrückmeldung
...	
N100 G18 X100 Z100 F1000	
N105 CONTPRON(KTAB,BT,HE,MODE)	; Konturaufbereitung einschalten.

```
Programmcode                                    Kommentar
N110 G1 G90 Z20 X20
N120 X45
N130 Z0
N140 G2 Z-15 X30 K=AC(-15) I=AC(45)
N150 G1 Z-30
N160 X80
N170 Z-40
N180 EXECUTE(ERR)                               ; Füllen der Konturtabelle beenden,
                                                  Umschalten auf normalen
                                                  Programmbetrieb.
...
```

Konturtabelle KTAB:

Nach Ende der Konturaufbereitung steht die Kontur in beiden Richtungen zur Verfügung.

Index	Spalte										
Zeile	(0)	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
0	6[1)]	7[2)]	11	100	100	20	20	0	45	0	0
1	0[3)]	2	11	20	20	20	45	-3	90	0	0
2	1	3	11	20	45	0	45	0	0	0	0
3	2	4	12	0	45	-15	30	5	90	-15	45
4	3	5	11	-15	30	-30	30	0	0	0	0
5	4	7	11	-30	30	-30	45	-1111	90	0	0
6	7	0[4)]	11	-30	80	-40	80	0	0	0	0
7	5	6	11	-30	45	-30	80	0	90	0	0
8	1[5)]	2[6)]	0	0	0	0	0	0	0	0	0
	...										
83	84	0[7)]	11	20	45	20	80	0	90	0	0
84	90	83	11	20	20	20	45	-1111	90	0	0
85	0[8)]	86	11	-40	80	-30	80	0	0	0	0
86	85	87	11	-30	80	-30	30	88	90	0	0
87	86	88	11	-30	30	-15	30	0	0	0	0
88	87	89	13	-15	30	0	45	-90	90	-15	45
89	88	90	11	0	45	20	45	0	0	0	0
90	89	84	11	20	45	20	20	84	90	0	0
91	83[9)]	85[10)]	11	20	20	100	100	0	45	0	0

Erläuterung der Spalteninhalte und der Anmerkungen zu den Zeilen 0, 1, 6, 8, 83, 85 und 91

Es gelten die im Beispiel 1 genannten Erläuterungen der Spalteninhalte.

Immer in Tabellen-Zeile 0:

1) Vorgänger: Zeile n enthält das Konturende vorwärts

2) Nachfolger: Zeile n ist das Konturtabellenende vorwärts

Je einmal innerhalb der Konturelemente vorwärts:

3) Vorgänger: Konturbeginn (vorwärts)

4) Nachfolger: Konturende (vorwärts)

Immer auf Zeile Konturtabellenende (vorwärts) +1:

5) Vorgänger: Anzahl der Hinterschnitte vorwärts

6) Nachfolger: Anzahl der Hinterschnitte rückwärts

Je einmal innerhalb der Konturelemente rückwärts:

7) Nachfolger: Konturende (rückwärts)

8) Vorgänger: Konturbeginn (rückwärts)

Immer in letzter Tabellen-Zeile:

9) Vorgänger: Zeile n ist der Konturtabellenanfang (rückwärts)

10) Nachfolger: Zeile n enthält den Konturanfang (rückwärts)

Weitere Informationen

Erlaubte Verfahrbefehle, Koordinatensystem

Für die Konturprogrammierung sind folgende G-Befehle zulässig:

- G-Gruppe 1: `G0`, `G1`, `G2`, `G3`

Zusätzlich möglich sind:

- Rundung und Fase
- Kreisprogrammierung über `CIP` und `CT`

Die Funktionen Spline, Polynom und Gewinde führen zu Fehlern.

Änderungen des Koordinatensystems durch Einschalten eines Frames sind zwischen `CONTPRON` und `EXECUTE` nicht zulässig. Gleiches gilt für einen Wechsel zwischen `G70` und `G71` bzw. `G700` und `G710`.

Ein Tausch der Geometrieachsen mit `GEOAX` während der Aufbereitung der Konturtabelle führt zu einem Alarm.

Hinterschnittelemente

Die Konturbeschreibung der einzelnen Hinterschnittelemente kann wahlweise in einem Unterprogramm oder in einzelnen Sätzen erfolgen.

Abspanen unabhängig von der programmierten Konturrichtung

Die Konturaufbereitung mit `CONTPRON` wurde so erweitert, dass nach ihrem Aufruf die Konturtabelle unabhängig von der programmierten Richtung zur Verfügung steht.

15.3 Codierte Konturtabelle erstellen (CONTDCON)

Funktion

Bei der mit CONTDCON eingeschalteten Konturaufbereitung werden die nachfolgend aufgerufenen NC-Sätze in einer 6-spaltigen Konturtabelle speichergünstig codiert abgelegt. Jedem Konturelement entspricht eine Tabellenzeile in der Konturtabelle. Aus Kenntnis der unten angegebenen Codierungsregeln können Sie z. B. für Zyklen aus den Tabellenzeilen DIN-Code-Programme zusammenstellen. In der Tabellenzeile mit der Nummer 0 werden die Daten des Ausgangspunkts gespeichert.

Syntax

Konturaufbereitung einschalten:
`CONTDCON(<Konturtabelle>,<Bearbeitungsrichtung>)`

Konturaufbereitung ausschalten und in den normalen Abarbeitungsmodus zurückkehren:
`EXECUTE(<FEHLER>)`

Siehe " Konturaufbereitung ausschalten (EXECUTE) "

Bedeutung

`CONTDCON`	Befehl zum Einschalten der Konturaufbereitung zur Erstellung einer codierten Konturtabelle	
`<Konturtabelle>`	Name der Konturtabelle	
`<Bearbeitungsrichtung>`	Parameter für Bearbeitungsrichtung	
	Typ:	INT
	Wert: 0	Konturaufbereitung gemäß der Folge der Kontursätze (Standardwert)
	1	unzulässig

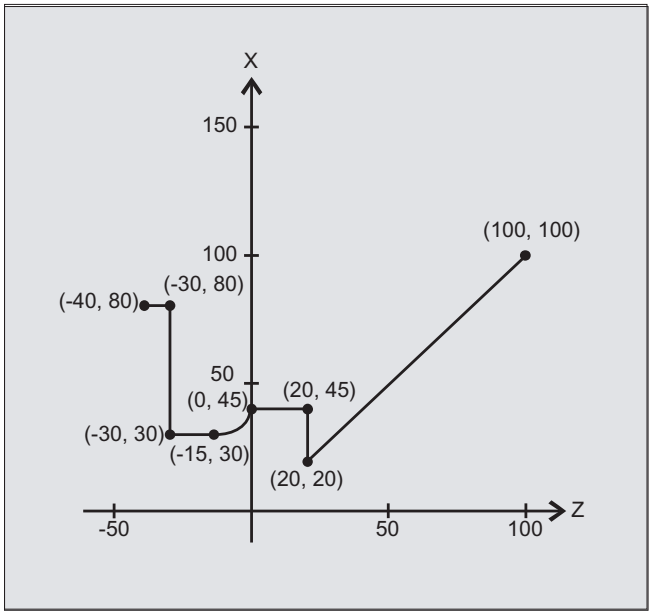
Hinweis

Die für CONTDCON zugelassenen G-Codes im zu tabellierenden Programmstück sind umfangreicher als bei CONTPRON. Darüber hinaus werden Vorschübe und Vorschubtyp pro Konturstück mitgespeichert.

Beispiel

Erstellen einer Konturtabelle mit:

- Namen "KTAB"
- Konturelementen (Kreise, Geraden)
- Betriebsart: Drehen
- Bearbeitungsrichtung: vorwärts



NC-Programm:

Programmcode	**Kommentar**
N10 DEF REAL KTAB[9,6]	; Konturtabelle mit Namen KTAB und 9 Tabellenzeilen. Diese erlauben 8 Kontursätze. Der Parameterwert 6 (Spaltenzahl der Tabelle) ist eine feste Größe.
N20 DEF INT MODE = 0	; Variable für die Bearbeitungsrichtung. Standardwert 0: nur in programmierter Richtung der Kontur.
N30 DEF INT ERROR = 0	; Variable für die Fehlerrückmeldung.
...	
N100 G18 G64 G90 G94 G710	
N101 G1 Z100 X100 F1000	

```
Programmcode                                    Kommentar
N105 CONTDCON (KTAB, MODE)                      ; Aufruf Konturaufbereitung (MODE
                                                  darf weggelassen werden).
N110 G1 Z20 X20 F200                            ; Konturbeschreibung.
N120 G9 X45 F300
N130 Z0 F400
N140 G2 Z-15 X30 K=AC(-15) I=AC(45)F100
N150 G64 Z-30 F600
N160 X80 F700
N170 Z-40 F800
N180 EXECUTE(ERROR)                             ; Füllen der Konturtabelle
                                                  beenden, Umschalten auf normalen
                                                  Programmbetrieb.
...
```

Konturtabelle KTAB:

	Spaltenindex					
	0	1	2	3	4	5
Zeilenindex	Kontur-modus	Endpunkt Abszisse	Endpunkt Ordinate	Mittelpunkt Abszisse	Mittelpunkt Ordinate	Vorschub
0	30	100	100	0	0	7
1	11031	20	20	0	0	200
2	111031	20	45	0	0	300
3	11031	0	45	0	0	400
4	11032	-15	30	-15	45	100
5	11031	-30	30	0	0	600
6	11031	-30	80	0	0	700
7	11031	-40	80	0	0	800
8	0	0	0	0	0	0

Erläuterung der Spalteninhalte:

Zeile 0: Codierungen für den **Startpunkt:**

Spalte 0: 10^0 (Einerstelle): G0 = 0
10^1 (Zehnerstelle): G70 = 0, G71 = 1, G700 = 2, G710 = 3
Spalte 1: Startpunkt Abszisse
Spalte 2: Startpunkt Ordinate
Spalte 3-4: 0
Spalte 5: Zeilenindex des letzten Konturstückes in der Tabelle

Zeilen 1-n: Einträge der **Konturstücke**

Spalte 0:	10^0 (Einerstelle): G0 = 0, G1 = 1, G2 = 2, G3 = 3
	10^1 (Zehnerstelle): G70 = 0, G71 = 1, G700 = 2, G710 = 3
	10^2 (Hunderterstelle): G90 = 0, G91 = 1
	10^3 (Tausenderstelle): G93 = 0, G94 = 1, G95 = 2, G96 = 3
	10^4 (Zehntausenderstelle): G60 = 0, G44 = 1, G641 = 2, G642 = 3
	10^5 (Hunderttausender Stelle): G9 = 1
Spalte 1:	Endpunkt Abszisse
Spalte 2:	Endpunkt Ordinate
Spalte 3:	Mittelpunkt Abszisse bei Kreisinterpolation
Spalte 4:	Mittelpunkt Ordinate bei Kreisinterpolation
Spalte 5:	Vorschub

Weitere Informationen

Erlaubte Verfahrbefehle, Koordinatensystem

Für die Konturprogrammierung sind folgende G-Gruppen und G-Befehle zulässig:

G-Gruppe 1:	`G0`, `G1`, `G2`, `G3`
G-Gruppe 10:	`G60`, `G64`, `G641`, `G642`
G-Gruppe 11:	`G9`
G-Gruppe 13:	`G70`, `G71`, `G700`, `G710`
G-Gruppe 14:	`G90`, `G91`
G-Gruppe 15:	`G93`, `G94`, `G95`, `G96`, `G961`

Zusätzlich möglich sind:

- Rundung und Fase
- Kreisprogrammierung über `CIP` und `CT`

Die Funktionen Spline, Polynom und Gewinde führen zu Fehlern.

Änderungen des Koordinatensystems durch Einschalten eines Frames sind zwischen `CONTDCON` und `EXECUTE` nicht zulässig. Gleiches gilt für einen Wechsel zwischen `G70` und `G71` bzw. `G700` und `G710`.

Ein Tausch der Geometrieachsen mit `GEOAX` während der Aufbereitung der Konturtabelle führt zu einem Alarm.

Bearbeitungsrichtung

Die mit `CONTDCON` erzeugte Konturtabelle ist zum Abspanen in der programmierten Richtung der Kontur vorgesehen.

15.4 Schnittpunkt zwischen zwei Konturelementen ermitteln (INTERSEC)

Funktion

INTERSEC ermittelt den Schnittpunkt von zwei normierten Konturelementen aus mit CONTPRON erzeugten Konturtabellen.

Syntax

```
<Status>=INTERSEC(<Konturtabelle_1>[<Konturelement_1>],
<Konturtabelle_2>[<Konturelement_2>],<Schnittpunkt>,<Bearbeitungsart
>)
```

Bedeutung

INTERSEC	Schlüsselwort zur Ermittlung des Schnittpunkts zweier Konturelemente aus mit CONTPRON erzeugten Konturtabellen		
<Status>	Variable für den Schnittpunktstatus		
	Typ:	BOOL	
	Wert:	TRUE	Schnittpunkt gefunden
		FALSE	kein Schnittpunkt gefunden
<Konturtabelle_1>	Name der ersten Konturtabelle		
<Konturelement_1>	Nummer des Konturelements der ersten Konturtabelle		
<Konturtabelle_2>	Name der zweiten Konturtabelle		
<Konturelement_2>	Nummer des Konturelements der zweiten Konturtabelle		
<Schnittpunkt>	Schnittpunkt-Koordinaten in der aktiven Ebene (G17 / G18 / G19)		
	Typ:	REAL	
<Bearbeitungsart>	Parameter für die Bearbeitungsart		
	Typ:	INT	
	Wert:	0	Schnittpunktberechnung in der mit Parameter 2 aktiven Ebene (Standardwert)
		1	Schnittpunktberechnung unabhängig der übergebenen Ebene

Hinweis

Beachten Sie, dass die Variablen vor ihrer Verwendung definiert sein müssen.

Die Übergabe der Konturen erfordert die Einhaltung der mit CONTPRON definierten Werte:

Parameter	Bedeutung
2	Codierung des Kontur-Mode für die Bewegung
3	Kontur-Anfangpunkt Abszisse
4	Kontur-Anfangpunkt Ordinate
5	Kontur-Endpunkt Abszisse
6	Kontur-Endpunkt Ordinate
9	Mittelpunkskoordinate für die Abszisse (nur bei Kreis-Kontur)
10	Mittelpunkskoordinate für die Ordinate (nur bei Kreis-Kontur)

Beispiel

Schnittpunkt von Konturelement 3 der Tabelle TABNAME1 und Konturelement 7 der Tabelle TABNAME2 ermitteln. Die Schnittpunkt-Koordinaten in der aktiven Ebene werden in der Variablen ISCOORD (1. Element = Abszisse, 2. Element = Ordinate) abgelegt. Existiert kein Schnittpunkt, erfolgt ein Sprung zu KEINSCH (kein Schnittpunkt gefunden).

```
Programmcode                                                    Kommentar
DEF REAL TABNAME1[12,11]                                        ; Konturtabelle 1
DEF REAL TABNAME2[10,11]                                        ; Konturtabelle 2
DEF REAL ISCOORD[2]                                             ; Variable für Schnittpunkt-
                                                                  Koordinaten.
DEF BOOL ISPOINT                                                ; Variable für
                                                                  Schnittpunktstatus.
DEF INT MODE                                                    ; Variable für Bearbeitungsart.
…
MODE=1                                                          ; Berechnung unabhängig von der
                                                                  aktiven Ebene.
N10 ISPOINT=INTERSEC(TABNAME1[3],TABNAME2[7],ISCOORD,MODE)      ; Aufruf Schnittpunkt der
                                                                  Konturelemente.
N20 IF ISPOINT==FALSE GOTOF KEINSCH                             ; Sprung zu KEINSCH.
…
```

15.5 Konturelemente einer Tabelle satzweise abfahren (EXECTAB)

Funktion

Mit dem Befehl EXECTAB können Sie Konturelemente einer Tabelle, die z. B. mit dem Befehl CONTPRON erzeugt wurde, satzweise abfahren.

Syntax

```
EXECTAB(<Konturtabelle>[<Konturelement>])
```

Bedeutung

EXECTAB	Befehl zum Abfahren eines Konturelements
<Konturtabelle>	Name der Konturtabelle
<Konturelement>	Nummer des Konturelements

Beispiel

Die Konturelemente 0 bis 2 der Tabelle KTAB sollen satzweise abgefahren werden.

Programmcode	**Kommentar**
N10 EXECTAB(KTAB[0])	; Element 0 der Tabelle KTAB verfahren.
N20 EXECTAB(KTAB[1])	; Element 1 der Tabelle KTAB verfahren.
N30 EXECTAB(KTAB[2])	; Element 2 der Tabelle KTAB verfahren.

15.6 Kreisdaten berechnen (CALCDAT)

Funktion

Mit dem Befehl CALCDAT können Sie aus drei oder vier bekannten Kreispunkten den Radius und die Kreismittelpunkt-Koordinaten berechnen. Die angegebenen Punkte müssen unterschiedlich sein. Bei 4 Punkten, die nicht exakt auf dem Kreis liegen, wird für Kreismittelpunkt und Radius ein Mittelwert gewählt.

Syntax

```
<Status>=CALCDAT(<Kreispunkte>[<Anzahl>,<Art>],<Anzahl>,<Ergebnis>)
```

Bedeutung

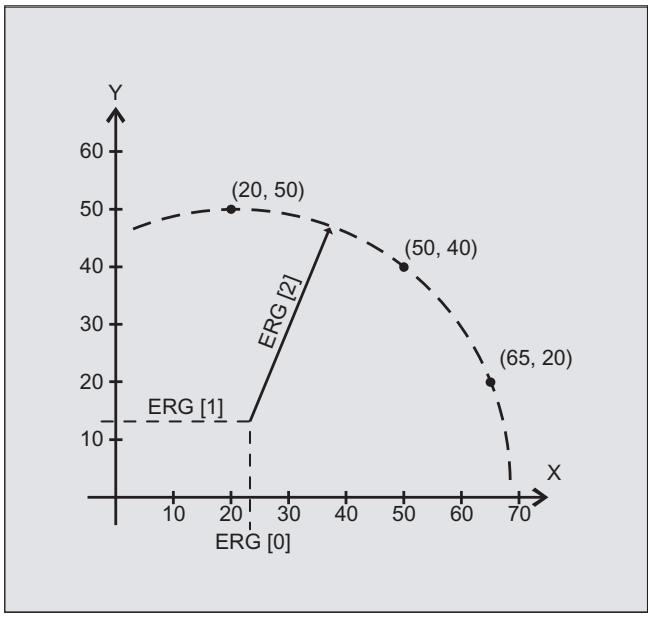
CALCDAT	Befehl zur Berechnung von Radius und Mittelpunkt-Koordinaten eines Kreises aus 3 oder 4 Punkten
<Status>	Variable für den Kreisberechnungsstatus Typ: BOOL Wert: TRUE – Die angegebenen Punkte liegen auf einem Kreis. FALSE – Die angegebenen Punkte liegen **nicht** auf einem Kreis.
<Kreispunkte>[]	Variable zur Angabe der Kreispunkte mit den Parametern: <Anzahl> – Anzahl der Kreispunkte (3 oder 4) <Art> – Art der Koordinatenangabe, z. B. 2 für 2-Punkt-Koordinaten
<Anzahl>	Parameter für die Anzahl der zur Berechnung verwendeten Punkte (3 oder 4)
<Ergebnis>[3]	Variable für Ergebnis: Angabe von Kreismittelpunkt-Koordinaten und Radius 0 Kreismittelpunkt-Koordinate: Abszissenwert 1 Kreismittelpunkt-Koordinate: Ordinatenwert 2 Radius

Hinweis

Beachten Sie, dass die Variablen vor ihrer Verwendung definiert sein müssen.

Beispiel

Von drei Punkten soll ermittelt werden, ob sie auf einem Kreisabschnitt liegen.



Programmcode	**Kommentar**
`N10 DEF REAL PKT[3,2]=(20,50,50,40,65,20)`	`; Variable zur Angabe der Kreispunkte`
`N20 DEF REAL ERG[3]`	`; Variable für Ergebnis`
`N30 DEF BOOL STATUS`	`; Variable für Status`
`N40 STATUS=CALCDAT(PKT,3,ERG)`	`; Aufruf der ermittelten Kreisdaten.`
`N50 IF STATUS == FALSE GOTOF ERROR`	`; Sprung zu Fehler`

15.7 Konturaufbereitung ausschalten (EXECUTE)

Funktion

Mit dem Befehl EXECUTE wird die Konturaufbereitung abgeschaltet und gleichzeitig in den normalen Abarbeitungsmodus zurückgeschaltet.

Syntax

```
EXECUTE(<FEHLER>)
```

Bedeutung

EXECUTE	Befehl zum Beenden der Konturaufbereitung
<FEHLER>	Variable für Fehlerrückmeldung Typ: INT Der Wert der Variablen zeigt an, ob die Kontur fehlerfrei aufbereitet werden konnte: 0 Fehler 1 kein Fehler

Beispiel

Programmcode

```
...
N30 CONTPRON(...)
N40 G1 X... Z...
...
N100 EXECUTE(...)
...
```

Tabellen 16

16.1 Liste der Anweisungen

Legende:

1) Verweis auf das Dokument, das die ausführliche Beschreibung der Anweisung enthält:

PGs/	Programmierhandbuch Grundlagen
PGAs/	Programmierhandbuch Arbeitsvorbereitung
BHDs/	Bedienhandbuch Drehen
BHFs/	Bedienhandbuch Fräsen
FB1 ( )	Funktionshandbuch Grundfunktionen (mit dem alphanumerischen Kürzel der betreffenden Funktionsbeschreibung in Klammern)
FB2 ( )	Funktionshandbuch Erweiterungsfunktionen (mit dem alphanumerischen Kürzel der betreffenden Funktionsbeschreibung in Klammern)
FB3 ( )	Funktionshandbuch Sonderfunktionen (mit dem alphanumerischen Kürzel der betreffenden Funktionsbeschreibung in Klammern)
FBSI/s/	Funktionshandbuch Safety Integrated
FBSY	Funktionshandbuch Synchronaktionen
FBW	Funktionshandbuch Werkzeugverwaltung

2) Wirksamkeit der Anweisung:

m	modal
s	satzweise

3) Verfügbarkeit bei SINUMERIK 828D (D = Drehen, F = Fräsen):

●	Standard
○	Option
-	Nicht verfügbar

4) Standardeinstellung bei Programmanfang (im Auslieferungsstand der Steuerung, wenn nichts anderes programmiert ist).

Anweisung	Bedeutung	Beschreibung siehe 1)	W 2)	828D 3)			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
:	NC-Hauptsatznummer, Sprungmarkenabschluss, Kettungsoperator	*PGAs/* Rechenfunktionen (Seite 61)		●	●	●	●
*	Operator für Multiplikation	*PGAs/* Rechenfunktionen (Seite 61)		●	●	●	●
+	Operator für Addition	*PGAs/* Rechenfunktionen (Seite 61)		●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
-	Operator für Subtraktion	*PGAs*/ Rechenfunktionen (Seite 61)		●	●	●	●
<	Vergleichsoperator, kleiner	*PGAs*/ Rechenfunktionen (Seite 61)		●	●	●	●
<<	Verkettungsoperator für Strings	*PGAs*/ Rechenfunktionen (Seite 61)		●	●	●	●
<=	Vergleichsoperator, kleiner gleich	*PGAs*/ Rechenfunktionen (Seite 61)		●	●	●	●
=	Zuweisungsoperator	*PGAs*/ Rechenfunktionen (Seite 61)		●	●	●	●
>=	Vergleichsoperator, größer gleich	*PGAs*/ Rechenfunktionen (Seite 61)		●	●	●	●
/	Operator für Division	*PGAs*/ Rechenfunktionen (Seite 61)		●	●	●	●
/0 … … /7	Satz wird ausgeblendet (1. Ausblendebene) Satz wird ausgeblendet (8. Ausblendebene)	*PGs*/		● ○	● ○	● ○	● ○
A	Achsname	*PGAs*/ Programmierung der Werkzeugorientierung (A..., B..., C..., LEAD, TILT) (Seite 323)	m/s	●	●	●	●
A2	Werkzeugorientierung: RPY- oder Eulerwinkel	*PGAs*/ Programmierung der Werkzeugorientierung (A..., B..., C..., LEAD, TILT) (Seite 323)	s	●	●	●	●
A3	Werkzeugorientierung: Vektorkomponente Richtung-/Flächennormal	*PGAs*/ Programmierung der Werkzeugorientierung (A..., B..., C..., LEAD, TILT) (Seite 323)	s	●	●	●	●
A4	Werkzeugorientierung: Flächennormalvektor für den Satzanfang	*PGAs*/ Stirnfräsen (3D-Fräsen A4, B4, C4, A5, B5, C5) (Seite 329)	s	●	●	●	●
A5	Werkzeugorientierung: Flächennormalenvektor für das Satzende	*PGAs*/ Stirnfräsen (3D-Fräsen A4, B4, C4, A5, B5, C5) (Seite 329)	s	●	●	●	●
ABS	Absolutwert (Betrag)	*PGAs*/ Rechenfunktionen (Seite 61)		●	●	●	●
AC	absolute Maßangabe von Koordinaten/Positionen	*PGs*/	s	●	●	●	●
ACC	Beeinflussung der aktuellen axialen Beschleunigung	*PGs*/	m	●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
ACCLIMA	Beeinflussung der aktuellen maximalen axialen Beschleunigung	*PGs/*	m	●	●	●	●
ACN	absolute Maßangabe für Rundachsen, Position in negativer Richtung anfahren	*PGs/*	s	●	●	●	●
ACOS	Arcus-Cosinus (Trigon. Funktion)	*PGAs/* Rechenfunktionen (Seite 61)		●	●	●	●
ACP	absolute Maßangabe für Rundachsen, Position in positiver Richtung anfahren	*PGs/*	s	●	●	●	●
ACTBLOCNO	Ausgabe der aktuellen Satznummer eines Alarmsatzes, auch wenn "aktuelle Satzanzeige unterdrückt" (DISPLOF) aktiv ist!	*PGAs/* Aktuelle Satzanzeige unterdrücken (DISPLOF, DISPLON, ACTBLOCNO) (Seite 164)		●	●	●	●
ADDFRAME	Einrechnung und evtl. Aktivierung eines gemessenen Frames	*PGAs/, FB1(K2)* Frame-Berechnung aus 3 Messpunkten im Raum (MEAFRAME) (Seite 298)		●	●	●	●
ADIS	Überschleifabstand für Bahnfunktionen G1, G2, G3, ...	*PGs/*	m	●	●	●	●
ADISPOS	Überschleifabstand für Eilgang G0	*PGs/*	m	●	●	●	●
ADISPOSA	Größe des Toleranzfenster für IPOBRKA	*PGAs/* Programmierbares Bewegungsendekriterium (FINEA, COARSEA, IPOENDA, IPOBRKA, ADISPOSA) (Seite 274)	m	●	●	●	●
ALF	Schnellabhebewinkel	*PGAs/* Schnellabheben von der Kontur (SETINT LIFTFAST, ALF) (Seite 115)	m	●	●	●	●
AMIRROR	Programmierbare Spiegelung	*PGs/*	s	●	●	●	●
AND	Logisches UND	*PGAs/* Vergleichs- und logische Operationen (Seite 64)		●	●	●	●
ANG	Konturzug-Winkel	*PGs/*	s	●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
AP	Polarwinkel	*PGs/*	m/s	•	•	•	•
APR	Zugriffsschutz lesen / anzeigen	*PGAs/* Attribut: Zugriffsrechte (APR, APW, APRP, APWP, APRB, APWB) (Seite 38)		•	•	•	•
APRB	Zugriffsrecht lesen, BTSS	*PGAs/* Attribut: Zugriffsrechte (APR, APW, APRP, APWP, APRB, APWB) (Seite 38)		•	•	•	•
APRP	Zugriffsrecht lesen, Teileprogramm	*PGAs/* Attribut: Zugriffsrechte (APR, APW, APRP, APWP, APRB, APWB) (Seite 38)		•	•	•	•
APW	Zugriffsschutz schreiben	*PGAs/* Attribut: Zugriffsrechte (APR, APW, APRP, APWP, APRB, APWB) (Seite 38)		•	•	•	•
APWB	Zugriffsrecht schreiben, BTSS	*PGAs/* Attribut: Zugriffsrechte (APR, APW, APRP, APWP, APRB, APWB) (Seite 38)		•	•	•	•
APWP	Zugriffsrecht schreiben, Teileprogramm	*PGAs/* Attribut: Zugriffsrechte (APR, APW, APRP, APWP, APRB, APWB) (Seite 38)		•	•	•	•
APX	Definition des Zugriffsschutzes für die Ausführung des angegebenen Sprachelements	*PGAs/* Redefinition von Systemvariablen, Anwendervariablen und NC-Sprachbefehlen (REDEF) (Seite 28)		•	•	•	•
AR	Öffnungswinkel	*PGs/*	m/s	•	•	•	•
AROT	Programmierbare Drehung	*PGs/*	s	•	•	•	•
AROTS	Programmierbare Framedrehungen mit Raumwinkeln	*PGs/*	s	•	•	•	•
AS	Makro-Definition	*PGAs/* Makrotechnik (DEFINE ... AS) (Seite 201)		•	•	•	•
ASCALE	Programmierbare Skalierung	*PGs/*	s	•	•	•	•

