DOKUMENTATION ISG-kernel

Handbuch Programmieranleitung

Kurzbezeichnung:
PROG

ISG Industrielle Steuerungstechnik GmbH
STEP, Gropiusplatz 10
D-70563 Stuttgart

www.isg-stuttgart.de
support@isg-stuttgart.de

Dokumentation Version: 1.31
13.11.2024

Vorwort

Rechtliche Hinweise

Diese Dokumentation wurde sorgfältig erstellt. Die beschriebenen Produkte und der Funktionsumfang werden jedoch ständig weiter entwickelt. Wir behalten uns das Recht vor, die Dokumentation jederzeit und ohne Ankündigung zu überarbeiten und zu ändern.

Aus den Angaben, Abbildungen und Beschreibungen in dieser Dokumentation können keine Ansprüche auf Änderung bereits gelieferter Produkte geltend gemacht werden.

Qualifikation des Personals

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

Zur Installation und Inbetriebnahme ist die Beachtung der Dokumentation, der nachfolgenden Hinweise und Erklärungen unbedingt notwendig. Das Fachpersonal ist verpflichtet, für jede Installation und Inbetriebnahme die zum betreffenden Zeitpunkt veröffentlichte Dokumentation zu verwenden.

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

Weiterführende Informationen

Unter dem Link

https://www.isg-stuttgart.de/de/isg-kernel/kernel-downloads.html

finden Sie neben der aktuellen Dokumentation weiterführende Informationen zu Meldungen aus dem NC-Kern, Onlinehilfen, SPS-Bibliotheken, Tools usw.

Haftungsausschluss

Änderungen der Software-Konfiguration, die über die dokumentierten Möglichkeiten hinausgehen, sind unzulässig.

Marken und Patente

Der Name ISG®, ISG kernel®, ISG virtuos®, ISG dirigent® und entsprechende Logos sind eingetragene und lizenzierte Marken der ISG Industrielle Steuerungstechnik GmbH.

Die Verwendung anderer in dieser Dokumentation enthaltene Marken oder Kennzeichen durch Dritte kann zu einer Verletzung von Rechten der Inhaber der entsprechenden Bezeichnungen führen.

Copyright

Allgemeine- und Sicherheitshinweise

Verwendete Symbole und ihre Bedeutung

In der vorliegenden Dokumentation werden die folgenden Symbole mit nebenstehendem Sicherheitshinweis und Text verwendet. Die (Sicherheits-) Hinweise sind aufmerksam zu lesen und unbedingt zu befolgen!

Symbole im Erklärtext

➢ Gibt eine Aktion an.

⇨ Gibt eine Handlungsanweisung an.

⚠ GEFAHR

Akute Verletzungsgefahr!

Wenn der Sicherheitshinweis neben diesem Symbol nicht beachtet wird, besteht unmittelbare Gefahr für Leben und Gesundheit von Personen!

⚠ VORSICHT

Schädigung von Personen und Maschinen!

Wenn der Sicherheitshinweis neben diesem Symbol nicht beachtet wird, können Personen und Maschinen geschädigt werden!

Achtung

Einschränkung oder Fehler

Dieses Symbol beschreibt Einschränkungen oder warnt vor Fehlern.

Hinweis

Tipps und weitere Hinweise

Dieses Symbol kennzeichnet Informationen, die zum grundsätzlichen Verständnis beitragen oder zusätzliche Hinweise geben.

Beispiel

Allgemeines Beispiel

Beispiel zu einem erklärten Sachverhalt.

Programmierbeispiel

NC-Programmierbeispiel

Programmierbeispiel (komplettes NC-Programm oder Programmsequenz) der beschriebenen Funktionalität bzw. des entsprechenden NC-Befehls.

Versionshinweis

Spezifischer Versionshinweis

Optionale, ggf. auch eingeschränkte Funktionalität. Die Verfügbarkeit dieser Funktionalität ist von der Konfiguration und dem Versionsumfang abhängig.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Kurzbeschreibung

Die Steuerung verarbeitet Syntax nach DIN 66025 bzw. üblicher Auslegung sowie Sprachelemente gemäß Erweiterungen:

- textorientierte Programmnamen
- umfassende Parameterrechnung (lokale und globale Parameter)
- Zugriff auf steuerungsinterne Daten wie z.B. Positionen, Messwerte, Werkzeugdaten, Nullpunktwerte, Verschiebungen etc. über Klartextbezeichnung (V.A.name, V.G.name)
- Definition von Klartexten zur Bezeichnung freier Parameter im NC-Programm (V.L.*name*, V.S.*name*, V.P.*name*)
- Steuersatzkonstrukte in Anlehnung an die Programmiersprache "C", z.B.:
  - Bedingte Sprünge : $IF, $ELSEIF, $ELSE, $ENDIF, $SWITCH, $CASE, $DEFAULT,$ENDSWITCH, $BREAK
  - Zählerschleifen : $FOR, $ENDFOR, $CONTINUE, $BREAK
  - Schleifen mit Laufbedingung: $WHILE, $ENDWHILE, $CONTINUE, $BREAK
  - Schleifen ohne Laufbedingung: $DO, $ENDDO, $CONTINUE, $BREAK
  - Sprünge innerhalb einer NC-Programmebene: $GOTO
- Unterscheidung zwischen globalen (von allen Hauptprogrammen erreichbar) und lokalen Unterprogrammen (nur vom zugehörigen Hauptprogramm erreichbar)
- Mathematische Ausdrücke, z.B.:
  - arithmetische Standardgrundoperationen : + , - , * , / , ** , MOD
  - Zahlenfunktionen wie ABS,SQR,SQRT,EXP,LN,DEXP,LOG
  - Winkelfunktionen wie SIN,COS,TAN,ASIN,ACOS,ATAN
  - Umwandlungsfunktionen wie INT,FRACT,ROUND
- Technologiebefehle mit konfigurierbarer Wirkung jeder Funktion:
  - Zusatzfunktionen (M0....M65535)
  - Hilfsfunktionen (H0....H65535)
  - Spindelfunktionen (S, M3, M4, M5, M19)
  - Werkzeugfunktionen (T, D),
- Verarbeitung von Koordinatenschreibweise A, B, C, ..., U, V, W, soweit nicht anderweitig verwendet und/oder Programmierung von Strings (z.B. X_ACHSE, SPINDEL_1 ...)

Hinweis

Die komplette NC-Befehlsübersicht ist im Anhang aufgelistet [▶ 877]!

Obligatorischer Hinweis zu Verweisen auf andere Dokumente

Zwecks Übersichtlichkeit wird eine verkürzte Darstellung der Verweise (Links) auf andere Dokumente bzw. Parameter gewählt, z.B. [PROG] für Programmieranleitung oder P-AXIS-00001 für einen Achsparameter.

Technisch bedingt funktionieren diese Verweise nur in der Online-Hilfe (HTML5, CHM), allerdings nicht in PDF-Dateien, da PDF keine dokumentenübergreifenden Verlinkungen unterstützt.

2 Grundlagen der Programmierung

2.1 Darstellung der Syntax

Für die Darstellung der Syntax der in dieser Dokumentation beschriebenen NC-Befehle gilt folgende Konvention:

Fett	Obligatorische Syntaxelemente des NC-Befehls.
[ ]	Optional 1-maliges Auftreten
{ }	Optional mehrmaliges Auftreten
|	Alternative zwischen Schlüsselworten („oder“)
=..	Mathematischer Ausdruck [▷ 31] oder String. Das Gleichheitszeichen ist optional.

2.2 Zeichen- und Zahlenformate

2.2.1 Zeichenvorrat und Fileformat

Der eingesetzte Decoder verarbeitet NC-Steuerdaten des ASCII-Zeichensatzes. NC-Steuerdaten sind mit einem im Bedienungssystem integrierten Programmiersystem bzw. einem Editor erstellbar.

Die CNC verarbeitet folgende Zeichen:

Zeichen =	Buchstabe	und / oder
	Ziffer	und / oder
	Sonderzeichen	und / oder
	Steuerzeichen	

Buchstabe	{ A B C ... Z a b c ... z }
Ziffern	{ 1 2 3 ... 9 0 }
Sonderzeichen	{ ! @ # $ % & * - + _ = ~ ( ) [ ] , ; : " < > / \ ? ' }
Steuerzeichen = HT	Horizontaltabulator TAB
SP Zwischenraum	SPACE
CR Wagenrücklauf	CARRIAGE RETURN
LF Zeilenvorschub	LINE FEED

Einschränkungen bei NC-Dateinamen

Für Dateinamen sind folgende Zeichen nicht zulässig:

Sonderzeichen	("] # $; ,
Steuerzeichen	TAB, CARRIAGE RETURN, LINE FEED

2.2.2 Zahleneingabe

Der Eingabebereich von Zahlen ist nur durch die interne Zahlendarstellung begrenzt.

Diese Zahlendarstellung erlaubt Verfahrbewegungen im Bereich von 200 m bei einer Auflösung von 0,1 µm!

Bei der Eingabe von Positionsangaben, Vorschüben, etc. können durch Änderung der internen Umrechnungsfaktoren in einer Spezifikationsdatei auch andere Einheiten als Millimeter oder Inch verwendet werden.

Beispiel

Zahleneingaben als Ganz- oder Dezimalzahlen:

- Dezimalstellen werden grundsätzlich mit einem "." getrennt, wobei führende Nullen weggelassen werden können.
- Je nach Konfigurierung und Programmierung erfolgen die Eingaben für Längen und Positionswerte in Millimetern oder Inch, die Eingabe von Winkeln in Grad [°] oder Neugrad [Gon].

Eingabeformat:

Werte: [µm]	Eingaben: [mm]
0,1	0.0001 oder .0001
1	0.001 oder .001
10	0.01 oder .01
100	0.1 oder .1
1000	1.0 oder 1
10000	10.0 oder 10
100000	100.0 oder 100
1000000	1000.0 oder 1000

Beispiel

Zahleneingaben als Hexadezimalzahlen:

- Diese Zahlen werden durch '...' eingeschlossen und mit der Zeichenkombination 16#, 0x oder alternativ mit H eingeleitet. Anschließende führende Nullen und Leerzeichen werden überlesen.
- Für die sechs Zusatzziffern können die Buchstaben A bis F bzw. a bis f verwendet werden.

Format:'16#<A...F, a..f, 0..9>**'** oder **'0x**<A...F, a..f, 0..9>**'**oder **'H**<A...F, a..f, 0..9>**'**

Eingabeformat:

'16#FA1B' entspricht dem Dezimalwert 64027
'0x0ED2' entspricht dem Dezimalwert 3794
'H1869f' entspricht dem Dezimalwert 99999

Beispiel

Zahleneingaben als Binärzahlen (Dualzahlen):

- Diese Zahlen werden durch '...' eingeschlossen und mit der Zeichenkombination 2#, 02# oder alternativ mit B eingeleitet. Anschließende führende Nullen und Leerzeichen werden überlesen.
- Für die Darstellung werden die zwei Ziffern 0 (Null) und 1 (Eins) verwendet.

Format: '2#<0..1>**'** oder **'02#**<0..1>**'** oder **'B**<0..1>**'**

Eingabeformat:

'2#1010011'	entspricht dem Dezimalwert 83
'02#010011'	entspricht dem Dezimalwert 19
'B11101010'	entspricht dem Dezimalwert 234

Beispiel

Zahleneingaben als Oktalzahlen:

- Diese Zahlen werden durch '...' eingeschlossen und mit der Zeichenkombination 8# oder 08# eingeleitet. Anschließende führende Nullen und Leerzeichen werden überlesen.
- Für die Darstellung werden die Ziffern 0 bis 7 verwendet.

Format: '8#<0..7>**'** oder **'08#**<0..7>**'**

Eingabeformat:

'8#12345'	entspricht dem Dezimalwert 5349
'08#0107302'	entspricht dem Dezimalwert 36546

2.3 Aufbau von NC-Steuerdaten: NC-Programme

Definitionen zum Verständnis:

NC-Programme sind neben Werkzeugdaten, Nullpunktverschiebungsdaten usw. Bestandteil der NC-Steuerdaten. NC-Programme beschreiben den Ablauf von Bearbeitungsvorgängen.

Trennzeichen lassen erkennen, dass eine Folge von Zeichen (Zeichenkette) abgeschlossen ist. Vom Decoder werden als Trennzeichen gewertet:

CR	LF	ETX	TAB	\0	Blank	(	;	"	]	,	#	$

Hinweis

Diese Zeichen dürfen nicht in Dateinamen oder als Teil von %-Programmnamen von NC-Haupt- oder Unterprogrammen (lokal, global) verwendet werden, andernfalls wird ein Fehler ausgegeben.

Kommentare enthalten nichtdecodierbare ASCII-Informationen, die zwischen den Zeichen "(" und ")" bzw. bei ";" dahinter stehen müssen. Stehen innerhalb eines Kommentares weitere "(", so werden entsprechende ")" erwartet. Das Satzendezeichen und das Dateiendezeichen schließen die Kommentare ab.

Mathematische Ausdrücke [▷ 31] sind aufgebaut aus Zahlen, Parametern, Operatoren, Funktionen etc. Sie werden von einer integrierten Rechenfunktion ausgewertet.

Beispiele: "100", "100+20", "100+P10", "100+PP10", "100*[2+P3]", "DEXP[2]", "100+SIN[P10+PP20]".

Ein NC-Programm besteht aus:

- Kommentaren,
- Definitionen lokaler Unterprogramme, bestehend aus:
  - String "%L ", gefolgt von einem Unterprogrammnamen,
  - einer Anzahl von NC-Programmsätzen,
  - einer Unterprogrammendekennung (M17 oder M29).
- Definition des Hauptprogrammes, bestehend aus:
  - String "%", gefolgt von einem Hauptprogrammnamen,
  - einer Anzahl von NC-Sätzen,
  - einer Hauptprogrammendekennung (M02 oder M30).

Hinweis

Wird in der Datei außerhalb von Kommentaren als erstes Zeichen weder ein Trennzeichen noch ein "%" vorgefunden, so wird dies als erstes Zeichen eines namenlosen Hauptprogrammes gewertet. Das bedeutet auch, dass vor "%" keine Satznummern programmiert werden dürfen.

2.4 NC-Satzaufbau

Ein NC-Satz besteht aus einer

- Satznummerierung (optional),
- Anzahl von Worten (NC-Befehlen),
- Satzendekennung.

Die maximale Länge eines NC-Satzes beträgt 4000 Zeichen.

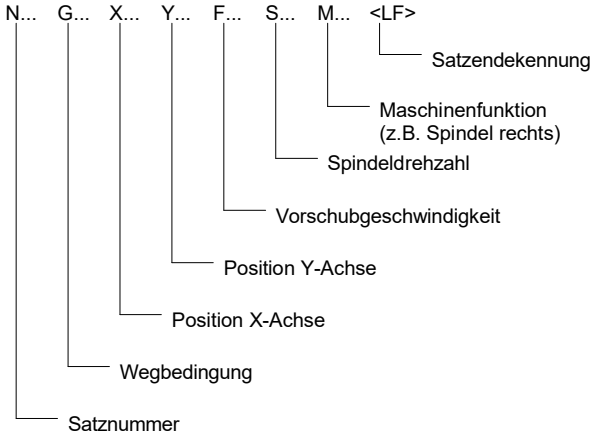
Die Verwendung von #-Befehlen (siehe Sonderfunktionen [▶ 311]) schließt die Programmierung weiterer Worte im NC-Satz aus (mit Ausnahme der Satznummer).

Jeder Satz beginnt in aller Regel mit einer Satznummer, bestehend aus einem N-Zeichen, gefolgt von einem mathematischen Ausdruck. Dieser Ausdruck wird ganzzahlig gerundet in den Anzeigedaten abgebildet.

Die Verwendung von Satzfolgen (SEQUENCE) und Programmsprüngen ($GOTO) erfordert eine eindeutige und aufsteigende Programmierung der Satznummern.

Ansonsten ist die Satznummer für den Programmablauf ohne Bedeutung. Sie muss dann auch nicht in aufsteigender Form programmiert werden.

Beispiel für einen Satzaufbau:



Der DIN 66025 entsprechende NC-Befehle müssen nicht zwingend durch Leerzeichen oder Tabulatoren getrennt werden. Bei der Programmierung von DIN-abweichenden Sprachbefehlen (Steuersatzanweisungen, Sonderfunktionen,...) sind Trennzeichen syntaktisch erforderlich und auch zur Strukturierung eines NC-Programms sinnvoll.

Beispiele für einen NC-Programmaufbau:

ohne Nummerierung % 100 "Satz 1" "Satz 1" "Satz 1" . . . M30	teilweise Nummerierung % 100 N10 "Satz 1" "Satz 2" N20 "Satz 3" "Satz 4" . . M30	vollständige Nummerierung % 100 N10 "Satz 1" N20 "Satz 2" N30 "Satz 3" N40 "Satz 4" . . N700 M30

Worte sind bezüglich ihrer Bedeutung zu unterscheiden in:

- geometrischen Informationen (z.B. Positionen),
- technologischen Informationen (z.B. Spindeldrehzahl, Vorschubgeschwindigkeit, Spindelrechtslauf),
- Informationen zur Steuerung des Programmablaufes (sog. Steuersätze wie z.B. Zählschleifen),
- arithmetischen Informationen (z.B. Berechnung einer Größe, Parameterrechnung).

Mehrere Worte können in einem Satz stehen (Ausnahme: Sonderbefehle aus Kapitel Sonderfunktionen [▶ 311]), wobei die Reihenfolge der Abarbeitung der Steuerinformation innerhalb des Satzes durch die Steuerung festgelegt ist. Damit können vom Programmierer die einzelnen Wortinformationen eines NC-Satzes in beliebiger Reihenfolge eingegeben werden, ohne dass dies Auswirkungen auf die Bearbeitung hat. Bei Ausnahmen wird in dieser Programmieranleitung speziell darauf hingewiesen.

Die Satzendekennung besteht üblicherweise aus der Kombination der Steuerzeichen "CR" und "LF".

2.4.1 Ausblenden von Sätzen '/'

Ausblendsätze dienen zur Festlegung optionaler Bearbeitungsschritte wie z.B. Messschleifen, Testsätze oder Leerschritte innerhalb eines NC-Programmes.

2.4.1.1 Standardausblenden

Durch Voranstellen des "/"-Zeichens können gezielt Sätze ausgeblendet werden. Die Steuerung überliest diese Sätze, wenn die Funktion "Satz ausblenden" am Bedienfeld (HMI) oder über die PLC vor Hauptprogrammstart über ein BOOL-Kommando eingeschaltet wurde.

```
/ N3412 X100 …
```

In Versionen bis V3.01.3020.01 wird jede Änderung der Ausblendeinstellung während aktivem NC-Programm erst beim nächsten Hauptprogrammstart berücksichtigt. In höheren Versionen ist dann das erweiterte Ausblenden [▶ 27] verfügbar.

/ N..

2.4.1.2 Erweitertes Ausblenden (Ausblendebenen)

Versionshinweis

Diese Funktionalität ist ab V3.01.3021.00 verfügbar.

Durch Kombination des Slash-Zeichens "/" mit einer Nummer kann der Anwender bis zu 10 unterschiedliche Ausblendebenen im NC-Programm setzen (z.B.: /5 N3412...). Diese können über das Bedienfeld (HMI) oder über die PLC vor Hauptprogrammstart über ein 32 Bit-Kommando eingeschaltet werden. Es ist möglich, mehrere Ausblendebenen gleichzeitig zu aktivieren.

Beim Erweiterten Ausblenden wirken geänderte Ausblendeinstellungen während aktivem NC-Programm sofort. Definierte Haltepunkte für die sichere Übernahme und Wirksamkeit dieser geänderten Ausblendeinstellungen im NC-Programm können z.B. durch M-Funktionen mit anschließendem #FLUSH WAIT realisiert werden.

Weitere Informationen hierzu siehe [FCT-M6].

Aus Kompatibilitätsgründen haben die Ausblendebenen ohne Nummer "/" und "/1" die gleiche Bedeutung.

/<**1** - **10**> N..

Programmierbeispiel

Erweitertes Ausblenden

```
%skip_levels
N05 G0 X0 Y0 Z0
/1 N10 G74 X1 Y1 Z1
N20 G01 F1000 X10 Y10
/2 N30 Z3
N40 X-1 Y-2
...
/10 N90 #CS ON [0,0,0,0,0,45]
N95 X30 Z50
/ N99 G92 X0
N999 M30
```

2.4.2 Satzspezifische Kommentare

Erläuternde Kommentare können an fast jeder Stelle eines NC-Programmes, z.B. auch am Programmanfang, eingefügt werden. Kommentare haben keine Auswirkungen auf die Abarbeitung eines NC-Programmes.

Kommentare werden durch eine "(" eingeleitet. Wenn sich der Kommentar bis zum Satzende erstreckt, so reicht zur Abgrenzung die "(". Bei Kommentaren innerhalb eines Satzes ist auch eine entsprechende ")" erforderlich.

Alternativ dazu kann ein Kommentar auch durch ein Semikolon ";" eingeleitet werden. Diese Art von Kommentar erstreckt sich immer bis zum Satzende.

Kommentare können beliebig lang sein, wobei aber die maximale Anzahl von Zeichen pro Satz nicht überschritten werden darf.

Eine Schachtelung von Kommentaren ist möglich.

Programmierbeispiel

Kommentare im NC-Code

```
% 100 (Kommentar in vollständiger Klammerung)


N200 ... (Kommentar nur mit einleitender Klammer
N300 (Kommentar (Geschachtelter Kommentar))

N500 X10 (Kommentar im Satz) Y20

N700 ... ;Kommentar nach Semikolon

N999 M30
```

Hinweis

Das Auskommentieren beliebig vieler Programmzeilen (siehe Satzübergreifender Kommentar [▶ 345]) ist mit den Sonderbefehlen #COMMENT BEGIN und #COMMENT END möglich.

2.4.3 Zeilenumbruch im NC-Satz '\'

Aus Gründen der Übersichtlichkeit und Lesbarkeit besteht bei bestimmten NC-Befehlen die Möglichkeit, die Syntaxsequenz eines NC-Satzes durch das Einfügen von Zeilenumbrüchen '\' auf zwei oder mehrere Zeilen aufzuteilen. Das Zeilenumbruchzeichen '\' darf hierbei nur nach vollständigen Termen platziert werden.

In den umgebrochenen Sätzen darf ausschließlich die zugehörige Syntaxsequenz fortgeführt werden.

Hinweis

AUSNAHME:

Am Anfang eines umgebrochenen Satzes ist eine neue Satznummer "N... " erlaubt.

Der Zeilenumbruch '\' kann jeweils in der Klammerung verwendet werden bei:

- Achsspezifischer Programmierung X[...]
- Spindelspezifischer Programmierung S[...]
- PLCopen-Befehlen S[MC_...]
- Zyklenprogrammierung L CYCLE [...]
- Deklaration von eigendefinierten Variablenarrays

Sowie allgemein bei:

- DIN/ISO-Programmierung G01 X10 ...

Achtung

NC-Befehle aus der Gruppe der Zusatzfunktionen (#-Befehle) dürfen **nicht** durch einen Zeilenumbruch aufgeteilt werden. Auf Ausnahmen wird gesondert hingewiesen!

Programmierbeispiel

Zeilenumbruch im NC-Satz '\'

```
N10 X[INDP_ASYN G90 POS=100 G01 FEED=2500 \
N20 SLOPE_TYPE=STEP M23 M56 H78]
N…

N10 S[M4 REV5000 M11] S2[M3 REV5000 M34] S2[REV1000 M3 POS=45 M19 \
N20 M11 M12 H56]
N…

N10 S[MC_MoveAbsolute Position=133 Velocity=1000 Acceleration=500 \
N20 Deceleration=600 Jerk=200 Direction=2]
N…

N… L CYCLE [NAME=pocketmill.cyc @P1=100 @P2=80 @P3=5 @P5=15 \
@P6=80 @P7=60 @P8=10 @P9=65 @P10=50 @P11=3 @P12=1 \
@P13=2 @P14=1 @P15=-1 @P16=40 @P17=3]

N10 G60 G90 F1000 G01 \
N20 X100 Y150 Z250
N…

#VAR
V.P.ARRAY_1[3][6] = [10,11,12,13,14,15, \
                     20,21,22,23,24,25, \
                     30,31,32,33,34,35 ]
#ENDVAR
```

2.5 Wortaufbau

Ein Wort besteht aus Adressbuchstaben, arithmetischen Ausdrücken und Texten <string>. Die Bedeutung der einzelnen Adressbuchstaben wird in den folgenden Kapiteln beschrieben. Eine Übersicht gibt es unter: "Zugeordnete Adressbuchstaben [▶ 42]".

2.5.1 Mathematische Ausdrücke

Bestehend aus:

- Zahlenwerte werden unterschieden in:
  - ganze Zahlen <int> und
  - Dezimalzahlen <double>

- Arithmetische Ausdrücke <expr> sind aufgebaut aus:
  - Zahlenwerten
  - Operatoren
  - Funktionen
  - Parametern
  - Variablen
  - Makros

Beispiel: [[sin["MAX_ANZ" * 30.00] + P2] / V.G.BLOCK_NR]

2.5.1.1 Ganze Zahlen <int>

Ganze Zahlen werden ohne Dezimalpunkt angegeben. Der intern zulässige Wertebereich entspricht dem von "long integer"-Variablen der Programmiersprache C, d.h. im Bereich von $-2.14*10^9$ bis $+2.14*10^9$.

Wenn intern mit der Einheit 0.1µm gerechnet wird, liegt bei Eingabe von Werten in mm der Zahlenbereich zwischen $-2.14*10^5$ und $+2.14*10^5$. Dies entspricht einem Verfahrbereich von mehr als 400 Metern. Durch eine interne Begrenzung im Lageregler wird der Verfahrbereich allerdings auf die Hälfte, d.h. auf etwas mehr als 200 Meter beschränkt.

Für negative Zahlen wird ein "-" Zeichen vorangestellt, bei positiven Zahlen ist kein Vorzeichen notwendig.

2.5.1.2 Dezimalzahlen <double>

Dezimalzahlen werden mit Dezimalpunkt oder alternativ auch in der Exponentialschreibweise mit kleinem 'e' (auch wissenschaftliche Schreibweise genannt) angegeben. Der Wertebereich entspricht dem von Integer-Zahlen. Wenn intern mit der Einheit 0.1µm gerechnet wird, werden bei Eingabe von Werten in mm demnach 4 Stellen nach dem Komma berücksichtigt.

Für negative Zahlen wird ein "-" Zeichen vorangestellt, bei positiven Zahlen ist kein Vorzeichen notwendig.

Beispiele für Dezimalzahlen:

123.3456	0.6789	.6789	-345.56	+78.987
12e5	+2.5e6	-4e7	-523.6e-3	

2.5.1.3 Arithmetische Ausdrücke <expr>

Bei der Behandlung von arithmetischen Ausdrücken gelten die üblichen Rechenregeln:

- Punkt-vor-Strich-Rechnung
- die Klammer-Regel, wobei eckige Klammern "[ ]" zu verwenden sind. Runde "(...)" entsprechen Kommentaren.

In arithmetischen Ausdrücken werden oft Parameter verwendet. Die Schreibweise von Parametern ist:

- P gefolgt von einer ganzen Zahl, z.B. P12

Beispiel für einen arithmetischen Ausdruck:

P5 = [[sin[R1*30.00] + P2] / P5]

Makros (Strings) können arithmetischen Ausdrücken und Teilen davon zugewiesen werden.

Ein Makroname führt zu dem Makroinhalt, der analysiert wird. Dabei ist auch eine rekursive Behandlung möglich.

Makronamen müssen in Anführungszeichen eingeschlossen sein. Bei deren Decodierung wird zwischen Groß- und Kleinschreibung unterschieden.

Die Schachtelung von Makros wird durch ein vorangestelltes ´\´-Zeichen vor den begrenzenden Anführungszeichen angezeigt. Es ist darauf zu achten, dass immer komplette Schachtelungsebenen in einem Makro zusammengefasst sind, d.h. das Einfügen von ´[´ am Anfang und ´]´ am Ende des Makroinhaltes darf sich nicht auf das Ergebnis des mathematischen Ausdrucks auswirken.

Programmierbeispiel

Geschachtelte Makros

Richtig:

```
N10 "STRING1" = "COS[\"STRING2\"]"
N20 "STRING2" = "5 * 12"
N30 "STRING3" = "SIN[89.5 + \"STRING1\"]"
N40 X[-2 * "STRING1" + "STRING2" + "STRING3"] (Fahren nach X60)

M30
```

Falsch:

Nur komplette Schachtelungsebenen dürfen im String zusammengefasst werden

```
N10 "STRING1" = "COS["
N20 "STRING2" = "90]"
N30 "STRING3" = "\"STRING1\" \"STRING2\" "
```

Die im NC-Programm definierten Makros sind programmübergreifend gültig!

Die Programmierung von Makros außerhalb mathematischer Ausdrücke wird in "Makroprogrammierung (#INIT MACRO TAB) [▶ 729] " beschrieben.

Übersicht über alle verfügbaren Rechenoperationen:

Grundrechenarten:

Addition	+	`P1 = P2 + P3 + 0.357`
Subtraktion	-	`P1 = P2 - 0.031`
Multiplikation	*	`P1 = P2 * [P3 + 0.5]`
Division	/	`P1 = P2 * P3 / [P5 + P6]`
Exponentenrechnung	**	`P1 = 2**P3 (zwei hoch P3)`
Modulorechnung	MOD	`P1 = 11 MOD 3 (-> 2)`

Zahlenfunktionen:

Absolutwertbildung	ABS [..]	`P1 = ABS [P2 - P4]`
Quadrieren	SQR [..]	`P1 = SQR [P2] + SQR [P3]`
Quadratwurzel	SQRT [..]	`P1 = SQRT [SQR[P2]+SQR[P3]]`
e - Funktion	EXP [..]	`P1 = EXP [P2 * P4]`
Natürlicher Logarithmus	LN [..]	`P1 = LN [P2] + LN [P3]`
Zehner - Exponent	DEXP [..]	`P1 = DEXP [P2]`
Zehner - Logarithmus	LOG [..]	`P1 = LOG [P2]`

Achtung

Bei LN, LOG und SQRT muss das Argument immer größer 0 sein!

Bitoperatoren:

UND-Verknüpfung	&	`P1 = P2 & P3`
ODER-Verknüpfung	|	`P1 = P2 | P3`
Exklusives ODER	^	`P1 = P2 ^ P3`
Komplement	INV[..]	`P1 = INV[P2]`

Achtung

Die Operanden können beliebige positive mathematische Ausdrücke oder Zahlen im Bereich 0 ... 2^32-1 (UNS32) sein. Negative Ausdrücke oder Zahlen sind nicht erlaubt. Gleitkommazahlen werden in ganze Zahlen konvertiert.

Das Ergebnis der Bitoperation liegt immer im Bereich 0 ... 2^32-1 (UNS32).

Logische Operatoren:

UND-Verknüpfung	&& / AND	`P1 = P2 && P3` `P1 = P2 AND P3`
ODER-Verknüpfung	|| / OR	`P1 = P2 || P3` `P1 = P2 OR P3`
Exklusive ODER-Verknüpfung	XOR	`P1 = P2 XOR P3`

NICHT-Operator	NOT[..]	`P1 = NOT[P2]`
		`P1 = NOT[1] (P1 = 0)`
		`P1 = NOT[0.5] (P1 = 0)`
		`P1 = NOT[0.49] (P1 = 1)`
		`P1 = NOT[0] (P1 = 1)`

Achtung

Die Operanden können beliebige positive mathematische Ausdrücke oder Zahlen sein. Negative Ausdrücke oder Zahlen sind nicht erlaubt.

Eine Gleitkommazahl wird mit TRUE (1) bewertet, wenn **ihr Wert > oder = 0.5 ist.**

Vergleichsoperatoren:

Bei Schleifen (Kapitel Anweisungen zur Beeinflussung des NC-Programmablaufes [▶ 234]) sind Vergleichsoperationen erforderlich. Es kann wie folgt geprüft werden auf:

Gleichheit	==	`$IF P1 == 10`
Ungleichheit	!=	`$IF P1 != 10`
größer gleich	>=	`$IF P1 >= 10`
kleiner gleich	<=	`$IF P1 <= 10`
kleiner	<	`$IF P1 < 10`
größer	>	`$IF P1 > 10`

Operator – Prioritäten:

Die verfügbaren Operatoren sind nach absteigender Priorität aufgelistet. 10 ist die höchste und 1 die niedrigste Priorität.

Priorität	Operator	Beschreibung
10	**	Potenz
9	*, /	Multiplikation, Division
8	+, -	Addition, Subtraktion
7	&	Bitweises UND
6	^	Bitweises exklusives ODER
5	|	Bitweises ODER
4	<=, >=, ==, <, >, !=	Vergleichsoperatoren
3	&&, AND	Logisches UND
2	XOR	Logisches exklusives ODER
1	||, OR	Logisches ODER

Mögliche Wahrheitswerte sind:

Wahr	TRUE	`$IF V.A.MERF.X == TRUE`
Nicht wahr	FALSE	`$WHILE V.G.WZ[2].OK == FALSE`

Achtung

Behandlung der Wahrheitswerte:

Für TRUE wird steuerungsintern der Wert 1 verwendet.

Für FALSE wird steuerungsintern der Wert 0 verwendet.

Winkelfunktionen (Angabe von Winkeln in Grad):

Sinus	SIN [..]	`P1 = SIN [P2]`
Kosinus	COS [..]	`P1 = COS [P2]`
Tangens	TAN [..]	`P1 = TAN [P2]`
Kotangens	COT [..]	`P1 = COT [P2]`
Arkus Sinus	ASIN [..]	`P1 = ASIN [P2]`
Arkus Kosinus	ACOS [..]	`P1 = ACOS [P2]`
Arkus Tangens	ATAN [..]	`P1 = ATAN [P2]`
Arkus Tangens mit 2 Argumenten	ATAN2 [*y,x*]	`P1 = ATAN2[100,100]` `(-> Ergebnis ist 45°)`
Arkuskotangens	ACOT [..]	`P1 = ACOT [P2]`

Achtung

Bei den Zahlenfunktionen ASIN und ACOS muss das Argument immer zwischen -1 und +1 sein.

Bei der Zahlenfunktion TAN darf das Argument nicht die Werte ... -90, 90, 270 ... Grad annehmen.

Bei der Zahlenfunktion COT darf das Argument nicht die Werte ... -180, 0, 180 ... Grad annehmen.

Die Zahlenfunktion ATAN2 liefert für x != 0 den Winkel einer Position zur X-Achse im korrekten Quadranten.

Sonderfall: Für ATAN2[0,0] (x = 0 und y = 0) ist das Ergebnis immer 0.

Umwandlungsfunktionen:

Nachkommastellen abschneiden	INT [..]	`P1 = INT [123.567] (P1 = 123)`
Ganzzahligen Anteil entfernen	FRACT [..]	`P1 = FRACT [123.567] (P1 = 0.567)`
Auf ganze Zahl runden	ROUND [..]	`P1 = ROUND [77.5] (P1 = 78)` `P1 = ROUND [45.4] (P1 = 45)`
Aufrunden	CEIL [..]	`P1 = CEIL [8.3] (P1 = 9)`
Abrunden	FLOOR [..]	`P1 = FLOOR [8.7] (P1 = 8)`

Konstanten:

3.141592654 (π)	PI	`P2 = 2*PI (P2 = 6.283185307)`

Spezielle Funktionen:

Test der Existenz von Variablen (V.P., V.L., V.S., V.E.) / Parametern / M/H-Funktionen / Makros	EXIST [<*Variable/ Parameter/ M Funktion/ H Funktion/ Makroname*>]	`$IF EXIST[V.P.MYVAR] == TRUE`
		`$IF EXIST[V.L.MYARRAY[0]] == TRUE *`
		`$IF EXIST[P1] != TRUE`
		`$IF EXIST[M55] == TRUE`
		`$IF EXIST[H20] == TRUE`
		`$IF EXIST["Macro1"] == TRUE`
		`*Bei Arrays mit Angabe gueltiger Indizes!`

Bestimmung der Größe einer Array-Dimension von Variablen (V.P., V.L., V.S., V.E.) / Parametern	SIZEOF [<*Array_name*>, <*Dimension*>]
	
	oder für 1. Dim.
	SIZEOF [<*Array_name*>]
		`#VAR`
		`P99[3][4]= [1,2,3,4,`
		`5,6,7,8,`
		`11,12,13,14]`
		`#ENDVAR`
		`(P12 u. P13 für 1.Dimension)`
		`P12 = SIZEOF[P99] (P12 = 3)`
		`P13 = SIZEOF[P99,1] (P13 ==P12 =3)`
		`(P14 für 2.Dimension)`
		`P14 = SIZEOF[P99,2] (P14 = 4)`
		`(P15 für 3.Dimension, die nicht existiert)`
		`P15 = SIZEOF[P99,3] (P15= -1)`
		`SIZEOF liefert für nicht vorhandene Arraydimensionen und für Variablen, die keine Arrays sind, immer -1.`

Kleineren Wert ermitteln	MIN [*x,y*]	`P1 = MIN [P2, P3]`
Größeren Wert ermitteln	MAX [*x,y*]	`P1 = MAX [P2, P3]`

Vorzeichen bestimmen	SIGN [..]	`P1 = SIGN [P2] liefert für`
		`positive Werte: 1`
		`negative Werte: -1`
		`Null: 0`

Bestimmung der Stringlänge eines Makroinhaltes.
[ab V2.11.2841.00]
	MACRO_LENGTH [<*Makroname*>]	`"Macro53" = "G53 X0 Y0 Z0"`
		`"Empty" = ""`
		
		`P1 = MACRO_LENGTH["Macro53"]`
		`(P1 = 12)`
		
		`P1 = MACRO_LENGTH["Empty"]`
		`(P1 = 0)`
		
		`MACRO_LENGTH liefert für nicht existierende Makros immer -1`

Lesen und Auflösung eines Makroinhaltes. Der Rückgabewert ist ein String.
[ab V3.1.3081.4]
	<MACRO_CONTENT[<*Makroname*>]	`"MACRO_1" = "1 + 2"`
		`"MACRO_2" = "SIN[\"MACRO_1\"]"`
		
		`MACRO_CONTENT["MACRO_1"] gibt "1 + 2" zurück`
		`MACRO_CONTENT["MACRO_2"] gibt "SIN[1 + 2]" zurück`

Verschlüsselungsfunktion:

Mit Hilfe dieser Funktion können Strings verschlüsselt werden. Der zugehörige Schlüssel ist vom Anwender frei definierbar. Die Strings können z.B. wichtige Daten beinhalten, welche man durch Verschlüsselung schützen möchte.

Die verschlüsselten Daten können dann z.B. mit #MSG SAVE [▶ 373] in einer Datei gesichert oder über V.E.-Variablen der SPS zur Verfügung gestellt werden.

String verschlüsseln	ENCRYPT ["Schluessel", "String"]

Hinweis

Aufgrund einer EU-Exportbestimmung ist die Funktion zur Entschlüsselung nicht mehr verfügbar.

Das Ergebnis von ENCRYPT wird Variablen vom Typ String zugewiesen. Dabei ist folgendes zu beachten:

- Die Stringvariable muss mindestens die doppelte Länge des zu verschlüsselnden Strings haben.

Programmierbeispiel

String verschlüsseln und sichern in einer Datei

```
N10 V.E.encrypted = ENCRYPT[ "Key", "String zum Verschluesseln" ]
N20 #MSG SAVE ["Encrypted Text = %s", V.E.encrypted ]
..
M30
```

2.5.2 Operationen für Zeichenketten (Strings)

Übersicht über alle verfügbaren Operationen:

Stringoperationen:

Zusammenfügen von Strings	Mit + werden 2 Strings aneinandergehängt.
	`V.E.str = "Hello" + " world!"`
	`(-> Ergebnis ist "Hello world!")`

+	

Linken Teilstring ermitteln	LEFT liefert den linken Anfangsstring eines Strings. Hole *anz* Zeichen aus dem String *str*, basierend auf dem ersten Zeichen.
	`V.E.str = LEFT["Hello world!", 5]`
	`(-> Ergebnis ist "Hello")`

LEFT[str, anz]	

Mittleren Teilstring ermitteln	MID liefert den Teilstring eines Strings. Hole *anz* Zeichen aus dem String *str*, beginnend mit dem Zeichen an der Stelle *pos*.
	`V.E.str = MID["How are you?", 3, 5]`
	`(-> Ergebnis ist "are")`

MID[*str, anz, pos*]	

Rechten Teilstring ermitteln	RIGHT liefert den rechten Endstring eines Strings. Hole *anz* Zeichen aus dem String *str*, basierend auf dem letzten Zeichen.
	`V.E.str = RIGHT["Hello world! How are you?", 12]`
	`(-> Ergebnis ist "How are you?")`

RIGHT[*str, anz*]	

Stringlänge ermitteln	LEN liefert die Länge (Anzahl Zeichen) eines Strings.
	`P1 = LEN["Hello world! How are you?"]`
	`(-> Ergebnis ist 25)`

LEN[*str*]	

Zeichenwert ermitteln **[ab V3.1.3079.21]**	ORD liefert den numerischen Wert eines Zeichens in einem String an der Stelle *pos*.
	`P1 = ORD["Hello world!", 1]`
	`(-> Ergebnis ist 72, Zeichenwert von "H")`
	`P2 = ORD["Hello world!", 7]`
	`(-> Ergebnis ist 119, Zeichenwert von "w")`
	Wird keine Position *pos* angegeben, so wird der Wert des ersten Zeichens zurückgegeben.
	Wird eine Position *pos* angegeben, die größer ist als die Länge der Zeichenkette, dann wir der Fehler ID 21545 ausgegeben.
	Für ASCII-Zeichen gibt die Funktion ORD exakt den ASCII-Wert zurück.
	Bei leeren Zeichenketten ist der Rückgabewert 0.

ORD[*str, pos*] oder ORD[*str*]	

Hinweis

Bei FIND[..] wird zwischen Groß- und Kleinbuchstaben unterschieden!

Teilstring suchen	FIND sucht einen String *str2* in einem String *str1* und liefert als Ergebnis die Position der ersten Übereinstimmung von *str2* in *str1*.

FIND[*str1, str2*]	`V.E.str1 = "Hello world! How are you?"`
	`V.E.str2 = "How"`
	`P1 = FIND[V.E.str1, V.E.str2]`
	`(-> Ergebnis ist 14)`
	
	Ist String *str2* in String *str1* nicht vorhanden, so liefert FIND als Ergebnis den Wert 0.
	`V.E.str1 = "Hello world! How are you?"`
	`V.E.str2 = "today"`
	`P1 = FIND[V.E.str1, V.E.str2]`
	`(-> Ergebnis ist 0)`

Teilstring löschen	DELETE löscht in einem String *str* eine bestimmte Anzahl Zeichen *anz,* beginnend mit dem Zeichen an der Stelle *pos*.
DELETE[*str, anz, pos*]	`V.E.str = DELETE["Hello world! How are you?", 5, 7]` `(-> Ergebnis ist "Hello ! How are you?")`

Teilstring einfügen	INSERT fügt einen String *str2* in einen String *str1* ein, beginnend **nach** dem Zeichen an der Stelle *pos*.

INSERT[*str1, str2, pos*]	`V.E.str1 = "Hello ! How are you?"`
	`V.E.str2 = "world"`
	`V.E.str = INSERT[V.E.str1, V.E.str2, 6]`
	`(-> Ergebnis ist "Hello world! How are you?")`

Teilstring ersetzen	REPLACE ersetzt eine Anzahl Zeichen *anz* in einem String *str1* durch den Teilstring *str2,* beginnend mit dem Zeichen an der Stelle *pos*.

REPLACE[*str1, str2, anz, pos*]	`V.E.str1 = "What is your name?"`
	`V.E.str2 = "age"`
	`V.E.str = REPLACE[V.E.str1, V.E.str2, 4, 14]`
	`(-> Ergebnis ist "What is your age?")`

Strings kombinieren **[ab V3.1.3079.40]**	FSTRING generiert Strings mit dynamischem Inhalt. Die Anzahl der Teilstrings ist prinzipiell unbeschränkt und besteht aus statischen und dynamischen Elementen.
	• Statische Elemente sind vorgegebene Strings ("*str_stat*"). Diese werden direkt in den resultierenden String übernommen.
	• Dynamische Elemente sind alle erlaubten Operationen mit String- und Zahlenwerten (*str_dyn*). Das Ergebnis dieser Berechnung wird dann ebenfalls als String in den resultierenden String übernommen.
	Das Ergebnis einer FSTRING-Operation ist ein String. Mit diesem kann auch wieder mit einem Plus (oder anderen für Strings zulässigen Operatoren) weitergerechnet werden:
	• FSTRING[…] + FSTRING[…]
	• FSTRING[…] + „anderer String“ ..
	Auch Verschachtelungen von FSTRING sind damit möglich:
	• FSTRING[.., FSTRING[..]]
	`V.L.Number = 123`
	`V.L.Float = 3.57`
	`V.E.str = FSTRING["V.L.Number: ", V.L.Number,`
	`" / V.L.Float: ", V.L.Float]]`
	`(-> Ergebnis ist: "V.L.Number: 123 / V.L.Float: 3,57")`

FSTRING["*str_stat*", *str_dyn*, {"*str_stat*", *str_dyn*,}]	

Vergleichsoperatoren:

Hinweis

Bei Vergleichsoperationen wird zwischen Groß- und Kleinbuchstaben unterschieden!

Gleichheit	`V.E.str1 = "Peter"`
	`V.E.str2 = "Peter"`
	`$IF V.E.str1 == V.E.str2`
	`#MSG ["%s ist gleich %s!", V.E.str1, V.E.str2]`
	`$ELSE`
	`#MSG ["Strings sind nicht gleich!"]`
	`$ENDIF`
	`(-> Ergebnis ist "Peter ist gleich Peter")`

==	

Ungleichheit	`V.E.str1 = "Peter"`
	`V.E.str2 = "Steve"`
	`$IF V.E.str1 != V.E.str2`
	`#MSG ["%s ist ungleich %s!", V.E.str1, V.E.str2]`
	`$ELSE`
	`#MSG ["Strings sind gleich!"]`
	`$ENDIF`
	`(-> Ergebnis ist "Peter ist ungleich Steve")`

!=	

größer bzw. größer gleich > >=	V.E.str1 = "Peter" V.E.str2 = "Peter" $IF V.E.str1 > V.E.str2 #MSG ["%s ist groesser als %s!", V.E.str1, V.E.str2] $ELSEIF V.E.str1 >= V.E.str2 #MSG ["%s ist groesser/gleich %s!", V.E.str1, V.E.str2] $ENDIF (-> Ergebnis ist "Peter ist groesser/gleich Peter!") V.E.str1 = "Peter" V.E.str2 = "Bob" $IF V.E.str1 > V.E.str2 #MSG ["%s ist groesser als %s!", V.E.str1, V.E.str2] $ELSEIF V.E.str1 >= V.E.str2 #MSG ["%s ist groesser/gleich %s!", V.E.str1, V.E.str2] $ENDIF (-> Ergebnis ist "Peter ist groesser als Bob!")

kleiner bzw. kleiner gleich < <=	V.E.str1 = "Peter" V.E.str2 = "Peter" $IF V.E.str1 < V.E.str2 #MSG ["%s ist kleiner als %s!", V.E.str1, V.E.str2] $ELSEIF V.E.str1 <= V.E.str2 #MSG ["%s ist kleiner/gleich %s!", V.E.str1, V.E.str2] $ENDIF (-> Ergebnis ist "Peter ist kleiner/gleich Peter!") V.E.str1 = "Bob" V.E.str2 = "Tim" $IF V.E.str1 < V.E.str2 #MSG ["%s ist kleiner als %s!", V.E.str1, V.E.str2] $ELSEIF V.E.str1 <= V.E.str2 #MSG ["%s ist kleiner/gleich %s!", V.E.str1, V.E.str2] $ENDIF (-> Ergebnis ist "Bob ist kleiner als Tim!")

Konvertierungsfunktionen:

Integer nach String	INT_TO_STR[...]	V.E.str = INT_TO_STR[123]
Real nach String	REAL_TO_STR[...]	V.E.str = REAL_TO_STR[12.34]
String nach Integer	STR_TO_INT[...]	V.E.sgn32 = STR_TO_INT["12"]
String nach Real	STR_TO_REAL[...]	V.E.real64 = STR_TO_REAL["123.45"]

2.5.3 Zugeordnete Adressbuchstaben

Im eingesetzten Decoder sind die folgenden Adressbuchstaben festen Bedeutungen zugeordnet.

Die Technologie betreffende Adressbuchstaben:

D,d	*<int, double, expr>*	Werkzeugkorrekturaufruf
E,e	*<int, double, expr>*	Vorschub am Satzende
F,f	*<int, double, expr>*	Vorschub am Satzanfang
H,h	*<int, double, expr>*	Hilfsfunktion
M,m	*<int, double, expr>*	Schaltfunktion
S,s	*<int, double, expr>*	Spindeldrehzahl, Synchronverhältnis etc.
T,t	*<int, double, expr>*	Werkzeugplatzaufruf

Die Geometrie betreffende Adressbuchstaben:

G,g	*<int, double, expr>*	Wegbedingung
I,i	*<int, double, expr>*	Interpolationsparameter für 1. Bahnachse
J,j	*<int, double, expr>*	Interpolationsparameter für 2. Bahnachse
K,k	*<int, double, expr>*	Interpolationsparameter für 3. Bahnachse
R,r	*<int, double, expr>*	Kreisradius

Den Programmablauf betreffende Adressbuchstaben:

L,l	*<string>*	Unterprogrammaufruf, global
LL,ll	*<string>*	Unterprogrammaufruf, lokal
N,n	*<int, double, expr>*	Satznummer
O,o	*<int, double, expr>*	Nicht belegt
$		Kennung für Steuersatzanweisungen
#		Kennung für erweiterte Sprachelemente

Die Arithmetik betreffende Adressbuchstaben:

P,p	*<int, double, expr>*	Parameter

Variabel und per Kanalparametersatz zuweisbar sind die Adressbuchstaben zur Bezeichnung der numerischen Achsen. Üblicherweise werden die Buchstaben X, Y und Z zur Bezeichnung der 3 Linearachsen eines kartesischen Raumkoordinatensystems sowie A, B und C zur Bezeichnung von Rundachsen verwendet (Klein-/Großbuchstaben werden unterschieden). Nach DIN 66025 wird mit U, V und W die zweite Bewegung parallel zum Raumkoordinatensystem angegeben.

Neben dieser einfachen Möglichkeit können Achsbezeichnungen auch aus mehreren Zeichen (Strings) bestehen (X_ACHSE, Y22, ZA3). Zur Unterscheidung des Achsnamens von der Koordinatenangabe ist in diesem Fall das "="-Zeichen zu verwenden, also X1=120.345.

Für die folgenden Kapitel der Programmieranleitung werden hauptsächlich die Erstgenannten und häufig verwendeten Adressbuchstaben eingesetzt.

2.5.4 Programmbeispiele

Zur Verdeutlichung der bis jetzt vorgestellten Sachverhalte wird im Vorgriff auf die noch folgenden Kapitel ein Programmierbeispiel vorgestellt.

Folgende Adressbuchstaben finden Verwendung:

Wegbedingungen:

G00 "Fahre im Eilgang"

G01 "Fahre auf linearer Bahn"

Spindel:

S1000 "Spindeldrehzahl 1000 U/min"

Vorschub:

F5000 "Vorschub 5000 mm/min"

Numerische Achsen:

X, Y, Z "drei kartesische Achsen"

Maschinenfunktionen:

M03 "Spindelrechtslauf mit programmierter Drehzahl"

M05 "Spindel Halt"

M30 "Programmende"

Programmierbeispiel

Zusammenfassung der bislang vorgestellten Befehle

```
% 100
N10 G00 X100 Y100           ;Fahre im Eilgang auf X=Y=100
N20 Z100                    ;Fahre im Eilgang auf Z=100. G00
                            ;bleibt wirksam, bis Abwahl durch
                            ;eine andere G-Funktion erfolgt
N30 G1 Z50 F5000 S1000 M3   ;Spindelrechtslauf 1000 U/min und
                            ;Fahre linear mit Vorschubgeschw.
                            ;5000 mm/min auf Z=5
N40 Z100                    ;Fahre mit Vorschubgeschw. auf Z=100
N50 G0 X200 Y200 Z200       ;Fahre im Eilgang auf X=Y=Z=200
N50 M5                      ;Spindel Halt
N60 M30                     ;Programmende
```

3 Weginformationen

3.1 Achsbefehle

Achsbezeichnungen sind konfigurierbar und müssen der konfigurationsspezifischen Beschreibung [1] [▶ 894]-5 entnommen werden. Bei deren Decodierung wird zwischen Groß- und Kleinschreibung unterschieden.

Als Achsbezeichnungen stehen zur Verfügung:

- Einzelne Adressbuchstaben: {A, B, C, U, V, W, X, Y, Z, Q}
- Nach der Programmierung einer Achsbezeichnung, die nur aus einem Adressbuchstaben besteht, muss nach dem Positionswert vor dem nächsten Zeichen ein Leerzeichen stehen, um bei einer nachfolgenden Zuweisung durch das Gleichheitszeichen nicht Verwechslungen hervorzurufen.

Beispiel

Es existieren die Achsbezeichungen "X" und "X50P1" im NC-Kanal und Achse "X" soll auf Position "50" gefahren werden.

X50P1=7	(FEHLER)	X50P1-Achse fährt auf Position 7.
X50 P1=7	(RICHTIG)	X-Achse fährt auf Position 50.

- Strings (z.B. X_SCHLITTEN, X1, Y22, Z_ACHSE).
- Das erste Zeichen des Strings muss einem der reservierten Adressbuchstaben (s. o.) entsprechen. Als weitere Zeichen dürfen auch die Ziffern 0-9 verwendet werden. Die Stringlänge der Achsbezeichnung darf die maximal mögliche Länge (fest vorgegeben) nicht überschreiten, ansonsten wird eine Fehlermeldung ausgegeben.
  Um Mehrdeutigkeiten zu vermeiden, muss nach allen Achsbezeichnungen, die mehr als ein Zeichen umfassen, vor der Positionsangabe ein Gleichheitszeichen stehen.
  Dies ist insbesondere für Achsbezeichnungen notwendig, die mit einer der Ziffern 0 –9 enden.

Hinweis

Nach Achsbezeichnern, die mehr als ein Zeichen umfassen, muss ein Gleichheitszeichen folgen.

X1 = <int, double, expr>

Beispiele:
X1 = 100.0
X22 = 0.001
X_SCHLITTEN = SIN [30]
Z_ACHSE = SQRT [2]/2

Außerdem gelten folgende Vereinbarungen:

- Jede Achsbezeichnung muss im Kanalparametersatz [1] [▶ 894]-5 vorgegeben werden.
- Einer Achsbezeichnung muss immer ein Zahlenwert oder ein Ausdruck folgen:

X <int, double, expr>

Beispiele:
X 100.0
Y 0.001
Z SIN [30]
A SQRT [2]/2
B 4 * R1/R2

Programmierbeispiel

Achsbefehle

```
;Verwendete Achsbezeichnungen:
;Y, Y50, Y_ACHSE_SCHL_1, Z7

N010 G01 F1500
N020 Y50 = 51                    ;Achse Y50 auf Position 51
N030 Y52                         ;Achse Y auf Position 52
N040 Y50 Z7 = 54                 ;Achse Y auf Position 50 und
                                 ;Achse Z7 auf Position 54
N050 Y 70 Z7 = 55                ;Achse Y auf Position 70 und
                                 ;Achse Z7 auf Position 55
N060 Y = 71 Z7 = 56              ;Achse Y auf Position 71 und
                                 ;Achse Z7 auf Position 56
N070 Y[2+3]                      ;Achse Y auf Position 5
N080 Y50 = [4*3]                 ;Achse Y50 auf Position 12
N090 Y_ACHSE_SCHL_1 = 23         ;Achse Y_ACHSE_SCHL_1 auf
                                 ;Position 23
N100 Y50 = P1                    ;Achse Y50 auf Position P1
N110 M30
```

In dieser Programmieranleitung werden die gebräuchlichen Bezeichnungen X, Y und Z für die 3 linearen Achsen eines kartesischen Koordinatensystems sowie A und B für 2 weitere Streckenachsen verwendet.

3.2 Maßsysteme, Eingabe- und Genauigkeitsbereiche

Die Maßsysteme für die Angaben von Positionswerten, Winkeln und Geschwindigkeiten sind wie folgt vereinbart:

Längen- und Positionsangaben:	mm oder inch
Lineare Geschwindigkeiten:	mm oder inch pro s oder min
.. oder auch applikationsspezifisch:	m pro min
Winkel:	grd oder ngrd
Winkelgeschwindigkeit:	grd oder ngrd pro s oder min

Darüber hinaus hat der Programmierer die Freiheit, mit Hilfe der Parameterrechnung selbstdefinierte Maßsysteme zu verwenden.

Alle Längenangaben und linearen Geschwindigkeiten werden standardmäßig mit einer Genauigkeit von 0.1 µm berechnet. Der maximale Verfahrbereich, der bei dieser Auflösung eingegeben werden kann, beträgt dann -214 m .... +214 m. Zahleneingaben bei der Programmierung dürfen demnach den Bereich von –214 000.0000 mm bis +214 000.0000 mm nicht überschreiten Dies gilt nicht für einzelne Elemente (z.B. Parameter) eines arithmetischen Ausdrucks, wenn das Ergebnis des arithmetischen Ausdrucks innerhalb des genannten Bereiches liegt. Es ist zu berücksichtigen, dass dieser Zahlenbereich auch nicht in Verbindung mit Verschiebungen und Korrekturen überschritten werden darf.

Ausnahme: Die Angabe des Kreisradius ist bis maximal 10^9 mm möglich. Der Endpunkt des Kreisbogens muss aber immer innerhalb des maximalen Verfahrbereiches der Achsen (-2,14 ... +2,14) * 10^5 mm liegen.

Im Kanalparametersatz ist der Achstyp einzutragen. Handelt es sich um eine rotatorische Achse (Rundachse oder Spindel), so werden Winkel standardmäßig mit einer Auflösung von 0.0001° behandelt. Damit ist ein Bereich von n * 360° mit n=1190 programmierbar.

Bei einer Auflösung von z.B. 0,0001° sind Winkel in einem Bereich von

(-2,14 ... +2,14) * 10^5 / 360 = 2 * 594 Umdrehungen möglich.

3.3 Koordinatensysteme

Nach der Referenzpunktfahrt befindet sich die Steuerung im Maschinennullpunkt bzw. im Maschinenkoordinatensystem. Erfolgen nun Eingaben aus dem NC-Programm (z.B. X100), so fallen die programmierten Koordinaten (*Index p*) mit den absoluten Koordinaten (*Index a*) zusammen:

$$x_a = x_p$$
$$y_a = y_p$$

Verschiebungen entstehen durch die Definition von Werkstück-Koordinatensystemen, die sich vom dem durch die physikalischen Maschinenachsen aufgespannten Koordinatensystem hinsichtlich ihrer Lage im Raum unterscheiden. Hierbei ist zu differenzieren zwischen programmierten, konstanten, translatorischen Verschiebungen in einer Einzelachse und Verschiebungen, die sich aufgrund von kinematischen (z.B. zylindrisch<->kartesisch) oder geometrischen (z.B. Werkzeugradiuskorrektur, Spiegelung) Transformationen dynamisch ergeben und im Allgemeinen auf mehrere Achsen wirken.

Zum Beispiel kann der Nullpunkt durch eine Nullpunktsverschiebung (NPV-G54...G59) vom Maschinennullpunkt M auf einen frei wählbaren Werkstücknullpunkt W bzw. ein Werkstückkoordinatensystem verschoben werden. Die absoluten Koordinaten ergeben sich damit aus der Addition von NPV und programmierten Koordinaten:

$$x_a = x_{NPV} + x_p$$
$$y_a = y_{NPV} + y_p$$

Unabhängig von diesen über den Nullpunktsverschiebungsdatensatz festlegbaren NPVs können durch zusätzliche Verschiebungsarten wie z.B. mit G92 X... Y... Z... explizit weitere Verschiebungen im Teileprogramm programmiert werden.

Diese s.g. Bezugspunktverschiebung (BPV) addiert sich zu der vorhergehenden NPV auf. Damit lassen sich die absoluten Koordinaten wie folgt bestimmen:

$$x_a = x_{NPV} + x_{BPV} + x_p$$
$$y_a = y_{NPV} + y_{BPV} + y_p$$

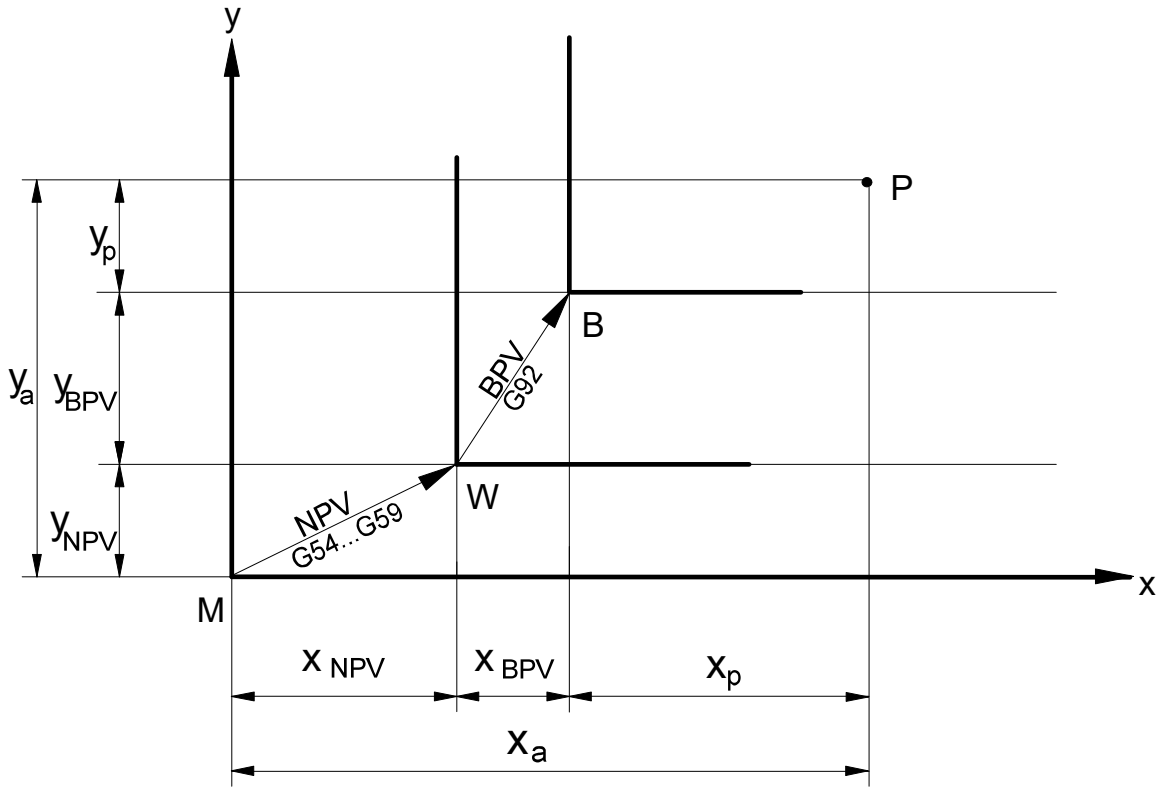


Abb. 1: Definition eines Werkstückkoordinatensystems mit NPV u. BPV (Legende s. unten)

xa, ya :	absolute Koordinaten
xp, yp :	programmierte Koordinaten
xNPV, yNPV :	Nullpunktsverschiebung
xBPV, yBPV :	Bezugspunktverschiebung
M :	Maschinennullpunkt
W :	Werkstücknullpunkt
B :	Bezugspunkt für die Koordinatenangaben
P :	Position

Durch die Koordinatenanzeige der Bedienoberfläche lassen sich aktive Verschiebungen an einer bleibenden Differenz zwischen den Koordinaten der physikalischen Maschinenachsen (ACS) und den Werkstückkoordinaten (PCS) erkennen. Einige Verschiebungen entstehen jedoch durch Manipulation der Maschinen- und Werkstückkoordinaten gleichermaßen (z.B. Werkzeugradiuskorrektur, Spiegelung) und führen deshalb nicht zu einer Koordinatendifferenz.

Folgende Tabellen geben im Vorgriff auf die weitere Dokumentation einen Überblick der zusätzlichen Verschiebungsarten. Hierbei gilt:

Aktivierung und **Deaktivierung** meint den Zeitpunkt, an dem die Verschiebung als Koordinatendifferenz bzw. Koordinatenänderung an der Bedienoberfläche sichtbar wird. Physikalisch wirksam wird eine Verschiebung jedoch grundsätzlich frühestens mit der ersten, auf die Aktivierung bzw. Deaktivierung folgenden Bewegung. Eine Deaktivierung am Programmende führt beispielsweise erst mit der ersten Bewegung des Folgeprogramms zu einer Ausgleichsbewegung.

Programmierbare Verschiebungen (linear, konstant):

Nr.	Bezeichnung	Definition	ACS – PCS Differenz wenn aktiv	Aktivierung	Deaktivierung	Temporäre Unterdrückung
1	Bezugspunkt-verschiebung	NC-Prg.	Ja	NC-Satz „*G92 X.. Y..*“ G90/91 Abhän-gigkeit beachten	NC-Satz „*G92 X0 Y0*“ oder NC-Programm-start	„#SUPPRESS OFFSETS“ „#MCS ON“
2	Nullpunkt-ver-schiebung	Liste NC-Prg.	Ja	NC-Satz „*G54...G59*”	NC-Satz „G53“ oder NC-Programm-start	„#SUPPRESS OFFSETS“ „#MCS ON“
3	Platzversatz	Liste	Ja	Programmstart Während Pro-gramm nicht änderbar	NC-Programm-start Während Pro-gramm nicht än-derbar	„#SUPPRESS OFFSETS“ „#MCS ON“
4	Werkzeug-versatz	Liste NC-Prg. Extern	Ja	NC-Satz „*D..*“	NC-Satz „D0“ oder NC-Programm-start	„#SUPPRESS OFFSETS“ „#MCS ON“
5	Istwertsetzen	NC-Prg.	Ja	NC-Satz „#PSET..“	NC-Satz „#PRESET...“ oder Programmstart	„#SUPPRESS OFFSETS“ „#MCS ON“
6	CS-Verschiebung	NC-Prg.	Ja	NC-Satz „*#CS ON[vx,vy,vz,..*“	NC-Satz „#CS OFF“ oder NC-Program-mende	„#MCS ON“
7	ACS-Verschie-bung	NC-Prg.	Ja	NC-Satz „*#ACS ON[vx,vy,vz,..*“	NC-Satz „#ACS OFF“ oder NC-Program-mende	„#MCS ON“

Verschiebungen, die sich durch geometrische Transformation ergeben (linear, dynamisch):

Nr.	Bezeichnung	Definition	ACS – PCS Differenz wenn aktiv	Aktivierung	Deaktivierung	Temporäre Unterdrückung
8	CS	NC-Prg.	Ja	NC-Satz „#CS ON[..,..,..]“	NC-Satz „#CS OFF” oder NC-Program-mende	„#MCS ON“
9	ACS	NC-Prg.	Ja	NC-Satz „#ACS ON[..,..,..]“	NC-Satz „#ACS OFF” oder NC-Program-mende	„#MCS ON“
10	Konturrotation	NC-Prg.	Nein	NC-Satz „#ROTATION ON[ANGLE..]”	NC-Satz „#ROTATION OFF”	„#MCS ON“
11	Spiegelung	NC-Prg.	Nein	NC-Bewegungs-satz nach G21/22/23	NC-Bewegungs-satz nach G20	Nicht unterdrückbar
12	Werkzeug-radi-uskorrektur	NC-Prg.	Nein	NC-Satz G41/42	NC-Satz G40	Nicht unterdrückbar
13	Kinematische Transformation	NC-Prg.	Nein	NC-Satz „#TRAFO ON“ Programmstart-automatik	NC-Satz „#TRAFO OFF	„#MCS ON“

Verschiebungen durch spezielle Funktionen:

Nr.	Bezeichnung	Definition	ACS – PCS Differenz wenn aktiv	Aktivierung	Deaktivierung	Temporäre Unterdrückung
14	Verschiebung durch Handbetrieb mit paralleler Interpolatuion	Handrad NC-Prg.	Ja	NC-Satz „G201“	NC-Satz „G202”	Nicht unterdrückbar
15	Verschiebung durch Messfahrt	NC-Prg.	Ja	NC-Satz „G101“	NC-Satz „G102“	„#SUPPRESS OFFSETS“ „#MCS ON“
16	Verschiebung nach Referenz-punktfahrt	NC-Prg.	Nein	NC-Satz „G74 X.. Y.. Z.. “	Nicht möglich	Nicht unterdrückbar

Der Befehl #SUPPRESS OFFSETS wirkt nur innerhalb eines NC-Satzes.

Der Befehl #MCS ON deaktiviert Verschiebungen bis zur Programmierung von #MCS OFF.

Innerhalb jedes (A)CS werden die Verschiebungstypen 1, 2 und 5 „lokal“ gespeichert.

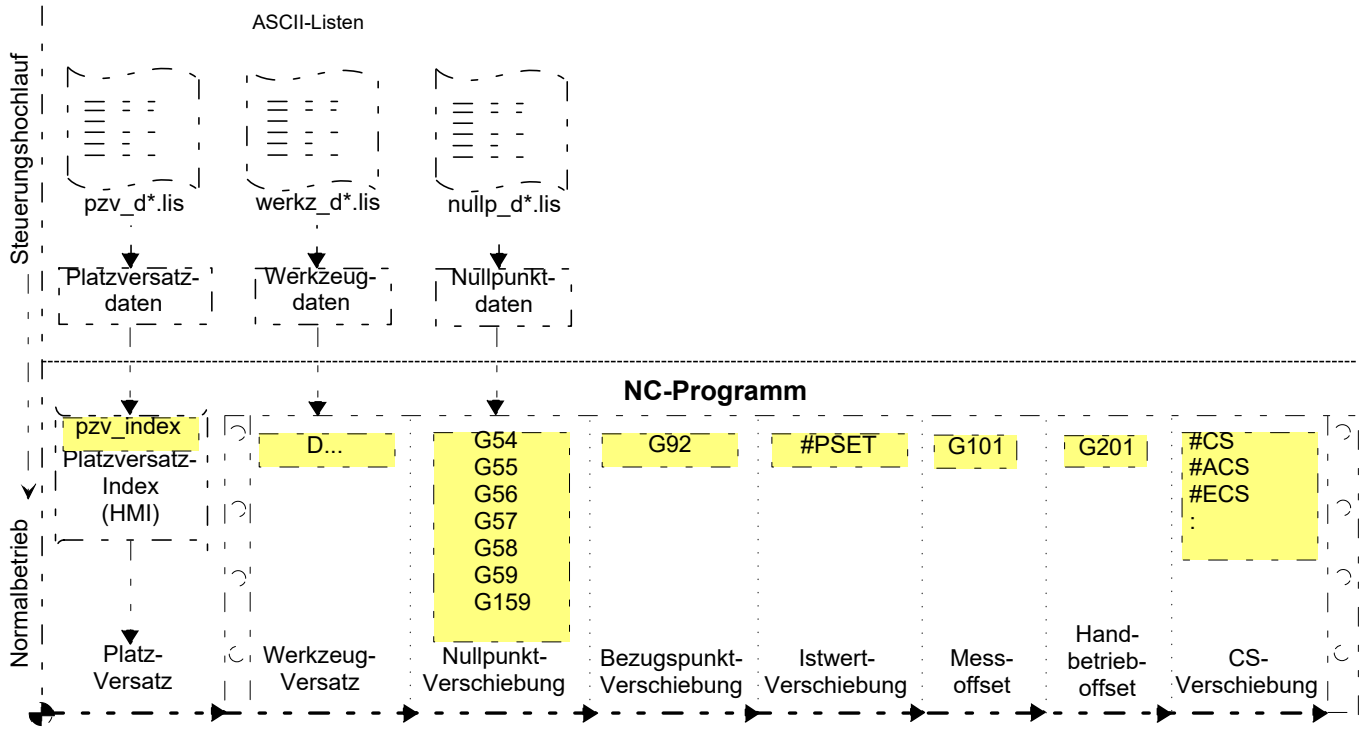


Abb. 2: Übersicht der zusätzlichen Verschiebungen und Koordinatensysteme

4 Die G-Funktionen

Eine vollständige Liste der G-Funktionen findet sich in der Befehlsübersicht im Anhang unter G-Funktionen (G..) [▶ 877].

Die "G"-Funktionen beschreiben die Art der Vorschubbewegung, Interpolationsart, Vermaßungsart, schalten zeitliche Beeinflussungen und aktivieren bestimmte Betriebszustände. Die Syntax besteht aus dem Buchstaben G in Kombination mit einer Kennzahl. Der Kennzahl ist eine Bedeutung fest zugeordnet und ihr kann optional eine 0 vorangestellt sein (z.B. haben G01 und G1 die gleiche Bedeutung):

Syntax:

G[0]<*id*>

Bei den G-Funktionen sind verschiedene Unterscheidungsmerkmale zu berücksichtigen:

Wirksamkeit:

Es gibt G-Funktionen, die nach ihrer Programmierung in ihrer Bedeutung nur für den jeweiligen Satz Gültigkeit besitzen (satzwirksam, **nicht modal**) und solche, die nach der erstmaligen Programmierung in ihrer Bedeutung solange gültig sind, bis sie explizit abgewählt werden (haltend wirksam, **modal**).

Ausschluss:

Bestimmte G-Funktionen schließen sich gegenseitig aus. Es kann z.B. nicht gleichzeitig G01 (Geradeninterpolation) und G02 (Kreisinterpolation) angewählt sein. Die in diesen Gruppen zusammengefassten Funktionen dürfen also nicht in einem NC-Satz programmiert werden.

Eine haltend wirksame Funktion wird automatisch abgewählt, wenn in einem nachfolgenden Satz eine andere Funktion der gleichen Gruppe angewählt wird.

Programmierbeispiel

G-Funktionen

```
:
N50 G01 X100 Y200                ; Geradeninterpolation wirksam
N60 G41 X200 Y200                ; Geradeninterpolation wirksam
N70 X300 Y250                    ; Geradeninterpolation wirksam
N80 X100 Y50                     ; Geradeninterpolation wirksam
N90 G02 X100 Y50 I100            ; Kreisinterpolation wirksam
:
```

Grundstellung:

Beim Einschalten, nach RESET oder am Programmende befindet sich die Steuerung in der Grundstellung. In dieser Grundstellung sind bereits einige G-Funktionen ohne explizite Anwahl wirksam.

4.1 Wegbedingungen

4.1.1 Eilgang (G00)

Syntax:

G00 Linearinterpolation im Eilgang modal

Bei angewähltem G00 wird für die Verfahrgeschwindigkeit die Eilganggeschwindigkeit der Achsen (festgelegt in den Maschinenparametern) zugrunde gelegt. Dabei ergibt sich die Geschwindigkeit der Achsen so, dass mindestens eine Achse mit ihrer Eilganggeschwindigkeit bewegt wird.

Es können beliebige Geraden im kartesischen Raumkoordinatensystem (X, Y, Z) programmiert werden. Alle programmierten Mitschleppachsen werden mit linearer Geschwindigkeit so bewegt, dass der Start und das Ende ihrer Bewegung zeitgleich mit den Hauptachsen erfolgen.

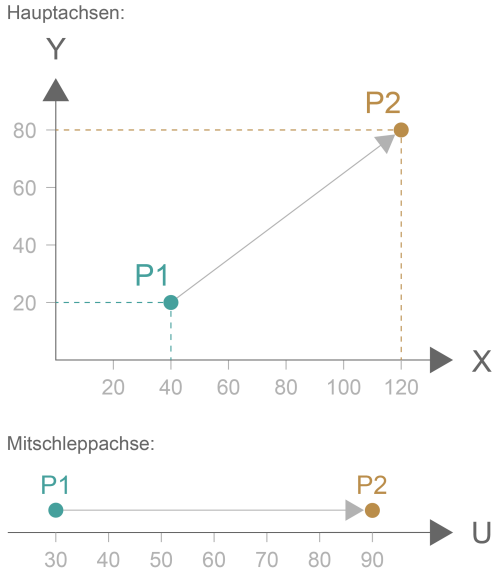


Abb. 3: Positionierung im Eilgang mit u.g. Parametern

Programmierbeispiel

Eilgang G00

```
N05 G00 G90 X40 Y20 U30        ;fahre zu Ausgangslage P1
;Absolutmaßeingabe:
N10 G00 G90 X120 Y80 U90       ;fahre von P1 nach P2
;Kettenmaßeingabe:
N10 G00 G91 X80 Y60 U60        ;fahre von P1 nach P2
```

Sonderfall: Richten einer Rundachse mit G00 G90

Wenn eine Rundachse über G00 mit wirksamem G90 programmiert wird (G90 : Absolute Programmierung), so wird die programmierte Zielposition modulo berechnet, d.h. die Rundachse bewegt sich maximal eine halbe Umdrehung.

4.1.2 Geradeninterpolation (G01)

Syntax:

G01 Linearinterpolation mit programmiertem Vorschub modal

Bei angewähltem G01 wird der programmierte Weg mit der unter dem F-Wort (z.B. mm/min) angegebenen Vorschubgeschwindigkeit auf einer Geraden zur Zielposition verfahren. Es können beliebige Geraden im kartesischen Raumkoordinatensystem (X, Y, Z) programmiert werden. Alle programmierten Mitschleppachsen werden mit linearer Geschwindigkeit so bewegt, dass der Start und das Ende ihrer Bewegung zeitgleich mit den Hauptachsen erfolgen.

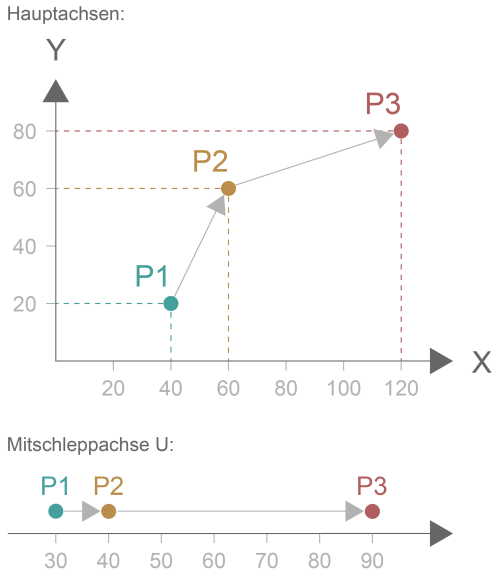


Abb. 4: Grafische Darstellung der Geradeninterpolation (G01)

Programmierbeispiel

Geradeninterpolation G01

```
N05 G00 G90 X40 Y20 U30       ; fahre zu Ausgangslage P1
; Absolutmaßeingabe:
N10 G01 G90 X60 Y60 U40 F1000 ;fahre von P1 nach P2 Vorschub 1000 mm/min
N20 X120 Y80 U90 ;fahre von P2 nach P3 Vorschub 1000 mm/min
; Kettenmaßeingabe:
N10 G01 G91 X20 Y40 U10 F1000 ;fahre von P1 nach P2 Vorschub 1000 mm/min
N20 X60 Y20 U50 ;fahre von P2 nach P3 Vorschub 1000 mm/min
```

4.1.3 Kreisinterpolation (G02/G03)

Syntax:

G02	Kreisinterpolation im Uhrzeigersinn (CW)	modal
G03	Kreisinterpolation im Gegenuhrzeigersinn (CCW)	modal

Bei angewähltem G02 oder G03 wird der programmierte Weg mit der unter dem F-Wort (z.B. mm/min) angegebenen Vorschubgeschwindigkeit auf einem Kreis zur Zielposition verfahren. Es können Kreise in den drei Hauptebenen des Raumkoordinatensystems (X-Y, Z-X, Y-Z) gefahren werden. Die Auswahl der Hauptebene erfolgt mit den Funktionen G17, G18, G19 (siehe Ebenenauswahl [▶ 119]).

Alle programmierten Mitschleppachsen werden mit linearer Geschwindigkeit so bewegt, dass der Start und das Ende ihrer Bewegung zeitgleich mit den Hauptachsen erfolgen.

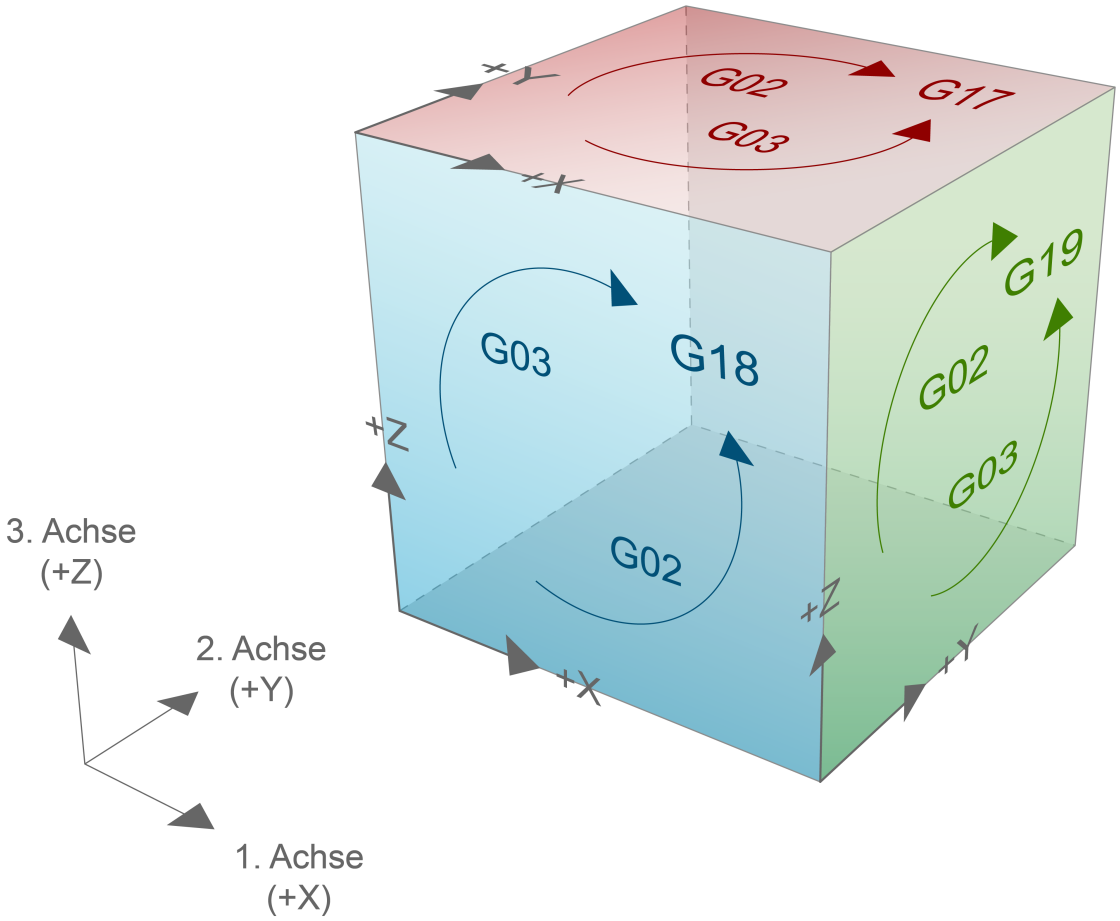


Abb. 5: Darstellung der Kreisfunktionen G02 und G03

Kreisprogrammierung über Mittelpunkt und Kreisendpunkt

Zur Definition des Kreises wird der Kreisanfangspunkt (ist durch den vorhergehenden Satz bestimmt), der Kreisendpunkt und der Kreismittelpunkt "Km" herangezogen.

Syntaxbeispiel für Ebene G17:

G02 | **G03** [**X**.. **Y**..] **I**.. **J**..

G02 | G03	Kreisinterpolation CW / CCW
X.. Y..	Kreisendpunkt in der Ebene XY in [mm, inch]
I..J..	Lage des Kreismittelpunktes der Interpolation in der Ebene XY (I in X, J in Y) in [mm, inch], entsprechend G161/G162

Syntax entsprechend der angewählten Interpolationsebene:

Ebene	Interpolationsart	Endpunkt in Ebene	Mittelpunkt
G17	G02/G03	X.. Y..	I.. J..
G18	G02/G03	Z.. X..	K.. I..
G19	G02/G03	Y.. Z..	J.. K..

Die Angabe des Kreismittelpunktes erfolgt durch die Interpolationsparameter I, J und K relativ zum Kreisanfangspunkt bei wirksamem G162, absolut bei wirksamem G161 (siehe Mittelpunktsangabe bei Kreisdefinition (G161/G162) [▶ 142]).

G162:	**(Grundeinstellung)**
I	relative Lage von Km in X-Richtung
J	relative Lage von Km in Y-Richtung
K	relative Lage von Km in Z-Richtung
G161:	
I	absolute Lage von Km in X-Richtung
J	absolute Lage von Km in Y-Richtung
K	absolute Lage von Km in Z-Richtung

Bei einer falschen Definition des Kreismittelpunktes erfolgt eine Fehlermeldung, wenn keine Kreismittelpunktkorrektur eingeschaltet ist (G165). Bei wirksamem G165 wird ein Mittelpunkt so bestimmt, dass ein Kreis gefahren werden kann. Das bedeutet auch, dass bei nicht angegebenen Interpolationsparametern die Kreismittelpunktkorrektur von I, J, K = 0 ausgeht. (siehe Mittelpunktskorrektursteuerung im Kreis (G164/G165) [▶ 143]).

Die Kreismittelpunktkoordinaten I, J, K sind "nicht-haltend" wirksam.

Sind bei aktivem G02/G03 die Interpolationsparameter I,J, K ohne Kreis**end**punkt programmiert, so wird ein Vollkreis gefahren.

Kreisprogrammierung über Radius und Kreisendpunkt

Alternativ zu I, J, K können Kreise auch mittels Radiusangabe programmiert werden. Dies ist möglich über den Adressbuchstaben R"Radiuswert" oder über G163="Radiuswert". Jedoch kann mit R kein Vollkreis programmiert werden. Die Radiusangabe mit R oder G163= ist "haltend" wirksam und wird bei mehreren kreisförmigen Verfahrbewegungen ohne wiederholte Angabe erneut verwendet.

Syntaxbeispiel für Ebene G17:

G02 | **G03 X**.. **Y**.. **R**..