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
ASIN	Rechenfunktion, Arcussinus	*PGAs*/ Rechenfunktionen (Seite 61)		●	●	●	●
ASPLINE	Akima-Spline	*PGAs*/ Spline-Interpolation (ASPLINE, BSPLINE, CSPLINE, BAUTO, BNAT, BTAN, EAUTO, ENAT, ETAN, PW, SD, PL) (Seite 233)	m	-	○	-	○
ATAN2	Arcus-Tangens2	*PGAs*/ Rechenfunktionen (Seite 61)		●	●	●	●
ATOL	achsspezifische Toleranz für Kompressor-Funktionen, Orientierungsglättung und Überschleifarten	*PGAs*/ Programmierbare Kontur-/Orientierungtoleranz (CTOL, OTOL, ATOL) (Seite 491)		-	●	-	●
ATRANS	additive programmierbare Verschiebung	*PGs*/	s	●	●	●	●
AX	Variabler Achsbezeichner	*PGAs*/ Achsfunktionen (AXNAME, AX, SPI, AXTOSPI, ISAXIS, AXSTRING, MODAXVAL) (Seite 669)	m/s	●	●	●	●
AXCTSWE	Containerachse weiterschalten	*PGAs*/ Achscontainer (AXCTSWE, AXCTSWED) (Seite 677)		-	-	-	-
AXCTSWED	Achscontainer drehen	*PGAs*/ Achscontainer (AXCTSWE, AXCTSWED) (Seite 677)		-	-	-	-
AXIS	Achsbezeichner, Achsadresse	*PGAs*/ Definition von Anwendervariablen (DEF) (Seite 22)		●	●	●	●
AXNAME	Konvertiert Eingangsstring in Achsbezeichner	*PGAs*/ Achsfunktionen (AXNAME, AX, SPI, AXTOSPI, ISAXIS, AXSTRING, MODAXVAL) (Seite 669)		●	●	●	●
AXSTRING	Konvertiert den String Spindelnummer	*PGAs*/ Achsfunktionen (AXNAME, AX, SPI, AXTOSPI, ISAXIS, AXSTRING, MODAXVAL) (Seite 669)		●	●	●	●
AXTOCHAN	Achse für einen bestimmten Kanal anfordern. Ist vom NC-Programm und aus Synchronaktion möglich.	*PGAs*/ Achse einem anderen Kanal übergeben (AXTOCHAN) (Seite 126)		●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
AXTOSPI	konvertiert Achsbezeichner in einen Spindelindex um	*PGAs*/ Achsfunktionen (AXNAME, AX, SPI, AXTOSPI, ISAXIS, AXSTRING, MODAXVAL) (Seite 669)		●	●	●	●
B	Achsname	*PGAs*/ Programmierung der Werkzeugorientierung (A..., B..., C..., LEAD, TILT) (Seite 323)	m/s	●	●	●	●
B2	Werkzeugorientierung: RPY- oder Eulerwinkel	*PGAs*/ Programmierung der Werkzeugorientierung (A..., B..., C..., LEAD, TILT) (Seite 323)	s	●	●	●	●
B3	Werkzeugorientierung: Vektorkomponente Richtung-/Flächen-normal	*PGAs*/ Programmierung der Werkzeugorientierung (A..., B..., C..., LEAD, TILT) (Seite 323)	s	●	●	●	●
B4	Werkzeugorientierung: Flächennormalvektor für den Satzanfang	*PGAs*/ Stirnfräsen (3D-Fräsen A4, B4, C4, A5, B5, C5) (Seite 329)	s	●	●	●	●
B5	Werkzeugorientierung: Flächennormalenvektor für das Satzende	*PGAs*/ Stirnfräsen (3D-Fräsen A4, B4, C4, A5, B5, C5) (Seite 329)	s	●	●	●	●
B_AND	Bitweises UND	*PGAs*/ Vergleichs- und logische Operationen (Seite 64)		●	●	●	●
B_OR	Bitweises ODER	*PGAs*/ Vergleichs- und logische Operationen (Seite 64)		●	●	●	●
B_NOT	Bitweise Negierung	*PGAs*/ Vergleichs- und logische Operationen (Seite 64)		●	●	●	●
B_XOR	Bitweises Exklusiv-ODER	*PGAs*/ Vergleichs- und logische Operationen (Seite 64)		●	●	●	●
BAUTO	Definieren des ersten Spline-Abschnitts durch die nachfolgenden 3 Punkte	*PGAs*/ Spline-Interpolation (ASPLINE, BSPLINE, CSPLINE, BAUTO, BNAT, BTAN, EAUTO, ENAT, ETAN, PW, SD, PL) (Seite 233)	m	-	○	-	○
BLOCK	Definiert zusammen mit dem Schlüsselwort TO den abzuarbeitenden Programmteil in einem indirekten Unterprogrammlauf	*PGAs*/ Indirekter Unterprogrammaufruf mit Angabe des auszuführenden Programmteils (CALL BLOCK ... TO ...) (Seite 188)		●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
BLSYNC	Bearbeitung der Interruptroutine soll erst mit dem nächsten Satzwechsel beginnen	*PGAs*/ Interruptroutine zuordnen und starten (SETINT, PRIO, BLSYNC) (Seite 111)		●	●	●	●
BNAT [4]	Natürlicher Übergang zum ersten Spline-Satz	*PGAs*/ Spline-Interpolation (ASPLINE, BSPLINE, CSPLINE, BAUTO, BNAT, BTAN, EAUTO, ENAT, ETAN, PW, SD, PL) (Seite 233)	m	-	○	-	○
BOOL	Datentyp: Wahrheitswerte TRUE/FALSE bzw. 1/0	*PGAs*/ Definition von Anwendervariablen (DEF) (Seite 22)		●	●	●	●
BOUND	Prüft, ob Wert innerhalb des definierten Wertebereichs liegt. Gleichheit gibt Prüfwert zurück.	*PGAs*/ Minimum, Maximum und Bereich von Variablen (MINVAL, MAXVAL, BOUND) (Seite 68)		●	●	●	●
BRISK [4]	Sprungförmige Bahnbeschleunigung	*PGs*/	m	●	●	●	●
BRISKA	Sprungförmige Bahnbeschleunigung für die programmierten Achsen einschalten	*PGs*/		●	●	●	●
BSPLINE	B-Spline	*PGAs*/ Spline-Interpolation (ASPLINE, BSPLINE, CSPLINE, BAUTO, BNAT, BTAN, EAUTO, ENAT, ETAN, PW, SD, PL) (Seite 233)	m	-	○	-	○
BTAN	Tangentialer Übergang zum ersten Spline-Satz	*PGAs*/ Spline-Interpolation (ASPLINE, BSPLINE, CSPLINE, BAUTO, BNAT, BTAN, EAUTO, ENAT, ETAN, PW, SD, PL) (Seite 233)	m	-	○	-	○
C	Achsname	*PGAs*/ Programmierung der Werkzeugorientierung (A..., B..., C..., LEAD, TILT) (Seite 323)	m/s	●	●	●	●
C2	Werkzeugorientierung: RPY- oder Eulerwinkel	*PGAs*/ Programmierung der Werkzeugorientierung (A..., B..., C..., LEAD, TILT) (Seite 323)	s	●	●	●	●
C3	Werkzeugorientierung: Vektorkomponente Richtung-/Flächen-normal	*PGAs*/ Programmierung der Werkzeugorientierung (A..., B..., C..., LEAD, TILT) (Seite 323)	s	●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
C4	Werkzeugorientierung: Flächennormalvektor für den Satzanfang	*PGAs*/ Stirnfräsen (3D-Fräsen A4, B4, C4, A5, B5, C5) (Seite 329)	s	●	●	●	●
C5	Werkzeugorientierung: Flächennormalenvektor für das Satzende	*PGAs*/ Stirnfräsen (3D-Fräsen A4, B4, C4, A5, B5, C5) (Seite 329)	s	●	●	●	●
CAC	Absolutes Anfahren einer Position	*PGAs*/ Codierte Positionen anfahren (CAC, CIC, CDC, CACP, CACN) (Seite 231)		●	●	●	●
CACN	In Tabelle abgelegter Wert wird absolut in negativer Richtung angefahren	*PGAs*/ Codierte Positionen anfahren (CAC, CIC, CDC, CACP, CACN) (Seite 231)		●	●	●	●
CACP	In Tabelle abgelegter Wert wird absolut in positiver Richtung angefahren	*PGAs*/ Codierte Positionen anfahren (CAC, CIC, CDC, CACP, CACN) (Seite 231)		●	●	●	●
CALCDAT	Berechnet Radius und Mittelpunkt eines Kreises aus 3 oder 4 Punkten	*PGAs*/ Kreisdaten berechnen (CALCDAT) (Seite 715)		●	●	●	●
CALCPOSI	Überprüfung auf Schutzbereichs-verletzung, Arbeitsfeldbegrenzung und Softwarelimits	*PGAs*/ Überprüfung auf Schutzbereichsverletzung, Arbeitsfeldbegrenzung und Softwarelimits (CALCPOSI) (Seite 223)		●	●	●	●
CALL	Indirekter Unterprogrammaufruf	*PGAs*/ Indirekter Unterprogrammaufruf (CALL) (Seite 187)		●	●	●	●
CALLPATH	Programmierbarer Suchpfad bei Unterprogrammaufrufen	*PGAs*/ Suchpfad bei Unterprogrammaufrufen erweitern (CALLPATH) (Seite 192)		●	●	●	●
CANCEL	Modale Synchronaktion abbrechen	*PGAs*/ Synchronaktion löschen (CANCEL) (Seite 636)		●	●	●	●
CASE	Bedingte Programmverzweigung	*PGAs*/ Programmverzweigung (CASE ... OF ... DEFAULT ...) (Seite 86)		●	●	●	●
CDC	Direktes Anfahren einer Position	*PGAs*/ Codierte Positionen anfahren (CAC, CIC, CDC, CACP, CACN) (Seite 231)		●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
CDOF [4]	Kollisionsüberwachung AUS	*PGs/*	m	●	●	●	●
CDOF2	Kollisionsüberwachung AUS, bei 3D-Umfangsfräsen	*PGs/*	m	●	●	●	●
CDON	Kollisionsüberwachung EIN	*PGs/*	m	●	●	●	●
CFC [4]	Konstanter Vorschub an der Kontur	*PGs/*	m	●	●	●	●
CFIN	Konstanter Vorschub nur bei Innenkrümmung, nicht bei Außenkrümmung	*PGs/*	m	●	●	●	●
CFINE	Zuweisung der Fein-Verschiebung an eine FRAME-Variable	*PGAs/* Grob- und Feinverschiebung (CFINE, CTRANS) (Seite 293)		●	●	●	●
CFTCP	Konstanter Vorschub im Werkzeugschneiden-Bezugspunkt, Mittelpunktsbahn	*PGs/*	m	●	●	●	●
CHAN	Spezifizierung des Gültigkeitsbereichs von Daten	*PGAs/* Definition von Anwendervariablen (DEF) (Seite 22)		●	●	●	●
CHANDATA	Kanalnummer für Kanaldatenzugriffe einstellen	*PGAs/* Arbeitsspeicher (CHANDATA, COMPLETE, INITIAL) (Seite 210)		●	●	●	●
CHAR	Datentyp: ASCII-Zeichen	*PGAs/* Definition von Anwendervariablen (DEF) (Seite 22)		●	●	●	●
CHECKSUM	Bildet die Checksumme über ein Feld als STRING mit einer festgesetzten Länge	*PGAs/* Checksummenberechnung über ein Feld (CHECKSUM) (Seite 142)		●	●	●	●
CHF	Fase; Wert = Länge der Fase	*PGs/*	s	●	●	●	●
CHKDM	Prüfung der Eindeutigkeit innerhalb eines Magazins	*FBW*		●	●	●	●
CHKDNO	Eindeutigkeitsprüfung der D-Nummern	*PGAs/* Freie D-Nummernvergabe: D-Nummern prüfen (CHKDNO) (Seite 430)		●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
CHR	Fase; Wert = Länge der Fase in Bewegungsrichtung	*PGsl*		●	●	●	●
CIC	Inkrementelles Anfahren einer Position	*PGAsl* Codierte Positionen anfahren (CAC, CIC, CDC, CACP, CACN) (Seite 231)		●	●	●	●
CIP	Kreisinterpolation über Zwischenpunkt	*PGsl*	m	●	●	●	●
CLEARM	Rücksetzen einer/mehrerer Marken für Kanalkoordinierung	*PGAsl* Programmkoordinierung (INIT, START, WAITM, WAITMC, WAITE, SETM, CLEARM) (Seite 103)		-	-	-	-
CLRINT	Interrupt abwählen	*PGAsl* Zuordnung einer Interruptroutine löschen (CLRINT) (Seite 114)		●	●	●	●
CMIRROR	Spiegeln an einer Koordinatenachse	*PGAsl* Rechenfunktionen (Seite 61)		●	●	●	●
COARSEA	Bewegungsende beim Erreichen von "Genauhalt Grob"	*PGAsl* Programmierbares Bewegungsendekriterium (FINEA, COARSEA, IPOENDA, IPOBRKA, ADISPOSA) (Seite 274)	m	●	●	●	●
COMPCAD	Kompressor EIN: Optimierte Oberflächengüte bei CAD-Programmen	*PGAsl* NC-Satz-Kompression (COMPON, COMPCURV, COMPCAD, COMPOF) (Seite 247)	m	-	○	-	○
COMPCURV	Kompressor EIN: krümmungsstetige Polynome	*PGAsl* NC-Satz-Kompression (COMPON, COMPCURV, COMPCAD, COMPOF) (Seite 247)	m	-	○	-	○
COMPLETE	Steueranweisung für das Aus- und Einlesen von Daten	PGAsl Arbeitsspeicher (CHANDATA, COMPLETE, INITIAL) (Seite 210)		●	●	●	●
COMPOF [4]	Kompressor AUS	*PGAsl* NC-Satz-Kompression (COMPON, COMPCURV, COMPCAD, COMPOF) (Seite 247)	m	-	○	-	○

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
COMPON	Kompressor EIN	*PGAs/* NC-Satz-Kompression (COMPON, COMPCURV, COMPCAD, COMPOF) (Seite 247)		-	○	-	○
CONTDCON	Konturdecodierung in Tabellenform EIN	*PGAs/* Codierte Konturtabelle erstellen (CONTDCON) (Seite 708)		●	●	●	●
CONTPRON	Referenzaufbereitung einschalten	*PGAs/* Konturtabelle erstellen (CONTPRON) (Seite 702)		●	●	●	●
CORROF	Alle aktiven überlagerten Bewegungen werden abgewählt.	*PGs/*		●	●	●	●
COS	Cosinus (Trigon. Funktion)	*PGAs/* Rechenfunktionen (Seite 61)		●	●	●	●
COUPDEF	Definition ELG-Verband / Synchronspindel-Verband	*PGAs/* Synchronspindel: Programmierung (COUPDEF, COUPDEL, COUPON, COUPONC, COUPOF, COUPOFS, COUPRES, WAITC) (Seite 535)		○	-	○	-
COUPDEL	ELG-Verband löschen	*PGAs/* Synchronspindel: Programmierung (COUPDEF, COUPDEL, COUPON, COUPONC, COUPOF, COUPOFS, COUPRES, WAITC) (Seite 535)		○	-	○	-
COUPOF	ELG-Verband / Synchronspindelpaar EIN	*PGAs/* Synchronspindel: Programmierung (COUPDEF, COUPDEL, COUPON, COUPONC, COUPOF, COUPOFS, COUPRES, WAITC) (Seite 535)		○	-	○	-
COUPOFS	Ausschalten ELG-Verband / Synchronspindelpaar mit Stopp der Folgespindel	*PGAs/* Synchronspindel: Programmierung (COUPDEF, COUPDEL, COUPON, COUPONC, COUPOF, COUPOFS, COUPRES, WAITC) (Seite 535)		○	-	○	-

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
COUPON	ELG-Verband / Synchronspindelpaar EIN	*PGAs/* Synchronspindel: Programmierung (COUPDEF, COUPDEL, COUPON, COUPONC, COUPOF, COUPOFS, COUPRES, WAITC) (Seite 535)		○	-	○	-
COUPONC	Einschalten ELG-Verband / Synchronspindelpaar mit vorhergehender Programmierung übernehmen	*PGAs/* Synchronspindel: Programmierung (COUPDEF, COUPDEL, COUPON, COUPONC, COUPOF, COUPOFS, COUPRES, WAITC) (Seite 535)		○	-	○	-
COUPRES	ELG-Verband rücksetzen	*PGAs/* Synchronspindel: Programmierung (COUPDEF, COUPDEL, COUPON, COUPONC, COUPOF, COUPOFS, COUPRES, WAITC) (Seite 535)		○	-	○	-
CP	Bahnbewegung	*PGAs/* Kartesisches PTP-Fahren (Seite 376)	m	●	●	●	●
CPRECOF [4]	Programmierbare Konturgenauigkeit AUS	*PGs/*	m	●	●	●	●
CPRECON	Programmierbare Konturgenauigkeit EIN	*PGs/*	m	●	●	●	●
CPROT	Kanalspezifischer Schutzbereich EIN/AUS	*PGAs/* Schutzbereiche aktivieren/deaktivieren (CPROT, NPROT) (Seite 219)		●	●	●	●
CPROTDEF	Definition eines kanalspezifischen Schutzbereichs	*PGAs/* Festlegung der Schutzbereiche (CPROTDEF, NPROTDEF) (Seite 215)		●	●	●	●
CR	Kreisradius	*PGs/*	s	●	●	●	●
CROT	Drehung des aktuellen Koordinatensystems	*PGAs/* Rechenfunktionen (Seite 61)		●	●	●	●
CROTS	Programmierbare Framedrehungen mit Raumwinkeln (Drehung in den angegebenen Achsen)	*PGs/*	s	●	●	●	●
CRPL	Frame-Drehung in einer beliebigen Ebene	*FB1(K2)*		●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
CSCALE	Maßstabsfaktor für mehrere Achsen	*PGAs*/ Rechenfunktionen (Seite 61)		●	●	●	●
CSPLINE	Kubischer Spline	*PGAs*/ Spline-Interpolation (ASPLINE, BSPLINE, CSPLINE, BAUTO, BNAT, BTAN, EAUTO, ENAT, ETAN, PW, SD, PL) (Seite 233)	m	-	○	-	○
CT	Kreis mit tangentialem Übergang	*PGs*/	m	●	●	●	●
CTAB	Ermittle Folgeachsposition anhand der Leitachsposition aus Kurventabelle	*PGAs*/ Kurventabellenwerte lesen (CTABTSV, CTABTEV, CTABTSP, CTABTEP, CTABSSV, CTABSEV, CTAB, CTABINV, CTABTMIN, CTABTMAX) (Seite 513)		-	-	-	-
CTABDEF	Tabellendefinition EIN	*PGAs*/ Kurventabellen definieren (CTABDEF, CATBEND) (Seite 502)		-	-	-	-
CTABDEL	Kurventabelle löschen	*PGAs*/ Kurventabellen löschen (CTABDEL) (Seite 508)		-	-	-	-
CTABEND	Tabellendefinition AUS	*PGAs*/ Kurventabellen definieren (CTABDEF, CATBEND) (Seite 502)		-	-	-	-
CTABEXISTS	Prüft die Kurventabelle mit der Nummer n	*PGAs*/ Vorhandensein einer Kurventabelle prüfen (CTABEXISTS) (Seite 508)		-	-	-	-
CTABFNO	Anzahl der noch möglichen Kurventabellen im Speicher	*PGAs*/ Kurventabellen: Ressourcennutzung prüfen (CTABNO, CTABNOMEM, CTABFNO, CTABSEGID, CTABSEG, CTABFSEG, CTABMSEG, CTABPOLID, CTABPOL, CTABFPOL, CTABMPOL) (Seite 518)		-	-	-	-

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
CTABFPOL	Anzahl der noch möglichen Polynome im Speicher	*PGAs*/ Kurventabellen: Ressourcennutzung prüfen (CTABNO, CTABNOMEM, CTABFNO, CTABSEGID, CTABSEG, CTABFSEG, CTABMSEG, CTABPOLID, CTABPOL, CTABFPOL, CTABMPOL) (Seite 518)		-	-	-	-
CTABFSEG	Anzahl der noch möglichen Kurvensegmente im Speicher	*PGAs*/ Kurventabellen: Ressourcennutzung prüfen (CTABNO, CTABNOMEM, CTABFNO, CTABSEGID, CTABSEG, CTABFSEG, CTABMSEG, CTABPOLID, CTABPOL, CTABFPOL, CTABMPOL) (Seite 518)		-	-	-	-
CTABID	Liefert Tabellen-Nummer der n-ten Kurventabelle	*PGAs*/ Kurventabellen: Tabelleneigenschaften ermitteln (CTABID, CTABISLOCK, CTABMEMTYP, CTABPERIOD) (Seite 511)		-	-	-	-
CTABINV	Ermittle Leitachsposition anhand der Folgeachsposition aus Kurventabelle	*PGAs*/ Kurventabellenwerte lesen (CTABTSV, CTABTEV, CTABTSP, CTABTEP, CTABSSV, CTABSEV, CTAB, CTABINV, CTABTMIN, CTABTMAX) (Seite 513)		-	-	-	-
CTABISLOCK	Gibt den Sperrzustand der Kurventabelle mit der Nummer n zurück	*PGAs*/ Kurventabellen: Tabelleneigenschaften ermitteln (CTABID, CTABISLOCK, CTABMEMTYP, CTABPERIOD) (Seite 511)		-	-	-	-
CTABLOCK	Löschen und Über-schreiben, sperren	*PGAs*/ Kurventabellen gegen Löschen und Überschreiben sperren (CTABLOCK, CTABUNLOCK) (Seite 510)		-	-	-	-
CTABMEMTYP	Gibt den Speicher zurück, in dem die Kurventabelle mit der Nummer n angelegt ist.	*PGAs*/ Kurventabellen: Tabelleneigenschaften ermitteln (CTABID, CTABISLOCK, CTABMEMTYP, CTABPERIOD) (Seite 511)		-	-	-	-

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
CTABMPOL	Anzahl der maximal möglichen Polynome im Speicher	*PGAs*/ Kurventabellen: Ressourcennutzung prüfen (CTABNO, CTABNOMEM, CTABFNO, CTABSEGID, CTABSEG, CTABFSEG, CTABMSEG, CTABPOLID, CTABPOL, CTABFPOL, CTABMPOL) (Seite 518)		-	-	-	-
CTABMSEG	Anzahl der maximal möglichen Kurvensegmente im Speicher	*PGAs*/ Kurventabellen: Ressourcennutzung prüfen (CTABNO, CTABNOMEM, CTABFNO, CTABSEGID, CTABSEG, CTABFSEG, CTABMSEG, CTABPOLID, CTABPOL, CTABFPOL, CTABMPOL) (Seite 518)		-	-	-	-
CTABNO	Anzahl der definierten Kurventabellen im SRAM oder DRAM	*FB3(M3)*		-	-	-	-
CTABNOMEM	Anzahl der definierten Kurventabellen im SRAM oder DRAM	*PGAs*/ Kurventabellen: Ressourcennutzung prüfen (CTABNO, CTABNOMEM, CTABFNO, CTABSEGID, CTABSEG, CTABFSEG, CTABMSEG, CTABPOLID, CTABPOL, CTABFPOL, CTABMPOL) (Seite 518)		-	-	-	-
CTABPERIOD	Gibt die Tabellenperiodizität der Kurventabelle mit der Nummer n zurück	*PGAs*/ Kurventabellen: Tabelleneigenschaften ermitteln (CTABID, CTABISLOCK, CTABMEMTYP, CTABPERIOD) (Seite 511)		-	-	-	-
CTABPOL	Anzahl der bereits verwendeten Polynome im Speicher	*PGAs*/ Kurventabellen: Ressourcennutzung prüfen (CTABNO, CTABNOMEM, CTABFNO, CTABSEGID, CTABSEG, CTABFSEG, CTABMSEG, CTABPOLID, CTABPOL, CTABFPOL, CTABMPOL) (Seite 518)		-	-	-	-

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
CTABPOLID	Anzahl der von der Kurventabelle mit der Nummer n verwendeten Kurvenpolynome	*PGAs*/ Kurventabellen: Ressourcennutzung prüfen (CTABNO, CTABNOMEM, CTABFNO, CTABSEGID, CTABSEG, CTABFSEG, CTABMSEG, CTABPOLID, CTABPOL, CTABFPOL, CTABMPOL) (Seite 518)		-	-	-	-
CTABSEG	Anzahl der bereits verwendeten Kurvensegmente im Speicher	*PGAs*/ Kurventabellen: Ressourcennutzung prüfen (CTABNO, CTABNOMEM, CTABFNO, CTABSEGID, CTABSEG, CTABFSEG, CTABMSEG, CTABPOLID, CTABPOL, CTABFPOL, CTABMPOL) (Seite 518)		-	-	-	-
CTABSEGID	Anzahl der von der Kurventabelle mit der Nummer n verwendeten Kurvensegmente	*PGAs*/ Kurventabellen: Ressourcennutzung prüfen (CTABNO, CTABNOMEM, CTABFNO, CTABSEGID, CTABSEG, CTABFSEG, CTABMSEG, CTABPOLID, CTABPOL, CTABFPOL, CTABMPOL) (Seite 518)		-	-	-	-
CTABSEV	Liefert den Endwert der Folgeachse eines Segments der Kurventabelle	*PGAs*/ Kurventabellenwerte lesen (CTABTSV, CTABTEV, CTABTSP, CTABTEP, CTABSSV, CTABSEV, CTAB, CTABINV, CTABTMIN, CTABTMAX) (Seite 513)		-	-	-	-
CTABSSV	Liefert den Startwert der Folgeachse eines Segments der Kurventabelle	*PGAs*/ Kurventabellenwerte lesen (CTABTSV, CTABTEV, CTABTSP, CTABTEP, CTABSSV, CTABSEV, CTAB, CTABINV, CTABTMIN, CTABTMAX) (Seite 513)		-	-	-	-
CTABTEP	Liefert den Wert der Leitachse am Kurventabellen-Ende	*PGAs*/ Kurventabellenwerte lesen (CTABTSV, CTABTEV, CTABTSP, CTABTEP, CTABSSV, CTABSEV, CTAB, CTABINV, CTABTMIN, CTABTMAX) (Seite 513)		-	-	-	-