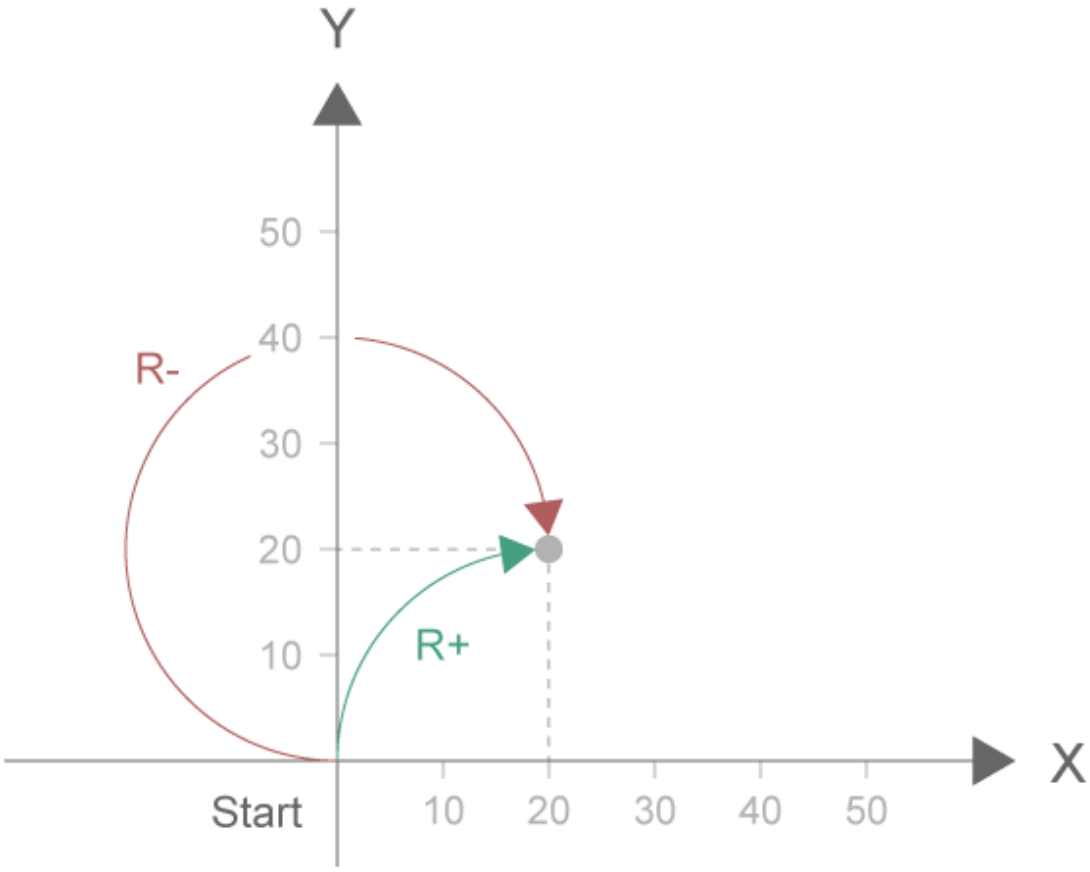
G02 | G03	Kreisinterpolation CW / CCW
X.. Y..	Kreisendpunkt in der Ebene XY in [mm, inch]
R..	Radiuswert des zu interpolierenden Teilkreises in [mm, inch].

Hinweis

Der maximal zulässige Kreisradius beträgt 10^9 mm. Der Endpunkt des Kreisbogens darf jedoch den maximal zulässigen Verfahrbereich der Achsen von +- 2,14*10^5 mm nicht überschreiten.

Hinweis

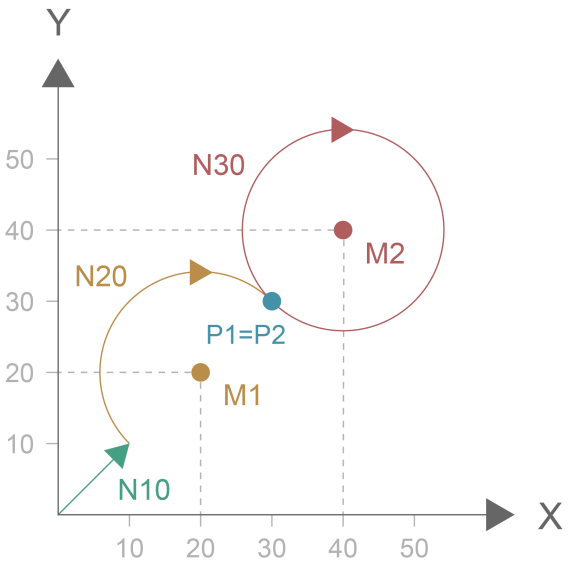
Bei positivem Radiuswert wird ein kürzestmöglicher Kreisbogen, bei negativem Radiuswert der größtmögliche Kreis bestimmt (siehe Bild unten).


Y
50
40
30
20
10
R-
R+
X
Start
10
20
30
40
50

Programmierbeispiel

Kreisinterpolation mit Teil- und Vollkreis

```
N05 G0 X0 Y0
N10 G01 X10 Y10 F1000
N20 G02 X30 Y30 I10 J10          ;Halbkreis um M1 Kreisendpunkt X30 Y30
;Alternativ N20:
N20 G02 X30 Y30 R[10*SQRT[2]]    ;Halbkreis um M1 Kreisendpunkt X30 Y30
N30 G02 I10 J10                  ;Vollkreis um M2
```



Programmierbeispiel

Kreisinterpolation mit programmierter Mitschleppachse

```
;Absolutmaßeingabe:
N05 G90 G00 X40 Y30 U40           ;Kreisanfang (Ka), Ausgangslage
N10 G90 F1000                     ;Absolutmaß, Vorschub
N20 G17                           ;Anwahl der X-Y-Ebene
N30 G03 G161 X60 Y50 I60 J30 U90  ;Kreis: Ka -> Ke u. Gerade: P1 -> P2


;Kettenmaßeingabe:
N05 G90 G00 X40 Y30 U40           ;Kreisanfang (Ka), Ausgangslage
N10 G91 F1000                     ;Kettenmaß, Vorschub
N20 G17                           ;Anwahl X-Y-Ebene
N30 G03 G162 X20 Y20 I20 U50      ;Kreis: Ka -> Ke u. Gerade: P1 -> P2
```

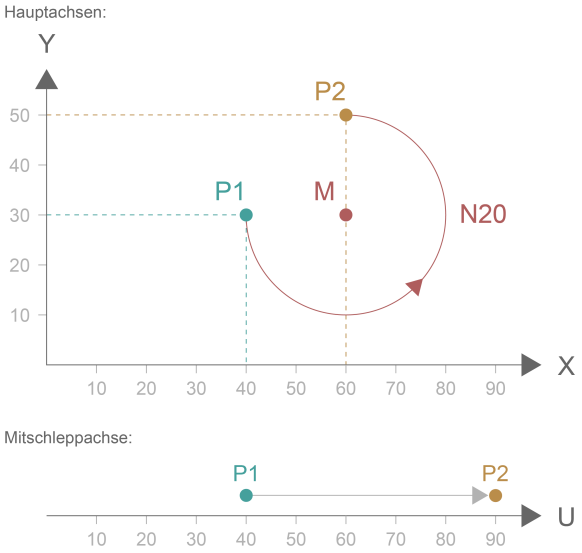


Abb. 6: Beispiele für die Kreisinterpolation

Bei der s.g. indizierten Radiusprogrammierung kann der Radius mit "R1=.." angegeben werden. Hierbei ist als Index nur der Wert 1 erlaubt. Wird "R1" auf der rechten Zuweisungsseite verwendet, so darf der Indexwert 1 nicht als mathematischer Ausdruck programmiert werden.

Programmierbeispiel

Indizierte Radiusprogrammierung (R1=..)

```
;Links- und Rechtsseite Verwendung direkt
N10 R1 = 5
N20 P2 = R1         ;erlaubt

;Linksseitige Verwendung indirekt
N10 P2 = 1
N20 RP2 = 5         ;erlaubt

;Rechtsseite Verwendung indirekt
N10 R1 = 5 P2 = 1
N20 P3 = RP2        ;nicht erlaubt
```

Programmierbeispiel

Diese Beispiele ergeben jeweils Halbkreise mit Radius 50.

```
:
N10  G90 G01 X0 Y0 F500
N20  G02 X100  R50 ;Halbkreis im Uhrzeigersinn
N30  G03 X200  R50 ;Halbkreis im Gegenuhrzeigersinn
```

```
:
N10  G90 G01 X0 Y0 F500
N20  G02 R=50 ;hier noch keine Bewegung
N30  X100 ;Halbkreis im Uhrzeigersinn
```

```
:
N10  G90 G01 X0 Y0 F500
N20  R1=50
N30  G02 X100 ;Halbkreis im Uhrzeigersinn
N40  G03 X200 ;Halbkreis im Gegenuhrzeigersinn
```

```
:
N10  G90 G01 X0 Y0 F500
N20  G02  X100  R1=50 ;Halbkreis im Uhrzeigersinn
N30  G03  X200 ;Halbkreis im Gegenuhrzeigersinn
```

Folgende Programmierung ergibt eine Fehlermeldung, da R1 als Radius mit dem Wert 1 interpretiert wird:

```
N10 G90 G01 X0 Y0 F500
N20 R1=50
N30 G02 X100 R1
```

Alternativ zur Kreisdefinition mit "R" oder "R1" kann eine Kreisradiusvorgabe auch mit G163 erfolgen.

Syntaxbeispiel für Ebene G17:

G02 | **G03 X**.. **Y**.. **G163=..**

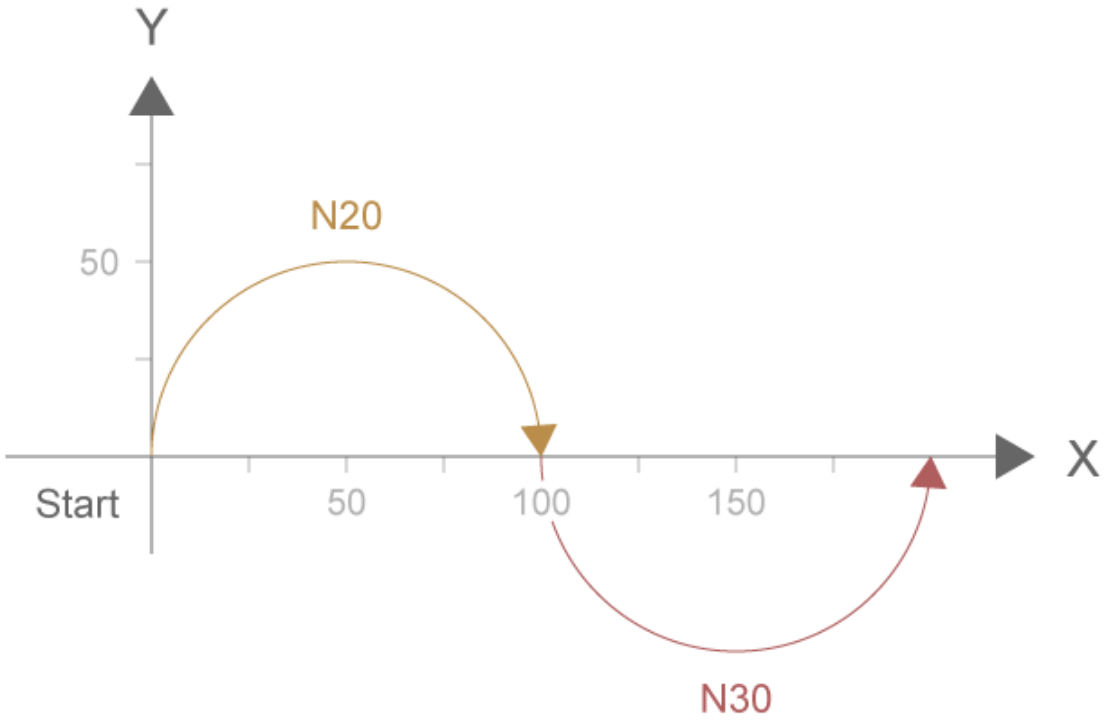
G02 | G03	Kreisinterpolation CW / CCW
X.. Y..	Kreisendpunkt in der Ebene XY in [mm, inch]
G163=..	Radiuswert des zu interpolierenden Teilkreises in [mm, inch].

Die Kreisdefinition über G163 ist bei angewählter Kreisinterpolation bis zur Neudefinition oder bis zur Abwahl über eine I- und/oder J- und/oder K-Angabe gültig.

Programmierbeispiel

Kreisradiusprogrammierung mit G163

```
;N10: Bewegung zum Ursprung
;N20: Halbkreis im Uhrzeigersinn mit Zielwert X100 und unter Vorgabe
;des Kreisradius durch G163 (Radiusangabe wirkt selbsthaltend)
;N30: Halbkreis gegen den Uhrzeigersinn mit Zielwert X200 und Radius,
;der in N20 festgelegt wurde und haltend wirksam ist

%Radiusprogrammierung_G163
N10 G90 G01 X0 Y0 F1000
N20 G02 G163=50 X100 ;Halbkreis im Uhrzeigersinn
N30 G03 X200 ;Halbkreis im Gegenuhrzeigersinn
N40 M30
```



Achtung

Fallen Start- und Endpunkt des mit "R", "R1" oder "G163" programmierten Kreises zusammen, so wird eine Fehlermeldung ausgegeben. Soll also ein Vollkreis gefahren werden, so muss dieser mit I/J/K programmiert werden.

Kreisprogrammierung über Winkel und Mittelpunkt

Als weitere Möglichkeit kann ein Kreis auch über die Angabe eines Winkels und eines Mittelpunktes programmiert werden. Basierend auf diesen Werten wird der Endpunkt des Kreises berechnet und der Kreis interpoliert. Abhängig von der aktiven Ebene müssen hierbei die zugeordneten Mittelpunktkoordinaten verwendet werden.

Funktionalität ist verfügbar ab V3.01.3080.15 bzw. V3.1.3107.49.

`Syntaxbeispiel für Ebene G17:`

G02 | **G03 I**.. **J**.. **#CANG=**..

G02 | G03	Kreisinterpolation CW / CCW
I.. J..	Lage des Kreismittelpunktes der Interpolation in der Ebene XY (I in X, J in Y) in [mm, inch], entsprechend G161/G162
#CANG=..	Kreisöffnungswinkel in Grad [°]. Der Endpunkt wird berechnet.

Programmierbeispiel

Kreisinterpolation mit Mittelpunkt und Winkel

```
N05 G17 G161                ;Ebene G17, Mittelpunktangaben absolut
N07 G00 X0 Y0
N10 G01 G90 X10 Y10 F1000   ;Kreisstartpunkt
N20 G02 I40 J20 #CANG=135   ;Endpunkt P1 wird berechnet
N30 ..
```

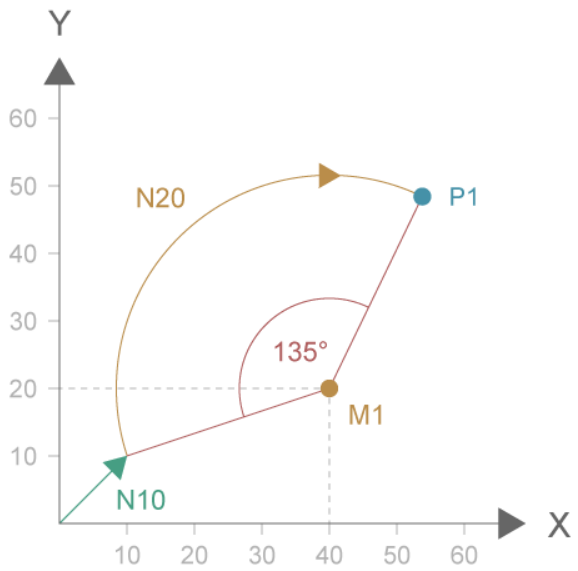


Abb. 7: Kreisinterpolation mit Mittelpunkt und Winkel

4.1.4 Helikalinterpolation (G02 Z.. K../G03 Z.. K..)

Die Helikalinterpolation ist die Überlagerung einer kreisförmigen Interpolation (Ebene der 1. und 2. Hauptachse) und einer linearen Bewegung in der 3. Hauptachse. Die sich ergebende schraubenförmige Bewegung (Helix) erfolgt mit konstanter Steigung. Die Steigung wird abhängig von der angewählten Ebene über den dritten Parameter der Zirkularinterpolation programmiert.

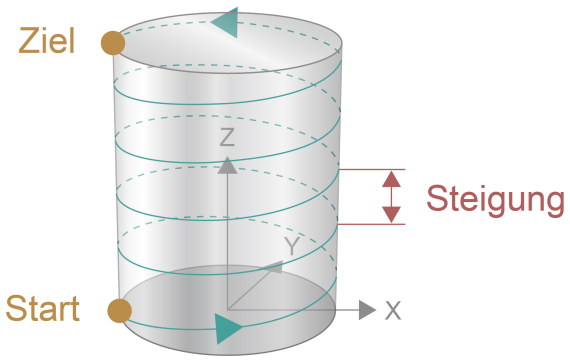


Abb. 8: Darstellung einer Helikalinterpolation mit konstanter Steigung

`Syntaxbeispiel für Ebene G17:`

G02 | **G03 X**.. **Y**.. **Z**..**I**.. **J**.. | **R**.. **K**..

G02 | G03	Kreisinterpolation CW / CCW
X.. Y..	Zielpunkt in der Ebene XY in [mm, inch]
Z..	Zielpunkt auf der Helixachse senkrecht zur Ebene XY in [mm, inch]
I.. J..	Lage des Kreismittelpunktes der Interpolation in der Ebene XY (I in X, J in Y) in [mm, inch], entsprechend G161/G162.
R..	Radius des zu interpolierenden Kreises (alternativ zu I,J) in [mm, inch]
K..	Steigung der Helix in Z (Wert generell ohne Vorzeichen) in [mm, inch]

Syntax entsprechend der angewählten Interpolationsebene:

Ebene	Interpolations-art	Zielpunkt in Ebene	Zielpunkt auf Helixachse	Mittelpunkt /Radius	Steigung
G17	G02/G03	X..Y..	Z..	I..J../R	K
G18	G02/G03	Z..X..	Y..	K..I../R	J
G19	G02/G03	Y..Z..	X..	J..K../R	I

Es ist nicht erforderlich, die Steigung so anzugeben, dass der programmierte Zielpunkt durch die Helix genau erreicht wird. Der NC-Kern berechnet in solchen Fällen unter Berücksichtigung der festen Positionen von Start- und Zielpunkt eine "korrigierte" Steigung, die der programmierten Steigung am nächsten kommt.

Hierzu wird auf Basis der programmierten Steigung zunächst der Zielpunkt der Helix berechnet. Weicht dieser berechnete Zielpunkt vom programmierten Zielpunkt ab, so ist eine Korrektur erforderlich. Kriterium für die Korrektur ist hierbei in Drehrichtung gesehen der Abstand zwischen programmiertem Zielpunkt und berechnetem Zielpunkt.

Ist der Abstand kleiner oder gleich π(180°), so wird der Zielpunkt der Helix entgegen der Drehrichtung auf den programmierten Zielpunkt verschoben, d.h. die Steigung wird vergrößert.

Ist er größer als π(180°), wird der Zielpunkt der Helix in Drehrichtung auf den programmierten Zielpunkt verschoben, d.h. die Steigung wird verringert.

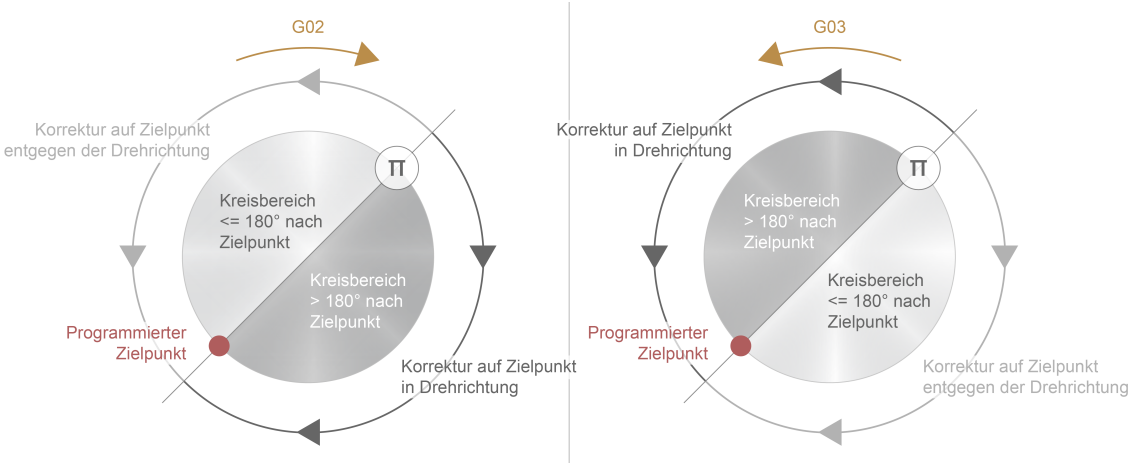


Abb. 9: Korrektur der Helixsteigung in Abhängigkeit von der Drehrichtung

Beispiel

Prinzipielle Korrektur einer Helix im Uhrzeigersinn (G02) (1. Fall)

Der mit Hilfe der programmierten Steigung P_{prog} berechnete Zielpunkt liegt im Bereich von 180° nach dem programmierten Zielpunkt (in Drehrichtung gesehen).

Zur Korrektur wird die Steigung P_{korr} vergrößert.

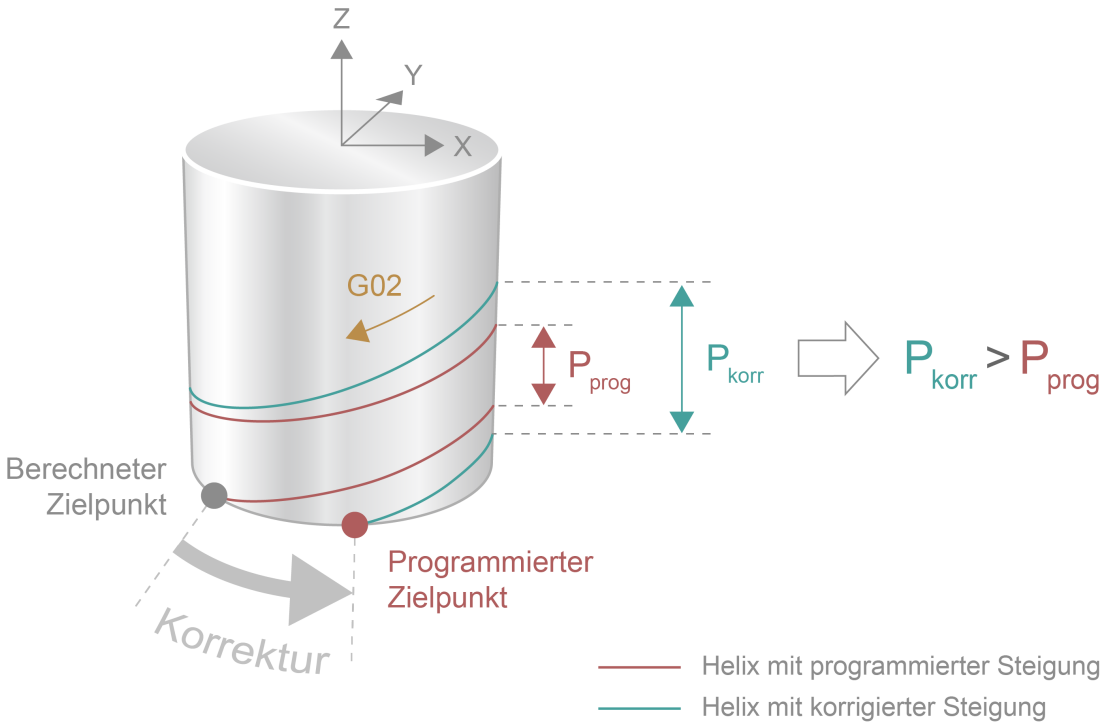


Abb. 10: Korrektur einer Helix im Bereich v. 180° nach dem programmierten Zielpunkt

Beispiel

Prinzipielle Korrektur einer Helix im Uhrzeigersinn (G02) (2. Fall)

Der mit Hilfe der programmierten Steigung P_{prog} berechnete Zielpunkt liegt im Bereich von 180° vor dem programmierten Zielpunkt (in Drehrichtung gesehen).

Zur Korrektur wird die Steigung P_{korr} verringert.

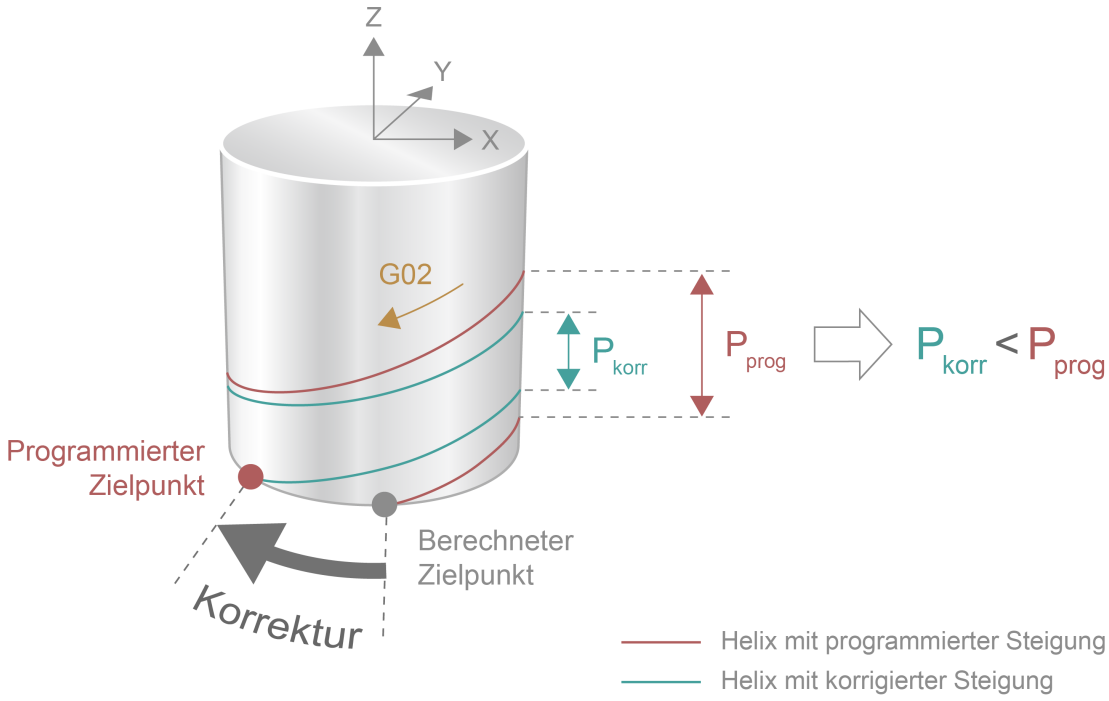


Abb. 11: Korrektur einer Helix im Bereich von 180° vor dem programmierten Zielpunkt

Programmierbeispiel

Helikalinterpolation in der Ebene XY im Uhrzeigersinn

Folgende Helix soll gefahren werden:

Startpunkt a: X-10 Y0 Z0
Zielpunkt b: X0 Y-10 Z-20
Helixmittelpunkt I, J: Nullpunkt
Helixsteigung K: variabel

```
:
N10 G17 G90 X-10 Y0 Z0 F500 G161
N20 G02 X0  Y-10 Z-20  I0   J0   K..
:
```

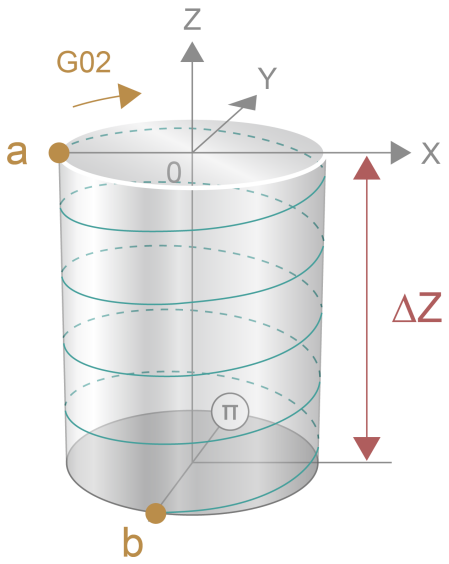


Abb. 12: Helikalinterpolation in der Ebene XY im Uhrzeigersinn

Minimale Umdrehung: ¾ → Steigung K=26,66

Steigung K größer oder gleich 26,66:

Die Helix von a nach b wird generell in ¾ Umdrehungen ausgeführt, da die Korrektur auf die maximal mögliche Steigung K = 26,66 begrenzt wird.

Steigung K kleiner als 26,666:

Programmierte Steigung K (in mm)	Helixumdrehungen von a nach b	Korrigierte Steigung K (in mm)
17,5	¾	26,66
16	1¾	11,4
15	1¾	11,4
12,5	1¾	11,4
10	1¾	11,4
7,5	2¾	7,27
5	3¾	5,33
2,5	7¾	2,58
2	9¾	2,05
1	19¾	1,01

4.1.4.1 Einfache Helikalinterpolation

Bei der vereinfachten Helikalprogrammierung wird keine Steigung, sondern nur ein Zielpunkt definiert. Abhängig vom Zielpunkt ergibt sich eine Helikalbewegung mit maximal einer vollständigen Umdrehung.

```
Syntaxbeispiel für Ebene G17:
```
G02 | **G03 X**.. **Y**.. **Z**.. **I**.. **J**.. | **R**..

G02 | G03	Kreisinterpolation CW / CCW
X.. Y..	Zielpunkt in der Ebene XY in [mm, inch]
Z..	Zielpunkt auf der Helixachse senkrecht zur Ebene XY in [mm, inch]
I.. J..	Lage des Kreismittelpunktes der Interpolation in der Ebene XY (I in X, J in Y) in [mm, inch], entsprechend G161/G162.
R..	Radius des zu interpolierenden Kreises (alternativ zu I,J) in [mm, inch]

```
Syntax entsprechend der angewählten Interpolationsebene:
```

Ebene	Interpolations-art	Zielpunkt in Ebene	Zielpunkt auf Helixachse	Mittelpunkt /Radius
G17	G02/G03	X..Y..	Z..	I..J../R
G18	G02/G03	Z..X..	Y..	K..I../R
G19	G02/G03	Y..Z..	X..	J..K../R

Programmierbeispiel

Helikalinterpolation in der Ebene XY im Gegenuhrzeigersinn

Folgende "Helix" soll gefahren werden:

Startpunkt a: X-10 Y0 Z0

Zielpunkt b: Z20

Helixmittelpunkt I, J: Nullpunkt

```
N10 G17 G90 X-10 Y0 Z0 F500 G161
N20 G03 I0 J0 Z20
```

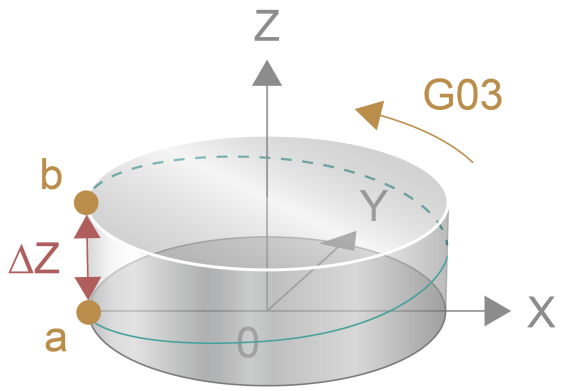


Abb. 13: Helikalinterpolation in der Ebene XY gegen den Uhrzeigersinn

4.1.5 Kreisbogen im Raum (G303)

Versionshinweis

Diese Funktionalität ist verfügbar ab CNC-Version V3.01.3061.0

Mit G303 können Kreisbögen im Raum programmiert werden. Der Kreisbogen wird eindeutig beschrieben über die 3 Punkte:

- Startpunkt
- Zwischenpunkt
- Zielpunkt

Die Programmierung der Koordinaten erfolgt für:

- den Zwischenpunkt mit I, J, K und
- für den Zielpunkt mit X, Y, Z

Startpunkt, Zwischenpunkt und Zielpunkt des Kreisbogens dürfen nicht auf einer Geraden liegen und der Abstand zwischen allen 3 Punkten muss jeweils größer 0 sein.

Die 3 Punkte definieren die Ebene des Kreisbogens. Über die Reihenfolge Startpunkt-Zwischenpunkt-Zielpunkt ist die Verfahrrichtung eindeutig festgelegt.

Die Koordinaten von Zwischenpunkt und Zielpunkt beziehen sich auf die aktuell wirksame Absolut- oder Relativmaßangabe (G90 oder G91). Bei G91 ist die zuletzt angefahrene Position, d.h. der Startpunkt, der Bezugspunkt.

Syntaxbeispiel für G303:

G303 I.. **J**.. **K**.. **X**.. **Y**.. **Z**.. modal

G303	Kreisbogen im Raum
I.. J.. K..	Koordinaten des Zwischenpunktes im Raum in [mm, inch]
X.. Y.. Z..	Koordinaten des Zielpunktes im Raum in [mm, inch]

Einschränkungen:

- Vollkreise können mit der Dreipunkt-Definition nicht programmiert werden.
- Die Kreisbogenfahrt ist nur mit abgewählter Werkzeugradiuskorrektur (G40) zulässig.

Programmierbeispiel

Kreisbogen im Raum (G303)

```
%Circle_G303
;Start bei X0 Y0 Z0
N10 G17 G90 G01 F2000 X0 Y0 Z0
N20 G01 X0 Y0 Z30 ;Kreisstartpunkt (1)
;Fahre Kreisbogen über Zwischenpunkt I,J,K (2) nach Zielpunkt X,Y,Z (3)
N30 G303 I30 J75 K15 X60 Y0 Z0

N100 M30
```

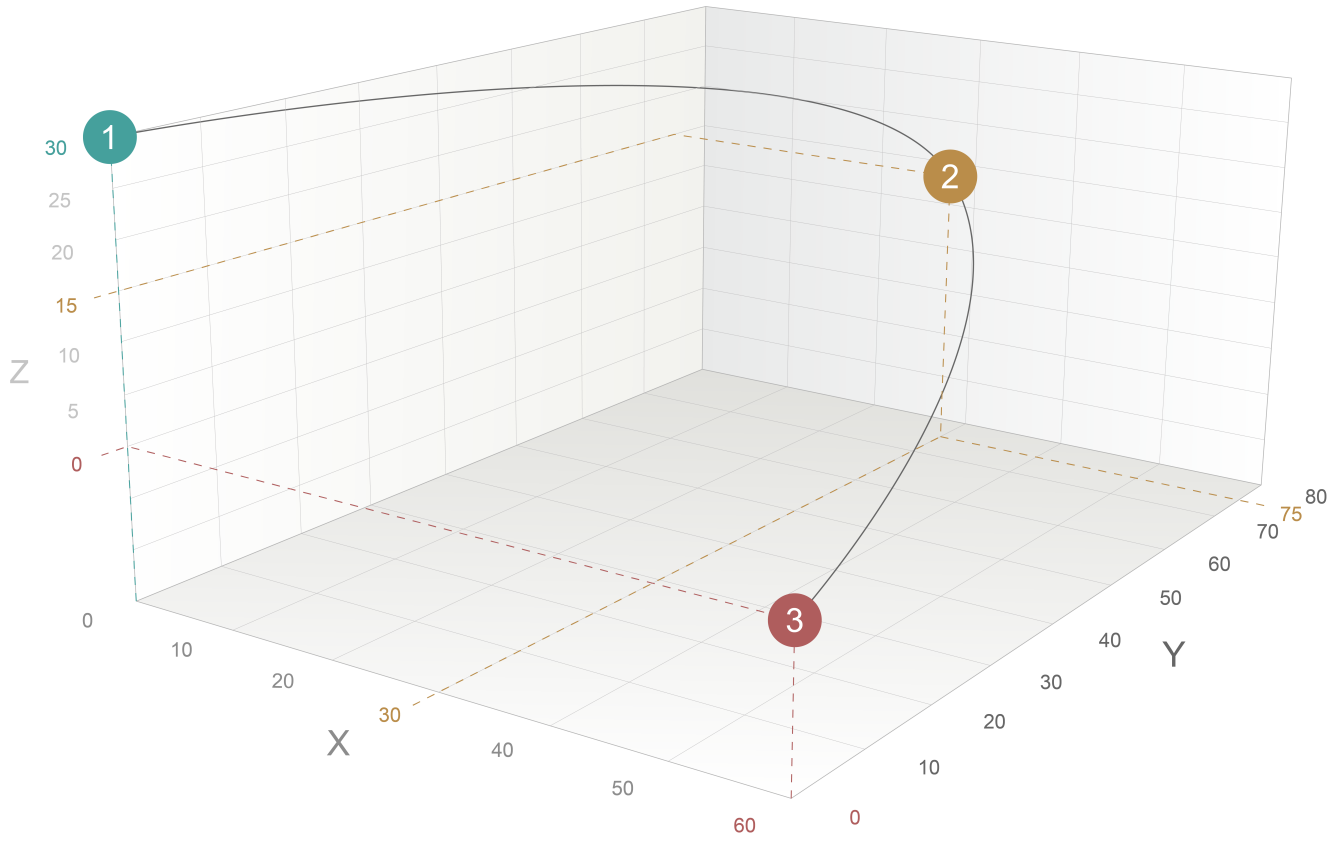


Abb. 14: Kreisbogen im Raum, definiert durch Startpunkt (1), Zwischen- (2) und Zielpunkt (3)

4.1.6 Konturzugprogrammierung (#ANG)

In technischen Zeichnungen werden einfache Konturen (z.B. von Drehteilen) häufig mit Hilfe von Winkeln und einzelnen Positionsangaben beschrieben. Mit der Konturzugprogrammierung kann diese Bemaßung schnell und direkt in ein NC-Programm übernommen werden.

Konturzüge liegen in einer Ebene (G17, G18, G19) und beschreiben in Form von Geraden eine andere Art der Programmierung von Linearsätzen.

Konturzugprogrammierung bei aktiver Zirkularinterpolation führt zur Ausgabe einer Fehlermeldung.

Konturzug, bestehend aus einer Geraden

Ausgehend von einem Startpunkt (SP) beschreibt ein Konturzug den Linearsatz durch:

- Angabe eines Winkels (ANG)
- und **einer** Koordinate (POS) des Zielpunktes (ZP).

Die unbekannte zweite Koordinate des Zielpunktes wird von der Steuerung über den Winkel und die programmierte Koordinate berechnet. Es spielt keine Rolle, welche der beiden Koordinaten der Zielposition angegeben wird. Üblicherweise ist dies abhängig von der Bemaßung in der vorliegenden Zeichnung.

```
Syntaxbeispiel für Ebene G17:
```
#ANG=.. X.. | Y.. modal

#ANG=..	Winkel im Bezug auf die erste Hauptachse der aktiven Ebene in [°]
X..	Koordinate des Zielpunktes in der ersten Hauptachse in [mm, inch]
Y..	Koordinate des Zielpunktes in der zweiten Hauptachse in [mm, inch]

Konturzug mit Koordinate in der ersten Hauptachse

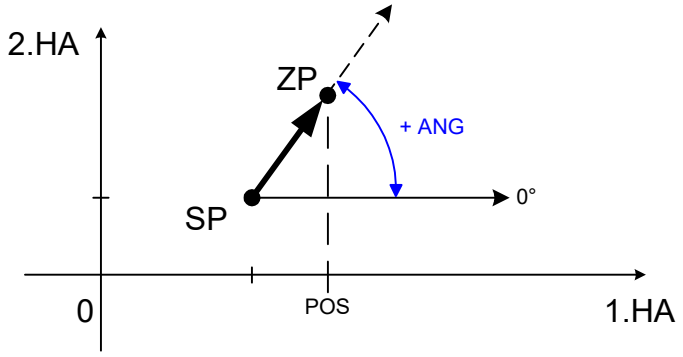


Abb. 15: Konturzug mit Koordinate in der ersten Hauptachse

Programmierbeispiel

Konturzug in G17 mit Zielkoordinate in X

```
N10 G17 G90 G0 X10 Y10          ;Startposition anfahren
N20 G01 F2000 #ANG=60 X20       ;Konturzug auf Zielpunkt: X20, Y27.3205
N30 ...
```

Konturzug mit Koordinate in der zweiten Hauptachse

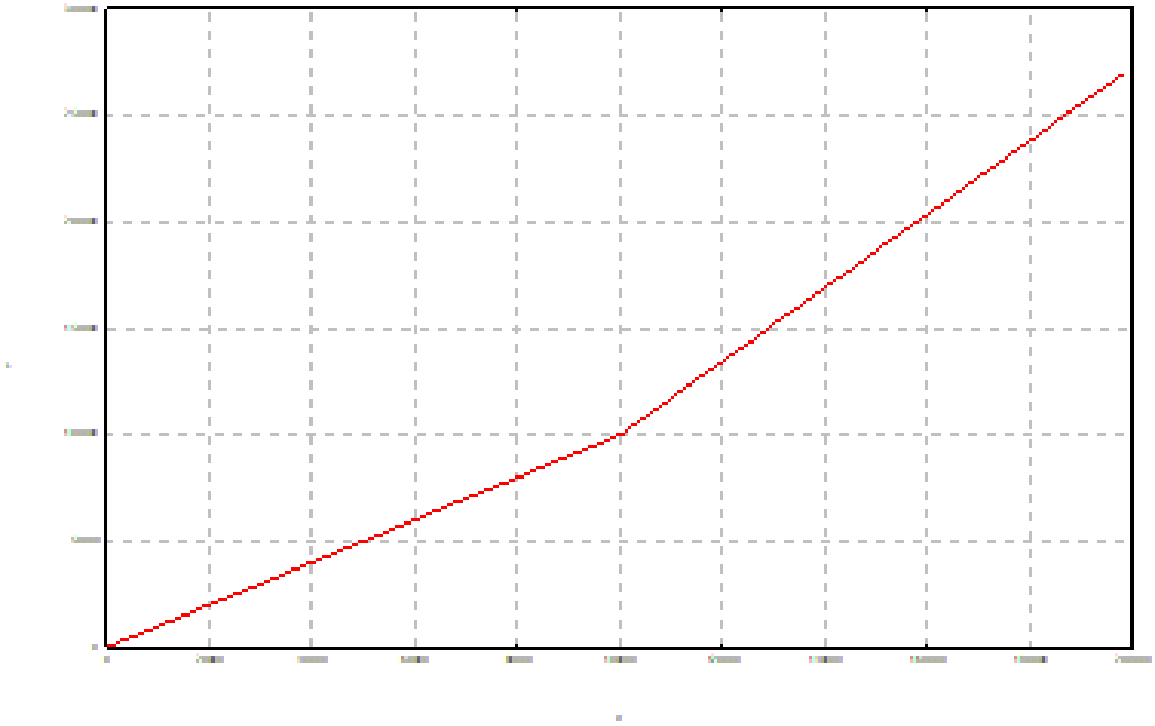
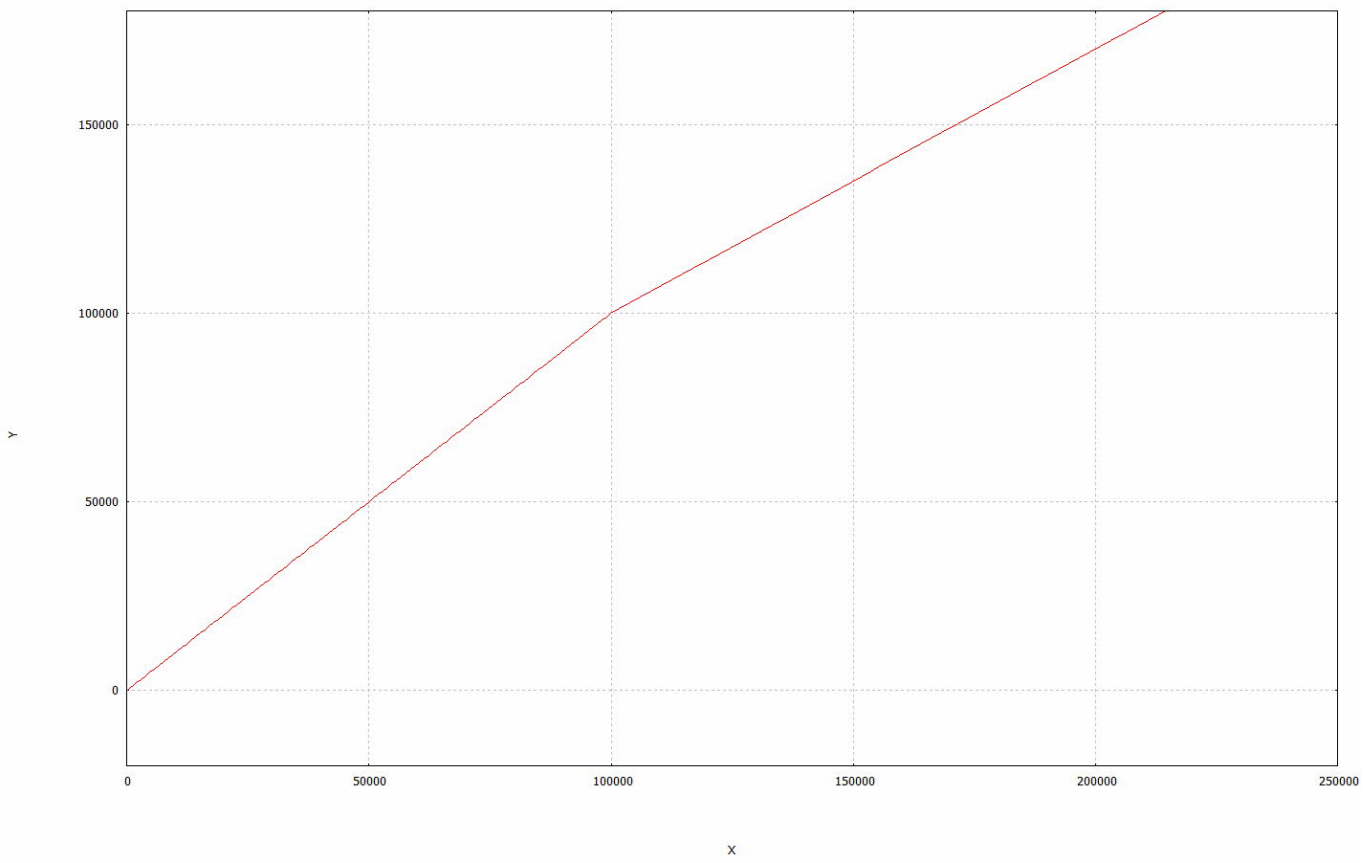


Abb. 16: Konturzug mit Koordinate in der zweiten Hauptachse

Programmierbeispiel

Konturzug in G17 mit Zielkoordinate in Y

```
N10 G17 G90 G0 X10 Y10 ;Startposition anfahren
N20 G01 F2000 #ANG=35 Y20 ;Konturzug auf Zielpunkt: X24.2812, Y20
N30 ...
```



Seitenumbruch

Gültigkeitsprüfung des Zielpunkts:

Bei der Ermittlung des vollständigen Zielpunktes wird geprüft, ob die programmierte Koordinate (POS) des Zielpunktes mit dem angegebenen Winkel erreicht werden kann. Wenn der Zielpunkt mit dem angegebenen Winkel nicht erreicht werden kann, dann wird eine Fehlermeldung ausgegeben.

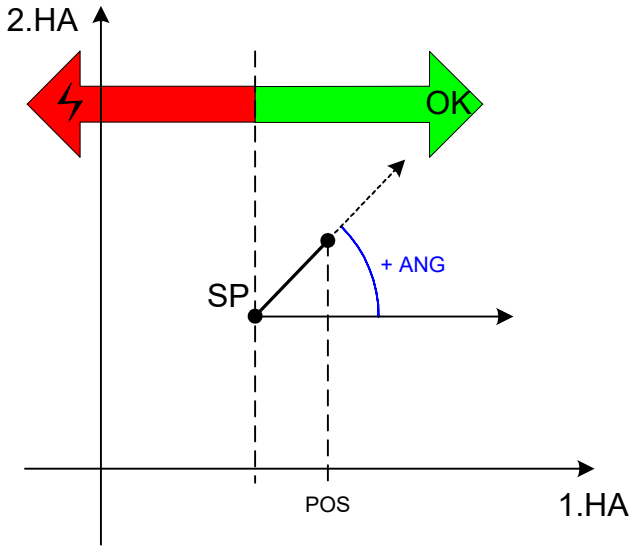


Abb. 17: Gültigkeitsbereich der Zielposition

Konturzug, bestehend aus 2 Geraden

Ein Konturzug, bestehend aus 2 Geraden, kann in verschiedenen Kombinationen von Winkeln und Zielkoordinaten programmiert werden. Die entsprechenden Regeln sind in den nachfolgend aufgeführten zulässigen Fällen dargestellt.

Seitenumbruch

Fall 1: Kombination von zwei Winkeln und zwei Zielkoordinaten

Der Zielpunkt ZP1 der ersten Gerade ergibt sich aus einem Winkel ANG1 und einer Zielkoordinate. Darauf aufbauend ergibt sich der Zielpunkt ZP2 der zweiten Gerade ebenfalls aus einem Winkel ANG2 und einer Zielkoordinate. Die Zielkoordinaten von ZP1 und ZP2 können absolut (G90) oder relativ (G91) programmiert werden.

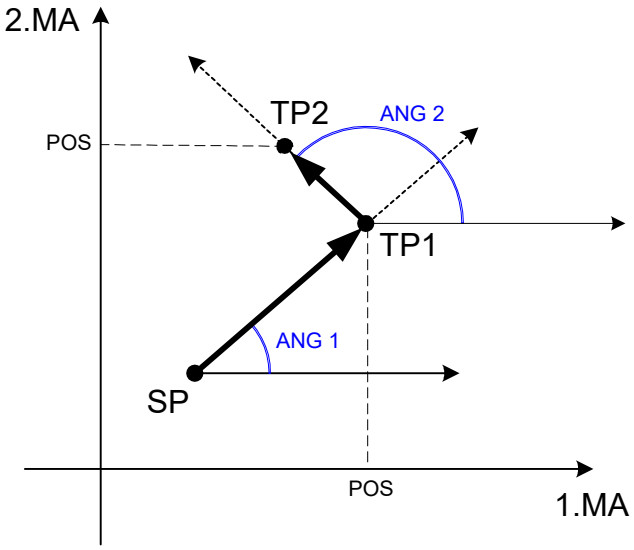
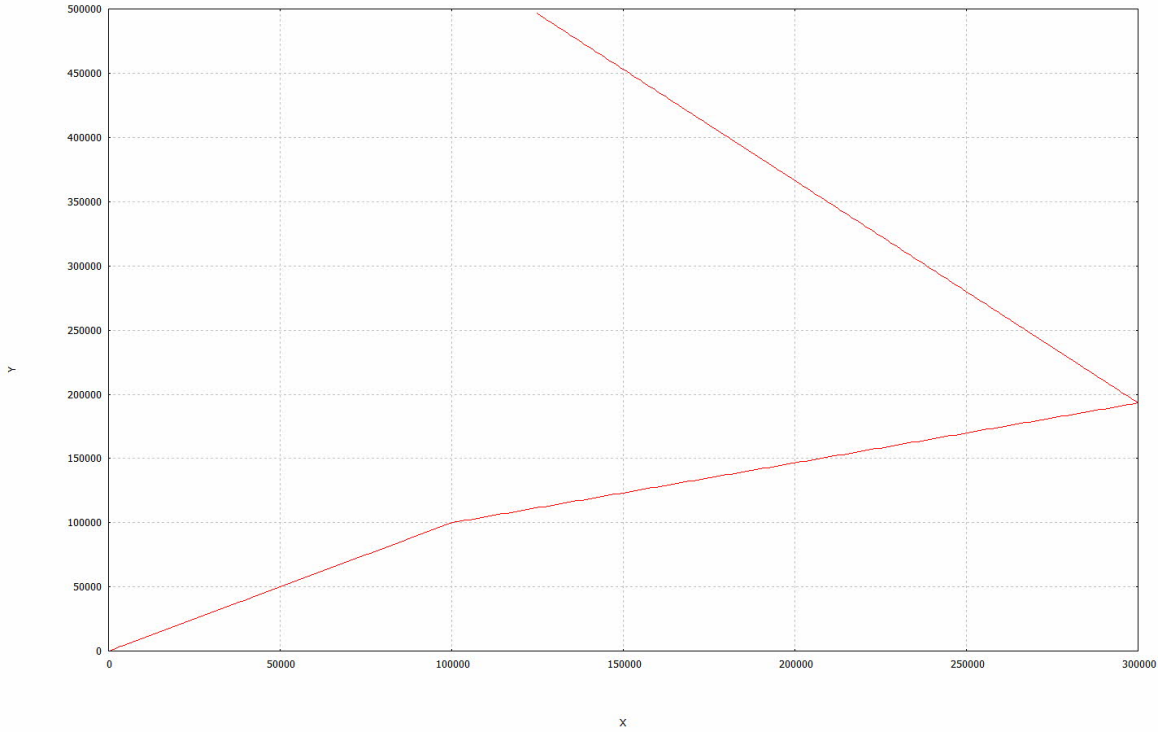


Abb. 18: Konturzug mit zwei Geraden (2 Winkel mit jeweils einer Zielkoordinate)

Programmierbeispiel

Konturzug mit 2 Geraden in G17 und 2 Winkeln mit Zielkoordinaten

```
N10 G17 G90 G0 X10 Y10 ;Startposition anfahren
N20 G01 F2000 #ANG=25 X30 ;Gerade 1, Zielpunkt: X30, Y19.3258
N30 #ANG=120 Y50 ;Gerade 2, Zielpunkt: X12.2904, Y50
N40 ...
```


500000
450000
400000
350000
300000
250000
200000
150000
100000
50000
0
Y
0
50000
100000
150000
200000
250000
300000
X

Fall 2: Kombination von zwei Winkeln und Zielpunkt 2

Für beide Geraden sind die jeweiligen Winkel ANG1 und ANG2 und für die zweite Gerade der Zielpunkt ZP2 vollständig (kartesisch) programmiert. Der Zielpunkt ZP2 muss immer absolut (G90) angegeben werden. Damit ist der Zielpunkt der ersten Gerade ZP1 als Schnittpunkt der Geraden bestimmbar.

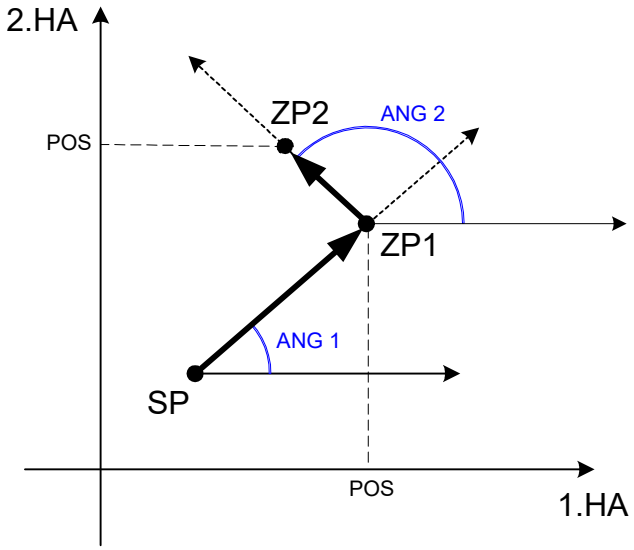
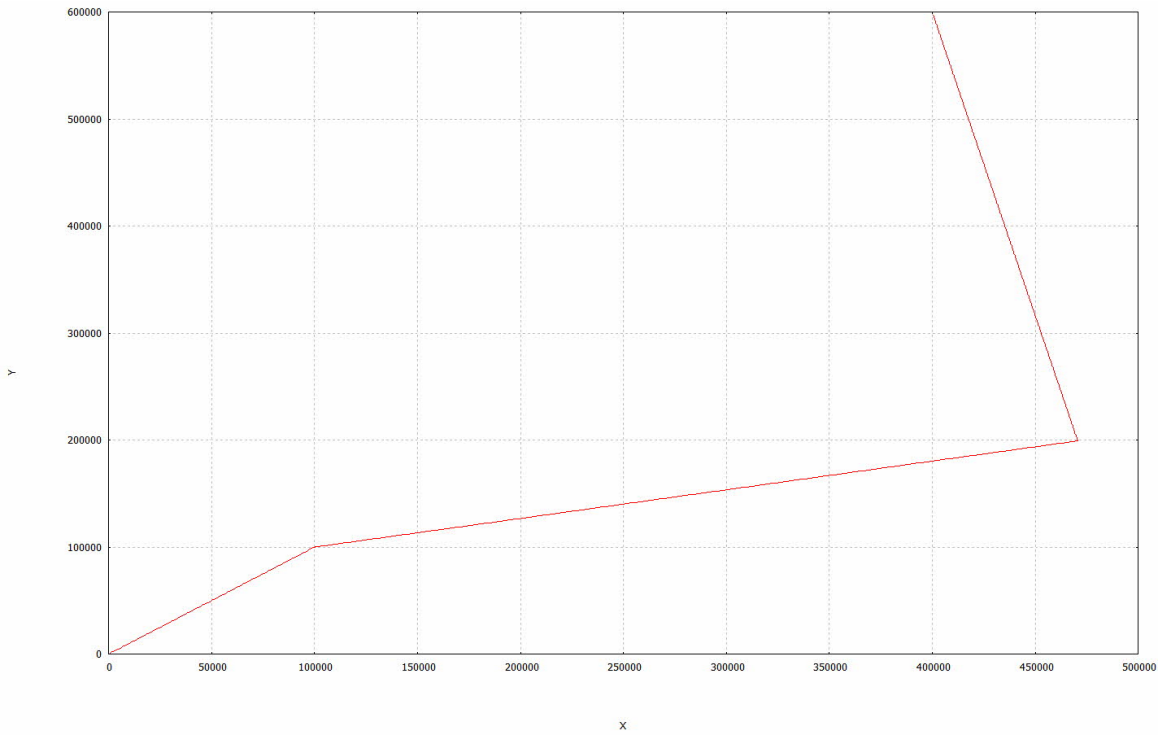


Abb. 19: Konturzug mit 2 Geraden, 2 Winkel, vollständiger Zielpunkt 2

Programmierbeispiel

Konturzug mit 2 Geraden in G17 und 2 Winkeln und vollständigem Zielpunkt 2

```
N10 G17 G90 G0 X10 Y10 ;Startposition anfahren
N20 G01 F2000 #ANG=15 ;Gerade 1
N30 #ANG=100 X40 Y60 ;Gerade 2, Zielpunkt 2
N40 ...
```


600000
500000
400000
300000
200000
100000
0
Y
0
50000
100000
150000
200000
250000
300000
350000
400000
450000
500000
X

Sonderfall 2-1: Kombination von zwei Winkeln und einer Zielkoordinate 2

Für beide Geraden sind die jeweiligen Winkel ANG1 und ANG2 und für die zweite Gerade nur eine Zielkoordinate von ZP2 programmiert. Die andere Koordinate des Zielpunktes ZP2 ergibt sich aus der entsprechenden Komponente des Startpunktes SP. Die Zielkoordinate von ZP2 muss immer absolut (G90) angegeben werden. Damit ist der Zielpunkt der ersten Gerade ZP1 als Schnittpunkt der Geraden bestimmbar.

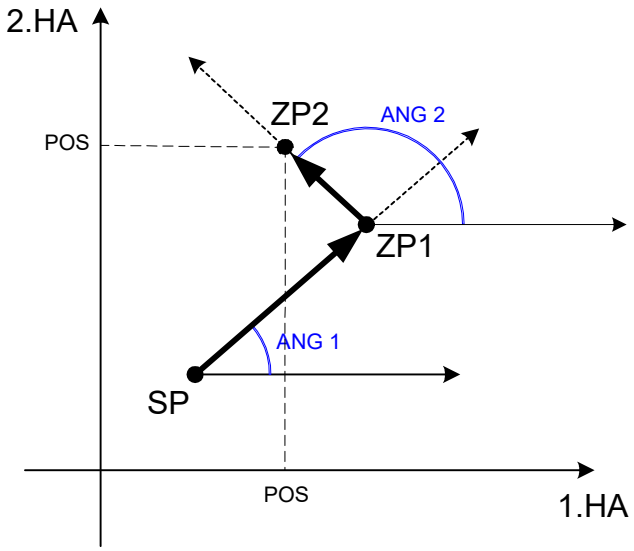
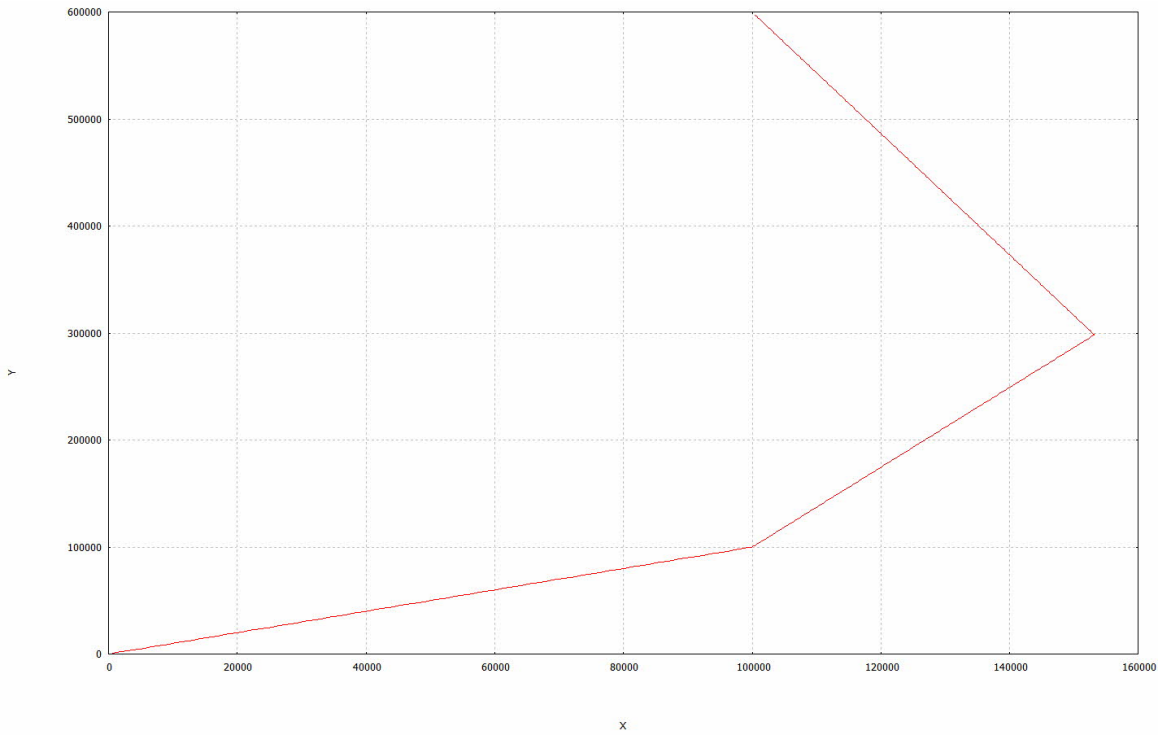


Abb. 20: Konturzug mit 2 Geraden, 2 Winkel, unvollständiger Zielpunkt 2

Programmierbeispiel

Konturzug mit 2 Geraden in G17, 2 Winkeln und unvollständigem Zielpunkt 2

```
N10 G17 G90 G0 X10 Y10 ;Startposition anfahren
N20 G01 F2000 #ANG=75 ;Gerade 1
N30 #ANG=100 Y60 ;Gerade 2, eine Zielkoordinate
N40 ...
```


600000
500000
400000
300000
200000
100000
0
Y
0
20000
40000
60000
80000
100000
120000
140000
160000
X

Sonderfall 2-2: Kombination von zwei Winkeln, keine Zielkoordinaten

Wenn nur Winkel und keine Zielkoordinaten programmiert sind, dann sind die Zielpositionen ZP1 und ZP2 identisch mit der Startposition SP. Es sind dann nur Bewegungen senkrecht zur aktuellen Ebene möglich.

Gültigkeitsprüfung der Zielpunkte:

Es wird geprüft, ob die programmierten Zielpositionen ausgehend von der Startposition mit den programmierten Winkeln erreicht werden können. Die aus den programmierten Winkeln resultierenden Orientierungen definieren den gültigen Bereich für die Zielpunkte.

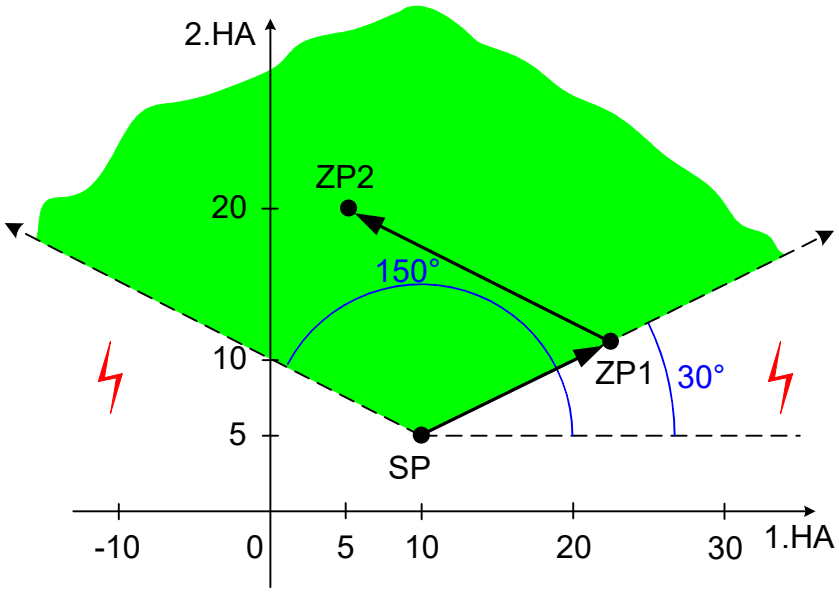


Abb. 21: Gültigkeitsbereich der Zielpositionen bei 2 Geraden

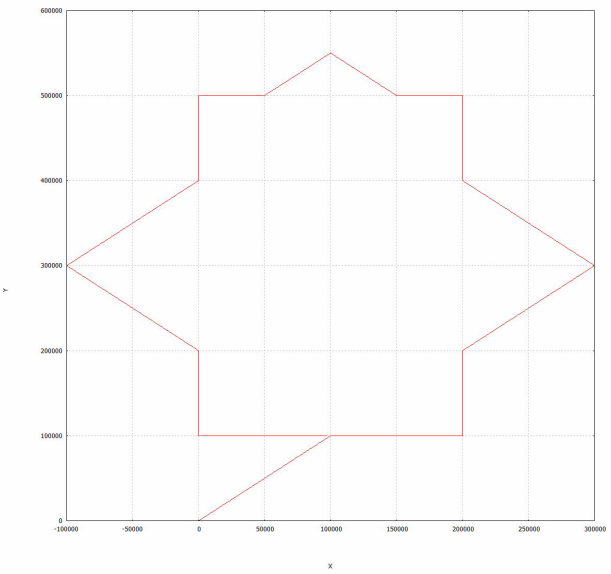
Konturzug, bestehend aus mehreren Geraden

Zur Beschreibung einer Kontur können beliebig viele Geraden miteinander verbunden werden. Hierbei müssen die Zielpunkte der Geraden immer eindeutig geometrisch bestimmbar sein. Die Programmierregeln für einen Konturzug mit 2 Geraden sind dementsprechend auch für verkettete Konturzüge zu beachten.

Programmierbeispiel

Konturzug mit mehreren Geraden in G17

```
N020  G17 G90 G01 F2000
N030  X10 Y10 ; Startposition anfahren
N040  #ANG=0 X20
N050  #ANG=90 Y20
N060  #ANG=45 X30
N070  #ANG=135 X20
N080  #ANG=90 Y50
N090  #ANG=180 X15
N100  #ANG=135 Y55
N110  #ANG=225 Y50
N120  #ANG=180 X0
N130  #ANG=270 Y40
N140  #ANG=225 Y30
N150  #ANG=315 Y20
N160  #ANG=270 Y10
N170  #ANG=0 X10
N180 M30
```



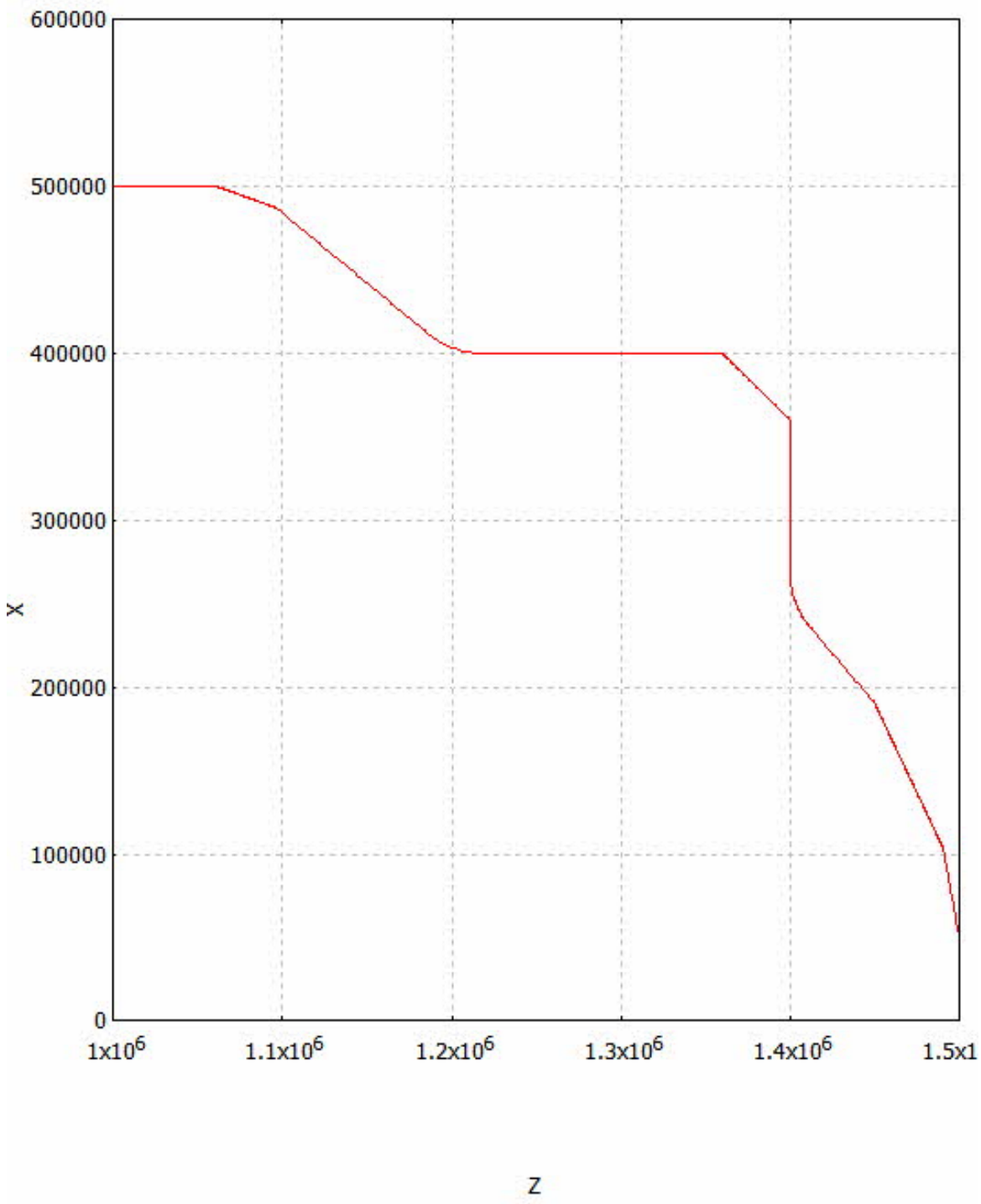
Konturzüge in Kombination mit Fasen und Radien

Konturzüge können mit dem vollen Funktionsumfang der Fasen und Radienprogrammierung kombiniert werden (siehe Fasen und Radien [▶ 156]). Dies ist im folgenden Programmierbeispiel eines Drehteiles dargestellt.

Programmierbeispiel

Konturzug eines Drehteiles mit Fasen und Radien

```
N030 G18 G90 G00 X0 Z150
N040 X5 G01 F2000
N050 #ANG=100 #CHR=5 #FRC=1000
N060 #ANG=130 X25 Z140 #RND=5 #FRC=1500
N070 #ANG=90 X40 #CHR=4 #FRC=1000
N080 Z120 #RND=5 #FRC=1500
N090 #ANG=140 X50 #CHR=2 #FRC=1000
N100 Z100
N110 ...
```


600000
500000
400000
300000
200000
100000
0
X
1x10^6
1.1x10^6
1.2x10^6
1.3x10^6
1.4x10^6
1.5x1
Z

4.1.7 Verweilzeit (G04), (#TIME)

Syntax:

G04 *<1.Hauptachse>..* | *<time>* | *<Hauptspindel>..* nicht modal

G04	Verweilzeit
<1.Hauptachse>..	Die Verweilzeit wird mit dem Namen der 1.Hauptachse in [s] angegeben, oder alternativ...
<time>	...als anschließende direkte oder parametrierte [ab V2.11.2026.02] Angabe der Verweilzeit in [s], oder alternativ…
<Hauptspindel>..	mit dem Namen der Hauptspindel und der Angabe einer Anzahl Umdrehungen [U] [ab V2.11.2023.02].

Verweilzeiten werden z.B. beim Freischneiden oder evtl. bei Maschinenfunktionen benötigt.

Die Verweilzeit darf nur alleine im NC-Satz programmiert werden (Ausnahme: Satz-Nr).

Programmierbeispiel

Verweilzeit (G04)

```
N10 G04 X4.5          (4.5 Sekunden warten)
N20 G04 3.0           (3.0 Sekunden warten)
N30 P1=2
N40 G04 P1            (2.0 Sekunden warten)
N50 V.L.TIME=3.5
N60 G04 V.L.TIME      (3.5 Sekunden warten)
N70 M3 S200
N80 G04 S10           (10 Umdrehungen (3 Sekunden) warten)
```

Eine weitere Möglichkeit zur Angabe der Verweilzeit ist mit der Funktion #TIME möglich.

Syntax:

#TIME *<time>* nicht modal

#TIME	Klartextbefehl Verweilzeit
<time>	Wert der Verweilzeit direkt oder parametriert in [s]

Programmierbeispiel

Verweilzeit (#TIME)

```
N10 #TIME 2.5              (2.5 Sekunden warten)
N20 P1=2
N30 #TIME P1               (2.0 Sekunden warten)
N40 V.L.TIME=3.5
N50 #TIME V.L.TIME         (3.5 Sekunden warten)
```

4.1.8 Programmierbare Referenzpunktfahrt (G74)

Syntax:

G74 Referenzpunkt anfahren nicht modal

G74 erlaubt die NC-programmgesteuerte Durchführung einer Referenzpunktfahrt (RPF), wobei die zu referenzierenden Achsen und die Reihenfolge, in der die Achsen die Referenzpunktfahrt durchführen sollen, angegeben werden müssen. Die mit den Achsnamen programmierten Werte legen die Reihenfolge der Referenzpunktfahrt fest.

Für Achsen mit dem gleichen Wert wird die Referenzpunktfahrt gleichzeitig angestoßen.

Weitergehende Informationen zum Thema Referenzierung können der Funktionsbeschreibung "Referenzpunktfahrt" [FCT-M1] entnommen werden.

Programmierbeispiel

Programmierbare Referenzpunktfahrt (G74)

```
;Sequentielle Beauftragung:
N10 G74 X2 Y3 Z1     ;Reihenfolge der RPF: Z-> X -> Y

;Parallele Beauftragung:
N10 G74 X1 Y1 Z1     ;Reihenfolge der RPF: X,Y,Z gleichzeitig
```

4.1.9 Bezugspunktverschiebung (G92)

Syntax:

G92 Bezugspunktverschiebung nicht modal

G92 erlaubt eine programmierbare Bezugspunktverschiebung in den angegebenen Achsen um einen freiprogrammierbaren Wert in [mm, inch] (additive Nullpunktsverschiebung). Je nach gesetztem G90/G91 wird die aktuell programmierte Bezugspunktverschiebung absolut gesetzt oder zur bisherigen addiert.

Hinweis

"Nicht modal"

…gilt nur für G92; die Bezugspunktverschiebung selbst gilt natürlich bis zur neuen G92-Programmierung

Programmierbeispiel

Bezugspunktverschiebung (G92)

```
N10 G90                 (Absolutmaßangabe)
N20 G92 X10 Z30         (Verschiebt programmierte und absolute)
.                       (Koordinaten um 10 in X, 30 in Z.)
.
.
Nnn G92 X0 Z0           (Rücknahme der Bezugspunktverschiebung)
```

Hinweis

Desweiteren gilt für die Anwahl der Bezugspunktverschiebung im G91-Modus:

Die Programmierung von...

N10 G92 X10 Y20

N20 G0 X0 G91

...darf keine Bewegung der X-Achse hervorrufen (entspricht Bewegung relativ um 0). Somit wirkt sich die Bezugspunktverschiebung für eine Achse erst dann aus, wenn die nächste Verfahrinformation absolut (G90) programmiert wird.

4.1.10 Negative Softwareendschalter setzen (G98)

Syntax:

G98	Negative Softwareendschalter setzen	nicht modal

G98 setzt die negativen Endschalterpositionen in [mm, inch] in allen programmierten Achsen. Je nach gesetztem G90/ G91 geschieht dies absolut oder additiv zur seitherigen Software-Endschalterposition.

Die Positionen für die negativen Endschalter sind in den achsspezifischen Variablen:

V.A.-SWE.X, V.A.-SWE.Y, V.A.-SWE.Z, etc.

abgelegt (siehe auch Achsspezifische Variablen [▶ 591]).

Hinweis

"Nicht modal"

…gilt nur für den Befehl G98, die Softwareendschalter selbst sind modal wirksam.

Nach dem Maschinenhochlauf gilt zunächst der Standardwert des Achsparameters P-AXIS-00177.

Für alle Versionsstände von V2.11.20xx und V2.11.28xx gilt bzgl. der Gültigkeit der Endschalterpositionen:

- Der Grenzwert kann im NC-Programm durch die Programmierung weiter eingeschränkt, aber nicht erhöht werden. D.h. der in der Achsparameterliste angegebene Grenzwert kann hierdurch nicht vergrößert werden.
- Bei statischen Achskonstellationen (ohne Achstausch) bleibt der im NC-Programm geänderte Grenzwert nach Programmende zunächst gültig und ist so auch für das nächste gestartete NC-Programm wirksam! Erst nach einem CNC-Reset und nachfolgendem Programmneustart gilt wieder der konfigurierte Standardwert.

Ab dem Versionsstand V3.1.3077.0 gilt:

- Siehe Beschreibung im Kapitel "Ergänzungen zu G98 und G99"

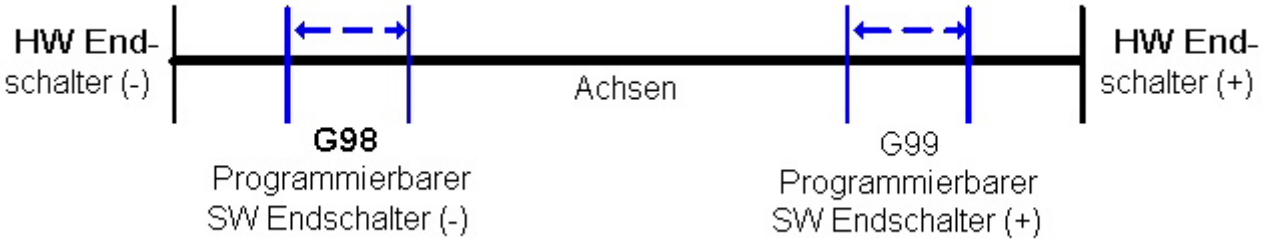
Bei dynamischen Achskonstellationen (mit Achstausch) erfolgt das Rücksetzen auf den ursprünglichen Standardwert mit der Übernahme der Achse in den Kanal.

Die G98-Änderung wirkt für den Achsverfahrbereich der Bahnbewegung, der unabhängigen Achsen und der Einzelachsen. Relative Handbetriebsverfahrbereiche sind hierdurch nicht betroffen; diese werden durch den NC-Befehl #MANUAL LIMITS [...] [▶ 173] beeinflusst.

Programmierbeispiel

Negative Softwareendschalter setzen (G98)

```
N10  G90
...
N100 G98 X-1000 Y-2000          Setzt negativen Softwareendschalter
                                in X auf -1000 und in Y auf -2000
```



4.1.11 Positive Softwareendschalter setzen (G99)

Syntax:

G99	positive Softwareendschalter setzen	nicht modal

G99 setzt die positiven Endschalterpositionen in [mm, inch] in allen programmierten Achsen. Je nach gesetztem G90/G91 geschieht dies absolut oder additiv zur seitherigen Softwareendschalterposition.

Die Positionen für die positiven Endschalter sind in den achsspezifischen Variablen

V.A.+SWE.X, V.A.+SWE.Y, V.A.+SWE.Z, etc.

abgelegt (siehe auch Achsspezifische Variablen [▶ 591]).

Hinweis

"Nicht modal"

…gilt nur für den Befehl G99, die Softwareendschalter selbst sind modal wirksam.

Nach dem Maschinenhochlauf gilt zunächst der Standardwert des Achsparameters P-AXIS-00178.

Für alle Versionsstände von V2.11.20xx und V2.11.28xx gilt bzgl. der Gültigkeit der Endschalterpositionen:

- Der Grenzwert kann im NC-Programm durch die Programmierung weiter eingeschränkt, aber nicht erhöht werden. D.h. der in der Achsparameterliste angegebene Grenzwert kann hierdurch nicht vergrößert werden.
- Bei statischen Achskonstellationen (ohne Achstausch) bleibt der im NC-Programm geänderte Grenzwert nach Programmende zunächst gültig und ist so auch für das nächste gestartete NC-Programm wirksam! Erst nach einem CNC-Reset und nachfolgendem Programmneustart gilt wieder der konfigurierte Standardwert.

Ab dem Versionsstand V3.1.3077.0 gilt:

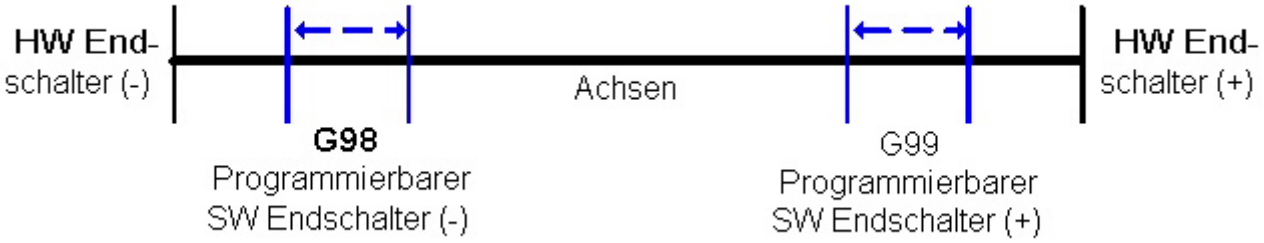
- Siehe Beschreibung im Kapitel "Ergänzungen zu G98 und G99"

Die G99-Änderung wirkt für den Achsverfahrbereich der Bahnbewegung, der unabhängigen Achsen und der Einzelachsen. Relative Handbetriebsverfahrbereiche sind hierdurch nicht betroffen; diese werden durch den NC-Befehl #MANUAL LIMITS [...] [▶ 173] beeinflusst.

Programmierbeispiel

Positive Softwareendschalter setzen (G99)

```
N10 G90
...
N100 G99 X+1000 Y+2000          Setzt positiven Softwareendschalter
                                in X auf +1000 und in Y auf +2000
```



4.1.12 Ergänzungen zu G98 und G99

Ab dem Versionsstand V3.1.3077.0 können die Endschalterpositionen bezogen auf die in der Achsparameterliste angegebenen Grenzwerte mit G98 und G99 auch **erweitert** werden. Dadurch ist es möglich, innerhalb eines NC Programms temporär auf einen erweiterten Bereich und zurück zu wechseln. Der positive Grenzwert muss weiterhin grösser als der negative Grenzwert sein. Beim nächsten gestarteten NC-Programm oder nach einem CNC-Reset gelten wieder die konfigurierten Standardwerte.

Es gilt somit im

- Automatikbetrieb: Mit Programmierung von G98/G99 vor der Verfahrbewegung kann der begrenzte Bereich gegenüber der konfigurierten Einstellung auch vergrößert werden.
- Handbetrieb: Mit Programmierung von G98/G99 vor der Aktivierung des Handbetriebs kann der begrenzte Bereich gegenüber der konfigurierten Einstellung erweitert werden:
  - Relative Offsetlimits P-AXIS-00137 und P-AXIS-00138. Die neue Begrenzung wirkt sofort mit Aktivierung des Handbetriebs.
  - Absolute Offsetlimits P-AXIS-00492 und P-AXIS-00493: Diese Werte sind wirksam, wenn sie in der Achsparameterliste !=0 eingetragen sind. Hierbei können sie maximal auf die konfigurierten Endschalterpositionen (P-AXIS-00177, P-AXIS-00178) eingestellt werden. Die Achse kann somit im Handbetrieb nicht über diese Grenzen hinaus verfahren werden, auch wenn über G98 und G99 die Grenzwerte erweitert wurden. Eine Anpassung dieser absoluten Handbetriebsgrenzen in Richtung der neuen Endschalterpositionen ist aber über CNC-Objekte möglich.

Mit Parameter P-CHAN-00498 (ab V3.1.3080.4) kann das Begrenzungsverhalten eingestellt werden. Bei gesetztem P-CHAN-00498 ist ein Erweitern der Begrenzung nicht zulässig.

Programmierbeispiel

Softwareendschalterbereich mit G98 und G99 vergrößern

```
;Annahme: Softwareendschalter sind auf +- 200 in X, Y konfiguriert
N10 G01 G90 X199
...
N100 G98 X-500 Y-500  ;neg. Softwareendschalter X und Y -> -500
N200 G99 X500 Y500    ;pos. Softwareendschalter X und Y -> +500
N300 G01 X450 Y450    ;Fahren im erweiterten Bereich
...
N400 G01 X100 Y100    ;zurück in eingeschränkten Bereich
N500 G98 X-200 Y-200  ;neg. Softwareendschalter X und Y -> -200
N600 G99 X200 Y200    ;pos. Softwareendschalter X und Y -> 200
...
```

4.1.13 Messfunktionen

Nach dem Hochlauf der Steuerung ist der Messtyp gültig, der in dem Kanalparameter P-CHAN-00057 angegeben ist. Im NC-Programm kann mit #MEAS MODE [▶ 351] bzw. #MEAS [TYPE..] [▶ 352] jederzeit ein neuer Messtyp angewählt werden.

Im NC-Programm liefert die Variable V.G.MEAS_TYPE [▶ 602] den aktuell gültigen Messtyp. Folgende 7 Messtypen stehen zur Verfügung:

Messtyp	Bedeutung
1*	Messfahrt mit mindestens einer Achse, Messvorschub über F-Wort programmierbar.
2*	Messfahrt mit genau einer Achse, Messvorschub wird in P-AXIS-00215 festgelegt. Bei fehlendem Messsignal wird eine Fehlermeldung ausgegeben!
3	Messfahrt mit mindestens einer Achse, Messvorschub über F-Wort programmierbar, wahlweise Weiterfahrt bis zum Zielpunkt.
4	Messfahrt nur mit den maximal 3 Hauptachsen, Messvorschub über F-Wort programmierbar.
5	Unterbrechbare Messfahrt mit mindestens einer Achse, Messvorschub über F-Wort programmierbar.
6	Unterbrechbare Messfahrt mit mindestens einer SERCOS-Achse, Messvorschub über F-Wort programmierbar.
7*	Messfahrt mit Fahren auf Festanschlag mit mindestens einer Achse, Messvorschub über F-Wort programmierbar.

* bei diesen Messtypen ist auch eine Messfahrt mit unabhängigen Achsen [▶ 823] möglich.

Folgende für die Messfahrt relevante Variablen stehen im NC-Programm zur Verfügung (siehe auch Kapitel Achsspezifische Variablen (V.A) [▶ 591]).

Hinweis

Nur die Variablenwerte der bei der Messfahrt programmierten Achsen werden aktualisiert. Alle anderen achsspezifischen Messvariablen beinhalten alte Werte oder 0.

V.A.MERF.<*Achse*>	Messfahrt erfolgt?
V.A.MESS.<*Achse*>	Liefert nach erfolgter Messfahrt den achsspezifischen Messwert in [mm, inch] in dem Koordinatensystem, in dem gemessen wurde. Im Wert sind immer alle Verschiebungen eingerechnet. Bei 2,5D: ACS-Werte bzw. bei CS / TRAFO: PCS-Werte
V.A.MOFFS.<*Achse*>	Messoffset in [mm, inch]
V.A.MEIN.<*Achse*>	Eingerechneter Messoffset in [mm, inch]

Versionshinweis

Ab V2.11.2020.07

V.A.MEAS.ACS.VALUE.<*Achse*>	Messwert im Achskoordinatensystem in [mm, inch] inklusive aller Verschiebungen
V.A.MEAS.PCS.VALUE.<*Achse*>	Messwert im Programmierkoordinatensystem in [mm, inch] ohne Verschiebungen

Einschränkungen:

Eine Messfahrt kann nicht programmiert werden, wenn:

- Polynomüberschleifen (G261, G61) aktiv ist
- Splineinterpolation (AKIMA, B-Spline) aktiv ist
- HSC-Funktionen aktiv sind

4.1.13.1 Messen mit mehreren Achsen (G100) (Typ 1)

Syntax:

G100 <*Achsname*>.. { <*Achsname*>.. } [**F..**] nicht modal

G100	Anwahl Messfahrt
<*Achsname*>..	Zielposition der Messachse in [mm, inch]
F..	Messvorschub in [mm/min, m/min, inch/min]

Es können beliebige Achsen an der Verfahrbewegung des Messsatzes teilnehmen. Alle im Messsatz programmierten Achsen müssen als Messachse gekennzeichnet sein (P-AXIS-00118). Das Messverfahren (Typ 1) muss parametriert sein (P-CHAN-00057).

Beim Messen wird im Messsatz das Eintreffen eines Messsignales registriert. Zwischen dem im NC-Befehl angegebenen Zielpunkt und der Startposition wird linear interpoliert (Wirkung wie bei G01). Die Bahngeschwindigkeit im Messsatz wird im Weiteren als Messvorschub bezeichnet. An einer Messfahrt muss mindestens eine Achse beteiligt sein. Der Messvorschub wird durch das F-Wort angegeben. Der Fahrweg im Messsatz muss größer als 0 sein.

Programmierbeispiel

Messen mit mehreren Achsen (G100) (Typ 1)

Allgemeine Darstellung einer Messfahrt.

```
%G100_Type_1

N10 G90 G00 X0 Y0
N20 G100 X10 Y20 F200 ; X10/Y20 Zielpunkt der Messfahrt
...
```

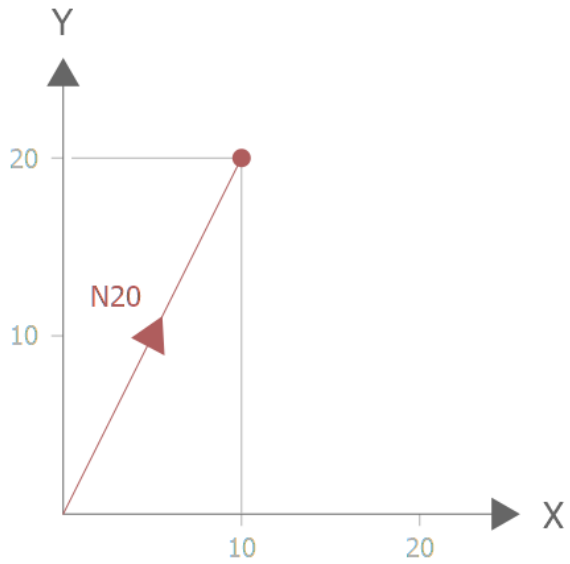


Abb. 22: Programmierte Messfahrt in N20 bei Messfunktion Typ 1

Nach der Registrierung des Messsignals wird angehalten. Der verbleibende Fahrweg des Messsatzes wird nicht mehr ausgegeben.

Programmierbeispiel

Messen mit mehreren Achsen (G100) (Typ 1)

Erfolgreiche Messfahrt mit anschließendem Weiterfahren der programmierten Bahn.

```
%Messfahrt

N10 G90 G00 X0 Y0 Z0
N20 X5
N30 G100 X10 Y10 F500
N40 G01 X7
N50 M30
```

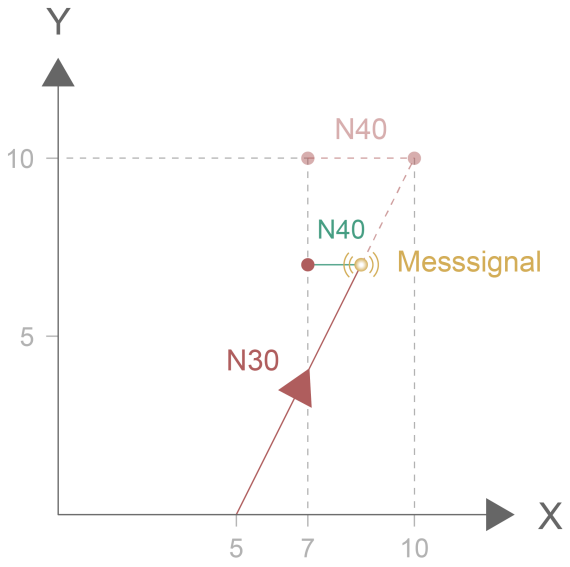


Abb. 23: Programmierte Bahn mit Messfunktion Typ 1

4.1.13.2 Messen mit einer Achse (G100) (Typ 2)

Syntax:

G100 <*Achsname*>.. nicht modal

G100	Anwahl Messfahrt
<*Achsname*>..	Zielposition der Messachse in [mm, inch]

Es darf genau eine Achse an der Verfahrbewegung des Messsatzes teilnehmen. Die im Messsatz programmierte Achse muss als Messachse gekennzeichnet sein (P-AXIS-00118). Das Messverfahren (Typ 2) muss parametriert sein (P-CHAN-00057).

Beim Messen wird im Messsatz das Eintreffen eines Messsignales registriert. Zwischen dem im NC-Befehl angegebenen Zielpunkt und der Startposition wird linear interpoliert (Wirkung wie bei G01). Die Bahngeschwindigkeit im Messsatz wird im Weiteren als Messvorschub bezeichnet. An einer Messfahrt muss genau eine Achse beteiligt sein. Der Messvorschub wird über P-AXIS-00215 angegeben. Der Fahrweg im Messsatz muss größer als Null sein.

Programmierbeispiel

Messen mit einer Achse (G100) (Typ 2)

Allgemeine Darstellung einer Messfahrt

```
%G100_Type_2

N10 G90 G00 X0 Y0
N20 G100 X10 ;X10 Zielpunkt der Messfahrt
...
```

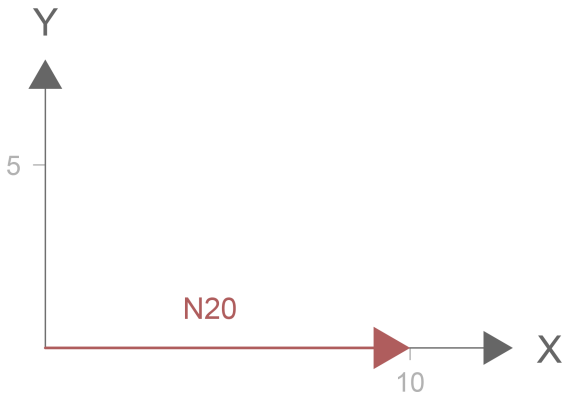


Abb. 24: Programmierung der Messfunktion Typ 2

Nach der Registrierung des Messsignals wird angehalten. Der verbleibende Fahrweg des Messsatzes wird nicht mehr ausgegeben. Kann innerhalb des Messsatzes kein Messsignal aufgenommen werden, wird eine Fehlermeldung ausgegeben.

Programmierbeispiel

Messen mit einer Achse (G100) (Typ 2)

Erfolgreiche Messfahrt mit anschließendem Weiterfahren der programmierten Bahn.

```
%Messfahrt
N10 G90 G00 X0 Y0 Z0
N20 Y5
N30 G100 X10
N40 G01 X7
N50 M30
```

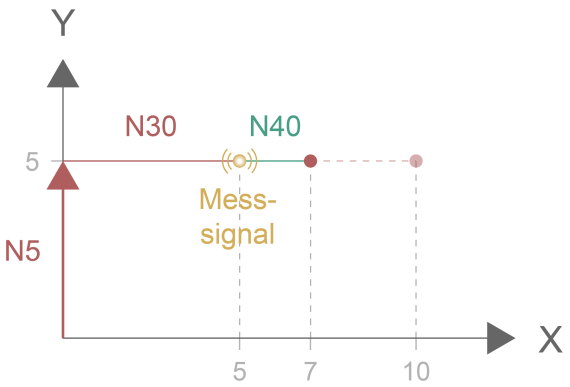


Abb. 25: Programmierte Bahn mit Messfunktion Typ 2

4.1.13.3 Messen mit Fahren bis zum Zielpunkt (G100/G106) (Typ 3)

Syntax:

G100 <*Achsname*>.. { <*Achsname*>.. } [**G106**] [**F..**] nicht modal

G100	Anwahl Messfahrt
<*Achsname*>..	Zielposition der Messachse in [mm, inch]
G106	Fahren bis zum Zielpunkt
F..	Messvorschub in [mm/min, m/min, inch/min]

Es können beliebige Achsen an der Verfahrbewegung des Messsatzes teilnehmen. Alle im Messsatz programmierten Achsen müssen als Messachse gekennzeichnet sein (P-AXIS-00118). Das Messverfahren (Typ 3) muss parametriert sein (P-CHAN-00057).

Beim Messen wird im Messsatz das Eintreffen eines Messsignales registriert. Zwischen dem im NC-Befehl angegebenen Zielpunkt und der Startposition wird linear interpoliert (Wirkung wie bei G01). Die Bahngeschwindigkeit im Messsatz wird im Weiteren als Messvorschub bezeichnet. An einer Messfahrt muss mindestens eine Achse beteiligt sein. Der Messvorschub wird durch das F-Wort angegeben. Der Fahrweg im Messsatz muss größer als 0 sein.

Programmierbeispiel

Messen mit Fahren bis zum Zielpunkt (G100/G106) (Typ 3)

Allgemeine Darstellung einer Messfahrt.

```
%G100_Type_3

N10 G90 G00 X0 Y0
N20 G100 X10 Y20 F200 G106 ;X10/Y20 Zielpunkt der Messfahrt
...
```

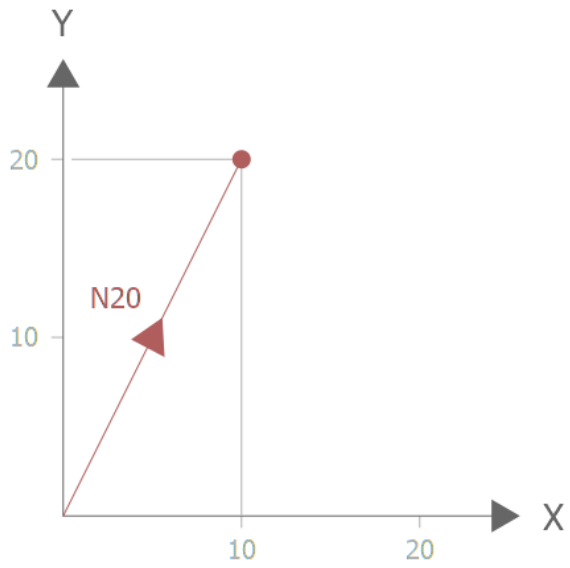


Abb. 26: Programmierte Messfahrt in N20 bei Messfunktion Typ 3

Nach erkanntem Messsignal wird bei programmiertem G106 bis zum Zielpunkt des Messsatzes weitergefahren. Ist G106 nicht programmiert, wird nach dem Messsignal abgebremst und der verbleibende Fahrweg nicht mehr ausgegeben (gleiches Verhalten wie beim Messen Typ1).

Programmierbeispiel

Messen mit Fahren bis zum Zielpunkt (G100/G106) (Typ 3)

Erfolgreiche Messfahrt mit anschließendem Weiterfahren der programmierten Bahn.

```
%Messfahrt

N10 G90 G00 X0 Y0 Z0
N20 G01 X5 F500
N30 G100 G106 X10 Y10 ;Nach Messsignal Fahren bis Zielpunkt
N40 G01 X7
N50 M30
```

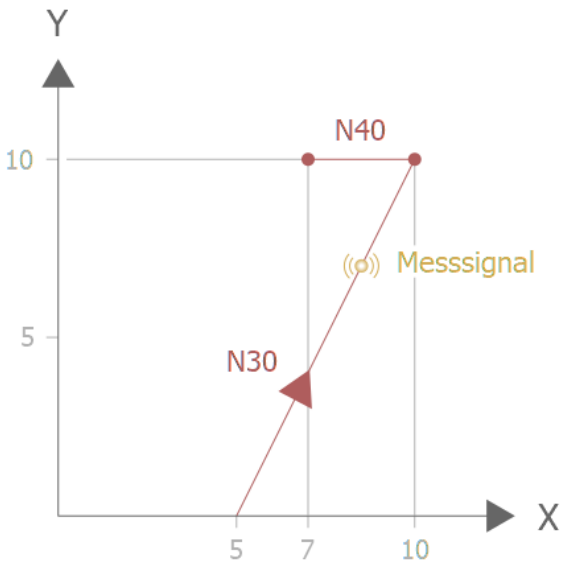


Abb. 27: Programmierte Bahn mit Messfunktion Typ 3

4.1.13.4 Messen mit Hauptachsen (G100) (Typ 4)

Syntax:

G100 <*Achsname*>.. { <*Achsname*>.. } [**F..**] nicht modal

G100	Anwahl Messfahrt
<*Achsname*>..	Zielposition der Messachse in [mm, inch]
F..	Messvorschub in [mm/min, m/min, inch/min]

Es können die drei Hauptachsen an der Verfahrbewegung des Messsatzes teilnehmen. Alle im Messsatz programmierten Achsen müssen als Messachse gekennzeichnet sein (P-AXIS-00118). Das Messverfahren (Typ 4) muss parametriert sein (P-CHAN-00057).

Beim Messen wird im Messsatz das Eintreffen eines Messsignales registriert. Zwischen dem im NC-Befehl angegebenen Zielpunkt und der Startposition wird linear interpoliert (Wirkung wie bei G01). Die Bahngeschwindigkeit im Messsatz wird im Weiteren als Messvorschub bezeichnet. An einer Messfahrt dürfen maximal die drei Hauptachsen beteiligt sein. Der Messvorschub wird durch das F-Wort angegeben. Der Fahrweg im Messsatz muss größer als 0 sein.

Programmierbeispiel

Messen mit Hauptachsen (G100) (Typ 4)

Allgemeine Darstellung einer Messfahrt.

```
%G100_Type_4

N10 G90 G00 X0 Y0
N20 G100 X10 Y20 F200 ;X10/Y20 Zielpunkt der Messfahrt
...
```

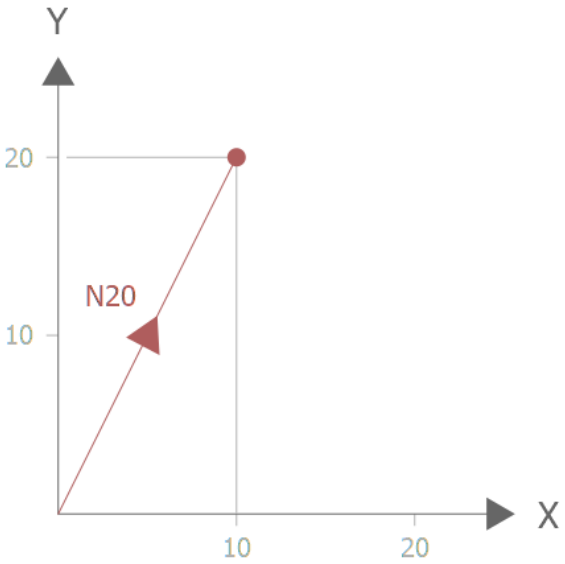


Abb. 28: Programmierte Messfahrt in N20 bei Messfunktion Typ 4

Nach der Registrierung des Messsignals wird angehalten. Der verbleibende Fahrweg des Messsatzes wird nicht mehr ausgegeben.

Programmierbeispiel

Messen mit Hauptachsen (G100) (Typ 4)

Erfolgreiche Messfahrt mit anschließendem Weiterfahren der programmierten Bahn.

```
%Messfahrt

N10 G90 G00 X0 Y0 Z0
N20 X5
N30 G100 X10 Y10 F500
N40 G01 X7
N50 M30
```

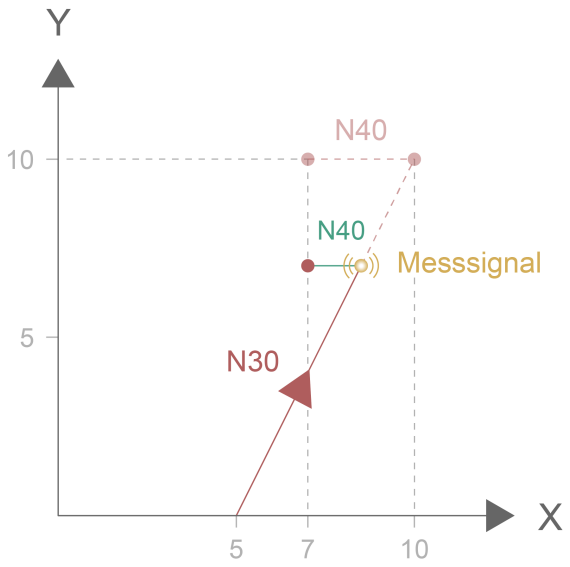


Abb. 29: Programmierte Bahn mit Messfunktion Typ 4

4.1.13.5 Messen mit Unterbrechung und Sprung (G310) (Typ 5,6)

Syntax:

G310 [**G00** | **G01 F..**] <*Achsname*>.. {<*Achsname*>..} [**$GOTO**<*Label*>] nicht modal

G310	Unterbrechbarer Satz
G00 | G01	Unterbrechbare Interpolationsarten
F..	Messvorschub in [mm/min, m/min, inch/min]
<*Achsname*>..	Messachsen mit Zielpositionen in [mm, inch]
$GOTO<*Label*>	Sprungziel nach unterbrochener Messfahrt

Es können beliebige Achsen an der Verfahrbewegung des Messsatzes teilnehmen. Alle im Messsatz programmierten Achsen müssen als Messachse gekennzeichnet sein (P-AXIS-00118). Das Messverfahren (Typ 5,6) muss parametriert sein (P-CHAN-00057).

Dieses Messverfahren bietet die Möglichkeit, eine Bewegung durch ein Messsignal abzubrechen. Die Verfahrbewegung muss dabei explizit im gleichen Satz programmiert werden. Bei Abbruch der Verfahrbewegung durch das Messsignal wird zu einem im G310-Satz angegebenen Sprungziel (Label) verzweigt. Wenn das Messsignal während des Verfahrsatzes nicht auftritt, wird die Ausführung des NC-Programms mit dem nächsten NC-Satz fortgesetzt.

Programmierbeispiel

Messen mit Unterbrechung und Sprung (G310) (Typ 5,6)

```
N10 G00 X0 Y0
N20 G310 G01 F100 X100 Y200 $GOTO[N_LABEL]
;Bei Unterbrechung Sprung zu N_LABEL
N30 G01 X200
N40 $GOTO[ENDE]
N50 [N_LABEL] X0 Y0
N60 [ENDE] M30
```

Nach Unterbrechung der Verfahrbewegung durch ein Messtastersignal werden die Koordinaten des programmierten Zielpunktes durch die Istpositionen aller im Kanal vorhandenen Messachsen ersetzt. Anschließend erfolgt der Sprung zum angegebenen Satz.

Wird kein Signal empfangen, so wird bis zum programmierten Zielpunkt gefahren. Danach erfolgt kein Sprung, sondern der nächste Satz wird ausgeführt.

Wurde kein Sprungziel programmiert, wird generell der nächste Satz ausgeführt.

Die aktuellen Achspositionen nach einer Unterbrechung der Verfahrbewegung können im NC-Programm über die Variable V.A.MEAS... gelesen werden.

Die Programmierung von G310 bei aktiver WRK (G41/G42) führt zum Programmabbruch und zur Ausgabe einer Fehlermeldung.

4.1.13.6 Messen mit Fahren auf Festanschlag (G100) (Typ 7)

Beim Messen mit Fahren auf Festanschlag muss in den beteiligten Antrieben eine Drehmomentbegrenzung aktiviert und eine eventuell vorhandene antriebsseitige Schleppabstandsüberwachung ausgeschaltet sein.

Die Messfahrt wird beendet, sobald in einer der an der Messfahrt beteiligten Achsen der Festanschlag erfasst wurde.

Die für das Messen mit Fahren auf Festanschlag notwendigen Einstellungen sowie ein Programmierbeispiel sind in der Funktionsbeschreibung "Messen (C4)" ([FCT-C4]) dargestellt.

4.1.13.7 Verrechnung des Messoffsets (G101/G102)

Der Messoffset ist in der folgenden Abb. dargestellte Offset zwischen aufgezeichneter Messposition und der programmierten Zielposition. Er berechnet sich wie folgt:

Messoffset = Messposition - Zielposition

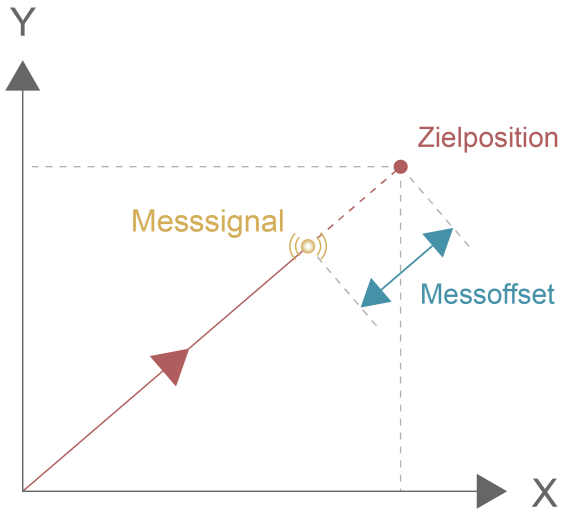


Abb. 30: Messoffset zwischen Messposition und programmierter Zielposition

Syntax:

G101 <*Achsname*>.. { <*Achsname*>.. } nicht modal

G101	Messoffset in Verschiebung einrechnen
<*Achsname*>..	Achsspezifischer Faktor, mit dem der Messoffset eingerechnet wird

Hinweis

Die Angabe mehrerer Achsen ist auch bei Messtyp 2 zulässig, wenn zuvor für jede dieser Achsen jeweils eine eigene Messfahrt durchgeführt wurde.

Für die programmierten Koordinaten wird der aus den Messwerten ermittelte Messoffset als weitere Verschiebung zwischen programmierten und absoluten Koordinaten eingerechnet. Es erfolgt eine Fehlermeldung, wenn zuvor keine Messwerte erfasst wurden. Die Zahl hinter der Achsbezeichnung stellt den Faktor für die Einrechnung dar.

Die Verschiebung durch den Messoffset gilt bis zur Abwahl mit G102.

Programmierbeispiel

Verrechnung des Messoffsets (G101/G102)

Rechnet den Messoffset für X mit Faktor 1 und für Y mit Faktor 7 in die Verschiebung zwischen programmierten und absoluten Koordinaten ein.

```
N10 G101 X1 Y7
```

Syntax:

G102 { *<Achsname>..* } nicht modal

G102	Messoffset aus Verschiebung rückrechnen
<Achsname>..	Achse, bei der der Messoffset rausgerechnet wird. Im Unterschied zu G101 hat der Zahlenwert hinter der Achsangabe keine Bedeutung; aus Syntaxgründen ist er jedoch erforderlich.

Die mit G100 aufgenommenen Messwerte und mit G101 als weitere Verschiebungen eingerechneten Messoffsets werden nach folgender Regel herausgerechnet:

Falls eine oder mehrere Achsen programmiert wurden, werden nur diese Achsen berechnet. Ist keine Achse programmiert worden, so werden alle Verschiebungen herausgerechnet.

Eine Fehlermeldung wird erzeugt, wenn eine Achse programmiert wurde, für die keine Messverschiebung eingerechnet ist.

Es wird immer der Offset herausgerechnet, der mit G101 eingerechnet wurde.

Programmierbeispiel

Verrechnung des Messoffsets (G101/G102)

Herausrechnung einer/aller Achsen.

```
N10 G102 X1  ;Nur Offset der X-Achse wird herausgerechnet
N20 G102     ;Offsets aller Achsen werden herausgerechnet
```

4.1.13.8 Kantenstoßen (G108)

Die Kantenstoßfunktion wird bei der Holzbearbeitung zum exakten Stoßverleimen von Furnierkanten benötigt. Beim Aufleimen einer Furnierleiste kann die Position des Kantenanfangs von Werkstück zu Werkstück ggf. um einige mm verschieden sein. Somit ist das Abschneiden der Furnierleiste an einer Position ungleich einer NC-Satz-Grenze erforderlich. Es handelt sich hierbei um ein zweidimensionales Problem in der XY-Ebene, wobei lineare und zirkulare Bewegungen erlaubt sind.

Der Beginn der Furnierleiste wird messtechnisch durch einen vorauseilenden Taster erfasst (siehe Abbildung unten). Die Position für ein exaktes Abschneiden der Furnierleiste muss aus den Messwerten bestimmt werden.

Es muss folgendes Verhalten erzielt werden:

Nach dem Überfahren des Messtasters wird im Interpolator ein festgelegter Restweg ausgegeben. Anschließend wird gewartet, bis alle Achsen im Regelfenster sind. Daraufhin wird eine M-Funktion vom Synchronisationstyp MNE_SNS (P-CHAN-00027) an die SPS-Schnittstelle weitergegeben, die das Abschneiden der Furnierleiste bewirkt. Nach der Quittierung erfolgt die Fortsetzung der Interpolation bis zum programmierten Zielpunkt. Es darf nur eine MNE_SNS-Funktion aktiv sein, d.h. mehrfache Programmierung in einem Satz ist nicht zulässig. Die MNE_SNS-Funktion kann jedoch mit M-Funktionen anderer Synchronisationarten in einem Satz programmiert werden.

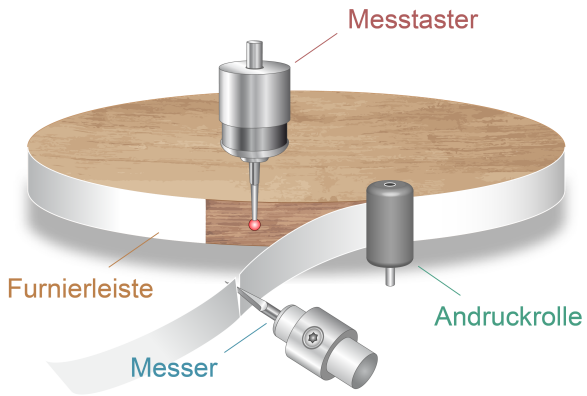


Abb. 31: Aufleimen einer Furnierleiste

Syntax:

G108	Kantenstoßen	modal

Die Anwahl des Kantenstoßen erfolgt über G108 und ist haltend wirksam. Abhängig von der Parametrierung im Kanalparametersatz P-CHAN-00029 sind die im Folgenden beschriebenen 2 Verfahren möglich.

Hinweis

Für den Sonderfall, dass nur eine CNC Achse bewegt wird, ist G108 auch ohne Hauptachsbewegung zulässig. In diesem Fall darf in einer der Mitschleppachsen gemessen werden.

4.1.13.8.1 Stoßverleimen in einem Bewegungssatz (Verfahren 1)

Die Anwahl des Messvorgangs über G108 ist haltend wirksam bis zur Programmierung einer M-Funktion vom Synchronisationstyp MNE_SNS (P-CHAN-00027). Bei dieser M-Funktion wird die Messfahrt gestartet. Das Messsignal muss in dem Verfahrsatz erfolgen, der zusammen mit dieser M-Funktion programmiert wurde.

Nach erkanntem Messsignal wird um den Restweg P-CHAN-00030 weitergefahren und G108 implizit abgewählt.

Programmierbeispiel

Stoßverleimen in einem Bewegungssatz (Verfahren 1)

```
N05 X0 Y0
N10 G108                        (Kantenstoßen einschalten)
N20 G01 X90 Y90 F20
N30 M97 G01 X150 Y150 F8        (M97 ist vom Typ MNE_SNS, Messfahrt starten)
                                (Nach Messsignal um den Restweg weiterfah-
ren)
M30
```

Achtung

Wird während der Messfahrt kein Messereignis registriert, erfolgt am Satzende die Ausgabe einer Fehlermeldung.

4.1.13.8.2 Stoßverleimen über mehrere Bewegungssätze (G107) (Verfahren 2)

Oft liegt das Problem vor, dass eine Kontur über kurze NC-Sätze (z.B. von einem CAD-System) beschrieben wird. Soll an solchen Konturen mit Kantenstoß gefahren werden, so ergibt sich das Problem, den Stoß genau auf einem Konturelement zu treffen.

Die Aktivierung der Kantenstoßfunktion erfolgt ebenfalls über G108 [▶ 110] und ist haltend wirksam.

Bei entsprechender Parametrierung im Kanalparametersatz (P-CHAN-00029) ermöglicht es die erweiterte Funktion, den Kantenstoß erst n Sätze nach der M-Funktion durchzuführen. Die M-Funktion wird ebenfalls sofort ausgegeben, das Triggerereignis wird jedoch erst zu einem späteren Zeitpunkt gegeben. Nach dem Messsignal wird der Restweg über weitere nachfolgende m Sätze abgefahren. Der Zeitpunkt, bis zu dem spätestens die Messfahrt abgeschlossen sein muss, wird explizit durch den NC-Befehl G107 programmiert.

Syntax:

G107	Abwahl satzübergreifendes Kantenstoßen	modal

Programmierbeispiel

Stoßverleimen über mehrere Bewegungssätze (Verfahren 2)

```
N05 X0 Y0
N10 G108                    (Kantenstoßen einschalten)
N20 G01 X90 Y90 F20
N30 M97 G01 X100 Y100 F8    (M97 ist vom Typ MNE_SNS,Messfahrt starten)
N40 X110 Y110
N50 X120 Y120
N60 X130 Y130
N70 X140 Y140
N80 X150 Y150               (<- letzter Messfahrtsatz!)
N90 G107                    (Ende Messen Kantenstoßen)
N80 G00 X200 Y200
M30
```

Hinweis

Der Messvorgang ist auch innerhalb eines Bewegungssatzes möglich. Jedoch bietet Verfahren 1 für diesen Fall eine vereinfachte Programmierung.

Achtung

Wird während der Messfahrt kein Messereignis registriert, erfolgt bei G107 die Ausgabe einer Fehlermeldung.

4.1.13.8.3 Programmierung des Restwegs

V.G.RW Restweg beim Kantenstoßen Lese- und Schreibzugriff

Mit der achsgruppenspezifischen Variable "V.G.RW" kann der Restweg für das Kantenstoßen in [mm, inch] neu festgelegt werden. Der Restweg ist der noch zu verfahrende Weg nach dem Eintreffen des Messsignals. Im Grundzustand wird das entsprechende Arbeitsdatum für den Restweg aus dem Kanalparametersatz (P-CHAN-00030) übernommen.

Programmierbeispiel

Programmierung des Restwegs

```
N010 G91 G00 X0 Y0 Z10       (Linearinterpolation)
N020 V.G.RW = 5              (Festlegung eines neuen Restwegs)
N030 G108                    (Kantenstoßen einschalten)
N040 G01 X10 Y10 F300        (Linearinterpolation)
N050 M97 G01 X30 Y10 F200    (M97 ist vom Typ MNE_SNS,)
                             (Messfahrt mit Linearinterpolation,)
                             (Nach Messsignal um den Restweg weiterfah-
ren)
```

Achtung

Die M-Funktion für das Kantenstoßen muss als Typ MNE_SNS (P-CHAN-00027) festgelegt sein.

4.2 Beschleunigungsbestimmung/Verzögerung (G08/G09/G900/G901)

Hinweis

Die Beschleunigungsbestimmungen sind in Verbindung mit den Slopetypen STEP, TRAPEZ und SIN2 wirksam. Bei Slopetyp HSC ist die Bestimmung wirkungslos.

Syntax:

G08	Beschleunigung am Satzanfang	modal, Grundzustand
G09	Verzögerung am Satzende	nicht modal, Wirkung abhängig von G901/G900
G901	Verzögerung nach Satzende, kann durch G09 satzweise aufgehoben werden	modal
G900	Verzögerung auf das Satzende unabhängig von G09 modal aktiv	modal

Werden zwei aufeinanderfolgende NC-Sätze mit unterschiedlichen Vorschubgeschwindigkeiten programmiert, dann erfolgt eine "weiche" Anpassung an der Satzgrenze. Eine Beschleunigung findet entsprechend G08 ausschließlich am Satzanfang statt. Mit G09 wird angegeben, dass eine Verzögerung auf den Vorschub des nächsten Satzes bereits am Ende des aktuellen Satzes erfolgt.

G08 und G09 sind die Grundeinstellungen beim Programmstart. Um die Grundeinstellung auf Verzögerung erst nach Satzende festzulegen, gibt es die G-Funktion G901. Die inverse G-Funktion G900 dient zum Zurückschalten und entspricht der Grundeinstellung des Kanals.

Achtung

Beim Übergang von G00 nach G01, G02 oder G03 wirkt grundsätzlich G09, d.h. Verzögerung am Satzende auf die Geschwindigkeit des Folgesatzes.

Wird bei aktivem G901 bereits am Bahngeschwindigkeitsgrenzwert gefahren, so wird die Geschwindigkeit des Folgesatzes bereits an der Satzgrenze erreicht, d.h. G901 wirkt dann nicht.

Programmierbeispiel

Beschleunigungsbestimmung (G08/G09/G900/G901)

Beschleunigung am Satzübergang im Grundzustand (entspricht G08).

```
N10 G01 X500 F400
N20     X900 F1000
```

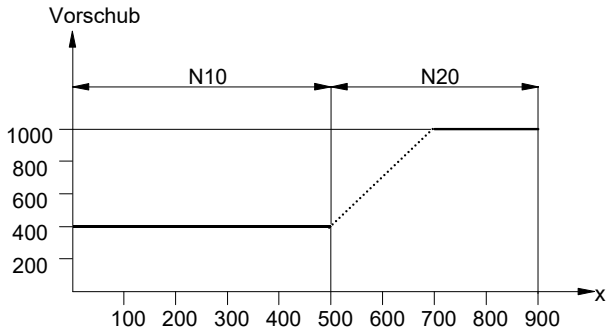


Abb. 32: Beschleunigung Satzübergang im Grundzustand (entspr. G08)

Programmierbeispiel

Beschleunigungsbestimmung (G900/G901)

Verzögerung am Satzübergang mit G901 und G900.

```
N10 G01 G901 X500 F1000
N20          X900 F400
```

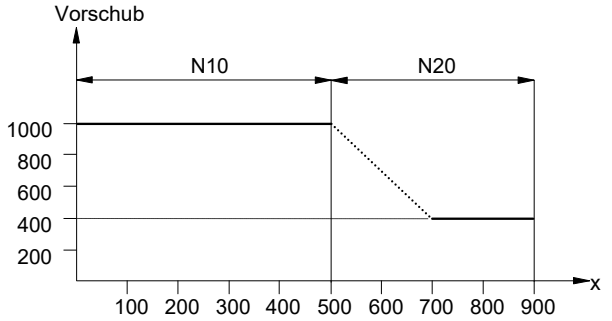


Abb. 33: Verzögerung am Satzübergang mit G901 und G900

```
N10 G01 G900 X500 F1000
N20          X900 F400
```

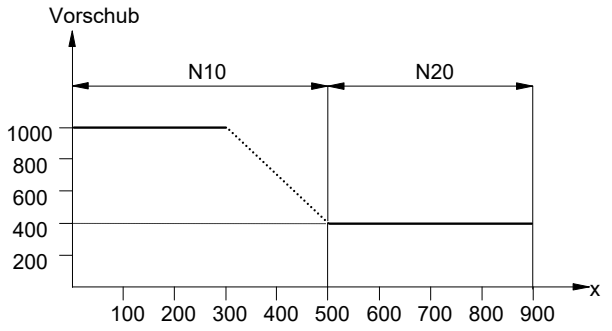


Abb. 34: Verzögerung am Satzübergang mit G901 und G900

Programmierbeispiel

Beschleunigungsbestimmung (G900/G901)

Kombination von G09 mit G901 und G900.

```
N10 G01 G901 X200 F2000
N20     G09  X400 F1600
N30          X600 F1200
N40     G900 X800 F800
N50          X1000 F400
:
```

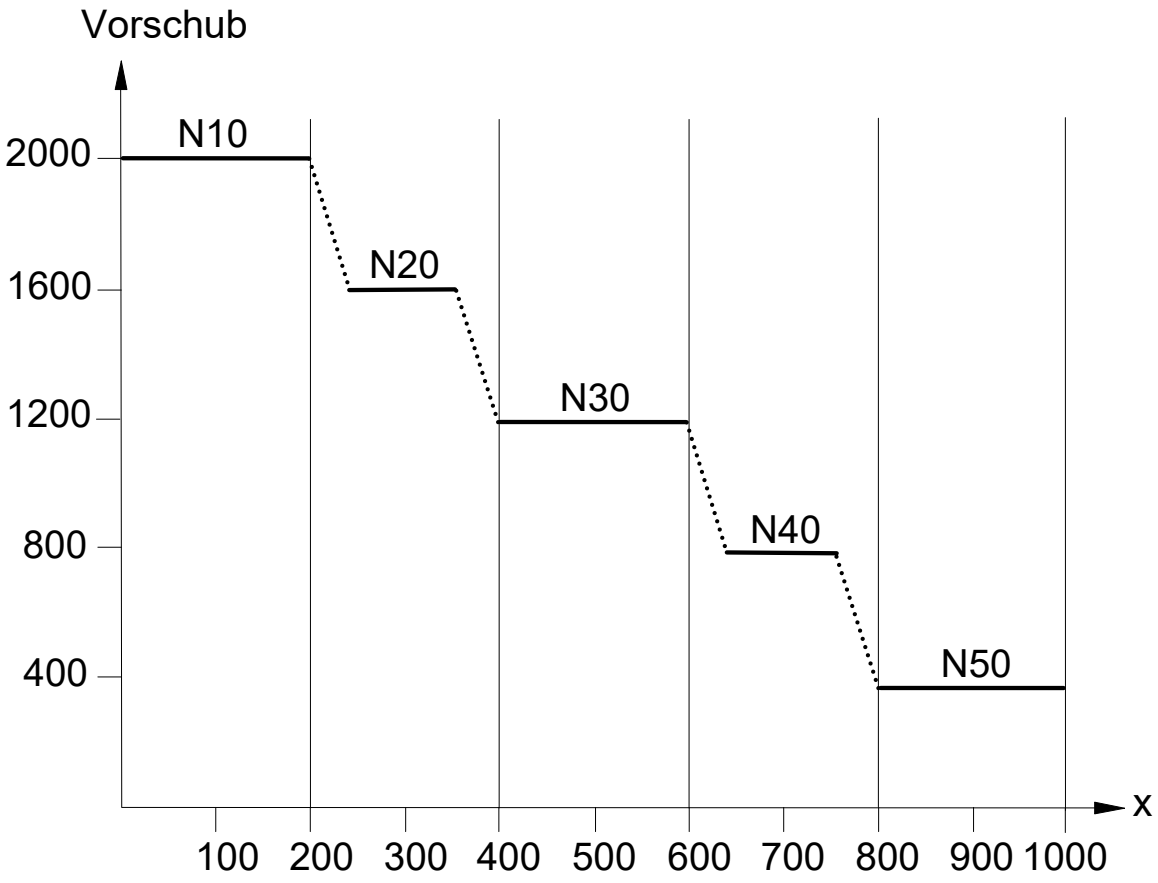
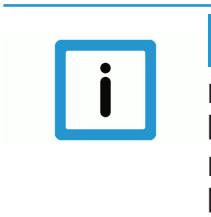


Abb. 35: Kombination von G09 mit G901 und G900

4.3 Weg-/zeitbezogene Vorschubinterpolation (G193/G293)



Hinweis

Die wegbezogene Vorschubinterpolation ist nur in Verbindung mit dem linearen Slope (#SLOPE [TYPE=STEP]) oder dem HSC-Slope (#SLOPE [TYPE=HSC]) wirksam!

Die zeitbezogene Vorschubinterpolation ist nur in Verbindung mit dem linearen Slope (#SLOPE [TYPE=STEP]) wirksam!

Syntax:

G193	Wegbezogene Vorschubinterpolation	nicht modal
G293	Zeitbezogene Vorschubinterpolation	nicht modal

Bei angewähltem G193/G293 wird die Vorschubgeschwindigkeit zwischen Ausgangs- und programmierter Endgeschwindigkeit linear interpoliert.

Hinweis

Die Programmierung von G193 mit G293, G08 oder G09 im selben NC-Satz ist nicht erlaubt.

Programmierbeispiel

Wegbezogene Vorschubinterpolation (G193)

```
N10 G01   X500 F1000
N20 G193  X900 F400
N30 X1000 F400
```

Am Satzübergang N10 / N20 wird die wegbezogene Vorschubinterpolation eingeschaltet und linear über den in N20 programmierten Weg auf die Endgeschwindigkeit verzögert.

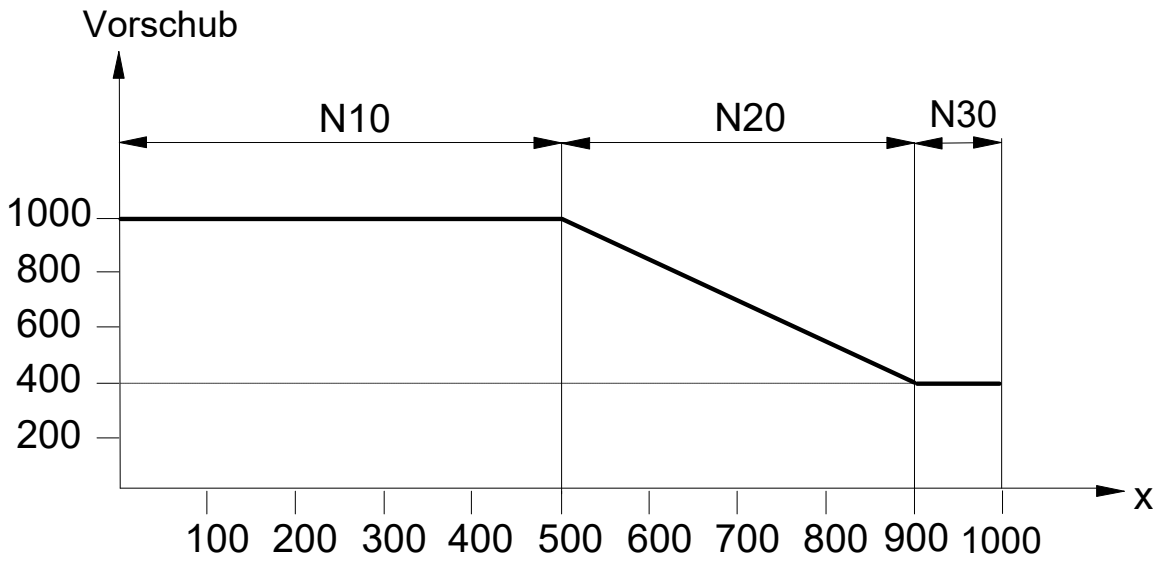


Abb. 36: Wegbezogene Vorschubinterpolation mit G193

Overrideänderungen werden unter Berücksichtigung der zulässigen Achsdynamik überlagert.

Programmierbeispiel

Zeitbezogene Vorschubinterpolation (G293)

```
N10 G01   X500  F1000
N20 G293  X900  F400
N30       X1000 F400
```

Am Satzübergang N10 / N20 wird die zeitbezogene Vorschubinterpolation eingeschaltet und linear über die Zeit auf die Endgeschwindigkeit verzögert.

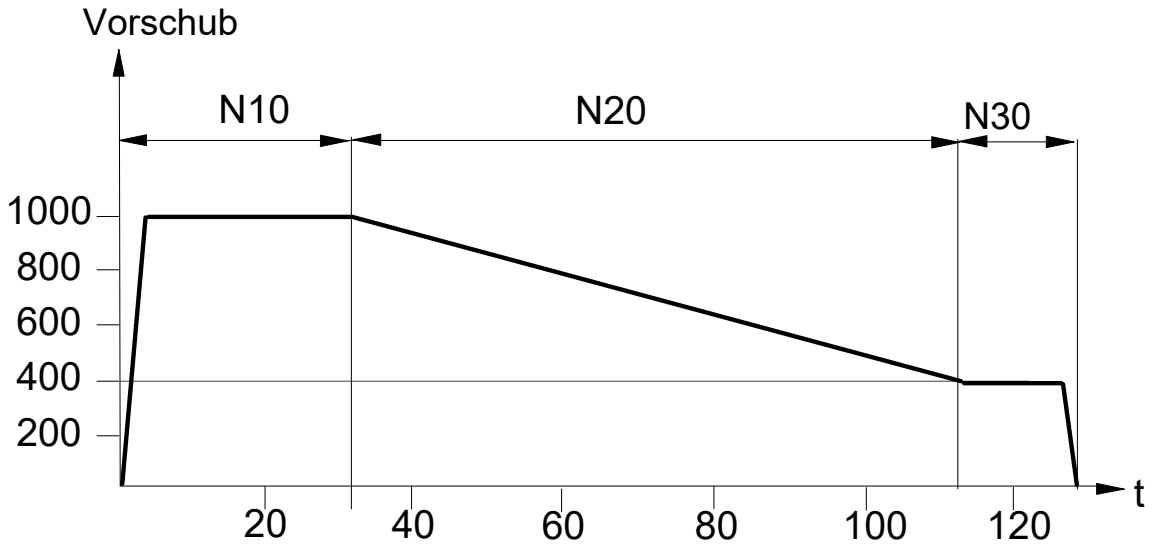


Abb. 37: Zeitbezogene Vorschubinterpolation mit G293

4.4 Ebenenauswahl (G17/G18/G19)

Syntax:

G17	X-Y – Ebene	modal, Grundzustand
G18	Z-X – Ebene	modal
G19	Y-Z – Ebene	modal

Durch Programmierung von G17, G18 oder G19 wird die Ebene festgelegt, in der die Werkzeugradiuskorrektur und die Kreisinterpolation (siehe Kreisinterpolation (G02/G03) [▶ 56]) wirksam sein soll.

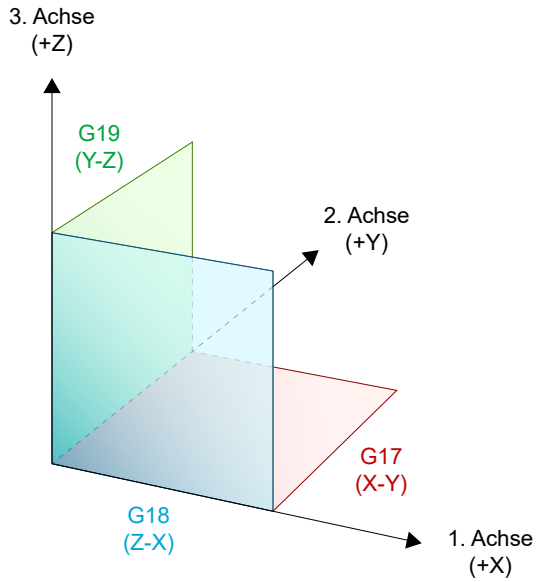


Abb. 38: Darstellung der Ebenenauswahl (G17/G18/G19)

Werkzeugkorrektur:

Es ist zu beachten, dass die eventuell angewählte WRK immer auf die ersten beiden Hauptachsen wirkt.

Kreisinterpolation:

Nach einer Ebenenumschaltung mit G17, G18 oder G19 gilt wieder die Zuordnung nach DIN: X, Y, Z werden I, J, K zugeordnet. Die Syntax entsprechend der angewählten Interpolationsebene zeigt folgende Tabelle:

Ebene	Interpolationsart	Zielpunkt in Ebene	Mittelpunkt /Radius
G17	G02/G03	X..Y..	I..J../R
G18	G02/G03	Z..X..	K..I../R
G19	G02/G03	Y..Z..	J..K../R

4.5 Spiegeln in der Ebene (G21/G22/G23/G20)

Für die Spiegelung in der Ebene wird der Begriff „mentale Koordinaten“ verwendet. Wird eine Spiegelung durchgeführt, so sind die mentalen Koordinaten (x_m) die im NC-Programm eingetragenen Werte. Die realen Koordinaten (x_r) hingegen sind gespiegelt, werden also tatsächlich abgefahren. Die Spiegelfunktionen wirken immer auf die erste und zweite Hauptachse der aktuellen Ebene (G17, G18, G19). Die dritte Hauptachse der aktuellen Ebene wird nicht gespiegelt.

Als Spiegelfunktionen sind vorhanden (Bildbeispiel für G17, X-Y Ebene):

G21 Spiegeln der programmierten Wege in der X-Achse — modal

$x_m = -x_r$

$y_m = y_r$

G22 Spiegeln der programmierten Wege in der Y-Achse — modal

$x_m = -x_r$

$y_m = y_r$

G23 Überlagerung von G21 und G22 — modal

$x_m = -x_r$

$y_m = y_r$

G20 Abwahl der Spiegelung — modal

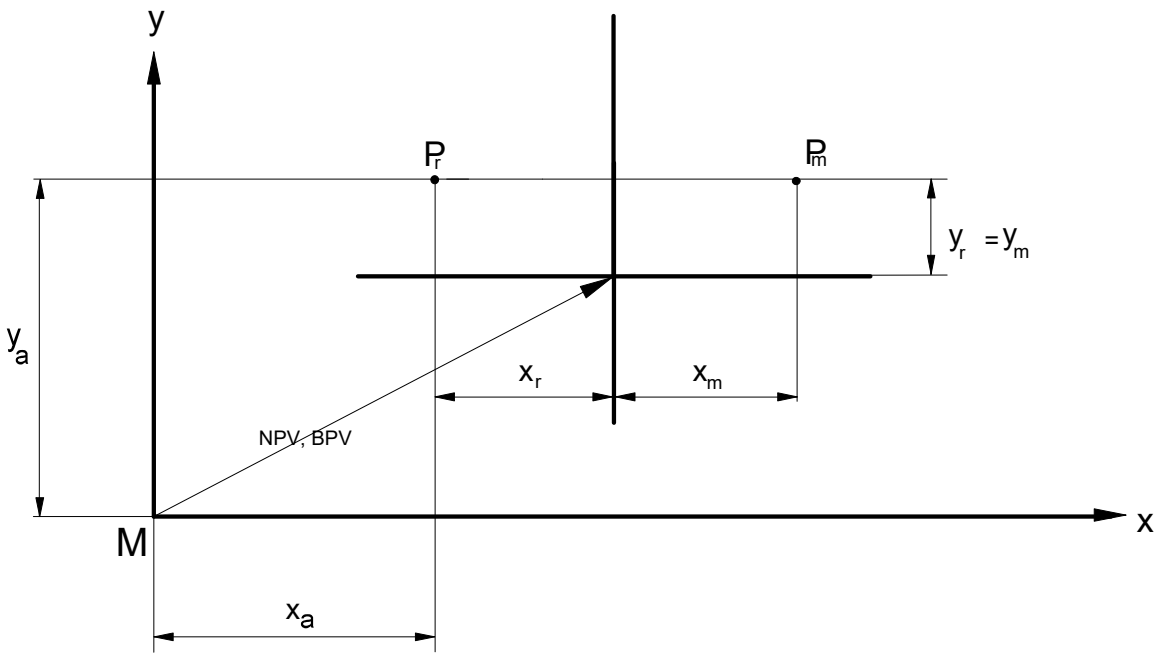


Abb. 39: Mentale und gespiegelte (reale) Koordinaten mit G21

x_m , y_m	Mentale Koordinaten
x_r , y_r	Gespiegelte Koordinaten
x_a , y_a	Absolute Koordinaten

Konturen, die zu spiegeln sind, sollten in ein Unterprogramm abgelegt werden. Dieses Unterprogramm wird dann nach der Spiegelfunktion aufgerufen. Die Spiegelung ist haltend wirksam.

Hinweis

Wird die Spiegelung bei angewählter Werkzeugradiuskorrektur programmiert, so ändert sich auch die Seite der Korrektur. D.h. bei aktivem G41 wird die Äquidistante zur programmierten Bahn nach der Spiegelung rechts zur Kontur berechnet (G42: WRK rechts der Kontur). Dies gilt auch bei geänderter Verfahrrichtung.

Werkzeugversätze und Verschiebungen (z.B. G54, G92, #PSET ...) werden bei G21, G22 und G23 nicht gespiegelt.

Im Gegensatz dazu wird bei der Spiegelfunktion G351 [▸ 124] die Bezugspunktverschiebung (G92) mit gespiegelt.

Programmierbeispiel

Spiegeln in der Ebene (G21)

```
%L DREIECK              (Unterprogramm)
N10 G90 X10 Y20         (Festlegung der zu spiegelnden)
N20 G91 X10 Y-10        (mentalen Koordinaten)
N30 X-10 Y-10
N40 Y20
N50 M29

%SPIEGELUNG             (Hauptprogramm)
N10 G92 G90 X60 Y40     (Bezugspunktverschiebung)
N20 G01 F500            (Gerade mit Vorschub 500)
N30 G21                 (Spiegelung in der X-Achse)
N40 LL DREIECK          (Aufruf Unterprogramm)
N60 G92 G90 X0 Y0       (Rücknahme der BPV)
N70 M30
```

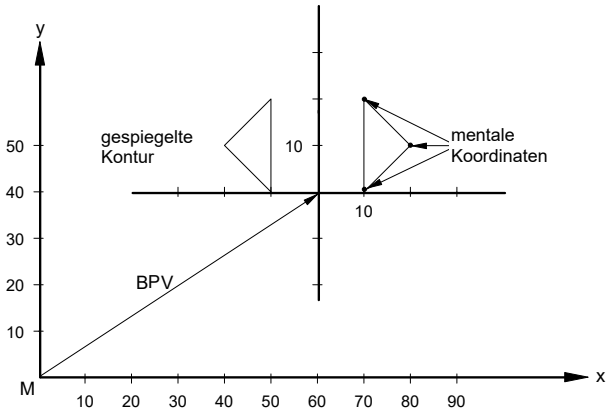


Abb. 40: Beispiel für die Spiegelung

Achtung

Bei der Spiegelung von Kreisen ist folgendes zu beachten:

Es werden generell nur die im NC-Programm gesetzten Koordinaten gespiegelt (bei einem Vollkreis also nur die Mittelpunktkoordinaten). Nach Anwahl der Spiegelung erfolgt eine Verfahrbewegung direkt auf den gespiegelten Endpunkt. Der Startpunkt der Verfahrbewegung wird nicht gespiegelt.

Folge: Im Verfahrsatz wird nur der Endpunkt, nicht die komplette Verfahrbewegung gespiegelt (siehe folgende Abb.: " Spiegelung des Endpunktes im Verfahrsatz")

Auswirkung: Wird als Verfahrsatz ein Vollkreis programmiert und Start-/Endpunkt liegen nicht in 0/0, so ergibt sich durch die Spiegelung der Mittelpunktkoordinaten ein neuer Kreisradius und damit eine geänderte Kontur (siehe Abb.: "Änderung der Kontur bei Spiegelung eines Vollkreises")!

Auswirkung:

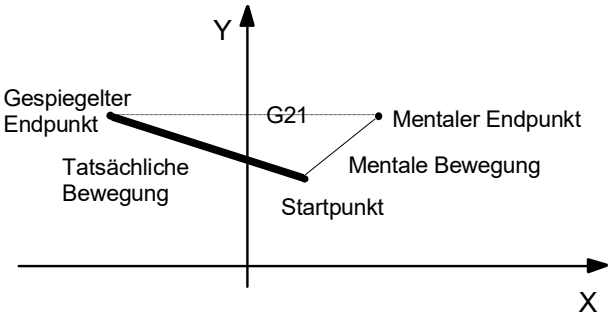


Abb. 41: Spiegelung des Endpunktes im Verfahrsatz

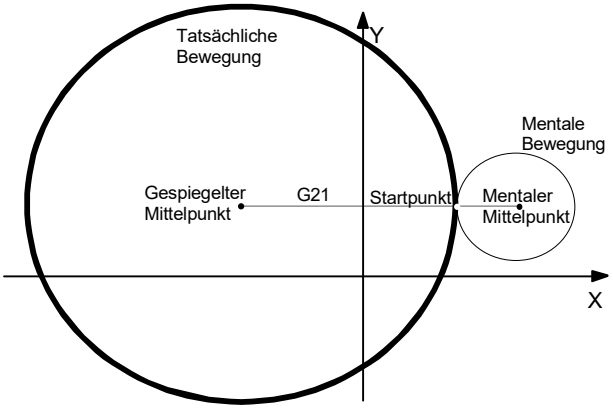
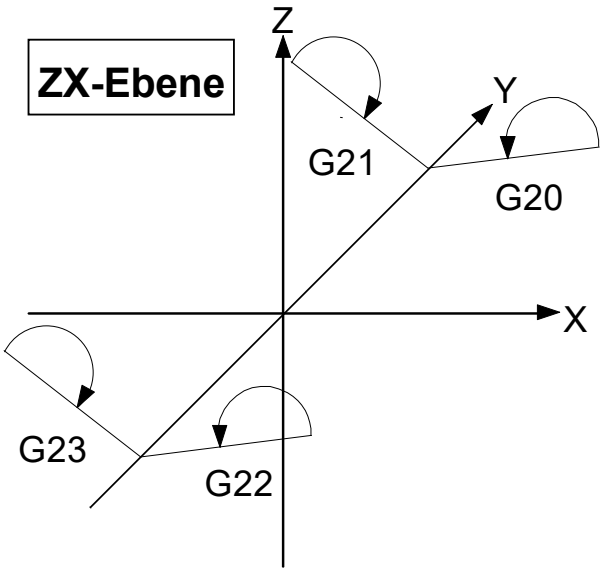


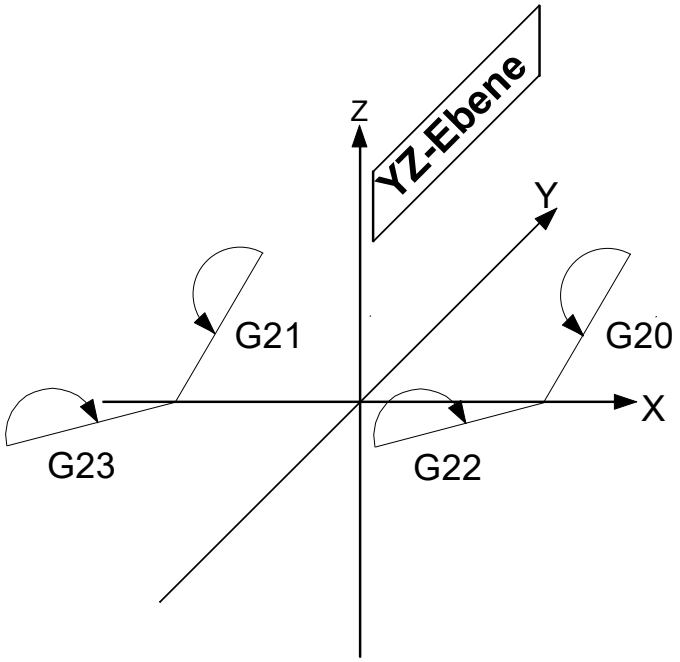
Abb. 42: Änderung der Kontur bei Spiegelung eines Vollkreises

Betrachtungen im Raum:

Werden die Spiegel-Funktionen in einer anderen als der XY-Ebene angewählt (G18, G19) so ist zu beachten, dass sich bei Zirkularinterpolation eventuell die Drehrichtung in Abhängigkeit von der angewählten Ebene ändert: Außerdem haben die Spiegelungen keinen Einfluß auf die Z-Koordinate, d.h. mentale und gespiegelte Werte in Z-Richtung sind immer identisch.



Betrachtet man die ZX-Ebene (Blickrichtung parallel zur Y-Achse), so liegen die Bewegungen der Modi G21 und G23 sowie G22 und G20 übereinander.



Betrachtet man die YZ-Ebene (Blickrichtung parallel zur X-Achse), so liegen die Bewegungen der Modi G21 und G20 sowie G22 und G23 übereinander.

Abb. 43: Auswirkungen der Spiegelfunktionen auf die Kreisdrehrichtung in verschiedenen Ebenen

4.6 Spiegelung mit Achsangabe (G351)

Mit G21 bis G23 wird die Spiegelung nur für die ersten beiden Hauptachsen angewählt. Die nachfolgend beschriebene Syntax mit G351 ermöglicht die freie Programmierung der Spiegelung von Achsen.

Syntax:

G351 <*Achsname*> [[**+**] | **-**] **1** { <*Achsname*> [[**+**] | **-**] **1** } nicht modal

G351	Achsspezifische Anwahl der Spiegelung. Die G-Funktion G351 ist nur satzweise gültig, aber die Spiegelung ist für eine mit dieser Funktion programmierten Achse haltend (modal).
<*Achsname*>..	Der Koordinatenwert der Achse bestimmt die An- oder Abwahl der Spiegelung in der Achse.
	Koordinatenwert -1: Anwahl der Spiegelung
	Koordinatenwert 1 oder +1: Abwahl der Spiegelung

Programmierbeispiel

Spiegelung mit Achsangabe (G351)

```
G351 X-1 Y1 Z+1     (Anwahl der Spiegelung in der X-Achse und Abwahl der)
                    (Spiegelung in der Y- und Z-Achse)
```

- Die zu spiegelnden Achsen können an beliebiger Stelle im NC-Satz programmiert werden.
- Zusammen mit G351 muss mindestens eine Achskoordinate programmiert sein.
- Die Funktion G351 muss alleine im NC-Satz programmiert sein. Ausnahme ist die Satznummer N.
- Eine wiederholte An- oder Abwahl der Spiegelung einer Achse ist zulässig. Bei wiederholter Programmierung im gleichen NC-Satz wird jedoch eine Fehlermeldung ausgegeben.
- Wird im Synchronbetrieb für die führende Achse (Masterachse) eine Spiegelung angewählt, so wird nicht automatisch auch für die geschleppte Achse (Slaveachse) die Spiegelung angewählt. Die geschleppte Achse wird aber immer entsprechend den Verfahrbewegungen der führenden Achse mitgeschleppt. Eine zusätzliche Verfahrbewegung infolge einer Spiegelung der Masterachse wirkt sich somit auch auf die Slaveachse aus.
- Bei Programmstart und Reset ist die Spiegelung für alle Achsen abgewählt. Beim Achstausch wird die Spiegelung der getauschten Achse abgewählt.
- Die Spiegelung der ersten oder zweiten Hauptachse beeinflußt die Verfahrrichtung bei Kreisinterpolation und die Werkzeugradiuskorrektur.
- Bei der Programmierung einer Spiegelung bei aktiver Werkzeugradiuskorrektur wird die Anwahlseite (G41, G42) automatisch gewechselt. Dies ist nur bei Linearsätzen erlaubt.

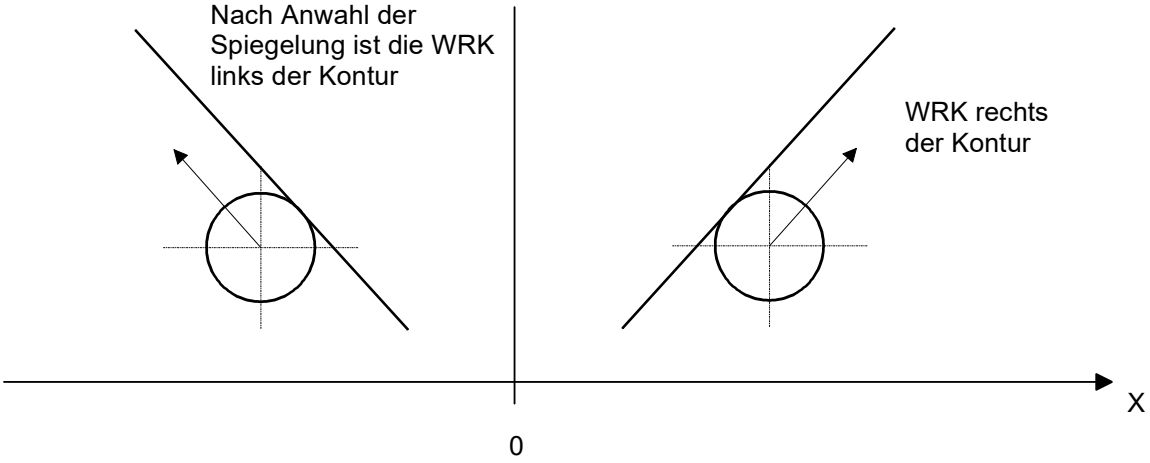


Abb. 44: Spiegelung der Anwahlseite bei aktiver WRK

- Wenn in einer mit G351 gespiegelten Achse eine Bezugspunktverschiebung G92 wirksam ist, so werden die Koordinaten der Bezugspunktverschiebung ebenfalls gespiegelt.

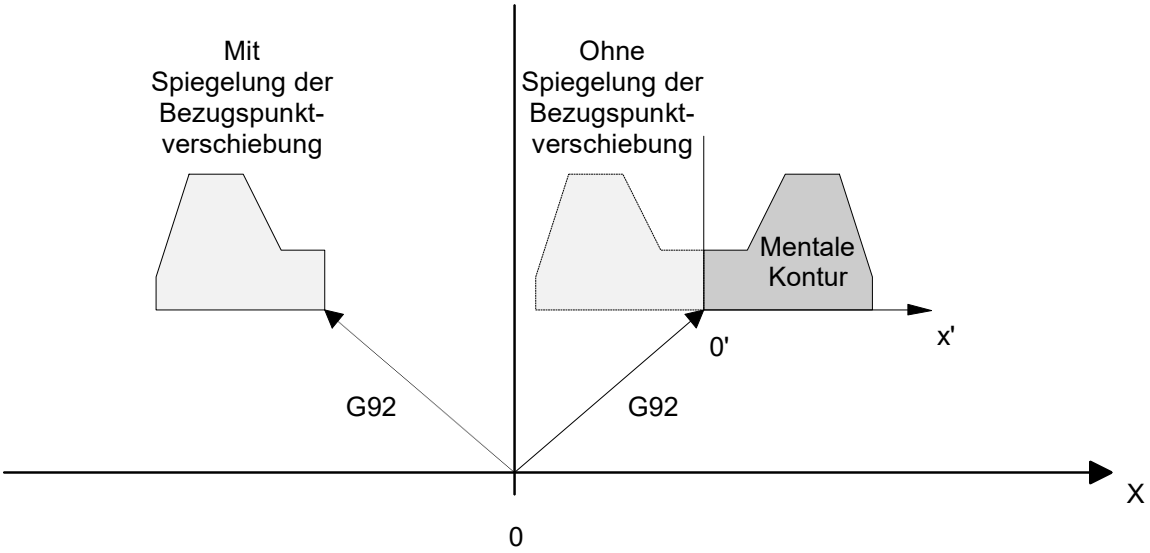


Abb. 45: Spiegelung der Bezugspunktverschiebung G92

- Die Koordinaten des Kreismittelpunkts I, J, K werden ebenfalls gespiegelt (siehe Kapitel Ebenenauswahl (G17/G18/G19) [▶ 119])
- Beim Einfügen von Fasen und Radien (G301/G302) wird das I-Wort als Fasenlänge bzw. als Radius eingelesen. Eine Berücksichtigung der Spiegelung ist somit nicht erforderlich.

Programmierbeispiel

Spiegelung mit Achsangabe (G351)

Nachfolgend einige Beispiele für die Verwendung der Funktion G351. Hierbei seien die Achsen X, Y, und Z die 1., 2. und 3. Hauptachse.

```
N10 G351 X-1          (Anwahl der Spiegelung in der X-Achse (G21))

N20 G351 Y-1          (Anwahl der Spiegelung in der Y-Achse(G22))

N30 G351 X-1 Y-1      (Anwahl der Spiegelung in der X- und Y-Achse (G23))

N40 G351 X1 Y+1       (Abwahl der Spiegelung in der X- und Y-Achse (G20))

N50 X1 G351 Y-1 Z1    (Anwahl der Spiegelung in der Y-Achse und Abwahl)
                      (der Spiegelung in der X- und Z-Achse)
```

4.7 Maßeinheiten (G70/G71)

Syntax:

G70	Eingaben in Zoll (mm, inch)	modal
G71	Eingaben metrisch	modal, Grundzustand

Die Anweisung G70 oder G71 wirkt auf alle Weg- und Koordinatenangaben.

Ausnahme: Schreiben/Lesen von Werkzeugparametern (V.G.WZxx.Pxx.) und Kinematikparametern (V.G.WZxx.KIN_PARAMxx) erfolgt immer direkt ohne Berücksichtigung der Maßeinheit.

4.8 Implizite Unterprogrammaufrufe (G80–G89/G800..)

Syntax:

G80 – **G89** [[<Val1>,<Val2>, - ,<Val50>]]	Unterprogrammaufruf	nicht modal
oder zusätzlich		
G800 -G8xx [[<Val1>,<Val2>, - ,<Val50>]]	Unterprogrammaufruf	nicht modal

G80-G89 / G800–G8xx

Bei der Programmierung von G80–G89, G800 – G819 bzw. G800-G839** wird implizit ein zugeordnetes globales Unterprogramm aufgerufen bzw. ausgeführt. Die Defaultnamen dieser Unterprogramme können entweder in den Kanalparametern P-CHAN-00160 - P-CHAN-00169 und P-CHAN-00187 konfiguriert oder zur Programmlaufzeit über den Befehl #FILE NAME [▶ 438] definiert werden.

Ist bei Programmierung von G80–G89, G800 – G819 bzw. G800-G839** kein Programmname hinterlegt, so wird die Fehlermeldung ID 20131 "Unbekannte G-Funktion" erzeugt. Das globale Unterprogramm wird nur einmal aufgerufen, d.h. ein G80–G89, G800 – G819 bzw. G800-G839** hat keine modale Wirkung.

<*Val1*>**, - ,**<*Val50*>

Optional können in einer nachfolgenden Klammerung maximal 50 Übergabeparameter (mathematische Ausdrücke im REAL-Format) in **einer festen Reihenfolge** zur Versorgung eines Unterprogrammes (Zyklus) aufgeführt werden. Die Parameter sind durch Kommas getrennt. Lücken in der Reihenfolge müssen durch aufeinanderfolgende Kommas ", ," markiert sein.

Durch die Angabe von Übergabeparametern wird der Unterprogrammaufruf zu einem Zyklusaufruf und gemäß den Regeln für Zyklen behandelt.

Im Unterprogramm können die Parameter analog zur Zyklenprogrammierung über @Px-Zugriffe ausgelesen werden. Hierbei besteht eine feste Zuordnung zwischen dem Parameter und dem @Px-Lesezugriff (z.B. @P1 liest Parameterwert 1, @P2 liest Parameterwert 2 usw.). Auch die zusätzliche erweiterte Zyklensyntax mit dem @-Zeichen kann in den auf diese Art aufgerufenen Unterprogrammen genutzt werden. Ob ein Parameter programmiert (gültig) ist, kann im Unterprogramm (Zyklus) über die Variable V.G.@P[i].VALID [▶ 602] ermittelt werden.

**Erweitert auf 40 Aufrufe (G800 – G839) ab V3.1.3079.23

Ein G80–G89 bzw. G800... wird immer als letzte Aktion am Satzende ausgeführt, d.h. sind im gleichen NC-Satz noch Achsbewegungen programmiert, so werden diese immer vor dem Aufruf des globalen Unterprogramms ausgefahren.

Programmierbeispiel

Implizite Unterprogrammaufrufe (G80–G89/G8xx)

Für G80 soll das globale UP - g80_up_test.nc - aufgerufen werden:

```
N10 #FILE NAME[ G80="g80_up_test.nc" ]
Nx ..
N30 G80        Aufruf g80_up_test.nc als globales UP
:
Für G815 soll das globale UP - g815_up_test.nc - aufgerufen werden:
..
N10 #FILE NAME[ G815="g815_up_test.nc" ]
Nx ..
N30 G815       Aufruf g815_up_test.nc als globales UP
:
G85 ruft das globale UP - cycle_test.nc - mit Parametern auf:
N10 #FILE NAME[ G85="cycle_test.nc" ]
Nx ..
N30 G85 [10,2, ,15,-3, ,5]     Aufruf cycle_test.nc als globales UP
```

Beispiel 2:

```
G803[5, @P1, @P2, @P3]
```

Die Bedeutung der obigen Zeile:

```
G803[@P1=5, @P2=@P1, @P3=@P2, @P4=@P3]
```

Das Ergebnis: alle übergebenen Parameter sind vom gleichen Wert: 5

4.9 Maßsysteme (Absolutmaß/Kettenmaß) (G90/G91)

Syntax:

G90	Absolutmaß	modal, Grundzustand
G91	Kettenmaß	modal

Bei der Absolutmaßeingabe (G90) beziehen sich alle Koordinatenangaben auf den Koordinatennullpunkt, d.h. der Zahlenwert eines Verfahrsatzes gibt die Zielposition im Koordinatensystem an.

Bei der Kettenmaßprogrammierung (G91) beziehen sich die Koordinatenangaben auf die Zielposition des vorhergehenden Wegsatzes, d.h. der Zahlenwert eines Wegsatzes gibt den zu fahrenden Weg an.

4.9.1 Exklusive Programmierung

In der Grundeinstellung darf im NC-Satz immer nur ein Maßsystem angewählt werden. Doppelprogrammierung bzw. Programmierung von G90 und G91 im gleichen NC-Satz ist nicht erlaubt. Die Position von G90/G91 innerhalb des NC-Satzes hat keine Bedeutung.

Programmierbeispiel

Exklusive Programmierung

```
:
N10 X10 Y10 (Absolutmaßsystem G90 ist angewählt, Grundeinstellung)
N20 G91 X20 Y20 (Abwahl Absolut- und Anwahl Relativprogrammierung)
N30 X30 G90 Y30 (Abwahl Relativ- und Anwahl Absolutprogrammierung)
:
N100 G90 Z30 G90     (Fehlermeldung: Doppelprogrammierung von G90/G91)
N110 G91 X10 G90 Z30 (Fehlermeldung: Doppelprogrammierung von G90/G91)
```

4.9.2 Kombinierte Programmierung

Über den Kanalparameter P-CHAN-00116 kann die exklusive Maßsystemprogrammierung für die Koordinatenangabe von Bahnachsen aufgehoben werden. Dann können in einem NC-Satz gleichzeitig Absolut- und Relativmaßangaben programmiert werden. Auch die wiederholte Programmierung von G90 bzw. G91 im gleichen NC-Satz ist zulässig.

Die Position von G90/G91 innerhalb des NC-Satzes ist dann von Bedeutung. Das zuletzt programmierte Maßsystem ist für die im NC-Satz folgenden Bahnachspositionen und alle weiteren NC-Sätze bis zum nächsten G90/G91 gültig.

Programmierbeispiel

Kombinierte Programmierung

```
:
N10 X10 Y10 (Absolutmaßsystem G90 ist angewählt, Grundeinstellung)
N20 G91 X20 G90 Y20 (Relativ für X-Achse, Absolut für Y-Achse)
N30 X30 G91 Y30 Z20 (Absolut für X-Achse, Relativ für Y/Z-Achse)
N30 G90 X30 G91 Y30 G90 Z20 (Absolut für X/Z-Achse, Relativ für Y-Achse)
:
N100 G90 G91 Z30 (Relativ für Z-Achse)
N110 G91 X10 G90 Z30 (Relativ für X-Achse, Absolut für Z-Achse)
```

Hinweis

G90/G91 hat keinen Einfluss auf die Hilfskoordinaten I, J, K der Kreis- bzw. Helikalinterpolation. Ihr Maßsystem wird ausschließlich durch G161/G162 bestimmt.

4.10 Genauhalt (G60/G360/G359)

Syntax:

G60	Genauhalt	nicht modal
... oder bei Genauhalt über mehrere Sätze:		
G360	Anwahl Genauhalt	modal
G359	Abwahl Genauhalt	modal

G60/G360 erlauben das genaue Anfahren einer Zielposition innerhalb der Genauhaltgrenzen. Die Vorschubgeschwindigkeit wird bis zum Satzende auf null verringert und der Schleppabstand abgebaut.

Der Genauhalt kann verwendet werden, wenn Ecken exakt zu bearbeiten sind oder wenn bei einer Richtungsumkehr die Zielposition exakt erreicht werden muss. Die Weiterschaltbedingung ist das Erreichen eines parametrierbaren Regelfensters (P-AXIS-00236).

Programmierbeispiel

Beispiele zu Genauhalt

```
N10 G1 F1000 X0 Y0
N20 G1 G60 X10  ( Genauhalt für dieses Satzende )
N30 G1 X20
N40 G60 Y20     ( Genauhalt für dieses Satzende )
N50 G1 X30
;...
```

Analog dazu die Umsetzung mit G360 und G359

```
N10 G1 F1000 X0 Y0
N20 G1 X10
N30 G360        ( Anwahl Genauhalt )
N40 G1 Y20

N60 G1 X30
N70 G359        ( Abwahl Genauhalt )
;..
```

Wird ein G60 ohne Bewegungsinformation in einer NC-Zeile programmiert, so wird der Fehler ID 20003 ausgegeben.

4.11 Polynomüberschleifen (G61/G261/G260)

Syntax:

G61	Polynomüberschleifen (am Satzende)	nicht modal

... oder bei Polynomüberschleifen über mehrere Sätze:

G261	Anwahl Polynomüberschleifen (am Satzende)	modal
G260	Abwahl Polynomüberschleifen	modal

Unter Polynomüberschleifen versteht man die krümmungs- und richtungsstetige Verbindung zweier Bewegungssätze. Dazu werden die ursprünglich programmierten Bewegungssätze verkürzt. Zwischen den Sätzen wird eine Überschleifkurve eingefügt. Das Verfahren erlaubt das Überschleifen zwischen den Übergängen Gerade-Gerade, Gerade-Kreis und Kreis-Kreis (siehe Abbildung unten). Es ist nicht auf die Ebene beschränkt, sondern erlaubt vielmehr das Überschleifen zwischen beliebig im Raum liegenden Kurven.

Bewegungssätze ohne Fahrweg werden hier nicht berücksichtigt. (Siehe Relevante Satzlänge [▶ 268]).

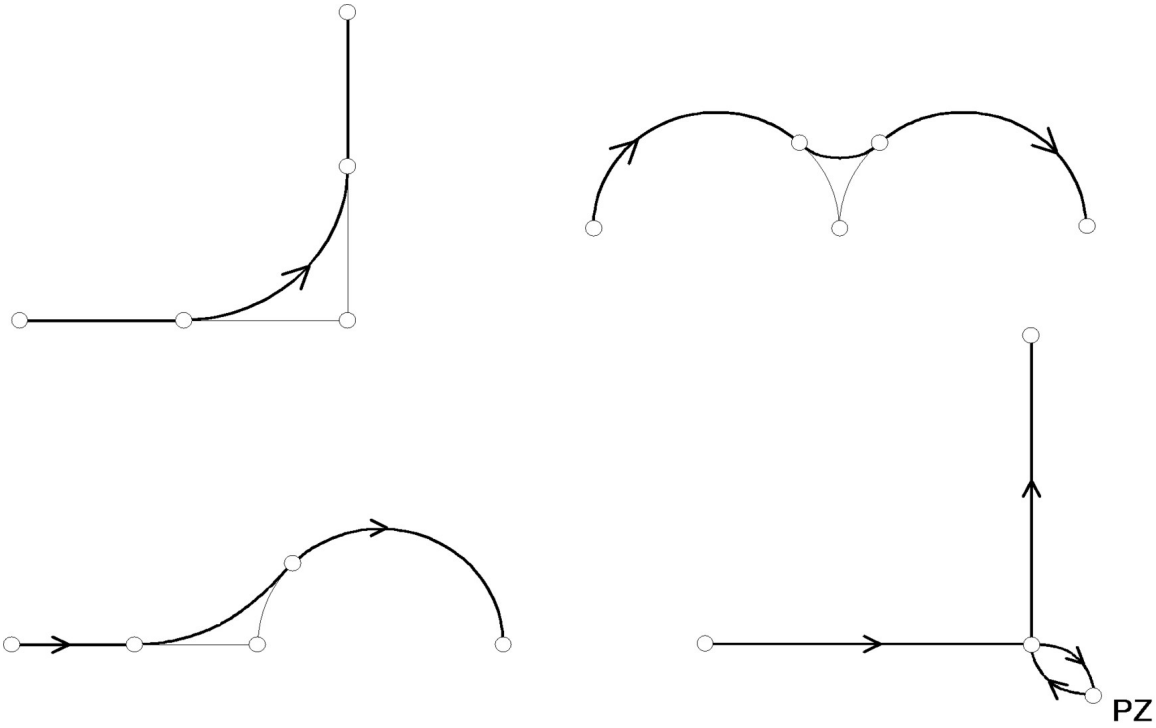


Abb. 46: Beispiele für Polynom-Überschleifen

Folgende Überschleifarten stehen zur Verfügung:

- Überschleifen mit Eckenabweichung
- Dynamisch optimiertes Überschleifen
- Dynamisch optimiertes Überschleifen mit Leitachse
- Überschleifen mit Zwischenpunkt
- Dynamisch optimiertes Überschleifen der Kontur

Je nach Überschleifart können die charakterisierenden Randbedingungen vorgegeben werden. Die Parameter bleiben so lange gültig, bis der Überschleifvorgang vollständig ausgeführt wurde. Werden die Überschleifparameter zwischen Vorsatz und Nachsatz geändert, so wirkt sich die Änderung erst auf den nächsten Überschleifvorgang aus.

Weitere Informationen sind im Kapitel Polynomüberschleifen für lange Sätze (G61/G261/G260) [▶ 264].

4.12 Eckenverzögerung

Im Bereich von Innenecken steigt beim Fräsen in Abhängigkeit von der Satzübergangsgeometrie das Zerspanvolumen V_z und die damit zusammenhängende notwendige Spindelleistung an, da das Werkzeug bereits Material der Anschlusskontur abträgt (siehe Abbildung unten).

Wird grundsätzlich an der Leistungsgrenze der Spindel gearbeitet, muss die Vorschubgeschwindigkeit im Eckenbereich so reduziert werden, dass die Spindelleistung auch an Innenecken ausreicht. Zur Einhaltung der Leistungsgrenze der Spindel muss daher die Möglichkeit bestehen, aus dem NC-Programm heraus einen Punkt auf der Bahn zu definieren, ab dem die Geschwindigkeit reduziert werden soll. Für diese Eckenverzögerung steht je ein NC-Befehl zur Parametrierung, zur Aktivierung und zur Deaktivierung zur Verfügung.

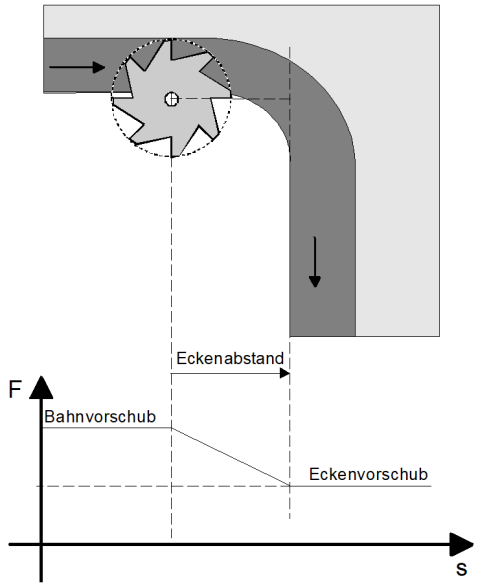


Abb. 47: Änderung der Zerspanvolumens Vz über der Zeit bei einer Innenecke von 90° bei konstantem Vorschub

4.12.1 Parametrierung der Eckenverzögerung (#CORNER PARAM)

Versionshinweis

Ab Version **V2.11.2010.02** ersetzt der Befehl **#CORNER PARAM [...]** den Befehl **#SET CORNER PARAM [...]**. Dieser ist aus Kompatibilitätsgründen weiterhin verfügbar, es wird aber empfohlen, diesen in neuen NC-Programmen nicht mehr zu verwenden.

Syntax:

#CORNER PARAM [DIST=.. **UNIT=**<*ident*> **FEED**=.. **]**

DIST=..	Eckenabstand in [mm, inch]
UNIT=<*ident*>	Einheit des Eckenvorschubes FEED=... Zulässige Kennungen:
	FWORD Einheit gemäß aktuellem F-Wort
	PERCENT Einheit in [%]
FEED=..	Eckenvorschub, [gemäß UNIT<*ident*>]

Durch die Vorgabe eines Eckenabstandes kann der Punkt im Raum errechnet werden, ab welchem der Bahnvorschub linear über dem Weg reduziert werden soll. Der zu programmierende Eckenabstand bezieht sich hierbei auf die korrigierte Bahn, nicht auf die ursprünglich programmierten Zielpunkte.