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
CTABTEV	Liefert den Wert der Folgeachse am Kurventabellen-Ende	*PGAs*/ Kurventabellenwerte lesen (CTABTSV, CTABTEV, CTABTSP, CTABTEP, CTABSSV, CTABSEV, CTAB, CTABINV, CTABTMIN, CTABTMAX) (Seite 513)		-	-	-	-
CTABTMAX	Liefert Maximalwert der Folgeachse der Kurventabelle	*PGAs*/ Kurventabellenwerte lesen (CTABTSV, CTABTEV, CTABTSP, CTABTEP, CTABSSV, CTABSEV, CTAB, CTABINV, CTABTMIN, CTABTMAX) (Seite 513)		-	-	-	-
CTABTMIN	Liefert Minimalwert der Folgeachse der Kurventabelle	*PGAs*/ Kurventabellenwerte lesen (CTABTSV, CTABTEV, CTABTSP, CTABTEP, CTABSSV, CTABSEV, CTAB, CTABINV, CTABTMIN, CTABTMAX) (Seite 513)		-	-	-	-
CTABTSP	Liefert den Wert der Leitachse am Kurventabellen-Anfang	*PGAs*/ Kurventabellenwerte lesen (CTABTSV, CTABTEV, CTABTSP, CTABTEP, CTABSSV, CTABSEV, CTAB, CTABINV, CTABTMIN, CTABTMAX) (Seite 513)		-	-	-	-
CTABTSV	Liefert den Wert der Folgeachse am Kurventabellen-Anfang	*PGAs*/ Kurventabellenwerte lesen (CTABTSV, CTABTEV, CTABTSP, CTABTEP, CTABSSV, CTABSEV, CTAB, CTABINV, CTABTMIN, CTABTMAX) (Seite 513)		-	-	-	-
CTABUNLOCK	Aufheben der Lösch- und Überschreibsperre	*PGAs*/ Kurventabellen gegen Löschen und Überschreiben sperren (CTABLOCK, CTABUNLOCK) (Seite 510)		-	-	-	-
CTOL	Konturtoleranz für Kompressor-Funktionen, Orientierungsglättung und Überschleifarten	*PGAs*/ Programmierbare Kontur-/Orientierungtoleranz (CTOL, OTOL, ATOL) (Seite 491)		-	○	-	○
CTRANS	Nullpunktverschiebung für mehrere Achsen	*PGAs*/ Grob- und Feinverschiebung (CFINE, CTRANS) (Seite 293)		●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
CUT2D [4]	2D-Werkzeugkorrektur	*PGs/*	m	●	●	●	●
CUT2DF	2D-Werkzeugkorrektur Die Werkzeugkorrektur wirkt relativ zum aktuellen Frame (schräge Ebene).	*PGs/*	m	●	●	●	●
CUT3DC	3D-Werkzeugkorrektur Umfangsfräsen	*PGAs/* Aktivierung 3D-Werkzeugkorrekturen (CUT3DC..., CUT3DF...) (Seite 409)	m	-	-	-	-
CUT3DCC	3D-Werkzeugkorrektur Umfangsfräsen mit Begrenzungsflächen	*PGAs/* 3D-Werkzeugkorrektur: Berücksichtigung einer Begrenzungsfläche (CUT3DCC, CUT3DCCD) (Seite 419)	m	-	-	-	-
CUT3DCCD	3D-Werkzeugkorrektur Umfangsfräsen mit Begrenzungsflächen mit Differenzwerkzeug	*PGAs/* 3D-Werkzeugkorrektur: Berücksichtigung einer Begrenzungsfläche (CUT3DCC, CUT3DCCD) (Seite 419)	m	-	-	-	-
CUT3DF	3D-Werkzeugkorrektur Stirnfräsen	*PGAs/* Aktivierung 3D-Werkzeugkorrekturen (CUT3DC..., CUT3DF...) (Seite 409)	m	-	-	-	-
CUT3DFF	3D-Werkzeugkorrektur Stirnfräsen mit konstanter Werkzeugorientierung abhängig vom aktiven Frame	*PGAs/* Aktivierung 3D-Werkzeugkorrekturen (CUT3DC..., CUT3DF...) (Seite 409)	m	-	-	-	-
CUT3DFS	3D-Werkzeugkorrektur Stirnfräsen mit konstanter Werkzeugorientierung unabhängig vom aktiven Frame	*PGAs/* Aktivierung 3D-Werkzeugkorrekturen (CUT3DC..., CUT3DF...) (Seite 409)	m	-	-	-	-
CUTCONOF [4]	Konstante Radiuskorrektur AUS	*PGs/*	m	●	●	●	●
CUTCONON	Konstante Radiuskorrektur EIN	*PGs/*	m	●	●	●	●
CUTMOD	Funktion "Modifikation der Korrekturdaten bei drehbaren Werkzeugen" einschalten	*PGAs/* Schneidendaten-Modifikation bei drehbaren Werkzeugen (CUTMOD) (Seite 446)		●	●	●	●
CYCLE...	Messzyklen	*BHDs/BHFs/*					

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
D	Werkzeugkorrektur-nummer	*PGs/*		●	●	●	●
D0	Bei D0 sind die Korrekturen für das Werkzeug unwirksam	*PGs/*		●	●	●	●
DAC	Absolut satzweise achsspezifische Durchmesser-programmierung	*PGs/*	s	●	●	●	●
DC	Absolute Maßangabe für Rundachsen, Position direkt anfahren	*PGs/*	s	●	●	●	●
DEF	Variablendefinition	*PGAs/* Definition von Anwendervariablen (DEF) (Seite 22)		●	●	●	●
DEFINE	Schlüsselwort für Makrodefinitionen	*PGAs/* Makrotechnik (DEFINE ... AS) (Seite 201)		●	●	●	●
DEFAULT	Zweig in der CASE-Verzweigung	*PGAs/* Programmverzweigung (CASE ... OF ... DEFAULT ...) (Seite 86)		●	●	●	●
DELAYFSTON	Beginn eines Stopp-Delay-Bereichs definieren	*PGAs/* Programmverzweigung (CASE ... OF ... DEFAULT ...) (Seite 86)	m	●	●	●	●
DELAYFSTOF	Ende eines Stopp-Delay-Bereichs definieren	*PGAs/* Bedingt unterbrechbare Programmabschnitte (DELAYFSTON, DELAYFSTOF) (Seite 468)	m	●	●	●	●
DELDL	Additive Korrekturen löschen	*PGAs/* Additive Korrekturen löschen (DELDL) (Seite 395)		●	●	●	●
DELDTG	Restweglöschen	*PGAs/* Restweglöschen (DELDTG) (Seite 582)		●	●	●	●
DELETE	Die angegebene Datei löschen. Der Dateiname kann mit Pfad und Datei-Kennung angegeben werden.	*PGAs/* Datei löschen (DELETE) (Seite 132)		●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
DELTOOLENV	Datensätze zur Beschreibung von Werkzeugumgebungen löschen	*FB1(W1)*		•	•	•	•
DIACYCOFA	Achsspezifische modale Durchmesser-programmierung: AUS in Zyklen	*FB1(P1)*	m	•	•	•	•
DIAM90	Durchmesserprogrammierung für G90, Radiusprogrammierung für G91	*PGAs/*	m	•	•	•	•
DIAM90A	Achsspezifische modale Durchmesser-programmierung für G90 und AC, Radiusprogrammierung für G91 und IC	*PGs/*	m	•	•	•	•
DIAMCHAN	Übernahme aller Achsen aus MD Achsfunktionen in den Kanalzustand der Durchmesser-programmierung	*PGs/*		•	•	•	•
DIAMCHANA	Übernahme Kanalzustand der Durchmesser-programmierung	*PGs/*		•	•	•	•
DIAMCYCOF	Kanalspezifische Durchmesser-programmierung: AUS in Zyklen	*FB1(P1)*	m	•	•	•	•
DIAMOF [4]	Durchmesser-programmierung: AUS Grundstellung siehe Maschinenhersteller	*PGs/*	m	•	•	•	•
DIAMOFA	Achsspezifische modale Durchmesser-programmierung: AUS Grundstellung siehe Maschinenhersteller	*PGs/*	m	•	•	•	•
DIAMON	Durchmesser-programmierung: EIN	*PGs/*	m	•	•	•	•
DIAMONA	Achsspezifische modale Durchmesser-programmierung: EIN Freischaltung siehe Maschinenhersteller	*PGs/*	m	•	•	•	•

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
DIC	Relativ satzweise achsspezifische Durchmesser-programmierung	*PGs/*	s	●	●	●	●
DILF	Rückzugsweg (Länge)	*PGs/*	m	●	●	●	●
DISABLE	Interrupt AUS	*PGAs/* Zuordnung einer Interruptroutine deaktivieren/reaktivieren (DISABLE, ENABLE) (Seite 113)		●	●	●	●
DISC	Überhöhung Übergangskreis Werkzeug-Radiuskorrektur	*PGs/*	m	●	●	●	●
DISCL	Abstand des Endpunkts der schnellen Zustellbe-wegung, von der Bearbeitungsebene	*PGs/*		●	●	●	●
DISPLOF	Aktuelle Satzanzeige unterdrücken	*PGAs/* Aktuelle Satzanzeige unterdrücken (DISPLOF, DISPLON, ACTBLOCNO) (Seite 164)		●	●	●	●
DISPLON	Unterdrückung der aktuellen Satzanzeige aufheben	*PGAs/* Aktuelle Satzanzeige unterdrücken (DISPLOF, DISPLON, ACTBLOCNO) (Seite 164)		●	●	●	●
DISPR	Repos-Bahndifferenz	*PGAs/* Wiederanfahren an Kontur (REPOSA, REPOSL, REPOSQ, REPOSQA, REPOSH, REPOSHA, DISR, DISPR, RMI, RMB, RME, RMN) (Seite 476)	s	●	●	●	●
DISR	Repos-Abstand	*PGAs/* Wiederanfahren an Kontur (REPOSA, REPOSL, REPOSQ, REPOSQA, REPOSH, REPOSHA, DISR, DISPR, RMI, RMB, RME, RMN) (Seite 476)	s	●	●	●	●
DITE	Gewindeauslaufweg	*PGs/*	m	●	●	●	●
DITS	Gewindeeinlaufweg	*PGs/*	m	●	●	●	●
DIV	Integer-Division	*PGAs/* Rechenfunktionen (Seite 61)		●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
DL	Ortsabhängige additive Werkzeugkorrektur anwählen (DL, Summen-Einrichtekorrektur)	*PGAs*/ Additive Korrekturen anwählen (DL) (Seite 392)	m	-	-	-	-
DO	Schlüsselwort für Synchronaktion, löst bei erfüllter Bedingung Aktion aus	*PGAs*/ Aktionen (DO) (Seite 557)		●	●	●	●
DRFOF	Ausschalten der Handradverschiebungen (DRF)	*PGs*/	m	●	●	●	●
DRIVE	Geschwindigkeits-abhängige Bahnbeschleunigung	*PGs*/	m	●	●	●	●
DRIVEA	Geknickte Beschleunigungs-kennlinie für die programmierten Achsen einschalten	*PGs*/		●	●	●	●
DYNFINISH	Dynamik für Feinschlichten	*PGs*/	m	●	●	●	●
DYNNORM	Normale Dynamik	*PGs*/	m	●	●	●	●
DYNPOS	Dynamik für Positionierbetrieb, Gewindebohren	*PGs*/	m	●	●	●	●
DYNROUGH	Dynamik für Schruppen	*PGs*/	m	●	●	●	●
DYNSEMIFIN	Dynamik für Schlichten	*PGs*/	m	●	●	●	●
DZERO	Kennzeichnet alle D-Nummern der TO-Einheit als ungültig	*PGAs*/ Freie D-Nummernvergabe: D-Nummern ungültig setzen (DZERO) (Seite 433)		●	●	●	●
EAUTO	Festlegung des letzten Spline-Abschnitts durch die letzten 3 Punkte	*PGAs*/ Spline-Interpolation (ASPLINE, BSPLINE, CSPLINE, BAUTO, BNAT, BTAN, EAUTO, ENAT, ETAN, PW, SD, PL) (Seite 233)	m	-	○	-	○
EGDEF	Definition eines elektronischen Getriebes	*PGAs*/ Elektronisches Getriebe definieren (EGDEF) (Seite 526)		-	-	-	-
EGDEL	Kopplungsdefinition für die Folgeachse löschen	*PGAs*/ Definition eines Elektronischen Getriebes löschen (EGDEL) (Seite 532)		-	-	-	-

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
EGOFC	Elektronisches Getriebe kontinuierlich ausschalten	*PGAs*/ Elektronisches Getriebe ausschalten (EGOFS, EGOFC) (Seite 531)		-	-	-	-
EGOFS	Elektronisches Getriebe selektiv ausschalten	*PGAs*/ Elektronisches Getriebe ausschalten (EGOFS, EGOFC) (Seite 531)		-	-	-	-
EGON	Elektronisches Getriebe einschalten	*PGAs*/ Elektronisches Getriebe ausschalten (EGOFS, EGOFC) (Seite 531)		-	-	-	-
EGONSYN	Elektronisches Getriebe einschalten	*PGAs*/ Elektronisches Getriebe ausschalten (EGOFS, EGOFC) (Seite 531)		-	-	-	-
EGONSYNE	Elektronisches Getriebe einschalten, mit Vorgabe von Anfahrmodus	*PGAs*/ Elektronisches Getriebe ausschalten (EGOFS, EGOFC) (Seite 531)		-	-	-	-
ELSE	Programmverzweigung, wenn IF-Bedingung nicht erfüllt	*PGAs*/ Programmschleife mit Alternative (IF, ELSE, ENDIF) (Seite 96)		●	●	●	●
ENABLE	Interrupt EIN	*PGAs*/ Zuordnung einer Interruptroutine deaktivieren/reaktivieren (DISABLE, ENABLE) (Seite 113)		●	●	●	●
ENAT [4]	Natürlicher Kurvenübergang zum nächsten Verfahrsatz	*PGAs*/ Spline-Interpolation (ASPLINE, BSPLINE, CSPLINE, BAUTO, BNAT, BTAN, EAUTO, ENAT, ETAN, PW, SD, PL) (Seite 233)	m	-	○	-	○
ENDFOR	Endezeile der FOR-Zählschleife	*PGAs*/ Zählschleife (FOR ... TO ..., ENDFOR) (Seite 99)		●	●	●	●
ENDIF	Endezeile der IF-Verzweigung	*PGAs*/ Programmschleife mit Alternative (IF, ELSE, ENDIF) (Seite 96)		●	●	●	●
ENDLABEL	Endmarke für Teilprogramm-wiederholungen über REPEAT	*PGAs*/, *FB1(K1)* Programmteilwiederholung (REPEAT, REPEATB, ENDLABEL, P) (Seite 88)		●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
ENDLOOP	Endezeile der Endlos-Programmschleife LOOP	*PGAs*/ Endlos-Programmschleife (LOOP, ENDLOOP) (Seite 98)		●	●	●	●
ENDPROC	Endezeile eines Programms mit der Anfangszeile PROC			●	●	●	●
ENDWHILE	Endezeile der WHILE-Schleife	*PGAs*/ Programmschleife mit Bedingung am Schleifenanfang (WHILE, ENDWHILE) (Seite 100)		●	●	●	●
ETAN	Tangentialer Kurvenübergang zum nächsten Verfahrsatz bei Spline-Beginn	*PGAs*/ Spline-Interpolation (ASPLINE, BSPLINE, CSPLINE, BAUTO, BNAT, BTAN, EAUTO, ENAT, ETAN, PW, SD, PL) (Seite 233)	m	-	○	-	○
EVERY	Synchronaktion ausführen bei Übergang der Bedingung von FALSE zu TRUE	*PGAs*/ Zyklische Prüfung der Bedingung (WHEN, WHENEVER, FROM, EVERY) (Seite 555)		●	●	●	●
EX	Schlüsselwert für die Wertzuweisung in exponentieller Schreibweise	*PGAs*/ Vordefinierte Anwendervariablen: Rechenparameter (R) (Seite 18)		●	●	●	●
EXECSTRING	Übergabe einer String-Variablen mit der auszuführenden Teileprogrammzeile	*PGAs*/ Indirekte Programmierung von Teileprogrammzeilen (EXECSTRING) (Seite 60)		●	●	●	●
EXECTAB	Ein Element aus einer Bewegungstabelle abarbeiten	*PGAs*/ Indirekte Programmierung von Teileprogrammzeilen (EXECSTRING) (Seite 60)		●	●	●	●
EXECUTE	Programmausführung EIN	*PGAs*/ Konturaufbereitung ausschalten (EXECUTE) (Seite 717)		●	●	●	●
EXP	Exponentialfunktion ex	*PGAs*/ Rechenfunktionen (Seite 61)		●	●	●	●
EXTCALL	Externes Unterprogramm abarbeiten	*PGAs*/ Externes Unterprogramm abarbeiten (EXTCALL) (Seite 193)		●	●	●	●
EXTERN	Bekanntmachung eines Unterprogramms mit Parameterübergabe	*PGAs*/ Unterprogrammaufruf ohne Parameterübergabe (Seite 178)		●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
F	Vorschubwert (in Verbindung mit G4 wird mit F auch die Verweilzeit programmiert)	*PGs/*		●	●	●	●
FA	Axialer Vorschub	*PGs/*	m	●	●	●	●
FAD	Zustell-Vorschub für Weiches An- und Abfahren	*PGs/*		●	●	●	●
FALSE	Logische Konstante: falsch	*PGAs/* Definition von Anwendervariablen (DEF) (Seite 22)		●	●	●	●
FB	Satzweiser Vorschub	*PGs/*		●	●	●	●
FCTDEF	Polynomfunktion definieren	*PGAs/* Online-Werkzeugkorrektur (PUTFTOCF, FCTDEF, PUTFTOC, FTOCON, FTOCOF) (Seite 404)		-	-	-	-
FCUB	Vorschub nach kubischem Spline veränderlich	*PGAs/* Vorschubverlauf (FNORM, FLIN, FCUB, FPO) (Seite 460)	m	●	●	●	●
FD	Bahnvorschub für Handradüberlagerung	*PGs/*	s	●	●	●	●
FDA	Axialer Vorschub für Handradüberlagerung	*PGs/*	s	●	●	●	●
FENDNORM	Eckenverzögerung AUS	*PGAs/* Vorschubreduzierung mit Eckenverzögerung (FENDNORM, G62, G621) (Seite 273)	m	●	●	●	●
FFWOF [4]	Vorsteuerung AUS	*PGs/*	m	●	●	●	●
FFWON	Vorsteuerung Ein	*PGs/*	m	●	●	●	●
FGREF	Bezugsradius bei Rundachsen oder Bahnbezugsfaktoren bei Orientierungsachsen (Vektorinterpolation)	*PGs/*	m	●	●	●	●
FGROUP	Festlegung der Achse(n) mit Bahnvorschub	*PGs/*		●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
FI	Parameter für Zugriff auf Framedaten: Feinverschiebung	*PGAs*/ Framekomponenten lesen und verändern (TR, FI, RT, SC, MI) (Seite 289)		•	•	•	•
FIFOCTRL	Steuerung des Vorlaufpuffers	*PGAs*/ Programmablauf mit Vorlaufspeicher (STOPFIFO, STARTFIFO, FIFOCTRL, STOPRE) (Seite 465)	m	•	•	•	•
FILEDATE	Liefert Datum des zuletzt schreibenden Zugriffs auf die Datei	*PGAs*/ Datei-Informationen auslesen (FILEDATE, FILETIME, FILESIZE, FILESTAT, FILEINFO) (Seite 139)		•	•	•	•
FILEINFO	Liefert Summe von FILEDATE, FILESIZE, FILESTAT und FILETIME zusammen	*PGAs*/ Datei-Informationen auslesen (FILEDATE, FILETIME, FILESIZE, FILESTAT, FILEINFO) (Seite 139)		•	•	•	•
FILESIZE	Liefert aktuelle Größe der Datei	*PGAs*/ Datei-Informationen auslesen (FILEDATE, FILETIME, FILESIZE, FILESTAT, FILEINFO) (Seite 139)		•	•	•	•
FILESTAT	Liefert Filestatus der Rechte Lesen, Schreiben, Execute, Anzeigen , Löschen (rwxsd)	*PGAs*/ Datei-Informationen auslesen (FILEDATE, FILETIME, FILESIZE, FILESTAT, FILEINFO) (Seite 139)		•	•	•	•
FILETIME	Liefert Uhrzeit des zuletzt schreibenden Zugriffs auf die Datei	*PGAs*/ Datei-Informationen auslesen (FILEDATE, FILETIME, FILESIZE, FILESTAT, FILEINFO) (Seite 139)		•	•	•	•
FINEA	Bewegungsende beim Erreichen von "Genauhalt Fein"	*PGAs*/ Programmierbares Bewegungsendekriterium (FINEA, COARSEA, IPOENDA, IPOBRKA, ADISPOSA) (Seite 274)	m	•	•	•	•
FL	Grenzgeschwindigkeit für Synchronachsen	*PGs*/ Framekomponenten lesen und verändern (TR, FI, RT, SC, MI) (Seite 289)	m	•	•	•	•
FLIN	Vorschub linear veränderlich	*PGAs*/ Vorschubverlauf (FNORM, FLIN, FCUB, FPO) (Seite 460)	m	•	•	•	•

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
FMA	Mehrere Vorschübe axial	*PGs/*	m	-	-	-	-
FNORM [4]	Vorschub normal nach DIN66025	*PGAs/* Vorschubverlauf (FNORM, FLIN, FCUB, FPO) (Seite 460)	m	●	●	●	●
FOCOF	Fahren mit begrenztem Moment/Kraft ausschalten	*PGAs/* Fahren auf Festanschlag (FXS, FXST, FXSW, FOCON, FOCOF) (Seite 620)	m	○	-	○	-
FOCON	Fahren mit begrenztem Moment/Kraft einschalten	*PGAs/* Fahren auf Festanschlag (FXS, FXST, FXSW, FOCON, FOCOF) (Seite 620)	m	○	-	○	-
FOR	Zählschleife mit fester Anzahl von Durchläufen	*PGAs/* Zählschleife (FOR ... TO ..., ENDFOR) (Seite 99)		●	●	●	●
FP	Festpunkt: Nummer des anzufahrenden Festpunkts	*PGs/*	s	●	●	●	●
FPO	Über ein Polynom programmierter Vorschubverlauf	*PGAs/* Vorschubverlauf (FNORM, FLIN, FCUB, FPO) (Seite 460)		-	-	-	-
FPR	Kennzeichnung Rundachse	*PGs/*		●	●	●	●
FPRAOF	Umdrehungsvorschub ausschalten	*PGs/*		●	●	●	●
FPRAON	Umdrehungsvorschub einschalten	*PGs/*		●	●	●	●
FRAME	Datentyp zur Festlegung von Koordinatensystemen	*PGAs/* Definition neuer Frames (DEF FRAME) (Seite 292)		●	●	●	●
FRC	Vorschub für Radius und Fase	*PGs/*	s	●	●	●	●
FRCM	Vorschub für Radius und Fase modal	*PGs/*	m	●	●	●	●
FROM	Die Aktion wird ausgeführt, wenn die Bedingung einmal erfüllt ist und solange die Synchronaktion aktiv ist	*PGAs/* Zyklische Prüfung der Bedingung (WHEN, WHENEVER, FROM, EVERY) (Seite 555)		●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
FTOC	Werkzeugfeinkorrektur ändern	*PGs/* Online-Werkzeugkorrektur (PUTFTOCF, FCTDEF, PUTFTOC, FTOCON, FTOCOF) (Seite 404)		•	•	•	•
FTOCOF [4]	Online wirksame Werkzeugfeinkorrektur AUS	*PGAs/* Online-Werkzeugkorrektur (PUTFTOCF, FCTDEF, PUTFTOC, FTOCON, FTOCOF) (Seite 404)	m	•	•	•	•
FTOCON	Online wirksame Werkzeugfeinkorrektur EIN	*PGAs/* Online-Werkzeugkorrektur (PUTFTOCF, FCTDEF, PUTFTOC, FTOCON, FTOCOF) (Seite 404)	m	•	•	•	•
FXS	Fahren auf Festanschlag ein	*PGs/* Fahren auf Festanschlag (FXS, FXST, FXSW, FOCON, FOCOF) (Seite 620)	m	•	•	•	•
FXST	Momentgrenze für Fahren auf Festanschlag	*PGs/* Fahren auf Festanschlag (FXS, FXST, FXSW, FOCON, FOCOF) (Seite 620)	m	•	•	•	•
FXSW	Überwachungsfenster für Fahren auf Festanschlag	*PGs/* Fahren auf Festanschlag (FXS, FXST, FXSW, FOCON, FOCOF) (Seite 620)		•	•	•	•
FZ	Zahnvorschub	*PGs/*	m	•	•	•	•
G0	Linearinterpolation mit Eilgang (Eilgangsbewegung)	*PGs/*	m	•	•	•	•
G1 [4]	Linearinterpolation mit Vorschub (Geradeninterpolation)	*PGs/*	m	•	•	•	•
G2	Kreisinterpolation im Uhrzeigersinn	*PGs/*	m	•	•	•	•
G3	Kreisinterpolation gegen Uhrzeigersinn	*PGs/*	m	•	•	•	•
G4	Verweilzeit, zeitlich vorbestimmt	*PGs/*	s	•	•	•	•
G5	Schrägeinstechschleifen	*PGAs/* Schräge Achse (TRAANG) (Seite 371)	s	•	•	•	•

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
G7	Ausgleichsbewegung beim Schrägeinstechschleifen	*PGAs*/ Schräge Achse (TRAANG) (Seite 371)	s	●	●	●	●
G9	Genauhalt - Geschwindigkeits-abnahme	*PGs*/	s	●	●	●	●
G17 [4]	Wahl der Arbeitsebene X/Y	*PGs*/	m	●	●	●	●
G18	Wahl der Arbeitsebene Z/X	*PGs*/	m	●	●	●	●
G19	Wahl der Arbeitsebene Y/Z	*PGs*/	m	●	●	●	●
G25	Untere Arbeitsfeldbegrenzung	*PGs*/	s	●	●	●	●
G26	Obere Arbeitsfeldbegrenzung	*PGs*/	s	●	●	●	●
G33	Gewindeschneiden mit konstanter Steigung	*PGs*/	m	●	●	●	●
G34	Gewindeschneiden mit linear zunehmender Steigung	*PGs*/	m	●	●	●	●
G35	Gewindeschneiden mit linear abnehmender Steigung	*PGs*/	m	●	●	●	●
G40 [4]	Werkzeugradiuskorrektur AUS	*PGs*/	m	●	●	●	●
G41	Werkzeugradiuskorrektur links von der Kontur	*PGs*/	m	●	●	●	●
G42	Werkzeugradiuskorrektur rechts von der Kontur	*PGs*/	m	●	●	●	●
G53	Unterdrückung der aktuellen Nullpunktverschiebung (satzweise)	*PGs*/	s	●	●	●	●
G54	1.Einstellbare Nullpunktverschiebung	*PGs*/	m	●	●	●	●
G55	2. Einstellbare Nullpunktverschiebung	*PGs*/	m	●	●	●	●
G56	3. Einstellbare Nullpunktverschiebung	*PGs*/	m	●	●	●	●
G57	4. Einstellbare Nullpunktverschiebung	*PGs*/	m	●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
G58	Axiale programmierbare Nullpunktverschiebung absolut, Grobverschiebung	*PGs/*	s	•	•	•	•
G59	Axiale programmierbare Nullpunktverschiebung additiv, Feinverschiebung	*PGs/*	s	•	•	•	•
G60 [4]	Genauhalt - Geschwindigkeits-abnahme	*PGs/*	m	•	•	•	•
G62	Eckenverzögerung an Innenecken bei aktiver Werkzeuradiuskorrektur (G41, G42)	*PGAs/* Vorschubreduzierung mit Eckenverzögerung (FENDNORM, G62, G621) (Seite 273)	m	•	•	•	•
G63	Gewindebohren mit Ausgleichsfutter	*PGs/*	s	•	•	•	•
G64	Bahnsteuerbetrieb	*PGs/*	m	•	•	•	•
G70	Inch-Maßangabe für geometrische Angaben (Längen)	*PGs/*	m	•	•	•	•
G71 [4]	Metrische Maßangabe für geometrische Angaben (Längen)	*PGs/*	m	•	•	•	•
G74	Referenzpunktanfahren	*PGs/*	s	•	•	•	•
G75	Festpunktanfahren	*PGs/*	s	•	•	•	•
G90 [4]	Maßangabe absolut	*PGs/*	m/s	•	•	•	•
G91	Kettenmaßangabe	*PGs/*	m/s	•	•	•	•
G93	Zeitreziproker Vorschub 1/min	*PGs/*	m	•	•	•	•
G94 [4]	Linearvorschub F in mm/min oder inch/min und Grad/min	*PGs/*	m	•	•	•	•
G95	Umdrehungsvorschub F in mm/U oder inch/U	*PGs/*	m	•	•	•	•
G96	konstante Schnittgeschwindigkeit (wie bei G95) EIN	*PGs/*	m	•	•	•	•