Hinweis

Die Eckenverzögerung ist nur in Verbindung mit dem linearen Slope P-CHAN-00071 wirksam.

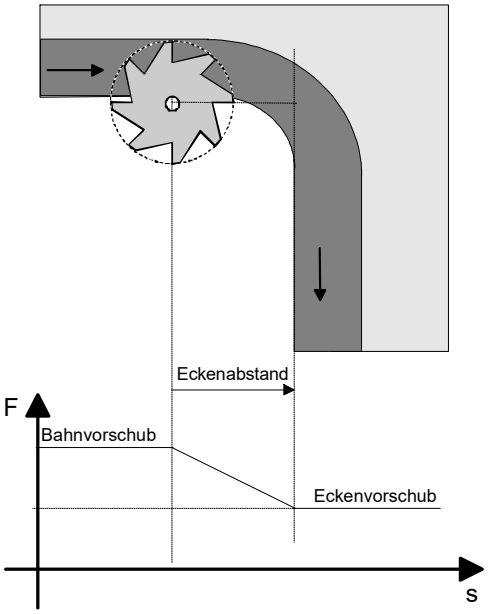


Abb. 48: Darstellung des Vorschubes an einer runden Innenkontur

4.12.2 An-/Abwahl der Eckenverzögerung (G12/G13)

Syntax:

G12	Deaktivierung der Eckenverzögerung	modal, Grundzustand
G13	Aktivierung der Eckenverzögerung	modal

Bei Programmierung von G13 ohne vorherige Festlegung der Eckenverzögerungs-Parameter erfolgt die Ausgabe einer Fehlermeldung.

Bei gleichzeitiger Programmierung von G12 und G13 in einem NC-Satz erfolgt die Ausgabe einer Fehlermeldung.

Diese G-Funktionen sind haltend wirksam. Der Grundzustand nach Systemhochlauf ist „Eckenverzögerung Inaktiv“ (G12).

Bei angewählter Eckenverzögerung (G13) und Programmende (M30) wird keine Fehlermeldung ausgegeben, allerdings wird vor Programmstart die Grundstellung G12 angewählt.

Bei einem Eckenabstand von 0 wird eine Fehlermeldung ausgegeben.

Programmierbeispiel

An- und Abwahl der Eckenverzögerung (G12/G13)

```
:
N10 G01 G90 X10 Y10 F1000
N15 #CORNER PARAM[DIST=10 UNIT=PERCENT FEED=50] ;Eckenabstand 10mm
                                                ;Eckenvorschub 50%
N20 G13      ;Aktivierung
N30 X50      ;Eckenverzögerung aktiv
N40 Y50      ;Eckenverzögerung aktiv
N50 G12      ;Abwahl
N60 X30 Y30
N70 M30
```

Es ist zu beachten, dass die Eckenverzögerung erst im Satz N030 aktiviert wird. Der Satzübergang N010/N030 ist von der Eckenverzögerung nicht betroffen.

Ebenso ist die Eckenverzögerung im Satz N040 noch aktiv, da sie nicht abgewählt worden ist.

Die Abwahl der Eckenverzögerung muss vor bzw. im NC-Satz programmiert sein, in dem die Eckenverzögerung nicht mehr aktiv sein soll.

4.13 Nullpunktverschiebungen (G53/G54/...G59)

Im Folgenden wird Nullpunktverschiebung mit NPV abgekürzt.

Syntax:

G53	Abwahl der NPV*	modal, Grundzustand
G54	Anwahl 1. NPV	modal
G55	Anwahl 2. NPV	modal
G56	Anwahl 3. NPV	modal
G57	Anwahl 4. NPV	modal
G58	Anwahl 5. NPV	modal
G59	Anwahl 6. NPV	modal

Die Bedeutung des G53-Datensatzes kann gesteuert werden durch P-ZERO-00001. Abhängig von der Parametrierung bedeutet G53 entweder die Abwahl der NPV oder die Nutzung des G53-Datensatzes als zusätzliche NPV.

Die Standardeinstellung der automatisch im Grundzustand aktiven NPV ist ebenfalls parametrierbar P-ZERO-00002.

Mit G54 ... G59 wird die entsprechende Nullpunktverschiebung aus der Nullpunktverschiebungstabelle gültig. Die Nullpunktverschiebung ist bereits in dem Satz gültig, in dem G53, G54, ... steht. Es finden jedoch ohne Koordinatenangaben keine Verfahrbewegungen statt!

In der Grundstellung (G53) ist keine Nullpunktkorrektur wirksam.

Hinweis

Des Weiteren gilt für die Anwahl der Nullpunktverschiebung im G91-Modus:

Die Programmierung von...

N10 G54

N20 G0 X0 G91

...ruft keine Bewegung der X-Achse hervor, da es sich hierbei um eine Bewegung relativ um Null handelt.

Eine Nullpunktverschiebung wirkt sich also erst dann aus, wenn die nächste Verfahrinformation absolut (G90) programmiert wird!

4.13.1 Zusätzliche Nullpunktverschiebungs-Variablen

Variable	Bedeutung	lesen	schreiben
V.G.NP[j].ALL	Ansprechen aller Achsen einer Nullpunktverschiebung	ja	ja
V.G.NP_AKT.V[i] oder V.G.NP_AKT.V.<*Achsname*>	Aktuelle (momentan aktive) Nullpunktverschiebung einer Achse mit Listenindex <*i*> oder <*Achsname*>	ja	ja
V.G.NP_AKT.ALL	Ansprechen der aktuellen (momentan aktiven) Nullpunktverschiebungen aller Achsen	ja	ja
V.G.NP_DEFAULT	Index der im Grundzustand wirksamen Nullpunktverschiebung	ja	ja

Der schreibende Zugriff über die Variablen V.G.NP[j] und V.G.NP_DEFAULT ist bis zur nächsten Listeninterpretation (expliziter Auftrag oder Hochlauf) wirksam. Hingegen wirken schreibende Zugriffe über V.G.NP_AKT nur auf die momentan im Speicher des Decoders befindliche Nullpunktverschiebung. Bei wiederholter Anwahl der zuvor über V.G.NP_AKT beschriebenen NPV werden wieder die ursprünglichen Werte aus der Liste wirksam. Soll der Schreibzugriff auf V.G.NP_AKT über die Abwahl hinaus gehalten werden, so muss der aktuelle Datensatz durch Zuweisung über V.G.NP[j].ALL in der Liste abgelegt werden.

Programmierbeispiel

Zusätzliche Nullpunktverschiebungs-Variablen

Sichern der momentan aktiven Nullpunktverschiebungswerte:

```
V.G.NP[12].ALL=V.G.NP_AKT.ALL
```

4.13.2 Addition/ Subtraktion von Verschiebungen

Die Neubelegung kompletter NPV-Datensätze durch additive Zuweisung bereits vorhandener NPV wird über die Variable V.G.NP[j].ALL ermöglicht. Dabei werden immer die Verschiebungen aller Achsen verrechnet. Erlaubt sind bei der NPV-Zuweisung die Schreibweisen ` += `, ` -= ` und `=`.

Programmierbeispiel

G54 (NPV1) wird die Kombination von G55 (NPV2) und G57 (NPV4) zugewiesen:

```
N10 V.G.NP[1].ALL = V.G.NP[2].ALL + V.G.NP[4].ALL
```

Programmierbeispiel

Die gleiche Operation, jedoch unter zusätzlicher Berücksichtigung von G54:

```
N10 V.G.NP[1].ALL = V.G.NP[1].ALL + V.G.NP[2].ALL + V.G.NP[4].ALL
```

oder

```
N10 V.G.NP[1].ALL += V.G.NP[2].ALL + V.G.NP[4].ALL
```

Achtung

Die Verknüpfung von V.G.NP[j].ALL-Variablen mit achsspezifischen V.G.NP[j].V[i]-Variablen oder Konstanten innerhalb einer Zuweisung ist nicht möglich!

Programmierbeispiel

G54 (NPV1) definiert sich durch die Kombination von G55 (NPV2), der X-Verschiebung von G57 (NPV4) und einem Korrekturwert:

FALSCH:

```
N10 V.G.NP[1].ALL = V.G.NP[2].ALL + V.G.NP[4].V.X + 100
```

RICHTIG:

Die Zuweisung muss in zwei Schritten erfolgen:

```
N10 V.G.NP[1].ALL = V.G.NP[2].ALL
N20 V.G.NP[1].V.X = V.G.NP[4].V.X + 100
```

4.13.3 Zugriff auf die aktuelle Nullpunktverschiebung

Über die Variable V.G.NP_AKT.V[i] bzw. V.G.NP_AKT.V.<*Achsname*> erfolgt der Zugriff auf die aktuell aktive NPV im Decoder. Der Bediener muss dabei nicht wissen, welche NPV (d.h. welcher Index) momentan angewählt ist.

Programmierbeispiel

Die aktuelle NPV der X-Achse soll den Wert 200 betragen.

```
N10 V.G.NP_AKT.V[0] = 200 oder V.G.NP_AKT.V.X = 200
```

Die aktuelle NPV soll in allen Achsen um die Verschiebungswerte aus G55 (NPV2) erweitert werden.

```
N10 V.G.NP_AKT.ALL = V.G.NP_AKT.ALL + V.G.NP[2].ALL
```

oder

```
N10 V.G.NP_AKT.ALL += V.G.NP[2].ALL
```

Die Änderung der aktuellen NPV ist *nicht* programmübergreifend. Durch eine weitere Zuweisung hat der Bediener jedoch die Möglichkeit, seine aktuelle NPV auch in anderen Programmen zu nutzen.

Programmierbeispiel

Die aktuelle NPV soll für spätere Verwendungen in anderen Programmen unter G54 (NPV1) gerettet werden.

```
N10 V.G.NP[1].ALL = V.G.NP_AKT.ALL
```

4.13.4 Default-Nullpunktverschiebung

Die nach Programmstart aktive NPV ist parametrierbar (P-ZERO-00002).

Im NC-Programm hat der Bediener die Möglichkeit, über die Variable V.G.NP_DEFAULT auf den Defaultindex zuzugreifen. Die Änderung gilt programmübergreifend.

Programmierbeispiel

Ab dieser Programmzeile soll G55 (NPV2) die neue Default-NPV sein, d.h. beim folgendem Programmstart ist automatisch G55 aktiv.

```
N100 V.G.NP_DEFAULT = 2
```

4.13.5 Bildung von Nullpunktverschiebungsgruppen

Eine Nullpunktverschiebungsgruppe (NPVG) definiert die Bildungsvorschrift für eine NPV, d.h. sie beschreibt, aus welchen Komponenten sich ein bestimmter NPV-Datensatz zusammensetzt. Die Definition einer NPVG erfolgt im NC-Programm oder durch Vorbelegung im Kanalparametersatz [1] [▶ 894]-1 in Verbindung mit einem Makronamen. Die Verwendung von Makronamen ist in Kapitel Makros [▶ 729] beschrieben. Um eine NPVG zu aktivieren, sind prinzipiell drei Schritte notwendig:

Programmierbeispiel

Bildung von Nullpunktverschiebungsgruppen

1. Schritt:

Definition einer NPVG im NC-Programm z.B. unter dem Makroname VERSCH_1:

```
N10 " VERSCH_1 " = " V.G.NP[1].ALL = V.G.NP[2].ALL + V.G.NP[4].ALL "
```

oder im Kanalparametersatz [1] [▶ 894]-1:

```
makro_def[0].symbol VERSCH_1
makro_def[0].nc_code V.G.NP[1].ALL = V.G.NP[2].ALL + V.G.NP[4].ALL
```

2. Schritt:

Bildung der NPV (Neubelegung des NPV1-Datensatzes):

```
N20 " VERSCH_1 "
```

3. Schritt:

Anwahl der NPV (in diesem Beispiel NPV1, d.h. G54)

```
N30 G54
```

4.13.6 Erweiterte Nullpunktverschiebung (G159)

Syntax:

G159=..	Erweiterte Nullpunktverschiebung mit Angabe Listenindex des Nullpunktdatensatzes	modal

Zugriff auf zusätzliche NPV-Datensätze [3] [▶ 894]. Der Zugriff auf die G53...G59-Datensätze ist ebenfalls möglich. Die maximale Anzahl der erweiterten NPV ist parametrierbar [6] [▶ 894]-6.12.

Programmierbeispiel

Erweiterte Nullpunktverschiebung (G159)

```
G159 = 0     ;entspricht G53, Listenindex 0
G159 = 1     ;entspricht G54, Listenindex 1
G159 = 2     ;entspricht G55, Listenindex 2
G159 = 3     ;entspricht G56, Listenindex 3
G159 = 4     ;entspricht G57, Listenindex 4
G159 = 5     ;entspricht G58, Listenindex 5
G159 = 6     ;entspricht G59, Listenindex 6
G159 = 7     ;Datensatz mit Listenindex 7
:
G159 = 10    ;Datensatz mit Listenindex 10
:
```

4.13.7 Nullpunktverschiebungen achsspezifisch freigeben/verriegeln (G160)

Syntax:

G160= .. *<Achsname>..* { *<Achsname>..* } modal

G160=..	Achsspezifische Nullpunktverschiebung mit Angabe Listenindex *<i>* des Nullpunktdatensatzes
<Achsname>..	Achse mit Gültigkennung ihrer Nullpunktverschiebung 0: Nullpunktverschiebung der Achse wird verrechnet. 1: Nullpunktverschiebung der Achse wird nicht verrechnet.

In jedem Nullpunktdatensatz *<i>* kann vorab durch Parametrierung mit P-ZERO-00004 festgelegt werden, für welche Achsen *<Achsname>* der Versatz eingerechnet bzw. nicht eingerechnet werden soll, d.h. einzelne Achsversätze können gezielt verriegelt bzw. freigeschaltet werden.

Diese achsspezifische Gültigkeit einer Nullpunktverschiebung kann im NC-Programm mit G160 verändert werden.

Programmierbeispiel

Vor der Anwahl von G55 (Nullpunktdatensatz mit Index 2) werden die Versätze in der X- und der Z-Achse verriegelt und die von der Y-Achse freigegeben.

```
:
N10 G160 = 2 X1 Y0 Z1
N20 G55
:
```

Im nächsten Verfahrsatz werden somit nur die Achsversätze von G55 in die Bewegung eingerechnet, die **nicht** verriegelt sind (Y-Achse).

4.14 Mittelpunktsangabe bei Kreisdefinition (G161/G162)

Syntax:

G161	Kreismittelpunkt absolut	modal
G162	Kreismittelpunkt relativ	modal, Grundzustand

Bei wirksamem G162 (Grundzustand) wird der Mittelpunkt über I, J und K relativ zum Kreisstartpunkt definiert.

Bei wirksamem G161 sind I, J und K absolute Angaben im Koordinatensystem des Programmierers.

4.15 Mittelpunktskorrektursteuerung im Kreis (G164/G165)

Syntax:

G164	Mittelpunktskorrektur aus	modal
G165	Mittelpunktskorrektur ein	modal, Grundzustand

Bei wirksamem G165 wird ein über I-, J- und K-Angabe programmierter Kreis so korrigiert, dass bei festgelegter Kreisrichtung (G02/G03), festgelegtem Startpunkt (Endpunkt des vorhergehenden Satzes) und festgelegtem Endpunkt (Koordinatenangabe im Kreissatz) ein Kreisbogen interpoliert werden kann. Eventuell wird dabei der mit I, J und K programmierte Mittelpunkt verschoben! Diese Verschiebung ist umso geringer, je exakter der genaue Mittelpunkt angegeben wurde.

Hinweis

Bei der Kreisprogrammierung mit Radiusangabe R wirkt keine Kreismittelpunktskorrektur, da der Mittelpunkt hier immer exakt berechnet wird.

Für die Abweichung von programmiertem zu korrigiertem Mittelpunkt werden zwei Grenzwerte P-CHAN-00059 und P-CHAN-00060 überwacht, bei deren Überschreitung eine Fehlermeldung abgesetzt wird:

mittelpkt_diff:	Zulässige Abweichung in 10-4 mm
mittelpkt_faktor:	Prozentuale Abweichung in 0,1 %

Es wird überprüft, ob die Mittelpunktverschiebung Δm größer ist, als die absolute Größe "mittelpkt_diff"

Δm > mittelpkt_diff ?

und, ob die Mittelpunktverschiebung Δm größer ist, als das Produkt von "mittelpkt_faktor/1000" und korrigiertem Radius "radius"

Δm > mittelpkt_faktor/1000 * radius ?

Die obere Grenze ist für Δm also vom berechneten Radius linear abhängig. Es ergibt sich der dargestellte Zusammenhang zwischen Mittelpunktsverschiebung Δm und berechnetem Radius "radius".

Beispiel:

"mittelpkt_faktor" = 5 bedeutet, dass der Abstand zwischen den programmierten Mittelpunktkoordinaten und den korrigierten Mittelpunktkoordinaten maximal 0,5 % des korrigierten Kreisradius sein darf.

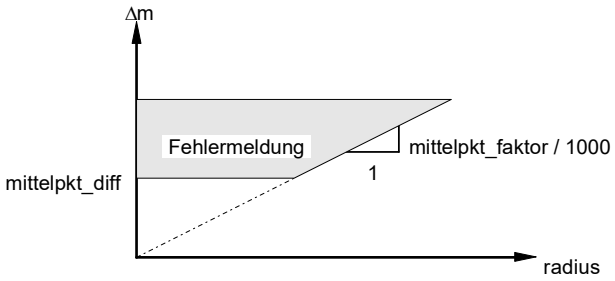


Abb. 49: Zusammenhang zwischen Mittelpunktverschiebung Δm und berechnetem "radius"

Die programmierten Mittelpunktkoordinaten müssen also in einem Umkreis um den korrigierten Kreismittelpunkt liegen. Der Radius des Umkreises entspricht der zulässigen Mittelpunktverschiebung Δm, die durch die beiden Parameter "mittelpkt_diff" und "mittelpkt_faktor" eingestellt werden kann.

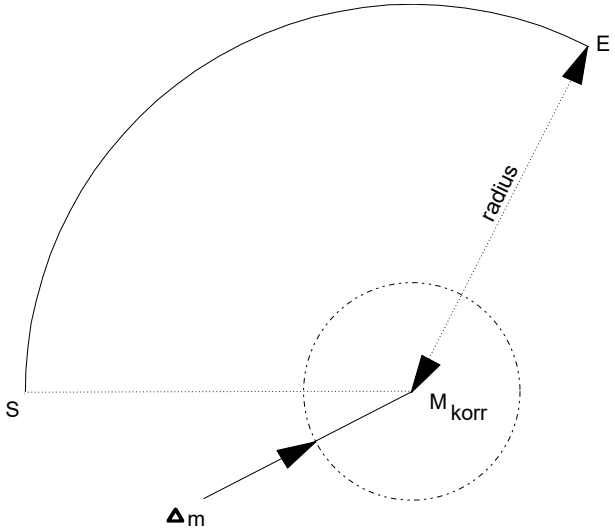


Abb. 50: Gebiet der zulässigen programmierten Mittelpunkte

4.15.1 Sonderfunktion Kreisradienausgleich in Verbindung mit G164

In bestimmten Situationen kann die Kreismittelpunktskorrekur (G165) zu einer ungünstigen Verschiebung des programmierten Kreismittelpunkts und damit der Lage des Kreises führen. Solche ungünstigen Situationen können auftreten, wenn Kreisstartpunkt und Kreiszielpunkt dicht beieinander liegen und der Kreis fast einem programmierten Vollkreis entspricht.

Im nachfolgenden Beispiel ist solch ein Kreis mit Zielpunktangabe mit und ohne Kreismittelpunktkorrektur (G165/G164) programmiert. Um den Auflösungsfehler im Postprozessor zu simulieren, ist bei aktiver Kreismittelpunktkorrektur der Kreiszielpunkt um jeweils 0.1µm in x und y Richtung gegenüber dem Startpunkt verschoben programmiert. Der Kreis mit G165 wird um den Startpunkt gedreht und die Position des korrigierten Kreismittelpunktes M_K verschiebt sich sehr stark gegenüber dem programmierten Mittelpunkt M.

Programmierbeispiel

Sonderfunktion Kreisradienausgleich in Verbindung mit G164

```
N10 G00 G90 X0 Y0 Z0
N20 G01 X-50 Y0 F20000
N30 G01 X0
N40 G165 G02 X0.0001 Y0.0001 J200
N50 G01 Y50
M30

N10 G00 X0 Y0 Z0
N20 G01 X-50 Y0 F20000
N30 G01 X0
N40 G164 G02 X0.0 Y0.0 J200
N50 G01 Y50
N60 M30
```

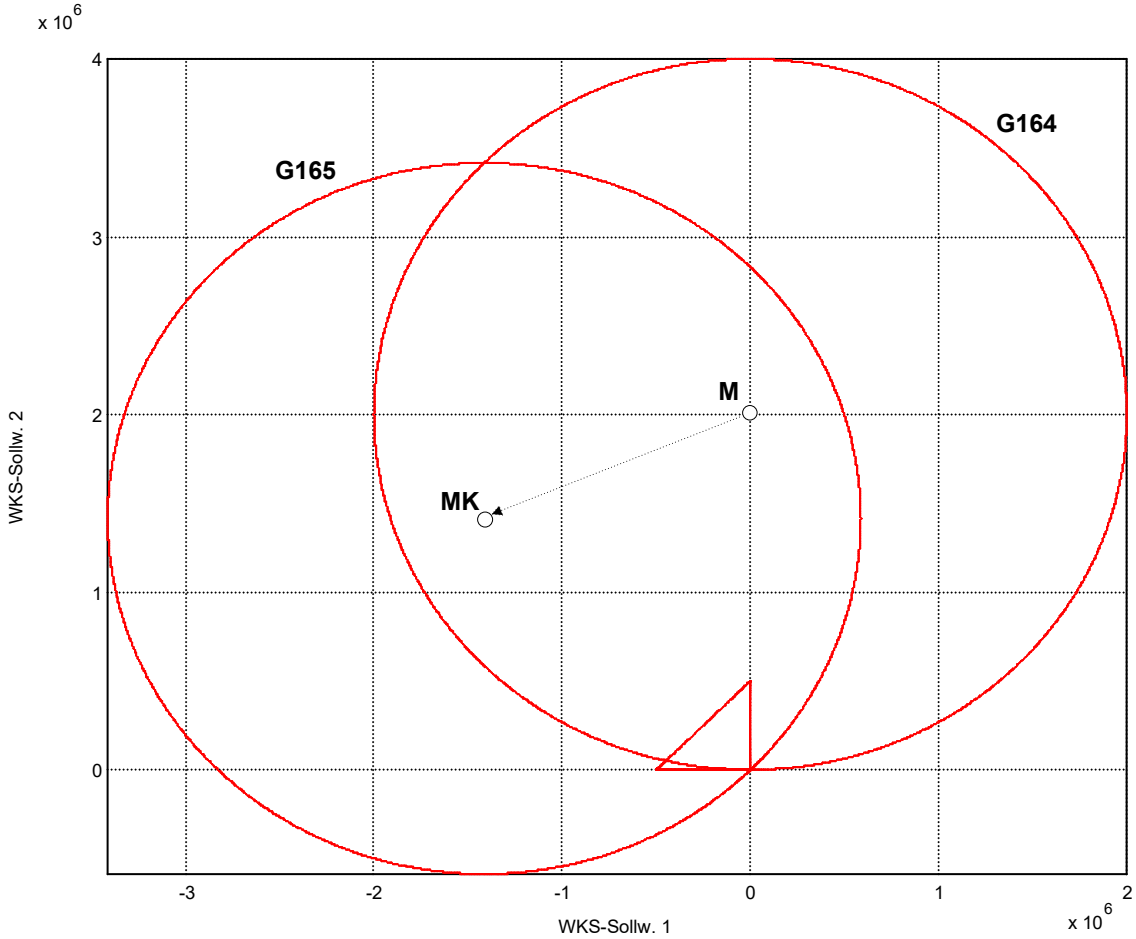


Abb. 51: Kreismittelpunktsverschiebung bei G165

In solchen Fällen kann mit inaktiver Kreismittelpunktskorrektur (G164) und Einstellung von max_radius_diff_circle > 0 (P-CHAN-00171) und max_proz_radius_diff_circle (P-CHAN-00172) i. A. ein besseres Ergebnis erzielt werden. Der programmierte Kreismittelpunkt wird durch die Funktion nicht verändert, die Kreisradiendifferenz wird linear über dem Kreiswinkel vom Startradius in den Zielradius überführt.

Wenn die Kreisradienabweichungen in der Größenordnung der Auflösungsgenauigkeit liegen, sind die Kreisverzerrungen und dynamische Effekte i. A. vernachlässigbar!

Hinweis

Für einen Vollkreis müssen auch hier Kreisstartpunkt und Kreiszielpunkt identisch sein!

4.16 Vorsteuerung (G135/G136/G137)

Syntax:

G135	Anwahl der Vorsteuerung	modal
G136 *<Achsname>..* { *<Achsname>..* }	Angabe der Gewichtung	modal
G137	Abwahl der Vorsteuerung	modal

Durch den Einsatz einer Geschwindigkeits- und Beschleunigungsvorsteuerung können die Bahnverzerrungen reduziert werden.

Die achsgruppenspezifische Aktivierung wird mit G135 programmiert. Eine achsspezifische, prozentuale Gewichtung der berechneten Vorsteuergrößen in [%] erfolgt über G136. Sie ist für alle Achsen auf 100% begrenzt.

Mit G137 wird die Vorsteuerung achsgruppenspezifisch ausgeschaltet. Bei Achsen, für die nach der globalen Anwahl mit G135 keine Vorsteuerung erfolgen soll, muss mit G136 als prozentuale Gewichtung 0% angegeben werden. Es ist auch möglich, die Anwahl und Gewichtung der Vorsteuerung in einem Satz einzugeben.

Bei jedem Programmstart wird im Interpolator die Vorsteuerung explizit ausgeschaltet und die Gewichtungsfaktoren werden auf 100% gesetzt.

Wenn die Vorsteuerung während des NC-Programms ein- bzw. ausgeschaltet wird, bleiben die Gewichtungsfaktoren auf dem durch G136 eingestellten Werten bzw., wenn kein G136 programmiert wird, auf 100%.

Achtung

Nach einem Achstausch werden die G136 – Gewichtungsfaktoren der daran beteiligten Achsen wieder auf 100% gesetzt.

Programmierbeispiel

Vorsteuerung (G135/G136/G137)

```
G135    (Anwahl der Vorsteuerung: Gewichtung für alle Achsen 100%)
G136 X80 Y95 Z0(Gewichtung;  Hier wird Z-Achse nicht vorgesteuert)
G137    (Abwahl der Vorsteuerung: Gewichtung für alle Achsen 100%)
```

4.17 Gewichtung der Maximalgeschwindigkeit (G127/G128)

Syntax:

G127 <*Achsname*>.. { <*Achsname*>.. }	Achsspezifische Gewichtung der Maximalgeschwindigkeit: Gewichtung für bestimmte Achsen in [%] **[ab V2.11.2808.14]**	modal
G128 =..	Achsgruppenspezifische Gewichtung der Maximalgeschwindigkeit. Gewichtung für alle Achsen in [%].	modal

Mit den Funktionen G127/G128 ist es möglich, die Maximalgeschwindigkeit zu verändern.

Eine Beeinflussung der Maximalgeschwindigkeit wird erreicht, indem man den entsprechenden Geschwindigkeitskennwert prozentual verändert.

Bei einer Programmierung mit G127/G128 sind alle nicht bzw. noch nicht programmierten Achsen auf 100% eingestellt. Jede weitere Anwahl dieser Funktionen bezieht sich unabhängig von vorhergehenden Programmierungen auf 100%, d.h. die Geometriedatenverarbeitung gewichtet stets den Default-Wert P-AXIS-00212 mit dem prozentualen Wert.

Zweimal hintereinander 50% programmiert bedeutet also, dass auf 50% und nicht auf 25% eingestellt wird.

Hinweis

Die Vorschubgewichtung der Maximalgeschwindigkeit wirkt nicht auf programmierte Einzelachsbewegungen wie z.B. Referenzpunktfahrt, Handbetrieb und unabhängige Achsen.

Achtung

Bei G127/G128 > 100% wird die Maximalgeschwindigkeit auf den MAX-Wert P-AXIS-00212 begrenzt.

Bei G127/G128 = 0 wird die Maximalgeschwindigkeit auf den MIN-Wert 1µm/s begrenzt.

Programmierbeispiel

Gewichtung der Maximalgeschwindigkeit (G127/G128)

```
N10 G127 X70 Y60          ;Achsspezifische Gewichtung von vb_max
                          ;vb_max der X-Achse wird auf 70% begrenzt
                          ;vb_max der Y-Achse wird auf 60% begrenzt
N20 G128 = 100            ;Achsgruppenspezifische Gewichtung von vb_max
                          ;vb_max aller Achsen auf 100%
Besonderheit:
N20 G128 = 100 X10 Y20    ;Bei G128 können im gleichen Satz auch Achs-
                          ;positionen programmiert werden!
```

4.18 Gewichtung der Eilganggeschwindigkeit (G129)

Syntax:

G129 =.. Achsgruppenspezifische Eilganggewichtung. Gewichtung für **alle** Achsen in [%] modal

Mit der Funktionen G129 ist es möglich, die Eilganggeschwindigkeit G00 zu verändern.

Eine Beeinflussung der Vorschubgeschwindigkeit wird erreicht, indem man den entsprechenden Geschwindigkeitskennwert prozentual verändert.

Bei einer Programmierung mit G129 sind alle nicht bzw. noch nicht programmierten Achsen auf 100% eingestellt. Jede weitere Anwahl dieser Funktionen bezieht sich unabhängig von vorhergehenden Programmierungen auf 100%, d.h. die Geometriedatenverarbeitung gewichtet stets den Standardwert P-AXIS-00209 mit dem prozentualen Wert.

Zweimal hintereinander 50% programmiert bedeutet also, dass auf 50% und nicht auf 25% eingestellt wird.

Hinweis

Die Vorschubgewichtung wirkt nur bei Eilgangsätzen (G00). Sie wirkt nicht bei Einzelachsbewegungen wie z.B. Referenzpunktfahrt, Handbetrieb und unabhängigen Achsen.

Achtung

Bei G129 > 100% wird die Eilganggeschwindigkeit auf den MAX-Wert P-AXIS-00212 begrenzt.

Bei G129 = 0 wird die Eilganggeschwindigkeit auf den MIN-Wert 1µm/s begrenzt.

Programmierbeispiel

Gewichtung der Eilganggeschwindigkeit (G129)

```
N10 G129 = 70                (Achsgruppenspezifische Eilganggewichtung)
                             (Eilganggeschwindigkeit aller Achsen auf 70%)
N20 G00 X100 Y150            (Linearinterpolation, Eilgangbewegung mit 70%)
Besonderheit:
N50 G129 = 70 X10 Y20        (Bei G129 können im gleichen Satz auch Achs-)
                             (positionen programmiert werden!
```

4.19 Parametrierung des Beschleunigungsprofils

4.19.1 Beschleunigungsgewichtung (G130/G131/G230/G231/G333/G334)

Mit den folgenden G-Funktionen kann die Achsbeschleunigung verändert werden.

Syntax der Gewichtungen für Beschleunigungen:

G130 <*Achsname*>.. { <*Achsname*>..}	Achsspezifische Beschleunigungsgewichtung, Gewichtung für bestimmte Achsen in [%], wirkt nur bei Vorschubsätzen G1/G2/G3	modal
G131=..	Achsgruppenspezifische Beschleunigungsgewichtung bei G01, G02, G03, Gewichtung für alle Achsen in [%]	modal
G230 <*Achsname*>.. { <*Achsname*>..}	Achsspezifische Beschleunigungsgewichtung bei G00, Gewichtung für bestimmte Achsen in [%] **[ab V3.1.3080.11]**	
G231=..	Achsgruppenspezifische Beschleunigungsgewichtung bei G00, Gewichtung für alle Achsen in [%]	modal
Gewichtungen für Beschleunigungen im Feedhold:		
G333 <*Achsname*>.. { <*Achsname*>..}	Achsspezifische Beschleunigungsgewichtung bei Feedhold, Gewichtung für bestimmte Achsen in [%] **[ab V3.1.3079.16]**	modal
G334=..	Achsgruppenspezifische Beschleunigungsgewichtung bei Feedhold, Gewichtung für alle Achsen in [%] **[ab V3.1.3079.16]**	modal

Eine Beeinflussung dieser Beschleunigung wird erreicht, indem man die entsprechende Standardbelegung der Beschleunigungskennwerte prozentual verändert. Bei ruckbegrenztem Profil sind dies die Achsparameter P-AXIS-00001 und P-AXIS-00002.

Bei gesetztem P-CHAN-00097 wird der Feedhold-Parametersatz für den Bremsvorgang verwendet. In diesem Fall kann mit G333/G334 die Verzögerung der Feedhold-Rampe P-AXIS-00024 prozentual verändert werden. Die Feedholdgewichtung ist nur wirksam wenn die resultierende Verzögerung gleich oder größer ist als die aktiven Werte der gewichteten G01/G00 Beschleunigungen.

Bei Programmierung der Beschleunigungsgewichtung sind alle nicht bzw. noch nicht programmierten Achsen auf 100% eingestellt. Jede weitere Anwahl dieser Funktionen bezieht sich unabhängig von vorhergehenden Programmierungen auf 100%, d.h. die Geometriedatenverarbeitung gewichtet stets die Standardwerte [2] [▶ 894]-1 bzw. [2] [▶ 894]-2 mit dem prozentualen Wert.

Zweimal hintereinander 50% programmiert bedeutet also, dass auf 50% und nicht auf 25% eingestellt wird. Eine Gewichtung über 100% hinaus ist bis zur Maximalbeschleunigung der Achse P-AXIS-00008 möglich.

Eine Reduzierung der Beschleunigung kann alternativ auch über die Control Unit „Reduzierung der Bahnbeschleunigung“ erfolgen.

Achtung

Nach einem Achstausch werden die G130/G230/G333-Gewichtungsfaktoren der daran beteiligten Achsen wieder auf 100% gesetzt.

Hinweis

Die Beschleunigungsgewichtung wirkt **nicht** bei Einzelachsbewegungen wie z.B. Referenzpunktfahrt, Handbetrieb und unabhängigen Achsen.

Programmierbeispiel

Beschleunigungsgewichtung (G130/G131/G230/G231/G333/G334)

```
N10 G130 X70                ;Achsspezifische Beschleunigungsgewichtung
                            ;Beschleunigung der X-Achse wird auf 70% begrenzt
N20 G01 F1000 X100          ;Linearinterpolation
N30 G130 Y60                ;Beschleunigung der Y-Achse wird auf 60% begrenzt
                            ;Beschleunigung der X-Achse bleibt bei 70%
N40 Y100                    ;Linearinterpolation
N50 G131 = 100              ;Achsgrp.spezifische Beschleunigungsgewichtung
                            ;G01,G02,G03 Beschleunigung aller Achsen auf 100%
N60 G231 = 80               ;Achsgrp.spezifische Beschleunigungsgewichtung
                            ;G0 Beschleunigung aller Achsen auf 80%
N65 G230 Z70                ;Achsspezifische Beschleunigungsgewichtung
                            ;G0 Beschleunigung der Z-Achse wird auf 70% begrenzt
N70 G00 X200                ;Eilgang
N80 G333 X150               ;Achsspezifische Beschleunigungsgewichtung
                            ;bei Feedhold. Verzögerung der X-Achse wird auf
                            ;150% erhöht.
N90 G334 = 200              ;Achsgrp.spezifische Beschleunigungsgewichtung
                            ;bei Feedhold. Verzögerung aller Achsen wird auf
                            ;200% erhöht.
```

Besonderheit:

```
N50 G131 = 100 X10 Y20      ;Bei G131/G231/G334 können im gleichen Satz
                            ;auch Achspositionen programmiert werden!
```

4.19.2 Rampenzeitgewichtung (G132/G133/G134/G233/G338/G339)

Syntax der Gewichtungen für Rampenzeiten:

G132 <*Achsname*>.. { <*Achsname*>..}	Achsspezifische Rampenzeitgewichtung, Gewichtung für bestimmte Achsen in [%], wirkt nur bei Vorschubsätzen G1/G2/G3	modal
G133=..	Achsgruppenspezifische Rampenzeitgewichtung bei G01, G02, G03, Gewichtung für alle Achsen in [%]	modal
G134=..	Achsgruppenspezifische Gewichtung der geometrischen Rampenzeit, Gewichtung für alle Achsen in [%]	modal
G233=..	Achsgruppenspezifische Rampenzeitgewichtung bei G00, Gewichtung für alle Achsen in [%]	modal

Gewichtungen für Rampenzeiten im Feedhold:

G338 <*Achsname*>.. { <*Achsname*>..}	Achsspezifische Rampenzeitgewichtung bei Feedhold, Gewichtung für bestimmte Achsen in [%] **[ab V3.1.3079.16]**	modal
G339=..	Achsgruppenspezifische Rampenzeitgewichtung bei Feedhold, Gewichtung für alle Achsen in [%] **[ab V3.1.3079.16]**	modal

Die Beeinflussung der Rampenzeiten wird erreicht, indem man die entsprechenden Standard-Rampenzeiten prozentual verändert. Mit den Funktionen G132/G133/G233 ist es möglich, die Rampenzeit der Achsbeschleunigung beim nichtlinearen Slope zu verändern [2] [▶ 894]-1. (Beim linearen Slope ist der Beschleunigungsverlauf sprungförmig (siehe Abbildung in Kapitel Ruckbegrenzender Slope [▶ 376])

Mit G134 ist es möglich, die geometrische Rampenzeit beim nichtlinearen Slope zu verändern (P-AXIS-00199).

Die Parameter für Rampenzeiten des Beschleunigungsauf- bzw. -abbaus sind P-AXIS-00196 bzw. P-AXIS-00195 sowie des Verzögerungsauf- bzw. -abbaus sind P-AXIS-00198 bzw. P-AXIS-00197.

Bei gesetztem P-CHAN-00097 wird der Feedhold-Parametersatz für den Bremsvorgang verwendet. In diesem Fall kann mit G338/G339 die Feedhold-Rampezeit P-AXIS-00081 prozentual verändert werden.

Bei einer Programmierung mit G132/G133/G134/G233/G338/G339 sind alle nicht bzw. noch nicht programmierten Achsen auf 100% eingestellt. Jede weitere Anwahl dieser Funktionen bezieht sich unabhängig von vorhergehenden Programmierungen auf 100%, d.h. die Geometriedatenverarbeitung gewichtet stets die Standardwerte mit dem prozentualen Wert. Zweimal hintereinander 50% programmiert bedeutet also, dass auf 50% und nicht auf 25% eingestellt wird.

Achtung

Nach einem Achstausch werden die G132/G338-Gewichtungsfaktoren der daran beteiligten Achsen wieder auf 100% gesetzt.

Hinweis

Die Rampenzeitgewichtung wirkt **nicht** bei Einzelachsbewegungen wie z.B. Referenzpunktfahrt, Handbetrieb und unabhängigen Achsen.

Programmierbeispiel

Rampenzeitgewichtung (G132/G133/G134/G233/G338/G339)

```
N10 G132 X200              ;Achsspezifische Rampenzeitgewichtung
                           ;Rampenzeit der X-Achse wird um 200% vergrößert
N20 G01 F1000 X100         ;Linearinterpolation
N30 G132 Y50               ;Rampenzeit der Y-Achse wird um 50% verkleinert
                           ;Rampenzeit der X-Achse bleibt bei 200%
N40 Y100                   ;Linearinterpolation
N50 G133 = 100             ;Achsgrp.spezifische Rampenzeitgewichtung
                           ;G01,G02,G03 Rampenzeiten aller Achsen auf 100%
N60 G134 = 50              ;Achsgrp.spezifische Rampenzeitgewichtung
                           ;Geometrische Rampenzeit aller Achsen auf 50%
N70 G233 = 80              ;Achsgrp.spezifische Rampenzeitgewichtung
                           ;G00 Rampenzeiten aller Achsen auf 80%
N80 G00 X200               ;Eilgang
N90 G338 X150              ;Achsspezifische Rampenzeitgewichtung
                           ;bei Feedhold. Rampenzeit der X-Achse wird auf
                           ;150% erhöht.
N100 G339 = 200            ;Achsgrp.spezifische Rampenzeitgewichtung
                           ;bei Feedhold. Rampenzeit aller Achsen wird auf
                           ;200% erhöht.
Besonderheit:
N50 G133 = 100 X10 Y20     ;Bei G133/G134/G233/G339 können im gleichen Satz
                           ;auch Achspositionen programmiert werden!
```

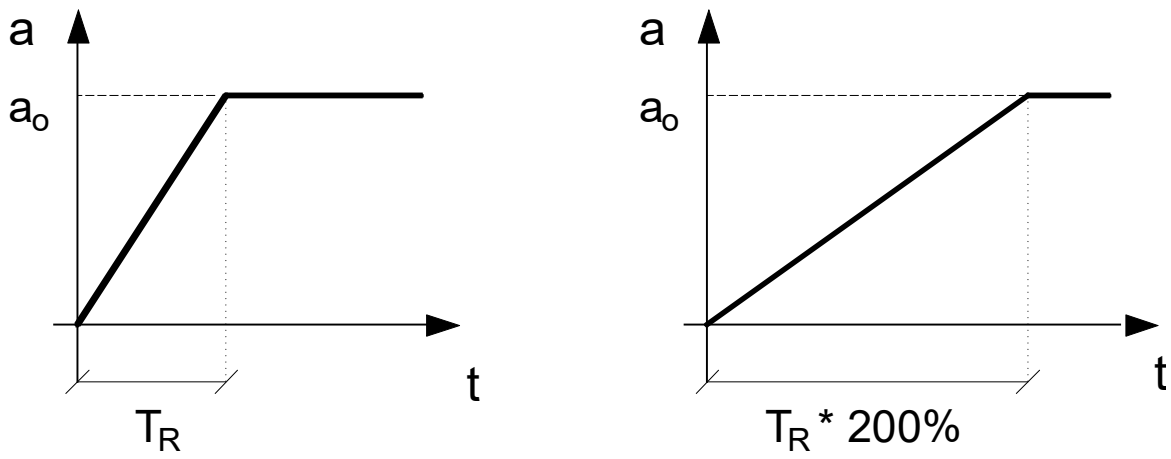


Abb. 52: Beispiel Rampenzeitgewichtung mit G132/G133/G233

Das folgende Bild zeigt für G134 den Einfluss auf die Beschleunigung quer zur Bahn (Zentrifugalbeschleunigung a1 --> a2) bei ansteigendem Vorschub (v1 --> v2) während einer Kreisfahrt (P1 --> P2).

Reduziert man die Zentrifugalbeschleunigungsänderung durch Vergrößerung der G134 Rampenzeit, dann wird sanfter beschleunigt und so die Zielbeschleunigung a2 erst im Punkt P3 erreicht.

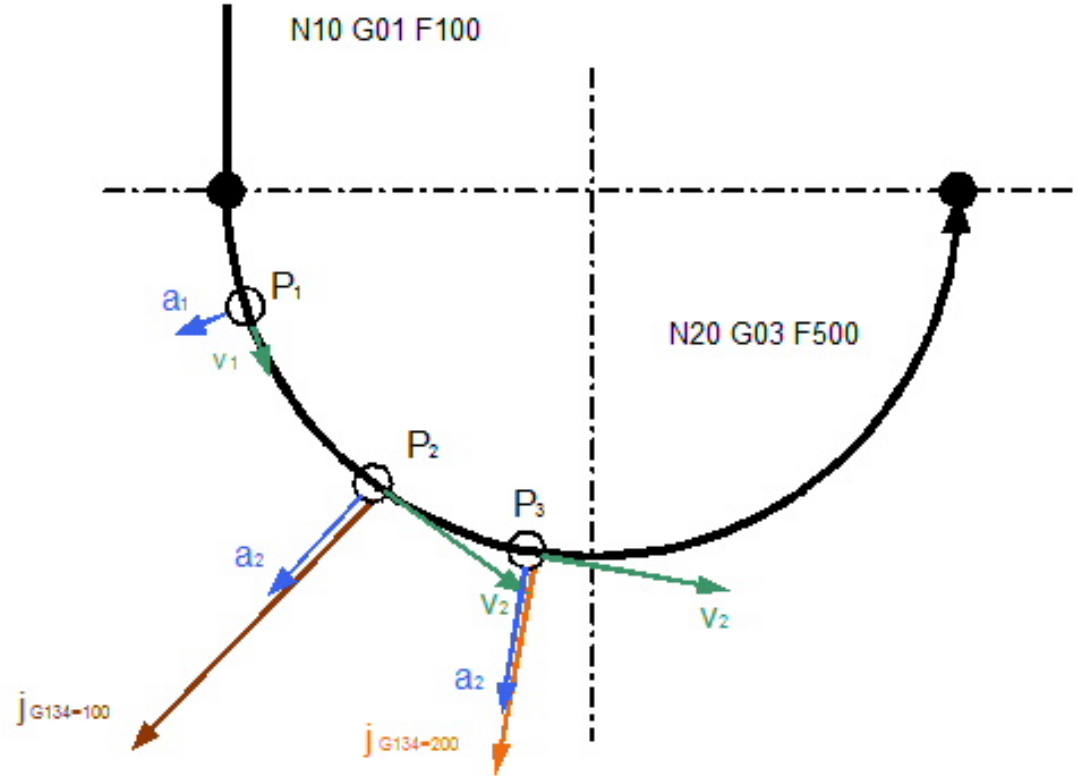


Abb. 53: Rampenzeitgewichtung mit G134 bei Zirkularinterpolation

4.20 Bearbeitungszeit/Vorschubgeschwindigkeit (G93/G94/G95/G194)

Syntax:

G93	Angabe der Bearbeitungszeit in Sekunden	modal
G94	Vorschub pro Minute	modal, Grundzustand
G95	Umdrehungsvorschub	modal
G194	Vorschubberechnung aufgrund gewichteter maximaler Achsvorschübe	modal

Mittels der G-Funktionen G93, G94, G95 und G194 kann die Wirkung des F-Wortes wahlweise umgeschaltet werden.

G93 definiert zusammen mit dem F-Wort eine Bearbeitungszeit in [s].

G94 definiert zusammen mit dem F-Wort:

- für Linearachsen den Vorschub in [mm/min, m/min, inch/min]
- für Rundachsen den Vorschub in [°/min]

G95 definiert zusammen mit dem F-Wort einen Umdrehungsvorschub in [mm/U, inch/U]. Die Funktion ist in Kapitel Umdrehungsvorschub (G95) [▶ 654] genauer beschrieben.

G194 definiert zusammen mit dem F-Wort einen Gewichtungsfaktor in [%] für die maximalen Vorschubgeschwindigkeiten. Die maximal zulässige Vorschubgeschwindigkeit auf der Bahn ergibt sich dann aus den gewichteten achsspezifischen Werten P-AXIS-00212. Mindestens eine Achse fährt dann mit ihrer gewichteten Maximalgeschwindigkeit. Es sind nur Gewichtungswerte kleiner 100% erlaubt.

Programmierbeispiel

Bearbeitungszeit/Vorschubgeschwindigkeit (G93/G94/G95/G194)

```
N10 G90 F1000 X100    (Vorschub 1000 mm/min (G94 Default))
N20 G194 F90          (Gewichtung 90% auf max. Achsvorschubgeschw.)
Nxx X200              (Vorschub z.B. 9000 mm/min bei vb_max=10000 mm/min)
N80 G94 X50           (Vorschub 1000 mm/min aus N10 gültig)
Nxx X.. Y.. Z..       (Interpolation)
N120 G93 F20          (Bearbeitungszeit 20 s)
Nxx X.. Y.. Z..       (Interpolation)
N160 G94 F1500 X150   (Vorschub 1500 mm/min)
Nxx X.. Y.. Z..       (Interpolation)
N200 M30
```

4.21 Einfügen von Fasen und Radien (G301/G302) (#FRC/#CHR/#CHF/#RND)

Syntax:

G301	Einfügen von Fasen	Beide Funktionen sind einmalig zwischen zwei Verfahrsätzen wirksam
G302	Einfügen von Radien	

Bei G301 wird eine Gerade mit gleichem Neigungswinkel zu den Nachbar-Konturelementen eingefügt (Fasen).

Bei G302 wird ein an zwei Nachbar-Konturelementen tangential übergehender Kreisbogen eingefügt (Runden).

Diese Funktionen sind satzweise wirksam und erzeugen genau ein Einfügesegment (Gerade bzw. Kreisbogen). G301/G302-Sätze dürfen nur zwischen Sätzen mit wirksamer G-Funktion der Gruppe "G00, G01, G02/G03 ohne G05" geschrieben werden.

Ein im selben NC-Satz programmiertes I-Wort definiert die Fasenbreite bzw. die Radiusgröße des Einfügesegmentes in [mm, inch]. Das I-Wort bleibt gespeichert wirksam, d.h. bei folgendem G301/G302 mit gleicher Fasenlänge bzw. gleichem Radius muss das I-Wort nicht mehr programmiert werden.

Bei erstmaliger Programmierung von G301/G302 muss im NC-Satz ein I-Wort ungleich Null programmiert werden, sonst wird eine Fehlermeldung erzeugt (Fehler, der zum Abbruch der Decodierung führt).

Wirksamkeit des Bahnvorschubs im eingefügten Fasen- oder Kreissegment:

- Geht G00 (Eilgang) voraus, so wird das Segment ebenfalls mit maximal möglicher Geschwindigkeit gefahren.
- Geht G01/G02/G03 voraus, so gilt im Segment ebenfalls der programmierte Vorschub.
- Im Satz mit G301/G302 darf ein Vorschub angegeben werden. Dieser gilt auch in allen nachfolgenden G01/G02/G03 Sätzen.
- Bei wirksamem G11 und G41/G42 wird der Vorschub auch angepasst.

Spezifischer Fasen- bzw. Radienvorschub:

Versionshinweis

Ab V3.1.3057.04 kann zusammen mit der Fasen- bzw. Radienangabe ein spezifischer, nur im eingefügten Fasen- oder Kreissegment wirksamer Vorschub programmiert werden.

Syntax:

#FRC=..	Vorschub im eingefügten Fasen- oder Kreissegment gemäß der Einheit des F-Wortes (z.B. mm/min)	nicht modal

Programmierbeispiel

Einfügen von Fasen und Radien (G301/G302)

Fasen: 90° Ecke mit 2 Geraden (für eine Fase 20x45° wird I=20 angegeben)

```
N100  G00  G91  X100  Y0
N110  G01           Y100 F200
N120  G301 I20
N130                X-60
N140  G00  G90  X0   Y0
```

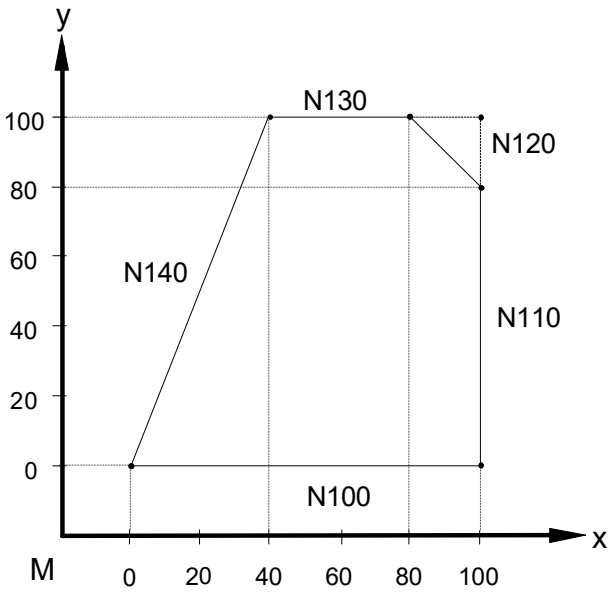
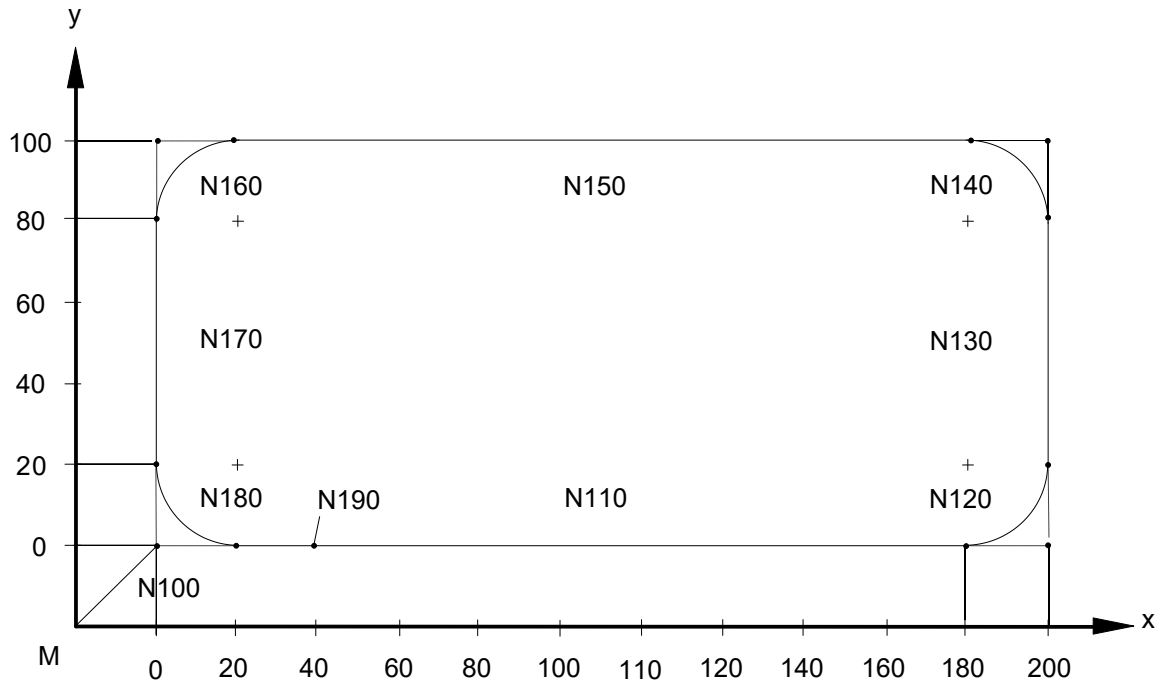


Abb. 54:

Runden: Rechtecktasche mit Eckenradius 20 mm, 200 mm lang, 100 mm breit, Vorschübe segmentspezifisch

```
Standard:
N100  G00   X0   Y0
N110  G01   X200      F200
N120  G302  I20       F150
N130           Y100   F200
N140  G302             F150
N150        X0         F200
N160  G302             F150
N170           Y0     F200
N180  G302             F150
N190       X40         F200
```

```
Alternativ:
N100  G00   X0   Y0
N110  G01   X200       F200
N120  G302  I20       #FRC=150
N130         Y100
N140  G302            #FRC=150
N150         X0
N160  G302            #FRC=150
N170         Y0
N180  G302            #FRC=150
N190         X40
```



Die Fase bzw. der Radius wird immer in der Ebene eingefügt, in der der zweite Bewegungssatz programmiert ist.

Beispiel A:

```
N100  G18  X20
N110  G19
N120  G301 I5
N130  Y20 Z20
```

Beispiel B:

```
N100  G18  X20
N110  G301 I5
N120  G19
N130  Y20 Z20
```

Beispiel A und B liefern das gleiche Ergebnis, Überschleifen in der Y-Z-Ebene.

Versionshinweis

Ab V3.1.3057.04 stehen weitere Möglichkeiten zur Programmierung von Fasen und Radien zur Verfügung:

Erweiterte G-Funktionen G301 und G302:

Fasen und Radien werden als Zusatzwert in Kombination mit G301 bzw. G302 programmiert. Der Fasen- bzw. Radiuswert in [mm, inch] muss immer mit angegeben werden, er ist nicht haltend.

G301 bzw. G302 können direkt im ersten Verfahrsatz mit programmiert werden, ein eigener NC-Satz ist nicht erforderlich.

Syntax:

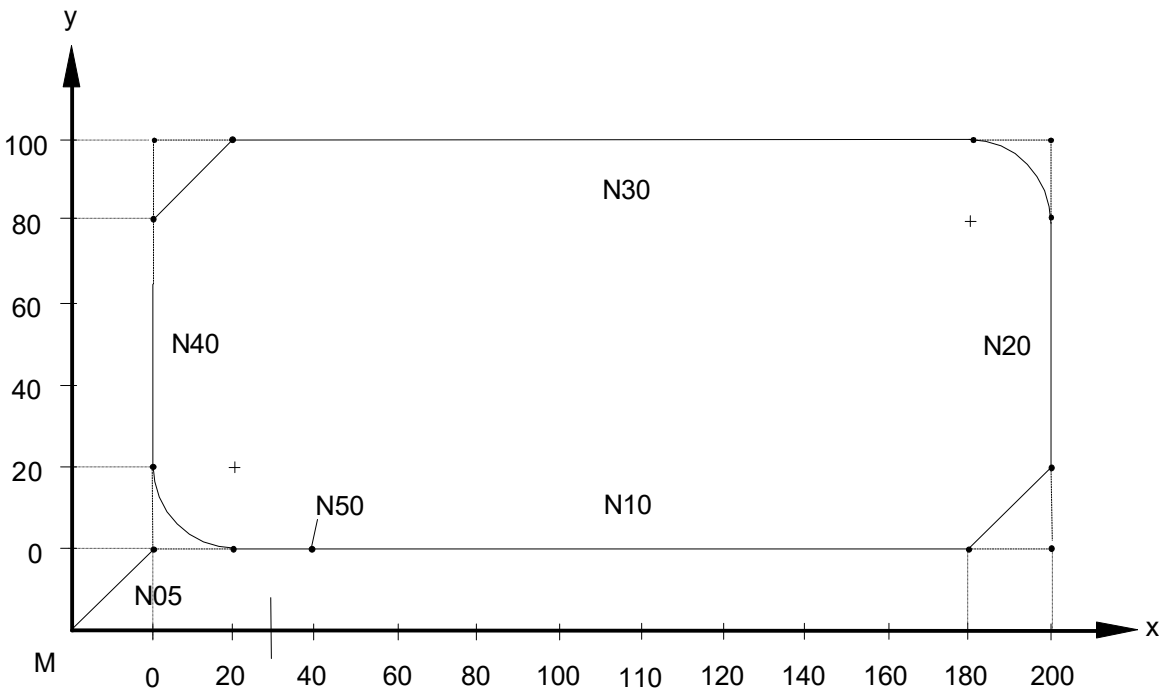
G301=..	Einfügen von Fasen mit Angabe der Fase in [mm, inch]	nicht modal
G302=..	Einfügen von Radien mit Angabe des Radius in [mm, inch]	nicht modal

Programmierbeispiel

Einfügen von Fasen und Radien (G301/G302)

Rechteckkontur mit 2 Fasen (G301) und 2 Radien (G302) mit spezifischen Vorschüben

```
N05  G17 G00 G90 X0 Y0
N10  G01 F2000 X100       G301=20 #FRC=500
N20                 Y100  G302=20 #FRC=1000
N30           X0          G301=20 #FRC=500
N40                 Y0    G302=20 #FRC=1000
N50  X40
N50  M30
```



Programmierung von Fasen und Radien mit #-Befehlen:

Fasen und Radien werden als Zusatzwert in Kombination mit spezifischen #-Befehlen programmiert. Der Fasen- bzw. Radiuswert in [mm, inch] muss immer mit angegeben werden, er ist nicht haltend.

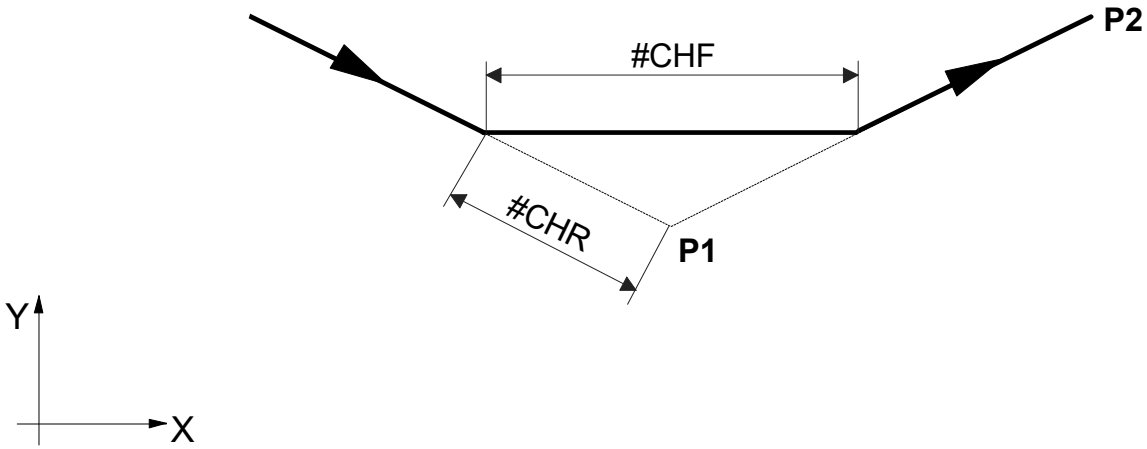
Die #-Befehle können direkt im ersten Verfahrsatz mit programmiert werden, ein eigener NC-Satz ist nicht erforderlich.

Eine Fase kann auf zwei Arten programmiert werden, entweder durch Vorgabe

- der Fasenbreite (analog zu G301) oder
- der Fasenlänge.

Syntax:

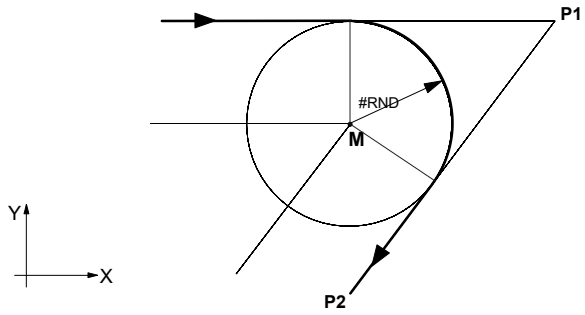
#CHR=..	Angabe der Fasenbreite in [mm, inch]	nicht modal
#CHF=..	Angabe der Fasenlänge in [mm, inch]	nicht modal



Die Radienprogrammierung erfolgt mit:

Syntax:

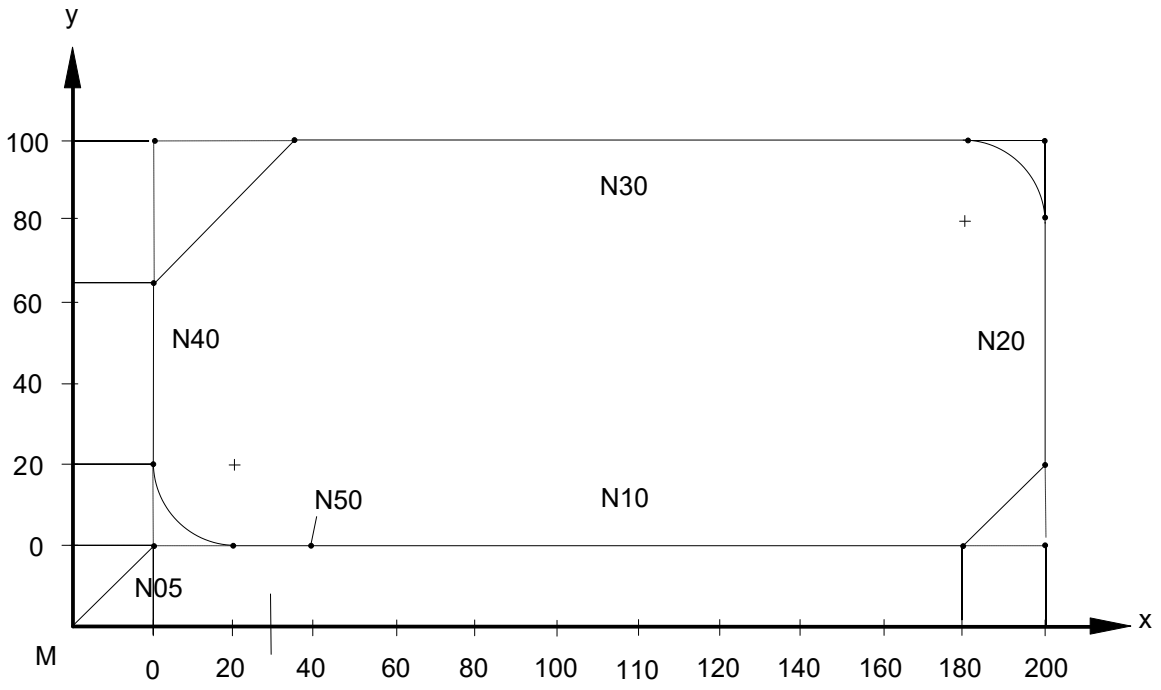
#RND=..	Angabe des Radius in [mm, inch]	nicht modal



Programmierbeispiel

Rechteckkontur mit 2 Fasen (#CHR, #CHF) und 2 Radien (#RND) mit spezifischen Vorschüben

```
N05  G17 G00 G90 X0 Y0
N10  G01 F2000 X200        #CHR=20 #FRC=500
N20                 Y100   #RND=20 #FRC=1000
N30            X0          #CHF=35 #FRC=500
N40                 Y0     #RND=20 #FRC=1000
N50            X40
N60  M30
```


y
100
80
60
40
20
0
M
N30
N40
N20
N50
N10
N05
x
0
20
40
60
80
100
110
120
140
160
180
200

4.21.1 Einfügen von Fasen am Beispiel G301

Bei G301 definiert das I-Wort den Abstand zwischen dem Eckpunkt der programmierten Kontur und dem jeweiligen Schnittpunkt der eingefügten Geraden mit den Konturelementen. Ist ein oder sind beide Konturelemente Kreisbögen, so wird dieser Abstand als Sehnenlänge betrachtet.

Ergibt sich durch die angegebene Größe des Einfügeradius bzw. der Fase auf einem oder beiden Konturelementen eine Richtungsumkehr, so erfolgt eine Fehlermeldung.

```
:
N10 G91 G01 X80 Y-40 F100          ;P1
N20 G301 I40
N30 G01 X80 Y40                    ;P2
:
```

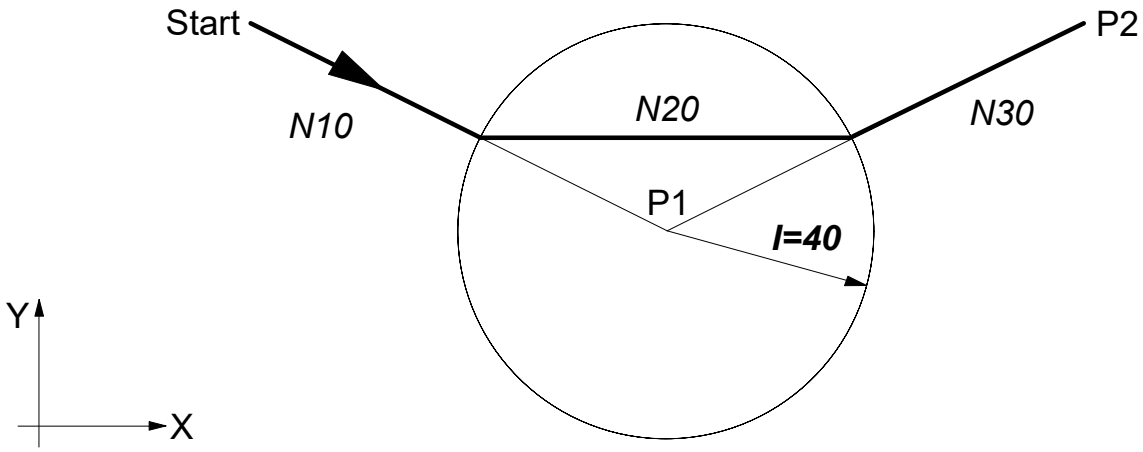


Abb. 55: Einfügen einer Fase zwischen zwei Geraden

```
:
N10 G91 G03 I50 X95 Y-15       ;P1
F100
N20 G301 I30
N30 G03 X80 Y-5 I40 J15        ;P2
:
```

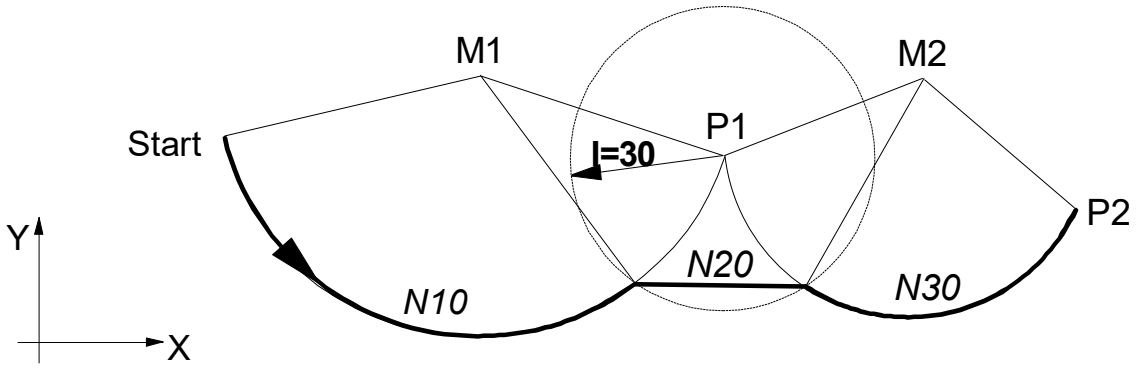


Abb. 56: Einfügen einer Fase zwischen zwei Kreisbögen

```
:
N10 G01 X20 Y-10 F100              ;P1
N20 X20                            ;P2
N30 G301 I30
N40 X60 Y50                        ;P3
:
```

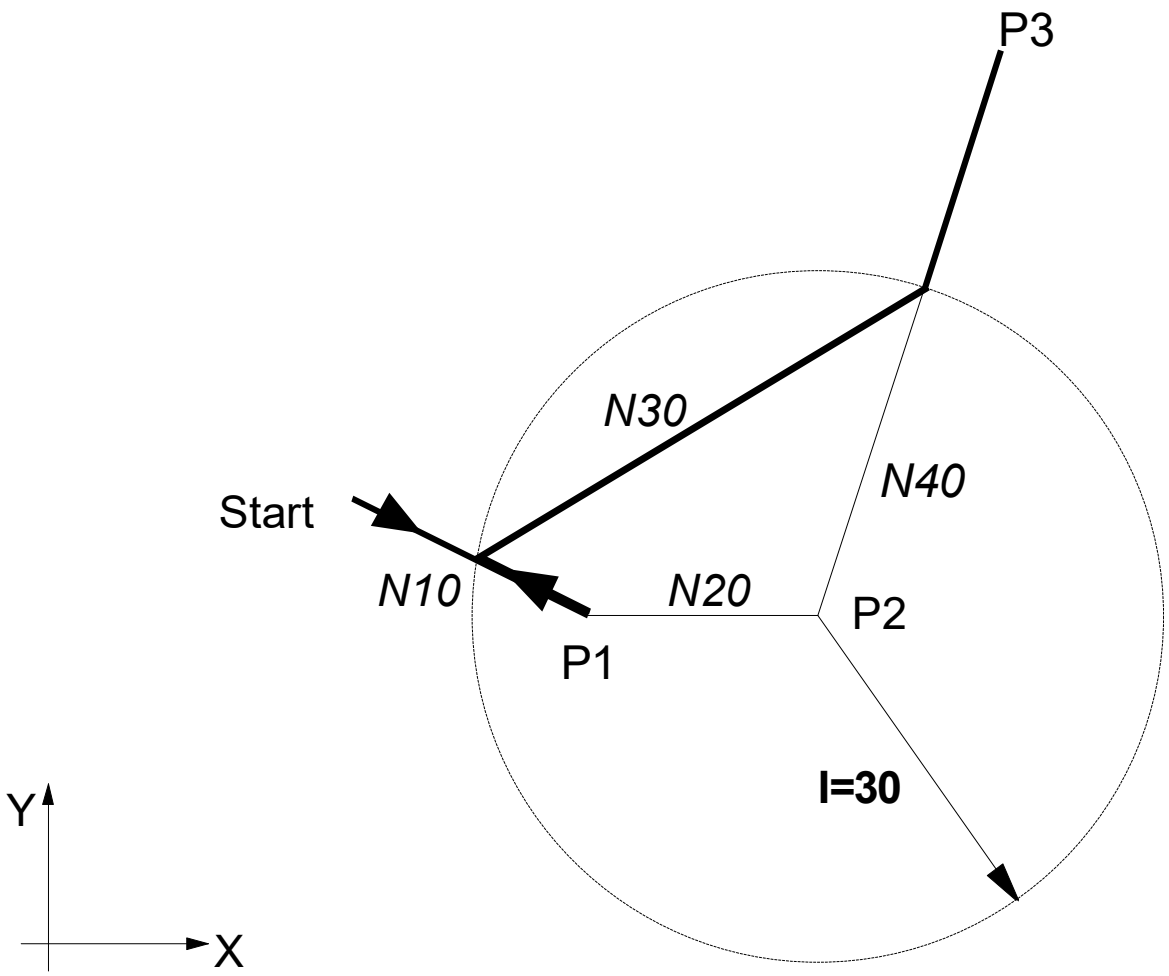


Abb. 57: Fehlerfall wegen Richtungsumkehr

4.21.2 Einfügen von Radien am Beispiel G302

Bei G302 definiert das I-Wort den Radius des eingefügten Kreisbogens. Dessen Mittelpunkt ist der Schnittpunkt der beiden Äquidistanten mit dem Abstand I zur programmierten Bahn. Die Lage der Äquidistanten wird so gewählt, dass die programmierte Kontur beim Überschleifen soweit wie möglich erhalten bleibt.

Hinweis

Bei tangentialen Satzübergängen kann kein Überschleifradius eingefügt werden.

```
:
N10 G91 G01 X60 F100                ;P1
N20 G302 I30
N30 X-40 Y-55                       ;P2
:
```

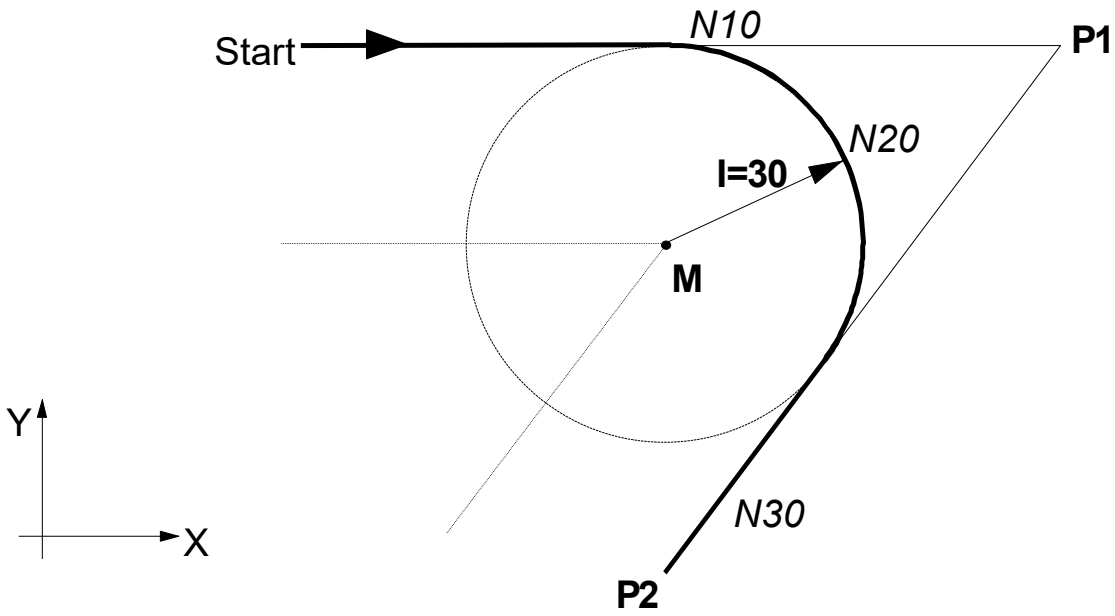


Abb. 58: Einfügen eines Kreisbogens zwischen zwei Geraden

```
:
N10 G91 G02 X80 I40 F100        ;P1
N20 G302 I40
N30 G02 X50 I25 J-15            ;P2
:
```

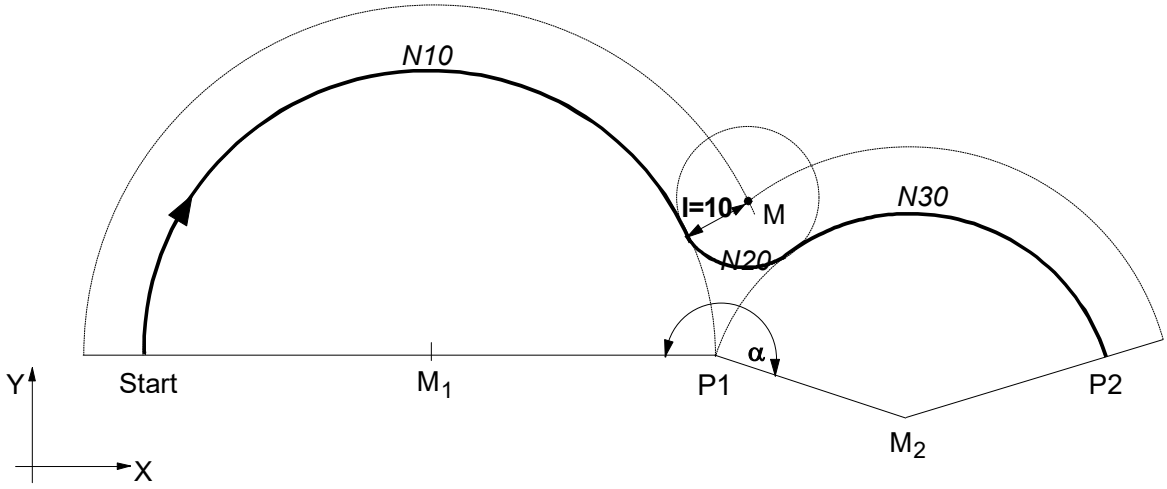


Abb. 59: Einfügen eines Kreisbogens zwischen zwei Kreisen (Winkel α ≥ 180°)

```
:
N10 G91 G02 X80 I40 F100      ;P1
N20 G302 I5
N30 G02 X60 Y35 I20 J35       ;P2
:
```

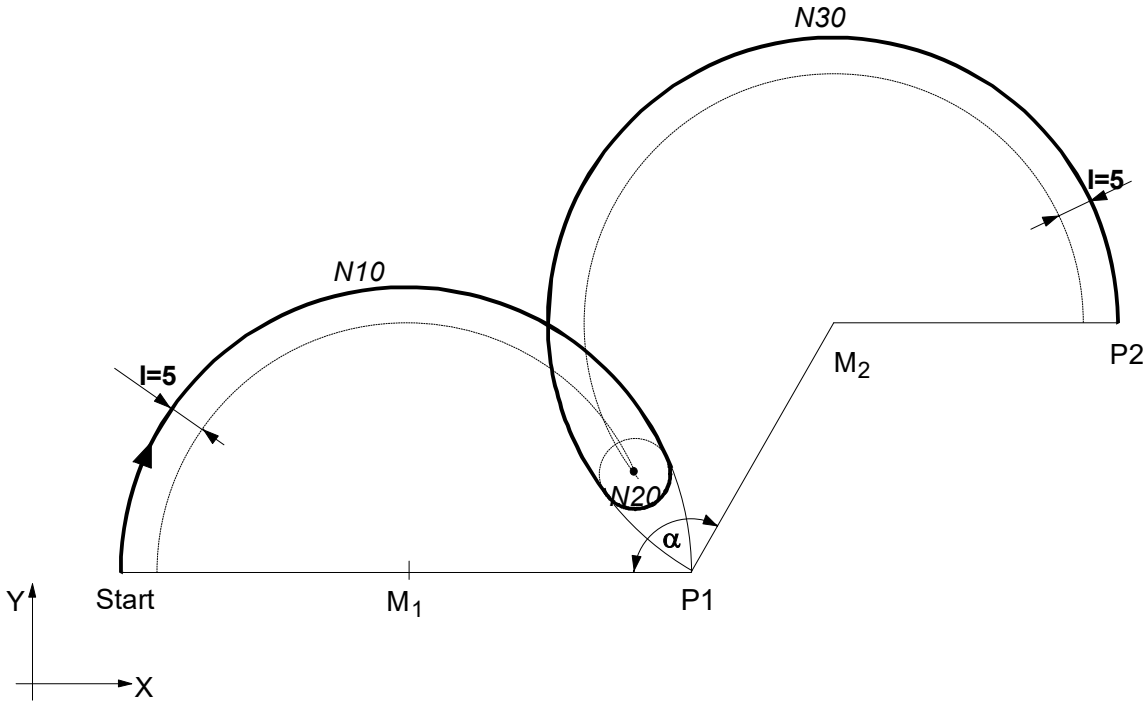


Abb. 60: Einfügen eines Kreisbogens zwischen zwei Kreisen (Winkel α < 180°)

4.22 Handbetrieb

Der Handbetrieb (HB) ermöglicht ein externes Ansteuern einzelner Achsen mit physikalischen Handbetriebselementen (Handrad, Tipptasten, Joystick). Der Bediener kann während laufender Interpolation, d.h. während dem Ablauf des NC-Programms im HB-Modus, Achsen mit zusätzlichen Sollwerten beaufschlagen. Folgende Handbetriebsarten stehen zur Verfügung:

Handradfunktion:	beliebiger Weg mit beliebiger Geschwindigkeit
Kontinuierlicher Jogbetrieb:	beliebiger Weg mit parametrierbarer Geschwindigkeit
Schrittweiser Jogbetrieb:	vorgebbarer Weg mit parametrierbarer Geschwindigkeit
Unterbrechbarer Jogbetrieb:	wie schrittweiser Jogbetrieb, jedoch Fahrtunterbrechung möglich

Diese Handbetriebsarten können von der Bedienoberfläche aus aktiviert werden. Die entsprechenden Parameter wie zum Beispiel Auflösungsstufen, Geschwindigkeiten, Schrittweiten etc. werden durch entsprechende NC-Befehle ("#"-Befehle, "G"-Befehle) programmiert.

Die aktuelle Handbetriebsart und die mit einem physikalischen Handbetriebselement bewegte Achse kann jederzeit geändert werden. Ein physikalisches Handbetriebselement kann gleichzeitig mehrere Achsen in verschiedenen Kanälen mit Sollwerten beaufschlagen. Eine Achse kann **nur in einer** Handbetriebsart und mit einem Betriebselement betrieben werden.

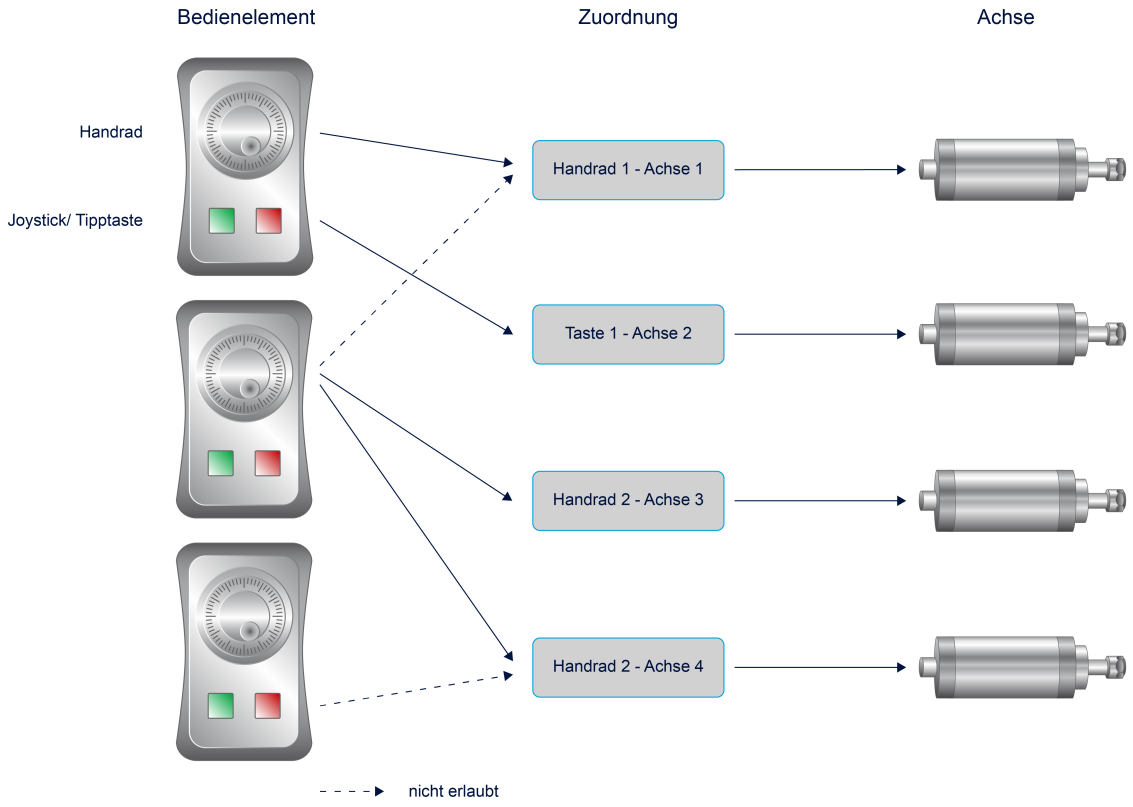


Abb. 61: Der Handbetrieb und seine Möglichkeiten

4.22.1 An-/Abwahl Handbetrieb mit paralleler Interpolation (G201/G202)

Für die programmierten Bahnachsen wird die Schnittstelle zwischen Interpolator und Handbetrieb aktiviert. Spindeln können nicht programmiert werden.

Versionshinweis

Ab Version **V2.11.2010.02** ersetzt die Angabe der Achsen **X.. Y..** den Befehl **#ACHSE [...]**. Dieser ist aus Kompatibilitätsgründen weiterhin verfügbar, es wird aber empfohlen, diesen in neuen NC-Programmen nicht mehr zu verwenden.

Mit G201 wird für bestimmte Achsen der Handbetrieb angewählt. Danach ist für diese Achsen der Handbetrieb mit paralleler Interpolation bis zur Abwahl mit G202 aktiv.

Syntax:

G201 <*Achsname*>**..** {<*Achsname*>**..**} modal

<*Achsname*>**..** Anwahl Handbetrieb für bestimmte Achsen. Der Koordinatenwert ist nur aus syntaktischen Gründen erforderlich, ansonsten jedoch nicht relevant. Bei der Anwahl müssen **immer** Achsen angegeben werden.

Mit G202 wird für alle oder bestimmte Achsen der Handbetrieb abgewählt.