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
G97	konstante Schnittgeschwindigkeit (wie bei G95) AUS	*PGs/*	m	●	●	●	●
G110	Polprogrammierung relativ zur letzten programmierten Sollposition	*PGs/*	s	●	●	●	●
G111	Polprogrammierung relativ zum Nullpunkt des aktuellen Werkstück-Koordinatensystems	*PGs/*	s	●	●	●	●
G112	Polprogrammierung relativ zum letzten gültigen Pol	*PGs/*	s	●	●	●	●
G140 [4]	Anfahrrichtung WAB festgelegt durch G41/G42	*PGs/*	m	●	●	●	●
G141	Anfahrrichtung WAB links der Kontur	*PGs/*	m	●	●	●	●
G142	Anfahrrichtung WAB rechts der Kontur	*PGs/*	m	●	●	●	●
G143	Anfahrrichtung WAB tangentenabhängig	*PGs/*	m	●	●	●	●
G147	Weiches Anfahren mit Gerade	*PGs/*	s	●	●	●	●
G148	Weiches Abfahren mit Gerade	*PGs/*	s	●	●	●	●
G153	Unterdrückung aktueller Frames inklusive Basisframe	*PGs/*	s	●	●	●	●
G247	Weiches Anfahren mit Viertelkreis	*PGs/*	s	●	●	●	●
G248	Weiches Abfahren mit Viertelkreis	*PGs/*	s	●	●	●	●
G290	Umschalten auf SINUMERIK-Mode EIN	*FBW*	m	●	●	●	●
G291	Umschalten auf ISO2/3-Mode EIN	*FBW*	m	●	●	●	●
G331	Gewindebohren ohne Ausgleichsfutter, positive Steigung, Rechtslauf	*PGs/*	m	●	●	●	●
G332	Gewindebohren ohne Ausgleichsfutter, negative Steigung, Linkslauf	*PGs/*	m	●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
G340 [4]	Anfahrsatz räumlich (Tiefe und in der Ebene zugleich (Helix))	*PGs/*	m	•	•	•	•
G341	Zuerst in der senkrechten Achse zustellen (z), dann Anfahren in der Ebene	*PGs/*	m	•	•	•	•
G347	Weiches Anfahren mit Halbkreis	*PGs/*	s	•	•	•	•
G348	Weiches Abfahren mit Halbkreis	*PGs/*	s	•	•	•	•
G450 [4]	Übergangskreis	*PGs/*	m	•	•	•	•
G451	Schnittpunkt der Äquidistanten	*PGs/*	m	•	•	•	•
G460 [4]	Einschalten der Kollisionsüberwachung für An- und Abfahrsatz	*PGs/*	m	•	•	•	•
G461	Einfügen eines Kreises im WRK-Satz	*PGs/*	m	•	•	•	•
G462	Einfügen einer Geraden im WRK-Satz	*PGs/*	m	•	•	•	•
G500 [4]	Ausschalten aller einstellbaren Frames, Basisframes sind aktiv	*PGs/*	m	•	•	•	•
G505 ... G599	5 ... 99. Einstellbare Nullpunktverschiebung	*PGs/*	m	•	•	•	•
G601 [4]	Satzwechsel bei Genauhalt fein	*PGs/*	m	•	•	•	•
G602	Satzwechsel bei Genauhalt grob	*PGs/*	m	•	•	•	•
G603	Satzwechsel bei IPO-Satzende	*PGs/*	m	•	•	•	•
G621	Eckenverzögerung an allen Ecken	*PGAs/* Vorschubreduzierung mit Eckenverzögerung (FENDNORM, G62, G621) (Seite 273)	m	•	•	•	•
G641	Bahnsteuerbetrieb mit Überschleifen nach Wegkriterium (= programmierbarer Überschleifabstand)	*PGs/*	m	•	•	•	•

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
G642	Bahnsteuerbetrieb mit Überschleifen unter Einhaltung definierter Toleranzen	*PGsl*	m	●	●	●	●
G643	Bahnsteuerbetrieb mit Überschleifen unter Einhaltung definierter Toleranzen (satzintern)	*PGsl*	m	●	●	●	●
G644	Bahnsteuerbetrieb mit Überschleifen mit maximal möglicher Dynamik	*PGsl*	m	●	●	●	●
G645	Bahnsteuerbetrieb mit Überschleifen von Ecken und tangentialer Satzübergänge unter Einhaltung definierter Toleranzen	*PGsl*	m	●	●	●	●
G700	Inch-Maßangabe für geometrische und technologische Angaben (Längen, Vorschub)	*PGsl*	m	●	●	●	●
G710 [4]	Metrische Maßangabe für geometrische und technologische Angaben (Längen, Vorschub)	*PGsl*	m	●	●	●	●
G751	Festpunkt über Zwischenpunkt anfahren	PGsl	s	●	●	●	●
G810 [4], ..., G819	Für den OEM-Anwender reservierte G-Gruppe	*PGAsl* Spezielle Funktionen für den OEM-Anwender (OEMIPO1, OEMIPO2, G810 bis G829) (Seite 272)		●	●	●	●
G820 [4], ..., G829	Für den OEM-Anwender reservierte G-Gruppe	*PGAsl* Spezielle Funktionen für den OEM-Anwender (OEMIPO1, OEMIPO2, G810 bis G829) (Seite 272)		●	●	●	●
G931	Vorschubvorgabe durch Verfahrzeit		m	●	●	●	●
G942	Linear-Vorschub und konstante Schnittgeschwindigkeit oder Spindeldrehzahl einfrieren		m	●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
G952	Umdrehungsvorschub und konstante Schnittgeschwindigkeit oder Spindeldrehzahl einfrieren		m	•	•	•	•
G961	konstante Schnittgeschwindigkeit und Linear-Vorschub	*PGsl*	m	•	•	•	•
G962	Linear-Vorschub oder Umdrehungsvorschub und konstante Schnittgeschwindigkeit	*PGsl*	m	•	•	•	•
G971	Spindeldrehzahl einfrieren und Linear-Vorschub	*PGsl*	m	•	•	•	•
G972	Linear-Vorschub oder Umdrehungsvorschub und konstante Spindeldrehzahl einfrieren	*PGsl*	m	•	•	•	•
G973	Umdrehungsvorschub ohne Spindeldrehzahl-begrenzung	*PGsl*	m	•	•	•	•
GEOAX	Den Geometrieachsen 1 - 3 neue Kanalachsen zuordnen	*PGAsl* Umschaltbare Geometrieachsen (GEOAX) (Seite 672)		•	•	•	•
GET	Freigegebene Achse zwischen Kanälen tauschen	*PGAsl* Achstausch, Spindeltausch (RELEASE, GET, GETD) (Seite 121)		•	•	•	•
GETACTT	Bestimmt das aktive Werkzeug aus einer Gruppe von gleichnamigen Werkzeugen	*FBW*		•	•	•	•
GETACTTD	Bestimmt zu einer absoluten D-Nummer die zugehörige T-Nummer	*PGAsl* Freie D-Nummernvergabe: T-Nummer zur vorgegebenen D-Nummer ermitteln (GETACTTD) (Seite 432)		•	•	•	•
GETD	Achse direkt zwischen Kanälen tauschen	*PGAsl* Achstausch, Spindeltausch (RELEASE, GET, GETD) (Seite 121)		•	•	•	•

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
GETDNO	Liefert D-Nummer einer Schneide (CE) eines Werkzeugs (T)	*PGAs*/ Freie D-Nummernvergabe: D-Nummern umbenennen (GETDNO, SETDNO) (Seite 431)		●	●	●	●
GETEXET	Lesen der eingewechselten T-Nummer	*FBW*		●	●	●	●
GETFREELOC	Für ein gegebenes Werkzeug einen Leerplatz in den Magazinen suchen	*FBW*		●	●	●	●
GETSELT	Vorgewählte T-Nummer liefern	*FBW*		●	●	●	●
GETT	T-Nummer zu Werkzeugnamen bestimmen	*FBW*		●	●	●	●
GETTCOR	Werkzeuglängen bzw. Werkzeuglängen-komponenten auslesen	*FB1(W1)*		●	●	●	●
GETTENV	T-, D-, und DL-Nummern lesen	*FB1(W1)*		●	●	●	●
GOTO	Sprunganweisung erst vorwärts dann rückwärts (Richtung erst zum Programm-Ende und dann zum Programm-Anfang)	*PGAs*/ Programmsprünge auf Sprungmarken (GOTOB, GOTOF, GOTO, GOTOC) (Seite 83)		●	●	●	●
GOTOB	Sprunganweisung rückwärts (Richtung Programm-Anfang)	*PGAs*/ Programmsprünge auf Sprungmarken (GOTOB, GOTOF, GOTO, GOTOC) (Seite 83)		●	●	●	●
GOTOC	Wie GOTO, aber Alarm 14080 "Sprungziel nicht gefunden" unterdrücken	*PGAs*/ Programmsprünge auf Sprungmarken (GOTOB, GOTOF, GOTO, GOTOC) (Seite 83)		●	●	●	●
GOTOF	Sprunganweisung vorwärts (Richtung Programm-Ende)	*PGAs*/ Programmsprünge auf Sprungmarken (GOTOB, GOTOF, GOTO, GOTOC) (Seite 83)		●	●	●	●
GOTOS	Rücksprung auf Programmanfang	*PGAs*/ Rücksprung auf Programmanfang (GOTOS) (Seite 82)		●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
GP	Schlüsselwort zur indirekten Programmierung von Positionsattributen	*PGAs/* Indirekte Programmierung von Positionsattributen (GP) (Seite 57)		●	●	●	●
GWPSOF	Konstante Scheibenumfangs-geschwindigkeit (SUG) abwählen	*PGs/*	s	●	●	●	●
GWPSON	Konstante Scheibenumfangs-geschwindigkeit (SUG) anwählen	*PGs/*	s	●	●	●	●
H...	Hilfsfunktionsausgabe an die PLC	*PGs//FB1(H2)*		●	●	●	●
HOLES1	Bohrbildzyklus, Lochreihe	*BHDs//BHFs/*		●	●	●	●
HOLES2	Bohrbildzyklus, Lochkreis	*BHDs//BHFs/*		●	●	●	●
I	Interpolationsparameter	*PGs/*	s	●	●	●	●
I1	Zwischenpunktkoordinate	*PGs/*	s	●	●	●	●
IC	Kettenmaßeingabe	*PGs/*	s	●	●	●	●
ICYCOF	Alle Sätze eines Technologiezyklus nach ICYCOF in einem IPO-Takt abarbeiten	*PGAs/* Steuerung der Abarbeitung von Technologiezyklen (ICYCOF, ICYCON) (Seite 631)		●	●	●	●
ICYCON	Jeden Satz eines Technologiezyklus nach ICYCON in einem separaten IPO-Takt abarbeiten	*PGAs/* Steuerung der Abarbeitung von Technologiezyklen (ICYCOF, ICYCON) (Seite 631)		●	●	●	●
ID	Kennzeichnung für modale Synchronaktionen	*PGAs/* Gültigkeitsbereich und Bearbeitungsreihenfolge (ID, IDS) (Seite 553)	m	●	●	●	●
IDS	Kennzeichnung für modale statische Synchronaktionen	*PGAs/* Gültigkeitsbereich und Bearbeitungsreihenfolge (ID, IDS) (Seite 553)		●	●	●	●
IF	Einleitung eines bedingten Sprungs im Teileprogramm / Technologiezyklus	*PGAs/* Programmschleife mit Alternative (IF, ELSE, ENDIF) (Seite 96)		●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
INDEX	Index eines Zeichens im Eingangsstring bestimmen	*PGAs/* Zeichen/String in String suchen (INDEX, RINDEX, MINDEX, MATCH) (Seite 78)		●	●	●	●
INIPO	Initialisierung der Variablen bei PowerOn	*PGAs/* Definition von Anwendervariablen (DEF) (Seite 22)		●	●	●	●
INIRE	Initialisierung der Variablen bei Reset	*PGAs/* Definition von Anwendervariablen (DEF) (Seite 22)		●	●	●	●
INICF	Initialisierung der Variablen bei NewConfig	*PGAs/* Definition von Anwendervariablen (DEF) (Seite 22)		●	●	●	●
INIT	Anwahl eines bestimmten NC-Programms zur Abarbeitung in einem bestimmten Kanal	*PGAs/* Programmkoordinierung (INIT, START, WAITM, WAITMC, WAITE, SETM, CLEARM) (Seite 103)		-	-	-	-
INITIAL	Erzeugen eines INI-Files über alle Bereiche	*PGAs/* Arbeitsspeicher (CHANDATA, COMPLETE, INITIAL) (Seite 210)		●	●	●	●
INT	Datentyp: Ganzzahliger Wert mit Vorzeichen	*PGAs/* Definition von Anwendervariablen (DEF) (Seite 22)		●	●	●	●
INTERSEC	Schnittpunkt zwischen zwei Konturelementen berechnen	*PGAs/* Schnittpunkt zwischen zwei Konturelementen ermitteln (INTERSEC) (Seite 712)		●	●	●	●
INVCCW	Evolvente fahren, gegen den Uhrzeigersinn	*PGs/*	m	-	-	-	-
INVCW	Evolvente fahren, im Uhrzeigersinn	*PGs/*	m	-	-	-	-
INVFRAME	Aus einem Frame den inversen Frame berechnen	*FB1(K2)*		●	●	●	●
IP	Variabler Interpolationsparameter	*PGAs/* Indirekte Programmierung (Seite 53)		●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
IPOBRKA	Bewegungskriterium ab Einsatzpunkt der Bremsrampe	*PGAs*/ Programmierbares Bewegungsendekriterium (FINEA, COARSEA, IPOENDA, IPOBRKA, ADISPOSA) (Seite 274)	m	•	•	•	•
IPOENDA	Bewegungsende beim Erreichen von "IPO-Stopp"	*PGAs*/ Programmierbares Bewegungsendekriterium (FINEA, COARSEA, IPOENDA, IPOBRKA, ADISPOSA) (Seite 274)	m	•	•	•	•
IPTRLOCK	Beginn des suchunfähigen Programmabschnitts auf nächsten Maschinenfunktionssatz einfrieren.	*PGAs*/ Programmstelle für SERUPRO verhindern (IPTRLOCK, IPTRUNLOCK) (Seite 473)	m	•	•	•	•
IPTRUNLOCK	Ende des suchunfähigen Programmabschnitts auf aktuellen Satz zum Unterbrechungszeitpunkt setzen.	*PGAs*/ Programmstelle für SERUPRO verhindern (IPTRLOCK, IPTRUNLOCK) (Seite 473)	m	•	•	•	•
ISAXIS	Prüfen, ob die als Parameter angegebene Geometrieachse 1 ist	*PGAs*/ Achsfunktionen (AXNAME, AX, SPI, AXTOSPI, ISAXIS, AXSTRING, MODAXVAL) (Seite 669)		•	•	•	•
ISD	Eintauchtiefe	*PGAs*/ Aktivierung von 3D-Werkzeugkorrekturen (CUT3DC, CUT3DF, CUT3DFS, CUT3DFF, ISD) (Seite 409)	m	•	•	•	•
ISFILE	Prüfen, ob eine Datei im NCK-Anwendungs-speicher vorhanden ist	*PGAs*/ Vorhandensein einer Datei prüfen (ISFILE) (Seite 137)		•	•	•	•
ISNUMBER	Prüfen, ob Eingangsstring in Zahl umgewandelt werden kann	*PGAs*/ Typenkonvertierung von STRING (NUMBER, ISNUMBER, AXNAME) (Seite 74)		•	•	•	•
ISOCALL	Indirekter Aufruf eines in ISO-Sprache programmierten Programms	*PGAs*/ Indirekter Aufruf eines in ISO-Sprache programmierten Programms (ISOCALL) (Seite 190)		•	•	•	•

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
ISVAR	Prüfen, ob der Übergabeparameter eine in der NC bekannte Variable enthält	*PGAs*/ Funktionsaufruf ISVAR und Maschinendaten Array-Index lesen (Seite 687)		●	●	●	●
J	Interpolationsparameter	*PGs*/	s	●	●	●	●
J1	Zwischenpunktkoordinate	*PGs*/	s	●	●	●	●
JERKA	Über MD eingestelltes Beschleunigungsverhalten für die programmierten Achsen aktivieren			●	●	●	●
JERKLIM	Reduktion oder Überhöhung des maximalen axialen Rucks	*PGAs*/ Prozentuale Ruckkorrektur (JERKLIM) (Seite 486)	m	●	●	●	●
JERKLIMA	Reduktion oder Überhöhung des maximalen axialen Rucks	*PGs*/	m	●	●	●	●
K	Interpolationsparameter	*PGs*/	s	●	●	●	●
K1	Zwischenpunktkoordinate	*PGs*/	s	●	●	●	●
KONT	Kontur umfahren bei der Werkzeugkorrektur	*PGs*/	m	●	●	●	●
KONTC	Mit krümmungsstetigem Polynom an-/abfahren	*PGs*/	m	●	●	●	●
KONTT	Mit tangentenstetigem Polynom an-/abfahren	*PGs*/	m	●	●	●	●
L	Unterprogramm-Nummer	*PGAs*/ Unterprogrammaufruf ohne Parameterübergabe (Seite 178)	s	●	●	●	●
LEAD	Voreilwinkel 1. Werkzeugorientierung 2. Orientierungpolynome	*PGAs*/ Programmierung der Werkzeugorientierung (A..., B..., C..., LEAD, TILT) (Seite 323)	m	● -	● -	● -	● -
LEADOF	Leitwerkkopplung AUS	*PGAs*/ Axiale Leitwertkopplung (LEADON, LEADOF) (Seite 520)		-	-	-	-
LEADON	Leitwerkkopplung EIN	*PGAs*/ Axiale Leitwertkopplung (LEADON, LEADOF) (Seite 520)		-	-	-	-

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
LENTOAX	Liefert Informationen über die Zuordnung der Werkzeuglängen L1, L2 und L3 des aktiven Werkzeugs zur Abszisse, Ordinate und Applikate	*FB1(W1)*		•	•	•	•
LFOF [4]	Schnellrückzug für Gewindeschneiden AUS	*PGs/*	m	•	•	•	•
LFON	Schnellrückzug für Gewindeschneiden EIN	*PGs/*	m	•	•	•	•
LFPOS	Rückzug der mit POLFMASK oder POLFMLIN bekannt gemachten Achse auf die mit POLF programmierte absolute Achsposition	*PGs/*	m	•	•	•	•
LFTXT	Ebene der Rückzugsbewegung beim Schnellabheben wird bestimmt aus der Bahntangente und der aktuellen Werkzeugrichtung	*PGs/*	m	•	•	•	•
LFWP	Ebene der Rückzugsbewegung beim Schnellabheben wird bestimmt durch die aktuelle Arbeitsebene (G17/G18/G19)	*PGs/*	m	•	•	•	•
LIFTFAST	Schnellabheben	*PGs/* Schnellabheben von der Kontur (SETINT LIFTFAST, ALF) (Seite 115)		•	•	•	•
LIMS	Drehzahlbegrenzung bei G96/G961 und G97	*PGs/*	m	•	•	•	•
LLI	Unterer Grenzwert von Variablen	*PGAs/* Attribut: Grenzwerte (LLI, ULI) (Seite 34)		•	•	•	•
LN	Natürlicher Logarithmus	*PGAs/* Rechenfunktionen (Seite 61)		•	•	•	•
LOCK	Synchronaktion mit ID sperren (Technologiezyklus stoppen)	*PGAs/* Sperren, Freischalten, Zurücksetzen (LOCK, UNLOCK, RESET) (Seite 634)		•	•	•	•

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
LONGHOLE	Fräsbildzyklus Langlöcher auf einem Kreis	*BHDs//BHFs/*		-	-	-	-
LOOP	Einleitung einer Endlosschleife	*PGAs/* Endlos-Programmschleife (LOOP, ENDLOOP) (Seite 98)		●	●	●	●
M0	Programmierter Halt	*PGs/*		●	●	●	●
M1	Wahlweiser Halt	*PGs/*		●	●	●	●
M2	Programmende Hauptprogramm mit Rücksetzen auf Programmanfang	*PGs/*		●	●	●	●
M3	Spindeldrehrichtung rechts	*PGs/*		●	●	●	●
M4	Spindeldrehrichtung links	*PGs/*		●	●	●	●
M5	Spindel halt	*PGs/*		●	●	●	●
M6	Werkzeugwechsel	*PGs/*		●	●	●	●
M17	Unterprogrammende	*PGs/*		●	●	●	●
M19	Spindelpositionierung auf die im SD43240 eingetragene Position	*PGs/*		●	●	●	●
M30	Programmende, wie M2	*PGs/*		●	●	●	●
M40	Automatische Getriebeschaltung	*PGs/*		●	●	●	●
M41 ... M45	Getriebestufe 1 ... 5	*PGs/*		●	●	●	●
M70	Übergang in Achsbetrieb	*PGs/*		●	●	●	●
MASLDEF	Master/Slave-Achsverband definieren	*PGAs/* Master-/Slave-Verband (MASLDEF, MASLDEL, MASLON, MASLOF, MASLOFS) (Seite 546)		●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
MASLDEL	Master/Slave-Achsverband trennen und Definition des Verbandes löschen	*PGAs*/ Master-/Slave-Verband (MASLDEF, MASLDEL, MASLON, MASLOF, MASLOFS) (Seite 546)		●	●	●	●
MASLOF	Ausschalten einer temporären Kopplung	*PGAs*/ Master-/Slave-Verband (MASLDEF, MASLDEL, MASLON, MASLOF, MASLOFS) (Seite 546)		●	●	●	●
MASLOFS	Ausschalten einer temporären Kopplung mit automatischem Stillsetzen der Slave-Achse	*PGAs*/ Master-/Slave-Verband (MASLDEF, MASLDEL, MASLON, MASLOF, MASLOFS) (Seite 546)		●	●	●	●
MASLON	Einschalten einer temporären Kopplung	*PGAs*/ Master-/Slave-Verband (MASLDEF, MASLDEL, MASLON, MASLOF, MASLOFS) (Seite 546)		●	●	●	●
MATCH	Suchen eines String im String	*PGAs*/ Zeichen/String in String suchen (INDEX, RINDEX, MINDEX, MATCH) (Seite 78)		●	●	●	●
MAXVAL	Größerer Wert zweier Variablen (arithm. Funktion)	*PGAs*/ Minimum, Maximum und Bereich von Variablen (MINVAL, MAXVAL, BOUND) (Seite 68)		●	●	●	●
MCALL	Modaler Unterprogrammaufruf	*PGAs*/ Modaler Unterprogrammaufruf (MCALL) (Seite 185)		●	●	●	●
MEAC	Kontinuierliches Messen ohne Restweglöschen	*PGAs*/ Erweiterte Messfunktion (MEASA, MEAWA, MEAC) (Option) (Seite 262)	s	-	-	-	-
MEAFRAME	Frame-Berechnung aus Messpunkten	*PGAs*/ Frame-Berechnung aus 3 Messpunkten im Raum (MEAFRAME) (Seite 298)		●	●	●	●
MEAS	Messen mit schaltendem Taster	*PGAs*/ Messen mit schaltendem Taster (MEAS, MEAW) (Seite 259)	s	●	●	●	●
MEASA	Messen mit Restweglöschen	*PGAs*/ Erweiterte Messfunktion (MEASA, MEAWA, MEAC) (Option) (Seite 262)	s	-	-	-	-

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
MEASURE	Berechnungsmethode für die Werkstück- und Werkzeugvermessung	*FB2(M5)* Messen mit schaltendem Taster (MEAS, MEAW) (Seite 259)		●	●	●	●
MEAW	Messen mit schaltendem Taster ohne Restweglöschen	*PGAs/* Messen mit schaltendem Taster (MEAS, MEAW) (Seite 259)	s	●	●	●	●
MEAWA	Messen ohne Restweglöschen	*PGAs/* Erweiterte Messfunktion (MEASA, MEAWA, MEAC) (Option) (Seite 262)	s	-	-	-	-
MI	Zugriff auf Frame-Daten: Spiegelung	*PGAs/* Framekomponenten lesen und verändern (TR, FI, RT, SC, MI) (Seite 289)		●	●	●	●
MINDEX	Index eines Zeichens im Eingangsstring bestimmen	*PGAs/* Zeichen/String in String suchen (INDEX, RINDEX, MINDEX, MATCH) (Seite 78)		●	●	●	●
MINVAL	Kleinerer Wert zweier Variablen (arithm. Funktion)	*PGAs/* Minimum, Maximum und Bereich von Variablen (MINVAL, MAXVAL, BOUND) (Seite 68)		●	●	●	●
MIRROR	Programmierbare Spiegelung	*PGAs/*	s	●	●	●	●
MMC	Aus dem Teileprogramm interaktiv Dialogfenster am HMI aufrufen	*PGAs/* Fenster aus dem Teileprogramm interaktiv aufrufen (MMC) (Seite 691)		●	●	●	●
MOD	Modulo-Division	*PGAs/* Rechenfunktionen (Seite 61)		●	●	●	●
MODAXVAL	Modulo-Position einer Modulo-Rundachse ermitteln	*PGAs/* Achsfunktionen (AXNAME, AX, SPI, AXTOSPI, ISAXIS, AXSTRING, MODAXVAL) (Seite 669)		●	●	●	●
MOV	Positionierachse starten	*PGAs/* Achse starten/stoppen (MOV) (Seite 602)		●	●	●	●
MSG	Programmierbare Meldungen	*PGs/*	m	●	●	●	●
MVTOOL	Sprachbefehl zum Bewegen eines Werkzeugs	*FBW*		●	●	●	●
N	NC-Nebensatznummer	*PGs/*		●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
NCK	Spezifizierung des Gültigkeitsbereichs von Daten	*PGAs*/ Definition von Anwendervariablen (DEF) (Seite 22)		•	•	•	•
NEWCONF	Geänderte Maschinendaten übernehmen (entspricht "Maschinendatum wirksam setzen")	*PGAs*/ Maschinendaten wirksam setzen (NEWCONF) (Seite 128)		•	•	•	•
NEWT	Neues Werkzeug anlegen	*PGAs*/ Framekomponenten lesen und verändern (TR, FI, RT, SC, MI) (Seite 289)		•	•	•	•
NORM [4]	Normaleinstellung im Anfangs-, Endpunkt bei der Werkzeugkorrektur	*PGs*/	m	•	•	•	•
NOT	Logisches NICHT (Negation)	*PGAs*/ Vergleichs- und logische Operationen (Seite 64)		•	•	•	•
NPROT	Maschinenspezifischer Schutzbereich EIN/AUS	*PGAs*/ Schutzbereiche aktivieren/deaktivieren (CPROT, NPROT) (Seite 219)		•	•	•	•
NPROTDEF	Definition eines maschinenspezifischen Schutzbereichs	*PGAs*/ Festlegung der Schutzbereiche (CPROTDEF, NPROTDEF) (Seite 215)		•	•	•	•
NUMBER	Eingangsstring in Zahl umwandeln	*PGAs*/ Typenkonvertierung von STRING (NUMBER, ISNUMBER, AXNAME) (Seite 74)		•	•	•	•
OEMIPO1	OEM-Interpolation 1	*PGAs*/ Spezielle Funktionen für den OEM-Anwender (OEMIPO1, OEMIPO2, G810 bis G829) (Seite 272)	m	•	•	•	•
OEMIPO2	OEM-Interpolation 2	*PGAs*/ Spezielle Funktionen für den OEM-Anwender (OEMIPO1, OEMIPO2, G810 bis G829) (Seite 272)	m	•	•	•	•
OF	Schlüsselwort in der CASE-Verzweigung	*PGAs*/ Programmverzweigung (CASE ... OF ... DEFAULT ...) (Seite 86)		•	•	•	•
OFFN	Aufmaß zur programmierten Kontur	*PGs*/	m	•	•	•	•

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
OMA1	OEM-Adresse 1		m	●	●	●	●
OMA2	OEM-Adresse 2		m	●	●	●	●
OMA3	OEM-Adresse 3		m	●	●	●	●
OMA4	OEM-Adresse 4		m	●	●	●	●
OMA5	OEM-Adresse 5		m	●	●	●	●
OR	Logischer Operator, ODER-Verknüpfung	*PGAs/* Vergleichs- und logische Operationen (Seite 64)		●	●	●	●
ORIAXES	Lineare Interpolation der Maschinenachsen oder Orientierungsachsen	*PGAs/* Programmierung der Orientierungsachsen (ORIAXES, ORIVECT, ORIEULER, ORIRPY, ORIRPY2, ORIVIRT1, ORIVIRT2) (Seite 333)	m	●	●	●	●
ORIAXPOS	Orientierungswinkel über virtuelle Orientierungsachsen mit Rundachspositionen		m	●	●	●	●
ORIC [4]	Orientierungsänderungen an Außenecken werden dem einzufügenden Kreissatz überlagert	*PGAs/* Werkzeugorientierung (ORIC, ORID, OSOF, OSC, OSS, OSSE, ORIS, OSD, OST) (Seite 423)	m	●	●	●	●
ORICONCCW	Interpolation auf einer Kreismantelfläche im Gegenuhrzeigersinn	*PGAs/FB3(F3)* Orientierungsprogrammierung entlang einer Kegelmantelfläche (ORIPLANE, ORICONCW, ORICONCCW, ORICONTO, ORICONIO) (Seite 335)	m	●	●	●	●
ORICONCW	Interpolation auf einer Kreismantelfläche im Uhrzeigersinn	*PGAs/FB3(F4)* Orientierungsprogrammierung entlang einer Kegelmantelfläche (ORIPLANE, ORICONCW, ORICONCCW, ORICONTO, ORICONIO) (Seite 335)	m	●	●	●	●
ORICONIO	Interpolation auf einer Kreismantelfläche mit Angabe einer Zwischenorientierung	*PGAs/FB3(F4)* Orientierungsprogrammierung entlang einer Kegelmantelfläche (ORIPLANE, ORICONCW, ORICONCCW, ORICONTO, ORICONIO) (Seite 335)	m	●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
ORICONTO	Interpolation auf einer Kreismantelfläche im tangentialen Übergang (Angabe der Endorientierung)	*PGAsl/FB3(F5)* Orientierungsprogrammierung entlang einer Kegelmantelfläche (ORIPLANE, ORICONCW, ORICONCCW, ORICONTO, ORICONIO) (Seite 335)	m	•	•	•	•
ORICURVE	Interpolation der Orientierung mit Vorgabe der Bewegung zweier Kontaktpunkte des Werkzeugs	*PGAsl/FB3(F6)* Orientierungsvorgabe zweier Kontaktpunke (ORICURVE, PO[XH]=, PO[YH]=, PO[ZH]=) (Seite 339)	m	•	•	•	•
ORID	Orientierungsänderungen werden vor dem Kreissatz ausgeführt	*PGAsl* Werkzeugorientierung (ORIC, ORID, OSOF, OSC, OSS, OSSE, ORIS, OSD, OST) (Seite 423)	m	•	•	•	•
ORIEULER	Orientierungswinkel über Euler-Winkel	*PGAsl* Programmierung der Orientierungsachsen (ORIAXES, ORIVECT, ORIEULER, ORIRPY, ORIRPY2, ORIVIRT1, ORIVIRT2) (Seite 333)	m	•	•	•	•
ORIMKS	Werkzeugorientierung im Maschinen-Koordinatensystem	*PGAsl* Bezug der Orientierungsachsen (ORIWKS, ORIMKS) (Seite 331)	m	•	•	•	•
ORIPATH	Werkzeugorientierung bezogen auf die Bahn	*PGAsl* Bahnrelative Drehung der Werkzeugorientierung (ORIPATH, ORIPATHS, Drehwinkel) (Seite 348)	m	•	•	•	•
ORIPATHS	Werkzeugorientierung bezogen auf die Bahn, ein Knick im Orientierungsverlauf wird geglättet	*PGAsl* Bahnrelative Drehung der Werkzeugorientierung (ORIPATH, ORIPATHS, Drehwinkel) (Seite 348)	m	•	•	•	•
ORIPLANE	Interpolation in einer Ebene (entspricht ORIVECT) Großkreisinterpolation	*PGAsl* Orientierungsprogrammierung entlang einer Kegelmantelfläche (ORIPLANE, ORICONCW, ORICONCCW, ORICONTO, ORICONIO) (Seite 335)	m	•	•	•	•
ORIRESET	Grundstellung der Werkzeugorientierung mit bis zu 3 Orientierungsachsen	*PGAsl* Varianten der Orientierungsprogrammierung und Grundstellung (ORIRESET) (Seite 321)		•	•	•	•

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
ORIROTA	Drehwinkel zu einer absolut vorgegebenen Drehrichtung	*PGAs*/ Drehungen der Werkzeugorientierung (ORIROTA, ORIROTR, ORIROTT, ORIROTC, THETA) (Seite 343)	m	●	●	●	●
ORIROTC	Tangentialer Drehvektor zur Bahntangente	*PGAs*/ Drehungen der Werkzeugorientierung (ORIROTA, ORIROTR, ORIROTT, ORIROTC, THETA) (Seite 343)	m	●	●	●	●
ORIROTR	Drehwinkel relativ zur Ebene zwischen Start- und Endorientierung	*PGAs*/ Drehungen der Werkzeugorientierung (ORIROTA, ORIROTR, ORIROTT, ORIROTC, THETA) (Seite 343)	m	●	●	●	●
ORIROTT	Drehwinkel relativ zur Änderung des Orientierungsvektors	*PGAs*/ Drehungen der Werkzeugorientierung (ORIROTA, ORIROTR, ORIROTT, ORIROTC, THETA) (Seite 343)	m	●	●	●	●
ORIRPY	Orientierungswinkel über RPY-Winkel (XYZ)	*PGAs*/ Programmierung der Orientierungsachsen (ORIAXES, ORIVECT, ORIEULER, ORIRPY, ORIRPY2, ORIVIRT1, ORIVIRT2) (Seite 333)	m	●	●	●	●
ORIRPY2	Orientierungswinkel über RPY-Winkel (ZYX)	*PGAs*/ Programmierung der Orientierungsachsen (ORIAXES, ORIVECT, ORIEULER, ORIRPY, ORIRPY2, ORIVIRT1, ORIVIRT2) (Seite 333)	m	●	●	●	●
ORIS	Orientierungsänderung	*PGAs*/ Werkzeugorientierung (ORIC, ORID, OSOF, OSC, OSS, OSSE, ORIS, OSD, OST) (Seite 423)	m	●	●	●	●
ORISOF [4]	Glättung des Orientierungsverlaufs AUS	*PGAs*/ Glättung des Orientierungsverlaufs (ORISON, ORISOF) (Seite 357)	m	●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
ORISON	Glättung des Orientierungsverlaufs EIN	*PGAs/* Glättung des Orientierungsverlaufs (ORISON, ORISOF) (Seite 357)	m	●	●	●	●
ORIVECT	Großkreisinterpolation (identisch mit ORIPLANE)	*PGAs/* Programmierung der Orientierungsachsen (ORIAXES, ORIVECT, ORIEULER, ORIRPY, ORIRPY2, ORIVIRT1, ORIVIRT2) (Seite 333)	m	●	●	●	●
ORIVIRT1	Orientierungswinkel über virtuelle Orientierungsachsen (Definition 1)	*PGAs/* Programmierung der Orientierungsachsen (ORIAXES, ORIVECT, ORIEULER, ORIRPY, ORIRPY2, ORIVIRT1, ORIVIRT2) (Seite 333)	m	●	●	●	●
ORIVIRT2	Orientierungswinkel über virtuelle Orientierungsachsen (Definition 1)	*PGAs/* Programmierung der Orientierungsachsen (ORIAXES, ORIVECT, ORIEULER, ORIRPY, ORIRPY2, ORIVIRT1, ORIVIRT2) (Seite 333)	m	●	●	●	●
ORIWKS [4]	Werkzeugorientierung im Werkstück-Koordinatensystem	*PGAs/* Bezug der Orientierungsachsen (ORIWKS, ORIMKS) (Seite 331)	m	●	●	●	●
OS	Pendeln ein/aus	*PGAs/* Asynchrones Pendeln (OS, OSP1, OSP2, OST1, OST2, OSCTRL, OSNSC, OSE, OSB) (Seite 641)		-	-	-	-
OSB	Pendeln: Startpunkt	*FB2(P5)*	m	-	-	-	-
OSC	Konstante Glättung Werkzeugorientierung	*PGAs/* Werkzeugorientierung (ORIC, ORID, OSOF, OSC, OSS, OSSE, ORIS, OSD, OST) (Seite 423)	m	●	●	●	●
OSCILL	Axis: 1 - 3 Zustellachsen	*PGAs/* Über Synchronaktionen gesteuertes Pendeln (OSCILL) (Seite 647)	m	-	-	-	-

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
OSCTRL	Optionen pendeln	*PGAs*/ Asynchrones Pendeln (OS, OSP1, OSP2, OST1, OST2, OSCTRL, OSNSC, OSE, OSB) (Seite 641)	m	-	-	-	-
OSD	Überschleifen de Werkzeugorientierung durch Vorgabe der Überschleiflänge mit SD	*PGAs*/ Werkzeugorientierung (ORIC, ORID, OSOF, OSC, OSS, OSSE, ORIS, OSD, OST) (Seite 423)	m	●	●	●	●
OSE	Pendeln Endpunkt	*PGAs*/ Asynchrones Pendeln (OS, OSP1, OSP2, OST1, OST2, OSCTRL, OSNSC, OSE, OSB) (Seite 641)	m	-	-	-	-
OSNSC	Pendeln: Ausfunkanzahl	*PGAs*/ Asynchrones Pendeln (OS, OSP1, OSP2, OST1, OST2, OSCTRL, OSNSC, OSE, OSB) (Seite 641)	m	-	-	-	-
OSOF [4]	Glättung der Werkzeugorientierung AUS	*PGAs*/ Asynchrones Pendeln (OS, OSP1, OSP2, OST1, OST2, OSCTRL, OSNSC, OSE, OSB) (Seite 641)	m	●	●	●	●
OSP1	Pendeln: linker Umkehrpunkt	*PGAs*/ Asynchrones Pendeln (OS, OSP1, OSP2, OST1, OST2, OSCTRL, OSNSC, OSE, OSB) (Seite 641)	m	-	-	-	-
OSP2	Pendeln rechter Umkehrpunkt	*PGAs*/ Asynchrones Pendeln (OS, OSP1, OSP2, OST1, OST2, OSCTRL, OSNSC, OSE, OSB) (Seite 641)	m	-	-	-	-
OSS	Glättung der Werkzeugorientierung am Satzende	*PGAs*/ Werkzeugorientierung (ORIC, ORID, OSOF, OSC, OSS, OSSE, ORIS, OSD, OST) (Seite 423)	m	●	●	●	●
OSSE	Glättung der Werkzeugorientierung am Satzanfang und Satzende	*PGAs*/ Werkzeugorientierung (ORIC, ORID, OSOF, OSC, OSS, OSSE, ORIS, OSD, OST) (Seite 423)	m	●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
OST	Überschleifen der Werkzeugorientierung durch Vorgabe der Winkeltoleranz in Grad mit dem SD (maximale Abweichung vom programmiert. Orientierungsverlauf)	*PGAs*/ Werkzeugorientierung (ORIC, ORID, OSOF, OSC, OSS, OSSE, ORIS, OSD, OST) (Seite 423)	m	●	●	●	●
OST1	Pendeln: Haltepunkt im linken Umkehrpunkt	*PGAs*/ Asynchrones Pendeln (OS, OSP1, OSP2, OST1, OST2, OSCTRL, OSNSC, OSE, OSB) (Seite 641)	m	-	-	-	-
OST2	Pendeln: Haltepunkt im rechten Umkehrpunkt	*PGAs*/ Asynchrones Pendeln (OS, OSP1, OSP2, OST1, OST2, OSCTRL, OSNSC, OSE, OSB) (Seite 641)	m	-	-	-	-
OTOL	Orientierungstoleranz für Kompressor-Funktionen, Orientierungsglättung und Überschleifarten	*PGAs*/ Programmierbare Kontur-/Orientierungtoleranz (CTOL, OTOL, ATOL) (Seite 491)		-	●	-	●
OVR	Drehzahlkorrektur	*PGAs*/	m	●	●	●	●
OVRA	Axiale Drehzahlkorrektur	*PGAs*/	m	●	●	●	●
OVRRAP	Eilgang-Korrektur	*PGAs*/	m	●	●	●	●
P	Anzahl Unterprogramm-durchläufe	*PGAs*/ Anzahl der Programmwiederholungen (P) (Seite 183)		●	●	●	●
PAROT	Werkstückkoordinatensystem am Werkstück ausrichten	*PGs*/	m	●	●	●	●
PAROTOF	Werkstückbezogene Frame-Drehung ausschalten	*PGs*/	m	●	●	●	●
PCALL	Unterprogramme mit absoluter Pfadangabe und Parameterübergabe aufrufen	*PGAs*/ Unterprogramm mit Pfadangabe und Parametern aufrufen (PCALL) (Seite 191)		●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
PDELAYOF	Verzögerung beim Stanzen AUS	*PGAs/* Stanzen und Nibbeln ein oder aus (SPOF, SON, PON, SONS, PONS, PDELAYON, PDELAYOF, PUNCHACC) (Seite 655)	m	-	-	-	-
PDELAYON [4]	Verzögerung beim Stanzen EIN	*PGAs/* Stanzen und Nibbeln ein oder aus (SPOF, SON, PON, SONS, PONS, PDELAYON, PDELAYOF, PUNCHACC) (Seite 655)	m	-	-	-	-
PHU	Physikalische Einheit einer Variablen	*PGAs/* Definition von Anwendervariablen (DEF) (Seite 22)		●	●	●	●
PL	1. B-Spline: Knotenabstand 2. Polynom-Interpolation: Länge des Parameterintervalls bei Polynom-Interpolation	*PGAs/* 1. Spline-Interpolation (ASPLINE, BSPLINE, CSPLINE, BAUTO, BNAT, BTAN, EAUTO, ENAT, ETAN, PW, SD, PL) (Seite 233) 2. Polynom-Interpolation (POLY, POLYPATH, PO, PL) (Seite 250)	s	- -	○ -	- -	○ -
PM	pro Minute	*PGs/*		●	●	●	●
PO	Polynomkoeffizient bei Polynom-Interpolation	*PGAs/* Polynom-Interpolation (POLY, POLYPATH, PO, PL) (Seite 250)	s	-	-	-	-
POCKET3	Fräszyklus, Rechtecktasche (beliebiger Fräser)	*BHDs//BHFs/*		●	●	●	●
POCKET4	Fräszyklus Kreistasche (beliebiger Fräser)	*BHDs//BHFs/*		●	●	●	●
POLF	Rückzugsposition LIFTFAST	*PGs//PGAs/*	m	●	●	●	●
POLFA	Rückzugsposition von Einzelachsen mit $AA_ESR_TRIGGER starten	*PGs/*	m	●	●	●	●
POLFMASK	Achsen für den Rückzug ohne Zusammenhang zwischen den Achsen freigeben	*PGs/*	m	●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
POLFMLIN	Achsen für den Rückzug mit linearen Zusammenhang zwischen den Achsen freigeben	*PGs/*	m	●	●	●	●
POLY	Polynom-Interpolation	*PGAs/* Polynom-Interpolation (POLY, POLYPATH, PO, PL) (Seite 250)	m	-	-	-	-
POLYPATH	Polynom-Interpolation selektierbar für die Achsgruppen AXIS oder VECT	*PGAs/* Polynom-Interpolation (POLY, POLYPATH, PO, PL) (Seite 250)	m	-	-	-	-
PON	Stanzen EIN	*PGAs/* Stanzen und Nibbeln ein oder aus (SPOF, SON, PON, SONS, PONS, PDELAYON, PDELAYOF, PUNCHACC) (Seite 655)	m	-	-	-	-
PONS	Stanzen EIN im IPO-Takt	*PGAs/* Stanzen und Nibbeln ein oder aus (SPOF, SON, PON, SONS, PONS, PDELAYON, PDELAYOF, PUNCHACC) (Seite 655)	m	-	-	-	-
POS	Achse positionieren	*PGs/*		●	●	●	●
POSA	Achse positionieren über Satzgrenze	*PGs/*		●	●	●	●
POSM	Magazin positionieren	*FBW*		●	●	●	●
POSP	Positionieren in Teilstücken (Pendeln)	*PGs/*		●	●	●	●
POSRANGE	Ermitteln, ob sich die aktuell interpolierte Sollposition einer Achse in einem Fenster um eine vorgegebene Referenzposition befindet	*PGAs/* Position im vorgegebenen Referenzbereich (POSRANGE) (Seite 601)		●	●	●	●
POT	Quadrat (Arithmetische Funktion)	*PGAs/* Rechenfunktionen (Seite 61)		●	●	●	●
PR	Pro Umdrehung	*PGs/*		●	●	●	●
PREPRO	Unterprogramme mit Vorbereitung kennzeichnen	*PGAs/* Unterprogramme mit Vorbereitung kennzeichnen (PREPRO) (Seite 168)		●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
PRESETON	Istwertsetzen für programmierte Achsen	*PGAs*/ Preset-Verschiebung (PRESETON) (Seite 296)		●	●	●	●
PRIO	Schlüsselwort zum Setzen der Priorität bei der Behandlung von Interrupts	*PGAs*/ Interruptroutine zuordnen und starten (SETINT, PRIO, BLSYNC) (Seite 111)		●	●	●	●
PROC	Erste Anweisung eines Programms	*PGAs*/ Unterprogramm mit Pfadangabe und Parametern aufrufen (PCALL) (Seite 191)		●	●	●	●
PTP	Punkt-zu-Punkt-Bewegung	*PGAs*/ Kartesisches PTP-Fahren (Seite 376)	m	●	●	●	●
PTPG0	Punkt-zu-Punkt-Bewegung nur bei G0, sonst CP	*PGAs*/ PTP bei TRANSMIT (Seite 381)	m	●	●	●	●
PUNCHACC	Wegabhängige Beschleunigung beim Nibbeln	*PGAs*/ Stanzen und Nibbeln ein oder aus (SPOF, SON, PON, SONS, PONS, PDELAYON, PDELAYOF, PUNCHACC) (Seite 655)		-	-	-	-
PUTFTOC	Werkzeugfeinkorrektur für paralleles Abrichten	*PGAs*/ Online-Werkzeugkorrektur (PUTFTOCF, FCTDEF, PUTFTOC, FTOCON, FTOCOF) (Seite 404)		●	●	●	●
PUTFTOCF	Werkzeugfeinkorrektur in Abhängigkeit einer mit FCTDEF festgelegten Funktion für paralleles Abrichten	*PGAs*/ Online-Werkzeugkorrektur (PUTFTOCF, FCTDEF, PUTFTOC, FTOCON, FTOCOF) (Seite 404)		●	●	●	●
PW	B-Spline, Punkt-Gewicht	*PGAs*/ Spline-Interpolation (ASPLINE, BSPLINE, CSPLINE, BAUTO, BNAT, BTAN, EAUTO, ENAT, ETAN, PW, SD, PL) (Seite 233)	s	-	○	-	○
QECLRNOF	Quadrantenfehler-kompensation lernen AUS	*PGAs*/ Kompensationskennlinien einlernen (QECLRNON, QECLRNOF) (Seite 689)		●	●	●	●
QECLRNON	Quadrantenfehler-kompensation lernen EIN	*PGAs*/ Kompensationskennlinien einlernen (QECLRNON, QECLRNOF) (Seite 689)		●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
QU	Schnelle Zusatz-(Hilfs-)funktionsausgabe	*PGs/*		•	•	•	•
R...	Rechenparameter auch als einstellbarer Adressbezeichner und mit numerischer Erweiterung	*PGAs/* Vordefinierte Anwendervariablen: Rechenparameter (R) (Seite 18)		•	•	•	•
RAC	Absolut satzweise achsspezifische Radiusprogrammierung	*PGs/*	s	•	•	•	•
RDISABLE	Einlesesperre	*PGAs/* Einlesesperre setzen (RDISABLE) (Seite 580)		•	•	•	•
READ	Liest in der angegebenen Datei eine oder mehrere Zeilen ein und legt gelesene Informationen im Feld ab	*PGAs/* Zeilen in Datei lesen (READ) (Seite 134)		•	•	•	•
REAL	Datentyp: Gleitpunktvariable mit Vorzeichen (reale Zahlen)	*PGAs/* Definition von Anwendervariablen (DEF) (Seite 22)		•	•	•	•
REDEF	Einstellung für Maschinendaten, NC-Sprachelemente und Systemvariablen, bei welchen Benutzergruppen sie angezeigt werden	*PGAs/* Redefinition von Systemvariablen, Anwendervariablen und NC-Sprachbefehlen (REDEF) (Seite 28)		•	•	•	•
RELEASE	Maschinenachsen zum Achstausch freigeben	*PGAs/* Achstausch, Spindeltausch (RELEASE, GET, GETD) (Seite 121)		•	•	•	•
REP	Schlüsselwort zur Initialisierung aller Elemente eines Feldes mit demselben Wert	*PGAs/* Definition und Initialisierung von Feldvariablen (DEF, SET, REP) (Seite 44)		•	•	•	•
REPEAT	Wiederholung einer Programmschleife	*PGAs/* Programmteilwiederholung (REPEAT, REPEATB, ENDLABEL, P) (Seite 88)		•	•	•	•
REPEATB	Wiederholung einer Programmzeile	*PGAs/* Programmteilwiederholung (REPEAT, REPEATB, ENDLABEL, P) (Seite 88)		•	•	•	•

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
REPOSA	Wiederanfahren an die Kontur linear mit allen Achsen	*PGAs*/ Wiederanfahren an Kontur (REPOSA, REPOSL, REPOSQ, REPOSQA, REPOSH, REPOSHA, DISR, DISPR, RMI, RMB, RME, RMN) (Seite 476)	s	●	●	●	●
REPOSH	Wiederanfahren an die Kontur mit Halbkreis	*PGAs*/ Wiederanfahren an Kontur (REPOSA, REPOSL, REPOSQ, REPOSQA, REPOSH, REPOSHA, DISR, DISPR, RMI, RMB, RME, RMN) (Seite 476)	s	●	●	●	●
REPOSHA	Wiederanfahren an die Kontur mit allen Achsen; Geometrieachsen im Halbkreis	*PGAs*/ Wiederanfahren an Kontur (REPOSA, REPOSL, REPOSQ, REPOSQA, REPOSH, REPOSHA, DISR, DISPR, RMI, RMB, RME, RMN) (Seite 476)	s	●	●	●	●
REPOSL	Wiederanfahren an die Kontur linear	*PGAs*/ Wiederanfahren an Kontur (REPOSA, REPOSL, REPOSQ, REPOSQA, REPOSH, REPOSHA, DISR, DISPR, RMI, RMB, RME, RMN) (Seite 476)	s	●	●	●	●
REPOSQ	Wiederanfahren an die Kontur im Viertelkreis	*PGAs*/ Wiederanfahren an Kontur (REPOSA, REPOSL, REPOSQ, REPOSQA, REPOSH, REPOSHA, DISR, DISPR, RMI, RMB, RME, RMN) (Seite 476)	s	●	●	●	●
REPOSQA	Wiederanfahren an die Kontur linear mit allen Achsen; Geometrieachsen im Viertelkreis	*PGAs*/ Wiederanfahren an Kontur (REPOSA, REPOSL, REPOSQ, REPOSQA, REPOSH, REPOSHA, DISR, DISPR, RMI, RMB, RME, RMN) (Seite 476)	s	●	●	●	●
RESET	Technologiezyklus rücksetzen	*PGAs*/ Sperren, Freischalten, Zurücksetzen (LOCK, UNLOCK, RESET) (Seite 634)		●	●	●	●
RESETMON	Sprachbefehl zur Sollwertaktivierung	*FBW*		●	●	●	●
RET	Unterprogrammende	*PGAs*/ Parametrierbarer Unterprogrammrücksprung (RET ...) (Seite 171)		●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
RIC	Relativ satzweise achsspezifische Radiusprogrammierung	*PGs*/ Achstausch, Spindeltausch (RELEASE, GET, GETD) (Seite 121)	s	●	●	●	●
RINDEX	Index eines Zeichens im Eingangsstring bestimmen	*PGAs*/ Zeichen/String in String suchen (INDEX, RINDEX, MINDEX, MATCH) (Seite 78)		●	●	●	●
RMB	Wiederanfahren an Satzanfangspunkt	*PGAs*/ Wiederanfahren an Kontur (REPOSA, REPOSL, REPOSQ, REPOSQA, REPOSH, REPOSHA, DISR, DISPR, RMI, RMB, RME, RMN) (Seite 476)	m	●	●	●	●
RME	Wiederanfahren an Satzendpunkt	*PGAs*/ Wiederanfahren an Kontur (REPOSA, REPOSL, REPOSQ, REPOSQA, REPOSH, REPOSHA, DISR, DISPR, RMI, RMB, RME, RMN) (Seite 476)	m	●	●	●	●
RMI [4]	Wiederanfahren an Unterbrechungspunkt	*PGAs*/ Wiederanfahren an Kontur (REPOSA, REPOSL, REPOSQ, REPOSQA, REPOSH, REPOSHA, DISR, DISPR, RMI, RMB, RME, RMN) (Seite 476)	m	●	●	●	●
RMN	Wiederanfahren an nächstliegenden Bahnpunkt	*PGAs*/ Wiederanfahren an Kontur (REPOSA, REPOSL, REPOSQ, REPOSQA, REPOSH, REPOSHA, DISR, DISPR, RMI, RMB, RME, RMN) (Seite 476)	m	●	●	●	●
RND	Konturecke verrunden	*PGs*/	s	●	●	●	●
RNDM	Modales Verrunden	*PGs*/	m	●	●	●	●
ROT	Programmierbare Drehung	*PGs*/	s	●	●	●	●
ROTS	Programmierbare Frame-Drehungen mit Raumwinkeln	*PGs*/	s	●	●	●	●
ROUND	Runden der Nachkommastellen	*PGAs*/ Rechenfunktionen (Seite 61)		●	●	●	●
ROUNDUP	Aufrunden eines Eingabewerts	*PGAs*/ Aufrunden (ROUNDUP) (Seite 144)		●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
RP	Polarradius	*PGs/*	m/s	●	●	●	●
RPL	Drehung in der Ebene	*PGs/*	s	●	●	●	●
RT	Parameter für Zugriff auf Framedaten: Drehung	*PGAs/* Framekomponenten lesen und verändern (TR, FI, RT, SC, MI) (Seite 289)		●	●	●	●
RTLIOF	G0 ohne Linearinterpolation (Einzelachsinterpolation)	*PGs/*	m	●	●	●	●
RTLION	G0 mit Linearinterpolation	*PGs/*	m	●	●	●	●
S	Spindeldrehzahl (bei G4, G96/G961 andere Bedeutung)	*PGs/*	m/s	●	●	●	●
SAVE	Attribut zur Rettung von Informationen bei Unterprogrammaufrufen	*PGAs/* Modale G-Funktionen sichern (SAVE) (Seite 157)		●	●	●	●
SBLOF	Einzelsatz unterdrücken	*PGAs/* Einzelsatzbearbeitung unterdrücken (SBLOF, SBLON) (Seite 158)		●	●	●	●
SBLON	Einzelsatzunterdrückung aufheben	*PGAs/* Einzelsatzbearbeitung unterdrücken (SBLOF, SBLON) (Seite 158)		●	●	●	●
SC	Parameter für Zugriff auf Framedaten: Skalierung	*PGAs/* Framekomponenten lesen und verändern (TR, FI, RT, SC, MI) (Seite 289)		●	●	●	●
SCALE	Programmierbare Skalierung	*PGs/*	s	●	●	●	●
SCC	Selektive Zuordnung einer Planachse zu G96/G961/G962. Achsbezeichner können Geo-, Kanal oder Maschinenachse sein.	*PGs/*		●	●	●	●
SCPARA	Servo-Parameterersatz programmieren	*PGAs/* Programmierbarer Servo-Parametersatz (SCPARA) (Seite 278)		●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
SD	Spline-Grad	*PGAsl* Spline-Interpolation (ASPLINE, BSPLINE, CSPLINE, BAUTO, BNAT, BTAN, EAUTO, ENAT, ETAN, PW, SD, PL) (Seite 233)	s	-	○	-	○
SEFORM	Strukturierungs-anweisung im Stepeditor, um daraus die Schrittansicht für HMI Advanced zu generieren	*PGAsl* Strukturierungsanweisung im Stepeditor (SEFORM) (Seite 213)		●	●	●	●
SET	Schlüsselwort zur Initialisierung aller Elemente eines Feldes mit aufgelisteten Werten	*PGAsl* Definition und Initialisierung von Feldvariablen (DEF, SET, REP) (Seite 44)		●	●	●	●
SETAL	Alarm setzen	*PGAsl* Alarme (SETAL) (Seite 699)		●	●	●	●
SETDNO	D-Nummer der Schneide (CE) eines Werkzeugs (T) zuordnen	*PGAsl* Freie D-Nummernvergabe: D-Nummern umbenennen (GETDNO, SETDNO) (Seite 431)		●	●	●	●
SETINT	Festlegung, welche Interruptroutine aktiviert werden soll, wenn ein NCK-Eingang ansteht	*PGAsl* Interruptroutine zuordnen und starten (SETINT, PRIO, BLSYNC) (Seite 111)		●	●	●	●
SETM	Setzen von Markern im eigenen Kanal	PGAsl Programmkoordinierung (INIT, START, WAITM, WAITMC, WAITE, SETM, CLEARM) (Seite 103)		-	-	-	-
SETMS	Zurückschalten auf die im Maschinendatum festgelegte Masterspindel			●	●	●	●
SETMS(n)	Spindel n soll als Masterspindel gelten	*PGsl*		●	●	●	●
SETMTH	Masterwerkzeughalter-nummer setzen	*FBW*		●	●	●	●
SETPIECE	Stückzahl für alle Werkzeuge berücksichtigen, die der Spindel zugeordnet sind	*FBW*		●	●	●	●
SETTA	Werkzeug aus Verschleißverbund aktiv setzen	*FBW*		●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
SETTCOR	Veränderung von Werkzeugkomponenten unter Berücksichtigung aller Randbedingungen	*FB1(W1)*		●	●	●	●
SETTIA	Werkzeug aus Verschleißverbund inaktiv setzen	*FBW*		●	●	●	●
SF	Startpunktversatz für Gewindeschneiden	*PGs/*	m	●	●	●	●
SIN	Sinus (Trigon. Funktion)	*PGAs/* Rechenfunktionen (Seite 61)		●	●	●	●
SIRELAY	Die mit SIRELIN, SIRELOUT und SIRELTIME parametrierten Sicherheitfunktionen aktivieren	*FBSIs/*		-	-	-	-
SIRELIN	Eingangsgrößen des Funktionsbausteins initialisieren	*FBSIs/*		-	-	-	-
SIRELOUT	Ausgangsgrößen des Funktionsbausteins initialisieren	*FBSIs/*		-	-	-	-
SIRELTIME	Timer des Funktionsbausteins initialisieren	*FBSIs/*		-	-	-	-
SLOT1	Fräsbildzyklus, Nuten auf einem Kreis	*BHDs//BHFs/*		●	●	●	●
SLOT2	Fräsbildzyklus Kreisnut	*BHDs//BHFs/*		●	●	●	●
SOFT	Ruckbegrenzte Bahnbeschleunigung	*PGs/*	m	●	●	●	●
SOFTA	Ruckbegrenzte Achsbeschleunigung für die programmierten Achsen einschalten	*PGs/*		●	●	●	●
SON	Nibbeln EIN	*PGAs/* Stanzen und Nibbeln ein oder aus (SPOF, SON, PON, SONS, PONS, PDELAYON, PDELAYOF, PUNCHACC) (Seite 655)	m	-	-	-	-
SONS	Nibbeln EIN im IPO-Takt	*PGAs/* Stanzen und Nibbeln ein oder aus (SPOF, SON, PON, SONS, PONS, PDELAYON, PDELAYOF, PUNCHACC) (Seite 655)	m	-	-	-	-