G202 Abwahl des Handbetriebs für **alle** Achsen modal, Grundzustand

... oder für bestimmte Achsen

G202 <Achsname>.. {<Achsname>..}

<*Achsname*>**..** Abwahl Handbetrieb für bestimmte Achsen. Der Koordinatenwert ist nur aus syntaktischen Gründen erforderlich, ansonsten jedoch nicht relevant.

Nach G202 werden die während G201 im Interpolator aufsummierten Handbetriebsoffsets **nicht** gelöscht. Dies erfolgt entweder

- beim nächsten impliziten Positionsabgleich im Kanal (z.B. ausgelöst durch Achstausch, Trafoanwahl, …) oder
- durch eine explizite Sollwertanforderung mit #CHANNEL INIT[CMDPOS].

Programmierbeispiel

An-/Abwahl mit paralleler Interpolation(G201/G202)

```
.....
G00 X100 Y100
;Zustandsübergang X/Y-Achsen in den Handbetrieb
G201 X1 Y1
P1 = 0
$WHILE P1 == 0
;Einrichten X/Y-Achsen im Handbetrieb
;Programmweiterlauf mit Setzen von P1 auf 1 durch Bedienung
$ENDWHILE
;Zustandsübergang aller Achsen in Normalbetrieb
G202
....
;Optional: Anfordern Sollpositionen, löschen Handbetriebsoffsets
#CHANNEL INIT[CMDPOS]

G01 Y200 F500
.....
```

4.22.2 Anwahl Handbetrieb ohne parallele Interpolation (G200)

Für die programmierten Bahnachsen wird die interne Handbetriebsschnittstelle des Interpolators aktiviert. Spindeln können nicht programmiert werden.

Versionshinweis

Ab Version **V2.11.2010.02** ersetzt die Angabe der Achsen **X.. Y..** den Befehl **#ACHSE [...]**. Dieser ist aus Kompatibilitätsgründen weiterhin verfügbar, es wird aber empfohlen, diesen in neuen NC-Programmen nicht mehr zu verwenden.

Der Handbetrieb **ohne** parallele Interpolation wird mit G200 angewählt.

Syntax:

G200	Handbetrieb für **alle** Achsen	nicht modal
... oder für bestimmte Achsen		
G200 <Achsname>**..** {<Achsname>**..**}		nicht modal

*<Achsname>***..** Anwahl Handbetrieb für bestimmte Achsen. Der Koordinatenwert ist nur aus syntaktischen Gründen erforderlich, ansonsten jedoch nicht relevant

Bei programmiertem G200 erfolgt die Unterbrechung der Bearbeitung des aktuellen NC-Programms im Interpolator. Die Handbetriebsarten können dann aktiviert, umgeschaltet und deaktiviert werden. Nach Verfahren der Achsen im Handbetrieb kann die Fortsetzung des NC-Programms über die Bedienung durch die Steueranweisung "Bewegung Fortsetzen" an den Interpolator beauftragt werden.

In diesem Betriebsmodus werden die Offsetgrenzen (maximaler Verfahrbereich im Handbetrieb) automatisch auf die Softwareendschalterpositionen gesetzt, sodass der gesamte Bereich zwischen den Softwareendschaltern im Handbetrieb verfahren werden kann. Nach Beenden dieses Handbetriebsmodus sind die vorherigen relativen Offsetgrenzen wieder gültig.

Eine parallele Interpolation der Achsen während dem Handbetrieb ist bei G200 nicht möglich. Nach der Bedienhandlung "*Bewegung Fortsetzen*" werden alle Achsen und Betriebsarten deaktiviert, so dass kein Handbetrieb mehr möglich ist.

Der Decoder fordert nach der Deaktivierung von G200 alle Handbetrieboffsets und Sollpositionen vom Interpolator an und sendet die aktuellen Sollpositionen an alle Teilnehmer im NC-Kanal. Die Handbetrieboffsets werden im Decoder in den Variablen V.A.MANUAL_OFFSETS abgelegt (s. a. Kap. Achsspezifische Variablen (V.A.) [▶ 594] und können so im NC-Programm angesprochen werden. Die Handbetrieboffsets im Interpolator werden gelöscht.

Wird G200 ohne Achsangabe programmiert, so wirkt dieser Befehl auf alle momentan vorhandenen Bahnachsen.

Programmierbeispiel

Anwahl ohne parallele Interpolation (G200)

```
.....
G00 X100 Y100
G200 X1 (Handbetrieb für X-Achse)
G01 X200 Y200 F600
G01 Y200 F500
G200 (Handbetrieb für alle Achsen)
G01 Y500 Z500
.....
```

4.22.3 Verhalten bei Programmende (M02, M30)

Nach NC-Programmende darf im Handbetrieb kein Verfahren der Achsen möglich sein. Deshalb wird der Befehl "M30" bzw. "M02" so behandelt, als ob zusätzlich "G202" programmiert worden wäre.

4.22.4 Parametrierung der Betriebsarten

Die #-Befehle zur Parametrierung dürfen nur bei **abgewähltem** Handbetrieb (G202) für die jeweilige Achse programmiert werden.

4.22.4.1 Betriebsart Handradbetrieb (#HANDWHEEL)

Versionshinweis

Ab Version **V2.11.2010.02** ersetzt der Befehl **#HANDWHEEL [...]** den Befehl **#SET HR [...]**. Dieser ist aus Kompatibilitätsgründen weiterhin verfügbar, es wird aber empfohlen, diesen in neuen NC-Programmen nicht mehr zu verwenden.

Syntax:

#HANDWHEEL [AX=<*Achsname*> | **AXNR**=.. **RES1**=.. [**RES2**=.. **RES3**=..] **]**

AX=<*Achsname*>	Name der Handbetriebsachse
AXNR=..	Logische Nummer der Handbetriebsachse, Positive Ganzzahl.
RES1=..,	Auflösungsstufen (maximal 3),
RES2=..,	[mm/Handradumdrehung, inch/Handradumdrehung, °/Handradumdrehung]
RES3=..	

Programmierbeispiel

Betriebsart Handradbetrieb

```
Mit Angabe des Achsnamens:
...
G202 X1
#HANDWHEEL [AX=X RES1=0.1 RES2=0.2 RES3=0.5]
...
G201 X1
...

..oder mit Angabe der logischen Achsnummer:
...
G202 X1
#HANDWHEEL [AXNR=1 RES1=0.1 RES2=0.2 RES3=0.5]
...
G201 X1
...
```

4.22.4.2 Betriebsart kontinuierlicher Jogbetrieb (#JOG CONT)

Versionshinweis

Ab Version **V2.11.2010.02** ersetzt der Befehl **#JOG CONT [...]** den Befehl **#SET TIP [...]**. Dieser ist aus Kompatibilitätsgründen weiterhin verfügbar, es wird aber empfohlen, diesen in neuen NC-Programmen nicht mehr zu verwenden.

Syntax:

#JOG CONT [AX=<*Achsname*>** | AXNR**=.. **FEED1**=.. [**FEED2**=.. **FEED3**=..] **]**

AX=<*Achsname*>	Name der Handbetriebsachse
AXNR=..	Logische Nummer der Handbetriebsachse, Positive Ganzzahl.
FEED1=..,	
FEED2=..,	Geschwindigkeitsstufen (maximal 3) in [mm/min, m/min, inch/min, °/min]
FEED3=..	

Programmierbeispiel

Betriebsart kontinuierlicher Jogbetrieb

```
Mit Angabe des Achsnamens:
G202 X1
...
#JOG CONT [AX=X FEED1=1.0 FEED2=1.5 FEED3=2.0]
...
G201 X1
...

..oder mit Angabe der logischen Achsnummer:
G202 X1
...
#JOG CONT [AXNR=1 FEED1=1.0 FEED2=1.5 FEED3=2.0]
...
G201 X1
...
```

4.22.4.3 Betriebsart schrittweiser bzw. unterbrechbarer Jogbetrieb (#JOG INCR)

Versionshinweis

Ab Version **V2.11.2010.02** ersetzt der Befehl **#JOG INCR [...]** den Befehl **#SET JOG [...]**. Dieser ist aus Kompatibilitätsgründen weiterhin verfügbar, es wird aber empfohlen, diesen in neuen NC-Programmen nicht mehr zu verwenden.

Syntax:

#JOG INCR [AX=<*Achsname*>** | AXNR**=.. **DIST1**=.. **FEED1=..** [**DIST2**=.. **FEED2=.. DIST3**=.. **FEED3**=..] **]**

AX=<*Achsname*>	Name der Handbetriebsachse
AXNR=..	Logische Nummer der Handbetriebsachse, Positive Ganzzahl.
DIST1=..	
FEED1=..,	Paarstufen (Schrittweite;Geschwindigkeit) (maximal 3) in
DIST2=..	[mm;mm/min, mm;m/min, inch;inch/min, °;°/min]
FEED2=..,	
DIST3=..	
FEED3=..	

Programmierbeispiel

Betriebsart schrittweiser bzw. unterbrechbarer Jogbetrieb

```
Mit Angabe des Achsnamens:
...
G202 X1
#JOG INCR [AX=X DIST1=0.1 FEED1=1.0 DIST2=0.2 FEED2=1.5 DIST3=0.5
FEED3=2.0]
...
G201 X1
...

..oder mit Angabe der logischen Achsnummer:
...
G202 X1
#JOG INCR [AXNR=1 DIST1=0.1 FEED1=1.0 DIST2=0.2 FEED2=1.5 DIST3=0.5
FEED3=2.0]
...
G201 X1
...
```

4.22.5 Vorgabe der Offsetgrenzen (#MANUAL LIMITS)

Versionshinweis

Ab Version **V2.11.2010.02** ersetzt der Befehl **#MANUAL LIMITS [...]** den Befehl **#SET OFFSET [...]**. Dieser ist aus Kompatibilitätsgründen weiterhin verfügbar, es wird aber empfohlen, diesen in neuen NC-Programmen nicht mehr zu verwenden.

Syntax:

#MANUAL LIMITS [AX=<*Achsname*> | **AXNR**=.. **NEGATIVE=.. POSITIVE**=.. **]**

AX=<*Achsname*>	Name der Achse, für die die Offsetgrenzen gelten sollen.
AXNR=..	Logische Nummer der Achse, für die die Offsetgrenzen gelten sollen, Positive Ganzzahl
NEGATIVE=..	Negativer relativer Offsetwert. Muss ≤ 0 programmiert werden, in [mm, inch]
POSITIVE=..	Positiver relativer Offsetwert. Muss ≥ 0 programmiert werden, in [mm, inch]

Mit diesem Befehl können positive und negative Grenzen für die zulässige relative Verfahrbewegung im G201/G202 - Handbetrieb für die jeweilige Bahnachse festgelegt werden. Hierbei beziehen sich die relativen negativen und positiven Offsetgrenzen auf die Startposition zum Zeitpunkt der Anwahl des Handbetriebes. Durch Setzen des Parameters P-CHAN-00114 werden die Offsetgrenzen auch bei G200 berücksichtigt.

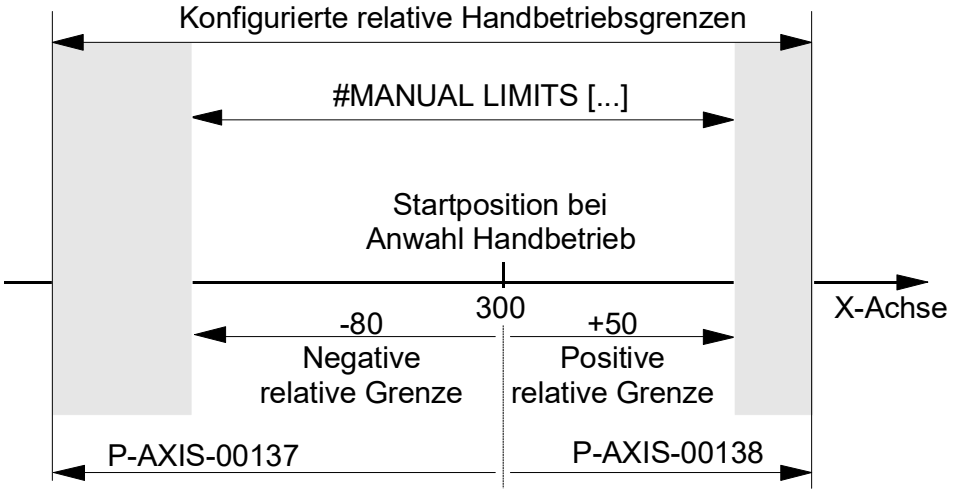
Hinweis

Die relativen Offsetgrenzen können jederzeit im NC-Programm überschrieben werden. Es erfolgt eine Vorzeichenprüfung. Relative Offsetgrenzen gelten für jede Achse im Programmierkoordinatensystem (PCS).

Programmierbeispiel

Vorgabe der Offsetgrenzen

```
N090 G01 F1000 X300
N100 #MANUAL LIMITS [AX=X NEGATIVE=-80 POSITIVE=50]
N110 G201 X1
:
```



Achsen, die als rotatorische Moduloachsen oder als Spindel betrieben werden (M3, M4, M5, M19, S...), werden bei diesem Befehl nicht berücksichtigt. Die Offsetgrenzen gelten für alle Handbetriebsarten. Sie können auch bei aktivem Handbetrieb (G201) überschrieben werden. Es erfolgt lediglich eine Vorzeichenüberprüfung der programmierten Werte. Weitere Überprüfungen z.B. bzgl. Bereichserweiterungen finden nicht statt. Die relativen Offsetgrenzen können auch über den Parameterdatensatz P-AXIS-00138 und P-AXIS-00137 der Achse vorgegeben werden.

4.22.6 Beispiel für die Parametrierung einer Achse für den Handbetrieb

Programmierbeispiel

Beispiel für die Parametrierung einer Achse für den Handbetrieb

```
%main
#HANDWHEEL [AX=X RES1=0.1 RES2=0.2 RES3=0.5]
#JOG CONT [AX=X FEED1=1.0 FEED2=1.5 FEED3=2.0]
#JOG INCR [AX=X DIST1=0.1 FEED1=1.0 DIST2=0.2 FEED2=1.5 DIST3=0.5
FEED3=2.0]
#MANUAL LIMITS [AX=X NEGATIVE=-5 POSITIVE=5]
...
G201 X1
...
G202 X1
...
```

Mit dem Befehl

#HANDWHEEL [AX=X RES1=0.1 RES2=0.2 RES3=0.5]

werden die Handradauflösungsstufen der X-Achse angegeben. Die Auflösungsstufen sind 0.1, 0.2 und 0.5 mm/Handradumdrehung.

Mit dem Befehl

#JOG CONT [AX=X FEED1=1.0 FEED2=1.5 FEED3=2.0]

werden die Verfahrgeschwindigkeiten für den kontinuierlichen Jogbetrieb (auch Tippbetrieb) der X-Achse vorgegeben. Die Geschwindigkeitsstufen sind 1.0, 1.5 und 2.0 mm/min.

Mit dem Befehl

#JOG INCR [AX=X DIST1=0.1 FEED1=1.0 DIST2=0.2 FEED2=1.5 DIST3=0.5 FEED3=2.0]

wird der schrittweise/unterbrechbare Jogbetrieb für die X-Achse parametriert. Die Schrittweiten und Geschwindigkeiten werden paarweise programmiert:

Stufe 1:	Schrittweite 0.1 mm,	Geschwindigkeit 1.0 mm/min
Stufe 2:	Schrittweite 0.2 mm,	Geschwindigkeit 1.5 mm/min
Stufe 3:	Schrittweite 0.5 mm,	Geschwindigkeit 2.0 mm/min

Mit dem Befehl

#MANUAL LIMITS [AX=X NEGATIVE=-5 POSITIVE=5]

wird für den Handbetrieb die negative Offsetgrenze auf – 5 mm, die positive Offsetgrenze auf +5 mm gesetzt.

Mit nachfolgenden Befehlen wird für die X-Achse der Handbetrieb aktiviert bzw. deaktiviert. Wird G202 allein programmiert, so wird für alle Achsen der Handbetrieb deaktiviert.

G201 X1

G202 X1

G202

4.22.7 #ECS in Verbindung mit Handbetrieb

Ausgangslage ist die Verwendung eines gedrehtes PCS. In diesem Zustand soll das Werkzeug in Werkzeugrichtung im Handbetrieb bewegt werden.

Im Standardfall erfolgt die Aktivierung des Handbetriebs über die SPS mit dem Aktivierungstring „G200“.

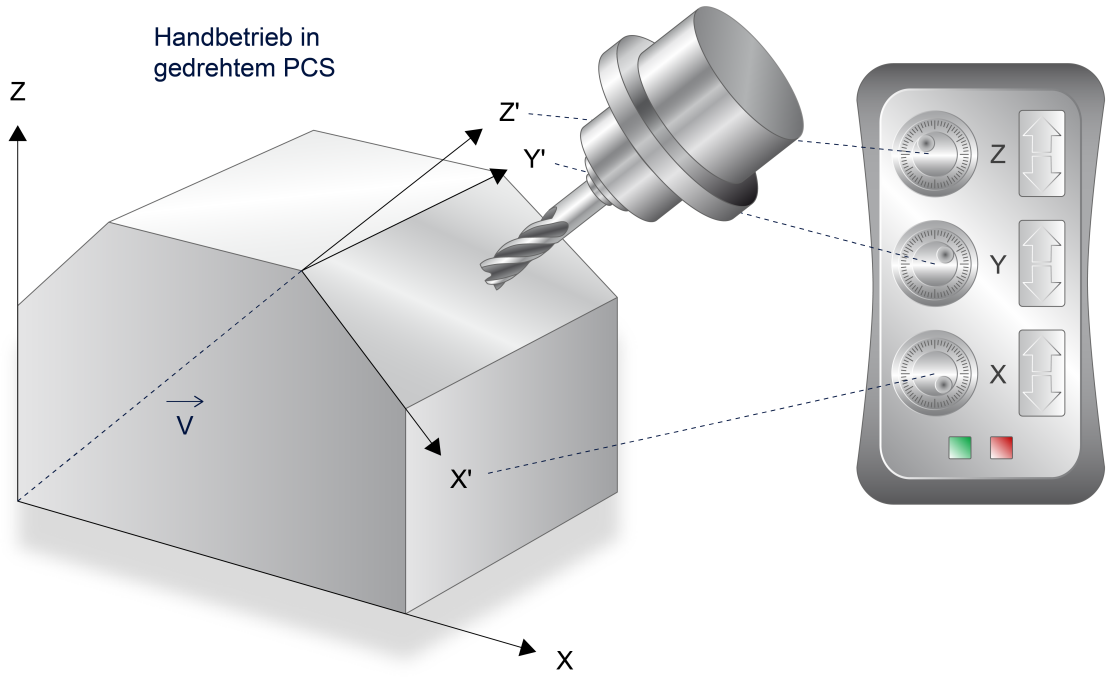


Abb. 62: Handbetrieb bei gedrehtem PCS

Für die Aktivierung im vorliegenden Fall ist eine erweiterte Sequenz in der SPS erforderlich. Diese muss folgendes enthalten:

- Aktivierung der verwendeten Kinematik ID(#KIN ID[<ID>]), alternativ ist die Kinematik-ID im Werkzeug hinterlegt
- Aktivierung des Effektorkoordinatensystems (#ECS ON)
- Aktivierung des Handbetriebs (G200)
- Aktivierung der Kontrollelemente der 3 beteiligten Achsen
- Bewegen der Z-Achse (3. Hauptachse) im Tipp- oder Jogbetrieb

Erweiterte Stringsequenz

```
#KIN ID[< kinematic ID >]$R§N (oder D <tool ID>$R§N)
#ECS ON $R§N
G200
```

Hinweis

Die Sequenz muss zwingend mit G200 abgeschlossen werden.
$R§N : Zeichenfolge für Zeilenumbruch in IEC61131-3.

4.23 Anfordern von Offset-, Soll- und Istwerten

Die im Folgenden beschriebenen NC-Befehle lösen Aufträge zur Übernahme von Handbetriebsoffset-, Soll- bzw. Istwerten des Interpolators in die Arbeitsdaten des NC-Programminterpreters aus. Auf die Daten kann dann mittels Variablen im NC-Programm zugegriffen werden.

Es können nur Offset-, Soll- und Istwerte von Bahnachsen angefordert werden. Achsen, die als rotatorische Moduloachsen oder als Spindel betrieben werden (M3, M4, M5, M19, S...), werden bei den folgenden Befehlen nicht berücksichtigt.

Die Synchronisierung zwischen NC-Programminterpreter und Interpolator ist bei allen beschriebenen NC-Befehlen gleich. Erst nach Übernahme aller Daten in die Arbeitsdaten des NC-Programminterpreters ist der Auftrag abgeschlossen und die Decodierung des NC-Programms wird fortgesetzt.

Befehlsübersicht und Eigenschaften:

Befehl	Kanal-initialisierung	Hand-betrieb-offsets im IPO	Aktualisierung von …		
			V.A.MANUAL_OFFSETS	V.A.ABS	P-Parameter, V.S./ V.P./ V.L.
#GET MANUAL OFFSETS	nein	bleiben erhalten	ja	nein	nein
#CHANNEL INIT [CMDPOS]	ja	werden gelöscht	nein	ja	nein
#CHANNEL INIT [ACTPOS]	ja	werden gelöscht	nein	ja	nein
#GET CMDPOS	nein	bleiben erhalten	nein	nein	ja
#GET ACTPOS	nein	bleiben erhalten	nein	nein	ja

4.23.1 Anfordern aktueller Handbetriebsoffsetwerte und Ablegen in "V.A.MANUAL_OFFSETS[ ]" (#GET MANUAL OFFSETS)

Versionshinweis

Ab Version **V2.11.2010.02** ersetzt der Befehl **#GET MANUAL OFFSETS** den Befehl **#GET IPO OFFSET**. Dieser ist aus Kompatibilitätsgründen weiterhin verfügbar, es wird aber empfohlen, diesen in neuen NC-Programmen nicht mehr zu verwenden.

Syntax:

#GET MANUAL OFFSETS

Dieser Befehl ist sinnvoll in Kombination mit G201/G202 (Handbetrieb mit paralleler Interpolation). Der NC-Programminterpreter fordert die aktuellen Handbetrieboffsets aller ihm zugeordneten Bahnachsen vom Interpolator an. Nach Bereitstellung der Werte werden diese im NC-Programminterpreter in den Variablen V.A.MANUAL_OFFSETS abgelegt (s. a. Kap. Achsspezifische Variablen (V.A.) [▶ 594]) und können so im NC-Programm angesprochen werden. Es erfolgt **keine** implizite Positionsinitialisierung des NC-Kanals. Die Handbetrieboffsets im Interpolator werden nicht gelöscht. Die Variablen V.A.ABS (s. a. Kap. Achsspezifische Variablen (V.A.) [▶ 594]) werden nicht aktualisiert.

Programmierbeispiel

#GET MANUAL OFFSETS

```
X100
G201
..... Verfahren im Handbetrieb
G202
#GET MANUAL OFFSETS
G01 X[100 + V.A.MANUAL_OFFSETS.X] F500 (Positionsinitialisierung des)
                                       (NC-Kanals nach Abwahl)
                                       (Handbetrieb)
```

4.23.2 Anfordern aktueller Sollpositionen und Ablegen in "V.A.ABS[ ]" (#CHANNEL INIT)

Versionshinweis

Ab Version **V2.10.1504** ersetzt der Befehl **#CHANNEL INIT [CMDPOS]** den Befehl **#SET DEC LR SOLL**. Dieser ist aus Kompatibilitätsgründen weiterhin verfügbar, es wird aber empfohlen, diesen in neuen NC-Programmen nicht mehr zu verwenden.

Syntax:

#CHANNEL INIT [CMDPOS { **AX**=*<Achsname>* | **AXNR**=.. } **]**

AX=*<Achsname>*	Name der Achse, für die der Sollwert angefordert wird.
AXNR=..	Logische Nummer der Achse, für die der Sollwert angefordert wird, Positive Ganzzahl.

Der NC-Programminterpreter fordert für alle oder bestimmte Bahnachsen* die aktuellen Sollpositionen beim Interpolator an, legt diese in den Arbeitsdaten ab und initialisiert mit diesen Positionen den NC-Kanal. Auf die aktuellen Sollpositionen in den Arbeitsdaten kann dann mittels Variablenprogrammierung V.A.ABS (s. a. Kap. Achsspezifische Variablen (V.A.) [▶ 594]) zugegriffen werden. Eventuell vorhandene Handbetriebsoffsets im Interpolator werden automatisch gelöscht. Die Achspositionen entsprechen denjenigen für Offset Null. Die in den Variablen V.A.MANUAL_OFFSETS abgelegten Werte (s. a. Kap. Achsspezifische Variablen (V.A.) [▶ 594]) werden nicht aktualisiert.

Versionshinweis

***Die achsspezifische Anforderung von Sollpositionen ist ab V2.11.2038.03 verfügbar.**

Hinweis

Sind **keine** Achsen programmiert, so werden für **alle** im Kanal vorhandenen Bahnachsen die Sollpositionen angefordert.

Sind Achsen programmiert, so werden nur für diese die Sollpositionen angefordert.

Programmierbeispiel

#CHANNEL INIT [CMDPOS …]

```
%channel_init_cmd
G01 F1000 X100 Y200
G201
..... Verfahren im Handbetrieb
G202
#CHANNEL INIT [CMDPOS AX=X AX=Y ]      ;Sollwertanforderung mit
                                       ;Achsangabe über Name..
#CHANNEL INIT [CMDPOS AXNR=1 AXNR=2 ]  ;..oder log. Achsnummer..
#CHANNEL INIT [CMDPOS]            ;..oder für alle Bahnachsen
#MSG ["Cmdpos X:%F, Y:%F ",V.A.ABS.X, V.A.ABS.Y]
M30
```

4.23.3 Anfordern aktueller Istpositionen und Ablegen in "V.A.ABS[ ]" (#CHANNEL INIT)

Syntax:

#CHANNEL INIT [ACTPOS { **AX**=<*Achsname*> | **AXNR**=.. } **]**

AX=<*Achsname*>	Name der Achse, für die der Istwert angefordert wird.
AXNR=..	Logische Nummer der Achse, für die der Istwert angefordert wird, Positive Ganzzahl.

Der NC-Programminterpreter fordert für alle oder bestimmte Bahnachsen die aktuellen Istpositionen beim Interpolator an, legt diese in den Arbeitsdaten ab und initialisiert mit diesen Positionen den NC-Kanal. Auf die aktuellen Istpositionen in den Arbeitsdaten kann dann mittels Variablenprogrammierung V.A.ABS (s. a. Kap. Achsspezifische Variablen (V.A.) [▶ 594]) zugegriffen werden. Eventuell vorhandene Handbetriebsoffsets im Interpolator werden automatisch gelöscht. Die Achspositionen entsprechen denjenigen für Offset Null. Die in den Variablen V.A.MANUAL_OFFSETS abgelegten Werte (s. a. Kap. Achsspezifische Variablen (V.A.) [▶ 594]) werden nicht aktualisiert.

Hinweis

Sind **keine** Achsen programmiert, so werden für **alle** im Kanal vorhandenen Bahnachsen die Istpositionen angefordert.

Sind Achsen programmiert, so werden nur für diese die Istpositionen angefordert. Für die nicht programmierten Achsen werden die Sollpositionen angefordert.

Achtung

Durch die Übernahme der Istposition als Sollposition kann sich die Achse bewegen. Eine Übernahme in einer NC-Programmschleife kann zu einem Driften des Antriebes führen.

Programmierbeispiel

#CHANNEL INIT [ACTPOS ...]

```
%channel_init_act
G01 F1000 X100 Y200

#CHANNEL INIT [ACTPOS AX=X AX=Y ]      ;Istwertanforderung mit
                                       ;Achsangabe über Name..
#CHANNEL INIT [ACTPOS AXNR=1 AXNR=2 ]  ;..oder log. Achsnummer..
#CHANNEL INIT [ACTPOS]            ;..oder fuer alle Bahnachsen
#MSG ["Actpos X:%F, Y:%F ",V.A.ABS.X, V.A.ABS.Y]
M30
```

4.23.4 Anfordern aktueller Sollpositionen von Achsen und Speichern in Variablen oder Parametern (#GET CMDPOS)

Versionshinweis

Ab Version **V2.11.2010.02** ersetzt der Befehl **#GET CMDPOS [...]** den Befehl **#SET IPO SOLLPOS [...]**. Dieser ist aus Kompatibilitätsgründen weiterhin verfügbar, es wird aber empfohlen, diesen in neuen NC-Programmen nicht mehr zu verwenden.

Syntax:

#GET CMDPOS [V.S.<*String*> | **V.P.**<*String*> | **V.L.**<*String*> | **P..=** <*Achsname*>
{ **,V.S.** <*String*> | **V.P.**<*String*> | **V.L.**<*String*> | **P..=** <*Achsname*> } **]**

Der NC-Programminterpreter fordert vom Interpolator die aktuellen Sollpositionen der angegebenen Bahnachsen an und legt diese in den angegebenen eigendefinierten Variablen (V.S./ V.P./ V.L.) oder Parametern (P..) ab. Es erfolgt **keine** implizite Positionsinitialisierung des NC-Kanals. Eventuell vorhandene Handbetrieboffsets im Interpolator der ausgewählten Achsen werden nicht gelöscht. Die Variablen V.A.ABS und V.A.MANUAL_OFFSETS werden nicht aktualisiert.

Programmierbeispiel

#GET CMDPOS [...]

```
.....
#GET CMDPOS [P1 = X, P2 = Y, V.P.POS1 = Z, V.P.POS2 = C]
G01 XP1 YP2
G01 ZV.P.POS1 CV.P.POS2
.....
```

4.23.5 Anfordern aktueller Istpositionen von Achsen und Speichern in Variablen oder Parametern (#GET ACTPOS)

Versionshinweis

Dieser Befehl ist ab der CNC-Version V2.11.2022.05 verfügbar.

Syntax:

#GET ACTPOS [(**V.S.**<*String*> | **V.P.** <*String*> | **V.L.**<*String*> | **P..**)**=** <*Achsname*>
{ (**,V.S.** <*String*> | **V.P.** <*String*> | **V.L.**<*String*> | **P..**)**=** <*Achsname*> } **]**

Der NC-Programminterpreter fordert vom Interpolator die aktuellen Istpositionen der angegebenen Bahnachsen an und legt diese in den angegebenen eigendefinierten Variablen (V.S./ V.P./ V.L.) oder Parametern (P...) ab. Es erfolgt **keine** implizite Positionsinitialisierung des NC-Kanals. Eventuell vorhandene Handbetrieboffsets im Interpolator der ausgewählten Achsen werden nicht gelöscht. Die Variablen V.A.ABS und V.A.MANUAL_OFFSETS werden nicht aktualisiert.

Programmierbeispiel

#GET ACTPOS [...]

```
.....
#GET ACTPOS [P1 = X, P2 = Y, V.P.POS1 = Z, V.P.POS2 = C]
G01 XP1 YP2
G01 ZV.P.POS1 CV.P.POS2
.....
```

4.24 Getriebeschalten (G112)

Syntax:

G112 <*Achsname*>.. { <*Achsname*>.. } Getriebeschalten nicht modal

Mit der Funktion G112 ist es möglich, für einzelne Bahnachsen Getriebestufen zu schalten. G112 steht auch für Spindelachsen zur Verfügung, jedoch sind hier die in Kapitel Getriebeschalten (G112) [▶ 687] beschriebenen Besonderheiten zu beachten.

Bei Achsen, die kein reales Getriebe besitzen, kann diese Funktionalität zum Schalten verschiedener Achsparameter eingesetzt werden. Somit kann z.B. für schwergängige Bearbeitungen die Reglerverstärkung und die Schleppabstandsüberwachung in einem dafür separaten Parametersatz aufeinander abgestimmt und durch die dafür festgelegte "Getriebenummer" im NC-Programm angewählt werden.

Achtung

G112 darf nicht mit einer Wegbedingung im selben NC-Satz programmiert werden.

Programmierbeispiel

Getriebeschalten (G112)

```
N50 G112 X4 Y8 (X-Achse: Getriebestufe 4, Y-Achse: Getriebestufe 8)
```

4.25 Beeinflussung der Look-Ahead Funktionalität (G115/G116/G117)

Syntax:

G115=..	Allgemeine Beeinflussung der Look-Ahead Funktionalität	modal
G116 *<Achsname>..* { *<Achsname>..* }	Beeinflussung der Berechnung der Satzübergangsgeschwindigkeit	modal
G117	Rücksetzen der Look-Ahead Funktionalität	modal, Grundzustand

Überblick über die Funktionalität der Look-Ahead-Funktion

Innerhalb der Look-Ahead-Funktion sind verschiedene Einzelfunktionen realisiert, die die gefahrenen Geschwindigkeiten auf Maximalwerte so begrenzen, dass zulässige Achsgeschwindigkeiten und -beschleunigungen immer eingehalten werden können.

Folgende Einzelfunktionen beeinflussen das Geschwindigkeitsprofil:

- (1) Berechnung der maximalen Bahngeschwindigkeiten aufgrund der im Kanalparametersatz P-CHAN-00071 festgelegten maximalen Achsgeschwindigkeiten der an der Bewegung beteiligten Achsen. Die Geschwindigkeit auf der programmierten Bahn darf nur so hoch sein, dass jede an der Bewegung beteiligte Achse ihre maximale Achsgeschwindigkeit nicht überschreitet.
- (2) Berechnung der maximalen Bahngeschwindigkeiten auf gekrümmten Bahnen aufgrund eines maximalen Sehnenfehlers. Dazu kann auf jeder gekrümmten Kurve im Punkt der stärksten Krümmung ein Krümmungsradius ermittelt werden. Auf dem dadurch beschriebenen Kreis kann nun wiederum ein Kreisabschnitt definiert werden, dessen Anfangs- und Endpunkt je einem generierten Bahnpunkt entspricht. Unter der Voraussetzung, dass sich das reale mechanische System auf der zugehörigen Sehne bewegen würde, kann die zugehörige Kreisabschnittshöhe als Fehlerkriterium herangezogen werden.
- (3) Berechnung der maximalen Bahngeschwindigkeiten auf gekrümmten Bahnen aufgrund der im Kanalparametersatz P-CHAN-00071 festgelegten maximalen Achsbeschleunigungen. Die berechnete maximale Bahngeschwindigkeit aufgrund des Sekantenfehlers kann insbesondere bei kurzer Abtastzeit und kleinen Zirkularbewegungen noch zu hoch sein, so dass die auftretenden Zentripetalbeschleunigungen die zulässigen Achsbeschleunigungen überschreiten würden.
- (4) Berechnung der Bahnübergangsgeschwindigkeit aufgrund der Satzübergangsgeometrie unter Berücksichtigung der an der Bewegung beteiligten Achsen und der im Kanalparametersatz P-CHAN-00071 festgelegten Grenzwerte. Bei einem Satzübergang wechseln in der Regel die jeweiligen Anteile der einzelnen Achsen an der Gesamtbewegung. Dieser Wechsel der Achsanteile tritt aber nur im ersten Abtastzyklus auf der neuen Bahn auf (bei Linearbewegungen). Wird vorausgesetzt, dass der Übergang ohne generierte Beschleunigung (z.B. vom Slope) erfolgen soll, kann eine Abschätzung der auftretenden Achsbeschleunigungen vorgenommen werden. Damit ist die Berechnung einer maximalen Bahngeschwindigkeit am Satzübergang möglich, bei der die maximalen Achsbeschleunigungen am Übergang nicht überschritten werden.

Allgemeiner Fall:

Normalbetrieb mit Look-Ahead

Nach Programmstart sind sämtliche Funktionalitäten des Look-Ahead eingeschaltet (G117). Damit ist sichergestellt, dass die im Kanalparametersatz angegebenen Parameter P-CHAN-00071 wie z.B. maximale Achsgeschwindigkeit und maximale Achsbeschleunigung immer eingehalten werden.

Sonderfall:

Ausschalten von Einzelfunktionen der Look-Ahead-Funktion durch G115

Programmierbeispiel

Beeinflussung der Look-Ahead Funktionalität (G115/G116/G117)

```
Nnn G115 = 0        (Ausschalten der Einzelfunktionen (2),(3) und(4))
                    (der Look-Ahead-Funktion)
```

In der folgenden Tabelle sind die möglichen Kombinationen der Einzelfunktionen erläutert.

Hinweis

Die entsprechenden an- bzw. abgewählten Funktionen beziehen sich immer auf alle Achsen.

Überblick über erlaubte Identifikationsnummern (ID) im Zusammenhang mit G115:

Einzelfunktion ID	(1)	(2)	(3)	(4)
0	*			
2	*			*
4	*	*		
6	*	*		*
8	*		*	
10	*		*	*
12	*	*	*	
14	*	*	*	*

*	Einzelfunktion aktiv		Einzelfunktion nicht aktiv

Programmierbeispiel

```
Nnn G115 = 2    (Auschalten der Einzelfunktionen (2) und (3))
Nnn G115 = 12   (Auschalten der Einzelfunktion (4))
Nnn G115 = 14   (Einschalten aller Einzelfunktionen, nach)
                (Programmstart, vgl. G117)
```

Spezialfall:

Beeinflussung der Look-Ahead Funktionalität durch G116, G117

Durch **G116** wird die Berechung der Satzübergangsgeschwindigkeit (Einzelfunktion (4)) für einzelne programmierbare Achsen ausgeschaltet. Es erfolgt keine Reduzierung der Satzübergangsgeschwindigkeit (Bahngeschwindigkeit) aufgrund von "Ecken" in der Bahn.

Hinweis

Die dynamischen Achsdaten (Achsbeschleunigungen) werden im Normalfall nicht eingehalten.

Programmierbeispiel

```
N10 G116 X1 Y2              (Keine Reduzierung der Satzübergangs-)
                            (geschwindigkeit unterschiedlicher Achs-)
                            (geschwindigkeiten der Achsen X bzw. Y)
N20 G01 G91 X100 Y-100 F1000
N30 X-100
```

In diesem Beispiel wird der Satzübergang N20/N30 ohne Sollgeschwindigkeitseinbruch gefahren. Die Sollwerte ändern sich am Satzübergang sprungförmig entsprechend den Achsanteilen an der Bahn.

Achtung

Der Koordinatenwert ist nur aus syntaktischen Gründen erforderlich, ansonsten jedoch nicht relevant.

Durch **G117** werden **sämtliche** Einzelfunktionen der Look-Ahead-Funktion wieder aktiv geschaltet (Defaulteinstellung nach Programmstart).

Programmierbeispiel

```
Nnn G117            (Einschalten aller Einzelfunktionen (1),(2),(3),(4))
                    (der Look-Ahead-Funktion)
```

4.26 Override (G166)

Syntax:

G166 Override auf 100 % festsetzen nicht modal

Mit der Funktion G166 wird **satzweise** die externe Beeinflussung des Overrides von Bahnachsen ausgeschaltet sowie die Wirkung der programmierten Overridewerte für die Bahn [▶ 464] oder eine Achse [▶ 842] unterdrückt.

Programmierbeispiel

Override (G166)

```
%override_G166_extern
;Annahme: Externer Override 50%
N10 G00 G90 X0 Y0 Z0     ;Eilgang 50%
N20 G01 X10 F2000        ;Vorschub F1000
N30 X20 Y20              ;Vorschub F1000
N40 G166 Z30             ;Vorschub F2000, Override 100%
N50 X30                  ;Vorschub F1000
N60 G166 Y30             ;Vorschub F2000, Override 100%
N70 G166 G00 X0 Y0 Z0    ;Eilgang 100%
N80 G01 F3000
N90 X10 Y20 Z30          ;Vorschub F1500

M30
MN10 G00
...
o%override_G166_path

;Bahnoverride: G01 120%, G00 75%
N10 #OVERRIDE [FEED_FACT=120 RAPID_FACT=75]

override_G166_path

;Bahnoverride: G01 120%, G00 75%
N10 #OVERRIDE [FEED_FACT=120 RAPID_FACT=75]

N20 G01 X10 Y10 Z10 F1000   ;Vorschub F1200
N30 G166 X20 Y20 Z20        ;Vorschub F1000, Override 100%
...
N50 G00 X50                 ;Eilgang 75%
N60 G166 Y50 Z50            ;Eilgang 100%
...
M30

%override_G166_ax

;Achsoverride fuer X: G01 20%, G00 60%
N10 X[OVERRIDE FEED_FACT=20 RAPID_FACT=60]

N20 G00 X10                 ;Eilgang in X mit Override 60%
N30 Y10 Z10                 ;Eilgang in Y/Z mit 100%
N40 G166 X20                ;Eilgang in X mit Override 100%

N50 G01 X30 F2000           ;Vorschub in X mit F400, Override 20%
N60 Y20 Z20                 ;Vorschub in Y/Z mit F2000 (100%)
N70 G166 X40                ;Vorschub in X mit F2000, Override 100%
...
M30
```

4.27 Taktsynchronisierung am Satzende (G66)

Syntax:

G66	Bewegungssatzende mit IPO-Takt synchronisieren	nicht modal

Mit der Funktion G66 wird im Bewegungssatz die Geschwindigkeit so angepasst, dass der jeweilige Stützpunkt immer im Interpolator (IPO)-Takt erreicht wird.

Ein in Kombination mit G66 programmierter Bewegungssatz kann so in Programmschleifen reproduzierbar ohne Interferenzen und unerwünschte Geschwindigkeitsschwankungen ausgeführt werden.

4.28 Drehung des Koordinatensystems in der Ebene (G68/G69)

Versionshinweis

Diese Funktionalität ist verfügbar ab CNC-Version V3.1.3079.33

Mit dieser Funktion kann ein Koordinatensystem in der aktuellen Ebene (G17/G18/G19) gedreht werden. Konturen, die im Maschinenkoordinatensystem programmiert sind, können so schnell und einfach an verdreht positionierte Werkstücke angepasst werden.

Die Konturrotation wirkt direkt auf die programmierten Achskoordinaten (Kontur) **vor** allen anderen konturbeeinflussenden Funktionalitäten, d.h. alle Verschiebungen und Spiegelungen werden von der Rotation nicht beeinflusst und können wie bisher benutzt werden (*).

Die Rotation kann auch innerhalb eines bereits gedrehten Koordinatensystems (#(A)CS) angewandt werden.

Ein Ebenenwechsel mit G17/G18/G19 wählt automatisch eine aktive Konturrotation ab und es wird eine Warnung ausgegeben.

Alternativ zu G68/G69 kann die Konturrotation auch mit #ROTATION [▶ 402] programmiert werden.

```
Syntax (Beispiel in G17):
```

G68 R.. X.. Y..	Anwahl der Konturrotation	modal
G69	Abwahl der Konturrotation	modal, Grundzustand

R..	Drehwinkel in Grad [°], wirkt absolut. Ist kein Winkel angegeben, wird der Wert 0° gesetzt. Der Drehwinkel hat keinen Einfluss auf bereits programmierte Kreisradien.
X.. Y..	Absolute Koordinaten des Drehpunktes in [mm, inch] in den Hauptachsen der aktuellen Ebene. Es gilt bei : G17 - X und Y, G18 - Z und X, G19 - Y und Z Bei nicht programmierten Koordinaten wird die aktuelle Istposition als Drehpunkt gesetzt.

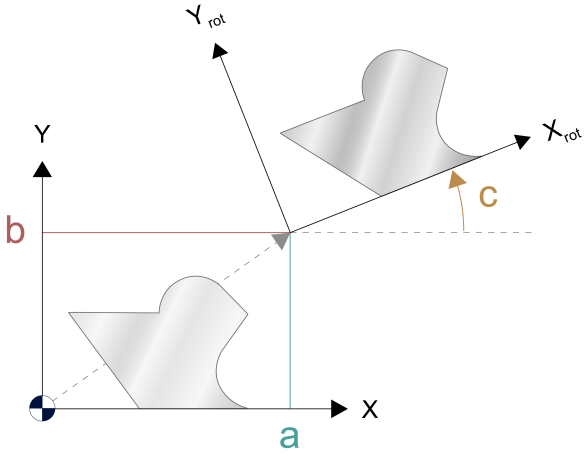


Abb. 63: Bedeutung der Rotationsparameter in der Hauptebene (Bsp. G17):

a: X..	b: Y..	c: R..

Mit nachfolgenden Variablen können die programmierten Rotationsparameter gelesen werden:

V.G.ROT_ACTIVE	Liefert den Wert 1, wenn eine Rotation aktiv ist
V.G.ROT_ANGLE	Rotationswinkel
V.G.ROT_CENTER1	Versatz der ersten Hauptachse zum Drehpunkt
V.G.ROT_CENTER2	Versatz der zweiten Hauptachse zum Drehpunkt

Hinweis

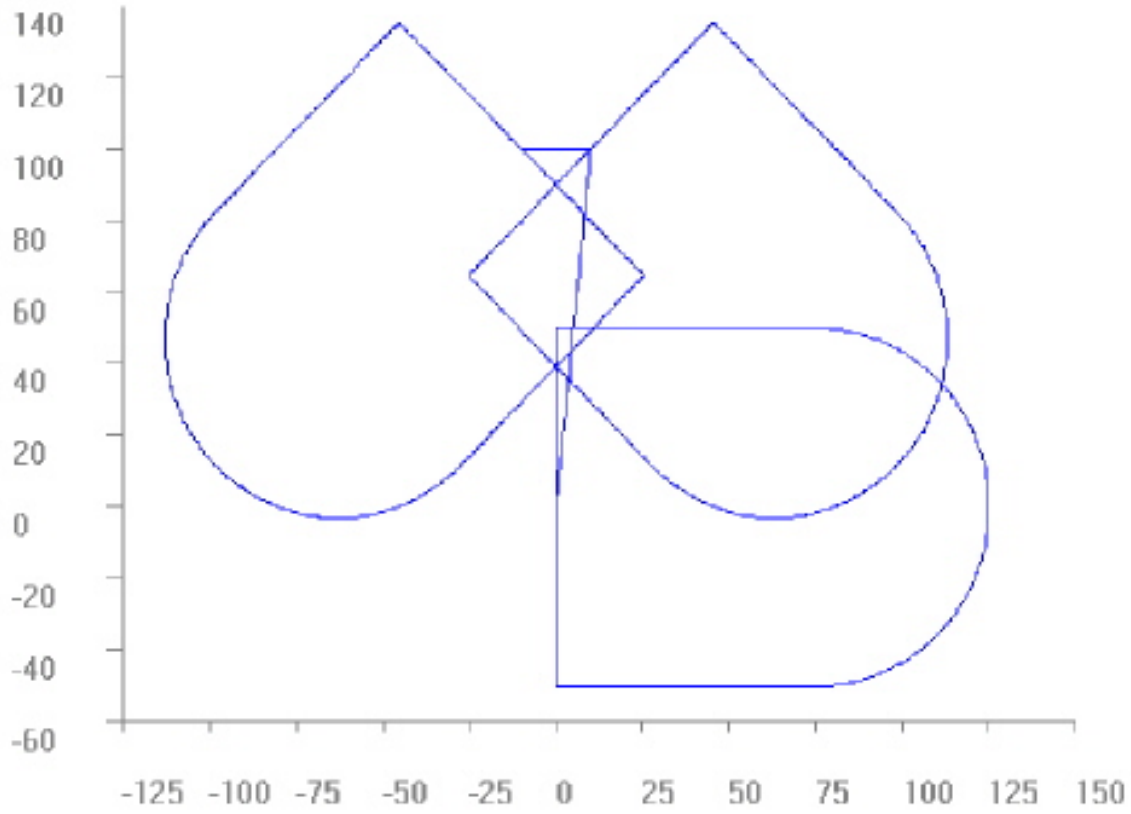
(*) Unabhängig davon, ob die Verschiebungen (z.B. G54, G92 etc.) vor oder nach G68 programmiert wurden, wirken diese immer in den Achsrichtungen des Grundkoordinatensystems der Maschine (MKS).

Auch die Werkzeugversätze wirken immer unabhängig von P-TOOL-00010 in den Achsrichtungen des MKS.

Programmierbeispiel

Drehung in der Ebene (Konturrotation)

```
%L part
N10 G0 G90 X0 Y0
N30 G1 F5000 Y50
N40 X75
N50 G2 Y-50 R50
N60 G1 X0
N70 Y0
N80 M29

%ang1.nc
N100 G53 G17
N110 LL part
N130 G68 R-45 X10 Y100
N140 LL part
N150 G21 (mirroring of X coordinates)
N160 LL part
N170 G69
M30
```



5 Schalt und Zusatzfunktionen (M/H/T)

Eine vollständige Liste der M-Funktionen findet sich in der Befehlsübersicht im Anhang unter M-Funktionen (M..) [▶ 881].

5.1 Anwenderspezifische M-/H-Funktionen

M/H-Funktionen werden im Wesentlichen durch die Programmierung der "Speicherprogrammierbaren Steuerung" (SPS,...) bestimmt. Diese Steuerung erhält über eine Schnittstelle ("Fenster zur SPS") Mitteilung über die programmierten M/H –Funktionen im aktuellen NC-Satz. Zusätzlich werden Synchronisationsbedingungen mitgegeben, die die Synchronisation zwischen NC und SPS steuern.

Syntax:

M..

bzw.

H..

M/H-Funktionen können mit allgemeinen mathematischen Ausdrücken programmiert werden. Dies erlaubt insbesondere die Wertzuweisung über Parameter, z.B. MP10. Vor dem Zugriff auf die M/H-Funktion wird der berechnete Zahlenwert gerundet und in eine Ganzzahl gewandelt. Der kleinste zulässige Wert ist 0 (Null). Negative Werte erzeugen eine Fehlermeldung.

Die maximale Anzahl von M/H-Funktionen pro NC-Kanal [6] [▶ 894]-8.1/-8.2 sowie die maximale Anzahl von M/H-Funktionen pro NC-Satz [6] [▶ 894]-8.3 ist fest vorgegeben.

Mit der EXIST-Funktion (siehe Kapitel Arithmetische Ausdrücke <expr> [▶ 32]) kann geprüft werden, ob eine M/H-Funktion überhaupt verfügbar ist.

Programmierbeispiel

Anwenderspezifische M/H-Funktionen

Durch die EXIST-Abfrage wird in einem ersten Schritt vor der Verwendung einer anwenderspezifischen M-Funktion zunächst geprüft, ob diese überhaupt in der Kanalparameterliste definiert ist P-CHAN-00027:

```
...
N10 G90 Y0
N20 $IF EXIST[M80] == TRUE
N30 X0 Y0 Z0 M80 (M80 ist verfügbar)
N40 $ELSE
: …
N60 $ENDIF
...
M30
```

Folgende M-Funktionen haben im NC-Programm eine vordefinierte Bedeutung, d.h. sie sind nicht frei verfügbar. Lediglich ihr Synchronisationsmode P-CHAN-00027 kann noch vom Anwender selbst belegt werden.

Syntax:

M00	Programmierter Halt
M01	Wahlweiser Halt
M02	Programmende, Stillsetzen der Maschine
M17	Unterprogrammende
M29	Unterprogrammende
M30	Programmende, Stillsetzen der Maschine

5.1.1 Programmierter Halt (M00)

M00 bewirkt eine Unterbrechung des laufenden NC-Programmes, um z.B. eine Messung oder eine Späneentsorgung durchzuführen. Die Bearbeitung wird wieder fortgesetzt, wenn die "Bewegung-Fortsetzen"-Taste gedrückt wird. Steht M00 in einem Satz mit Verfahranweisungen, dann erfolgt die Unterbrechung nach Ausführung dieser Verfahranweisung.

5.1.2 Wahlweiser Halt (M01)

M01 wirkt wie M00, nur muss die Wirksamkeit von M01 am Bedienfeld der Steuerung aktiviert werden.

5.1.3 Programmende (M02/M30)

M02 und M30 sind identisch und bewirken im Haupt- und Unterprogramm das Rücksetzen an den Hauptprogrammanfang und die Rückführung der NC in die Grundstellung.

Zusätzlich ist es durch Konfigurierung möglich, ein Signal an die SPS weiterzugeben. Die SPS verwendet dieses Signal z.B. um einen Werkstückwechselprozess zu synchronisieren.

5.1.4 Unterprogrammende (M17/M29)

M17 und M29 bewirken das ordentliche Beenden und Schließen eines lokalen oder globalen Unterprogrammes. Danach erfolgt der Rücksprung in das aufrufende NC-Programm an den Anfang der NC-Zeile, die dem Unterprogrammaufruf folgt, um die Bearbeitung fortzusetzen. Siehe hierzu auch Kapitel Unterprogrammtechniken [▶ 205]. Durch Konfigurierung ist es möglich, zusätzlich ein Signal an die SPS weiterzugeben.

Parametrierbarer Rücksprung M17/M29 aus einem Unterprogramm

Versionshinweis

Diese Funktionalität ist verfügbar ab V3.01.3081.02

Durch zusätzliche Angaben nach M17/M29 kann der Rücksprung parametriert werden, sodass der Anwender selbst vorgeben kann, in welcher höheren Programmebene und an welcher Programmstelle die Bearbeitung fortgesetzt werden soll. Ein Praxisfall wäre z.B. ein detektierter Werkzeugbruch in einem Unterprogramm, der nach sofortigem Verlassen des Unterprogramms unmittelbar im Hauptprogramm behandelt werden soll.

Die Parametrierung des Rücksprungs mit M17/M29 wird wie folgt programmiert:

`Syntax parametrierbarer Unterprogrammrücksprung:`

M17 | **M29 [** [**MAIN** | **JUMPS=..**] [**[***<LABEL>***]** | **N..**] **]**

MAIN	Rücksprung auf Hauptprogrammebene
JUMPS=..	Anzahl der Programmebenen, die zurückgesprungen werden soll, um die Bearbeitung fortzusetzen. Ausgabe einer Warnung, falls Anzahl der Rücksprungebenen die aktuelle Aufruftiefe übersteigt und Rücksprung auf Hauptprogrammebene.
[*<LABEL>*]	Angabe eines String-Labels [▶ 240] als Sprungziel in der Rücksprungebene. Das String-Label kann dort sowohl in Vorwärts- als auch in Rückwärtsrichtung liegen.
N..	Angabe einer Satznummer als Sprungziel in der Rücksprungebene. Es wird zuerst geprüft, ob die Satznummer als Expression-Label N..: [▶ 240] bereits existiert und dann evtl. dorthin verzweigt. Das Expression-Label kann dort sowohl in Vorwärts- als auch in Rückwärtsrichtung liegen. Ist dies nicht der Fall, wird in der Rücksprungebene diese Satznummer ausschließlich Vorwärts in Richtung Programmende gesucht.

MAIN und JUMPS sind explizit aber optional. Ohne Angabe wird in die nächste Unterprogrammebene zurückgesprungen (implizites JUMPS=1). Die Hauptprogrammebene ist die Ebene 1. Die aktuelle Programmebene kann im NC-Programm mit V.G.PROG_LEVEL [▶ 602] gelesen werden.

[LBL] und N.. sind ebenfalls explizit und optional. Ohne Angabe wird die Programmbearbeitung nach Rücksprung mit dem nächsten NC-Satz fortgesetzt.

Hinweis

Das Sprungziel muss in der Rücksprungebene liegen. In lokalen Unterprogrammen oder in weiteren Unterprogrammaufrufen erfolgt keine Suche.

Programmierbeispiel

Syntaxbeispiele parametrierter Unterprogrammrücksprünge mit M17

```
M17 [MAIN]            ;Rücksprung in Hauptprogramm an den Ursprung der
                      ;Aufrufkette und nächsten NC-Satz ausführen.
M17 [JUMPS=3 N150]    ;3 Programmebenen zurückspringen und dort N150
                      ;als nächsten NC-Satz ausführen.
M17 [MAIN N500]       ;Rücksprung in Hauptprogramm und dort N500 als
                      ;nächsten NC-Satz ausführen.
M17 [N300]            ;1 Programmebene zurückspringen und dort N300 als
                      ;nächsten NC-Satz ausführen.
M17 [JUMPS=2 [LBL4]]  ;2 Programmebenen zurückspringen und dort den
                      ;NC-Satz mit der Sprungmarke [LBL4] ausführen.
M17 [MAIN [END] ]     ;Rücksprung in Hauptprogramm und dort den
                      ;NC-Satz mit der Sprungmarke [END] ausführen.
M17 []                ;1 Programmebene zurückspringen und nächsten
                      ;NC-Satz ausführen (analog zu M17)
```

5.1.5 Aufruf eines Werkzeugwechselprogramms (M06)

Bei entsprechender Konfiguration (P-CHAN-00118) wird die M-Funktion M06 nicht als Technofunktion interpretiert, sondern bewirkt den Aufruf eines globalen Unterprogramms. In diesem Unterprogramm kann der Anwender alle Aktionen programmieren, die für den physikalischen Werkzeugwechsel notwendig sind.

Hinweis

Versionsspezifisch sind in der Defaulteinstellung im laufenden NC-Programm oder im Einzelsatzbetrieb während der Ausführung von M6 in der Anzeige die Bearbeitungszeilen des Werkzeugwechselprogramms ausgeblendet oder sichtbar. Bei aktiver Ausblendung wird in dieser Zeit nur der M6-Aufruf angezeigt.

Dieses Verhalten kann über den Kanalparameter P-CHAN-00211 geschaltet werden.

Hinweis

In TwinCAT-Systemen sind standardmäßig in der Anzeige alle Zykluszeilen sichtbar.

5.2 Achsspezifische M-/H-Funktionen

Allgemein werden anwenderspezifische M/H-Funktionen, die in der Standard-DIN-Syntax programmiert werden, kanalspezifisch behandelt und ausgeführt.

Für jede anwenderspezifische M/H –Funktion besteht jedoch auch die Möglichkeit durch Vorbelegung in den Kanalparametern P-CHAN-00039 und P-CHAN-00025 eine achsspezifische Behandlung zu erzwingen.

Syntax:

m_default_outp_ax_name[<*m_nr*>] <*Achsname*>

bzw.

h_default_outp_ax_name[<*h_nr*>] <*Achsname*>

<*m_nr*>	Nummer einer anwenderspezifischen M-Funktion mit achsspezifischer Wirkung.
<*h_nr*>	Nummer einer anwenderspezifischen H-Funktion mit achsspezifischer Wirkung.
<*Achsname*>	Achsbezeichnung der Achse, auf die die M/H-Funktion wirken soll. Hierbei sind Bahnachsen und Spindelachsen zulässig.

Programmierbeispiel

Achsspezifische M/H-Funktionen

Kanalparametersatz [1] [▶ 894]:

```
:
# Festlegung der achsspezifischen M-Funktionen
# ============================================
m_default_outp_ax_name[20] S2
m_default_outp_ax_name[21] S3
m_default_outp_ax_name[22] Z
:
# Festlegung der achsspezifischen H-Funktionen
# ============================================
h_default_outp_ax_name[10] X
h_default_outp_ax_name[11] Y
h_default_outp_ax_name[12] Z
:

N.. S1000 M3 M20 M21 M22 H10  ;S, M3 wirken auf die Hauptspindelachse
                              ;M20 wirkt auf die S2-Spindelachse
                              ;M21 wirkt auf die S3-Spindelachse
                              ;M22 wirkt auf die Z-Achse
                              ;H10 wirkt auf die X-Achse
:
```

Hinweis

Wird eine achsspezifische M/H-Funktion in einer spindelspezifischen Klammerung programmiert, so wird die Defaulteinstellung ignoriert und die M/H-Funktion wirkt auf die entsprechende Spindelachse.

Für Bahnachsen besteht darüber hinaus im NC-Programm die Möglichkeit, durch Klammerung bei M/H-Funktionen ebenfalls eine achsspezifische Ausgabe zu erzwingen:

Syntax:

<Achsname> **[** [**M**..] [**H**..] { [**M**..] [**H**..] } **]** { *<Achsname>* [...] }

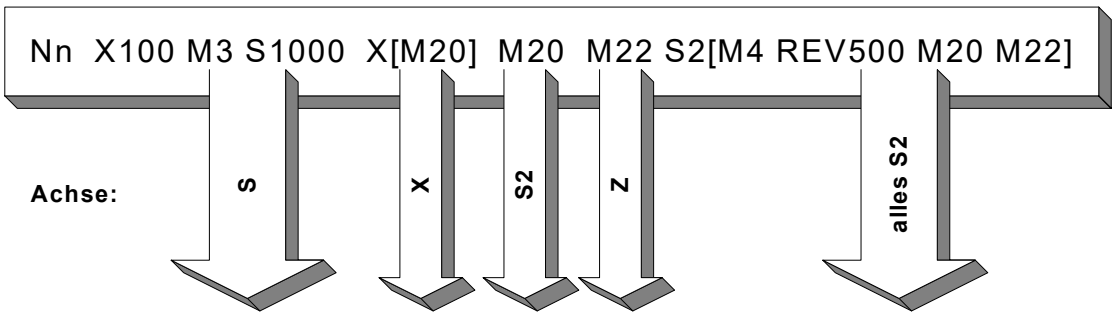
<Achsname>	Achsbezeichnung der Achse, auf die die M/H-Funktion wirken soll. Hierbei sind nur Bahnachsen zulässig.
M..	Anwenderspezifische M-Funktion mit achsspezifischer Wirkung.
H..	Anwenderspezifische H-Funktion mit achsspezifischer Wirkung.

Programmierbeispiel

```
:
N10 S1000 M3 X100 X[M20 H12] Y[H10]      (S, M3 wirken auf die Hauptspindel)
                                         (M20, H12 wirken auf die X-Achse)
                                         (H10 wirkt auf die Y-Achse)
:
```

Der Anwender hat somit die Möglichkeit sehr flexibel seine speziellen M/H-Funktionen durch Programmierung oder Konfigurierung achsspezifisch oder kanalspezifisch auszuführen.

Programmierbeispiel



5.3 M-/H-Funktionen mit optionaler Zusatzinformation

Bei M/H-Funktionen kann im NC-Programm optional ein Zusatzwert programmiert werden, der über die Technologieschnittstelle mit der M/H-Funktion der SPS zur Verfügung gestellt wird.

Der Zusatzwert kann zu allen anwenderspezifischen M/H-Funktionen sowie zu den internen M-Funktionen M00, M01, M02, M17, M29 und M30 programmiert werden. Die M/H-Funktionen sind hierbei ohne Einschränkungen sowohl in kanal- als auch in achsspezifischer Programmiersyntax verwendbar.

Folgende M-Funktionen dürfen nur ohne Zusatzwert programmiert werden:

- Die Spindelfunktionen M03, M04, M05 und M19,
- die Getriebeschaltfunktionen M40-M45 und
- die Unterprogrammfunktion M6 (wenn mit P-CHAN-00118 als solche konfiguriert)

Syntax:

M.. [**=** <*Additive_value*>]

bzw.

H.. [**=** <*Additive_value*>]

M..	Anwenderspezifische M-Funktion
H..	Anwenderspezifische H-Funktion
= <*Additive_value*>	Optionaler Zusatzwert, der als negative oder positive Ganzzahl direkt oder als allgemeiner mathematischer Ausdruck programmiert wird

Programmierbeispiel

M/H-Funktionen mit optionaler Zusatzinformation

```
%m_h_add_fkt

(M-Funktionen mit Zusatzwert)
N10 M52=-345
N20 M12=123       ;mit Kanalparameter m_default_outp_ax_name[12] Z
N30 M10=321       ;mit Kanalparameter m_default_outp_ax_name[10] S
N35 P1 = 567 P2 = 345
N40 X[M54=P1]
N50 S[REV 1000 M03 M63=-789]
N60 M12=123 M10=321 M52=-345 X[M54=567] S[REV 1000 M03 M63=-789]
N70 M63=-789 M52=-P2 M54=567
N80 X[M52=-345 M54=567] Y[M63=-789] S[M05 M63=789 M54=-567] M54 M63


(H-Funktionen mit Zusatzwert)
N110 H5=-345
N120 H6=123       ;mit Kanalparameter h_default_outp_ax_name[6] Z
N130 H9=321       ;mit Kanalparameter h_default_outp_ax_name[9] S
N135 P3 = 567 P4 = -345
N140 X[H7=P3]
N150 S[REV 1500 M04 H8=-789]
N160 H6=123 H9=321 H5=-345 X[H7=567] S[REV 1500 M04 H8=-789]
N170 H8=-789 H5=P4 H7=567
N180 X[H8=-789 H4 H5=-345] Y[H7=567] S[M05 H5=345 H7=567] H3 H8
```

```
(Gemischte M/H- Funktionen mit Zusatzwert)
N200 X[M52=-345 H4 H8=-789 M54=567 H5=345] H3=333 M54=444 H7=567 M63

(M/H- Funktionen mit Zusatzwert in achsspezifischer Funktion(INDP))
N05 X[INDP G90 G01 FEED=2000 POS=555 M54=151 H8=-181]

N999 M30=111
```

Diese Zusatzinformationen können in der SPS über die Daten der M-/H-Funktion gelesen werden.

Siehe auch[HLI// Daten der M-/ H-Funktion]

5.4 Werkzeugplatzanwahl (T-)

Syntax:

T.. Werkzeugplatzanwahl modal

Der Werkzeugbefehl bestimmt das für den Bearbeitungsabschnitt benötigte Werkzeug. Die Werkzeugnummer wird an die SPS nach außen weitergereicht, die dann in einem Werkzeugmagazin das angewählte Werkzeug für einen Werkzeugwechsel bereitstellt. Es ist zu beachten, dass das T-Wort selbst keinen Einfluss auf die steuerungsinterne Verrechnung der Werkzeuggeometrie (Werkzeugdaten) hat. Hierzu wird das D-Wort [▶ 499] verwendet, welches jedoch bei entsprechender Parametrierung P-CHAN-00014 automatisch mit dem T-Wort ausgelöst wird.

Das T-Wort initiiert noch keinen Werkzeugwechsel. Dies erfolgt üblicherweise mit der Maschinenfunktion M06.

Die Werkzeugdaten [5] [▶ 894], [6] [▶ 894]-9.7 werden innerhalb der Werkzeugdatentabelle des Decoders abgelegt [6] [▶ 894]-9.1. Daneben besteht alternativ die Möglichkeit, über eine zusätzliche Schnittstelle mit einer vom Anwender bereitgestellten ggf. dezentralen (externen) Werkzeugverwaltung zu kommunizieren und Werkzeugdaten anzufordern. Dieser interne bzw. externe Zugriff auf Werkzeugdaten wird parametriert P-CHAN-00016.

Um Wartezeiten aus der Bereitstellung und Übermittlung der Werkzeugdaten zu vermeiden, wird das von außen bereitzustellende Werkzeug üblicherweise einen oder mehrere Bewegungssätze vor der eigentlichen Einrechnung der Werkzeugdaten angefordert P-CHAN-00087. Werden mehrere eventuell verschiedene T-Nummern programmiert, bevor die der ersten T-Nummer entsprechende D-Nummer erscheint, so werden alle T-Nummern vor der zuletzt programmierten ignoriert. Das heißt, die von der externen Werkzeugverwaltung gelieferten Daten werden verworfen, falls die D-Nummer des Werkzeugs nicht mit der letztprogrammierten T-Nummer übereinstimmt.

6 Geschwindigkeiten (F/E)

Syntax:

F..	Vorschubgeschwindigkeit im Satz	modal
E..	Vorschubgeschwindigkeit am Satzende	nicht modal

Bei den Interpolationsarten G1 [▶ 55], G2, G3 [▶ 56] oder für G100 [▶ 104] werden die programmierten Bahnen mit der im F-Wort vereinbarten Bahngeschwindigkeit abgefahren. Die Angabe für das F- und E-Wort erfolgt

- bei translatorischen Achsen in [mm/min, m/min, inch/min]
- bei rotatorischen Achsen in [°/min].

Das F-Wort ist haltend wirksam. Die Programmierung in mm/min bzw. m/min kann mit dem Kanalparameter P-CHAN-00108 konfiguriert werden.

Mit dem Befehl G93 [▶ 155] ist es auch möglich, über das F-Wort statt einer Bahngeschwindigkeit eine Bearbeitungszeit vorzugeben. Die Beschreibung befindet sich im Kapitel "G-Funktionen".

Mit dem E-Wort wird die Bahngeschwindigkeit am Satzende programmiert. Der Wert ist satzspezifisch, d.h. nicht haltend wirksam. Ist das E-Wort nicht angegeben oder der programmierte Wert ist größer als das F-Wort, so wird das E-Wort mit dem Wert des F-Wortes belegt.

Hinweis

Das E-Wort ist nur in Verbindung mit G94 [▶ 155] wirksam!

Eingefügte Konturelemente, bedingt durch aktive konturverändernde Funktionen wie G301, G302 [▶ 156], G41, G42 [▶ 506], G261 [▶ 131], #HSC [▶ 254] [OPMODE 1…] übernehmen den E-Vorschub der primären Sätze als F-Vorschub.

Bei allen anderen Spline- und HSC-Funktionen wird empfohlen, den E-Vorschub nicht zu verwenden.

Dem F- und E-Wort können Werte direkt oder per Parameter zugewiesen werden, wobei auch Dezimalzahlen (REAL-Format) zulässig sind.

Hinweis

Über die SPS kann der Bahnvorschub auch extern vorgegeben und mit dem NC-Befehl #FF [▶ 486] zusätzlich gewichtet werden.

Programmierbeispiel

Geschwindigkeiten (F-, E-)

```
N10 X200 G01 F1000 G90  ;F-Vorschub 1000 mm/min
N20 X300
N30 X350 F800           ;F-Vorschub Absenken von 1000 auf 800 mm/min
```

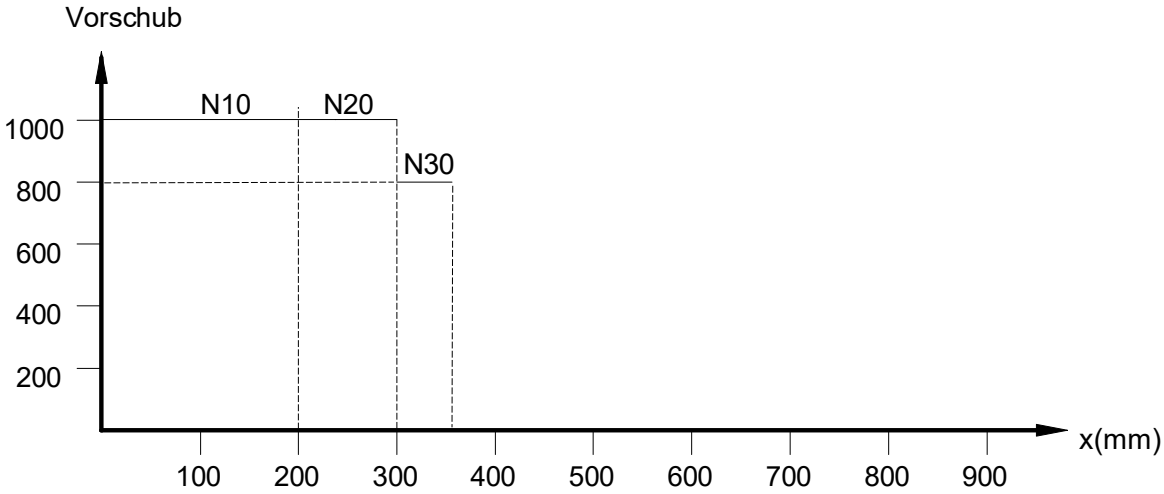


Abb. 64: Vorschubprogrammierung mit dem F-Wort

Programmierbeispiel

```
N05 G01 G94 G90
N10 X200 F1000 E500      ;F-Vorschub 1000 mm/min, E-Vorschub 500mm/min
N20 X250 E400            ;F-Vorschub 1000 mm/min, E-Vorschub 400mm/min
N30 X350 F800 E300       ;F-Vorschub 800 mm/min, E-Vorschub 300mm/min
N40 X400                 ;F-Vorschub 800mm/min, E-Vorschub 800mm/min
```

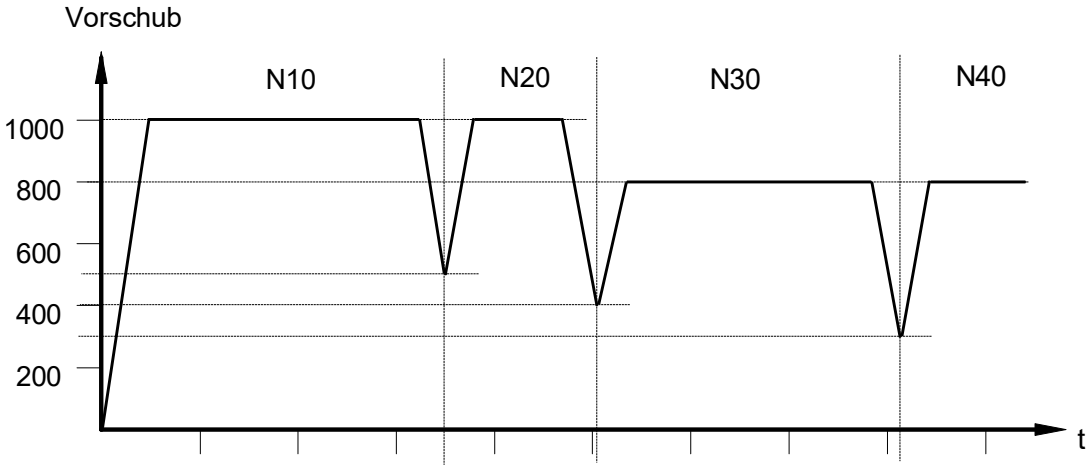


Abb. 65: Vorschubprogrammierung mit dem F- und E-Wort

Programmierbeispiel

```
N10 G01 G94 G90 G261
N20 X100 F1000 E200       ;F-Vorschub 1000 mm/min, E-Vorschub 200mm/min
N30 Y100 E200             ;F-Vorschub 1000 mm/min, E-Vorschub 200mm/min
N40 X0                    ;F-Vorschub 1000 mm/min
N50 Y0 E200               ;F-Vorschub 1000 mm/min, E-Vorschub 200mm/min
N60 X50
N70 G260
```

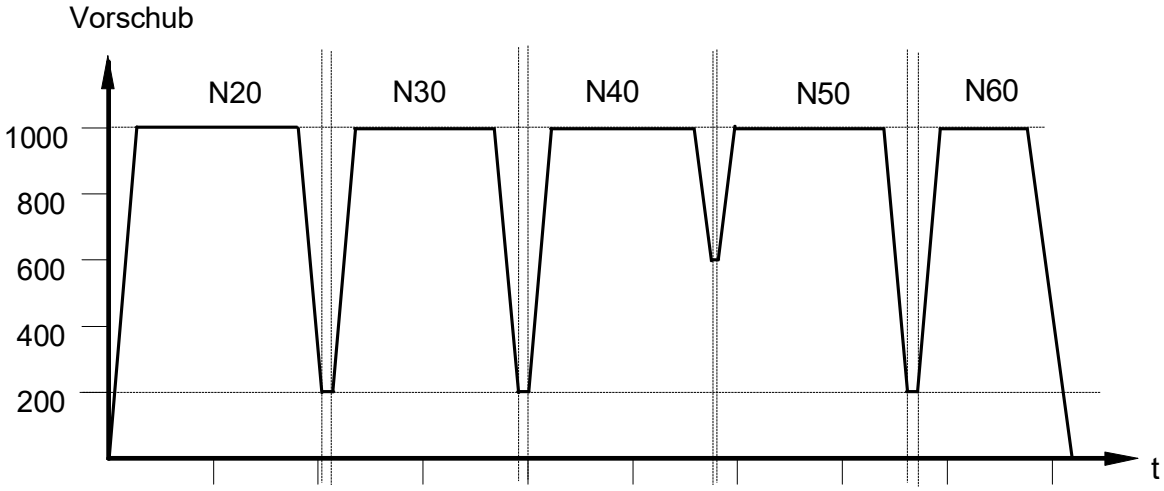


Abb. 66: Wirkung des E-Wortes auf eingefügte Konturelemente (hier G261)

7 NC-Satznummern (N)

Mit dem Adressbuchstaben "N" kann die NC-Satznummer programmiert werden. Diese Nummer wird bei entsprechender Konfigurierung sowohl in die Anzeigedaten, als auch in die Fehlermeldungen eingetragen (alternativ dazu kann auch der Offsetwert im File verwendet werden).

N..

Die Satznummer kann mit allgemeinen mathematischen Ausdrücken programmiert werden. Dies erlaubt insbesondere die Wertzuweisung über Parameter, z.B. über den Laufparameter in Programmschleifen. Der berechnete Zahlenwert wird intern automatisch gerundet und in eine Ganzzahl gewandelt.

Programmierbeispiel

NC-Satznummern (N-Funktion)

```
N10 G01 X200 F500
N20 $FOR P1=1, 10, 1
N[P1*100] XP1
.
.
N30 $ENDFOR
```

Für den Programmablauf ist die Satznummer ohne Bedeutung. Sie muss daher auch nicht in aufsteigender Form verwendet werden.

8 Unterprogrammtechniken

Mehrfach zu wiederholende gleiche Bewegungsfolgen und Funktionsabläufe können als Unterprogramme realisiert werden. Es werden 2 Arten von Unterprogrammen unterschieden:

1. lokale Unterprogramme,
2. globale Unterprogramme bzw. Zyklen.

Mehrere Unterprogramme (lokal oder global) können ineinander geschachtelt aufgerufen werden, wobei die Anzahl verwendeter Unterprogramme sowie die Schachtelungstiefe fest vorgegeben sind [6] [▶ 894]-24.

Die Übergabe von Datenwerten vom Haupt- an das Unterprogramm erfolgt indirekt über P-Parameter bzw. bei Zyklen direkt im Aufruf über spezifische Versorgungsparameter (@P...).

Übersicht über Programmdefinition und -aufruf:

Hauptprogramm

Definition:	%PROG_NAME oder % PROG_NAME	Die Hauptprogrammdefinition ist nur dann notwendig, wenn zuvor lokale Unterprogramme definiert wurden.
Aufruf:	Über die Bedienung durch Angabe des vollständigen Dateinamens (inklusive Dateiendung)	

Lokales Unterprogramm

Definition:	%L UP_NAME	Trennzeichen zwischen "L" und Programmdefinition, da es sich sonst um die Definition eines Hauptprogramms handelt
Aufruf im NC-Programm:	LL UP_NAME	Trennzeichen zwischen "LL" und Programmaufruf, da es sich sonst um den Aufruf eines globalen Unterprogramms handelt

Globales Unterprogramm

Definition:	%PROG_NAME oder % PROG_NAME	Die Unterprogrammdefinition ist nur dann notwendig, wenn zuvor lokale Unterprogramme definiert wurden.
Aufruf im NC-Programm:	L DATEI_NAME	Angabe des vollständigen Dateinamens (inklusive Dateiendung). Trennzeichen zwischen "L" und Programmaufruf.

Hinweis

Wird in der Datei außerhalb von Kommentaren als erstes Zeichen eines gefunden, das weder ein Trennzeichen noch ein "%" ist, so wird dieses Zeichen als erstes Zeichen eines namenlosen Hauptprogrammes gewertet. Das bedeutet auch, dass vor "%" keine Satznummern programmiert werden dürfen.

8.1 Lokale Unterprogramme (Aufruf LL <string>)

Der Aufruf eines lokalen Unterprogramms vom Hauptprogramm aus erfolgt mit

LL <string> (Achtung: Leerzeichen zwischen LL und <string> zwingend erforderlich).

<string> Name des lokalen Unterprogramms ohne Hochkommas, max. 83 Zeichen.

Lokale Unterprogramme (LUP) stehen zusammen mit dem Hauptprogramm in einer gemeinsamen Datei, wobei zuerst alle Unterprogramme vor dem eigentlichen Hauptprogramm aufgeführt sein müssen.

Es ist zu beachten, dass lokale Unterprogramme nur von dem Hauptprogramm in der gleichen Datei aufgerufen werden können.

Lokale Unterprogramme beginnen mit "%L" und einer Programmbezeichnung. Zwischen "L" und der Programmbezeichnung muss mindestens ein Trennzeichen stehen.

Das Unterprogrammende wird durch die Funktionen M17 oder M29 markiert. Auch ein Programmabbruch mit M02 oder M30 ist möglich. Es wird dann ein Warnhinweis gegeben.

Fehlen M17 oder M29, so beendet "%" (erstes Zeichen des nachfolgenden Hauptprogramms oder eines weiteren Unterprogramms) das Unterprogramm.

Programmierbeispiel

Lokale Unterprogramme (Aufruf LL <string>)

```
Aufbau eines Datenfiles, bestehend aus NC-Hauptprogramm und lokalen Un-
terprogrammen:
%L UP1              (1. lokales Unterprogramm)
N1 .....
N2 .....
N9 M17              (M17 kann auch entfallen)


%L UP2              (2. lokales Unterprogramm)
N11 .....
N12 .....
N19 M29             (M29 kann auch entfallen)


%MAIN               (Hauptprogramm)
N100 .....
N105 .....
N200 LL UP1         (Aufruf des 1. LUPs)
.
N250 LL UP2         (Aufruf des 2. LUPs)
.
N300 M30            (Hauptprogrammende)
```

Vor der Definition eines Programmes sind keine NC-Befehle erlaubt. Hier findet sich eine Ausnahme der Regel, dass die Reihenfolge im NC-Satz für die Programmbearbeitung ohne Bedeutung ist (vgl. Kapitel NC-Satzaufbau [▶ 25]).

8.2 Globale Unterprogramme (Aufruf L <string>)

Der Aufruf eines globalen Unterprogramms erfolgt mit

L *<string>*

<string> Dateiname des globalen Unterprogramms ohne Hochkommas, max. 83 Zeichen (inklusive Dateiendung und evtl. einer absoluten oder relativen Pfadangabe)

Globale Unterprogramme (GUP) stehen als eigenständige Programmeinheiten in einer separaten Datei. Der Aufruf eines globalen Unterprogramms erfolgt über den vollständigen Dateinamen (inklusive Dateiendung). Ein globales Unterprogramm kann wiederum aus lokalen Unterprogrammen und einem Hauptprogrammteil bestehen.

Die Angabe eines Namens des globalen Unterprogramms mit % in der Datei ist nur erforderlich, um nach der Definition lokaler Unterprogramme den Beginn des Hauptprogrammteils anzuzeigen. Sie kann entfallen, falls keine lokalen Unterprogramme in der Datei enthalten sind.

Das aufrufende Hauptprogramm ist ebenfalls als eigenständige Programmeinheit in einer anderen Datei abgelegt. Globale Unterprogramme können von allen Hauptprogrammen aus aufgerufen werden.

Das Ende eines globalen Unterprogramms wird durch die Funktionen M17 oder M29 markiert. Auch ein Programmabbruch mit M02 oder M30 ist möglich. Es wird dann ein Warnhinweis gegeben.

Programmierbeispiel

Globale Unterprogramme (Aufruf L <string>)

```
;Aufruf von globalen Unterprogrammen
%MAIN (Hauptprogramm)
N100 .....
N105 .....
N200 L gup_1.nc  ;Aufruf eines globalen Unterprogramms direkt über
                 ;Dateinamen, wenn Programmpfad konfiguriert ist
N250 L D:\prog\ini_1.nc  ;Aufruf eines globalen Unterprogramms mit
                         ;absoluter Programmpfadangabe
N300 M30
```

8.3 Parametrierter Unterprogrammaufruf (LL / L V.E. oder Makro)

Anstelle von festen Namen kann der Aufruf von lokalen sowie globalen Unterprogrammen auch über externe Variablen oder Makros erfolgen. Dies ermöglicht einen parametrierbaren Ablauf des NC-Programms.

Die externen Variablen müssen vom Typ String bzw. Stringarray sein (siehe auch Kapitel Externe Variablen (V.E.) [▶ 633]). Die Stringlänge lokaler Unterprogrammnamen beträgt maximal 83 Zeichen, bei globalen Unterprogrammnamen ebenfalls 83 Zeichen jedoch inklusive einer evtl. absoluten oder relativen Pfadangabe.

Makros müssen vor ihrer Verwendung mit L oder LL definiert sein. Der Makroinhalt hat eine maximale Stringlänge von 80 Zeichen.

Der Aufruf eines **lokalen** Unterprogramms vom Hauptprogramm aus erfolgt mit LL :

LL V.E. ... (Achtung: Leerzeichen zwischen LL und V.E. ... zwingend erforderlich).
oder
LL "<Makroname>" (Achtung: Leerzeichen zwischen LL und Makroname zwingend erforderlich).

V.E. ...	Über externe Variable parametrierter Name des lokalen Unterprogramms
"<*Makroname*>"	Über Makro parametrierter Name des lokalen Unterprogramms. Ist das Makro nicht definiert, so wird <*Makroname*> als normaler lokaler Unterprogrammname behandelt.

Der Aufruf eines **globalen** Unterprogramms erfolgt mit L :

LV.E. … oder **L V.E**. …
oder
L"<Makroname>" oder **L** "<Makroname>"

V.E. ...	Über externe Variable parametrierter Name der Datei, in dem dieses globale Unterprogramm abgelegt ist.
"<*Makroname*>"	Über Makro parametrierter Name der Datei, in dem dieses globale Unterprogramm abgelegt ist. Ist das Makro nicht definiert, so wird <*Makroname*> als normaler globaler Unterprogrammname behandelt.

Programmierbeispiel

Parametrierter Unterprogrammaufruf (LL / L V.E. oder Makro)

Aufruf von Unterprogrammen über externe Variablen vom Typ String

```
;lokales Unterprogramm
%L TASCHE
N10 .....
.
N99 M17

;Hauptprogramm
%MAIN
N100 .....
N105 .....
;Aufruf des lokalen Unterprogramms über ext. Variable V.E.LUP,
;die den String TASCHE enthält
N110 LL V.E.LUP
.

;Aufruf des globalen Unterprogramms über ext. Variable V.E.GUP,
;die den String eines Dateinamens enthält
N200 L V.E.GUP


N300 M30
```

Programmierbeispiel

Parametrierter Unterprogrammaufruf (LL / L V.E. oder Makro)

Aufruf von Unterprogrammen über Makros

```
;lokales Unterprogramm 1
%L TASCHE_1
N10 .....
.
N99 M17

;lokales Unterprogramm 2
%L TASCHE_2
N10 .....
.
N99 M17

;lokales Unterprogramm 3
%L TASCHE_3
N10 .....
.
N99 M17

;Hauptprogramm
%MAIN
;Makrodefinitionen
N10 "LUP_1" = "TASCHE_1"
N20 "LUP_2" = "TASCHE_2"
N30 "LUP_3" = "TASCHE_3"

N40 "GUP_1" = "gup_1.nc"
N50 "GUP_2" = "D:\prog\ini_1.nc"

N100 .....
;Aufruf der lokalen Unterprogramme über Makros,
;welche die Strings TASCHE_1, TASCHE_2, TASCHE_3
;enthalten
N110 LL "LUP_1"
N120 LL "LUP_2"
N130 LL "LUP_3"

N200 .....
;Aufruf der globalen Unterprogramme über Makros,
;welche die Strings der Dateinamen enthalten
N210 L "GUP_1"
N220 L "GUP_2"

N300 M30
```

8.4 Impliziter globaler Unterprogrammaufruf bei Programmstart

Häufig müssen beim Start unterschiedlicher NC Hauptprogramme immer wiederkehrende gleichbleibende Initialisierungen von Daten durchgeführt werden. Werden diese Initialisierungen in einem globalen Unterprogramm zusammengefasst, so besteht über den Kanalparameter P-CHAN-00119 die Möglichkeit, dieses Programm bei jedem NC Programmstart implizit als erste Aktion ausführen zu lassen.

In diesem globalen Unterprogramm kann der volle Funktionsumfang der NC-Programmierung genutzt werden. D.h. es verhält sich genauso wie ein Unterprogrammaufruf mit dem L-Wort im NC Hauptprogramm.

Hinweis

In erster Linie dient dieser Mechanismus zur Initialisierung, Definition und Voreinstellung von Verschiebungen, G-Funktionen, Parameterwerten, Variablen etc. Konkrete Bearbeitungsvorgänge sollten hier nicht programmiert sein.

8.5 Impliziter globaler Unterprogrammaufruf bei Programmende

Häufig müssen am Ende (M02, M30) unterschiedlicher NC Hauptprogramme immer wiederkehrende gleichbleibende Aktionen (z.B. Abwahl von Transformationen, Verschiebungen, Funktionalitäten etc.) durchgeführt werden. Werden diese Initialisierungen in einem globalen Unterprogramm zusammengefasst, so besteht über den Kanalparameter P-CHAN-00252 die Möglichkeit, dieses Programm **vor** M02 bzw. M30 implizit ausführen zu lassen.

In diesem globalen Unterprogramm kann der volle Funktionsumfang der NC-Programmierung genutzt werden. D.h. es verhält sich genauso wie ein Unterprogrammaufruf mit dem L-Wort im NC Hauptprogramm.

Hinweis

In erster Linie dient dieser Mechanismus zum Zurücksetzen von Verschiebungen, G-Funktionen, Parameterwerten, Variablen etc. Konkrete Bearbeitungsvorgänge sollten hier nicht programmiert sein.

8.6 Zyklen als globale oder lokale Unterprogramme (Aufruf L / LL CYCLE)

Zyklen stehen im NC-Kern in der Form globaler oder lokaler Unterprogramme zur Verfügung und ermöglichen spezielle Bearbeitungsvorgänge wie zum Beispiel das Tieflochbohren oder das Taschenfräsen. Die im Zyklus definierte Bearbeitungsaufgabe ist in allgemeiner Form geschrieben. Die Datenversorgung erfolgt beim Zyklusaufruf mit der Belegung der Versorgungsparameter.

Ein Zyklus ist unabhängig von der aktuell gültigen Ebene (G17, G18, G19) und unabhängig von den im NC-Kanal konfigurierten Achsnamen programmiert. Lediglich die Richtung, aus der die Bearbeitung in der aktuellen Ebene durchgeführt werden soll, muss beim Zyklusaufruf angegeben werden. Im Zyklus kann auf eine eigene gekapselte Gruppe von Parametern zugegriffen werden. Diese werden beim Zyklusaufruf mit Werten belegt.

Hierzu steht eine spezielle Syntax zur Verfügung, die durch das „@“-Zeichen charakterisiert ist. Dieses Zeichen wird in der Zyklenprogrammierung verwendet in Verbindung mit:

@Pxx	Versorgungsparameter im Zyklusaufruf und Zyklus
@X, @Y, @Z	Hauptachsen im Zyklus
@I, @J, @K	Mittelpunktskoordinaten im Zyklus
@S	Hauptspindel im Zyklus

Der **Zyklusaufruf** muss in einem eigenen NC-Satz ohne weitere NC-Befehle programmiert werden. Die Syntax besteht aus einem globalen oder lokalen Unterprogrammaufruf mit zusätzlicher Angabe der zyklusabhängigen Versorgungsparameter.

Syntax:

L | **LL CYCLE [NAME**=<*cycle*> [**MODAL_MOVE** / **MODAL_BLOCK**] **@P1**=.. **@P200**=.. { \ } **]**

NAME=<*cycle*>	Name des Zyklus (Dateiname)
MODAL_MOVE (MODAL, alte Syntax)	Modaler Zyklusaufruf. Nach jedem weiteren NC-Satz in der aktiven Programmebene, der **Bewegungsbefehle** enthält, wird der Zyklus erneut implizit ausgeführt.
MODAL_BLOCK	Modaler Zyklusaufruf. Nach jedem weiteren NC- Satz in der aktiven Programmebene wird der Zyklus erneut implizit ausgeführt. Bei folgenden NC-Befehlen wird der implizite blockmodale Zyklusaufruf unterdrückt: • Leerzeilen, Kommentarzeilen • Unterprogrammaufrufen (L, LL, M6, G8xx). • $-Befehlen ($GOTO, $IF, $FOR etc...) • Programmende M-Funktionen (M2, M30, M17; M29) Die Abwahl der modalen Wirkung von MODAL_MOVE / _BLOCK erfolgt am Ende der Programmebene ((M2, M30, M17; M29) oder durch #DISABLE MODAL CYCLE.
@Pxx=..	Auflistung der Versorgungsparameter. Es können maximal 200 @Pxx-Parameter vom Typ REAL oder STRING (ab V3.3079.25) übergeben werden. Schreib- und Lesezugriffe sind nur innerhalb eines Zyklus‘ erlaubt. Den @Pxx-Parametern können direkt Werte, beliebige Variablen, P-Parameter und mathematische Ausdrücke übergeben werden.
\	Trennzeichen ('Backslash') für Programmierung des Befehls über mehrere Zeilen

Versorgungsparameter - @P-Parameter

- Innerhalb der Klammerung ist **keine** Reihenfolge bei der Angabe der Schlüsselworte und Versorgungsparameter erforderlich. Bei der Programmierung muss der Anwender lediglich wissen, welche @P-Parameter für den Zyklus belegt werden müssen.
- Nicht benötigte @P-Parameter können weggelassen werden.
- Beim Lesezugriff auf einen nicht übergebenen @P-Parameter im Zyklus wird dieser implizit angelegt (Standard) und mit 0 (Null) initialisiert. Hierbei erhöht sich dann der Speicherbedarf. Dieses Verhalten kann der Anwender über P-CHAN-00463 ab der CNC-Version V3.1.3079.20 abschalten. Beim Lesezugriff auf nicht übergebene @P-Parameter wird dann der Fehler ID 20394 ausgegeben.
  In CNC-Versionen bis V3.1.3079.19 sind nicht programmierte @P-Parameter beim Lesezugriff mit 0 (Null) initialisiert.
- Ob ein @P-Parameter verwendet bzw. gültig ist, kann im Zyklus über die Variable V.G.@P[i].VALID ermittelt werden.
- Über die Funktionen IS_STRING und IS_NUMBER können @P-Parameter im Zyklus geprüft werden, ob es sich um einen String oder eine Zahl handelt, siehe Beispiel 4 [▶ 218]. (ab V3.3079.25)
- In CNC-Versionen bis V3.1.3079.19 bleiben die Versorgungsparameter bis zum programmierten Aufruf eines anderen Zyklus (L CYCLE.. oder G80.. [▶ 127]) erhalten. Ab der CNC-Version V3.1.3079.20 werden die Versorgungsparameter beim Schließen des Zyklus (M17 bzw. M29) gelöscht.
- Im Zyklusaufruf programmierte, aber im Zyklus selbst nicht verwendete @P-Parameter werden ignoriert.
- Ob das aktuelle Unterprogramm bzw. die aktuelle Programmebene ein Zyklus ist, kann über die Variable V.G.CYCLE_ACTIVE [▶ 602] ermittelt werden.

Achtung

Ein Zyklus ist eine abgeschlossene NC-Programmeinheit mit einer definierten Bearbeitungsaufgabe. Es wird empfohlen, die Schachtelung von Zyklen zu vermeiden, da dabei die Gefahr der Mehrfachbelegung von Versorgungsparametern besteht.

Hinweis

Versionsspezifisch sind in der Standardeinstellung im laufenden NC-Programm oder im Einzelsatzbetrieb während der Ausführung eines Zyklus in der Anzeige die Bearbeitungszeilen des Zyklus ausgeblendet oder sichtbar. Bei aktiver Ausblendung wird in dieser Zeit nur der Zyklusaufruf angezeigt.

Dieses Verhalten kann über den Kanalparameter P-CHAN-00211 geschaltet werden.

Hinweis

In TwinCAT-Systemen sind standardmäßig in der Anzeige alle Zykluszeilen sichtbar.

Erforderliche Definitionen vor Zyklusaufruf:

- Die vor dem Zyklusaufruf aktiven modalen G-Funktionen, sowie Kreisgeometriedaten und der aktuell aktive Vorschub (F-Wort) bleiben über den Zyklus hinaus erhalten. Dieses Verhalten kann über den Kanalparameter P-CHAN-00210 geschaltet werden.
- Modale G-Funktionen, die in Verbindung mit Achsnamen programmiert sind (z.B. G92, G98, G99, G100, G112, G130 ...) werden nicht wieder hergestellt, wenn diese im Zyklus selbst programmiert wurden.
- Die Bearbeitungsebene (G17, G18, G19) sollte vor dem Zyklusaufruf im übergeordneten NC-Programm definiert werden. Die senkrecht auf dieser Ebene stehende Achse ist dann bei Bohrzyklen die Achse, in der die Bohrung ausgeführt wird und bei Fräszyklen die Zustellachse für die Tiefe.
- Auch eventuelle Werkzeuggeometriekorrekturen (z.B. Längenkorrektur) müssen vor Aufruf des Zykluses angewählt werden.
- Die erforderlichen Werte für Vorschub, Spindeldrehzahl und Spindeldrehrichtung sind im übergeordneten NC-Programm festzulegen, es sei denn, entsprechende Versorgungsparameter sind im Zyklus vorhanden.
- Die in den Zyklen programmierten Spindelbefehle beziehen sich stets auf die aktive Hauptspindel des NC-Kanals. Es ist sicherzustellen, dass diese Hauptspindel vor dem Zyklusaufruf definiert wurde.
- Die Startposition für eine entsprechende Bohr- oder Fräsbearbeitung sowie die Orientierung des Werkzeuges sind stets vor dem Zyklusaufruf im übergeordneten NC-Programm anzufahren.

Abwahl eines modalen Zyklus:

Mit nachfolgendem NC-Befehl wird ein modal wirkender Zyklus (Schlüsselwort MODAL_MOVE oder MODAL_BLOCK im Zyklusaufruf) abgewählt. Der Befehl muss allein im NC-Satz programmiert werden.

Syntax:

#DISABLE MODAL CYCLE

Verfügbare Zyklen:

Folgende Zyklen stehen zur Verfügung:

- Bearbeitungszyklen
- Kalibrier- und Messzyklen
- und Zyklen zur kinematischen Optimierung

Programmierbeispiel

Verfügbare Zyklen

Am Beispiel eines Zyklusaufrufes zum Bohren (drilling.cyc) werden im Folgenden die verschiedenen Möglichkeiten der Parameterbelegung dargestellt.

Der Bohrzyklus drill.cyc benötigt folgende Versorgungsparameter:

@P1	Position der Rückzugsebene (absolut)
@P2	Position der Bearbeitungsebene (absolut)
@P3	Sicherheitsabstand (ohne Vorzeichen)
@P4	Endbohrtiefe (absolut) oder
@P5	Endbohrtiefe relativ zur Bearbeitungsebene (ohne Vorzeichen)

Zyklusaufruf mit konstanten Werten:

```
..
Nxx L CYCLE [NAME=drilling.cyc @P1=110 @P2=100 @P3=4 @P4=40]
..
oder mit Angabe einer relativen Bohrtiefe @P5:
..
Nxx L CYCLE [NAME=drilling.cyc @P1=110 @P2=100 @P3=4 @P5=60]
..
```

Zyklusaufruf mit Variablen:

```
Die Variablen müssen vor dem Zyklusaufruf definiert und mit Werten be-
legt werden.
..
#VAR
V.L.RPL = 110
V.L.WPL = 100
V.L.SDST = 4
V.L.DEP = 50
#ENDVAR
Nxx L CYCLE [NAME=drilling.cyc @P1= V.L.RPL @P2=V.L.WPL @P3=V.L.SDST
@P4=V.P.DEP]
..
```

Zyklusaufruf mit P-Parametern:

```
Die Parameter müssen vor dem Zyklusaufruf definiert und mit Werten be-
legt werden.
..
Nxx P10 = 110
Nxx P11 = 100
Nxx P15 = 4
Nxx P17 = 50

Nxx L CYCLE [NAME=drilling.cyc @P1= P10 @P2=P11 @P3=P15 @P4=P17]
..
```

Zyklusaufruf mit mathematischen Ausdrücken:

```
..
Nxx P20 = 100

Nxx L CYCLE [NAME=drilling.cyc @P1= 10+P20 @P2=2*50 @P3=5-1 @P4=P20/2]
..
```

Zyklusaufruf mit konstanten Werten; Reihenfolge der Parameter in der Klammerung beliebig:

```
..
Nxx L CYCLE [@P4=40 NAME=drilling.cyc @P2=100 @P3=4 @P1=110]
..
```

Zyklusaufruf mit konstanten Werten; Zyklus soll modal wirken:

```
..
Nxx L CYCLE [NAME=drilling.cyc @P1=110 @P2=100 @P3=4 @P4=40 MODAL_MOVE]
..
[Beispiel:]
%drill_main
N05 T1 D1
N10 M06
N15 G53 G17 G90 M3 S300 F200 S300
N16 G0 X0 Y0 Z0
N20 Z110
N30 X40 Y40 (Bohrposition 1)
N40 L CYCLE [NAME=drilling.cyc @P1=110 @P2=100 @P3=2 @P4=55 MODAL_MOVE]
N50 X60 Y60 (Bohrposition 2 und impliziter Zyklusaufruf, da modal)
N60 X100 Y60 (Bohrposition 3 und impliziter Zyklusaufruf, da modal)
N70 X100 Y20 (Bohrposition 4 und impliziter Zyklusaufruf, da modal)
#DISABLE MODAL CYCLE
N80 X0 Y0 M5
N100 M30
```

Hinweis

Hinweise für das Erstellen von Zyklen

Zyklen müssen so weit wie möglich allgemeingültig und unabhängig von den aktuell im NC-Kanal verwendeten Achsnamen und der Ebenenfestlegung programmiert sein. Zu diesem Zweck besteht im Zyklus die Möglichkeit, für die ersten drei Hauptachsen die ebenenunabhängigen „neutralen Achsnamen“ @X, @Y und @Z zu verwenden. Hierbei bedeutet:

@X	immer die erste Hauptachse
@Y	immer die zweite Hauptachse
@Z	immer die dritte Hauptachse

Beispiel 1: Achsen im Zyklus

```
Nxx G91 @X=@P1 @Y=@P2 @Z=@P3 F1000 G01
```

Analog dazu stehen auch bei der Kreisprogrammierung s.g. 'neutrale Mittelpunktskoordinaten' zur Verfügung. Hierbei bedeutet:

@I	immer die Mittelpunktskoordinate in der ersten Hauptachse
@J	immer die Mittelpunktskoordinate in der zweiten Hauptachse
@K	immer die Mittelpunktskoordinate in der dritten Hauptachse

Beispiel 2: Kreis im Zyklus

```
Nxx G91 G02 @X=@P1 @Y=@P2 @I=@P4 @J=@P5 F1000
```

Um auch bei der Spindelprogrammierung vom konfigurierten Spindelnamen unabhängig zu sein, kann im Zyklus die Hauptspindel immer mit dem neutralen Spindelnamen @S angesprochen werden.

@S	immer die Hauptspindel

Beispiel 3: Spindel im Zyklus

```
Nxx @S=1000  M3  (Hauptspindel cw mit 1000 U/min)
```

Beispiel 4: Prüfung @P-Parameter

Die übergebenen @P-Parameter können mit den Funktionen IS_STRING und IS_NUMBER überprüft werden.

```
%L cycle
( Check variables)
$IF IS_STRING[@P1] == TRUE
#MSG["Text: %s",@P1]
$ELSE
#MSG["Error no String"]
$ENDIF

$IF IS_NUMBER[@P2] == TRUE
#MSG["Zahl: %f",@P2]
$ELSE
#MSG["Error not a number"]
$ENDIF

M17

% Main
LL CYCLE [NAME=cycle @P1 ="String1" @P2=12.34]
M30
```

8.7 Aufruf von Satzfolgen (L SEQUENCE)

Satzfolgen sind zusammenhängende Programmteile (Sequenzen) oder einzelne NC-Sätze im aktuellen NC-Programm oder einem globalen Unterprogramm, die mit L SEQUENCE.. ein- oder mehrfach ausgeführt werden können.

Eine Satzfolge wird festgelegt durch Angabe von Start-/ Endemarkierungen über:

- Satznummern N.. oder
- Sprungmarken ([Stringlabel], analog zur Definition für $GOTO)

Hinweis

Jeder Aufruf einer Satzfolge ist mit einem Unterprogrammaufruf gleichzusetzen. Es gelten die gleichen Regeln zur Verschachtelungstiefe wie bei globalen Unterprogrammen.

Achtung

Kontextauswertung:

Der Programmkontext im Unterprogramm wird erst ab der ersten ausgeführten NC-Zeile der Satzfolge aufgebaut. Alle NC-Zeilen zuvor werden nicht ausgewertet. Vorher definierte Variablen/Koordinatensysteme, Parameter, modale Anweisungen usw. werden weder angelegt noch initialisiert. Sie sind also in der Satzfolge nicht bekannt bzw. verfügbar.

Insbesondere bei der Ausführung von Satzfolgen mit Steuersatzkonstrukten ($IF-$ELSE-$ENDIF, $SWITCH, ..) muss der Anwender daher selbst sicherstellen, dass diese durch den Ein- und Rücksprung konfliktfrei durchlaufen werden.

Syntax von L SEQUENCE bei Verwendung von Satznummern:

L SEQUENCE [[**NAME**=<*string*>] **N**.. [**N**..] [**REPEAT**=..] [**ENDTAG**] **]**

NAME=<*string*>	Name des aktuellen oder eines globalen Unterprogrammes, in dem die Satzfolge durchlaufen werden soll. Optional: ist kein Name programmiert, so wird die Satzfolge im aktuellen NC-Programm durchlaufen.
N..	Nummer des ersten auszuführenden Satzes (Startnummer, Beginn der Satzfolge)
N..	Nummer des letzten auszuführenden Satzes (Rücksprungnummer, Ende der Satzfolge). Optional, sind beide Satznummern identisch oder ist nur die Startnummer angegeben, so wird nur dieser Satz ausgeführt.
REPEAT=..	Anzahl der Wiederholungen einer Satzfolge, positive Ganzzahl ≥ 1. Optional, ohne Angabe von REPEAT wird die Satzfolge einmal durchlaufen.
ENDTAG	Markiert den Aufruf L SEQUENCE selbst als ein zusätzliches gültiges Ende der Satzfolge. Optional, bei gleichzeitiger Angabe von ENDTAG und einer Rücksprungnummer N.. gilt das zuerst gefundene Sequenzende.

Die Steuerung sucht in dem angegebenen NC-Programm (kann auch das gleiche Programm sein, das den Befehl aufruft) nach den programmierten N-(Satz)nummern. Die beiden N-Nummern markieren den ersten und letzten auszuführenden NC-Satz der Satzfolge; NC-Sätze außerhalb dieser Satzfolge werden nicht ausgeführt.

Es wird empfohlen, die NC-Zeilen eindeutig und in aufsteigender Form zu nummerieren.

Start- und Rücksprungnummer können im Befehl auch vertauscht programmiert werden. Im NC-Programm wird jedoch immer von der kleineren N-Nummer bis zur größeren N-Nummer die Satzfolge durchlaufen.

Werden Start- oder Rücksprungnummer nicht gefunden, so erfolgt die Ausgabe einer Fehlermeldung.

Soll die Satzfolge mehrfach ausgeführt werden (REPEAT > 1), so wird am Ende der Satzfolge wieder auf der Startnummer aufgesetzt. Nachdem alle Durchläufe abgearbeitet sind, erfolgt der Rücksprung aus der Satzfolge und die Fortsetzung des weiteren Programmablaufes.

Wenn im Befehl nur eine N-Nummer angegeben wurde, dann wird nur diese Zeile durchlaufen. Dies entspricht dem Aufruf mit zwei gleichen N-Nummern.

Der Aufruf L SEQUENCE kann selbst innerhalb der durch die N-Nummern definierten Satzfolge liegen. Beim erneuten Einlesen des gleichen Aufrufs sind zwei Reaktionen möglich:

Ohne ENDTAG wird der erneute Aufruf ignoriert und die Satzfolge bis zur Rücksprungnummer ausgeführt.

Mit ENDTAG ist L SEQUENCE als gültiges Sequenzende markiert und die Satzfolge wird beendet.

Programmierbeispiel

Aufruf von Satzfolgen mit Satznummern (L SEQUENCE)

Satzfolge zwischen 2 Satznummern 1 mal wiederholen:

```
...
N20 ...                              ;Startnummer
...
N50 ...                              ;Rücksprungnummer
...
N80 L SEQUENCE [N20 N50]             ;oder..

N80 L SEQUENCE [N20 N50 REPEAT=1]
...
```

Satzfolge zwischen 2 Satznummern mehrfach wiederholen:

```
...
N20 ...                              ;Startnummer
...
N50 ...                              ;Rücksprungnummer
...
N80 L SEQUENCE [N20 N50 REPEAT=4]
...
```

Satzfolge zwischen 2 Satznummern. Geklammerter eigener Sequenzaufruf wird bei Ausführen der Satzfolge ignoriert, da ENDTAG nicht gesetzt ist:

```
...
N20 ...                          ;Startnummer
...
N40 L SEQUENCE [N20 N80]
...
N80 ...                          ;Rücksprungnummer
...
```

Satzfolge zwischen 2 Satznummern. Geklammerter eigener Sequenzaufruf ist zuerst gefundenes Ende der Satzfolge, da ENDTAG gesetzt ist:

```
...
N20 ...                          ;Startnummer
...
N40 L SEQUENCE [N20 N80 ENDTAG]
...
N80 ...                          ;Rücksprungnummer
...
```

Satzfolge zwischen 2 Satznummern mit ENDTAG. ENDTAG nicht relevant, da Rücksprungnummer vor Sequenzaufruf liegt:

```
...
N20 ...                          ;Startnummer
...
N50 ...                          ;Rücksprungnummer
...
N80 L SEQUENCE [N20 N50 REPEAT=4 ENDTAG]
...
```

Einzelnen NC-Satz mehrfach wiederholen:

```
...
N20 ...                          ;Startnummer
...
N80 L SEQUENCE [N20 REPEAT=4]    ;oder..

N80 L SEQUENCE [N20 N20 REPEAT=4]
...
```

Satzfolge zwischen 1 Satznummer und ENDTAG:

```
...
N20 ...                          ;Startnummer
...
N80 L SEQUENCE [N20 ENDTAG]
...
```

Satzfolge zwischen 2 Satznummern mehrfach wiederholen. Sequenzaufruf liegt vor der Satzfolge:

```
...
N80 L SEQUENCE [N100 N150 REPEAT=4]
...
N100 ...                          ;Startnummer
...
N150 ...                          ;Rücksprungnummer
```

Geschachtelter mehrfacher Aufruf von Satzfolgen zwischen Satznummern:

```
...
N40 L SEQUENCE [N60 N150 REPEAT=2] ;Sequenzaufruf 1
...
N60 ...                           ;Startnummer 1
...
N90 ...                           ;Startnummer 2
...
N120 ...                           ;Rücksprungnummer 2
...
N130 L SEQUENCE [N90 N120 REPEAT=4] ;Sequenzaufruf 2
...
N150 ...                           ;Rücksprungnummer 1
...
```

Satzfolge zwischen 2 Satznummern in einem globalen Unterprogramm mehrfach wiederholen:

```
...
N20 ...
...
N80 L SEQUENCE [NAME="glob_1.nc" N50 N150 REPEAT=4]
...
```

Geschachtelter mehrfacher Aufruf von Satzfolgen im aktuellen Programm und einem globalen Unterprogramm zwischen Satznummern:

```
...
N20 L SEQUENCE [N60 N150 REPEAT=2] ;Sequenzaufruf 1
...
N60 ...                           ;Startnummer 1
...
N80 L SEQUENCE [NAME="glob_1.nc" N50 N150 REPEAT=3]   ;Sequenzaufruf 2
...
N150 ...                          ;Rücksprungnummer 1
...
```

Alternativ kann eine Satzfolge auch über Sprungmarken programmiert werden:

Syntax von L SEQUENCE bei Verwendung von Sprungmarken (Stringlabel):

L SEQUENCE [[**NAME**=<*string*>] **[**<*START*>**]** [**[**<*END*>**]**] [**REPEAT**=..] [**ENDTAG**] **]**

NAME=<*string*>	Name des aktuellen oder eines globalen Unterprogrammes, in dem die Satzfolge durchlaufen werden soll. Optional: ist kein Name programmiert, so wird die Satzfolge im aktuellen NC-Programm durchlaufen.
[<*START*>]	Startmarke des ersten auszuführenden Satzes (Beginn der Satzfolge)
[<*END*>]	Endmarke des letzten auszuführenden Satzes (Rücksprung, Ende der Satzfolge). Optional, sind beide Marken identisch oder ist nur die Startmarke angegeben, so wird nur dieser Satz ausgeführt.
REPEAT=..	Anzahl der Wiederholungen einer Satzfolge, positive Ganzzahl ≥ 1. Optional, ohne Angabe von REPEAT wird die Satzfolge einmal durchlaufen.
ENDTAG	Markiert den Aufruf L SEQUENCE selbst als ein zusätzliches gültiges Ende der Satzfolge. Optional, bei gleichzeitiger Angabe von ENDTAG und einer Endmarke gilt das zuerst gefundene Sequenzende.

Die Steuerung sucht in dem angegebenen NC-Programm (kann auch das gleiche Programm sein, das den Befehl aufruft) nach den programmierten Sprungmarken. Die beiden Sprungmarken markieren den ersten und letzten auszuführenden NC-Satz der Satzfolge; - NC-Sätze außerhalb dieser Satzfolge werden nicht ausgeführt.

Sprungmarken werden am Satzanfang oder direkt nach der Satznummer gesetzt. Werden Start- oder Rücksprungmarke nicht gefunden, so erfolgt die Ausgabe einer Fehlermeldung.

Soll die Satzfolge mehrfach ausgeführt werden (REPEAT > 1), so wird am Ende der Satzfolge wieder auf der Startmarke aufgesetzt. Nachdem alle Durchläufe abgearbeitet sind, erfolgt der Rücksprung aus der Satzfolge und die Fortsetzung des weiteren Programmablaufes.

Wenn im Befehl nur eine Startmarke angegeben wurde, dann wird nur diese Zeile durchlaufen. Dies entspricht dem Aufruf mit zwei gleichen Sprungmarken.

Der Aufruf L SEQUENCE kann selbst innerhalb der durch die Sprungmarken definierten Satzfolge liegen. Beim erneuten Einlesen des gleichen Aufrufs sind zwei Reaktionen möglich:

Ohne ENDTAG wird der erneute Aufruf ignoriert und die Satzfolge bis zur Rücksprungmarke ausgeführt.

Mit ENDTAG ist L SEQUENCE als gültiges Sequenzende markiert und die Satzfolge wird beendet.

Programmierbeispiel

Aufruf von Satzfolgen mit Sprungmarken (L SEQUENCE)

Satzfolge zwischen 2 Sprungmarken 1 mal wiederholen:

```
...
N20 [STARTLBL] ...                              ;Startmarke
...
N50 [ENDLBL] ...                                ;Rücksprungmarke
...
N80 L SEQUENCE [[STARTLBL] [ENDLBL]] ;oder..

N80 L SEQUENCE [[STARTLBL] [ENDLBL] REPEAT=1]
...
```

Satzfolge zwischen 2 Sprungmarken mehrfach wiederholen:

```
...
N20 [STARTLBL] ...                                ;Startmarke
...
N50 [ENDLBL] ...                                  ;Rücksprungmarke
...
N80 L SEQUENCE [[STARTLBL] [ENDLBL] REPEAT=4]

...
```

Satzfolge zwischen 2 Sprungmarken. Geklammerter eigener Sequenzaufruf wird bei Ausführen der Satzfolge ignoriert, da ENDTAG nicht gesetzt ist:

```
...
N20 [STARTLBL] ...                                ;Startmarke
...
N40 L SEQUENCE [[STARTLBL] [ENDLBL]]
...
N80 [ENDLBL]...                                   ;Rücksprungmarke
...
```

Satzfolge zwischen 2 Sprungmarken. Geklammerter eigener Sequenzaufruf ist zuerst gefundenes Ende der Satzfolge, da ENDTAG gesetzt ist:

```
...
N20 [STARTLBL]...                                 ;Startmarke
...
N40 L SEQUENCE [[STARTLBL] [ENDLBL] ENDTAG]
...
N80 [ENDLBL] ...                                  ;Rücksprungmarke
...
```

Satzfolge zwischen 2 Sprungmarken mit ENDTAG. ENDTAG nicht relevant, da Rücksprungmarke vor Sequenzaufruf liegt:

```
...
N20 [STARTLBL]...                                  ;Startmarke
...
N50 [ENDLBL] ...                                 ;Rücksprungmarke
...
N80 L SEQUENCE [[STARTLBL] [ENDLBL] REPEAT=4 ENDTAG]
...
```

Einzelnen NC-Satz mehrfach wiederholen:

```
...
N20 [STARTLBL] ...                                ;Startmarke
...
N80 L SEQUENCE [[STARTLBL] REPEAT=4]              ;oder..

N80 L SEQUENCE [[STARTLBL] [STARTLBL] REPEAT=4]
...
```

Satzfolge zwischen 1 Sprungmarke und ENDTAG:

```
...
N20 [STARTLBL] ...                                ;Startmarke
...
N80 L SEQUENCE [[STARTLBL] ENDTAG]
...
```

Satzfolge zwischen 2 Sprungmarken mehrfach wiederholen. Sequenzaufruf liegt vor der Satzfolge:

```
...
N80 L SEQUENCE [[STARTLBL] [ENDLBL] REPEAT=4]
...
N100 [STARTLBL] ...                                ;Startmarke
...
N150 [ENDLBL] ...                                 ;Rücksprungmarke
```

Geschachtelter mehrfacher Aufruf von Satzfolgen zwischen Sprungmarken:

```
...
N40 L SEQUENCE [[STARTLBL1] [ENDLBL1] REPEAT=2] ;Sequenzaufruf 1
...
N60 [STARTLBL1] ...                                 ;Startmarke 1
...
N90 [STARTLBL2] ...                                 ;Startmarke 2
...
N120 [ENDLBL2] ...                                  ;Rücksprungmarke 2
...
N130 L SEQUENCE [[STARTLBL2] [ENDLBL2] REPEAT=4]  ;Sequenzaufruf 2
...
N150 [ENDLBL1]...                                   ;Rücksprungmarke 1
...
```

Satzfolge zwischen 2 Sprungmarken in einem globalen Unterprogramm mehrfach wiederholen:

```
...
N20 ...
...
N80 L SEQUENCE [NAME="glob_1.nc" [SUP1] [EUP1] REPEAT=4]
...
```

Geschachtelter mehrfacher Aufruf von Satzfolgen im aktuellen Programm und einem globalen Unterprogramm zwischen Sprungmarken:

```
...
N20 L SEQUENCE [[STARTLBL] [ENDLBL] REPEAT=2]      ;Sequenzaufruf 1
...
N60 [STARTLBL] ...                                  ;Startmarke 1
...
N80 L SEQUENCE [NAME="glob_1.nc" [SUP1] [EUP1] REPEAT=3] ;Sequenzaufruf
2
...
N150 [ENDLBL]...                                   ;Rücksprungmarke 1
...
```

Das Ende einer Satzfolge erfolgt bei der Programmierung mit Satznummern durch Angabe der Rücksprungnummer, bei der Programmierung mit Sprungmarken durch Angabe der Endemarke oder im Aufruf L SEQUENCE selbst durch Angabe des Schlüsselwortes ENDTAG.

Alternativ kann eine Satzfolge auch durch Angabe des NC-Befehls #SEQUENCE END beendet werden.

Syntax der Standardendemarke:

#SEQUENCE END

Hinweis

Zusammen mit #SEQUENCE END dürfen im gleichen NC-Satz mit Ausnahme einer Satznummer keine weiteren NC-Befehle programmiert sein.

Im Aufruf L SEQUENCE ist dann die spezifische Angabe einer Endemarke nicht erforderlich. Es muss nur die Startmarke und die Zusatzinformation gesetzt sein, dass diese Startmarke keinen einzelnen NC-Satz, sondern den Beginn einer Satzfolge adressiert. Programmname, Anzahl der Wiederholungen und ENDTAG sind ebenfalls optional zulässig.

Syntax von L SEQUENCE mit Satznummer oder Sprungmarke in Kombination mit #SEQUENCE END:

L SEQUENCE [[**NAME=**<*string*>] **N..** | **[**<*START*>**] BEGIN** [**REPEAT**=..] [**ENDTAG**] **]**

NAME=<*string*>	Name des aktuellen oder eines globalen Unterprogrammes, in dem die Satzfolge durchlaufen werden soll. Optional: ist kein Name programmiert, so wird die Satzfolge im aktuellen NC-Programm durchlaufen.
N..	Nummer des ersten auszuführenden Satzes (Beginn der Satzfolge)
[<*START*>]	Startmarke des ersten auszuführenden Satzes (Beginn der Satzfolge)
BEGIN	Startnummer oder Startmarke adressiert den Beginn einer Satzfolge. #SEQUENCE END oder ENDTAG markieren das Sequenzende.
REPEAT=..	Anzahl der Wiederholungen einer Satzfolge, positive Ganzzahl ≥ 1. Optional, ohne Angabe von REPEAT wird die Satzfolge einmal durchlaufen.
ENDTAG	Markiert den Aufruf L SEQUENCE selbst als ein zusätzliches gültiges Ende der Satzfolge. Optional, bei gleichzeitiger Angabe von ENDTAG und #SEQUENCE END gilt das zuerst gefundene Sequenzende.

Programmierbeispiel

Aufruf von Satzfolgen (L SEQUENCE) mit Endemarke #SEQUENCE END

Satzfolge zwischen Startnummer und Endemarke 1 mal ausführen:

```
...
N20 ...                              ;Startnummer
...
N50 #SEQUENCE END                    ;Endemarke
...
N80 L SEQUENCE [N20 BEGIN]
...
```

Satzfolge zwischen Startmarke und Endemarke mehrfach wiederholen:

```
...
N20 [STARTLBL]...                    ;Startmarke
...
N50 #SEQUENCE END                    ;Endemarke
...
N80 L SEQUENCE [[STARTLBL] BEGIN REPEAT=4]
...
```

Satzfolge zwischen Startmarke und Endemarke 1 mal ausführen. Sequenzaufruf liegt vor der Satzfolge:

```
...
N80 L SEQUENCE [[STARTLBL] BEGIN]
...
N100 [STARTLBL] ...                   ;Startmarke
...
N150 #SEQUENCE END                    ;Endemarke
```

Satzfolge zwischen Startnummer und Endemarke in einem globalen Unterprogramm mehrfach wiederholen:

```
...
N80 L SEQUENCE [NAME="glob_1.nc" N50 BEGIN REPEAT=4]
...
```

Satzfolge zwischen Startmarke und Endemarke. Geklammerter eigener Sequenzaufruf ist zuerst gefundenes Ende der Satzfolge, da ENDTAG gesetzt ist:

```
...
N20 [STARTLBL]...                                ;Startmarke
...
N40 L SEQUENCE [[STARTLBL] BEGIN ENDTAG]
...
N80 #SEQUENCE END                               ;Endemarke
...
```

9 Parameter und Parameterrechnung (P)

Parameter können in NC-Programmen als Platzhalter für Zahlenwerte verwendet werden. Der Vorteil von Parametern liegt darin, dass der Wert eines Parameters während des Programmablaufes verändert werden kann. Dies erlaubt die Erstellung flexibler NC-Programme.

Für die Parameterbezeichnung wird "P" gefolgt von einer Zahl ohne Leerzeichen verwendet.

Programmierbeispiel

Parameter und Parameterrechnung

In einem Unterprogramm, z.B. einem Bohrzyklus, werden anstelle von Koordinatenwerten (Bohrtiefe, Bohrvorschub, Verweilzeit usw.) Parameter verwendet. Die Parameter werden dann im jeweils aufrufenden Hauptprogramm mit den endgültigen Werten belegt:

Für das globale Unterprogramm

%4712(Bohren,Plansenken)

sind folgende Parameter zu definieren:

P10 -Referenzebene = Rückzugsebene .

P11 -Bohrtiefe

P12 -Verweilzeit

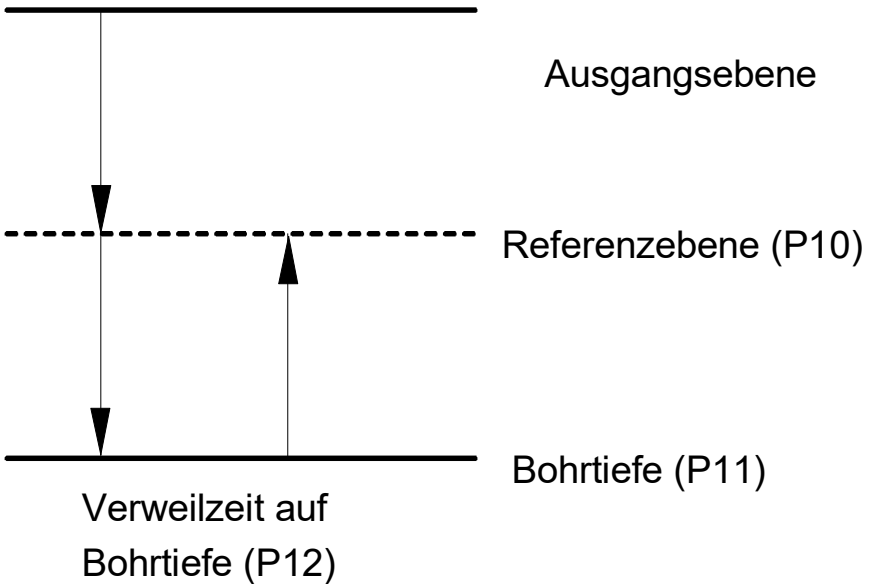


Abb. 67: Anwendungsbeispiel zur Parameterrechnung

Der Aufruf im Hauptprogramm sieht dann wie folgt aus:

```
:
N100 P10=20.5 P11=12.6 P12=1.2
N110 L4712
:
```

Syntax:

P.. einfacher Parameter

P.. Der Parameterindex muss immer größer Null sein, kann jedoch einen beliebigen Wert annehmen. Die maximale Anzahl zu verwendender Parameter im Kanal ist fest vorgegeben [6] [▷ 894]-6.19.

Zusätzlich zu einfachen Parametern sind auch Parameter-Arrays (z.B. P100[50]) erlaubt. Die Dimension des Arrays ist fest vorgegeben [6] [▷ 894]-6.20 .

Syntax:

P..**[**<*idx*>**]** {**[**<*idx*>**]**} Parameterarrays

Bei entsprechender Parametrierung mit P-CHAN-00067 sind die P-Parameter programmübergreifend wirksam.

Parameter können im NC-Programm entweder in einem Deklarationsblock angelegt (und ggf. initialisiert) werden, der mit **#VAR** beginnt und mit **#ENDVAR** abgeschlossen wird, oder implizit beim ersten Schreibzugriff. Parameter-Arrays müssen allerdings immer in einem Deklarationsblock angelegt werden.

Zur besseren Übersicht kann die Initialisierung eines Parameter-Arrays mit dem „\“-Zeichen auch über mehrere NC-Zeilen geschrieben werden.

Syntax:

#VAR Beginn des Deklarationsblockes

:

:

:

#ENDVAR Ende des Deklarationsblockes

Programmierbeispiel

#VAR und #ENDVAR

```
#VAR
P10[3][6] = [10,11,12,13,14,15, \
             20,21,22,23,24,25, \
             30,31,32,33,34,35 ]
P20[3][4] = [40,41,42,43, 50,51,52,53, 60,61,62,63]
P100
#ENDVAR

P200 = 10 P201=11
:
```

Hinweis

Der Zugriff auf Parameterarrays beginnt bei Index 0! Mit obigem Beispiel liefert der Zugriff P10[0] [5] somit den Wert 15.

Parameter und Parameter-Arrays können im NC-Programm auch wieder gelöscht werden. Hierzu steht der Befehl #DELETE zur Verfügung:

Syntax:

#DELETE P.. {, P..}

Programmierbeispiel

#DELETE P..

```
#DELETE P10, P20, P100, P200, P201
```

Des Weiteren stehen die SIZEOF- und die EXIST-Funktion zur Verfügung (siehe Kapitel Arithmetische Ausdrücke <expr> [▶ 32]), die die Dimensionsgröße von Parameter-Arrays ermitteln und die Existenz von Parametern überprüfen.

Parameter erhalten ihre Wertzuweisung durch das NC-Programm, z.B. P12=0.12. Sie ermöglichen auch die Verarbeitung von steuerungs- oder prozessabhängigen Werten des Steuerungssystems, wie z.B.:

- aktuelle Spindeldrehzahlen,
- Momente in den Antrieben,
- Werte externer Messvorrichtungen,
- Werte von Wärme- oder Kraftsensoren,
- Tastatureingaben über das Bedienmenü

usw.

Anstelle der direkten Zuweisung von Zahlen können auch verkettete arithmetische Ausdrücke verwendet werden (siehe Kapitel Mathematische Ausdrücke [▶ 31]) Bei der Eingabe gelten die bekannten mathematischen Regeln wie:

- Punkt-vor-Strich-Rechnung,
- die Klammer-Regel, wobei aber eckige Klammern "[ ]" zu verwenden sind.

9.1 Zuweisung von Parametern zu Adressbuchstaben

Die Syntax bei der Zuweisung von Parametern zu den Achsbezeichnungen ist:

<Achsname> **P**..

<Achsname>	Achsbezeichnung der Achse
P..	Zugewiesener Parameter

P.. kann auch durch einen mathematischen Ausdruck gebildet werden.

Programmierbeispiel

Zuweisung von Parametern zu Adressbuchstaben

```
XP1*SIN[P2*30]
```

9.2 Indirekte Parameter

Indirekte Parameter werden in arithmetischen Ausdrücken und bei Zuweisungen wie direkte Parameter verwendet.

Sowohl die direkte (Pnn) als auch die indirekte Programmierung (PPnn) erfolgt über P. Hierbei gilt bei Verwendung von indirekten Parametern:

PPnn zeigt auf den Parameter Pnn.

Bei der Initialisierung eines PPnn wird die Adresse eines Pnn zugewiesen. Auch die Verwendung von PPP... ist möglich.

Beispiel

Indirekte Parameter

Mit P120=10 wird der Wert 10 in den Parameter 120 geladen. Die Anweisung PP120=123.456 ordnet aber diesen Wert dem Parameter zu, dessen Adresse in P120 steht, also P10. Entsprechend führt PP121=SQRT[2.0] zu folgendem Ergebnis:

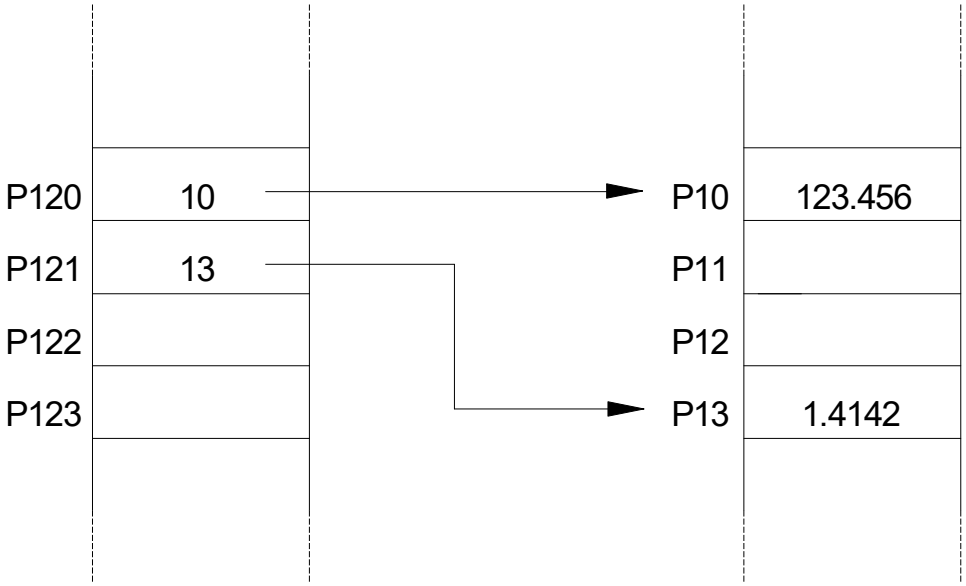


Abb. 68: Veranschaulichung der Wirkung indirekter P-Parameter

Die Verwendung indirekter Parameter erlaubt es, z.B. ganze Felder von Parametern zu belegen:

Programmierbeispiel

Indirekte Parameter

```
Belegung der P-Parameter P20 und P40 mit 50
:
N110 P1 = 20  P2 = 40
N120 PP1 = 50
N130 PP2 = PP1
Belegung der P-Parameter P15 bis P25 mit 0.0
N110 $FOR P1 = 10,20,1
N120 P[P1 + 5] = 0.0
N130 $ENDFOR
:
```

10 Anweisungen zur Beeinflussung des NC-Programmablaufes

Eine vollständige Liste der Steuersatzanweisungen findet sich in der Befehlsübersicht im Anhang unter Steuersatzanweisungen ($..) [▶ 883].

Die Syntax für Steuersatzanweisungen lautet:

$<*Anweisung*>

<*Anweisung*> Steuersatzstrings gemäß Kapitel Bedingte Sprünge [▶ 236]. Zwischen $ und <*Anweisung*> sind keine Leerzeichen erlaubt.

Anweisungen zur Beeinflussung des Programmablaufs (Steuersätze) erlauben die Realisierung von:

- Bedingten Sprüngen, um z.B. in Abhängigkeit eines Messwertes optionale Bearbeitungsschritte auszulösen,
- Zählerschleifen, um mehrfach wiederkehrende Bearbeitungsschritte, z.B. beim Zeilenfräsen oder beim Bohren von Lochkreisen, einfach zu programmieren,
- Schleifen mit Laufbedingung, um mehrfach wiederkehrende Bearbeitungsschritte solange durchlaufen zu lassen, bis eine Abbruchbedingung erfüllt ist. Z.B. soll das Zustellen des Werkzeuges und ein Abspanvorgang solange durchgeführt werden, bis ein bestimmter Koordinatenwert erreicht ist. Schleifen können bei fehlender bzw. nicht erfüllter Laufbedingung als Endlosschleife programmiert werden.

Bei der Verwendung von Steuersätzen gelten die folgenden Regeln:

- In einem NC-Satz darf ausschließlich ein Steuersatz vorhanden sein!
- Steuersatzkonstrukte dürfen geschachtelt werden. Die Schachtelungstiefe ist fest vorgegeben.
- Vor einem Steuersatz darf nur die Satznummer und "/" stehen.
- Im ungültigen Zweig einer Steuersatzanweisung wird auf Syntaxfehler von Satznummern und weiteren (geschachtelten) Steuersatzanweisungen überprüft (siehe Beispiele der IF-ELSE-Verzweigung).

Programmierbeispiel

Syntaxüberprüfung im ungültigen Zweig:

```
N10 $IF 0
N20 XY              (Hier findet keine Syntaxprüfung statt)
N30 $ENDIF
N10 $IF 0
N20 ...
N30 $IF XY          (Syntaxprüfung, da geschachtelte Steuersatzanweisung;)
                    (Fehlermeldung, da unbekannter Term)
N40 ...
N50 $ENDIF
N60 $ENDIF
N10 $IF 0
NXY                 (Syntaxprüfung bei Angabe der Satznummer;)
                    (Fehlermeldung, da unbekannter Term)
N30 $ENDIF
```

Wegen Ungenauigkeiten bei der Berechnung und internen Darstellung von Parametern können Vergleichsoperationen (siehe Kap. Arithmetische Ausdrücke <expr> [▶ 32]) in Steuersatzanweisungen zu einem falschen Ergebnis führen. Deshalb sollte in Zweifelsfällen nicht auf genaue Werte, sondern einen Toleranzbereich abgeprüft werden.

Programmierbeispiel

FALSCH:

```
N10 $FOR P1 = 0, 10, 1
N20 P2 = P2 + 0.01
N30 $ENDFOR
N40 $IF P2 == 0.1                (Wegen Rechenungenauigkeiten kann P2)
N50 ...                          (ungleich 0.1 sein, sodass der $ELSE-)
N60 $ELSE                        (Zweig durchlaufen wird)
N70 G04 X20
N80 $ENDIF
```

RICHTIG:

```
N10 $FOR P1 = 0, 10, 1
N20 P2 = P2 + 0.01
N30 $ENDFOR
N40 $IF ABS[P1 - 0.1] <= .000001 (Überprüfung auf einen für die)
N50 G04 X5                       (NC-Fertigung unproblematischen)
N60 $ELSE                        (Toleranzbereich; $IF-Zweig wird)
N70 ...                          (durchlaufen)
N80 $ENDIF
```

10.1 Bedingte Sprünge

10.1.1 Die IF – ELSE – Verzweigung

Für die IF-ELSE-Verzweigung werden folgende Steueranweisungen benutzt:

$IF, $ELSE, $ELSEIF, $ENDIF.

Syntax:

Die Verzweigung beginnt immer mit

$IF=..

und endet immer mit

$ENDIF

Die Steueranweisungen

$ELSE
und
$ELSEIF

sind optional und dienen dazu, Mehrfachverzweigungen aufzubauen.

Achtung

Die Bedingung im $IF-Steuersatz wird geprüft, indem der mathematische Ausdruck auf "wahr" bzw. "nicht wahr" geprüft wird (TRUE und FALSE). Um auch dezimale Größen verwenden zu können, gilt die Bedingung als erfüllt (TRUE), wenn…

...der Betragswert des mathematischen Ausdruckes > oder = 0.5 ist.

Programmierbeispiel

Die IF – ELSE – Verzweigung

```
N10 ...
N20 $IF P1       Nur wenn |P1| größer oder gleich 0,5 ist, werden die
                 Anweisungen N30 bis N50 abgearbeitet.
N30 ...
N40
N50
N60 $ENDIF
```

Aber auch folgendes ist möglich:

```
N10 ...
N20 $IF P1 >= 0.5     Nur wenn P1 größer oder gleich 0.5 ist, werden die
                      Anweisungen N30 bis N50 abgearbeitet
N30 ...
N40
N50
N60 $ENDIF
```

Oder aber:

```
N10 ...
N20 $IF P1 > P2       Nur wenn P1 größer P2 ist, werden die Anweisungen
                      N30 bis N50 abgearbeitet, ansonsten N70 bis N90
N30 ...
N40 ...
N50 ...
N60 $ELSE
N70 ...
N80 ...
N90 ...
N100 $ENDIF
```

Die Verwendung von ELSEIF erlaubt:

```
N10 ...
N20 $IF P1 == 0      Nur wenn P1 gleich 0 ist, werden die Anweisungen N30
                     bis N50 abgearbeitet, ansonsten wird in der
                     $ELSEIF-Bedingung geprüft,ob P2 >= 0.5 ist und
                     entspr. N70 bis N90 bzw. N110 bis N130 abgearbeitet.
N30 ...
N40
N50
N60 $ELSEIF P2>=0.5  Die $ELSEIF - Bedingung dient zum Aufbau
                     ineinandergeschachtelter Verzweigungen.
N70 ...
N80
N90
N100 $ELSE
N110 ...
N120
N130
N140 $ENDIF
```

Achtung

Entsprechend der Programmiersprache C existiert auch hier der Unterschied in der Syntax zwischen

Zuweisung: P5 = 3

und

Vergleich: $IF P5 == 3

Für die Version 2.3 und frühere gilt: Weil bei mathematischen Ausdrücken immer die Folge...

Operator -> Term -> Operator -> Term -> usw.

...erwartet wird, müssen bei Vergleichsoperationen die Ausdrücke geklammert werden, denen ein Minuszeichen vorsteht ("-" wird als Operator interpretiert).

Programmierbeispiel

```
$IF P1 >= -5      falsch, da Term->Operator->Operator->Term

$IF P1 >= [-5]    richtig, da Term->Operator->Term->Operator->Term
```

10.1.2 Der Sprungverteiler ($SWITCH)

Der Sprungverteiler erlaubt in Abhängigkeit eines arithmetischen Ausdrucks, verschiedene NC-Programmvarianten abzuarbeiten.

Für den Sprungverteiler werden die Steueranweisungen $SWITCH, $CASE, $DEFAULT, $ENDSWITCH verwendet.

Syntax:

Der Sprungverteiler beginnt immer mit:

$SWITCH *<expr1>*

gefolgt von mehreren

$CASE *<expr2>*

...

$BREAK

optional gefolgt von:

$DEFAULT

und endet immer mit

$ENDSWITCH

Programmierbeispiel

Der Sprungverteiler

```
N100 $SWITCH P1=INT [P1*P2/P3] Wenn das Ergebnis des arithmetischen
                               Ausdrucks gleich 1 ist, dann werden die
N110 $CASE 1                   Sätze nach $CASE 1 abgearbeitet
(N120-140)
N120 ...
N130
N140 $BREAK
N150 $CASE P2                  Ist das Ergebnis gleich P2, dann werden
                               die Sätze N160 ..N170 abgearbeitet.
N160 ...
N170 $BREAK
N300 $CASE n
N320 ...
N330 $BREAK
N350 $DEFAULT                  Der $DEFAULT Satz ist optional und dient
                               dazu, die NC-Sätze N360-N380 abzuarbeiten
N360 ...                       ,wenn das Ergebnis des $SWITCH Satzes
N370                           keinem der $CASE-Fälle entsprochen hat.
N380
N390 $ENDSWITCH
```

Hinweis

Der Vergleich des Ausdruckes <expr1> und <expr2> wird mit der internen REAL-Darstellung durchgeführt. Dabei werden die beiden Ausdrücke als gleich bewertet, wenn die Betragsdifferenz < 0.001 ist.

Die Ausdrücke <expr1> und <expr2> können auch negative Werte annehmen.

10.1.3 Die $GOTO-Anweisung

Diese Funktionalität bietet neben der Unterprogrammtechnik oder der Verwendung von Steuersatzanweisungen ($IF, $FOR...) eine weitere Möglichkeit der Verzweigung in andere Programmteile. Durch das Setzen von Sprungmarken (s.g. Labeln) im NC-Programm kann durch Aufruf des GOTO-Befehls an jede Stelle des NC-Programmes gesprungen werden.

Syntax:

Es gibt zwei Möglichkeiten der Programmierung von Sprunganweisungen:

Expression – **Label:**

N<*satz_nr*> **:**	Definition Sprungziel, Satznummer mit Doppelpunkt
$GOTO N<*satz_nr*>	Sprungaufruf

String – **Label:**

[<*name*>**]**	Definition Sprungziel, Name in eckigen Klammern
$GOTO [<*name*>**]**	Sprungaufruf

Eigenschaften:

- Der Aufruf $GOTO kann im NC-Programm vor oder nach der Labeldefinition erfolgen. Die Suche nach einem Label wird vorwärts und rückwärts im Programm durchgeführt.
- Labelaufruf und Definition müssen immer in der gleichen Programmebene stattfinden (programmlokal). Programmsprünge zwischen Hauptprogramm und einem Unterprogramm sowie Sprünge zwischen Unterprogrammen sind nicht zulässig (siehe folgende Abb.).
- In Haupt-/ und Unterprogrammen können gleiche Label definiert werden.
- Es kann von mehreren Stellen im NC-Programm zum gleichen Label gesprungen werden.
- Eine $IF-Abfrage kann in der gleichen NC-Zeile mit einem $GOTO kombiniert werden. In diesem Fall darf kein zugehöriges $ELSE - $ENDIF programmiert werden.
- Vor und nach dem $GOTO-Befehl dürfen andere NC-Befehle in der gleichen Zeile programmiert werden. Der Sprung erfolgt jedoch erst als letzte Aktion im NC-Satz.
- Sprünge von außen in beliebige Ebenen eines $IF-$ELSE-$ENDIF- Steuersatzblockes sowie innerhalb und zwischen diesen Ebenen sind möglich. Dann jedoch gilt diese Einsprungebene als gültig (d.h. es wird angenommen, dass die Bedingung wahr ist; siehe Programmierbeispiel).
- Sprünge innerhalb von $WHILE, $FOR, $DO, $REPEAT sind **nicht** erlaubt.
- Das vollständige Verlassen eines beliebigen Steuersatzkonstruktes mit $GOTO aus jeder Ebene heraus ist immer erlaubt.
- Label in Kommentaren (#COMMENT BEGIN, #COMMENT END) werden nicht erkannt.
- Bei String-Label werden Groß- und Kleinbuchstaben nicht unterschieden.
- Die bei der Dekodierung überlesenen Sprungmarken werden gespeichert. Die maximale Anzahl speicherbarer Expressionlabel [6] [▸ 894]-6.41 und Stringlabel [6] [▸ 894]-6.42 sowie die Stringlabellänge [6] [▸ 894]-6.43 sind vorgegeben.
- Bei jedem Sprungaufruf wird zuerst geprüft, ob die Sprungmarke bereits bekannt, also gespeichert ist. Bei positiver Prüfung wird der Sprung direkt ausgeführt. Wenn die Sprungmarke nicht bekannt ist, wird ab der **aktuellen** NC-Zeile in der Programmebene bis zum Programmende(M29/M30) gesucht. Wird die Sprungmarke nicht gefunden, so erfolgt die Ausgabe der Fehlermeldung P-ERR-20840.
- Wenn die maximale Anzahl speicherbarer Sprungmarken erreicht ist und die Dekodierung weitere neue Sprungmarken einliest, so werden diese Sprungmarken nicht mehr gespeichert. Dies wird durch die Warnings P-ERR-20829 bzw. P-ERR-20831 angezeigt. Bei jedem weiteren

Sprungaufruf mit unbekannter Sprungmarke beginnt dann die Suche erneut am **Anfang** der aktuellen Programmebene. In diesem Fall kann der Sprungvorgang bei sehr großen NC-Programmen mehr Zeit in Anspruch nehmen.

Programmierbeispiel

Die $GOTO-Anweisung

```
%goto
N05  P1=1
N06  P2=1
N10  G74 X1 Y2 Z3
N11       X0 Y0 Z0

N15  $IF P1==1 $GOTO N40: ;   Sprung von außen nach N40 in einen
                          ;   Steuerblock
N20  X10
N25  Y10
N30  $IF P1==2
N35    X20
N40:   $IF P2==1
N45    X30
N50:     Y30 $GOTO N65: ;   Sprung nach N65 zwischen Steuersatzebenen
                        ;   IF-ELSE
N51    $ENDIF
N55  $ELSE
N60    Y40
N65:   X40
N70  $ENDIF
N80   Z99
N999  M30
```

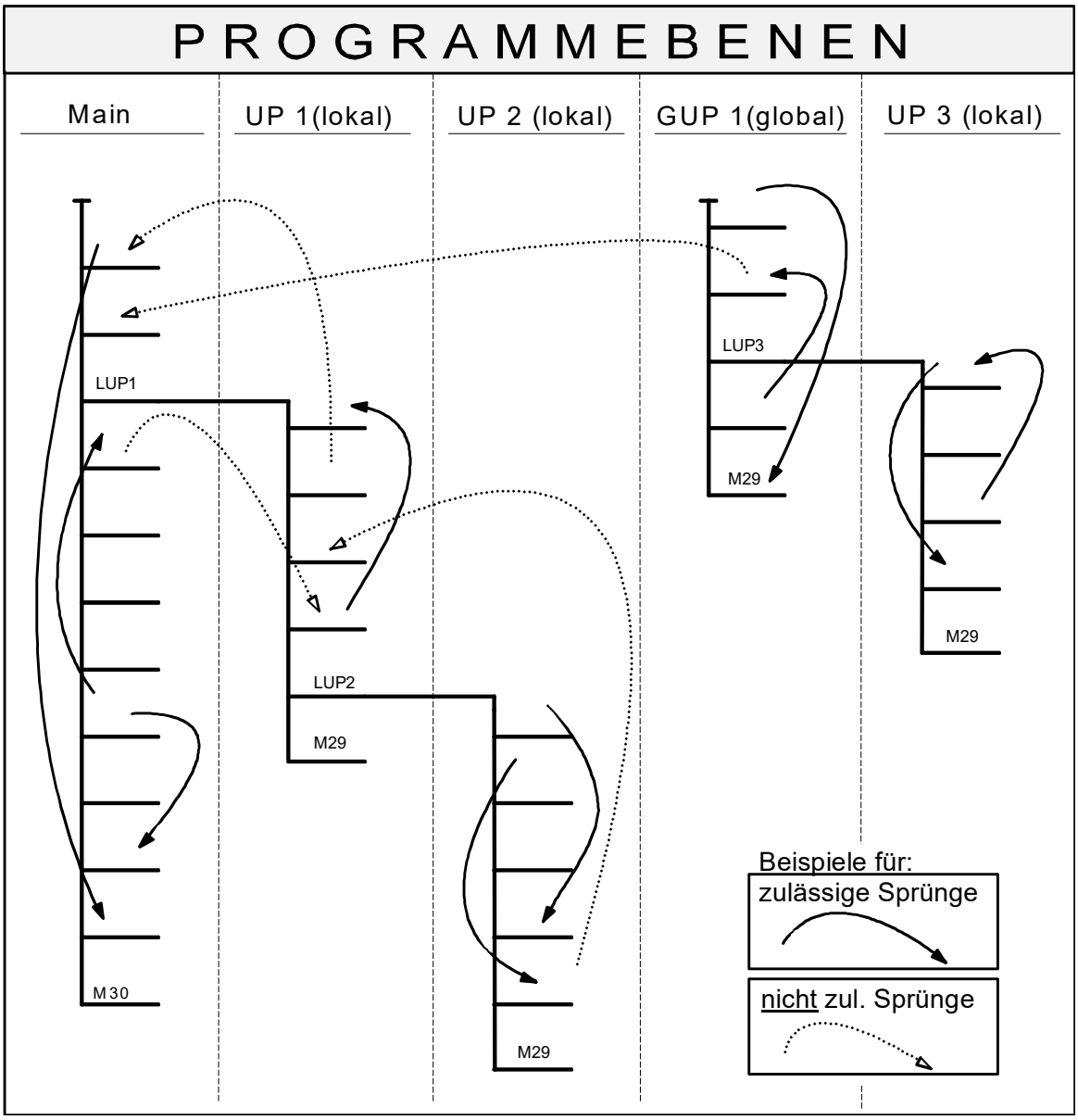


Abb. 69: Zulässige und nicht zulässige Sprünge beim $GOTO-Befehl

Programmierbeispiel

```
N10 G1 XY
N20: X100                                   ;Label Definition N20:
$IF V.L.dummy_1 <100 $GOTO N20              ;Sprung zu Label N20
$IF V.L.dummy_1 >200
$GOTO [LABEL_1]                             ;Sprung zu Label [LABEL_1]
Y20
$ENDIF
[LABEL_1] X0                                ;Label Definition [LABEL_1]
N30 A0
$FOR V.P.my-var = 0, 4, 1
$IF V.L.dummy_2 <200 $GOTO [CONTINUE]       ;Sprung zu Label [CONTINUE]
$SWITCH V.P.my-var
$CASE 0
V.P.AXE-X=V.P.GROUP[1].position[V.P.my-var]
$BREAK
$CASE 1
V.P.AXE-Y=V.P.GROUP[1].position[V.P.my-var]
$BREAK
$CASE 2
V.P.AXE-Z=V.P.GROUP[1].position[V.P.my-var]
$BREAK
$CASE 3
V.P.AXE-A=V.P.GROUP[1].position[V.P.my-var]
$DEFAULT
$ENDSWITCH
$ENDFOR
[CONTINUE]                              ;Label Definition [CONTINUE]
N1000 ...
...
```

10.1.3.1 Parametrierter Sprungaufruf

Bei der Verwendung des $GOTO-Befehles können die Sprungmarkenziele auch in parametrierter Form programmiert werden. Dies ermöglicht einen von außen beeinflussbaren Ablauf des NC-Programms (z.B. von der PLC).

Beim Sprungaufruf von Expressionlabeln sind zur Darstellung der Satznummer <*expr*> alle im Rahmen des Sprachumfanges verfügbaren mathematischen Ausdrücke zulässig, wie z.B. Parameter, lokale und globale Variablen sowie externe Variablen.

Syntax:

$GOTO N.. Sprungaufruf

Der Sprungaufruf von Stringlabeln kann über externe Variablen vom Typ String bzw. Stringarray parametriert werden (siehe auch Kapitel Externe Variablen (V.E.) [▶ 633]). Hierbei wird der Name der Sprungmarke in der externen Variablen abgelegt.

$GOTO V.E. … Sprungaufruf

Programmierbeispiel

Parametrierter Sprungaufruf

```
N10 ...
:
N50 $GOTO NV.E.JUMP_EXPR  ;Sprung z.B. zu N200 über ext. Variable
                          ;V.E.JUMP_EXPR, die den Wert 200 enthält
:
:
N100 $GOTO V.E.JUMP_STR   ;Sprung z.B. zu [CONTINUE) über ext. Variable
                          ;V.E.JUMP_STR, die den String CONTINUE enthält
:
:
N200:...
:
N500:…
:
:
:
[CONTINUE]…
:
N…
```

10.2 Zählschleifen ($FOR)

Zählschleifen erlauben das n-malige Abarbeiten von Anweisungen. Die Anzahl der Schleifendurchgänge wird durch eine Zählvariable kontrolliert.

Syntax:

Die Zählschleife beginnt mit:

$FOR P..= *<expr1>* **,** *<expr2>* **,** *<expr3>*

und endet immer mit:

$ENDFOR

Dabei ist P.. die Zählvariable, deren Startwert durch <expr1>, deren Endwert durch <expr2> und das Zählinkrement durch <expr3> festgelegt wird.

Hinweis

Als Zählvariable dürfen nur ganzzahlige Werte verwendet werden.

Bei der Verwendung von Dezimalzahlen kann die Schrittweite i.a. nicht exakt dargestellt werden (Ausnahme: Zweierpotenzen), da sich beim Aufsummieren ein Rundungsfehler akkumuliert. Dies kann dazu führen, dass eine Schleife einmal zu wenig durchlaufen wird.

Anstelle des P-Parameters können auch Variablen ("V.") mit Schreibzugriff verwendet werden.

Ist das Zählinkrement negativ, so erfolgt der Schleifenabbruch bei Unterschreitung des Endwertes, ist es positiv, so wird die Schleife bei Überschreitung des Endwertes abgebrochen. Die Programmierung des Zählinkrements 0 führt zu einer Endlosschleife und zur Ausgabe einer Warnung.

Programmierbeispiel

Zählschleifen

```
N100 $FOR P1= 10, 100, 2   P1 wird bei Schleifenbeginn mit 10 vorbelegt.
                           Die Zählschleife wird solange durchlaufen,
                           bis P1 den Wert 100 überschritten hat, wobei
                           P1 am Ende jedes Schleifendurchlaufs um 2
                           erhöht wird.
N110 X SIN [P1 * 5]        Innerhalb der Zählschleife kommen die NC-Sätze
                           bis N130 zur Ausführung.
N120 Y COS [P1 * 5]
N130 ...
N150 $ENDFOR
```

Programmierbeispiel

```
Negative Schrittweite:
N100 $FOR P1= 100, 10, -2  P1 wird bei Schleifenbeginn mit 100 vorbe-
legt.
                           Die Zählschleife wird solange durchlaufen,
bis
N110     X SIN [P1 * 5]    P1 den Wert 10 unterschritten hat, wobei P1
am
                           Ende jedes Schleifendurchlaufs um 2 reduziert
                           wird. Innerhalb der Zählschleife kommen N110
                           bis N130 zur Ausführung.
N120 Y COS [P1 * 5]
N130 ...
N150 $ENDFOR

Nicht ausgeführte Schleife:
N100 $FOR P1= 100, 10, 1   P1 wird bei Schleifenbeginn mit 100 vorbe-
legt.
                           Die Zählschleife soll solange durchlaufen wer
                           den, bis P1 den Wert 10 überschritten hat.
N110 X SIN [P1 * 5]        Hier jedoch nie, da P1 mit 100 vorbelegt ist.
N120 Y COS [P1 * 5]
N130 ...
N150 $ENDFOR

Endlosschleife:
N100 P2=20
N110 $FOR P1= 100, 10, 0 Endlosschleife
N120 $IF P2 == 50
N130 $BREAK
N140 $ENDIF
N150 $ENDFOR
```

10.3 Schleifen mit Laufbedingung

10.3.1 Prüfung der Laufbedingung am Schleifenanfang ($WHILE)

Syntax:

Die WHILE-Schleife beginnt mit:

$WHILE *<expr>*

und endet immer mit

$ENDWHILE

Zu Beginn jedes Schleifendurchlaufs wird der aufgeführte Parameter überprüft. Die Schleife wird abgebrochen, wenn der Ausdruck <expr> den Wertebereich FALSE (-0.5 < expr < 0.5) annimmt.

Programmierbeispiel

Prüfung der Laufbedingung am Schleifenanfang

```
N90 P1 = 100.0
N100 $WHILE P1 > 0.5       P1 > 0.5 wird bei Schleifenbeginn auf FALSE
N110 P1 = P1 - 1.5 YP1     überprüft. Die Schleife wird so oft durch-
N120 $ENDWHILE             laufen, bis P1 die Abbruchbedingung erfüllt.
N130 ...
```

10.3.2 Prüfung der Laufbedingung am Schleifenende ($DO), ($REPEAT)

Es stehen zwei Arten von Schleifen zur Verfügung.

Syntax:

Die DO-Schleife beginnt mit:

$DO

und endet immer mit

$ENDDO *<expr>*

Am Ende jedes Schleifendurchlaufs wird der aufgeführte Parameter überprüft. Die Schleife wird abgebrochen, wenn der Ausdruck <expr> den Wertebereich FALSE (expr < 0.5) annimmt.

Syntax:

Die REPEAT-Schleife beginnt mit:

$REPEAT

und endet immer mit

$UNTIL *<expr>*

Am Ende jedes Schleifendurchlaufs wird der aufgeführte Parameter überprüft. Die Schleife wird abgebrochen, wenn der Ausdruck <expr> den Wertebereich TRUE (expr > 0.5) annimmt.

Hinweis

Im Gegensatz zu $WHILE- und $FOR-Schleifen werden die $DO- und $REPEAT-Schleifen mindestens einmal durchlaufen.

Programmierbeispiel

Prüfung der Laufbedingung am Schleifenende

```
N10 X0 Y0 Z0
N20 P2=10 P1=0
N30 $DO
N40 P1=P1+1
N50 XP1
N60 $ENDDO P1 <= P2        P1 wird bei Schleifenende auf FALSE überprüft.
                           Die Schleife wird solange durchlaufen, bis P1
                           die Bedingung nicht mehr erfüllt.
N99 M30

N10 X0 Y0 Z0
N20 P2=10 P1=0
N30 $REPEAT
N40 P1=P1+1
N50 XP1
N60 $UNTIL P1 > P2         P1 wird bei Schleifenende auf TRUE überprüft.
                           Die Schleife wird solange durchlaufen, bis P1
                           die Bedingung erfüllt.
N99 M30
```

10.4 Beeinflussung von Schleifenabläufen

10.4.1 Die $BREAK-Anweisung

Syntax:

$BREAK

Nicht immer ist es sinnvoll, eine Schleife mit dem Abbruchkriterium zu verlassen. Das Schlüsselwort $BREAK kann neben der Programmausführung bei den einzelnen $CASE-Marken der $SWITCH-Anweisung (siehe Kapitel Der Sprungverteiler [▶ 239]) auch die Abarbeitung einer Schleife abrupt beenden.

Dies ist beispielsweise bei stark verschachtelten Schleifen sinnvoll, wenn die Abarbeitung der innersten Schleife unterbrochen werden soll.

Programmierbeispiel

Die $BREAK-Anweisung

```
N10 $WHILE <expr1>          Die Schleife wird beendet, wenn
N20 ...                     expr1 "nicht gültig" oder
N30                         expr2 "gültig" ist.
N40 $IF <expr2>
N50 $BREAK
N60 $ENDIF
N70 ...
N80
N90 $ENDWHILE
N100 ...
```

10.4.2 Die $CONTINUE-Anweisung

Syntax:

$CONTINUE

Im Gegensatz zu $BREAK wird mit der $CONTINUE-Anweisung die Schleife nicht abgebrochen, sondern zum Schleifenanfang verzweigt. Die Ausführung aller nach $CONTINUE stehenden Anweisungen findet also nicht statt.

Programmierbeispiel

Die $CONTINUE-Anweisung

```
N10 $FOR <expr1>          Die Anweisungen der Zeilen N70 und N80
N20 ...                   werden nur dann ausgeführt, wenn
N30                       expr2 "nicht gültig" ist.
N40 $IF <expr2>
N50 $CONTINUE
N60 $ENDIF
N70 ...
N80
N90 $ENDFOR
N100 ...
```

11 Glättungsverfahren

Einleitung

Um eine programmierte Kontur auch an Ecken ohne Stopp möglichst schnell und gleichmäßig durchfahren zu können, muss diese unter Einhaltung von Toleranzen gerundet bzw. geglättet werden. Für das s.g. Verschleifen stehen verschiedene Verfahren zur Verfügung.

Bei einfachen Konturen mit wenigen langen linearen und zirkularen Sätzen bietet sich Polynomüberschleifen an, das mit **G261** an- und mit **G260** abgewählt wird. Dieses Verfahren wird im Kapitel G-Funktionen [▶ 131] beschrieben.

Bei vielen kurzen Linearsätzen hingegen sind **#HSC**-Verfahren vorteilhaft. Hier gibt es das sehr robuste SURFACE-Verfahren [▶ 257], das sich insbesondere bei Freiformflächenbearbeitung eignet. Es liefert auch bei Störungen in der programmierten Kontur und sehr ungleichmäßig verteilten Satzlängen optimale Ergebnisse. Jedoch sind erhöhte Anforderungen an die Hardware zu berücksichtigen. Für das Besäumen einer Kontur lässt sich auch das B-Spline-Verfahren [▶ 254] verwenden. Es erfordert eine weniger leistungsfähige Hardware, kann bei ungünstig programmierten Konturen jedoch zu Bahngeschwindigkeitseinbrüchen führen.

Gibt es in diesen HSC-Programmen auch Kreissätze, können die Übergänge mit den **#CONTOUR MODE**-Funktionen verschliffen werden. Die hierfür notwendige Option CIR_MODE [▶ 257] bzw. der Kanalparameter P-CHAN-00239 werden in den oben genannten Kapiteln erläutert. Somit können HSC-Programme auch Kreissätze enthalten. In NC-Programmen mit vielen kurzen Sätzen bietet es sich an, den HSC-Profilgenerator #SLOPE[TYPE=HSC] [▶ 377] zu verwenden.

Neben der Glättung der programmierten Kontur bietet es sich häufig an, die Achssollwerte symmetrisch zu filtern. Die entsprechenden Funktionen werden im Kapitel Filterprogrammierung beschrieben.

Neben diesen empfohlenen Standard-Verfahren gibt es eine Reihe weiterer Methoden, wie die Interpolation mit dem Akima-Spline [▶ 297], die direkte Programmierung von B-Spline-Kontrollpunkten [▶ 303] und ältere HSC-Funktionen [▶ 306].

Name der Funktion	Wofür geeignet	Vorteile	Nachteile
#CONTOUR MODE	Für einfache Konturen mit wenigen langen Sätzen	Höhere Bahngeschwindigkeit an Konturknicken	Nicht für kurze Sätze
SURFACE-Verfahren	Für komplexe Konturen mit vielen kurzen Sätzen	Sehr robust	Erhöhte Anforderungen an die Hardware
B-Spline-Verfahren	Besäumen einer Kontur	Bei ungünstiger Programmierung, langsame Fahrstellen	Weniger hohe Anforderungen an Hardware
Filterprogrammierung	Um Achssollwerte symmetrisch zu filtern		
Akima-Spline	Interpolation von vorgegebenen Stützpunkten	Läuft exakt durch die programmierten Punkte	Erfordert im allgemeinen eine dichte und exakt berechnete Punktevorgabe
HSC-Funktionen mit OP1 und OP2	Steife Maschinen	Geringe Hardwareanforderungen	Relativ starke Anregung der Maschinenstruktur

11.1 Programme mit vielen kurzen Sätzen

Hinweis

Die Nutzung dieser Funktionalität erfordert die Lizenzierung des Erweiterungspaketes "HSC". Sie ist nicht im Umfang der Standardlizenz enthalten.

Achtung

Bei der erweiterten HSC-Programmierung werden B-Splines für die programmierten Kontrollpunkte erzeugt. Es wird empfohlen, für diese Art der HSC-Bearbeitung zuerst über den Befehl #SLOPE [TYPE...] [▶ 377] den Profiltyp HSC (Slope 3) anzuwählen.

Abhängig von der Bearbeitungsaufgabe stehen für die HSC-An/Abwahl sowie für die Parametrierung 2 Verfahren zur Verfügung:

Verfahren 1 eignet sich insbesondere für das einmalige Umfahren einer Kontur (Besäumen). Hierbei besteht die Kontur aus sehr vielen kurzen Sätzen, die mit hohem Vorschub gefahren werden sollen.

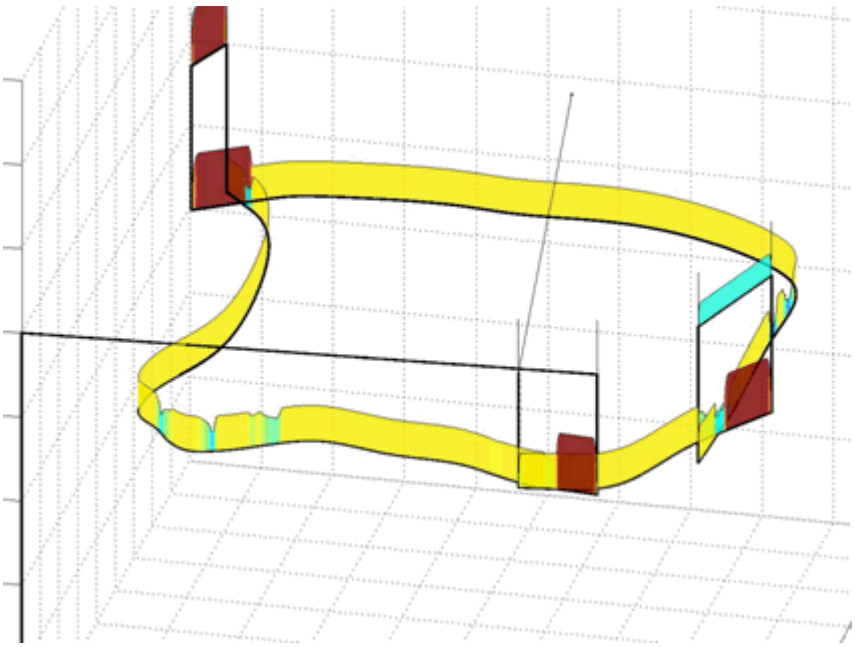


Abb. 70: Besäumen einer Kontur

Verfahren 2 eignet sich insbesondere für die Bearbeitung von Freiformflächen. Bei diesen aus dem CAD-System generierten NC-Programmen wird das Werkstück in der Regel in vielen Bahnen (zeilen- oder spiralförmig) abgearbeitet. Zur Erreichung einer hohen Oberflächengüte bei möglichst kurzer Bearbeitungszeit kommen hierbei spezielle Algorithmen zum Einsatz (Surface Optimizer).

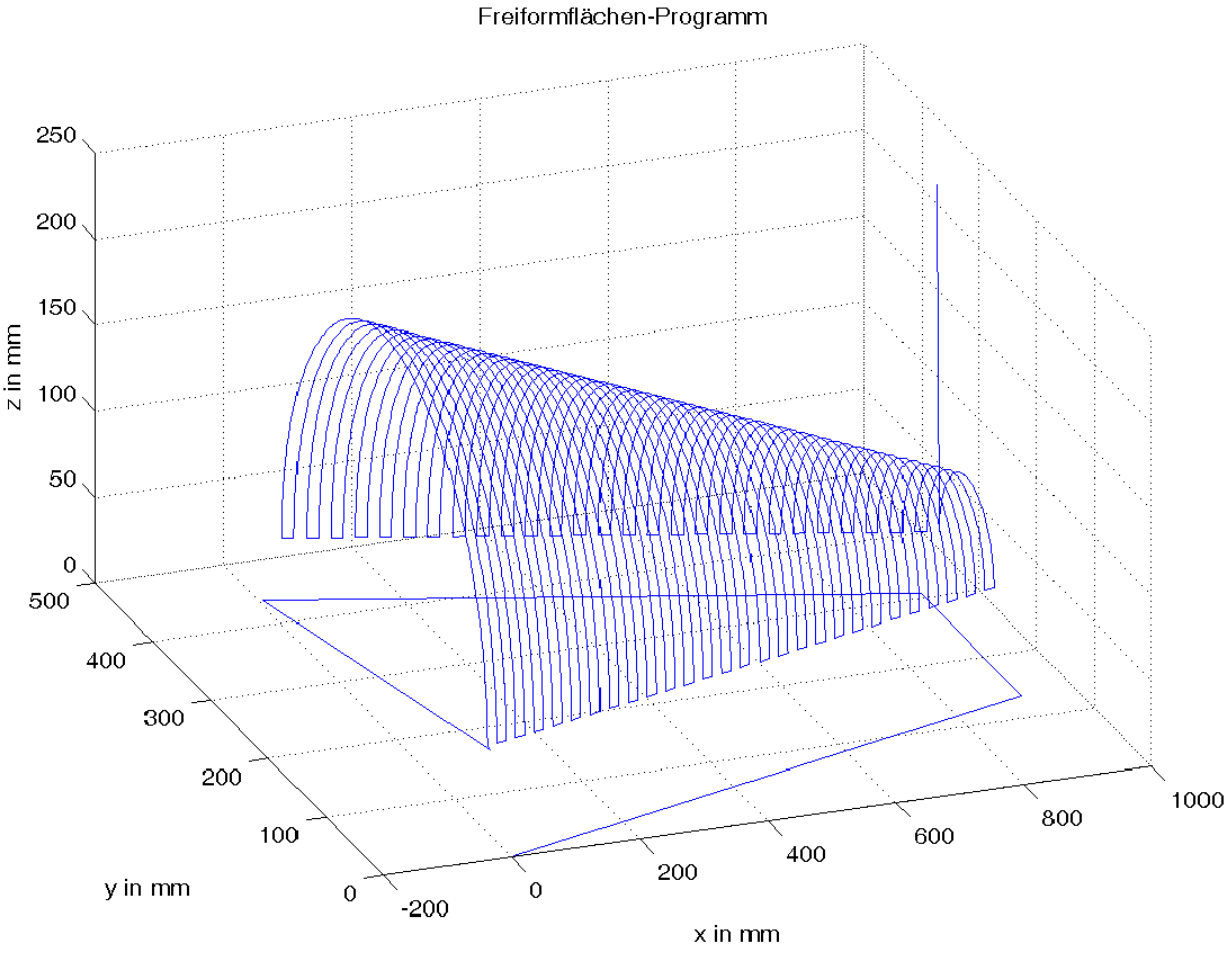


Abb. 71: Zeilenförmiges Bearbeiten einer Oberfläche

11.1.1 Besäumen einer Kontur (#HSC ON/OFF)

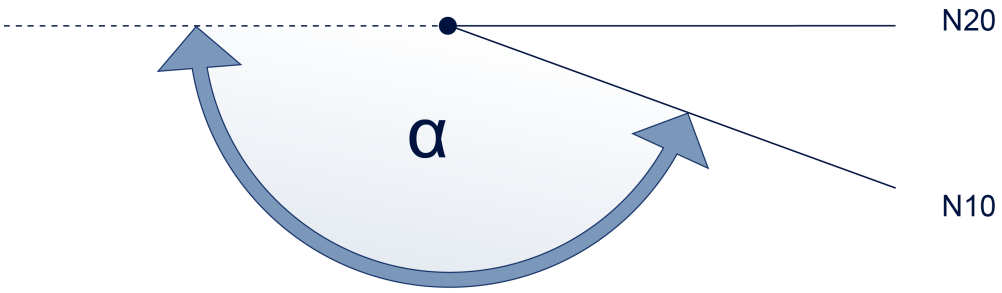
Syntax:

#HSC [**ON** | **OFF**] [**BSPLINE** [**PATH_DEV=..**] [**TRACK_DEV=..**] [**MERGE=..**] [**AUTO_OFF_PATH=..**] [**AUTO_OFF_TRACK=..**] [**AUTO_OFF_G00=..**] [**AUTO_OFF_G60=..**] [**MAX_PATH_LENGTH=..**] [**MAX_ANGLE=..**] **]**]

ON	HSC-Bearbeitung aktivieren.
OFF	HSC-Bearbeitung deaktivieren.
BSPLINE	Kennwort für die HSC-Bearbeitung mit BSPLINE. Muss immer als **erstes** Schlüsselwort programmiert sein!
PATH_DEV=..	Maximale Abweichung des B-Splines von der programmierten Bahnkontur in [mm, inch *]. Wird diese Abweichung überschritten, so wird der Spline automatisch abgewählt. Wird als maximale Abweichung 0 angegeben, so findet keine Überwachung der Bahnabweichung statt. Standardwert: 0.2 mm *bei aktivem P-CHAN-00439
TRACK_DEV=..	Maximale Abweichung der Mitschleppachsen in [°]. Wird als maximale Abweichung 0 angegeben, so findet keine Überwachung der Mitschleppachsen statt. Standardwert: 5°
MERGE=..	Zusammenfassen der Sätze. Die max. Abweichung wird entsprechend der Werte aus PATH_DEV und TRACK_DEV bestimmt. 0: Kein Zusammenfassen von Sätzen (Standard) 1: Zusammenfassen von Sätzen
AUTO_OFF_PATH=..	Automatische Satzunterteilung bei Überschreiten der programmierten B-Splineabweichung der Hauptachsen (PATH_DEV). 0: Keine Abwahl bei zu großer Abweichung (Standard), Satz wird unterteilt 1: Abwahl bei zu großer Abweichung
AUTO_OFF_TRACK=..	Automatische Satzunterteilung bei Überschreiten der programmierten B-Splineabweichung der Mitschleppachsen (TRACK_DEV). 0: Keine Abwahl bei zu großer Abweichung (Standard), Satz wird unterteilt 1: Abwahl bei zu großer Abweichung
AUTO_OFF_G00=..	Automatische Abwahl der B-Splineinterpolation bei G00-Sätzen. 0: Keine implizite Abwahl aufgrund eines Eilgangsatzes (Standard) 1: Implizite Abwahl aufgrund eines Eilgangsatzes
AUTO_OFF_G60=..	Automatische Abwahl der B-Splineinterpolation bei programmiertem Genauhalt G60 oder G360. 0: Keine implizite Abwahl aufgrund von Genauhalt (Standard) 1: Implizite Abwahl aufgrund von Genauhalt
MAX_PATH_LENGTH=..	Maximale Bahnlänge relevanter Sätze in [mm, inch *]. Sind Sätze länger als die angegebene Länge, so wird der B-Spline implizit abgewählt. Standardwert: 0 mm (implizite Abwahl aufgrund der Satzlänge findet nicht statt) *bei aktivem P-CHAN-00439

MAX_ANGLE=.. Maximaler Konturknickwinkel in [°] für Übergänge zwischen zwei Linearsätzen bis zu dem ein B-Spline eingefügt wird. Ist der Winkel zwischen den beiden Linearsätzen größer, so erfolgt eine interne Abwahl des B-Splines.