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
SPATH [4]	Bahnbezug für FGROUP-Achsen ist Bogenlänge	*PGAs/* Einstellbarer Bahnbezug (SPATH, UPATH) (Seite 256)	m	●	●	●	●
SPCOF	Masterspindel oder Spindel (n) von Lageregelung in Drehzahlregelung umschalten	*PGs/*	m	●	●	●	●
SPCON	Masterspindel oder Spindel (n) von Drehzahlregelung in Lageregelung umschalten	*PGAs/*	m	●	●	●	●
SPI	Konvertiert Spindelnummer in Achsbezeichner	*PGAs/* Achsfunktionen (AXNAME, AX, SPI, AXTOSPI, ISAXIS, AXSTRING, MODAXVAL) (Seite 669)		●	●	●	●
SPIF1 [4]	Schnelle NCK-Ein-/Ausgänge für Stanzen/Nibbeln Byte 1	*FB2(N4)*	m	-	-	-	-
SPIF2	Schnelle NCK-Ein-/Ausgänge für Stanzen/Nibbeln Byte 2	*FB2(N4)*	m	-	-	-	-
SPLINEPATH	Spline-Verband festlegen	*PGAs/* Spline-Verbund (SPLINEPATH) (Seite 245)		-	○	-	○
SPN	Anzahl der Teilstrecken pro Satz	*PGAs/* Automatische Wegaufbereitung (Seite 660)	s	-	-	-	-
SPOF [4]	Hub AUS, Stanzen, Nibbeln AUS	*PGAs/* Stanzen und Nibbeln ein oder aus (SPOF, SON, PON, SONS, PONS, PDELAYON, PDELAYOF, PUNCHACC) (Seite 655)	m	-	-	-	-
SPOS	Spindelposition	*PGs/*	m	●	●	●	●
SPOSA	Spindelposition über Satzgrenzen hinweg	*PGs/*	m	●	●	●	●
SPP	Länge einer Teilstrecke	*PGAs/* Automatische Wegaufbereitung (Seite 660)	m	-	-	-	-
SQRT	Quadratwurzel (arithmetische Funktion) (square root)	*PGAs/* Rechenfunktionen (Seite 61)		●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
SR	Pendelrückzugsweg für Synchronaktion	*PGs/*	s	-	-	-	-
SRA	Pendelrückzugsweg bei externem Eingang axial für Synchronaktion	*PGs/*	m	-	-	-	-
ST	Pendelausfeuerzeit für Synchronaktion	*PGs/*	s	-	-	-	-
STA	Pendelausfeuerzeit axial für Synchronaktion	*PGs/*	m	-	-	-	-
START	Starten der ausgewählten Programme in mehreren Kanälen gleichzeitig aus dem laufenden Programm	*PGAs/* Programmkoordinierung (INIT, START, WAITM, WAITMC, WAITE, SETM, CLEARM) (Seite 103)		-	-	-	-
STARTFIFO [4]	Abarbeiten; parallel dazu Auffüllen des Vorlaufpuffers	*PGAs/* Programmablauf mit Vorlaufspeicher (STOPFIFO, STARTFIFO, FIFOCTRL, STOPRE) (Seite 465)	m	●	●	●	●
STAT	Stellung der Gelenke	*PGAs/* Kartesisches PTP-Fahren (Seite 376)	s	●	●	●	●
STOLF	G0-Toleranzfaktor	*PGAs/* Toleranz bei G0-Bewegungen (STOLF) (Seite 495)	m	-	-	-	-
STOPFIFO	Anhalten der Bearbeitung; Auffüllen des Vorlaufpuffers, bis STARTFIFO erkannt wird, Vorlaufpuffer voll oder Programmende	*PGAs/* Programmablauf mit Vorlaufspeicher (STOPFIFO, STARTFIFO, FIFOCTRL, STOPRE) (Seite 465)	m	●	●	●	●
STOPRE	Vorlaufstopp, bis alle vorbereiteten Sätze vom Hauptlauf abgearbeitet sind	*PGAs/* Programmablauf mit Vorlaufspeicher (STOPFIFO, STARTFIFO, FIFOCTRL, STOPRE) (Seite 465)		●	●	●	●
STOPREOF	Vorlaufstopp aufheben	*PGAs/* Vorlaufstopp aufheben (STOPREOF) (Seite 581)		●	●	●	●
STRING	Datentyp: Zeichenkette	*PGAs/* Definition von Anwendervariablen (DEF) (Seite 22)		●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
STRINGFELD	Selektion eines Einzelzeichens aus dem progr. Springfeld	*PGAs*/ Selektion eines Einzelzeichens (STRINGVAR, STRINGFELD) (Seite 80)		●	●	●	●
STRINGIS	Prüft vorhandenen NC-Sprachumfang und speziell für diesen Befehl gehörende NC-Zyklennamen, Anwendervariablen, Makros und Labelnamen, ob diese existieren, gültig, definiert oder aktiv sind.	*PGAs*/ Vorhandenen NC-Sprachumfang prüfen (STRINGIS) (Seite 683)		●	●	●	●
STRINGVAR	Selektion eines Einzelzeichens aus dem progr. String	*PGAs*/ Selektion eines Einzelzeichens (STRINGVAR, STRINGFELD) (Seite 80)		-	-	-	-
STRLEN	Länge eines Strings bestimmen	*PGAs*/ Länge eines Strings bestimmen (STRLEN) (Seite 78)		●	●	●	●
SUBSTR	Index eines Zeichens im Eingangsstring bestimmen	*PGAs*/ Auswahl eines Teilstrings (SUBSTR) (Seite 80)		●	●	●	●
SUPA	Unterdrückung der aktuellen Nullpunktverschiebung, einschließlich programmierter Verschiebungen, Systemframes, Handradverschiebungen (DRF), externer Nullpunktverschiebung und überlagerte Bewegung	*PGs*/	s	●	●	●	●
SVC	Werkzeug-Schnittgeschwindigkeit	*PGs*/	m	●	●	●	●
SYNFCT	Auswertung eines Polynoms abhängig von einer Bedingung in der Bewegungs-synchronaktion	*PGAs*/ Synchronfunktion (SYNFCT) (Seite 586)		●	●	●	●
SYNR	Lesen der Variable erfolgt synchron, d. h. zum Abarbeitungszeitpunkt	*PGAs*/ Definition von Anwendervariablen (DEF) (Seite 22)		●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
SYNRW	Lesen und Schreiben der Variable erfolgt synchron, d. h. zum Abarbeitungszeitpunkt	*PGAs*/ Definition von Anwendervariablen (DEF) (Seite 22)		●	●	●	●
SYNW	Schreiben der Variable erfolgt synchron, d. h. zum Abarbeitungszeitpunkt	*PGAs*/ Definition von Anwendervariablen (DEF) (Seite 22)		●	●	●	●
T	Werkzeug aufrufen (wechseln nur, wenn im Maschinendatum festgelegt; ansonsten M6-Befehl nötig)	*PGs*/		●	●	●	●
TAN	Tangens (Trigon. Funktion)	*PGAs*/ Rechenfunktionen (Seite 61)		●	●	●	●
TANG	Definition des Achsverbandes Tangentiales Nachführen	*PGAs*/ Tangentialsteuerung (TANG, TANGON, TANGOF, TLIFT, TANGDEL) (Seite 453)		-	-	-	-
TANGDEL	Löschen der Definition des Achsverbandes Tangentiales Nachführen	*PGAs*/ Tangentialsteuerung (TANG, TANGON, TANGOF, TLIFT, TANGDEL) (Seite 453)		-	-	-	-
TANGOF	Tangentielles Nachführen AUS	*PGAs*/ Tangentialsteuerung (TANG, TANGON, TANGOF, TLIFT, TANGDEL) (Seite 453)		-	-	-	-
TANGON	Tangentielles Nachführen EIN	*PGAs*/ Tangentialsteuerung (TANG, TANGON, TANGOF, TLIFT, TANGDEL) (Seite 453)		-	-	-	-
TCA	Werkzeuganwahl / Werkzeugwechsel unabhängig vom Status des Werkzeugs	*FBW*		●	●	●	●
TCARR	Werkzeugträger (Nummer "m") anfordern	*PGAs*/ Werkzeuglängenkorrektur für orientierbare Werkzeugträger (TCARR, TCOABS, TCOFR, TCOFRX, TCOFRY, TCOFRZ) (Seite 439)		-	●	-	●
TCI	Wechsle Werkzeug aus Zwischenspeicher in das Magazin	*FBW*		●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
TCOABS [4]	Werkzeuglängenkomponenten aus der aktuellen Werkzeugorientierung bestimmen	*PGAs*/ Werkzeuglängenkorrektur für orientierbare Werkzeugträger (TCARR, TCOABS, TCOFR, TCOFRX, TCOFRY, TCOFRZ) (Seite 439)	m	-	●	-	●
TCOFR	Werkzeuglängenkomponenten aus der Orientierung des aktiven Frames bestimmen	*PGAs*/ Werkzeuglängenkorrektur für orientierbare Werkzeugträger (TCARR, TCOABS, TCOFR, TCOFRX, TCOFRY, TCOFRZ) (Seite 439)	m	-	●	-	●
TCOFRX	Werkzeugorientierung eines aktiven Frames bei der Werkzeugwahl bestimmen, Werkzeug zeigt in X-Richtung	*PGAs*/ Werkzeuglängenkorrektur für orientierbare Werkzeugträger (TCARR, TCOABS, TCOFR, TCOFRX, TCOFRY, TCOFRZ) (Seite 439)	m	-	●	-	●
TCOFRY	Werkzeugorientierung eines aktiven Frames bei der Werkzeugwahl bestimmen, Werkzeug zeigt in Y-Richtung	*PGAs*/ Werkzeuglängenkorrektur für orientierbare Werkzeugträger (TCARR, TCOABS, TCOFR, TCOFRX, TCOFRY, TCOFRZ) (Seite 439)	m	-	●	-	●
TCOFRZ	Werkzeugorientierung eines aktiven Frames bei der Werkzeugwahl bestimmen, Werkzeug zeigt in Z-Richtung	*PGAs*/ Werkzeuglängenkorrektur für orientierbare Werkzeugträger (TCARR, TCOABS, TCOFR, TCOFRX, TCOFRY, TCOFRZ) (Seite 439)	m	-	●	-	●
THETA	Drehwinkel	*PGAs*/ Drehungen der Werkzeugorientierung (ORIROTA, ORIROTR, ORIROTT, ORIROTC, THETA) (Seite 343)	s	●	●	●	●
TILT	Seitwärtswinkel	*PGAs*/ Programmierung der Werkzeugorientierung (A..., B..., C..., LEAD, TILT) (Seite 323)	m	●	●	●	●
TLIFT	Bei Tangentialsteuerung Zwischensatz an Konturecken einfügen	*PGAs*/ Tangentialsteuerung (TANG, TANGON, TANGOF, TLIFT, TANGDEL) (Seite 453)		-	-	-	-

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
TMOF	Werkzeugüberwachung abwählen	*PGAs*/ Schleifenspezifische Werkzeugüberwachung im Teileprogramm (TMON, TMOF) (Seite 667)		●	●	●	●
TMON	Werkzeugüberwachung anwählen	*PGAs*/ Schleifenspezifische Werkzeugüberwachung im Teileprogramm (TMON, TMOF) (Seite 667)		●	●	●	●
TO	Bezeichnet den Endwert in einer FOR-Zählschleife	*PGAs*/ Zählschleife (FOR ... TO ..., ENDFOR) (Seite 99)		●	●	●	●
TOFF	Werkzeuglängen-Offset in Richtung der Werkzeuglängenkomponente, die parallel zu der im Index angegebenen Geometrieachse wirkt.	*PGs*/	m	●	●	●	●
TOFFL	Werkzeuglängen-Offset in Richtung der Werkzeuglängen-komponente L1, L2 bzw. L3	*PGs*/	m	●	●	●	●
TOFFOF	Online-Werkzeuglängen-korrektur rücksetzen	*PGAs*/ Online-Werkzeuglängenkorrektur (TOFFON, TOFFOF) (Seite 443)		●	●	●	●
TOFFON	Online-Werkzeuglängen-korrektur aktivieren	*PGAs*/ Online-Werkzeuglängenkorrektur (TOFFON, TOFFOF) (Seite 443)		●	●	●	●
TOFFR	Werkzeugradius-Offset	*PGs*/	m	●	●	●	●
TOFRAME	Z-Achse des WKS durch Frame-Drehung parallel zur Werkzeugorientierung ausrichten	*PGs*/	m	●	●	●	●
TOFRAMEX	X-Achse des WKS durch Frame-Drehung parallel zur Werkzeugorientierung ausrichten	*PGs*/	m	●	●	●	●
TOFRAMEY	Y-Achse des WKS durch Frame-Drehung parallel zur Werkzeugorientierung ausrichten	*PGs*/	m	●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
TOFRAMEZ	wie TOFRAME	*PGs/*	m	●	●	●	●
TOLOWER	Buchstaben eines Strings in Kleinbuchstaben wandeln	*PGAs/* Wandlung in Klein-/Großbuchstaben (TOLOWER, TOUPPER) (Seite 77)		●	●	●	●
TOOLENV	Alle aktuellen Zustände speichern, die für die Bewertung der im Speicher abgelegten Werkzeugdaten von Bedeutung sind	*FB1(W1)*		●	●	●	●
TOROT	Z-Achse des WKS durch Frame-Drehung parallel zur Werkzeugorientierung ausrichten	*PGs/*	m	●	●	●	●
TOROTOF	Framedrehungen in Werkzeugrichung AUS	*PGs/*	m	●	●	●	●
TOROTX	X-Achse des WKS durch Frame-Drehung parallel zur Werkzeugorientierung ausrichten	*PGs/*	m	●	●	●	●
TOROTY	Y-Achse des WKS durch Frame-Drehung parallel zur Werkzeugorientierung ausrichten	*PGs/*	m	●	●	●	●
TOROTZ	wie TOROT	*PGs/*	m	●	●	●	●
TOUPPER	Buchstaben eines Strings in Großbuchstaben wandeln	*PGAs/* Wandlung in Klein-/Großbuchstaben (TOLOWER, TOUPPER) (Seite 77)		●	●	●	●
TOWBCS	Verschleißwerte im Basiskoordinatensystem (BKS)	*PGAs/* Koordinatensystem der aktiven Bearbeitung (TOWSTD, TOWMCS, TOWWCS, TOWBCS, TOWTCS, TOWKCS) (Seite 400)	m	-	●	-	●
TOWKCS	Verschleißwerte im Koordinatensystem des Werkzeugkopfes bei kinetischer Transformation (unterscheidet sich vom MKS durch Werkzeugdrehung)	*PGAs/* Koordinatensystem der aktiven Bearbeitung (TOWSTD, TOWMCS, TOWWCS, TOWBCS, TOWTCS, TOWKCS) (Seite 400)	m	-	●	-	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
TOWMCS	Verschleißwerte im Maschinen-Koordinatensystem (MKS)	*PGAs*/ Koordinatensystem der aktiven Bearbeitung (TOWSTD, TOWMCS, TOWWCS, TOWBCS, TOWTCS, TOWKCS) (Seite 400)	m	-	●	-	●
TOWSTD	Grundstellungswert für Korrekturen in der Werkzeuglänge	*PGAs*/ Koordinatensystem der aktiven Bearbeitung (TOWSTD, TOWMCS, TOWWCS, TOWBCS, TOWTCS, TOWKCS) (Seite 400)	m	-	●	-	●
TOWTCS	Verschleißwerte im Werkzeug-Koordinatensystem (Werkzeugträger-bezugspunkt T an der Werkzeughalter-aufnahme)	*PGAs*/ Koordinatensystem der aktiven Bearbeitung (TOWSTD, TOWMCS, TOWWCS, TOWBCS, TOWTCS, TOWKCS) (Seite 400)	m	-	●	-	●
TOWWCS	Verschleißwerte im Werkstück-Koordinatensystem (WKS)	*PGAs*/ Koordinatensystem der aktiven Bearbeitung (TOWSTD, TOWMCS, TOWWCS, TOWBCS, TOWTCS, TOWKCS) (Seite 400)	m	-	●	-	●
TR	Verschiebungs-komponente einer Frame-Variablen	*PGAs*/ Framekomponenten lesen und verändern (TR, FI, RT, SC, MI) (Seite 289)		●	●	●	●
TRAANG	Transformation schräge Achse	*PGAs*/ Schräge Achse (TRAANG) (Seite 371)		-	-	○	-
TRACON	Kaskadierte Transformation	*PGAs*/ Verkettete Transformationen (TRACON, TRAFOOF) (Seite 387)		-	-	○	-
TRACYL	Zylinder: Mantelflächen-Transformation	*PGAs*/ Zylindermanteltransformation (TRACYL) (Seite 362)		○	○	○	○
TRAFOOF	Im Kanal aktive Transformationen ausschalten	*PGAs*/ Verkettete Transformationen (TRACON, TRAFOOF) (Seite 387)		●	●	●	●
TRAILOF	Achssynchrones Mitschleppen AUS	*PGAs*/ Mitschleppen (TRAILON, TRAILOF) (Seite 497)		●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
TRAILON	Achssynchrones Mitschleppen EIN	*PGAs*/ Mitschleppen (TRAILON, TRAILOF) (Seite 497)		●	●	●	●
TRANS	Programmierbare Verschiebung	*PGs*/	s	●	●	●	●
TRANSMIT	Polar-Transformation (Stirnflächenbearbeitung)	*PGAs*/ Fräsbearbeitung an Drehteilen (TRANSMIT) (Seite 359)		○	○	○	○
TRAORI	4-, 5-Achstransformation, Generische Transformation	*PGAs*/ Drei, Vier, und Fünf- Achs-Transformation (TRAORI) (Seite 320)		-	●	-	●
TRUE	Logische Konstante: wahr	*PGAs*/ Definition von Anwendervariablen (DEF) (Seite 22)		●	●	●	●
TRUNC	Abschneiden der Nachkommastellen	*PGAs*/ Genauigkeitskorrektur bei Vergleichsfehlern (TRUNC) (Seite 66)		●	●	●	●
TU	Achswinkel	*PGAs*/ Kartesisches PTP-Fahren (Seite 376)	s	●	●	●	●
TURN	Windungsanzahl für Schraubenlinie	*PGs*/	s	●	●	●	●
ULI	Oberer Grenzwert von Variablen	*PGAs*/ Attribut: Grenzwerte (LLI, ULI) (Seite 34)		●	●	●	●
UNLOCK	Synchronaktion mit ID freigeben (Technologiezyklus fortsetzen)	*PGAs*/ Sperren, Freischalten, Zurücksetzen (LOCK, UNLOCK, RESET) (Seite 634)		●	●	●	●
UNTIL	Bedingung zur Beendigung einer REPEAT-Schleife	*PGAs*/ Programmschleife mit Bedingung am Schleifenanfang (WHILE, ENDWHILE) (Seite 100)		●	●	●	●
UPATH	Bahnbezug für FGROUP-Achsen ist Kurvenparameter	*PGAs*/ Einstellbarer Bahnbezug (SPATH, UPATH) (Seite 256)	m	●	●	●	●
VAR	Schlüsselwort: Art der Parameterübergabe	*PGAs*/ Unterprogrammaufruf mit Parameterübergabe (EXTERN) (Seite 180)		●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
VELOLIM	Reduktion der maximalen axialen Geschwindigkeit	*PGAs/* Prozentuale Geschwindigkeitskorrektur (VELOLIM) (Seite 487)	m	●	●	●	●
VELOLIMA	Reduktion oder Überhöhung der maximalen axialen Geschwindigkeit der Folgeachse	*PGs/*	m	●	●	●	●
WAITC	Warten, bis Kopplungssatzwechsel-kriterium für die Achsen/Spindeln erfüllt ist	*PGAs/* Programmkoordinierung (INIT, START, WAITM, WAITMC, WAITE, SETM, CLEARM) (Seite 103)		-	-	○	-
WAITE	Warten auf das Programmende in einem anderen Kanal.	*PGAs/* Programmkoordinierung (INIT, START, WAITM, WAITMC, WAITE, SETM, CLEARM) (Seite 103)		-	-	-	-
WAITENC	Warten auf synchronisierte bzw. restaurierte Achspositionen	*PGAs/* Warten auf gültige Achsposition (WAITENC) (Seite 681)		-	-	-	-
WAITM	Warten auf Marker im angegebenen Kanal; vorhergehenden Satz mit Genauhalt beenden.	*PGAs/* Programmkoordinierung (INIT, START, WAITM, WAITMC, WAITE, SETM, CLEARM) (Seite 103)		-	-	-	-
WAITMC	Warten auf Marker im angegeben. Kanal; Genauhalt nur, wenn die anderen Kanäle den Marker noch nicht erreicht haben.	*PGAs/* Programmkoordinierung (INIT, START, WAITM, WAITMC, WAITE, SETM, CLEARM) (Seite 103)		-	-	-	-
WAITP	Warten auf Verfahrende der Positionierachse	*PGs/*		●	●	●	●
WAITS	Warten auf Erreichen der Spindelposition	*PGs/*		●	●	●	●
WALCS0	WKS-Arbeitsfeldbegrenzung abgewählt	*PGs/*	m	●	●	●	●
WALCS1	WKS-Arbeitsfeldbegrenzungs-gruppe 1 aktiv	*PGs/*	m	●	●	●	●
WALCS2	WKS-Arbeitsfeldbegrenzungs-gruppe 2 aktiv	*PGs/*	m	●	●	●	●

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
WALCS3	WKS-Arbeitsfeldbegrenzungs-gruppe 3 aktiv	*PGs/*	m	•	•	•	•
WALCS4	WKS-Arbeitsfeldbegrenzungs-gruppe 4 aktiv	*PGs/*	m	•	•	•	•
WALCS5	WKS-Arbeitsfeldbegrenzungs-gruppe 5 aktiv	*PGs/*	m	•	•	•	•
WALCS6	WKS-Arbeitsfeldbegrenzungs-gruppe 6 aktiv	*PGs/*	m	•	•	•	•
WALCS7	WKS-Arbeitsfeldbegrenzungs-gruppe 7 aktiv	*PGs/*	m	•	•	•	•
WALCS8	WKS-Arbeitsfeldbegrenzungs-gruppe 8 aktiv	*PGs/*	m	•	•	•	•
WALCS9	WKS-Arbeitsfeldbegrenzungs-gruppe 9 aktiv	*PGs/*	m	•	•	•	•
WALCS10	WKS-Arbeitsfeldbegrenzungs-gruppe 10 aktiv	*PGs/*	m	•	•	•	•
WALIMOF	BKS-Arbeitsfeldbegrenzung AUS	*PGs/*	m	•	•	•	•
WALIMON [4]	BKS-Arbeitsfeldbegrenzung EIN	*PGs/*	m	•	•	•	•
WHEN	Die Aktion wird zyklisch ausgeführt, wenn die Bedingung erfüllt ist.	*PGAs/* Zyklische Prüfung der Bedingung (WHEN, WHENEVER, FROM, EVERY) (Seite 555)		•	•	•	•
WHENEVER	Die Aktion wird einmal ausgeführt, wenn die Bedingung einmal erfüllt ist.	*PGAs/* Zyklische Prüfung der Bedingung (WHEN, WHENEVER, FROM, EVERY) (Seite 555)		•	•	•	•
WHILE	Beginn der WHILE-Programmschleife	*PGAs/* Programmschleife mit Bedingung am Schleifenanfang (WHILE, ENDWHILE) (Seite 100)		•	•	•	•

Anweisung	Bedeutung	Beschreibung siehe [1]	W [2]	828D [3]			
				PPU260 / 261		PPU280 / 281	
				D	F	D	F
WRITE	Text ins Dateisystem schreiben. Fügt einen Satz am Ende der angegebenen Datei an.	*PGAs*/ Datei schreiben (WRITE) (Seite 129)		•	•	•	•
WRTPR	Verzögert den Bearbeitungsauftrag ohne dabei den Bahnsteuerbetrieb zu unterbrechen	*PGAs*/		•	•	•	•
X	Achsname	*PGs*/	m/s	•	•	•	•
XOR	Logisches Exklusiv-ODER	*PGAs*/ Vergleichs- und logische Operationen (Seite 64)		•	•	•	•
Y	Achsname	*PGs*/	m/s	•	•	•	•
Z	Achsname	*PGs*/	m/s	•	•	•	•

Anhang

A

A.1 Liste der Abkürzungen

A	Ausgang
AS	Automatisierungssystem
ASCII	American Standard Code for Information Interchange: Amerikanische Code-Norm für den Informationsaustausch
ASIC	Application Specific Integrated Circuit: Anwender-Schaltkreis
ASUP	Asynchrones Unterprogramm
AV	Arbeitsvorbereitung
AWL	Anweisungsliste
BA	Betriebsart
BAG	Betriebsartengruppe
BB	Betriebsbereit
BuB, B&B	Bedienen und Beobachten
BCD	Binary Coded Decimals: Im Binärcode verschlüsselte Dezimalzahlen
BHG	Bedienhandgerät
BIN	Binärdateien (**Bin**ary Files)
BIOS	Basic Input Output System
BKS	Basiskoordinatensystem
BOF	Bedienoberfläche
BOT	Boot Files: Bootdateien für SIMODRIVE 611 digital
BT	Bedientafel
BTSS	Bedientafelschnittstelle
CAD	Computer-Aided Design
CAM	Computer-Aided Manufacturing
CNC	Computerized Numerical Control: Computerunterstützte numerische Steuerung
COM	Communication
CP	Communication Processor
CPU	Central Processing Unit: Zentrale Rechnereinheit
CR	Carriage Return
CRT	Cathode Ray Tube: Bildröhre
CSB	Central Service Board: PLC-Baugruppe
CTS	Clear To Send: Meldung der Sendebereitschaft bei seriellen Daten-Schnittstellen
CUTCOM	Cutter radius compensation: Werkzeugradiuskorrektur
DAU	Digital-Analog-Umwandler
DB	Datenbaustein in der PLC
DBB	Datenbausteinbyte in der PLC
DBW	Datenbausteinwort in der PLC

DBX	Datenbausteinbit in der PLC
DC	Direct Control: Bewegung der Rundachse auf kürzestem Weg auf die absolute Position innerhalb einer Umdrehung
DCD	Carrier Detect
DDE	Dynamic Data Exchange
DEE	Datenendeinrichtung
DIN	Deutsche Industrie Norm
DIO	Data Input/Output: Datenübertragungs-Anzeige
DIR	Directory: Verzeichnis
DLL	Dynamic Link Library
DOE	Datenübertragungseinrichtung
DOS	Disk Operating System
DPM	Dual Port Memory
DPR	Dual-Port-RAM
DRAM	Dynamic Random Access Memory
DRF	Differential Resolver Function: Differential-Drehmelder-Funktion (Handrad)
DRY	Dry Run: Probelaufvorschub
DSB	Decoding Single Block: Dekodierungseinzelsatz
DW	Datenwort
E	Eingang
E/A	Ein-/Ausgabe
E/R	Einspeise-/Rückspeiseeinheit (Stromversorgung) des SIMODRIVE 611 digital
EIA-Code	Spezieller Lochstreifencode, Lochanzahl pro Zeichen stets ungerade
ENC	Encoder: Istwertgeber
EPROM	Erasable Programmable Read Only Memory (Löschbarer, elektrisch programmierbarer Lesespeicher)
ERROR	Error from printer
FB	Funktionsbaustein
FBS	Flachbildschirm
FC	Function Call: Funktionsbaustein in der PLC
FDB	Fabrikate-Datenbank
FDD	Floppy Disk Drive
FEPROM	Flash-EPROM: Les- und schreibbarer Speicher
FIFO	First In First Out: Speicher, der ohne Adressangabe arbeitet und dessen Daten in derselben Reihenfolge gelesen werden, in der sie gespeichert wurden.
FIPO	Feininterpolator
FM	Funktionsmodul
FPU	Floating Point Unit: Gleitpunkteinheit
FRA	Frame-Baustein
FRAME	Datensatz (Rahmen)
FRK	Fräsradiuskorrektur
FST	Feed Stop: Vorschub Halt
FUP	Funktionsplan (Programmiermethode für PLC)

GP	Grundprogramm
GUD	Global User Data: Globale Anwenderdaten
HD	Hard Disk: Festplatte
HEX	Kurzbezeichnung für hexadezimale Zahl
HiFu	Hilfsfunktion
HMI	Human Machine Interface: Bedienfunktionalität der SINUMERIK für Bedienen, Programmieren und Simulieren.
HMS	Hochauflösendes Messsystem
HSA	Hauptspindelantrieb
HW	Hardware
IBN	Inbetriebnahme
IF	Impulsfreigabe des Antriebsmoduls
IK (GD)	Implizite Kommunikation (Globale Daten)
IKA	Interpolative Compensation: Interpolatorische Kompensation
IM	Interface-Modul: Anschaltungsbaugruppe
IMR	Interface-Modul Receive: Anschaltungsbaugruppe für Empfangsbetrieb
IMS	Interface-Modul Send: Anschaltungsbaugruppe für Sendebetrieb
INC	Increment: Schrittmaß
INI	Initializing Data: Initialisierungsdaten
IPO	Interpolator
ISA	International Standard Architecture
ISO	International Standard Organization
ISO-Code	Spezieller Lochstreifencode, Lochanzahl pro Zeichen stets gerade
JOG	Jogging: Einrichtbetrieb
K1 .. K4	Kanal 1 bis Kanal 4
K-Bus	Kommunikationsbus
KD	Koordinatendrehung
KOP	Kontaktplan (Programmiermethode für PLC)
K_v	Kreisverstärkungsfaktor
$K_Ü$	Übersetzungsverhältnis
LCD	Liquid-Crystal Display: Flüssigkristallanzeige
LED	Light-Emitting Diode: Leuchtdiodenanzeige
LF	Line Feed
LMS	Lagemesssystem
LR	Lageregler
LUD	Local User Data
MB	Megabyte
MD	Maschinendaten
MDA	Manual Data Automatic: Handeingabe
MK	Messkreis
MKS	Maschinenkoordinatensystem
MLFB	Maschinenlesbare Fabrikatbezeichnung
MPF	Main Program File: NC-Teileprogramm (Hauptprogramm)

MPI	Multi Port Interface: Mehrpunktfähige Schnittstelle
MS-	Microsoft (Software-Hersteller)
MSTT	Maschinensteuertafel
NC	Numerical Control: Numerische Steuerung
NCK	Numerical Control Kernel: Numerik-Kern mit Satzaufbereitung, Verfahrbereich usw.
NCU	Numerical Control Unit: Hardware Einheit des NCK
NRK	Bezeichnung des Betriebssystems des NCK
NST	Nahtstellensignal
NURBS	Non-Uniform Rational B-Spline
NV	Nullpunktverschiebung
OB	Organisationsbaustein in der PLC
OEM	Original Equipment Manufacturer
OP	Operation Panel: Bedieneinrichtung
OPI	Operation Panel Interface: Bedientafel-Anschaltung
OPT	Options: Optionen
OSI	Open Systems Interconnection: Normung für Rechnerkommunikation
P-Bus	Peripheriebus
PC	Personal Computer
PCIN	Name der SW für den Datenaustausch mit der Steuerung
PCMCIA	Personal Computer Memory Card International Association: Speichersteckkarten Normierung
PCU	PC Unit: PC-Box (Rechnereinheit)
PG	Programmiergerät
PLC	Programmable Logic Control: Anpass-Steuerung
POS	Positionier-
RAM	Random Access Memory: Programmspeicher, der gelesen und beschrieben werden kann
REF	Funktion Referenzpunkt anfahren
REPOS	Funktion Repositionieren
RISC	Reduced Instruction Set Computer: Prozessortyp mit kleinem Befehlssatz und schnellem Befehlsdurchsatz
ROV	Rapid Override: Eingangskorrektur
RPA	R-Parameter Active: Speicherbereich in NCK für R- NCK für R-Parameternummern
RPY	Roll Pitch Yaw: Drehungsart eines Koordinatensystems
RTS	Request To Send: Sendeteil einschalten, Steuersignal von seriellen Daten-Schnittstellen
SBL	Single Block: Einzelsatz
SD	Setting-Datum
SDB	System Datenbaustein
SEA	Setting Data Active: Kennzeichnung (Dateityp) für Settingdaten
SFB	System Funktionsbaustein
SFC	System Function Call
SK	Softkey

SKP	Skip: Satz ausblenden
SM	Schrittmotor
SPF	Sub Program File: Unterprogramm
SPS	Speicherprogrammierbare Steuerung
SRAM	Statischer Speicher (gepuffert)
SRK	Schneidenradiuskorrektur
SSFK	Spindelsteigungsfehlerkompensation
SSI	Serial Synchron Interface: Serielle synchrone Schnittstelle
SW	Software
SYF	System Files: Systemdateien
TEA	Testing Data Active: Kennung für Maschinendaten
TO	Tool Offset: Werkzeugkorrektur
TOA	Tool Offset Active: Kennzeichnung (Dateityp) für Werkzeugkorrekturen
TRANSMIT	Transform Milling into Turning: Koordinatenumrechnung an Drehmaschinen für Fräsbearbeitung
UFR	User Frame: Nullpunktverschiebung
UP	Unterprogramm
VSA	Vorschubantrieb
V.24	Serielle Schnittstelle (Definition der Austauschleitungen zwischen DEE und DÜE)
WKS	Werkstückkoordinatensystem
WKZ	Werkzeug
WLK	Werkzeuglängenkorrektur
WOP	Werkstatt orientierte Programmierung
WPD	Work Piece Directory: Werkstückverzeichnis
WRK	Werkzeug-Radius-Korrektur
WZK	Werkzeugkorrektur
WZW	Werkzeugwechsel
ZOA	Zero Offset Active: Kennzeichnung (Dateityp) für Nullpunktverschiebungsdaten
µC	Mikro-Controller

A.2 Feedback zur Dokumentation

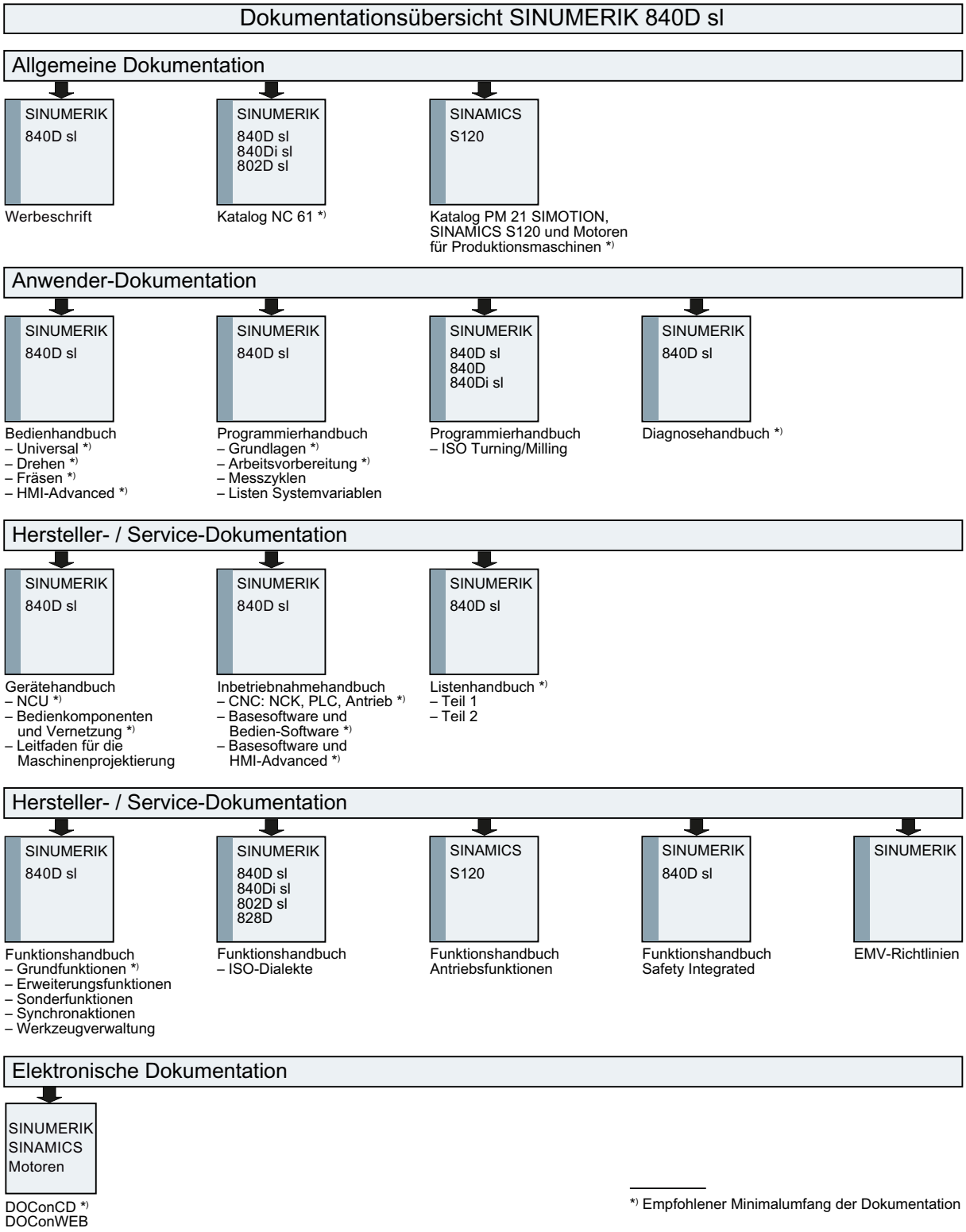
Das vorliegende Dokument wird bezüglich seiner Qualität und Benutzerfreundlichkeit ständig weiterentwickelt. Bitte helfen Sie uns dabei, indem Sie Ihre Anmerkungen und Verbesserungsvorschläge per E-Mail oder Fax senden an:

E-Mail:	mailto:docu.motioncontrol@siemens.com
Fax:	+49 9131 - 98 2176 Bitte verwenden Sie die Faxvorlage auf der Blattrückseite.

An	Absender
SIEMENS AG I DT MC MS1 Postfach 3180 D-91050 Erlangen	Name:
	Anschrift Ihrer Firma / Dienststelle
	Straße:
	PLZ: Ort:
	Telefon: /
Fax.: +49 9131 - 98 2176 (Dokumentation)	Telefax: /

Vorschläge und / oder Korrekturen

A.3 Dokumentationsübersicht



Glossar

Absolutmaß

Angabe des Bewegungsziels einer Achsbewegung durch ein Maß, das sich auf den Nullpunkt des momentan gültigen Koordinatensystems bezieht. Siehe → Kettenmaß.

Achsadresse

Siehe → Achsbezeichner

Achsbezeichner

Achsen werden nach DIN 66217 für ein rechtsdrehendes, rechtwinkliges → Koordinatensystem bezeichnet mit X, Y, Z.

Um X, Y, Z drehende → Rundachsen erhalten die Bezeichner A, B, C. Zusätzliche Achsen, parallel zu den angegebenen, können mit weiteren Adressbuchstaben gekennzeichnet werden.

Achsen

Die CNC-Achsen werden entsprechend ihres Funktionsumfangs abgestuft in:

- Achsen: interpolierende Bahnachsen
- Hilfsachsen: nicht interpolierende Zustell- und Positionierachsen mit achsspezifischem Vorschub. Hilfsachsen sind an der eigentlichen Bearbeitung nicht beteiligt, z. B. Werkzeugzubringer, Werkzeugmagazin.

Achsname

Siehe → Achsbezeichner

Adresse

Eine Adresse ist die Kennzeichnung für einen bestimmten Operanden oder Operandenbereich, z. B. Eingang, Ausgang usw.

Alarme

Alle → Meldungen und Alarme werden auf der Bedientafel im Klartext mit Datum und Uhrzeit und dem entsprechenden Symbol für das Löschkriterium angezeigt. Die Anzeige erfolgt getrennt nach Alarmen und Meldungen.

1. Alarme und Meldungen im Teileprogramm

   Alarme und Meldungen können direkt aus dem Teileprogramm im Klartext zur Anzeige gebracht werden.

2. Alarme und Meldungen von PLC

   Alarme- und Meldungen der Maschine können aus dem PLC-Programm im Klartext zur Anzeige gebracht werden. Dazu sind keine zusätzlichen Funktionsbaustein-Pakete notwendig.

Antrieb

Der Antrieb ist diejenige Einheit der CNC, welche die Drehzahl- und Momentenregelung aufgrund der Vorgaben der NC ausführt.

Anwenderdefinierte Variable

Anwender können für beliebige Nutzung im → Teileprogramm oder Datenbaustein (globale Anwenderdaten) anwenderdefinierte Variablen vereinbaren. Eine Definition enthält eine Datentypangabe und den Variablennamen. Siehe → Systemvariable.

Anwenderprogramm

Anwenderprogramme für Automatisierungssysteme S7-300 werden mit der Programmiersprache STEP 7 erstellt. Das Anwenderprogramm ist modular aufgebaut und besteht aus einzelnen Bausteinen.

Die grundlegenden Bausteintypen sind:

- Code-Bausteine

  Diese Bausteine enthalten die STEP 7-Befehle.

- Datenbausteine

  Diese Bausteine enthalten Konstanten und Variablen für das STEP 7-Programm.

Anwenderspeicher

Alle Programme und Daten wie Teileprogramme, Unterprogramme, Kommentare, Werkzeugkorrekturen, Nullpunktverschiebungen/Frames sowie Kanal- und Programmanwenderdaten können in den gemeinsamen CNC-Anwenderspeicher abgelegt werden.

Arbeitsfeldbegrenzung

Mit der Arbeitsfeldbegrenzung kann der Verfahrbereich der Achsen zusätzlich zu den Endschaltern eingeschränkt werden. Je Achse ist ein Wertepaar zur Beschreibung des geschützten Arbeitsraumes möglich.

Arbeitsraum

Dreidimensionaler Raum, in den die Werkzeugspitze aufgrund der Konstruktion der Werkzeugmaschine hineinfahren kann. Siehe → Schutzraum.

Arbeitsspeicher

Der Arbeitsspeicher ist ein RAM-Speicher in der → CPU, auf den der Prozessor während der Programmbearbeitung auf das Anwenderprogramm zugreift.

Archivieren

Auslesen von Dateien und/oder Verzeichnissen auf ein **externes** Speichergerät.

Asynchrones Unterprogramm

Teileprogramm, das asynchron (unabhängig) zum aktuellen Programmzustand durch ein Interruptsignal (z. B. Signal "schneller NC-Eingang") gestartet werden kann.

Automatik

Betriebsart der Steuerung (Satzfolgebetrieb nach DIN): Betriebsart bei NC-Systemen, in der ein → Teileprogramm angewählt und kontinuierlich abgearbeitet wird.

Bahnachse

Bahnachsen sind alle Bearbeitungsachsen des → Kanals, die vom → Interpolator so geführt werden, dass sie gleichzeitig starten, beschleunigen, stoppen und den Endpunkt erreichen.

Bahngeschwindigkeit

Die maximal programmierbare Bahngeschwindigkeit ist abhängig von der Eingabefeinheit. Bei einer Auflösung von beispielsweise 0,1 mm beträgt die maximal programmierbare Bahngeschwindigkeit 1000 m/min.

Bahnsteuerbetrieb

Ziel des Bahnsteuerbetriebes ist es, ein größeres Abbremsen der → Bahnachsen an den Teileprogramm-Satzgrenzen zu vermeiden und mit möglichst gleicher Bahngeschwindigkeit in den nächsten Satz zu wechseln.

Bahnvorschub

Bahnvorschub wirkt auf → Bahnachsen. Er stellt die geometrische Summe der Vorschübe der beteiligten → Geometrieachsen dar.

Basisachse

Achse, deren Soll- oder Istwert für die Berechnung eines Kompensationswertes herangezogen wird.

Basiskoordinatensystem

Kartesisches Koordinatensystem, wird durch Transformation auf das Maschinenkoordinatensystem abgebildet.

Im → Teileprogramm verwendet der Programmierer Achsnamen des Basiskoordinatensystems. Es besteht, wenn keine → Transformation aktiv ist, parallel zum → Maschinenkoordinatensystem. Der Unterschied zu diesem liegt in den → Achsbezeichnern.

Baudrate

Geschwindigkeit bei der Datenübertragung (Bit/s).

Baustein

Als Bausteine werden alle Dateien bezeichnet, die für die Programmerstellung und Programmverarbeitung benötigt werden.

Bearbeitungskanal

Über eine Kanalstruktur können durch parallele Bewegungsabläufe Nebenzeiten verkürzt werden, z. B. Verfahren eines Ladeportals simultan zur Bearbeitung. Ein CNC-Kanal ist dabei als eigene CNC-Steuerung mit Dekodierung, Satzaufbereitung und Interpolation anzusehen.

Bedienoberfläche

Die Bedienoberfläche (BOF) ist das Anzeigemedium einer CNC-Steuerung in Gestalt eines Bildschirms. Sie ist mit horizontalen und vertikalen Softkeys gestaltet.

Beschleunigung mit Ruckbegrenzung

Zur Erzielung eines optimalen Beschleunigungsverhaltens an der Maschine bei gleichzeitiger Schonung der Mechanik kann im Bearbeitungsprogramm zwischen sprunghafter Beschleunigung und stetiger (ruckfreier) Beschleunigung umgeschaltet werden.

Betriebsart

Ablaufkonzept für den Betrieb einer SINUMERIK-Steuerung. Es sind die Betriebsarten → Jog, → MDA, → Automatik definiert.

Betriebsartengruppe

Technologisch zusammengehörige Achsen und Spindeln können zu einer Betriebsartengruppe (BAG) zusammengefasst werden. Achsen/Spindeln einer BAG können von einem oder mehreren → Kanälen gesteuert werden. Den Kanälen der BAG ist immer die gleiche → Betriebsart zugeordnet.

Bezeichner

Die Wörter nach DIN 66025 werden durch Bezeichner (Namen) für Variable (Rechenvariable, Systemvariable, Anwendervariable), für Unterprogramme, für Schlüsselwörter und Wörter mit mehreren Adressbuchstaben ergänzt. Diese Ergänzungen kommen in der Bedeutung den Wörtern beim Satzaufbau gleich. Bezeichner müssen eindeutig sein. Derselbe Bezeichner darf nicht für verschiedene Objekte verwendet werden.

Booten

Laden des Systemprogramms nach Power On.

C-Achse

Achse, um die eine gesteuerte Drehbewegung und Positionierung mit der Werkstückspindel erfolgt.

CNC

Siehe → NC

COM

Komponente der NC-Steuerung zur Durchführung und Koordination von Kommunikation.

CPU

Central Processing Unit, siehe → Speicherprogrammierbare Steuerung

C-Spline

Der C-Spline ist der bekannteste und am meisten verwendete Spline. Die Übergänge an den Stützpunkten sind tangenten- und krümmungsstetig. Es werden Polynome 3. Grades verwendet.

Datenbaustein

1. Dateneinheit der → PLC, auf die → HIGHSTEP-Programme zugreifen können.
2. Dateneinheit der → NC: Datenbausteine enthalten Datendefinitionen für globale Anwenderdaten. Die Daten können bei der Definition direkt initialisiert werden.

Datenübertragungsprogramm PCIN

PCIN ist ein Hilfsprogramm zum Senden und Empfangen von CNC-Anwenderdaten über die serielle Schnittstelle, wie z. B. Teileprogramme, Werkzeugkorrekturen etc. Das PCIN-Programm ist unter MS-DOS auf Standard-Industrie-PCs lauffähig.

Datenwort

Zwei Byte große Dateneinheit innerhalb eines → Datenbausteins.

Diagnose

1. Bedienbereich der Steuerung
2. Die Steuerung besitzt sowohl ein Selbstdiagnose-Programm als auch Testhilfen für den Service: Status-, Alarm- und Serviceanzeigen

DRF

Differential Resolver Function: NC-Funktion, die in Verbindung mit einem elektronischen Handrad eine inkrementale Nullpunktverschiebung im Automatik-Betrieb erzeugt.

Editor

Der Editor ermöglicht das Erstellen, Ändern, Ergänzen, Zusammenschieben und Einfügen von Programmen/Texten/Programmsätzen.

Eilgang

Schnellste Verfahrgeschwindigkeit einer Achse. Sie wird z. B. verwendet, wenn das Werkzeug aus einer Ruhestellung an die → Werkstückkontur herangefahren oder von der Werkstückkontur zurückgezogen wird. Die Eilganggeschwindigkeit wird maschinenspezifisch über Maschinendatum eingestellt.

Externe Nullpunktverschiebung

Von der → PLC vorgegebene Nullpunktverschiebung.

Fertigteilkontur

Kontur des fertig bearbeiteten Werkstücks. Siehe → Rohteil.

Festpunkt-Anfahren

Werkzeugmaschinen können feste Punkte wie Werkzeugwechselpunkt, Beladepunkt, Palettenwechselpunkt etc. definiert anfahren. Die Koordinaten dieser Punkte sind in der Steuerung hinterlegt. Die Steuerung verfährt die betroffenen Achsen, wenn möglich, im → Eilgang.

Frame

Ein Frame stellt eine Rechenvorschrift dar, die ein kartesisches Koordinatensystem in ein anderes kartesisches Koordinatensystem überführt. Ein Frame enthält die Komponenten → Nullpunktverschiebung, → Rotation, → Skalierung, → Spiegelung.

Führungsachse

Die Führungsachse ist die → Gantry–Achse, die aus Sicht des Bedieners und des Programmierers vorhanden und damit entsprechend wie eine normale NC-Achse beeinflussbar ist.

Genauhalt

Bei programmierter Genauhalt-Anweisung wird die in einem Satz angegebene Position genau und ggf. sehr langsam angefahren. Zur Reduktion der Annäherungszeit werden für Eilgang und Vorschub → Genauhaltsgrenzen definiert.

Genauhaltgrenze

Erreichen alle Bahnachsen ihre Genauhaltgrenze, so verhält sich die Steuerung als habe sie einen Zielpunkt exakt erreicht. Es erfolgt Satzweiterschaltung des → Teileprogramms.

Geometrie

Beschreibung eines → Werkstücks im → Werkstückkoordinatensystem.

Geometrieachse

Geometrieachsen dienen der Beschreibung eines 2- oder 3-dimensionalen Bereichs im Werkstückkoordinatensystem.

Geradeninterpolation

Das Werkzeug wird auf einer Geraden zum Zielpunkt verfahren und dabei das Werkstück bearbeitet.

Geschwindigkeitsführung

Um bei Verfahrbewegungen um sehr kleine Beträge je Satz eine akzeptable Verfahrgeschwindigkeit erreichen zu können, kann vorausschauende Auswertung über mehrere Sätze (→ Look Ahead) eingestellt werden.

Gewindebohren ohne Ausgleichsfutter

Mit dieser Funktion können Gewinde ohne Ausgleichsfutter gebohrt werden. Durch das interpolierende Verfahren der Spindel als Rundachse und der Bohrachse werden Gewinde exakt auf Endbohrtiefe geschnitten, z. B. Sacklochgewinde (Voraussetzung: Achsbetrieb der Spindel).

Gleichlaufachse

Die Gleichlaufachse ist die → Gantry-Achse, deren Sollposition stets von der Verfahrbewegung der → Führungsachse abgeleitet und damit synchron verfahren wird. Aus Sicht des Bedieners und des Programmierers ist die Gleichlaufachse "nicht vorhanden".

Grenzdrehzahl

Maximale/minimale (Spindel-)Drehzahl: Durch Vorgaben von Maschinendaten, der → PLC oder → Settingdaten kann die maximale Drehzahl einer Spindel begrenzt sein.