Standardwert: 160°



Die Programmierung der Kontrollpunkte erfolgt mit Linearsätzen (G00 und G01), deren Zielpunkte als Kontrollpunkte dienen. Es ist zu beachten, dass die Kurve nur am Anfang und am Ende sicher durch die Kontrollpunkte hindurch verläuft.

Hinweis

Die Parameter können auch in mehreren Schritten angegeben werden. Das heißt, es ist z.B. möglich, zunächst die maximale Abweichung von der Kontur ("PATH_DEV ") anzugeben und in einem zweiten Befehl dann die maximale Bahnlänge ("MAX_PATH_LENGTH ") und die Anwahl der B-Splineinterpolation ("ON") festzulegen.

Achtung

Die Parametrierung kann während aktiver B-Splineinterpolation nicht geändert werden.

Programmierbeispiel

Besäumen einer Kontur

Die Splinekurve basiert auf den Kontrollpunkten N40 - N155, wobei die Splinekurve nur bei N20 und N150 direkt hindurch verläuft.

```
N20 G00 X0 Y0 Z0 F10000
N30 #HSC ON [BSPLINE PATH_DEV=0.2 MERGE=1 ...] Parametrierung + Anwahl
N40   X3     Y25
N50   X15    Y15
N60   X23    Y12
N70   X25    Y25
N80   X30    Y35
N90   X50    Y37.5
N100  X55    Y32.5
N110  X58    Y12
N120  X70    Y12
N130  X77.5  Y10
N140  X90    Y35
N150  X100   Y37.5
N160 #HSC OFF
N170 M30
```

... oder auch

```
N20  G00 X0 Y0 Z0 F10000
N25  #HSC [BSPLINE PATH_DEV=0.2 MERGE=1 ...] Parametrierung
N30  #HSC ON Anwahl
N40   X3     Y25
N50   X15    Y15
N60   X23    Y12
N70   X25    Y25
N80   X30    Y35
N90   X50    Y37.5
N100  X55    Y32.5
N110  X58    Y12
N120  X70    Y12
N130  X77.5  Y10
N140  X90    Y35
N150  X100   Y37.5
N160 #HSC OFF Abwahl
N170 M30
```

11.1.2 Oberflächenbearbeitung mit Surface Optimizer (Verfahren 3)

Der HSC Surface Optimizer wurde entwickelt um gleichmäßige Bearbeitungsergebnisse unabhängig von der Punkteverteilung durch das CAM-System zu erreichen. Insbesondere die Dichte von Stützpunkten auf benachbarten Bearbeitungsbahnen kann bei einigen CAM-Systemen schwanken, was zu einem unregelmäßigen Bearbeitungsergebnis führen würde. In der nachfolgenden Abbildung ist ein solches Bearbeitungsergebnis dargestellt. Die beiden rot markierten Punkte fehlen auf einer der benachbarten Bearbeitungsbahnen. Dadurch entsteht bei einem nicht optimierten Glättungsverfahren eine andere Werkzeugbahn (blau) im Gegensatz zu den benachbarten Bahnen.

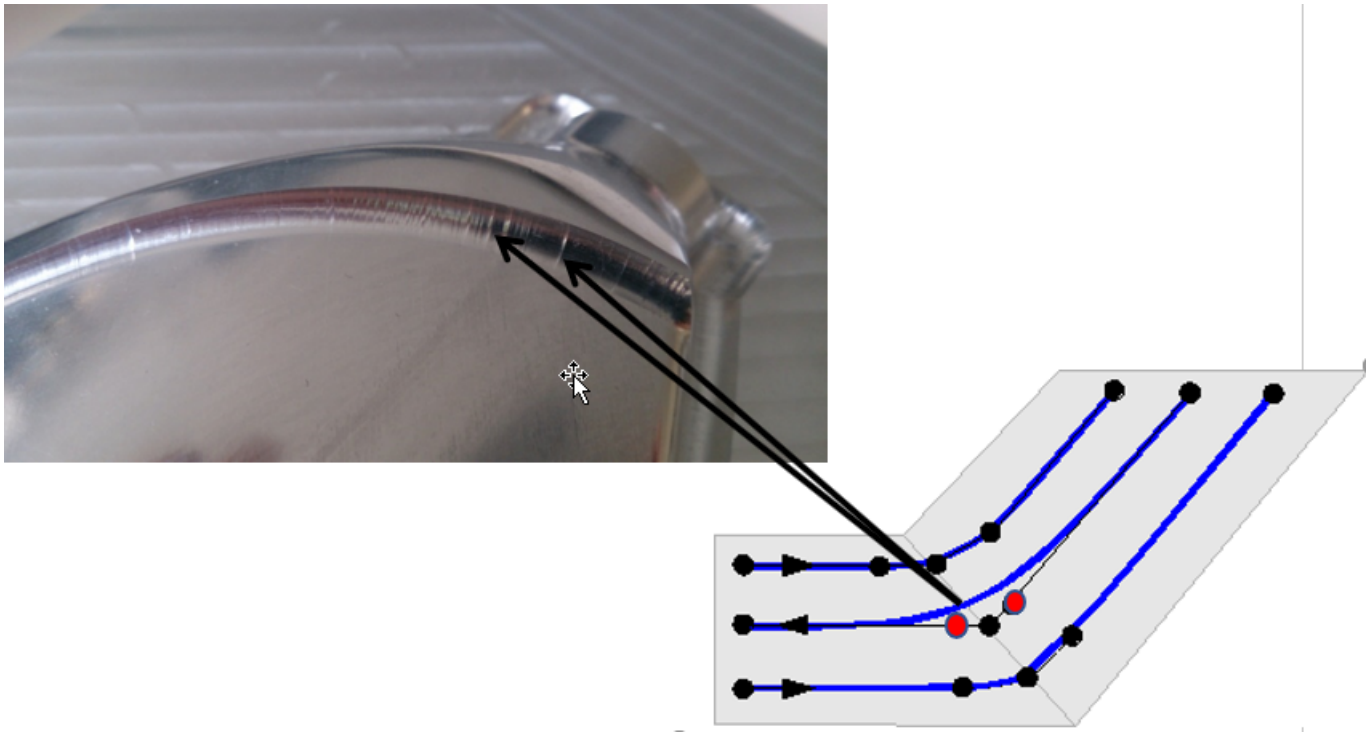


Abb. 72: Probleme in der Werkstückqualität aufgrund unregelmäßiger Punkteverteilung durch CAM-System.

Der HSC Surface Optimizer sorgt zusätzlich zu gleichmäßigen Bearbeitungsbahnen für eine hohen und möglichst konstanten Vorschub. Aufgrund der notwendigen Berechnungen erfordert die Verwendung des Surface Optimizers eine möglichst performante Steuerungshardware.

Programmierung

Syntax:

#HSC [**ON** | **OFF**] [[**SURFACE** [**PATH_DEV=..**] [**PATH_DEV_G00=..**] [**TRACK_DEV=..**] [**TRACK_DEV_G00=..**]
[**MAX_ANGLE=..**] [**CHECK_JERK=..**] [**AUTO_OFF_G00=..**] [**CIR_MODE=..**]
[**CIR_MIN_ANGLE=..**] [**CIR_MIN_RADIUS**=..] [**MERGE=..**] [**LENGTH_LONG_CIR=..**] **]**]

ON	HSC-Bearbeitung aktivieren.
OFF	HSC-Bearbeitung deaktivieren.
SURFACE	Kennwort für die HSC-Bearbeitung mit Surface Optimizer. Muss immer als **erstes** Schlüsselwort programmiert sein!
PATH_DEV=..	Festlegung des maximalen Konturfehlers. > 0.0: Maximale Bahnabweichung in [mm, inch *] Standardwert: 0.2 mm

Hinweis

Erfahrungsgemäß sollte der Fehler auf das 2 bis 3-fache des bei der Erzeugung des NC-Programms im CAM-System festgelegten Sehnenfehlers eingestellt werden.

Während G0-Bewegungen befindet sich das Werkzeug nicht im Eingriff. Deshalb kann die Toleranz deutlich größer als PATH_DEV gewählt werden, ohne die Werkstückgenauigkeit zu verändern.

PATH_DEV_G00=..	Festlegung des maximalen Konturfehlers bei G0-G0-Übergängen. > 0.0: Maximale Bahnabweichung in [mm, inch *] Standardwert: Es gilt der Wert von PATH_DEV
TRACK_DEV=..	Festlegung des maximalen Orientierungsfehlers. >= 0.0: Maximale Bahnabweichung in [°] Standardwert: 2°
TRACK_DEV_G00=..	Festlegung des maximalen Orientierungsfehlers bei G0-G0-Übergängen. >= 0.0: Maximale Bahnabweichung in [°] Standardwert: Es gilt der Wert von TRACK_DEV

Hinweis

Wird ein Kugelfräser verwendet, kann der eingestellte Wert deutlich größer als PATH_DEV eingestellt werden (z.B. 10-fach).

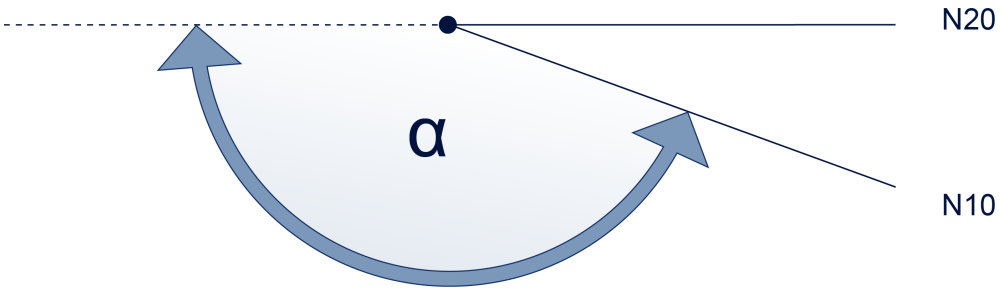
Während G0-Bewegungen befindet sich das Werkzeug nicht im Eingriff. Deshalb kann die Toleranz deutlich größer als TRACK_DEV gewählt werden, ohne die Werkstückgenauigkeit zu verändern.

MAX_ANGLE=..

Festlegung des maximalen Konturknickwinkels in Grad für Übergänge zwischen zwei Linearsätzen bis zu dem das Verfahren angewendet wird. Ist der Winkel zwischen den beiden Linearsätzen größer, so erfolgt eine interne Abwahl des Verfahrens.

>= 0.0: Maximaler Knickwinkel in [°]

Standardwert: 160°



CHECK_JERK=..

Überwachung des Rucks, hervorgerufen durch die Krümmung des Polynoms (vgl. P-CHAN-00110). Dieser Parameter überschreibt die in der Kanalparameterliste durch P-CHAN-00110 (check_jerk_on_poly_path) vorgenommene Grundeinstellung.

0: Ohne Rucküberwachung

1: Rucküberwachung basierend auf der geom. Rampenzeit P-AXIS-00199. Evtl. wird hierdurch die Bahngeschwindigkeit reduziert.

2: Rucküberwachung basierend auf den Rampenzeiten P-AXIS-00195, P-AXIS-00198 des nichtlinearen Geschwindigkeitsprofils.

AUTO_OFF_G00=..

Automatische Abwahl der Optimierung bei G00-Sätzen

0: Keine implizite Abwahl aufgrund eines Eilgangsatzes (Standard)

1: Implizite Abwahl aufgrund eines Eilgangsatzes

CIR_MODE=..

Festlegung Überschleifen von Kreisbewegungen:

0 : Kein Überschleifen von Kreissätzen G02/G03.

1 : Überschleifen von Kreissätzen. (Standard)

2 : Überschleifen von Kreissätzen und Optimierung von langen Kreissätzen
Verfügbar ab V3.1.3075.01

CIR_MIN_ANGLE=..

Festlegung des minimalen Kreiswinkels

Gültige Werte: >= 0.0 : Minimaler Kreiswinkel in Grad

Der minimale Kreiswinkel bestimmt, ab welchem Kreiswinkel Zirkularbewegungen durch das Verfahren mittels exakter Interpolation abgefahren werden. Kreissätze mit kleinerem überstrichenem Winkel werden durch eine Splinekurve für die schnellere Bearbeitung approximiert. (Standardwert = 30°)

CIR_MIN_RADIUS=..

Festlegung des minimalen Kreisradius

Gültige Werte: >= 0.0 : Minimaler Kreisradius in [mm, inch *]

Der minimale Kreisradius bestimmt, ab welchem Kreisradius Zirkularbewegungen durch das Verfahren mittels exakter Interpolation abgefahren werden. Kreissätze mit kleinerem Radius oder in der Größenordnung von PATH_DEV werden durch eine Splinekurve approximiert.

Verfügbar ab V3.1.3075.01

MERGE=..	Zusammenfassen der Sätze. Die max. Abweichung wird entsprechend der Werte aus PATH_DEV und TRACK_DEV bestimmt. 0: Kein Zusammenfassen von Sätzen (Standard) 1: Zusammenfassen von Sätzen
LENGTH_LONG_CIR=..	Mindestlänge der Kreissegmente für lange Kreissätze bei Verwendung von CIR_MODE= 2 in [mm, inch *] (Standardwert = 2) Verfügbar ab V3.1.3075.01 *bei aktivem P-CHAN-00439

Standardwerte der Freiformflächenbearbeitung

PATH_DEV	0.2mm (Standardwert von PATH_DEV)
TRACK_DEV	2° (Standardwert von TRACK_DEV)
PATH_DEV_G00	PATH_DEV
TRACK_DEV_G00	TRACK_DEV
CIR_MODE	1
MAX_ANGLE	160°
CHECK_JERK	Es gilt der Kanalparameter P-CHAN-00110 (check_jerk_on_poly_path, Standardwert = 1)
AUTO_OFF_G00	0
CIR_MIN_ANGLE	30°
CIR_MIN_RADIUS	0.0
LENGTH_LONG_CIR	2mm

Hinweis

Die Parameter können auch in mehreren Schritten angegeben werden. Z.B. ist es möglich, zunächst die maximale Abweichung von der Kontur ("PATH_DEV ") anzugeben und in einem 2ten Befehl dann die Rucküberwachung ("CHECK_JERK") und die Anwahl der HSC-Surfaceinterpolation ("ON") festzulegen.

Hinweis

Bei Verwendung von #HSC[SURFACE] wird empfohlen gleichzeitig für die Bahngeschwindigkeitsplanung den #SLOPE[TYPE=HSC] zu nutzen.

Achtung

Die Parametrierung kann während aktiver Glättung nicht geändert werden.

Voraussetzung für die Nutzung dieser Funktionalität ist die Parametrierung in der Hochlaufliste für jeden Kanal, in dem die Funktion verwendet werden soll.

Programmierbeispiel

Oberflächenbearbeitung mit Surface Optimizer

Beispiel für die Einstellung in der Hochlaufliste:

configuration.channel[].path_preparation.function FCT_DEFAULT|FCT_SURFACE

```
N20 G00 X0 Y0 Z0 F10000
;Parametrierung + Anwahl
N30 #HSC ON [SURFACE PATH_DEV=0.02 CHECK_JERK=0]
N40   X3     Y25
N50   15     Y15
N60   23     Y12
N70   X25    Y25
N80   X30    Y35
N90   X50    Y37.5
N100  X55    Y32.5
N110  X58    Y12
N120  X70    Y12
N130  X77.5  Y10
N140  X90    Y35
N150  X100   Y37.5
N160 #HSC OFF
N170 M30
```

Alternative Programmierung:

```
N20 G00  X0    Y0    Z0 F10000
N25 #HSC [SURFACE PATH_DEV=0.02 CHECK_JERK=0] ;Parametrierung
N30  #HSC ON ;Anwahl
N40   X3     Y25
N50   15     Y15
N60   23     Y12
N70   X25    Y25
N80   X30    Y35
N90   X50    Y37.5
N100  X55    Y32.5
N110  X58    Y12
N120  X70    Y12
N130  X77.5  Y10
N140  X90    Y35
N150  X100   Y37.5
N160 #HSC OFF
N170 M30
```

11.1.3 FIR-Filter (#FILTER)

Versionshinweis

Die Verfügbarkeit dieser Funktionalität ist von der Konfiguration und dem Versionsumfang abhängig.

Um bei der Freiformflächenbearbeitung eine gute Oberflächengüte zu erreichen, muss die Anregung von Maschinenschwingungen so weit wie möglich vermieden werden.

FIR-Achsfilter (Finite-Impulse-Response-Filter) bieten dem Anwender die Möglichkeit, die Achssollwerte für die Antriebe zu glätten und somit Anregungen der Maschine zu minimieren.

Voraussetzung für die Nutzung eines FIR-Filters über den #FILTER Befehl ist ein konfigurierter Filtertyp (P-AXIS-00586) **der entsprechenden Achsen.**

Hinweis

Diese Funktionalität ist Bestandteil einer lizenzpflichtigen Zusatzoption.

Syntax:

#FILTER [**ON** | **OFF**] [**ORDER**=.. **ORDER_TIME**=.. **SHARE**=.. **AX_DEV**=.. **FCUT**=.. **ACC_FACT**=.. **QUALITY**=.. **]**

ON	FIR-Filter aktivieren.
OFF	FIR-Filter deaktivieren.
ORDER=..	Angabe der Filterordnung.
ORDER_TIME=..	Angabe der Filterordnung über der Zeit in [µs]
SHARE=..	Festlegen des Wirkungsgrads (analog zu P-AXIS-00590) des Filters in [%] Wertebereich 0 – 100 Standardwert = 100
AX_DEV=..	Angabe der Toleranz für Toleranzüberwachung in [mm, inch *]. Standardwert = 0 (keine Toleranzüberwachung). *bei aktivem P-CHAN-00439
FCUT=..	Angabe der Grenzfrequenz (analog zu(P-AXIS-00585) des Filters in [Hz] Standardwert = 30
ACC_FACT=..	Erhöhen der Bahngeschwindigkeit an Satzübergängen bei aktivem FIR-Filter. Je größer der Wert eingestellt wird, desto weniger wird die Geschwindigkeit am Satzübergang reduziert. Voraussetzung ist eine gültige Einstellung von P-AXIS-00013 (a_trans_weight) der Achsen Wertebereich = 1.0 – 10.0 Standardwert =: 1.0
QUALITY=..	Filtergüte- Angabe zur Breite der Filterkern-Kurve Wertebereich: 0 < QUALITY <= 1 Standardwert = 1.0 Parameter verfügbar ab V3.1.3075.04

Hinweis

Mit dem Befehl #FILTER ON/OFF werden alle FIR-Filter der im Kanal vorhandenen Achsen aktiviert bzw. deaktiviert.

Es ist möglich, FIR-Filter auf allen Achsen zu nutzen. Durch die achsspezifische Konfiguration über die Achslisten ist es außerdem möglich, unterschiedliche Filter je Achse zu verwenden.

Die FIR-Filter können über das NC-Programm während der Bearbeitung global über alle Achsen an- und ausgeschaltet sowie umparametriert werden (siehe Programmierbeispiel).

Hinweis

Die Toleranzüberwachung kann nur im NC-Programm konfiguriert und aktiviert werden.

Mit dem Parameter AX_DEV wird die Toleranzüberwachung programmiert. Sie stellt sicher, dass jede Achse innerhalb der vorgegebenen Toleranz [mm, inch] bleibt.

Die Toleranzüberwachung überwacht immer alle Achsen und kann daher nur global über das NC-Programm gesteuert werden.

Die Toleranzüberwachung ist nur aktiv wenn AX_DEV mit einer entsprechenden Toleranz vorgegeben ist.

Weitere Informationen unter [FCT-C37//Beschreibung]

Dieser Befehl ersetzt den bisher verfügbaren #FILTER ON [HSC] Befehl.

11.2 Polynomüberschleifen für lange Sätze (G61/G261/G260)

Syntax:

G61	Polynomüberschleifen (am Satzende)	nicht modal

... oder bei Polynomüberschleifen über mehrere Sätze:

G261	Anwahl Polynomüberschleifen (am Satzende)	modal
G260	Abwahl Polynomüberschleifen	modal

11.2.1 Begriffsdefinitionen

Folgende Begriffe sollen kurz erläutert werden:

Polynomüberschleifen:	Krümmungs- und richtungsstetige Verbindung zweier Bewegungssätze.
Überschleifkurve:	Kurve aus zwei Polynomen 4. Ordnung je Achse.
Satzlänge:	Die Bahnlänge der dem Bewegungssatz entsprechenden Kurve.
Eckenabstand:	Abstand des Beginns/des Endes der Überschleifkurve vom programmierten Zielpunkt/ Startpunkt eines Bewegungssatzes (siehe folgende Abbildung). Der Eckenabstand wird stets auf die halbe Satzlänge begrenzt. Bei einem Zirkularsatz versteht man unter dem Eckenabstand die Bogenlänge vom Startpunkt der Überschleifkurve bis zum programmierten Endpunkt des Kreisbogens.

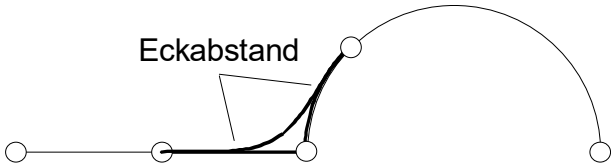


Abb. 73: Definition des Eckenabstands

Vorsatz:	Der Bewegungssatz vor der Überschleifkurve
Nachsatz:	Der Bewegungssatz nach der Überschleifkurve
Vorabstand:	Der Eckenabstand des Vorsatzes
Nachabstand:	Der Eckenabstand des Nachsatzes
Zwischenpunkt:	Punkt, an dem die beiden Teilkurven der Überschleifkurve aneinandergrenzen.
Eckenabweichung:	Der Abstand zwischen dem programmierten Eckpunkt und dem Zwischenpunkt der Überschleifkurve (siehe folgende Abbildung).

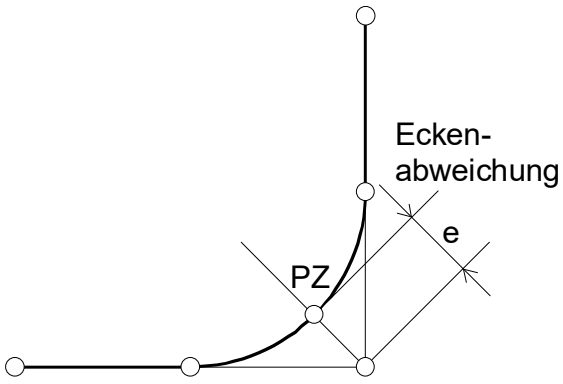


Abb. 74: Definition der Eckenabweichung

Programmierbeispiel

Vergleich der Programmierung von G61 – G261/G260

Die drei NC-Programme erzeugen alle die gleiche im Bild dargestellte Kontur.

```
%poly_G61
N10  X0   Y0 G01 F1000
N20  X20  Y100
N30  G61  X40 Y100
N40  G61  X60 Y20
N50  G61  X80 Y20
N60  G61  X100 Y100
N70  X120 Y100
N80  X140 Y20
N90  X160 Y20
N100 M30
```

```
%poly_G261_1
N10  X0   Y0 G01 F1000
N20  X20  Y100
N30  G261 X40 Y100
N40  X60  Y20
N50  X80  Y20
N60  X100 Y100
N70  G260 X120 Y100
N80  X140 Y20
N90  X160 Y20
N100 M30
```

```
%poly_G261_2
N10 X0   Y0 G01 F1000
N20 X20  Y100
N25 G261
N30 X40  Y100
N40 X60  Y20
N50 X80  Y20
N60 X100 Y100
N70 X120 Y100
N75 G260
N80 X140 Y20
```

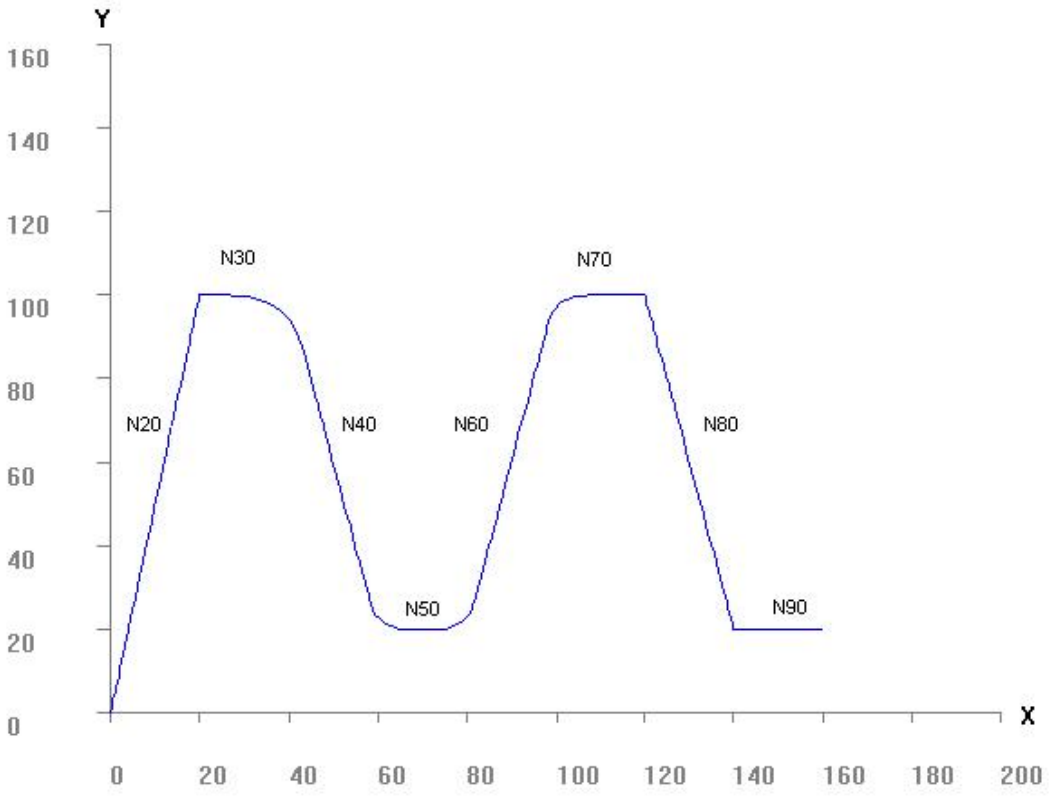


Abb. 75: Kontur der Programmierung von G61 – G261/G260

11.2.2 Allgemeine Eigenschaften

Das Verfahren des Polynomüberschleifens berechnet sich nach der geometrischen Bahnkontur der Hauptachsen im Raum. Mit den vorgegebenen Randbedingungen, z.B. Eckenabweichung oder prozentuale Bahngeschwindigkeit, ergibt sich eine Position, auf der ursprünglichen Kontur, ab welcher diese verändert bzw. durch eine Überschleifkurve (Polynom) ersetzt werden kann. D.h. der Start- bzw. Endpunkt der Überschleifkurve auf der ursprünglichen Bahnkurve ist somit bekannt.

Aus dem ermittelten Start- und Zielpunkt des gemäß den Randbedingungen ermittelten Polynoms der Hauptachsen lässt sich ebenso die Position der Mitschleppachsen angeben, ab welcher ihre ursprüngliche Kontur durch ein Polynom ersetzt werden kann.

Bei den Mitschleppachsen wird, ebenso wie bei den Hauptachsen, zwischen den Eckenabständen des Vor- und Nachsatzes ein krümmungs- und richtungsstetiges Polynom unter Berücksichtigung der max. Beschleunigung dieser Achse eingefügt. Die evtl. ursprünglich angegebene Eckenabweichung bezieht sich jedoch nur auf die Abweichung der Hauptachse im Raum, sodass falls gewünscht für die maximale Abweichung der Mitschleppachsen ein zusätzlicher Grenzwert angegeben werden kann. Ein theoretisches Überschreiten dieser Abweichung durch die Mitschleppachse führt zur Reduktion der Überschleifkurve (Verkleinern des Eckenabstandes).

Das Polynomüberschleifen wird in Abhängigkeit des Übergangs zwischen dem Vor- und Nachsatz automatisch unterdrückt falls:

- Der Übergang aller Achsen tangential bzw. direkt gespiegelt ist.
- Der Übergang der Hauptachsen tangential ist und keine maximale Abweichung (Wert = 0) für die Mitschleppachsen angegeben wurde.
- Nach G61-Programmierung das Programmende ohne Folgesatz erreicht wird. Hier wird zusätzlich eine Warnung ausgegeben.

11.2.2.1 Maximaler Eckenabstand, verbleibende Mindestsatzlänge

Um eine „Entartung“ der Polynome zu verhindern gelten zusätzlich folgende Einschränkungen:

- Der Eckenabstand kann maximal 50% der ursprünglichen Satzlänge annehmen. Wurde dieser größer gewählt, so wird der Abstand des Vor- bzw. Nachsatzes entsprechend begrenzt. Beträgt der Eckenabstand am Satzanfang und am Satzende jeweils 50% der ursprünglichen Satzlänge, so entfällt der Satz vollständig.
- Bei der Parametrierung des Überschleifens kann die minimal verbleibende Satzlänge zwischen 0% und 100% eingestellt werden. Dies entspricht einem variablen maximalen Eckenabstand von 50% bis 0%. Bei jedem Programmstart ist die minimal verbleibende Satzlänge zunächst auf 0% gesetzt (Satz kann vollständig überschliffen werden). Wird die minimal verbleibende Satzlänge z.B. mit 10% vorgegeben, so können die Eckenabstände dieses Satzes maximal (100% - 10%) / 2 = 45% der ursprünglichen Satzlänge annehmen.
- Des Weiteren wird bei Zirkularsätzen der maximale Eckenabstand (zurückgelegte Strecke auf Kreis) so beschränkt, dass der dadurch überstrichene Winkel 90° nicht überschreitet.

11.2.2.2 Relevante Satzlänge

Ist die programmierte relevante Satzlänge, die über RELEVANT_PATH festgelegt wird, kleiner als die definierte Mindestlänge von 32µm, so wird der Satz auf diese Mindestlänge begrenzt.

Darüber hinaus kann die Bahnkontur sehr kleine Ausgleichssätze enthalten, welche durch ein Programmiersystem (CAD/CAM) oder eine Werkzeugradiuskorrektur eingefügt wurden, damit die Bahn auch nach der Korrektur einen stetigen Verlauf beibehält.

Um einen Abbruch des Überschleifens durch diese kurzen Sätze zu verhindern, kann eine minimale Satzlänge angegeben werden, ab welcher der Nachsatz für das Polynomüberschleifen relevant ist. Sätze, welche kürzer sind, werden bei angewähltem Überschleifen ausgelassen, d.h. es wird in den darauffolgenden Satz überschliffen.

Hierbei kann sowohl eine Grenze für den Raumfahrweg der Hauptachsen wie auch eine Grenze für den Fahrweg der Mitschleppachsen angegeben werden. Erst wenn sowohl der Raumfahrweg der Hauptachse wie auch der Fahrweg jeder einzelnen Mitschleppachse unterhalb der angegebenen Grenze liegt, wird der Satz ausgelassen. Das Polynomüberschleifen verbindet den vorgehenden und nachfolgenden Satz richtungs- und krümmungsstetig, wobei die Ausgangssätze nicht angrenzend sein müssen (Kontur muss nicht stetig sein).

Wird ein Satz ausgelassen, so werden die maximale Eckenabweichung der Hauptachsen und Mitschleppachsen nur näherungsweise betrachtet. D.h. es wird angenommen, dass die ausgelassenen Sätze bezüglich der Abweichung des Überschleifens vernachlässigt werden können.

Programmierbeispiel

Relevante Satzlänge

```
#CONTOUR MODE [DEV, PATH_DEV 5, RELEVANT_PATH 2]
N03 G01 X0 Y0 Z0 C0 F4
N907090 G04 X0.1
N04 X5 G261
N05 Y1
N09 X10 Y3 G260
N907091 Y0
```

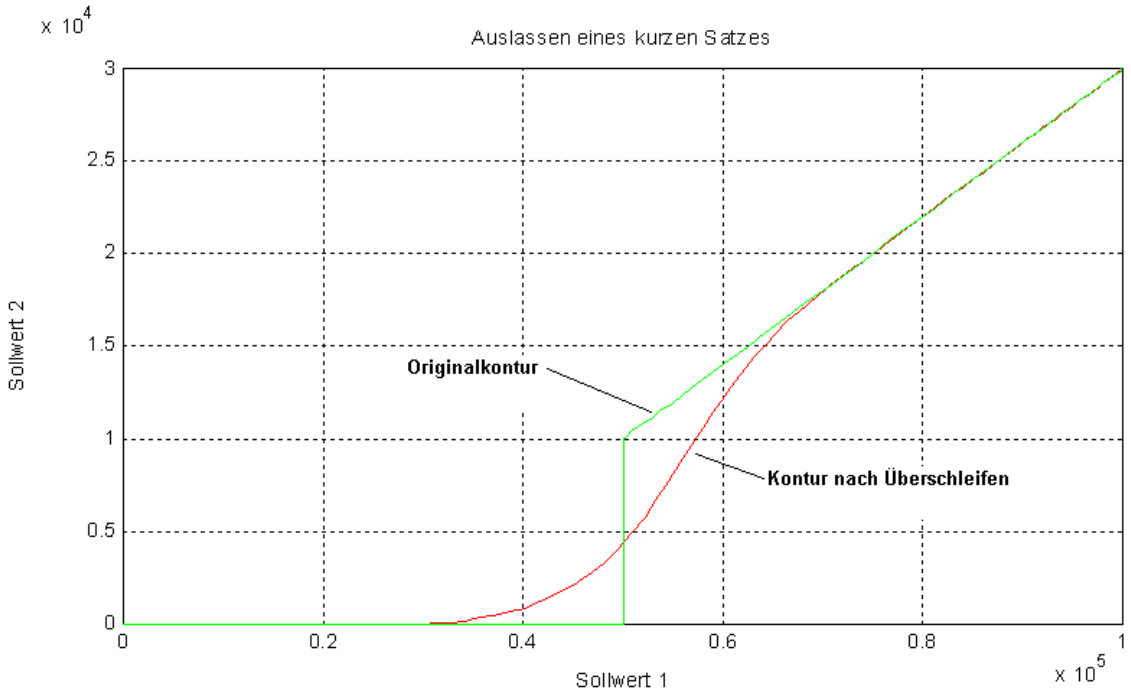


Abb. 76: Auslassen eines kurzen Satzes N05 beim Überschleifen

Sonderfall 1: Mehrere kurze Sätze in Folge nach dem Satzübergang

Sind mehrere nacheinander folgende Sätze (N20, N30, N40) kleiner als der angegebene Mindestfahrweg, so werden diese ausgelassen, solange der Abstand der Zielposition von der letzten noch relevanten Endposition (N10) kleiner als der angegebene Mindestfahrweg ist. Liegt die Zielposition des ausgelassenen Satzes außerhalb dieser Hüllkugel, so wird dieser Satz (N40) für die Berechnung der Überschleifkurve herangezogen, auch wenn er selbst kürzer als die angegebene Mindestlänge ist. Hierdurch erhält man selbst bei mehreren in Folge ausgelassenen Sätzen eine geringe Abweichung von der Originalkontur.

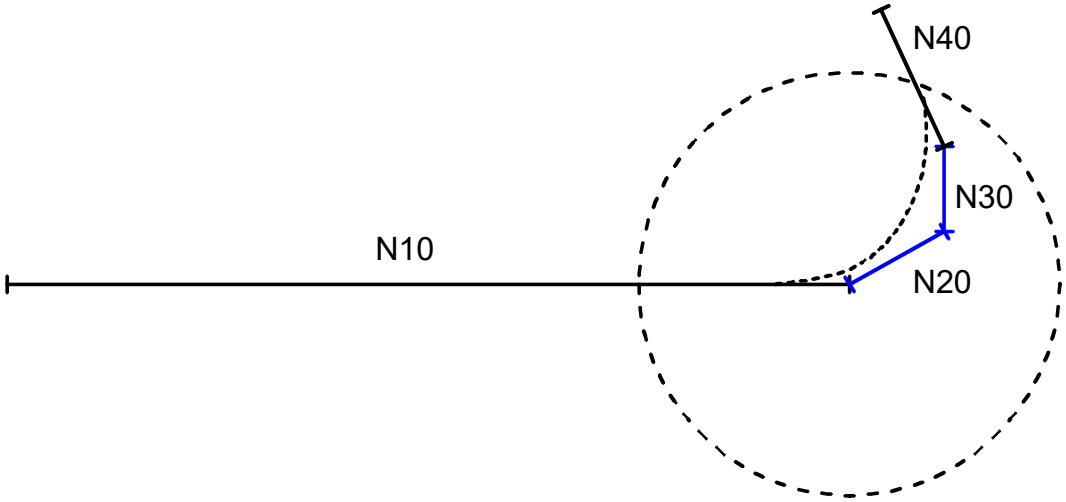


Abb. 77: Einzelne Sätze (N20, N30 und N40) sind zu kurz, jedoch liegt die Zielposition außerhalb der Mindestsatzlänge

Sonderfall 2: Mehrere kurze Sätze in Folge nach Satzübergang, letzter Satz sehr kurz

Im Ausnahmefall kann der Satz N40 selbst kürzer als die für das Überschleifen erforderliche Systemmindestlänge (ca. 16 µm.) sein. In diesem Falle wird die letzte Endposition mit der neuen Zielposition durch einen Linearsatz verbunden. Dieser neue Linearsatz N20´ wird dann für die Berechnung der Überschleifkurven herangezogen.

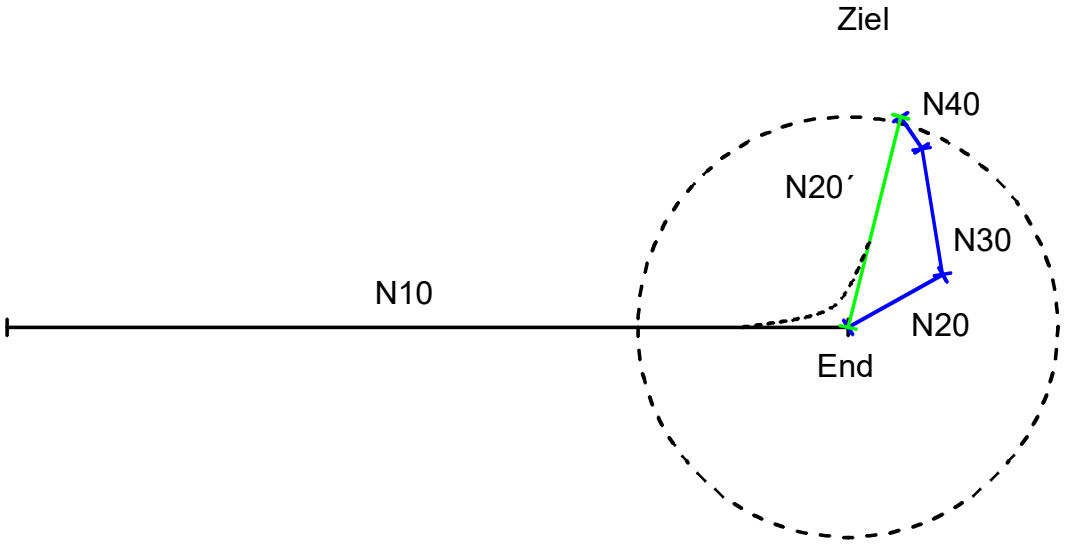


Abb. 78: Einzelne Sätze (N20, N30 und N40) sind zu kurz, jedoch überschreitet die Summe aller Sätze die Mindestsatzlänge

Sonderfall 3: Kurze Sätze vor dem Satzübergang

Sind die Sätze bei Beginn des Überschleifens (vor dem Satzübergang) bereits kürzer als die durch das Verfahren gegebene Mindestlänge, so werden diese Sätze ausgelassen. Die Sätze werden solange ausgelassen, bis der Abstand zwischen letzter gültiger Position und aktueller Zielposition die Mindestsatzlänge überschreitet. Ist dies der Fall, so wird die letzte Endposition und die aktuelle Zielposition durch einen Linearsatz N10' verbunden, welcher dann als Startsatz für das Überschleifen verwendet wird.

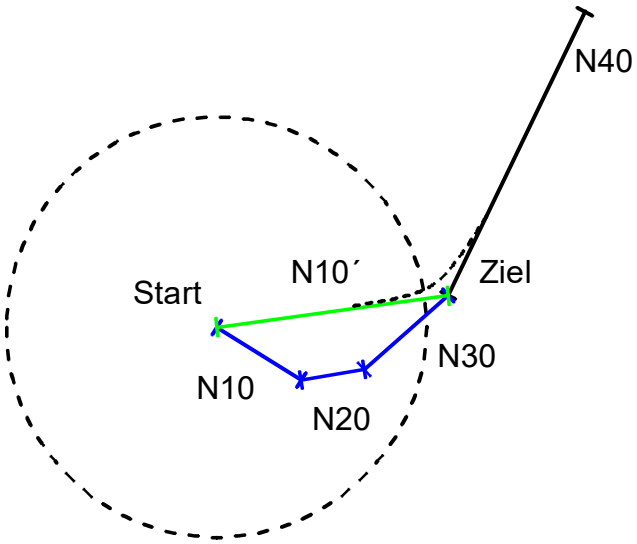


Abb. 79: Mehrere Sätze (N10, N20 und N30) sind zu kurz, jedoch überschreitet die Summe aller Sätze die systemgegebene Mindestsatzlänge

Sonderfall 4: Abwahl des Überschleifens oder Ändern der Parametrierung

Wird während des Auslassens von Sätzen das Überschleifen abgewählt oder werden die Randbedingungen für das Überschleifen geändert, so darf das aktuelle Überschleifen nur bis zur Abwahl / Parameteränderung durchgeführt werden. Danach kann das Überschleifen evtl. mit den neuen Parametern fortgesetzt werden.

```
#CONTOUR MODE [ DEV, PATH_DEV 5, RELEVANT_PATH 2]
N10 G91 G01 F1000 X10     G61
N20 X2 Y1
N30 Y1.5
N40 X-1 Y2…
```

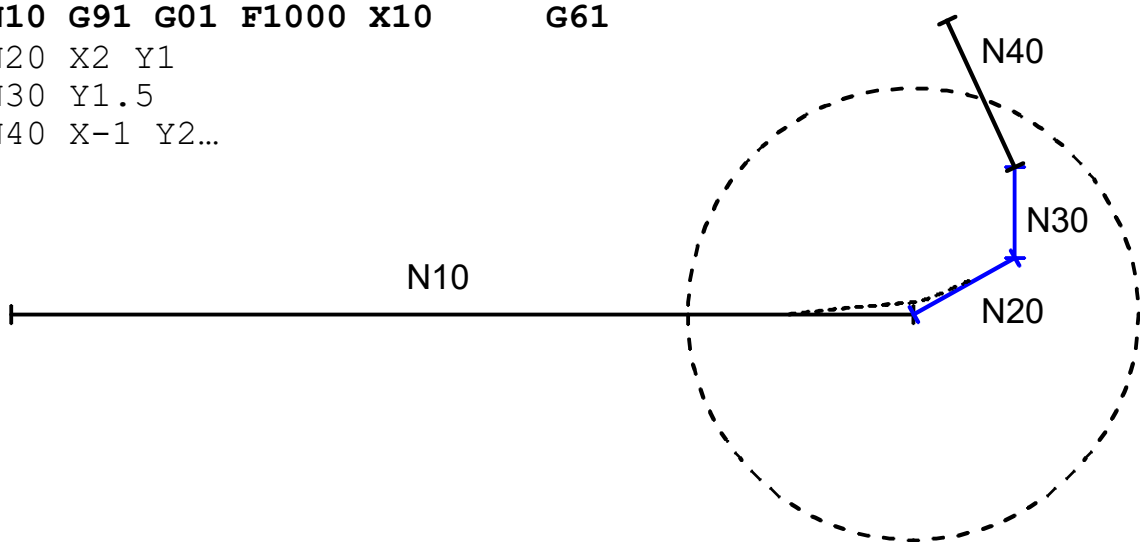


Abb. 80: Einzelne Sätze (N20, N30 und N40) sind zu kurz, jedoch wurde das Überschleifen bereits ab Satz N20 abgewählt.

11.2.2.3 Abarbeitung von zusätzlichen Sätzen

Wird zusätzlich zu den Bewegungssätzen an der Satzgrenze (N10 – N20) ein Befehl ohne Konturinformation (z.B. quittierungspflichtige M-Funktion mit Ausgabe vor Satz und Synchronisation nach Satz, MVS_SNS) programmiert, so kann dieser wahlweise vor, während oder nach der Überschleifkurve ausgeführt werden.

Programmierbeispiel

Abarbeitung von zusätzlichen Sätzen

```
N10 X100 G61 M25
N20 Y100
```

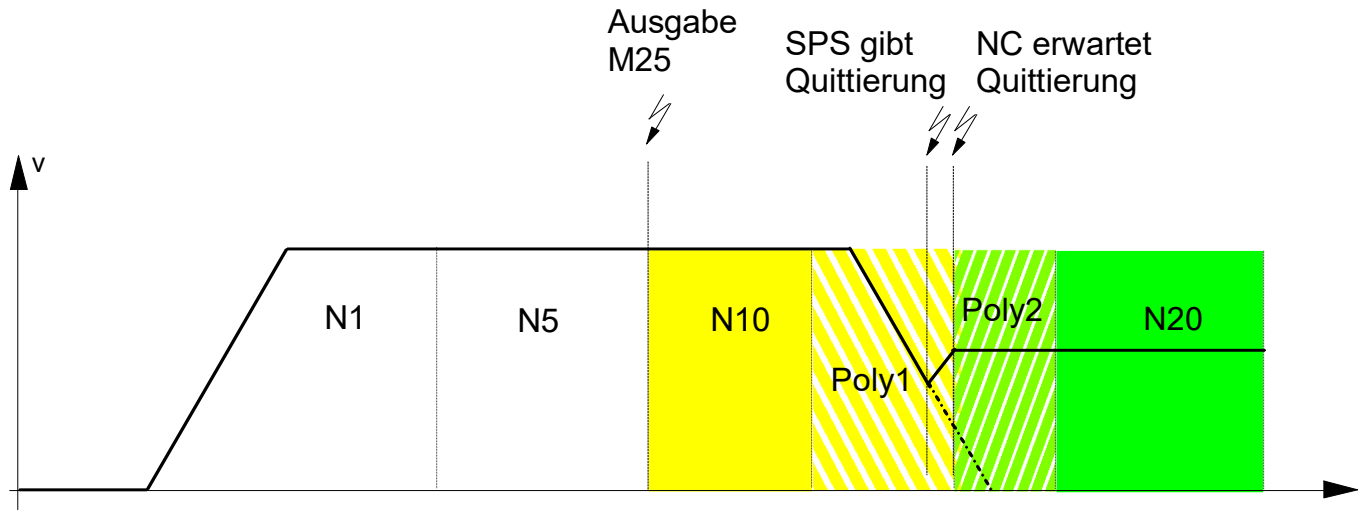


Abb. 81: Synchronisation mit nicht konturrelevanten Aktionen während Überschleifen

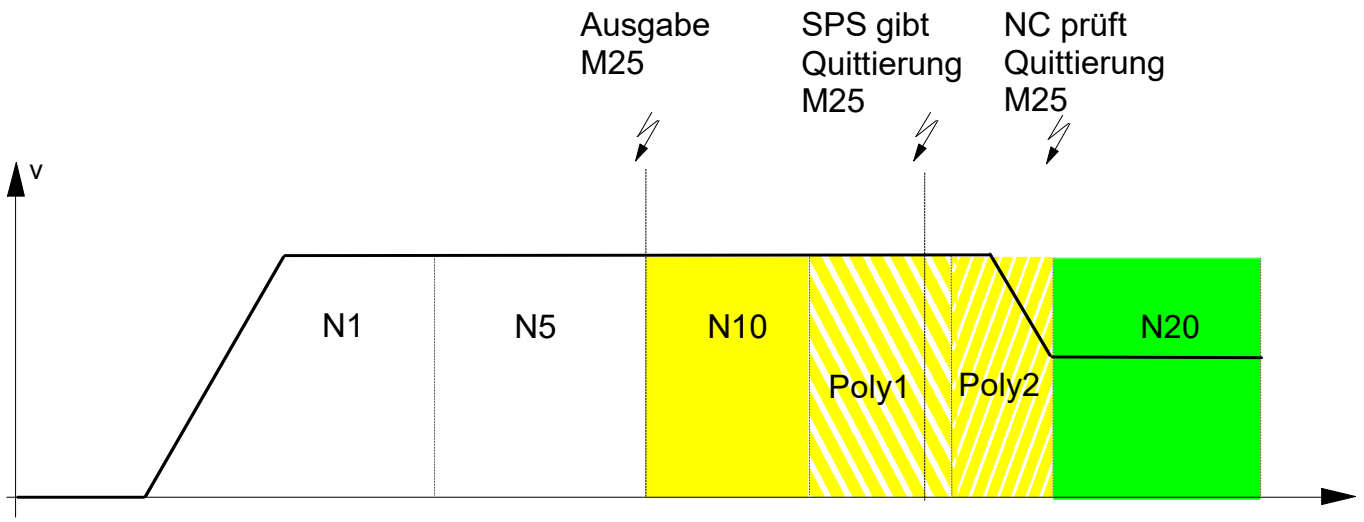


Abb. 82: Synchronisation mit nicht konturrelevanten Aktionen nach Überschleifen

Für die Ausführung dieser Befehle existieren folgende 3 Möglichkeiten:

1. direkt nach Vorsatz (N10) und vor dem ersten Überschleifpolynom
2. zwischen erstem und zweitem Überschleifpolynom
3. nach zweitem Überschleifpolynom und vor dem Nachsatz (N20)

11.2.2.4 Ruck innerhalb des Polynoms

Durch die Krümmung des Polynoms ergibt sich ein Ruck quer zur Bahn für die Achsen. Dieser Ruck wird normalerweise mit den maximalen Dynamikparametern der Achse überwacht (P-AXIS-00199). Ist der Ruck zu groß, so wird die Bahngeschwindigkeit entsprechend reduziert. Bei einzelnen Anwendungsfällen kann diese Geschwindigkeitsreduktion aufgrund des maximalen Rucks nicht gewünscht sein. Das lässt sich durch entsprechende Steuerkommandos im NC-Befehle #CONTOUR MODE spezifisch einstellen. Die Steuerkommandos überschreiben die Voreinstellung der Kanalparameterliste P-CHAN-00110 und sind modal bis zum Programmende gültig.

Im nachfolgenden Beispiel wird der Satzübergang von N6 nach N7 durch Polynome überschliffen, bei welchen der Ruck berücksichtig wird. Der Übergang N7 nach N8 wird ebenso überschliffen, jedoch ohne Berücksichtigung des Rucks auf der Bahnkontur.

Programmierbeispiel

Ruck innerhalb des Polynoms

```
%poly_jerk.nc
(Standardeinstellung in der Kanalparameterliste:-
(check_jerk_on_poly_path)

#SLOPE[TYPE=TRAPEZ]
#CONTOUR MODE [ DEV, PATH_DEV 4, RELEVANT_PATH 51]
N0003 G1 X0 Y100 Z0 F4

N0004 G261
N0005 G1 G91 X100
N0006 Y-50
N0007 #CONTOUR MODE [CHECK_JERK=1]
N0008 X100
N0009 #CONTOUR MODE [CHECK_JERK=0]
N0010 Y-50
N0009 G260
N0055 M30
```

11.2.2.5 Geschwindigkeitsverlauf im Überschleifbereich

Abhängig von der Achsparametrierung und Anwendung kann es erforderlich sein, den Geschwindigkeitverlauf im Überschleifbereich zu beeinflussen. In der Defaulteinstellung wird der Überschleifbereich mit maximal zulässiger Bahngeschwindigkeit durchfahren. Bei stark unterschiedlicher Achsdynamik der beteiligten Achsen kann dies zu einer unzulässig hohen Schwingungsanregung der Maschine führen, da die Bahngeschwindigkeit im Überschleifbereich angepasst wird.

Das Verhalten im Überschleifbereich lässt sich durch entsprechende Steuerkommandos im NC-Befehle #CONTOUR MODE einstellen.

Im nachfolgenden Beispiel wird der Satzübergang von N6 nach N7 durch Polynome überschliffen, bei dem im Überschleifbereich mit maximaler Geschwindigkeit durchfahren wird, d.h. bei unterschiedlicher Achsdynamik erfolgt hier die Geschwindigkeitsanpassung. Der Übergang N9 nach N10 wird ebenso überschliffen, jedoch ohne Geschwindigkeitsanpassung im Überschleifbereich, d.h. die Bahngeschwindigkeit ist im Überschleifbereich konstant.

Programmierbeispiel

Geschwindigkeitsverlauf im Überschleifbereich

```
%poly_const_speed
N0003 #SLOPE[TYPE=TRAPEZ]
N0004 G1 X0 Y0 Z0 F8000
N0005 #CONTOUR MODE [CONST_VEL=0]
N0006 X100 G61
N0007 Y100
N0008 #CONTOUR MODE [CONST_VEL=1]
N0009 X0 G61
N0010 Y0
N0020 M30
```

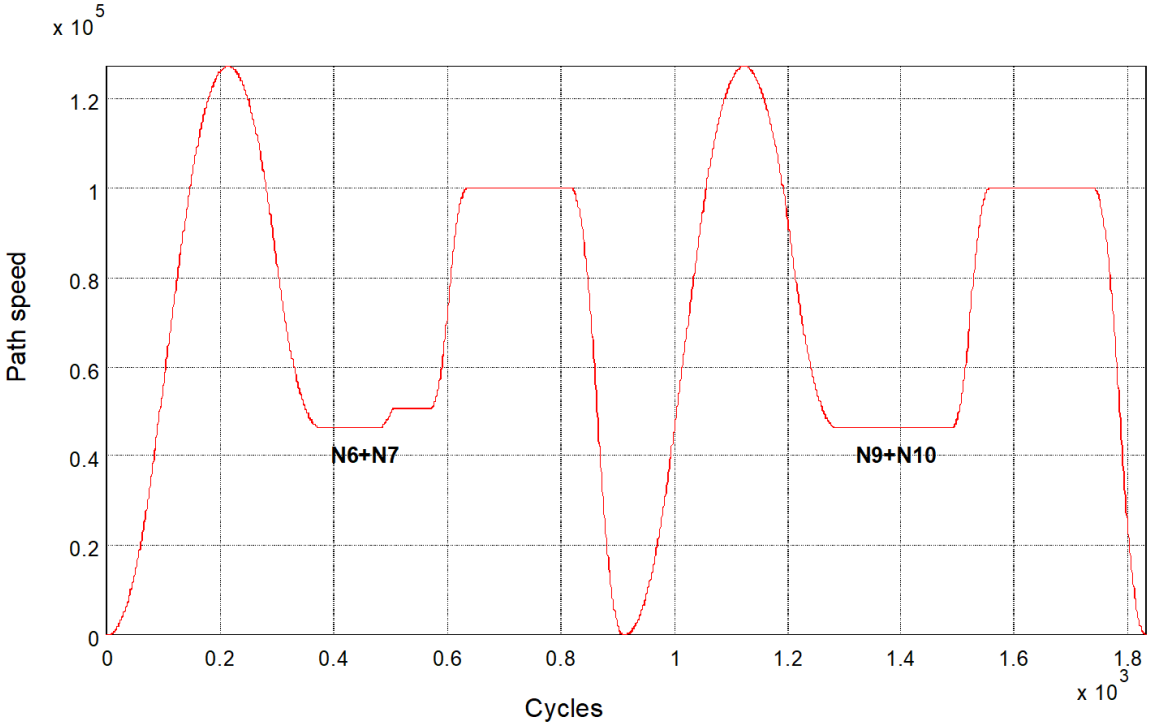


Abb. 83: Verhalten im Überschleifbereich

11.2.3 Parametrierung der Überschleifverfahren im NC-Programm (#CONTOUR MODE)

Die einzelnen Optionen werden über den NC-Befehl **#CONTOUR MODE** vor dem eigentlichen Aktivieren des Überschleifens (G61/G261) parametriert.

Abhängig vom Überschleifverfahren stehen bestimmte Schlüsselworte zur Parametrierung zur Verfügung. Der Befehl besitzt folgenden syntaktischen Aufbau:

#CONTOUR MODE [<*Ueberschleifverfahren*> <*Parameter*> <*action*>]

<*Ueberschleifverfahren*>	DEV	Überschleifen mit Eckenabweichung (Default)
	DIST	Überschleifen mit Eckenabstand
	DIST_SOFT	Dynamisch optimiertes Überschleifen
	DIST_MASTER	Dynamisch optimiertes Überschleifen mit Leitachse
	POS	Überschleifen mit Zwischenpunkt
	PTP	Dynamisch optimiertes Überschleifen der Kontur

<*Parameter*>	PATH_DEV TRACK_DEV ...	**Achtung:** Die Parameter für Abweichungen und Toleranzen werden in [mm, inch] oder [°] angegeben! Für die Angabe in [inch] bitte Hinweis auf P-CHAN-00439 beachten.

<*action*>	PRE_ACTION	M/H-Aktionen ausführen bzgl. der Überschleifkurve
	INTER_ACTION	
	POST_ACTION	

11.2.4 Aktivierung der Überschleifverfahren im NC-Programm

Die Aktivierung des Überschleifens erfolgt nach der Parametrierung des entsprechenden Überschleifverfahrens durch die G-Funktion G61 (satzweise) bzw. G261 (modal).

Versionshinweis

Ab der Version V2.11.2022.13…

…kann alternativ durch die zusätzliche Angabe von ON/OFF im Befehl #CONTOUR MODE das Überschleifen mit an-/abgewählt werden. Die Programmierung von G261/G260 ist dann nicht mehr notwendig.

Programmierbeispiel

Aktivierung der Überschleifverfahren im NC-Programm

```
%Contour_on_off
N10 G90 G01 X0 Y0 Z0 A0 C0 F60
N20 #CONTOUR MODE ON [DEV PATH_DEV=1.0]     ;Parametrierung und
                                            ;Aktivierung (= G261)
N30 X100
N40 Y100
N50 X0
N60 Y0
N70 #CONTOUR MODE OFF                       ;Deaktivierung (= G260)
N80 M30
```

11.2.4.1 Überschleifen mit Eckenabweichung

Hinweis

Die Standardparametrierung dieser Überschleifart ist nach Programmstart wirksam.

Die Eckenabstände, um welche die angrenzenden Verfahrsätze verkürzt werden, werden nach rein geometrischen Betrachtungen automatisch so bestimmt, dass eine vom Anwender vorgegebene Eckenabweichung nicht überschritten wird.

Die Eckenabstände werden gemäß der vorgegebenen verbleibenden minimalen Satzlänge jeweils begrenzt, wobei beide Abstände symmetrisch begrenzt werden. Die programmierte Bahngeschwindigkeit hat hier auf die Überschleifkontur keinen Einfluss.

Für optimales Überschleifen sollte der Parameter RELEVANT_PATH verwendet werden. Es wird empfohlen, den Wert der maximalen Eckenabweichung PATH_DEV zu übernehmen.

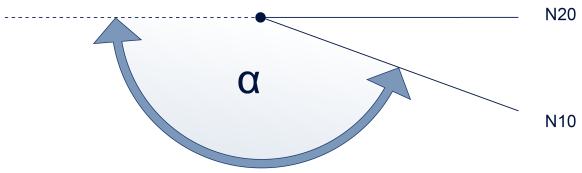
Syntax zur Parametrierung:

#CONTOUR MODE [DEV [**PATH_DEV**=..] [**RELEVANT_PATH**=..] [**TRACK_DEV**=..] [**RELEVANT_TRACK**=..] [**REMAIN_PART**=..] [<*action*>] [**CHECK_JERK**=..] [**MAX_ANGLE**=..] [**CONST_VEL**=..] **]**

DEV	Überschleifen mit maximaler Eckenabweichung
PATH_DEV=..	Maximale Abweichung von der programmierten Kontur in [mm, inch *] Standardwert: 1 mm * bei aktivem P-CHAN-00439
RELEVANT_PATH=..	Minimale Bahnlänge relevanter Sätze in [mm, inch *]. Standardwert: 0 mm * bei aktivem P-CHAN-00439
TRACK_DEV=..	Maximale Abweichung der Mitschleppachsen in [°] Standardwert: 0°
RELEVANT_TRACK=..	Mindestweg der Mitschleppachsen für relevante Sätze in [°] Standardwert: 0°
REMAIN_PART=..	Verbleibender Anteil in [0%-100%] des Originalsatzes Standardwert: 0 %
<*action*>	Kennung für den Ausführungszeitpunkt zusätzlicher Aktionen (M/H): PRE_ACTION: Aktionen **vor** der Überschleifkurve. INTER_ACTION: Aktionen **in** der Überschleifkurve (Standard). POST_ACTION: Aktionen **nach** der Überschleifkurve.
CHECK_JERK=..	Überwachung des Rucks, hervorgerufen durch die Krümmung des Polynoms (vgl. P-CHAN-00110) mit: 0: Ohne Rucküberwachung (Standard). 1: Rucküberwachung basierend auf der geom. Rampenzeit (P-AXIS-00199). Evtl. wird hierdurch die Bahngeschwindigkeit reduziert. 2: Rucküberwachung basierend auf den Rampenzeiten P-AXIS-00195 bis.P-AXIS-00198 des nichtlinearen Geschwindigkeitsprofils.

MAX_ANGLE=.. Maximaler Konturknickwinkel in [°] für Übergänge zwischen zwei Linearsätzen, bis zu dem überschliffen wird.

Standardwert: 178° (d.h. gesamte Kontur wird überschliffen)



CONST_VEL=.. Konstante Bahngeschwindigkeit im Überschleifbereich mit:

0: Ohne konstante Bahngeschwindigkeit (Standard).

1: Mit konstanter Bahngeschwindigkeit.

Programmierbeispiel

Überschleifen mit Eckenabweichung

```
...
N100 #CONTOUR MODE [DEV PATH_DEV=5]
N110 G01 X100 G61
N120 G01 Y100
...
```

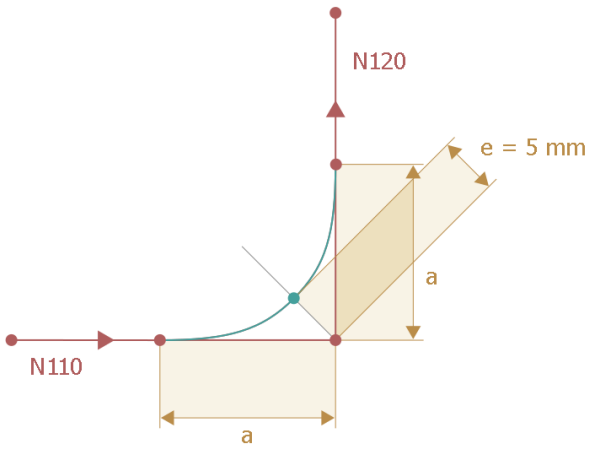


Abb. 84: Überschleifen mit Eckenabweichung

11.2.4.2 Überschleifen mit Eckenabstand

Falls der Punkt, ab dem die Originalkontur verlassen werden darf, bekannt ist, kann der Anwender die Eckenabstände des Vor- und Nachsatzes, um welche die angrenzenden Bewegungssätze verkürzt werden sollen, direkt angeben.

Die Eckenabstände werden so beschränkt, dass die vorgegebene minimal verbleibende Satzlänge nicht unterschritten wird.

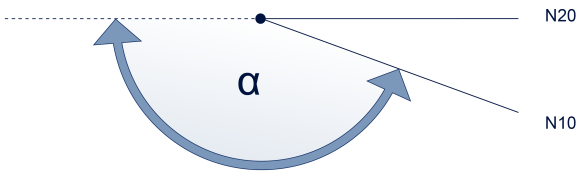
Werden die Eckenabstände a und b identisch angegeben, so wird bei einer Begrenzung eines Eckenabstands auf die minimal verbleibende Mindestsatzlänge auch der andere Eckenabstand symmetrisch begrenzt.

Werden die Eckenabstände a und b verschieden angegeben, wird bei einer Begrenzung nur der entsprechend zu große Abstand verkleinert. Dies kann bei ungleichen Satzlängen zu einem „Ausholen“ der Kontur führen, was aber evtl. erwünscht ist.

Syntax zur Parametrierung:

#CONTOUR MODE [DIST [**PRE_DIST**=..] [**POST_DIST**=..] [**RELEVANT_PATH**=..]
[**RELEVANT_TRACK**=..] [**TRACK_DEV**=..] [**REMAIN_PART**=..]
[<*action*>] [**CHECK_JERK**=..] [**MAX_ANGLE**=..] [**CONST_VEL**=..] **]**

DIST	Überschleifen mit Angabe des Eckenabstands
PRE_DIST=..	Eckenabstand in [mm, inch *], ab dem von der Originalkontur abgewichen wird. Standardwert: 1 mm *bei aktivem P-CHAN-00439
POST_DIST=..	Eckenabstand in [mm, inch *], bei dem auf die Originalkontur zurückgekehrt wird. Standardwert: 1 mm *bei aktivem P-CHAN-00439
RELEVANT_PATH=..	Minimale Bahnlänge relevanter Nachsätze in [mm, inch *] Standardwert: 0 mm
RELEVANT_TRACK=..	Mindestweg der Mitschleppachsen für relevante Nachsätze in [°]. Standardwert: 0°
TRACK_DEV=..	Maximale Abweichung der Mitschleppachsen in [°] Standardwert: 0°
REMAIN_PART=..	Verbleibender Anteil in [0%-100%] des Originalsatzes Standardwert: 0 %
<action>	Kennung für den Ausführungszeitpunkt zusätzlicher Aktionen (M/H): PRE_ACTION: Aktionen **vor** der Überschleifkurve. INTER_ACTION: Aktionen **in** der Überschleifkurve (Standard). POST_ACTION: Aktionen **nach** der Überschleifkurve.
CHECK_JERK=..	Überwachung des Rucks, hervorgerufen durch die Krümmung des Polynoms (vgl. P-CHAN-00110) mit: 0: Ohne Rucküberwachung (Standard). 1: Rucküberwachung basierend auf der geom. Rampenzeit (P-AXIS-00199). Evtl. wird hierdurch die Bahngeschwindigkeit reduziert. 2: Rucküberwachung basierend auf den Rampenzeiten P-AXIS-00195 bis P-AXIS-00198 des nichtlinearen Geschwindigkeitsprofils.
MAX_ANGLE=..	Maximaler Konturknickwinkel in [°] für Übergänge zwischen zwei Linearsätzen, bis zu dem überschliffen wird. Standardwert: 178° (d.h. gesamte Kontur wird überschliffen)



CONST_VEL=.. Konstante Bahngeschwindigkeit im Überschleifbereich mit:
0: Ohne konstante Bahngeschwindigkeit (Standard).
1: Mit konstanter Bahngeschwindigkeit.

Programmierbeispiel

Überschleifen mit Eckenabstand

```
...
N100 #CONTOUR MODE [DIST PRE_DIST=10 POST_DIST=5]
N110 G01 X100 G61
N120 G01 Y100
...
```

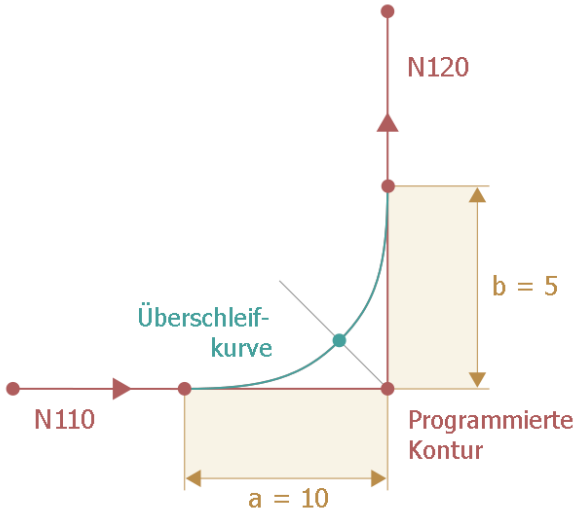


Abb. 85: Eckabstand-Überschleifen

11.2.4.3 Dynamisch optimiertes Überschleifen

Die Überschleifarten mit Eckenabweichung und Zwischenpunkt legen die Überschleifkurve durch eine richtungs- und krümmungsstetige Verbindung zweier Bewegungssätze fest. Diese Überschleifkurve führt auf die Achsen bezogen evtl. zu einer Schwankung der Beschleunigung.

Bei der achsbezogenen Betrachtung der möglichen Dynamikdaten (Beschleunigung, Ruck) wird die Überschleifkurve unter einer möglichst **gleichmäßigen Beschleunigung** (ruckminimal) der beteiligten Achsen bestimmt. Unter Ausnutzung einer maximalen Beschleunigung der Achsen wird zusätzlich die **Zeitdauer** des Überschleifvorgangs reduziert.

Syntax der Parametrierung:

#CONTOUR MODE [DIST_SOFT [**PATH_DIST**=..] [**TRACK_DIST**=..] [**ACC_MAX**=..] [**ACC_MIN**=..] [**RAMP_TIME**=..] [**DIST_WEIGHT**=..] **]**

DIST_SOFT	Dynamisch optimiertes Überschleifen
PATH_DIST=..	Eckenabstand zum Vor- und Nachsatz (symmetrisch) in [mm, inch *], ab dem von der Originalkontur abgewichen werden darf. Die Angabe bezieht sich auf den Fahrweg der Vorschubachsen. Standardwert : 1 mm Überwachung aus: -1 mm *bei aktivem P-CHAN-00439
TRACK_DIST=..	Eckenabstand zum Vor- und Nachsatz in [°], ab dem die Nicht-Vorschubachsen (Mitschleppachsen) von der Originalkontur abweichen dürfen. Standardwert: Wert wird automatisch von PATH_DIST übernommen, solange dieser noch nicht (seit Programmstart) explizit angegeben wurde. Überwachung aus: -1°
ACC_MAX=..	Prozentualer Anteil in [0%-100%] der maximalen Achsbeschleunigung (Maschinendatum), die durch den Konturverlauf genutzt werden darf. Standardwert : 100 %
ACC_MIN=..	Prozentualer Anteil in [0%-100%] der maximalen Achsbeschleunigung (Maschinendatum), die durch den Konturverlauf genutzt werden soll. Wird hierbei der vorgegebene Eckenabstand (s. PATH_DIST) nicht eingehalten, so wird die Beschleunigung bis zu Maximalwert (ACC_MAX) erhöht. Standardwert : 50 %
RAMP_TIME=..	Prozentuale Gewichtung der Rampenzeit in [0%-10000%]. Standardwert : 100 %
DIST_WEIGHT=..	Beeinflusst die Aufteilung verschliffener Linearsätze in [0%-100%]: In der Voreinstellung 0% werden alle Sätze halbiert, bei 100% entspricht das Teilungsverhältnis den Längen der benachbarten Sätze. Durch den Wert lassen sich beide Methoden prozentual vermischen. Standardwert : 0 %

Einschränkungen:

- Wird beim Überschleifen ein Zirkularsatz verwendet, so wird die Überschleifkurve ohne dynamische Optimierung mit Eckenabstand berechnet.
- Für die Berechnung wird nur eine Rampenzeit (Maximalwert der vier individuellen Rampenzeiten) verwendet.
- Keine Behandlung von kinematischen Transformationen. In diesem Fall wird ohne dynamische Optimierung mit Eckenabstand gerechnet.
- Die Gewichtung der Eckenabstände in Abhängigkeit des vorhergehenden bzw. nachfolgenden Satzes durch den Parameter DIST_WEIGHT ergibt in vielen Fällen eine bessere Ausnutzung der zur Verfügung stehenden Satzlänge.

Beim Verfahren des achsbezogenen Überschleifens sind die Eckenabstände des Vor- und Nachsatzes grundsätzlich gleich (symmetrisch). Werden die maximalen Eckenabstände zusätzlich auf den halben Satzfahrweg begrenzt, so ergibt sich bei längeren Fahrwegen aufgrund eines vorhergehenden/ nachfolgenden kurzen Fahrwegs ein geringerer Verschleifbereich und somit eine kleinere Überschleifgeschwindigkeit.

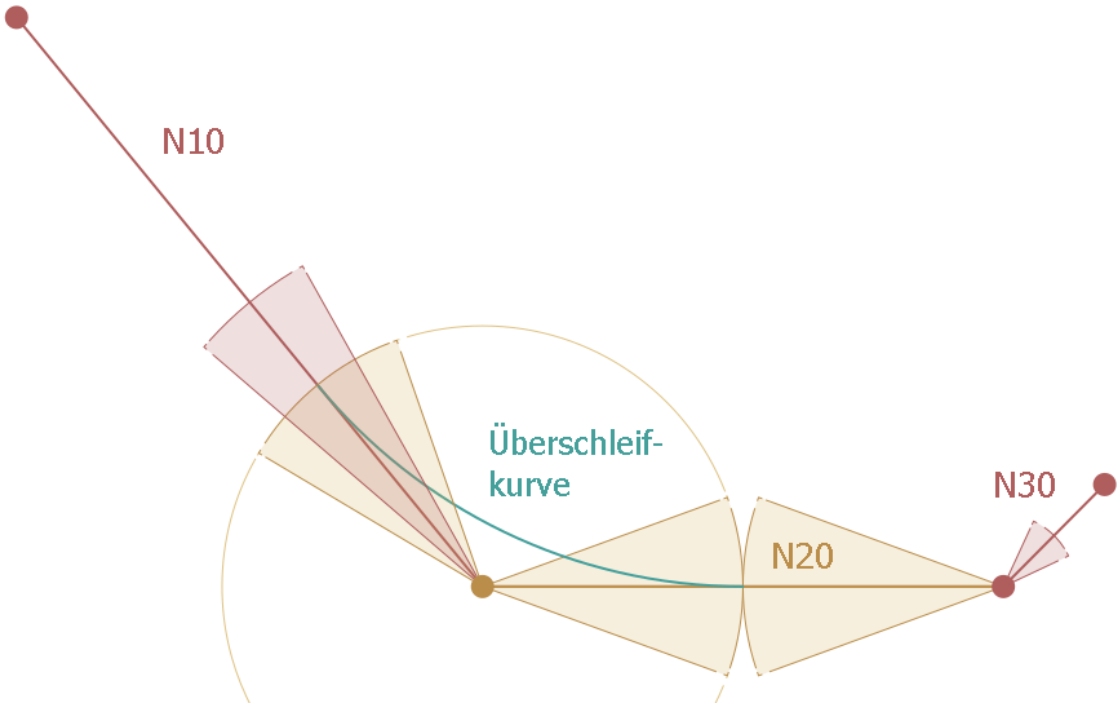


Abb. 86: Maximaler Eckenabstand des Satzes N20 unabhängig von den Satzlängen von N10 und N20 (DIST_WEIGHT = 0 %)

Werden die Länge des vorhergehenden und nachfolgenden Satzes für die Ermittlung der maximalen Eckenabstände mitberücksichtigt, so kann der Verschleifbereich erhöht werden.

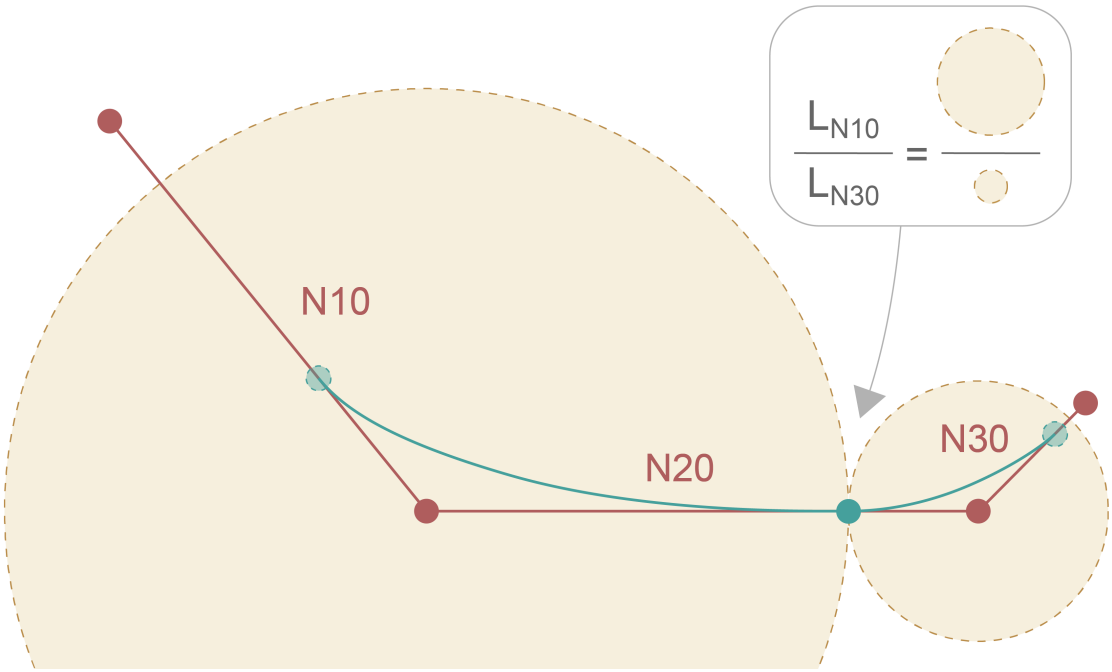


Abb. 87: Maximaler Eckenabstand des Satzes N20 unterteilt im Verhältnis der Satzlängen von N10 und N30 (DIST_WEIGHT = 100 %)

Programmierbeispiel

Dynamisch optimiertes Überschleifen

Vergleich des Überschleifens einer 90° Ecke mit den Methoden:

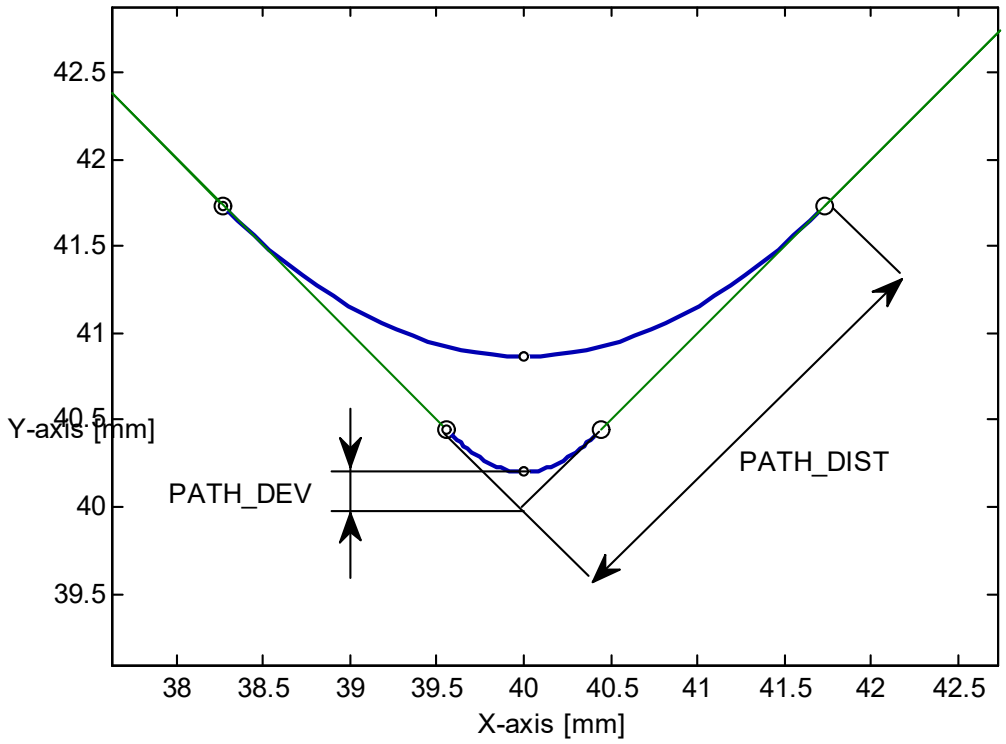
- Dynamisch optimiertes Überschleifen (DIST_SOFT):

```
N010 #CONTOUR MODE [DIST_SOFT PATH_DIST=12]
N020 G0 X0 Y80
N030 G261
N040 G01 X40 Y40 F2.5
N050 G01 X80 Y80
N060 G260
N070 M30
```

- Überschleifen mit Eckenabweichung (DEV):

```
N010 #CONTOUR MODE [DEV PATH_DEV=0.2]
N020 G0 X0 Y80
N030 G261
N040 G01 X40 Y40 F2.5
N050 G01 X80 Y80
N060 G260
N070 M30
```

Vergleich der Überschleifkurven:



11.2.4.4 Dynamisch optimiertes Überschleifen mit Leitachse

In dieser Variante des dynamisch optimierten Überschleifens wird eine Vorschubleitachse verwendet. Dadurch ergibt sich im Allgemeinen ein günstigeres Geschwindigkeitsprofil.

Die Vorschubleitachse ist in der Achsparameterliste über einen Eintrag in P-AXIS-00015 gekennzeichnet und in der Kanalparameterliste als alleinige Vorschubachse markiert (P-CHAN-00011).

Weitere Eigenschaften und Einschränkungen entsprechen dem dynamisch optimierten Überschleifverfahren.

Syntax der Parametrierung:

#CONTOUR MODE [DIST_MASTER [**SYM_DIST**=..] [**ACC_MAX**=..] [**ACC_MIN**=..]
[**RAMP_TIME**=..] [**DIST_WEIGHT**=..] **]**

DIST_MASTER	Dynamisch optimiertes Überschleifen mit Vorschubleitachse
SYM_DIST=..	Eckenabstand zum Vor- und Nachsatz (symmetrisch) in [mm, inch *], ab dem von der Originalkontur abgewichen werden darf. Standardwert : 1 mm Überwachung aus: -1 mm *bei aktivem P-CHAN-00439
ACC_MAX=..	Prozentualer Anteil in [0%-100%] der maximalen Achsbeschleunigung (Maschinendatum), die durch den Konturverlauf genutzt werden darf. Standardwert : 100 %
ACC_MIN=..	Prozentualer Anteil in [0%-100%] der maximalen Achsbeschleunigung (Maschinendatum), die durch den Konturverlauf genutzt werden soll. Wird hierbei der vorgegebene Eckenabstand (s. SYM_DIST) nicht eingehalten, so wird die Beschleunigung bis zu Maximalwert (ACC_MAX) erhöht. Standardwert : 50 %
RAMP_TIME=..	Prozentuale Gewichtung der Rampenzeit in [0%-10000%]. Standardwert : 100 %
DIST_WEIGHT=..	Prozentuale Gewichtung der Eckenabstände im Verhältnis zum vorhergehenden / nachfolgenden Satz in [0%-100%]. Standardwert : 0 %

11.2.4.5 Überschleifen mit Zwischenpunkt

Der Anwender gibt hier sowohl die Eckenabstände als auch einen Zwischenpunkt P vor, an den die beiden Polynomkurven angrenzen (Expertenmodus). Mit diesem Verfahren ist es z.B. möglich, durch Vorgabe des Eckenabstandes Null die programmierte Kontur beizubehalten und die Dynamik dennoch voll auszunutzen. D.h., die Eckenabstände müssen hier nicht unbedingt symmetrisch sein.

Syntax der Parametrierung:

#CONTOUR MODE [POS [**PRE_DIST**=..] [**POST_DIST**=..] [**X**..] [**Y**..] [**Z**..] [<*action*>]
[**CHECK_JERK**=..] [**CONST_VEL**=..] **]**

POS	Überschleifen mit Angabe des Zwischenpunktes
PRE_DIST=..	Eckenabstand in [mm, inch *], ab dem von der Originalkontur abgewichen wird. Die Angabe von 0 mm ist hier möglich. Standardwert: 1 mm *bei aktivem P-CHAN-00439
POST_DIST=..	Eckenabstand in [mm, inch *], bei dem auf die Originalkontur zurückgekehrt wird. Die Angabe von 0 mm ist hier möglich. Standardwert: 1 mm *bei aktivem P-CHAN-00439
X..	Position des Zwischenpunkts in der ersten Hauptachse in [mm, inch]
Y..	Position des Zwischenpunkts in der zweiten Hauptachse in [mm, inch]
Z..	Position des Zwischenpunkts in der dritten Hauptachse in [mm, inch]
<*action*>	Kennung für Ausführungszeitpunkt zusätzlicher Aktionen (M/H): PRE_ACTION: Aktionen **vor** der Überschleifkurve. INTER_ACTION: Aktionen **während** der Überschleifkurve (Standard). POST_ACTION: Aktionen **nach** der Überschleifkurve.
CHECK_JERK=..	Überwachung des Rucks, hervorgerufen durch die Krümmung des Polynoms (vgl. P-CHAN-00110) mit: 0: Ohne Rucküberwachung (Standard). 1: Rucküberwachung basierend auf der geom. Rampenzeit (P-AXIS-00199). Evtl. wird hierdurch die Bahngeschwindigkeit reduziert 2: Rucküberwachung basierend auf den Rampenzeiten P-AXIS-00195 bis P-AXIS-00198 des nichtlinearen Geschwindigkeitsprofils
CONST_VEL=..	Konstante Bahngeschwindigkeit im Überschleifbereich mit: 0: Ohne konstante Bahngeschwindigkeit (Standard). 1: Mit konstanter Bahngeschwindigkeit.