Hauptprogramm

Mit Nummer oder Bezeichner gekennzeichnetes → Teileprogramm, in dem weitere Hauptprogramme, Unterprogramme oder → Zyklen aufgerufen werden können.

Hauptsatz

Durch ":" eingeleiteter Satz, der alle Angaben enthält, um den Arbeitsablauf in einem → Teileprogramm starten zu können.

HIGHSTEP

Zusammenfassung der Programmiermöglichkeiten für die → PLC des Systems AS300/AS400.

Hilfsfunktionen

Mit Hilfsfunktionen können in → Teileprogrammen → Parameter an die → PLC übergeben werden, die dort vom Maschinenhersteller definierte Reaktionen auslösen.

Hochsprache CNC

Die Hochsprache bietet: → Anwenderdefinierte Variable, → Systemvariable, → Makrotechnik.

Interpolator

Logische Einheit des → NCK, die nach Angaben von Zielpositionen im Teileprogramm Zwischenwerte für die in den einzelnen Achsen zu fahrenden Bewegungen bestimmt.

Interpolatorische Kompensation

Mit Hilfe der interpolatorischen Kompensation können fertigungsbedingte **S**pindel**s**teigungs**f**ehler und **M**ess**s**ystem**f**ehler **k**ompensiert werden (SSFK, MSFK).

Interruptroutine

Interruptroutinen sind spezielle → Unterprogramme, die durch Ereignisse (externe Signale) vom Bearbeitungsprozess gestartet werden können. Ein in Abarbeitung befindlicher Teileprogrammsatz wird abgebrochen, die Unterbrechungsposition der Achsen wird automatisch gespeichert.

JOG

Betriebsart der Steuerung (Einrichtebetrieb): In der Betriebsart JOG kann die Maschine eingerichtet werden. Einzelne Achsen und Spindeln können über die Richtungstasten im Tippbetrieb verfahren werden. Weitere Funktionen in der Betriebsart JOG sind das → Referenzpunktfahren, → Repos sowie → Preset (Istwert setzen).

Kanal

Ein Kanal ist dadurch gekennzeichnet, dass er unabhängig von anderen Kanälen ein → Teileprogramm abarbeiten kann. Ein Kanal steuert exklusiv die ihm zugeordneten Achsen und Spindeln. Teileprogrammabläufe verschiedener Kanäle können durch → Synchronisation koordiniert werden.

Kettenmaß

Auch Inkrementmaß: Angabe eines Bewegungsziels einer Achse durch eine zu verfahrende Wegstrecke und Richtung bezogen auf einen bereits erreichten Punkt. Siehe → Absolutmaß.

Kompensationsachse

Achse, deren Soll- oder Istwert durch den Kompensationswert modifiziert wird.

Kompensationstabelle

Tabelle von Stützpunkten. Sie liefert für ausgewählte Positionen der Basisachse die Kompensationswerte der Kompensationsachse.

Kompensationswert

Differenz zwischen der durch den Messgeber gemessenen Achsposition und der gewünschten, programmierten Achsposition.

Kontur

Umriss des → Werkstücks

Konturüberwachung

Als Maß für die Konturtreue wird der Schleppfehler innerhalb eines definierbaren Toleranzbandes überwacht. Ein unzulässig hoher Schleppfehler kann sich z. B. durch Überlastung des Antriebs ergeben. In diesem Fall kommt es zu einem Alarm und die Achsen werden stillgesetzt.

Koordinatensystem

Siehe → Maschinenkoordinatensystem, → Werkstückkoordinatensystem

Korrekturspeicher

Datenbereich in der Steuerung, in dem Werkzeugkorrekturdaten hinterlegt sind.

Kreisinterpolation

Das → Werkzeug soll zwischen festgelegten Punkten der Kontur mit einem gegebenen Vorschub auf einem Kreis fahren und dabei das Werkstück bearbeiten.

Krümmung

Die Krümmung k einer Kontur ist das Inverse des Radius r des anschmiegenden Kreises in einem Konturpunkt (k = 1/r).

KÜ

Übersetzungsverhältnis

KV

Kreisverstärkungsfaktor, regelungstechnische Größe eines Regelkreises

Ladespeicher

Der Ladespeicher ist bei der CPU 314 der → SPS gleich dem → Arbeitsspeicher.

Linearachse

Die Linearachse ist eine Achse, welche im Gegensatz zur Rundachse eine Gerade beschreibt.

Look Ahead

Mit der Funktion **Look Ahead** wird durch das "Vorausschauen" über eine parametrierbare Anzahl von Verfahrsätzen ein Optimum an Bearbeitungsgeschwindigkeit erzielt.

Losekompensation

Ausgleich einer mechanischen Maschinenlose, z. B. Umkehrlose bei Kugelrollspindeln. Für jede Achse kann die Losekompensation getrennt eingegeben werden.

Makrotechnik

Zusammenfassung einer Menge von Anweisungen unter einem Bezeichner. Der Bezeichner repräsentiert im Programm die Menge der zusammengefassten Anweisungen.

Maschinenachsen

In der Werkzeugmaschine physikalisch existierende Achsen.

Maschinenfestpunkt

Durch die Werkzeugmaschine eindeutig definierter Punkt, z. B. Maschinen-Referenzpunkt.

Maschinenkoordinatensystem

Koordinatensystem, das auf die Achsen der Werkzeugmaschine bezogen ist.

Maschinennullpunkt

Fester Punkt der Werkzeugmaschine, auf den sich alle (abgeleiteten) Messsysteme zurückführen lassen.

Maschinensteuertafel

Bedientafel der Werkzeugmaschine mit den Bedienelementen Tasten, Drehschalter usw. und einfachen Anzeigeelementen wie LEDs. Sie dient der unmittelbaren Beeinflussung der Werkzeugmaschine über die PLC.

Maßangabe metrisch und inch

Im Bearbeitungsprogramm können Positions- und Steigungswerte in inch programmiert werden. Unabhängig von der programmierbaren Maßangabe (G70/G71) wird die Steuerung auf ein Grundsystem eingestellt.

Masse

Als Masse gilt die Gesamtheit aller untereinander verbundenen inaktiven Teile eines Betriebsmittels, die auch im Fehlerfall keine gefährliche Berührungsspannung annehmen können.

MDA

Betriebsart der Steuerung: Manual Data Automatic. In der Betriebsart MDA können einzelne Programmsätze oder Satzfolgen ohne Bezug auf ein Haupt- oder Unterprogramm eingegeben und anschließend über die Taste NC-Start sofort ausgeführt werden.

Meldungen

Alle im Teileprogramm programmierten Meldungen und vom System erkannte → Alarme werden auf der Bedientafel im Klartext mit Datum und Uhrzeit und dem entsprechenden Symbol für das Löschkriterium angezeigt. Die Anzeige erfolgt getrennt nach Alarmen und Meldungen.

Metrisches Messsystem

Genormtes System von Einheiten: für Längen z. B. mm (Millimeter), m (Meter).

NC

Numerical Control: NC-Steuerung umfasst alle Komponenten der Werkzeugmaschinensteuerung: → NCK, → PLC, HMI, → COM.

Hinweis

Für die Steuerungen SINUMERIK 840D wäre CNC-Steuerung korrekter: Computerized Numerical Control.

NCK

Numerical Control Kernel: Komponente der NC-Steuerung, die → Teileprogramme abarbeitet und im Wesentlichen die Bewegungsvorgänge für die Werkzeugmaschine koordiniert.

Nebensatz

Durch "N" eingeleiteter Satz mit Informationen für einen Arbeitsschritt, z. B. eine Positionsangabe.

Netz

Ein Netz ist die Verbindung von mehreren S7-300 und weiteren Endgeräten, z. B. einem PG, über → Verbindungskabel. Über das Netz erfolgt ein Datenaustausch zwischen den angeschlossenen Geräten.

NRK

Numeric Robotic Kernel (Betriebssystem des → NCK)

Nullpunktverschiebung

Vorgabe eines neuen Bezugspunktes für ein Koordinatensystem durch Bezug auf einen bestehenden Nullpunkt und ein → Frame.

1. Einstellbar

   SINUMERIK 840D: Es steht eine projektierbare Anzahl von einstellbaren Nullpunktverschiebungen für jede CNC-Achse zur Verfügung. Die über G-Funktionen anwählbaren Verschiebungen sind alternativ wirksam.

2. Extern

   Zusätzlich zu allen Verschiebungen, die die Lage des Werkstücknullpunktes festlegen, kann eine externe Nullpunktverschiebung durch Handrad (DRF-Verschiebung) oder von der PLC überlagert werden.

3. Programmierbar

   Mit der Anweisung `TRANS` sind für alle Bahn- und Positionierachsen Nullpunktverschiebungen programmierbar.

NURBS

Die steuerungsinterne Bewegungsführung und Bahninterpolation wird auf Basis von NURBS (**N**on **U**niform **R**ational **B**-**S**plines) durchgeführt. Damit steht bei SINUMERIK 840D steuerungsintern für alle Interpolationen ein einheitliches Verfahren zur Verfügung.

OEM

Für Maschinenhersteller, die ihre eigene Bedienoberfläche erstellen oder technologiespezifische Funktionen in die Steuerung einbringen wollen, sind Freiräume für individuelle Lösungen (OEM-Applikationen) für SINUMERIK 840D vorgesehen.

Orientierter Spindelhalt

Halt der Werkstückspindel in vorgegebener Winkellage, z. B. um an bestimmter Stelle eine Zusatzbearbeitung vorzunehmen.

Orientierter Werkzeugrückzug

`RETTOOL`: Bei Bearbeitungsunterbrechungen (z. B. bei Werkzeugbruch) kann das Werkzeug per Programmbefehl mit vorgebbarer Orientierung um einen definierten Weg zurückgezogen werden.

Override

Manuelle bzw. programmierbare Eingriffsmöglichkeit, die es dem Bediener gestattet, programmierte Vorschübe oder Drehzahlen zu überlagern, um sie einem bestimmten Werkstück oder Werkstoff anzupassen.

Peripheriebaugruppe

Peripheriebaugruppen stellen die Verbindung zwischen CPU und Prozess her.

Peripheriebaugruppen sind:

- → Digital-Ein-/Ausgabebaugruppen
- → Analog-Ein-/Ausgabebaugruppen
- → Simulatorbaugruppen

PLC

Programmable **L**ogic **C**ontrol: → Speicherprogrammierbare Steuerung. Komponente der → NC: Anpass-Steuerung zur Bearbeitung der Kontroll-Logik der Werkzeugmaschine.

PLC-Programmierung

Die PLC wird mit der Software **STEP 7** programmiert. Die Programmiersoftware STEP 7 basiert auf dem Standardbetriebssystem **WINDOWS** und enthält die Funktionen der STEP 5 -Programmierung mit innovativen Weiterentwicklungen.

PLC-Programmspeicher

SINUMERIK 840D: Im PLC-Anwenderspeicher werden das PLC-Anwenderprogramm und die Anwenderdaten gemeinsam mit dem PLC-Grundprogramm abgelegt.

Polarkoordinaten

Koordinatensystem, das die Lage eines Punktes in einer Ebene durch seinen Abstand vom Nullpunkt und den Winkel festlegt, den der Radiusvektor mit einer festgelegten Achse bildet.

Polynom-Interpolation

Mit der Polynom-Interpolation können die unterschiedlichsten Kurvenverläufe erzeugt werden, wie **Gerade-, Parabel-, Potenzfunktionen** (SINUMERIK 840D).

Positionierachse

Achse, die eine Hilfsbewegung an einer Werkzeugmaschine ausführt. (z. B. Werkzeugmagazin, Palettentransport). Positionierachsen sind Achsen, die nicht mit den → Bahnachsen interpolieren.

Programmbaustein

Programmbausteine enthalten die Haupt- und Unterprogramme der → Teileprogramme.

Programmierbare Arbeitsfeldbegrenzung

Begrenzung des Bewegungsraumes des Werkzeugs auf einen durch programmierte Begrenzungen definierten Raum.

Programmierbare Frames

Mit programmierbaren → Frames können dynamisch im Zuge der Teileprogramm-Abarbeitung neue Koordinatensystem-Ausgangspunkte definiert werden. Es wird unterschieden nach absoluter Festlegung anhand eines neuen Frames und additiver Festlegung unter Bezug auf einen bestehenden Ausgangspunkt.

Programmierschlüssel

Zeichen und Zeichenfolgen, die in der Programmiersprache für → Teileprogramme eine festgelegte Bedeutung haben.

Pufferbatterie

Die Pufferbatterie gewährleistet, dass das → Anwenderprogramm in der → CPU netzausfallsicher hinterlegt ist und festgelegte Datenbereiche und Merker, Zeiten und Zähler remanent gehalten werden.

Quadrantenfehlerkompensation

Konturfehler an Quadrantenübergängen, die durch wechselnde Reibverhältnisse an Führungsbahnen entstehen, sind mit der Quadrantenfehlerkompensation weitgehend eliminierbar. Die Parametrierung der Quadrantenfehlerkompensation erfolgt durch einen Kreisformtest.

Referenzpunkt

Punkt der Werkzeugmaschine, auf den sich das Messsystem der → Maschinenachsen bezieht.

Rohteil

Teil, mit dem die Bearbeitung eines Werkstücks begonnen wird.

Rotation

Komponente eines → Frames, die eine Drehung des Koordinatensystems um einen bestimmten Winkel definiert.

R-Parameter

Rechenparameter, kann vom Programmierer des → Teileprogramms für beliebige Zwecke im Programm gesetzt oder abgefragt werden.

Rundachse

Rundachsen bewirken eine Werkstück- oder Werkzeugdrehung in eine vorgegebene Winkellage.

Rundungsachse

Rundungsachsen bewirken eine Werkstück- oder Werkzeugdrehung in eine einem Teilungsraster entsprechende Winkellage. Beim Erreichen eines Rasters ist die Rundungsachse "in Position".

Satzsuchlauf

Zum Austesten von Teileprogrammen oder nach einem Abbruch der Bearbeitung kann über die Funktion "Satzsuchlauf" eine beliebige Stelle im Teileprogramm angewählt werden, an der die Bearbeitung gestartet oder fortgesetzt werden soll.

Schlüsselschalter

Der Schlüsselschalter auf der → Maschinensteuertafel besitzt 4 Stellungen, die vom Betriebssystem der Steuerung mit Funktionen belegt sind. Zum Schlüsselschalter gehören drei verschiedenfarbige Schlüssel, die in den angegebenen Stellungen abgezogen werden können.

Schlüsselwörter

Wörter mit festgelegter Schreibweise, die in der Programmiersprache für → Teileprogramme eine definierte Bedeutung haben.

Schneidenradiuskorrektur

Bei der Programmierung einer Kontur wird von einem spitzen Werkzeug ausgegangen. Da dies in der Praxis nicht realisierbar ist, wird der Krümmungsradius des eingesetzten Werkzeugs der Steuerung angegeben und von dieser berücksichtigt. Dabei wird der Krümmungsmittelpunkt um den Krümmungsradius verschoben äquidistant um die Kontur geführt.

Schnellabheben von der Kontur

Beim Eintreffen eines Interrupts kann über das CNC-Bearbeitungsprogramm eine Bewegung eingeleitet werden, die ein schnelles Abheben des Werkzeugs von der gerade bearbeiteten Werkstückkontur ermöglicht. Zusätzlich kann der Rückzugwinkel und der Betrag des Weges parametriert werden. Nach dem Schnellabheben kann zusätzlich eine Interruptroutine ausgeführt werden (SINUMERIK 840D).

Schnelle digitale Ein-/Ausgänge

Über die digitalen Eingänge können z. B. schnelle CNC-Programmroutinen (Interruptroutinen) gestartet werden. Über die digitalen CNC-Ausgänge können schnelle, programmgesteuerte Schaltfunktionen ausgelöst werden (SINUMERIK 840D).

Schrägenbearbeitung

Bohr- und Fräsbearbeitungen an Werkstückflächen, die nicht in den Koordinatenebenen der Maschine liegen, können mit Unterstützung der Funktion "Schrägenbearbeitung" komfortabel ausgeführt werden.

Schraubenlinien-Interpolation

Die Schraubenlinien-Interpolation eignet sich besonders zum einfachen Herstellen von Innen- oder Außengewinden mit Formfräsern und zum Fräsen von Schmiernuten.

Dabei setzt sich die Schraubenlinie aus zwei Bewegungen zusammen:

- Kreisbewegung in einer Ebene
- Linearbewegung senkrecht zu dieser Ebene

Schrittmaß

Verfahrweglängenangabe über Inkrementanzahl (Schrittmaß). Inkrementanzahl kann als → Settingdatum hinterlegt sein bzw. durch entsprechend beschriftete Tasten 10, 100, 1000, 10000 gewählt werden.

Schutzraum

Dreidimensionaler Raum innerhalb des → Arbeitsraumes, in den die Werkzeugspitze nicht hineinreichen darf.

Serielle Schnittstelle V.24

Für die Dateneingabe/-ausgabe ist auf der PCU 20 eine serielle V.24-Schnittstelle (RS232), auf der PCU 50/70 sind zwei V.24-Schnittstellen vorhanden. Über diese Schnittstellen können Bearbeitungsprogramme sowie Hersteller- und Anwenderdaten geladen und gesichert werden.

Settingdaten

Daten, die Eigenschaften der Werkzeugmaschine auf durch die Systemsoftware definierte Weise der NC-Steuerung mitteilen.

Sicherheitsfunktionen

Die Steuerung enthält ständig aktive Überwachungen, die Störungen in der → CNC, der Anpass-Steuerung (→ PLC) und der Maschine so frühzeitig erkennen, dass Schäden an Werkstück, Werkzeug oder Maschine weitgehend ausgeschlossen werden. Im Störungsfall wird der Bearbeitungsablauf unterbrochen und die Antriebe werden stillgesetzt, die Störungsursache gespeichert und als Alarm angezeigt. Gleichzeitig wird der PLC mitgeteilt, dass ein CNC-Alarm ansteht.

Skalierung

Komponente eines → Frames, die achsspezifische Maßstabsveränderungen bewirkt.

Softkey

Taste, deren Beschriftung durch ein Feld im Bildschirm repräsentiert wird, das sich dynamisch der aktuellen Bediensituation anpasst. Die frei belegbaren Funktionstasten (Softkeys) werden softwaremäßig definierten Funktionen zugeordnet.

Software-Endschalter

Software-Endschalter begrenzen den Verfahrbereich einer Achse und verhindern ein Auffahren des Schlittens auf die Hardware-Endschalter. Je Achse sind 2 Wertepaare vorgebbar, die getrennt über die → PLC aktiviert werden können.

Speicherprogrammierbare Steuerung

Speicherprogrammierbare Steuerungen (SPS) sind elektronische Steuerungen, deren Funktion als Programm im Steuerungsgerät gespeichert ist. Aufbau und Verdrahtung des Gerätes hängen also nicht von der Funktion der Steuerung ab. Die speicherprogrammierbare Steuerung hat die Struktur eines Rechners; sie besteht aus CPU (Zentralbaugruppe) mit Speicher, Ein-/Ausgabebaugruppen und internem Bus-System. Die Peripherie und die Programmiersprache sind auf die Belange der Steuerungstechnik ausgerichtet.

Spiegelung

Bei Spiegelung werden die Vorzeichen der Koordinatenwerte einer Kontur bezüglich einer Achse vertauscht. Es kann bezüglich mehrerer Achsen zugleich gespiegelt werden.

Spindelsteigungsfehler-Kompensation

Ausgleich mechanischer Ungenauigkeiten einer am Vorschub beteiligten Kugelrollspindel durch die Steuerung anhand von hinterlegten Messwerten der Abweichungen.

Spline-Interpolation

Mit der Spline-Interpolation kann die Steuerung aus nur wenigen vorgegebenen Stützpunkten einer Sollkontur einen glatten Kurvenverlauf erzeugen.

Standardzyklen

Für häufig wiederkehrende Bearbeitungsaufgaben stehen Standardzyklen zur Verfügung:

- für die Technologie Bohren/Fräsen
- für die Technologie Drehen

Im Bedienbereich "Programm" werden unter dem Menü "Zyklenunterstützung" die zur Verfügung stehenden Zyklen aufgelistet. Nach Anwahl des gewünschten Bearbeitungszyklus werden die notwendigen Parameter für die Wertzuweisung im Klartext angezeigt.

Synchronachsen

Synchronachsen benötigen für ihren Weg die gleiche Zeit wie die Geometrieachsen für ihren Bahnweg.

Synchronaktionen

1. Hilfsfunktionsausgabe

   Während der Werkstückbearbeitung können aus dem CNC-Programm heraus technologische Funktionen (→ Hilfsfunktionen) an die PLC ausgegeben werden. Über diese Hilfsfunktionen werden beispielsweise Zusatzeinrichtungen der Werkzeugmaschine gesteuert, wie Pinole, Greifer, Spannfutter etc.

2. Schnelle Hilfsfunktionsausgabe

   Für zeitkritische Schaltfunktionen können die Quittierungszeiten für die → Hilfsfunktionen minimiert und unnötige Haltepunkte im Bearbeitungsprozess vermieden werden.

Synchronisation

Anweisungen in → Teileprogrammen zur Koordination der Abläufe in verschiedenen → Kanälen an bestimmten Bearbeitungsstellen.

Systemspeicher

Der Systemspeicher ist ein Speicher in der CPU, in der folgende Daten abgelegt werden:

- Daten, die das Betriebssystem benötigt
- die Operanden Zeiten, Zähler, Merker

Systemvariable

Ohne Zutun des Programmierers eines → Teileprogramms existierende Variable. Sie ist definiert durch einen Datentyp und dem Variablennamen, der durch das Zeichen **$** eingeleitet wird. Siehe → Anwenderdefinierte Variable.

Teileprogramm

Folge von Anweisungen an die NC-Steuerung, die insgesamt die Erzeugung eines bestimmten → Werkstücks bewirken. Ebenso Vornahme einer bestimmten Bearbeitung an einem gegebenen → Rohteil.

Teileprogrammsatz

Teil eines → Teileprogramms, durch Line Feed abgegrenzt. Es werden → Hauptsätze und → Nebensätze unterschieden.

Teileprogrammverwaltung

Die Teileprogrammverwaltung kann nach → Werkstücken organisiert werden. Die Größe des Anwenderspeichers bestimmt die Anzahl der zu verwaltenden Programme und Daten. Jede Datei (Programme und Daten) kann mit einem Namen von maximal 24 alphanumerischen Zeichen versehen werden.

Text-Editor

Siehe → Editor

TOA–Bereich

Der TOA–Bereich umfasst alle Werkzeug- und Magazindaten. Standardmäßig fällt der Bereich bzgl. der Reichweite der Daten mit dem Bereich → Kanal zusammen. Über Maschinendaten kann jedoch festgelegt werden, dass sich mehrere Kanäle eine → TOA-Einheit teilen, so dass diesen Kanälen dann gemeinsame WZV-Daten zur Verfügung stehen.

TOA–Einheit

Jeder → TOA-Bereich kann mehrere TOA-Einheiten enthalten. Die Anzahl der möglichen TOA-Einheiten wird über die maximale Anzahl aktiver → Kanäle begrenzt. Eine TOA-Einheit umfasst genau einen WZ-Daten-Baustein und einen Magazindaten-Baustein. Zusätzlich kann noch ein WZ-Trägerdaten-Baustein enthalten sein (optional).

Transformation

Additive oder absolute Nullpunktsverschiebung einer Achse.

Unterprogramm

Folge von Anweisungen eines → Teileprogramms, die mit unterschiedlichen Versorgungsparametern wiederholt aufgerufen werden kann. Der Aufruf des Unterprogramms erfolgt aus einem Hauptprogramm. Jedes Unterprogramm kann gegen nicht autorisiertes Auslesen und Anzeigen gesperrt werden. → Zyklen sind eine Form von Unterprogrammen.

Urlöschen

Beim Urlöschen werden folgende Speicher der → CPU gelöscht:

- → Arbeitsspeicher
- Schreib-/Lesebereich des → Ladespeichers
- → Systemspeicher
- → Backup-Speicher

Variablendefinition

Eine Variablendefinition umfasst die Festlegung eines Datentyps und eines Variablennamens. Mit dem Variablennamen kann der Wert der Variablen angesprochen werden.

Verbindungskabel

Verbindungskabel sind vorgefertigte bzw. vom Anwender selbst anzufertigende 2-Draht-Leitungen mit 2 Anschlussstecker. Diese Verbindungskabel verbinden die → CPU über die → Mehrpunkt-Schnittstelle (MPI) mit einem → PG bzw. mit anderen CPUs.

Verfahrbereich

Der maximal zulässige Verfahrbereich bei Linearachsen beträgt ± 9 Dekaden. Der absolute Wert ist abhängig von der gewählten Eingabe- und Lageregelfeinheit und dem Einheitensystem (inch oder metrisch).

Vorkoinzidenz

Satzwechsel bereits, wenn Bahnweg um ein vorgegebenes Delta der Endposition nahe gekommen ist.

Vorschub-Override

Der programmierten Geschwindigkeit wird die aktuelle Geschwindigkeitseinstellung über → Maschinensteuertafel oder von der → PLC überlagert (0-200%). Die Vorschubgeschwindigkeit kann zusätzlich im Bearbeitungsprogramm durch einen programmierbaren Prozentfaktor (1-200%) korrigiert werden.

Vorsteuerung, dynamisch

Ungenauigkeiten der → Kontur, bedingt durch Schleppfehler, lassen sich durch die dynamische, beschleunigungsabhängige Vorsteuerung nahezu eliminieren. Dadurch ergibt sich auch bei hohen → Bahngeschwindigkeiten eine hervorragende Bearbeitungsgenauigkeit. Die Vorsteuerung kann achsspezifisch über das → Teileprogramm an- und abgewählt werden.

Werkstück

Von der Werkzeugmaschine zu erstellendes/zu bearbeitendes Teil.

Werkstückkontur

Sollkontur des zu erstellenden/bearbeitenden → Werkstücks.

Werkstückkoordinatensystem

Das Werkstückkoordinatensystem hat seinen Ausgangspunkt im → Werkstücknullpunkt. Bei Programmierung im Werkstückkoordinatensystem beziehen sich Maße und Richtungen auf dieses System.

Werkstücknullpunkt

Der Werkstücknullpunkt bildet den Ausgangspunkt für das → Werkstückkoordinatensystem. Er ist durch Abstände zum → Maschinennullpunkt definiert.

Werkzeug

An der Werkzeugmaschine wirksames Teil, das die Bearbeitung bewirkt (z. B. Drehmeißel, Fräser, Bohrer, LASER-Strahl ...).

Werkzeugkorrektur

Berücksichtigung der Werkzeug-Abmessungen bei der Berechnung der Bahn.

Werkzeugradiuskorrektur

Um eine gewünschte → Werkstückkontur direkt programmieren zu können, muss die Steuerung unter Berücksichtigung des Radius des eingesetzten Werkzeugs eine äquidistante Bahn zur programmierten Kontur verfahren (G41/G42).

WinSCP

WinSCP ist ein frei verfügbares Open Source-Programm für Windows zum Transferieren von Dateien.

Zeitreziproker Vorschub

Bei SINUMERIK 840D kann anstelle der Vorschubgeschwindigkeit für die Achsbewegung die Zeit programmiert werden, die der Bahnweg eines Satzes benötigen soll (G93).

Zoll-Maßsystem

Maßsystem, das Entfernungen in "inch" und Bruchteilen davon definiert.

Zwischensätze

Verfahrbewegungen mit angewählter → Werkzeugkorrektur (G41/G42) dürfen durch eine begrenzte Anzahl Zwischensätze (Sätze ohne Achsbewegungen in der Korrekturebene) unterbrochen werden, wobei die Werkzeugkorrektur noch korrekt verrechnet werden kann. Die zulässige Anzahl Zwischensätze, die die Steuerung vorausliest, ist über Systemparameter einstellbar.

Zyklen

Geschützte Unterprogramme zur Ausführung von wiederholt auftretenden Bearbeitungsvorgängen am → Werkstück.

Index

<

=

>

0

3

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z