Programmierbeispiel

Überschleifen mit Zwischenpunkt

```
...
N100 #CONTOUR MODE [POS PRE_DIST=2 POST_DIST=3 X110 Y-10 Z0]
N110 G01 X100 G61
N120 G01 Y100
...
```

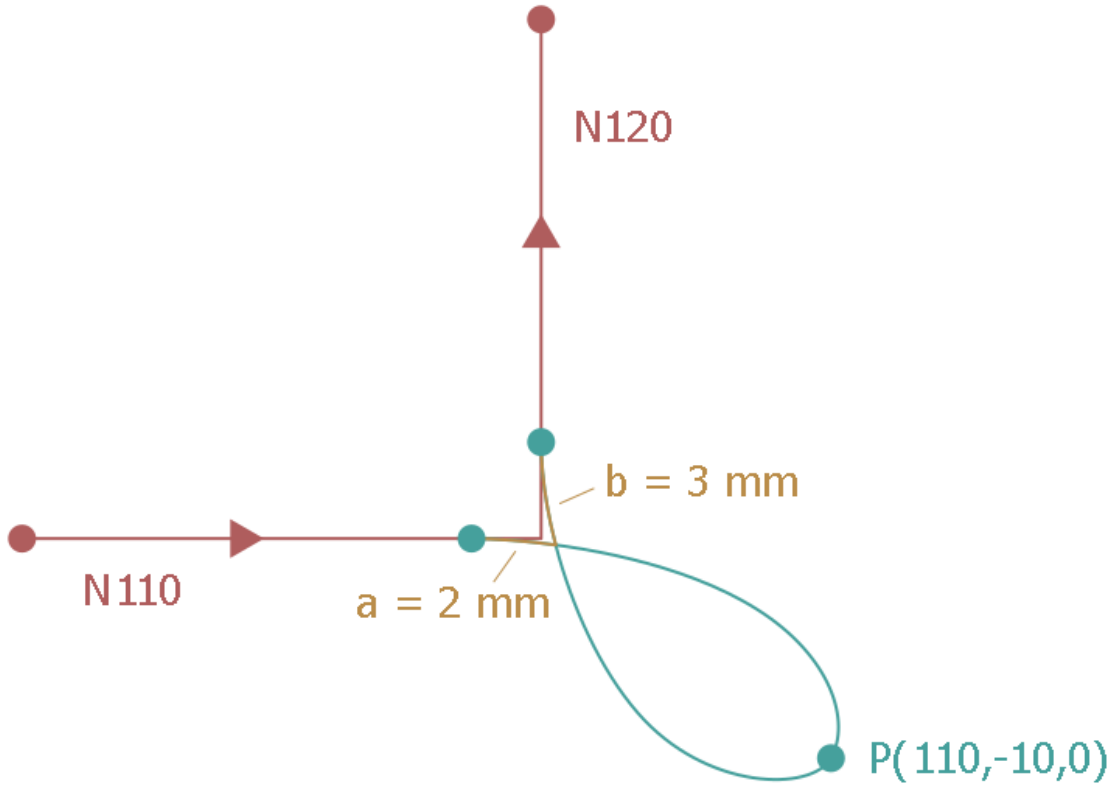
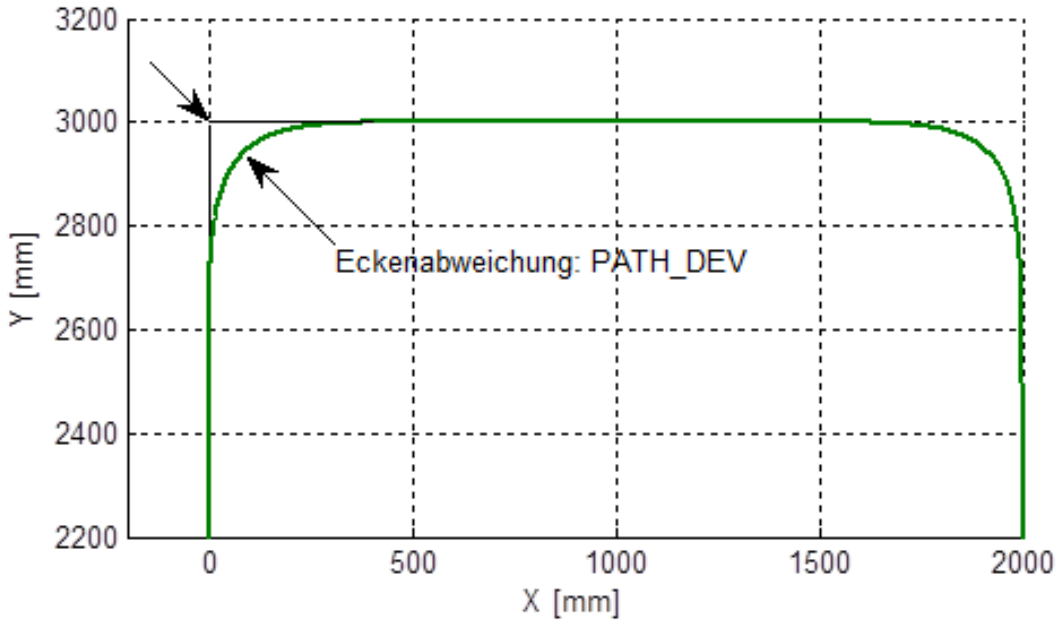


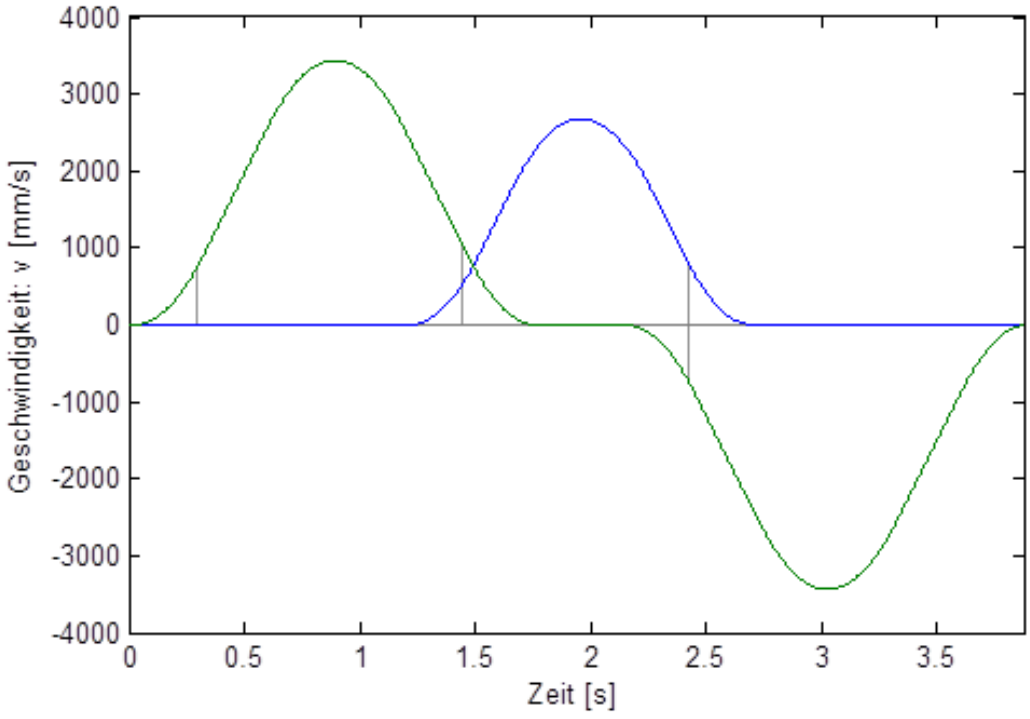
Abb. 88: Überschleifen mit Zwischenpunkt

11.2.4.6 Dynamisch optimiertes Überschleifen der gesamten Kontur

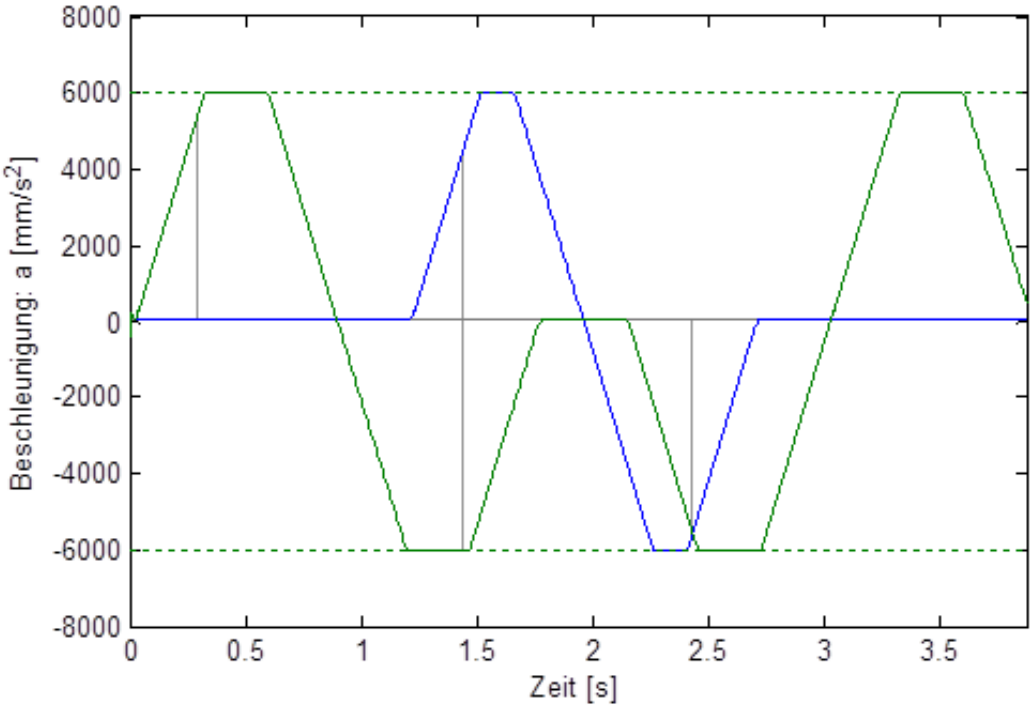
Dieses Verfahren ist für Handhabungsaufgaben geeignet, bei denen der Vorschub in der verrundeten Kontur nicht konstant sein muss. Die Überschleifkurve wird so gewählt, dass mindestens eine beteiligte Achse die zur Verfügung stehende Dynamik ausnutzt. Im Gegensatz zum dynamisch optimierten Überschleifen (DIST_SOFT) wird bei diesem Verfahren die ganze Kontur einbezogen. Folgende Abbildung zeigt eine typische Anwendung:



Die übergreifende Planung vermeidet unnötige Beschleunigungsnullstellen an den Satzgrenzen und berechnet somit gleichförmige Geschwindigkeitsprofile wie in der Abbildung unten dargestellt.



Um weitere Anregungen zu reduzieren, ändern sich Beschleunigungen mit konstantem Ruck. Dabei setzen die Beschleunigungsphasen bereits in den Geradenstücken vor und nach der verrundeten Kontur ein:



Die Eckenabweichung definiert den Abstand der verrundeten Kontur zum programmierten Eckpunkt.

Falls der Punkt, ab dem die Originalkontur verlassen werden darf, bekannt ist, kann der Anwender alternativ auch die Eckenabstände des Vor- und Nachsatzes, um welche die angrenzenden Bewegungssätze verkürzt werden sollen, direkt angeben. Die Eckenabstände werden so beschränkt, dass die vorgegebene minimal verbleibende Satzlänge nicht unterschritten wird.

`Syntax der Parametrierung:`

#CONTOUR MODE [PTP [**PATH_DEV**=..] [**PATH_DIST**=..] [**MERGE**=..] [<*action*>] **]**

PTP	Achsspezifisches Überschleifen mit Angabe des Eckenabstandes [ab V3.1.3052.01]
PATH_DEV=..	Maximale Eckenabweichung von der programmierten Kontur in [mm, inch *].
	Standardwert: 1 mm
	*bei aktivem P-CHAN-00439
PATH_DIST=..	Eckenabstand zum Vor- und Nachsatz (symmetrisch) in [mm, inch *], ab dem von der Originalkontur abgewichen werden darf. Die Angabe bezieht sich auf den Fahrweg der Vorschubachsen **[ab V3.1.3079.16].**
	Standardwert : 1 mm
	*bei aktivem P-CHAN-00439
MERGE=..	Zusammenfassen tangentialer Sätze **[ab V3.1.3079.16]** mit.
	0: Kein Zusammenfassen
	1: Zusammenfassen (Standard)
<*action*>	Kennung für Ausführungszeitpunkt zusätzlicher Aktionen (M/H) mit:
	PRE_ACTION: Aktionen **vor** der Überschleifkurve.
	INTER_ACTION: Aktionen **in** der Überschleifkurve (Standard).
	POST_ACTION: Aktionen **nach** der Überschleifkurve.

Achtung

Das Verfahren ist nicht geeignet für:

a) Programme mit vielen kurzen Verfahrbewegungen (siehe auch HSC).

b) Programme mit **Kreissätzen**, da eine automatische Abwahl des Verfahrens erfolgt.

Achtung

Voraussetzung für die Nutzung dieser Funktionalität ist die Parametrierung des Hochlaufparameters für jeden Kanal, in dem die Funktion verwendet werden soll.

Beispiel für die Einstellung in der Hochlaufliste :

```
configuration.channel[].path_preparation.function FCT_DEFAULT|FCT_PTP
```

Programmierbeispiel

Dynamisch optimiertes Überschleifen der gesamten Kontur

```
...
N100 #CONTOUR MODE [PTP PATH_DEV=5]
N110 G01 X100 G61
N120 G01 Y100
...
```

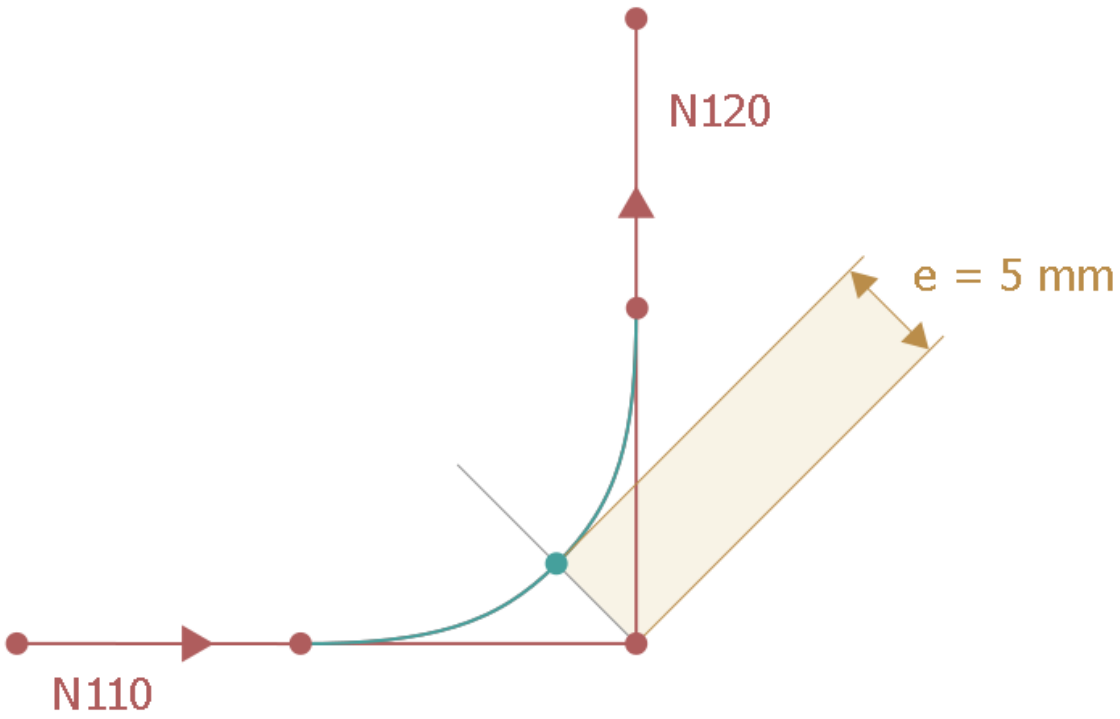


Abb. 89: Dyn. optimiertes Überschleifen der gesamten Kontur mit Angabe der Eckenabweichung

11.2.5 Beispiele

Programmierbeispiel

In nachfolgenden Beispielen soll der Einfluss der unterschiedlichen Ausgabe von M-Funktionen während des Überschleifens aufgezeigt werden.

```
N907090              X0 Y0
G91 G01 F6000
N01 #CONTOUR MODE [DEV PATH_DEV=10 POST_ACTION]

N10 X100 G61 M25     (MVS_SNS)
N20 Y100 F3000
N30 X100 G61 F6000
N40 G04 X2
N50 Y100
N00 X0 Y0

N60 X100 G61
N70 Y100 M26         (MVS_SVS)

N907091 G04 X1
```

Ausgabe vor der Überschleifkurve:

```
N01 #CONTOUR MODE [DEV PATH_DEV=10.0 PRE_ACTION]
```

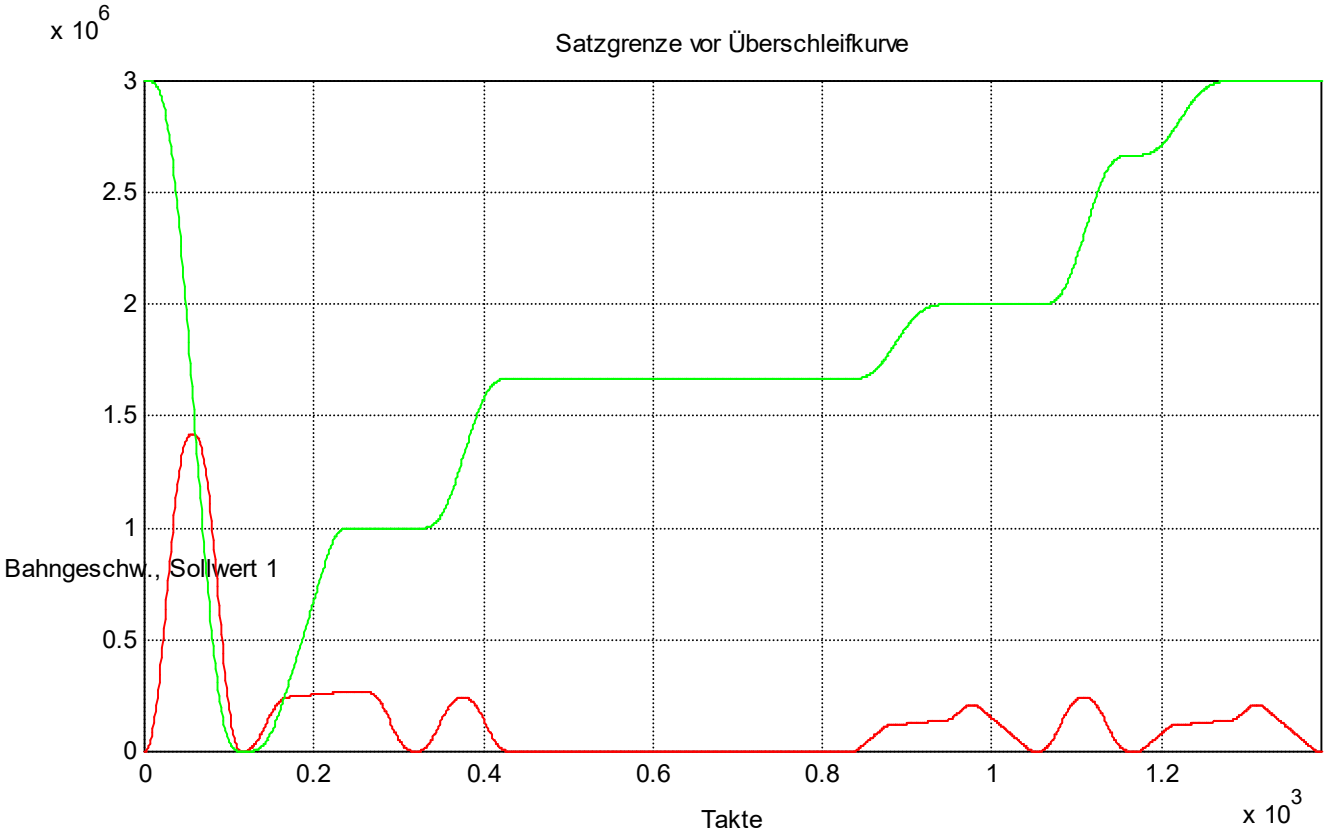


Abb. 90: Satzgrenze vor Überschleifkurve

Ausgabe innerhalb der Überschleifkurve:

```
N01 #CONTOUR MODE [DEV PATH_DEV=10.0 INTER_ACTION]
```

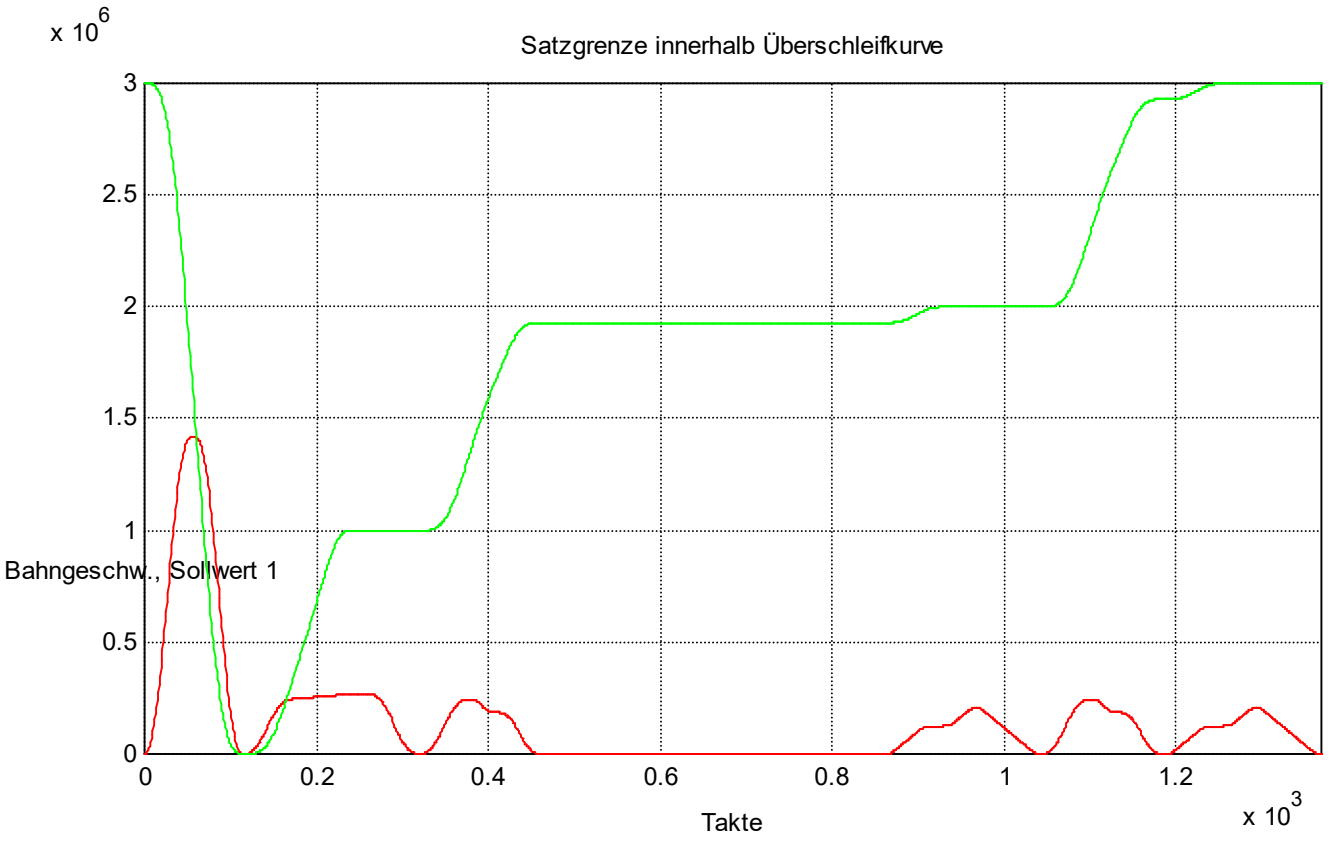


Abb. 91: Satzgrenze innerhalb Überschleifkurve

Ausgabe nach der Überschleifkurve:

```
N01 #CONTOUR MODE [DEV PATH_DEV=10.0 POST_ACTION]
```

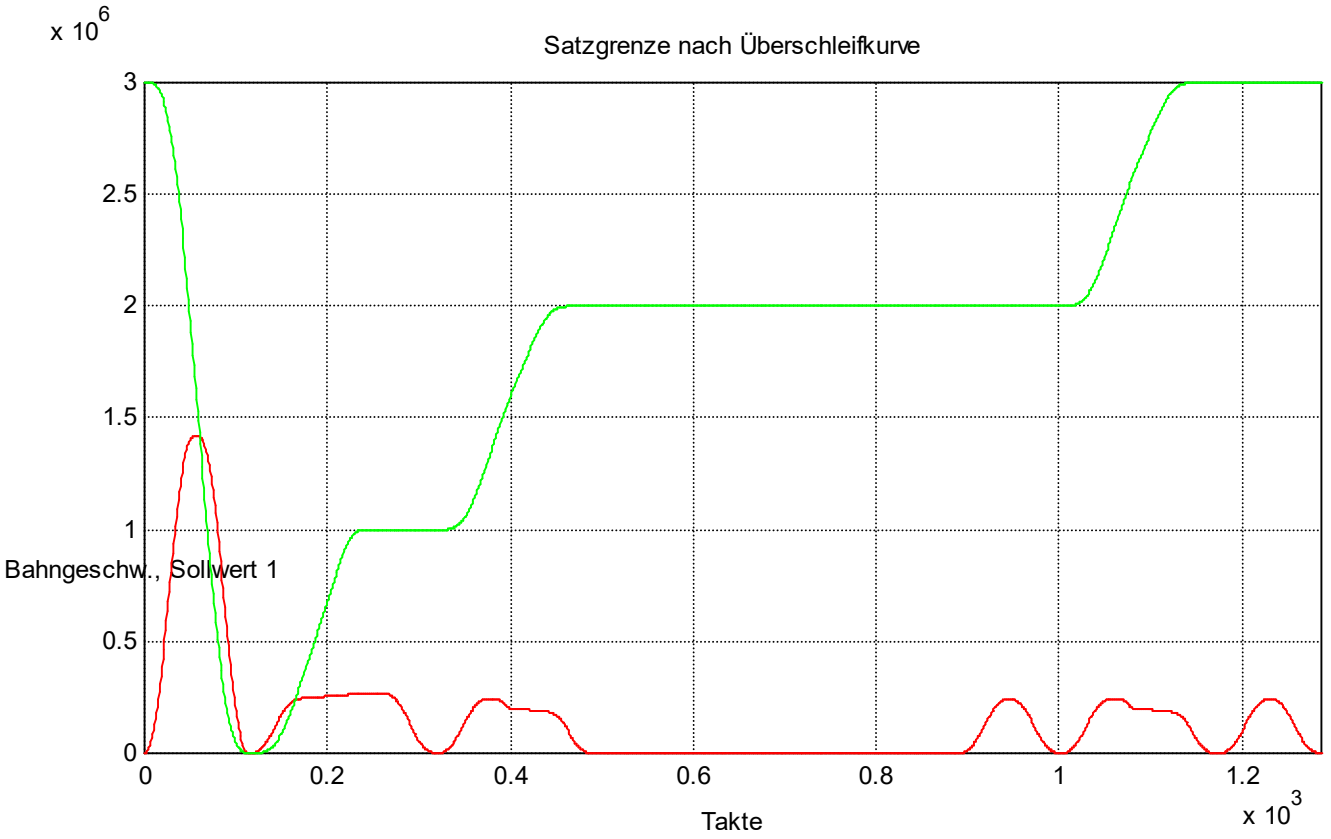


Abb. 92: Satzgrenze nach Überschleifkurve

Wird die Quittierung der M25 von Satz N10 zeitlich verzögert, so wird zunächst nach der Überschleifkurve angehalten und auf die SPS Quittierung gewartet.

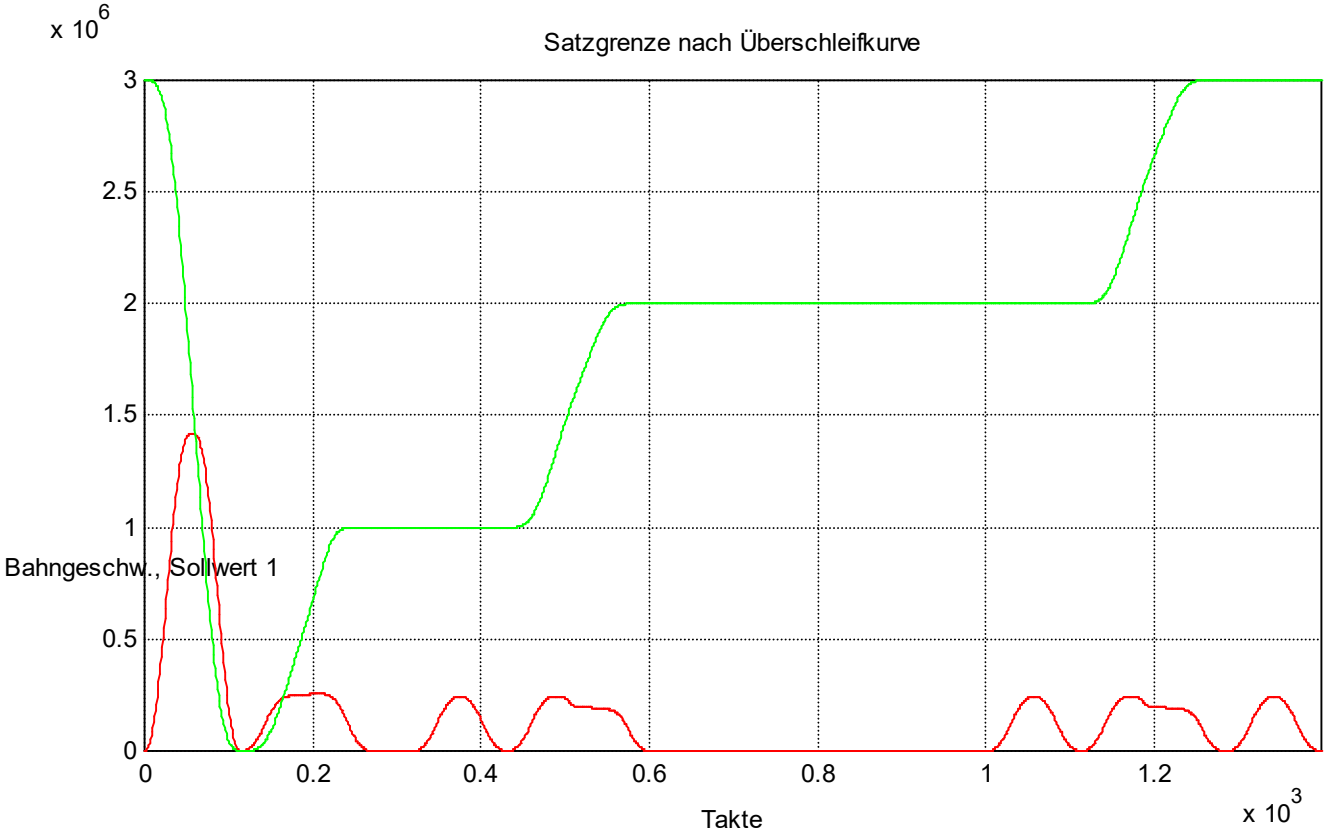


Abb. 93: Satzgrenze nach Überschleifkurve

Programmierbeispiel

Ändern des Grenzwinkels während des Überschleifens:

```
#CONTOUR MODE [DEV PATH_DEV=0.50 RELEVANT_PATH=0.1 TRACK_DEV=2 RELE-
VANT_TRACK=0.2]
F10000

G261
N5 #CONTOUR MODE [MAX_ANGLE=3]
N10 G01 X0 Y0 Z0 G61
N15 #CONTOUR MODE [MAX_ANGLE=4]
N20 G01 X100 Y0 Z0
N25 #CONTOUR MODE [MAX_ANGLE=5]
N30 G01 X100 Y100 Z0
N35 #CONTOUR MODE [MAX_ANGLE=6]
N40 G01 X0 Y0 Z0 G61
G260
```

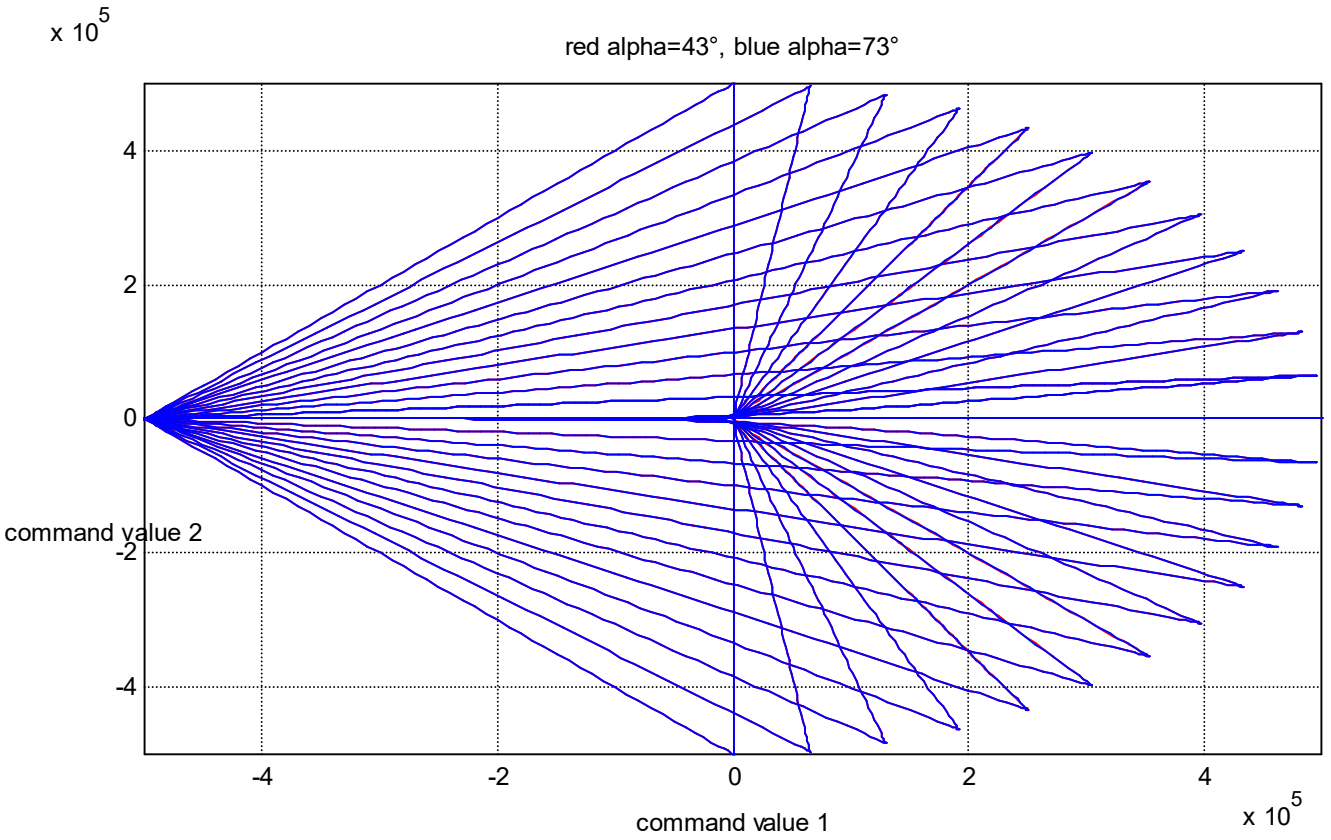
Ergebnis:

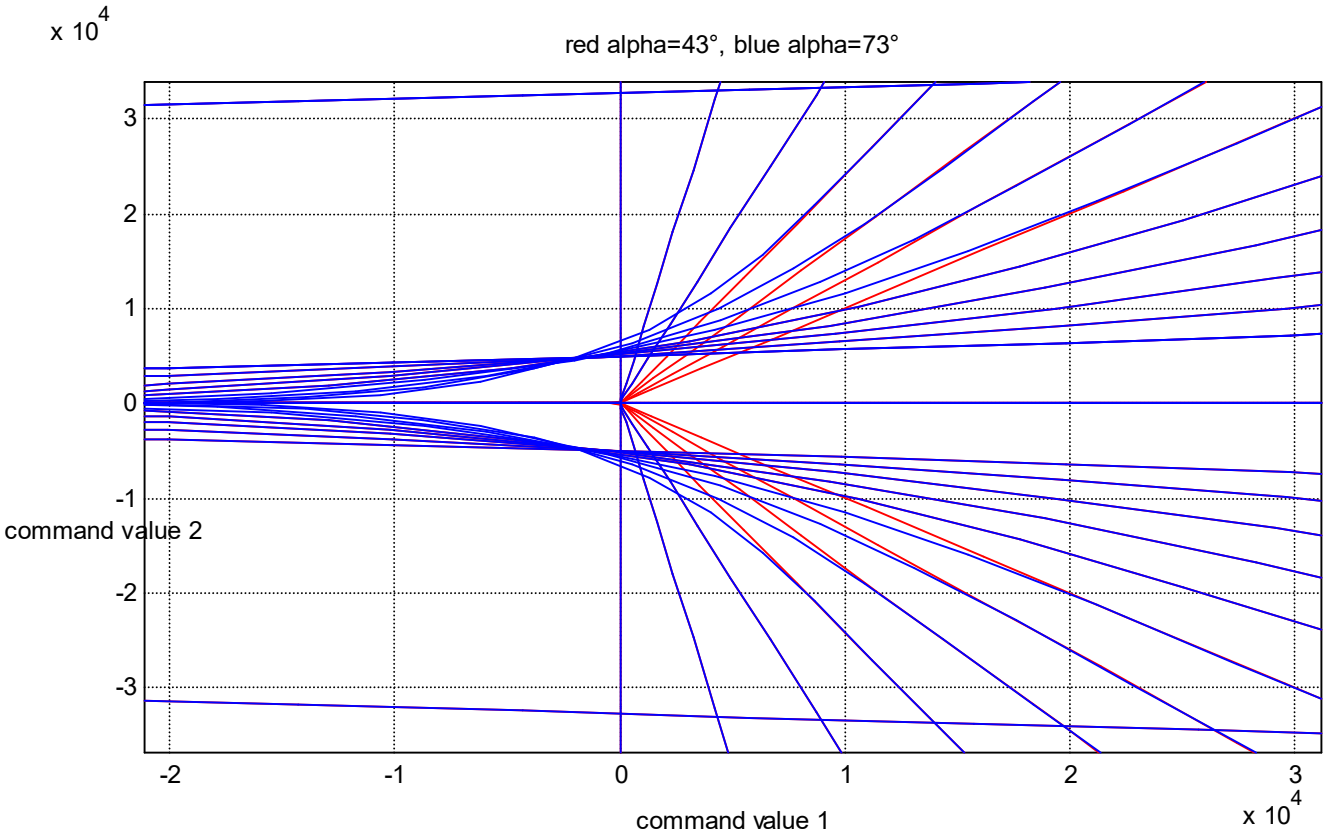
Überschleifen des Satzes N<i> findet immer mit dem Grenzwinkel des vorhergehenden Satzes N<i-5> statt.

Programmierbeispiel

Variation der Konturwinkel bei konstantem Grenzwinkel:

```
#CONTOUR MODE [DEV PATH_DEV=0.50 RELEVANT_PATH=0.1 TRACK_DEV=2 RELE-
VANT_TRACK=0.2]
#CONTOUR MODE [RELEVANT_TRACK=0.3]
P100 = 50
F10000

#CONTOUR MODE [MAX_ANGLE=73]
N10 G01 X-P100 Y0 Z0 C0 A0

$FOR P123 = 0, 90, 7.5
  N2 G01 X0 Y0 Z0 C0 A0 G61
  P1 = COS[P123]*P100
  P2 = SIN[P123]*P100
  NP123 XP1 YP2
  N100 G01 X-P100 Y0 Z0 C0 A0
$ENDFOR

$FOR P123 = 270, 370, 7.5
  N120 G01 X0 Y0 Z0 C0 A0 G61
  P1 = COS[P123]*P100
  P2 = SIN[P123]*P100
  NP123 XP1 YP2
  N400 G01 X-P100 Y0 Z0 C0 A0
$ENDFOR

M30
```


x 10^5
red alpha=43°, blue alpha=73°
4
2
0
-2
-4
command value 2
-4
-2
0
2
4
command value 1
x 10^5


x 10^4
red alpha=43°, blue alpha=73°
3
2
1
0
-1
-2
-3
command value 2
-2
-1
0
1
2
3
command value 1
x 10^4

11.2.6 Anmerkungen

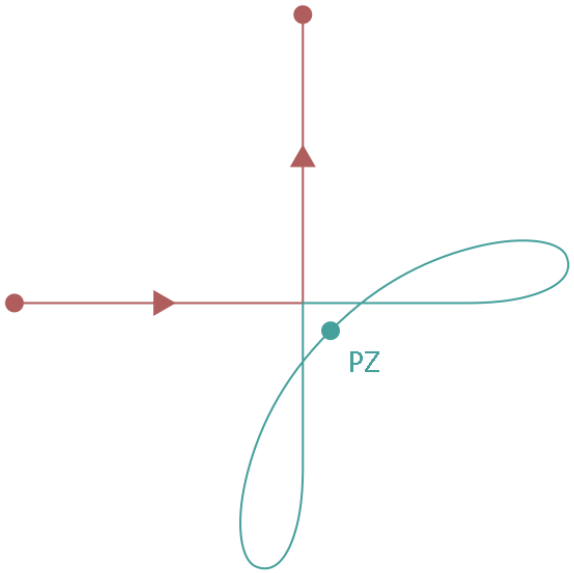
Werden nach Programmierung von G61 bzw. G261 (Überschleifen am Satzende) Achsen abgegeben oder geholt, so kann der Überschleifvorgang nicht ausgeführt werden.

Programmierbeispiel

Überschleifvorgang wird nicht ausgeführt

```
N10 G01  X100  Y0  Z0  F1000
N20 G01  X50   Y50  G61
N30 #PUT AX [Z](Überschleifvorgang wird nicht ausgeführt)
N40 G01  X100
N50 M30
```

Der Kurvenverlauf beim Zwischenpunkt-Überschleifen hängt von der Wahl des Zwischenpunkts ab. Auch folgender Kurvenverlauf ist möglich:



11.3 Weitere Verfahren

11.3.1 Akima-Spline-Interpolation

Hinweis

Die Nutzung dieser Funktionalität erfordert die Lizenzierung des Erweiterungspaketes "Spline". Sie ist nicht im Umfang der Standardlizenz enthalten.

Die Zielpunkte der programmierten Linearsätzen (G00 und G01) sind die Stützstellen, durch die die Akima-Splines gelegt werden.

11.3.1.1 Auswahl des AKIMA-Splinetyps (#SPLINE TYPE AKIMA)

Versionshinweis

Ab Version **V2.11.2010.02** ersetzt der Befehl **#SPLINE TYPE AKIMA** den Befehl **#SET SPLINE-TYPE AKIMA**. Dieser ist aus Kompatibilitätsgründen weiterhin verfügbar, es wird aber empfohlen, diesen in neuen NC-Programmen nicht mehr zu verwenden.

Syntax:

#SPLINE TYPE AKIMA

Falls im System verfügbar, ist der Akima-Spline der Defaultsplinetyp. Es wird jedoch empfohlen, den Splinetyp immer explizit zu programmieren.

Unmittelbar nach einer Anwahl des Splinetyps ist kein tangentialer Übergang aus dem vorhergehenden Bewegungssatz in die Spline-Kurve möglich.

11.3.1.2 Anwahl der Akima-Spline-Interpolation (#SPLINE ON)

Versionshinweis

Ab Version **V2.11.2010.02** ersetzt der Befehl **#SPLINE ON** den Befehl **#SET SPLINE ON**. Dieser ist aus Kompatibilitätsgründen weiterhin verfügbar, es wird aber empfohlen, diesen in neuen NC-Programmen nicht mehr zu verwenden.

Syntax:

#SPLINE ON

Die Anwahl der Akima-Spline-Interpolation erfolgt im zuletzt gewählten Modus. Die Akima-Spline-Kurve beginnt an der zuletzt programmierten Zielposition. Diese Position ist die erste Stützstelle der Akima-Spline-Kurve.

Der Befehl kann im gleichen Satz mit der zweiten Stützstelle der Akima-Spline-Kurve stehen oder im Satz davor.

Die Anwahl der Akima-Spline-Interpolation ist alternativ über G151 möglich.

11.3.1.3 Abwahl der Akima-Spline-Interpolation (#SPLINE OFF)

Versionshinweis

Ab Version **V2.11.2010.02** ersetzt der Befehl **#SPLINE OFF** den Befehl **#SET SPLINE OFF**. Dieser ist aus Kompatibilitätsgründen weiterhin verfügbar, es wird aber empfohlen, diesen in neuen NC-Programmen nicht mehr zu verwenden.

Syntax:

#SPLINE OFF

Die Abwahl der Akima-Spline-Interpolation darf frühestens nach 3 programmierten Stützpunkten erfolgen bzw. nach 2 Stützpunkten, falls alle Tangenten vorgegeben wurden (tangentialer Übergang an beiden Kurvenenden bzw. explizite Programmierung aller Tangenten).

Steht der Befehl in einem Satz zusammen mit einer Positionsangabe, so ist der entsprechende Punkt bereits nicht mehr Teil der Akima-Spline-Kurve.

Die Abwahl der Akima-Spline-Interpolation ist alternativ über G150 möglich.

11.3.1.4 Angabe der Übergangsart (Splinekurve) (#AKIMA TRANS)

Versionshinweis

Ab Version **V2.11.2010.02** ersetzt der Befehl **#AKIMA TRANS [...]** den Befehl **#SET ASPLINE MODE [...]**. Dieser ist aus Kompatibilitätsgründen weiterhin verfügbar, es wird aber empfohlen, diesen in neuen NC-Programmen nicht mehr zu verwenden.

Syntax:

#AKIMA TRANS [**[START**=<*ident*> **END**=<*ident*> **]**]

START=<*ident*>
END=<*ident*>

Tangentiale Übergangsarten zwischen Spline-Kurve und den angrenzenden (Linear- oder Zirkular-)Bewegungssätzen. Die Angabe der Übergangsarten ist optional. Erfolgt **keine** Angabe, so wird als Voreinstellung AUTO verwendet. Zulässige Kennungen:

AUTO	Tangente der Splinekurve am Übergang ergibt sich automatisch
TANGENTIAL	Tangentialer Übergang in den vorhergehenden bzw. nachfolgenden Satz
USER	Tangente der Splinekurve am Übergang wird durch Anwender explizit vorgegeben

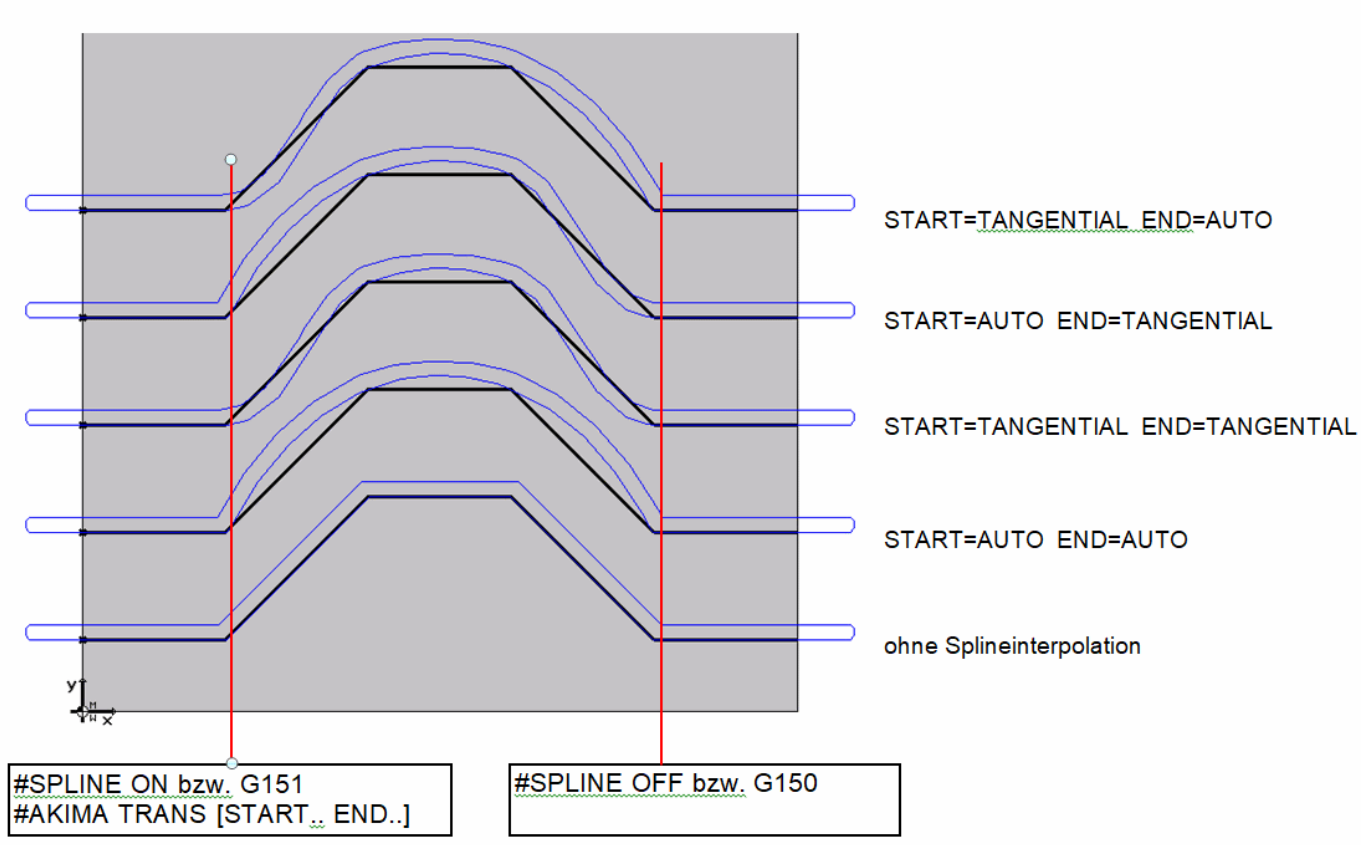


Abb. 94: Beispiele für die Kombination der Übergangsarten 1 und 2

Ist die explizite Tangentenvorgabe für einen bestimmten Übergang angewählt (Übergangsart 3) und wird für eine bestimmte Achse keine Tangente programmiert, so wird die Tangente automatisch bestimmt (Übergangsart 1).

11.3.1.5 Definition der Starttangente (#AKIMA STARTVECTOR)

Versionshinweis

Ab Version **V2.11.2010.02** ersetzt der Befehl **#AKIMA STARTVECTOR ...** den Befehl **#SET ASPLINE STARTTANG ...**. Dieser ist aus Kompatibilitätsgründen weiterhin verfügbar, es wird aber empfohlen, diesen in neuen NC-Programmen nicht mehr zu verwenden.

Syntax:

#AKIMA STARTVECTOR {<*Achsname*>..}

<*Achsname*>.. Komponenten des Tangentenvektors, Realzahl

Definition der Starttangente. Unter Verwendung der Achsbezeichnungen wird ein Vektor angegeben, dessen Richtung der Tangentenrichtung entspricht – der Vektor braucht also nicht normiert zu sein, es müssen jedoch stets alle Komponenten in den Hauptachsen angegeben werden.

Für Mitschleppachsen berechnet sich die Vektorkomponente aus dem Verhältnis von Fahrweg Mitschleppachse zu Bahnfahrweg:

$$\text{Vektorkomponente}_{\text{Mitschlepp}} = S_{\text{Mitschlepp}} / S_{\text{Bahn}} \text{ mit } S_{\text{Bahn}} = \sqrt{S_x^2 + S_y^2 + S_z^2}$$

11.3.1.6 Definition der Zieltangente (#AKIMA ENDVECTOR)

Versionshinweis

Ab Version **V2.11.2010.02** ersetzt der Befehl **#AKIMA ENDVECTOR ...** den Befehl **#SET ASPLINE ZIELTANG ...**. Dieser ist aus Kompatibilitätsgründen weiterhin verfügbar, es wird aber empfohlen, diesen in neuen NC-Programmen nicht mehr zu verwenden.

Syntax:

#AKIMA ENDVECTOR {<*Achsname*>..}

<*Achsname*>.. Komponenten des Tangentenvektors, Realzahl

Definition der Zieltangente; analog zur Definition der Starttangente.

Programmierbeispiel

Definition der Zieltangente

```
N10 G01 X20 Y0 F1000                    (wird durch nachf. G151 zum ers-
ten)                                         (Stützpunkt der Spline-Kurve)
N20 #AKIMA TRANS[START=USER END=AUTO]  (Übergangsart mit Vorgabe der)
                                       (Starttangente + autom. Bestim-
mung)
                                       (der Zieltangente)
N30 #AKIMA STARTVECTOR X1 Y1 Z0        (Vorgabe Starttangente)
N40 G151                               (Anwahl Spline-Interpolation)
N50 G01 X40 Y20
N60 X60
N70 Y0
N80 X80
N90 Y10                               (wird durch nachf. G150 zum letz-
ten)                                       (Stützpunkt der Spline-Kurve)
N100 G150                             (Abwahl der Spline - Interpolation)
N110 X70
N120 M30
```

Das folgende NC-Programm liefert dasselbe Resultat, verwendet jedoch für die An- und Abwahl der Spline-Interpolation die zweite Variante.

```
N10 G01 X20 Y0 F1000
N20 #AKIMA TRANS[START=USER END=AUTO]  (Übergangsart mit Vorgabe der)
                                       (Starttangente + autom. Bestim-
mung)
                                       (der Zieltangente)
N30 #AKIMA STARTVECTOR X1 Y1 Z0        (Vorgabe Starttangente)
N40 G151 G01 X40 Y20                   (Anwahl Spline-Interpolation)
N50 X60
N60 Y0
N70 X80
N80 Y10
N90 G150 X70                         (Abwahl der Spline - Interpolation)
N100 M30
```

Das folgende NC-Programm liefert dasselbe Resultat, verwendet jedoch für die An- und Abwahl der Spline-Interpolation die zweite Variante.

```
N10 G01 X20 Y0 F1000
N20 #AKIMA TRANS[START=USER END=AUTO]    (Übergangsart mit Vorgabe der)
                                         (Starttangente + autom. Bestim-
mung)
                                         (der Zieltangente)
N30 #AKIMA STARTVECTOR X1 Y1 Z0          (Vorgabe Starttangente)
N40 G151 G01 X40 Y20                     (Anwahl Spline-Interpolation)
N50 X60
N60 Y0
N70 X80
N80 Y10
N90 G150 X70                         (Abwahl der Spline - Interpolation)
N100 M30
```

Achtung: Satz Nr. 80 enthält den Endpunkt des Splines!

Das Programm erzeugt folgende Kontur:

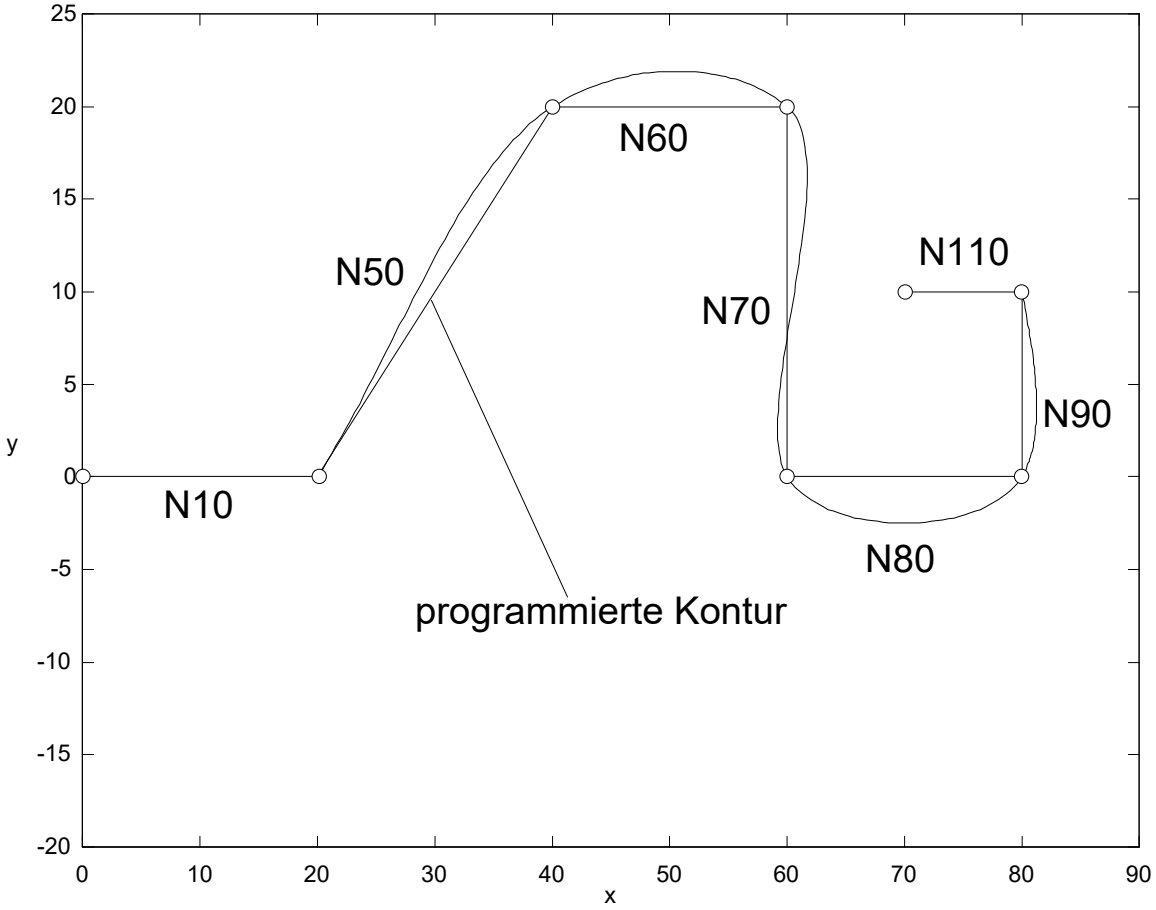


Abb. 95: Bahnverlauf des Beispielprogramms (Nr. bezieht sich auf das 1. Programmbeispiel)

Es ist deutlich zu erkennen, dass der dem Satz N50 entsprechende Kurvenabschnitt an seinem Beginn (der dem Beginn der Spline-Kurve entspricht) die programmierte Steigung 1 aufweist. Die Steigung am Ende des Splines (Ende von Satz N90) ergibt sich automatisch.

Hinweis

Werden Zirkularsätze (G02 bzw. G03) eingefügt, so wird die Spline-Kurve vor dem Zirkularsatz unterbrochen und mit dem Eintreffen des nächsten Linearsatzes automatisch eine neue Spline-Kurve begonnen. Die Übergänge in und vom Zirkularsatz erfolgen tangential.

Die Spline-Kurve wird ebenfalls unterbrochen, wenn ein Linearsatz mit stehenden Hauptachsen und bewegten Mitschleppachsen programmiert wurde. Die Abwahl der Spline-Interpolation erfolgt mit automatischer Tangentenbestimmung am Kurvenende. Die Mitschleppachsen werden so lange linear interpoliert, bis ein Linearsatz mit bewegten Hauptachsen programmiert wird. Ist dies der Fall, erfolgt eine automatische Wiederanwahl der Spline-Interpolation. Der Übergang in die Spline-Kurve erfolgt sowohl für die Hauptachsen als auch für die Mitschleppachsen tangential.

Zwischen den Linearsätzen, welche als Stützpunkte dienen, dürfen auch andere Funktionen (z.B. M-Funktionen) programmiert werden. Die Anzahl dieser Funktionen, die jeweils insgesamt zwischen fünf aufeinanderfolgenden Stützstellen programmiert werden dürfen, ist jedoch konfigurationsabhängig beschränkt.

11.3.2 B-Spline-Interpolation

Hinweis

Die Nutzung dieser Funktionalität erfordert die Lizenzierung des Erweiterungspaketes "Spline". Sie ist nicht im Umfang der Standardlizenz enthalten.

Die Zielpunkte der programmierten Linearsätzen (G00 und G01) sind die Kontrollpunkte, die zur Generierung der B-Spline-Kurve dienen. Es ist zu beachten, dass die B-Spline-Kurve nur am Anfang und am Ende durch die Kontrollpunkte hindurch verläuft.

11.3.2.1 Auswahl des B-Splinetyps (#SPLINE TYPE BSPLINE)

Versionshinweis

Ab Version **V2.11.2010.02** ersetzt der Befehl **#SPLINE TYPE BSPLINE** den Befehl **#SET SPLINETYPE BSPLINE**. Dieser ist aus Kompatibilitätsgründen weiterhin verfügbar, es wird aber empfohlen, diesen in neuen NC-Programmen nicht mehr zu verwenden.

Syntax:

#SPLINE TYPE BSPLINE

Falls im System nur der B-Spline verfügbar ist, so ist dieser automatisch angewählt. Es wird jedoch empfohlen, den Splinetyp immer explizit zu programmieren.

11.3.2.2 Anwahl der B-Spline-Interpolation (#SPLINE ON)

Versionshinweis

Ab Version **V2.11.2010.02** ersetzt der Befehl **#SPLINE ON** den Befehl **#SET SPLINE ON**. Dieser ist aus Kompatibilitätsgründen weiterhin verfügbar, es wird aber empfohlen, diesen in neuen NC-Programmen nicht mehr zu verwenden.

Syntax:

#SPLINE ON

Die B-Spline-Kurve beginnt an der zuletzt programmierten Zielposition. Der Befehl kann im gleichen Satz mit dem zweiten Kontrollpunkt der B-Spline-Kurve stehen oder im Satz davor.

Die Anwahl der B-Spline-Interpolation ist alternativ über G151 möglich.

11.3.2.3 Abwahl der B-Spline-Interpolation (#SPLINE OFF)

Versionshinweis

Ab Version **V2.11.2010.02** ersetzt der Befehl **#SPLINE OFF** den Befehl **#SET SPLINE OFF**. Dieser ist aus Kompatibilitätsgründen weiterhin verfügbar, es wird aber empfohlen, diesen in neuen NC-Programmen nicht mehr zu verwenden.

Syntax:

#SPLINE OFF

Die Abwahl der B-Spline-Interpolation darf frühestens nach 4 programmierten Kontrollpunkten erfolgen.

Steht der Befehl in einem Satz zusammen mit einer Positionsangabe, so ist der entsprechende Punkt bereits nicht mehr Teil der B-Spline-Kurve.

Die Abwahl der B-Spline-Interpolation ist alternativ über G150 möglich.

Programmierbeispiel

Abwahl der Spline-Interpolation

```
N10  #SPLINE TYPE BSPLINE
N20  G01 X0 Y50 Z0 F10000
N30  #SPLINE ON
N40   X3     Y25
N50   X15    Y15
N60   X23    Y12
N70   X25    Y25
N80   X30    Y35
N90   X50    Y37.5
N100  X55    Y32.5
N110  X58    Y12
N120  X70    Y12
N130  X77.5  Y10
N140  X90    Y35
N150  X100   Y37.5
N160 #SPLINE OFF
N170 M30
```

Das Beispielprogramm ergibt folgenden Bahnverlauf:

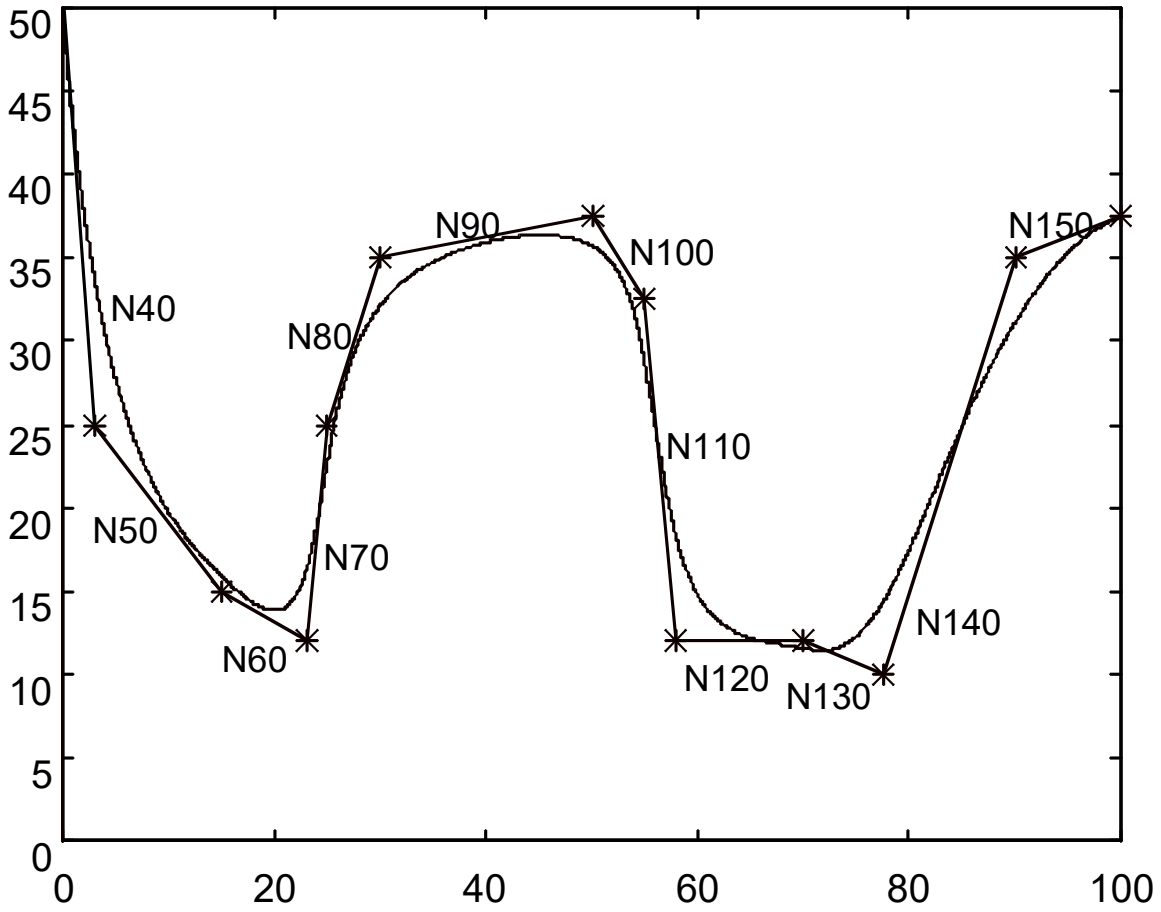


Abb. 96: Bahnverlauf des Beispielprogramms

Das Bild zeigt deutlich die glättende Eigenschaft der B-Spline-Kurve, besonders bei Satz N120/ N130. Außerdem wird deutlich, dass die Kurve nicht durch die Kontrollpunkte verläuft. Das durch Verbinden der Kontrollpunkte entstehende Polygon ermöglicht jedoch eine Abschätzung des Verlaufs der B-Spline-Kurve.

Hinweis

Es ist beim B-Spline nicht möglich, die Tangenten an den Kurvenenden direkt vorzugeben. Da die B-Spline-Kurve an ihren Enden jedoch die Tangente der entsprechenden Verfahrsätze aufweist, können die Tangenten durch passende Programmierung des ersten und letzten Verfahrsatzes vorgegeben werden.

11.3.3 HSC-Programmierung mit OP1 und OP2

Syntax:

HSC [**ON** | **OFF**] **[** [**OPMODE**=..] [**CONTERROR**=..] **]**

ON	HSC-Bearbeitung aktivieren
OFF	HSC-Bearbeitung deaktivieren
OPMODE=..	Betriebsmodus setzen mit: 1: Einfügen von Übergangspolynomen. 2: Erzeugung von interpolierenden Splinekurven.
CONTERROR=..	Festlegung des maximalen Konturfehlers in [mm, inch] mit Werten >= 0.0: Maximale Abweichung "ε". Gilt nur in Verbindung mit aktivem OPMODE 1! Wird der Parameter im OPMODE 2 gesetzt, so wirkt er erst bei Wechsel in OPMODE 1.

Wird bei einer Aktivierung der HSC-Bearbeitung keine Parametrierung vorgenommen, ist folgender Grundzustand gültig:

- OPMODE1
- CONTERROR=0.1 mm

Hinweis

Die Parameter können auch in mehreren Schritten angegeben werden. Das heißt, es ist z.B. möglich zunächst den Betriebsmodus ("OPMODE") sowie die HSC-Bearbeitung ("ON") in einem ersten Befehl zu aktivieren und in einem zweiten Befehl dann den maximalen Konturfehler ("CONTERROR") festzulegen.

Es wird eine Fehlermeldung erzeugt, wenn während aktivierter HSC-Bearbeitung der Betriebsmodus geändert werden soll.

11.3.3.1 Verfügbare Betriebsmodi

OPMODE 1: Einfügen von Übergangspolynomen

Im Betriebsmodus 1 werden bei Bewegungssatzübergängen die Sätze gekürzt und Übergangspolynome eingefügt.

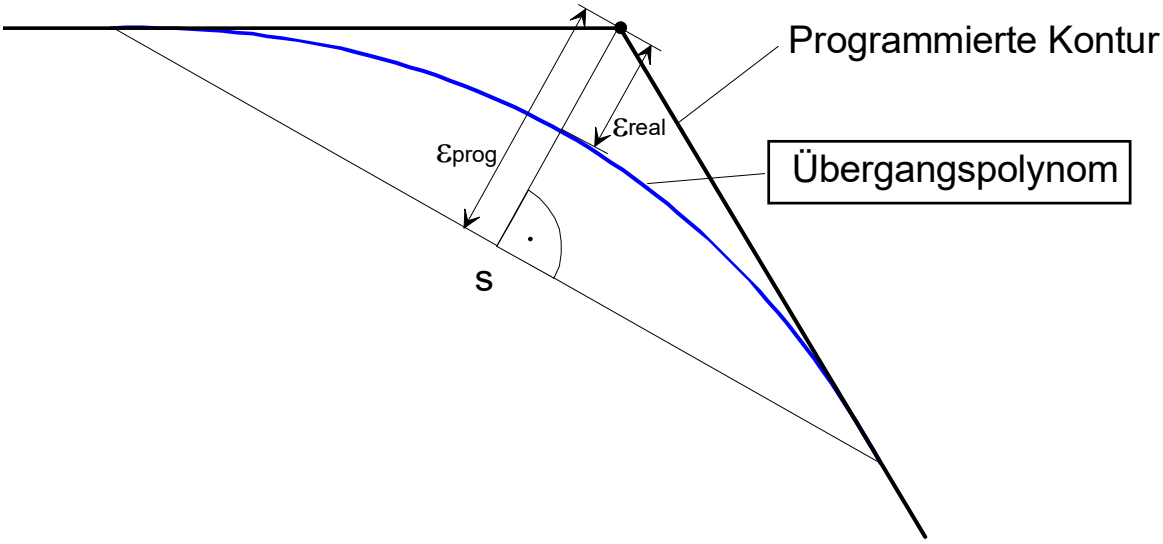


Abb. 97: Einfügen von Übergangspolynomen

Der maximale Konturfehler CONTERROR ("ε") wird zur Generierung des Übergangspolynoms benötigt. Durch das eingefügte Polynom wird der tatsächliche Konturfehler für gewöhnlich kleiner (d.h. es gilt εreal ≤ εprog).

Hinweis

Im aktuellen Satzübergang wird kein Polynom eingefügt, wenn

- die Satzlänge des Anfangssatzes < 1.1µm ist oder
- die Satzlänge des Zielsatzes < 2.2µm ist oder
- der Knickwinkel > 178° ist.

OPMODE 2: Erzeugung von Splinekurven für die HSC-Bearbeitung

Im Betriebsmodus 2 werden Splinekurven durch die vorgegebenen Eckpunkte erzeugt. Am Anfang von prismatischen Abschnitten wird die Splineerzeugung automatisch abgewählt und ein tangentialer Übergang durchgeführt.

Bei den Satzübergängen im prismatischen Bereich, die nach den zusätzlichen Parametern gemäß Kapitel HSC-Programmierung mit OP1 und OP2 [▶ 306] erkannt werden, wird automatisch in den Betriebsmodus 1 (OPMODE 1) gewechselt, d.h. es werden Übergangspolynome eingesetzt. Bei den eingefügten Splinekurven wird keine Konturabweichung zwischen den Stützpunkten überwacht.

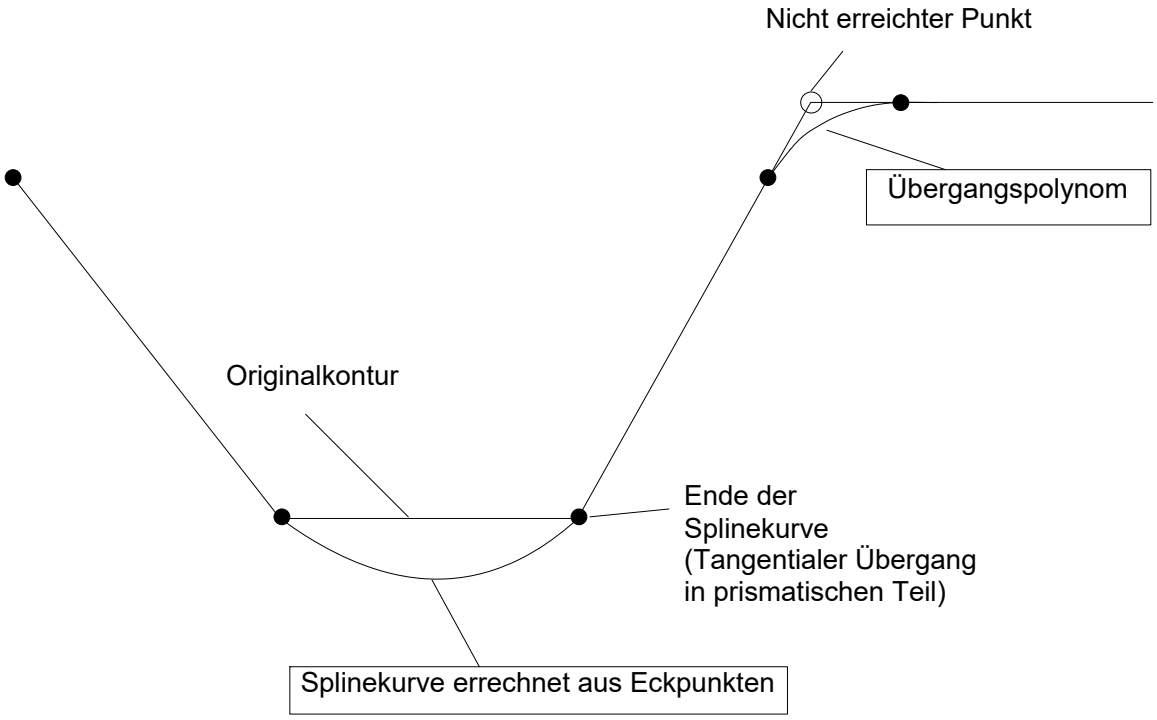


Abb. 98: Erzeugung von Splinekurven für die HSC-Bearbeitung

Programmierbeispiel

OPMODE 1: Einfügen von Übergangspolynomen

```
...
#HSC[OPMODE 1 CONTERROR 0.01]           (Betriebsmodus 1)
                                        (Max. Konturfehler <= 0,01 mm)
...
#HSC ON               (Anwahl HSC-Bearbeitung)
...
#HSC OFF              (Abwahl HSC-Bearbeitung)
...
#HSC ON               (Anwahl HSC-Bearbeitung)
                      (Zuvor angewählte Parameter sind aktiv)
                      (Betriebsmodus 1(Max. Konturfehler <= 0,01 mm)
...
#HSC OFF              (Abwahl HSC-Bearbeitung)
...
#HSC ON [OPMODE 2 CONTERROR 0.002]  (Anwahl Freiformflächenbearbeitung)
                                    (mit Betriebsmodus 2)
                                    (Max. Kont.fehler <= 0,002 mm)
...
#HSC[CONTERROR 0.005]  (Ändern des max. Kont.fehlers <= 0,005 mm)
...
#HSC OFF               (Abwahl HSC-Bearbeitung)
```

Programmierbeispiel

OPMODE 2: Erzeugung von Splinekurven für die HSC-Bearbeitung

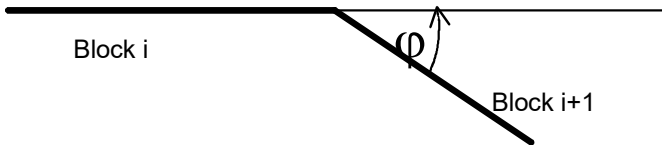
```
...
#HSC[OPMODE 1 CONTERROR 0.01]       (Anwahl Betriebsmodus 1)
                                    (Max. Konturfehler <= 0,01 mm)
#HSC ON                             (Anwahl HSC-Bearbeitung)
...
#HSC[OPMODE 2]                      (Fehlermeldung!)
                                    (Wechsel des Betriebsmodus während)
                                    (aktiver HSC-Bearbeitung)
                                    (nicht erlaubt)
```

11.3.3.2 Zusätzliche Parameter

In der Regel sind nur die bereits beschriebenen Parameter erforderlich. Für den Zugriff auf interne Einstellungen bei Betriebsmodus 2 sind noch folgende zusätzliche Parameter verfügbar:

Syntax:

#HSC [[**COS_PHI_MIN**=..] [**FACT_BLOCK_LEN**=..] **]**

HSC-Parameter	Typ	Gültiger Bereich (Default)	Beschreibung
COS_PHI_MIN=..	Real	-1.0...1.0 (0.7)	 Minimaler Wert für cos(φ). Kleinere Werte führen zur Abwahl der Splinekurvenerzeugung.
FACT_BLOCK_LEN=..	Real	> 0.0 (3.0)	Maximaler Faktor um den die Satzlänge die mittlere Satzlänge* überschreiten darf. Größere Werte führen zur Abwahl der Splinekurvenerzeugung. * Die mittlere Satzlänge ergibt sich aus den Satzlängen der vorhergehenden 5 Bewegungssätzen.

12 Zusatzfunktionen

Eine vollständige Liste der Zusatzfunktionen findet sich in der Befehlsübersicht im Anhang unter Zusatzfunktionen [▶ 884] (#..).

Zusatzfunktionen sind eine eigene Gruppe von NC-Sprachbefehlen. Sie ermöglichen die Programmierung spezifischer Erweiterungen und technologischer Verfahren, die durch die DIN/ISO-Programmierung nicht abgedeckt sind. Die Syntax für Zusatzfunktionen lautet:

#<*string*> <*spezifische Zusatzsyntax*>

#<string>	Klartextbefehl. Zwischen # und *<string>* sind keine Leerzeichen erlaubt.
<spezifische Zusatzsyntax>	Nachfolgende befehlsspezifische Syntaxelemente, die als weitere Strings direkt oder in einer Klammerung programmiert sind.

Achtung

#-Befehle müssen jeweils alleine in einer NC-Zeile stehen. Auf Ausnahmen wird speziell hingewiesen!

Hinweis

Wenn nicht explizit dargestellt, sind Kommas "," und Gleichheitszeichen "=" in der spezifischen Zusatzsyntax **optional** und dienen nur der besseren Lesbarkeit des NC-Programms.

Beispiel:

#STRING [A_WERT 10 B_WERT 20] ←→ #STRING [A_WERT=10, B_WERT=20]

12.1 Wiederherstellen von Achskonfigurationen und Achskopplungen

Komplexe mehrkanalige Maschinen, bei denen während der Bearbeitung Achstauschoperationen und synchrone Achsen (Synchronbetrieb) zum Einsatz kommen, werden im Fehlerfall durch einen NC-Reset abgebrochen. Danach sind die NC-Kanäle grundinitialisiert und die zum Zeitpunkt des Abbruchs gültigen Kanalkonfigurationen sind verloren. Um ein Freifahren der Werkzeugachsen zu ermöglichen, muss somit die zum Zeitpunkt des Abbruchs aktive Konfiguration bestehend aus im Kanal vorhandenen Achsen sowie angewählten Achskopplungen wiederhergestellt werden. Dies wird durch die im folgenden beschriebenen NC-Befehle ermöglicht.

12.1.1 Sichern einer aktuellen Konfiguration (#SAVE CONFIG)

Syntax:

#SAVE CONFIG [[**AX**] [**AXLINK**] **]**

AX	Sichern der momentan im Kanal vorhandenen Achskonfiguration.
AXLINK	Sichern der momentan im Kanal angewählten Achskopplungen. Bereits zuvor gesicherte Kopplungen werden gelöscht, wenn AXLINK programmiert wird, obwohl keine Kopplungen aktiv sind.

Der Befehl #SAVE CONFIG findet üblicherweise im NC-Bearbeitungsprogramm immer nach Aktionen Verwendung, welche eine Änderung der Achskonfiguration bzw. die Aktivierung von Achskopplungen bewirken. Also z.B. nach einem Achstauschbefehl oder nach dem Befehl für die Anwahl eines Synchronbetriebes.

Während der Sicherung werden die Konfigurationsdaten vom Decoder durch den NC-Kanal an den Interpolator übermittelt und von diesem dann wieder direkt an den Decoder zurückgeschickt und dort intern gespeichert. Damit liegen die aktuellen Konfigurationsdaten bzgl. Achsen und Achskopplungen im Decoder synchron zur aktuellen Bearbeitung vor.

Programmierbeispiel

Sichern einer aktuellen Konfiguration

```
%main
N10 X0 Y0 Z0
N15 #AX REQUEST [C,4,5] [B,5,6]
N20 #AX LINK [1, C=X, B=Y]
N25 #AX LINK [2, B=X]
N30 #AX LINK ON [1]
N35 #SAVE CONFIG [AX AXLINK]  ;Sichern der Achskonfiguration X,Y,Z,B,C
                              ;und der aktiven Koppelgruppe 1
N.. X.. Y.. Z..
N200 #AX LINK OFF ALL
N210 #AX RELEASE [C]
N220 #SAVE CONFIG [AX] ;Sichern der neuen Achskonfiguration X,Y,Z,B.
                       ;Die bisherige gesicherte Achskonfiguration
                       ;aus Satz N35 wird überschrieben!
N.. X.. Y.. Z..
N99 M30
```

Hinweis

Eine gesicherte Konfiguration bleibt programmübergreifend gespeichert. Sie kann nur durch nachfolgende #SAVE CONFIG – Befehle überschrieben oder durch #CLEAR CONFIG gelöscht werden!

12.1.2 Laden bzw. Wiederherstellen einer gesicherten Konfiguration (#LOAD CONFIG)

Syntax:

#LOAD CONFIG [[AX] [AXLINK]]

AX	Laden der zuletzt gesicherten Achskonfiguration. Wenn keine gesicherte Achskonfiguration vorhanden ist, erfolgt die Ausgabe einer Fehlermeldung. Im Kanal nicht vorhandene Achsen werden ohne Achsversätze angefordert.
AXLINK	Laden der zuletzt gesicherten aktiven Achskopplungen. Sämtliche Achskopplungen werden zusammengefasst unter der Koppelgruppe 1 wiederhergestellt und aktiviert. Wenn keine gesicherten Achskopplungen vorhanden sind, erfolgt die Ausgabe einer Fehlermeldung.

Der Befehl #LOAD CONFIG wird sinnvollerweise nach einem NC-RESET im s.g. Freifahrprogramm dazu verwendet, die zuletzt gesicherte Konfiguration wiederherzustellen. Es liegt dabei in der Verantwortung des Maschinenbedieners, dass er im Hauptprogramm mit #SAVE CONFIG jeweils die entsprechenden Konfigurationen gesichert hat und er somit auch korrekt wiederaufsetzt. Sind beide Schlüsselworte gesetzt, so wird unabhängig von deren programmierter Reihenfolge zuerst immer die Achskonfiguration vollständig im NC-Kanal wiederhergestellt (ohne "FAST") und dann der zuletzt gesicherte Synchronbetrieb wieder aktiviert.

Programmierbeispiel

Laden bzw. Wiederherstellen einer gesicherten Konfiguration

Starten eines Freifahrprogrammes nach Bearbeitungsabbruch und NC-RESET:

```
%Freifahren
N10 G53
N35 #LOAD CONFIG [AX AXLINK]    ;Wiederherstellen der gesicherten Achs-
                                ;konfiguration und der Achskopplungen
                                ;unter Koppelgruppe 1
N40 #ECS ON           ;Definition eines Effektor-KS zur Durchführung der
                      ;Rückzugsstrategie
N.. X.. Y.. Z..
:
N200 #AX LINK OFF ALL
:
N.. X.. Y.. Z..
N99 M30
```

Hinweis

Bei der Verwendung von #SAVE CONFIG und #LOAD CONFIG im **gleichen** NC-Programm wird zu Beginn von #LOAD CONFIG stets ein implizites FLUSH ausgeführt, um im Kanal die Konsistenz der gesicherten Konfigurationen sicherzustellen.

Programmierbeispiel

```
Nxx #SAVE CONFIG [AX AXLINK]
N..
Nxx #LOAD CONFIG [AX AXLINK]       ;Zuerst implizites FLUSH, dann Wie-
der-
                                   ;herstellen der Achskonfiguration,
                                   ;dann Wiederherstellen der Achskopp-
lungen
N.. X.. Y.. Z..
N99 M30
```

12.1.3 Löschen einer gesicherten Konfiguration (#CLEAR CONFIG)

Syntax:

#CLEAR CONFIG

Mit #CLEAR CONFIG wird die zuletzt gesicherte Konfiguration komplett gelöscht. Ein direkt nachfolgendes #LOAD CONFIG verursacht die Fehlermeldung "Keine wiederherstellbare Konfiguration vorhanden". D.h. es muss zuerst über #SAVE CONFIG wieder eine Konfiguration gesichert werden, bevor diese mit #LOAD CONFIG restauriert werden kann.

Ein #CLEAR CONFIG ist für den Maschinenbediener immer dann hilfreich, wenn er aus Sicherheitsgründen den Zugriff auf eine eventuell falsche gesicherte Konfiguration in anderen NC-Programmen verhindern möchte.

Programmierbeispiel

Löschen einer gesicherten Konfiguration

Freifahrprogramm abarbeiten und dann die gesicherte Konfiguration löschen:

```
%Freifahren
N10 G53
N35 #LOAD CONFIG [AX AXLINK]    ;Wiederherstellen der gesicherten Achs-
                                ;konfiguration und der Achskopplungen
                                ;unter Koppelgruppe 1
N40 #ECS ON           ;Definition eines Effektor-KS zur Durchführung der
                      ;Rückzugsstrategie
N.. X.. Y.. Z..
:
N200 #AX LINK OFF ALL
:
N.. X.. Y.. Z..
N.. #CLEAR CONFIG     ;Nach Beenden der Rückzugsbewegung löschen
                      ;der gesicherten Konfiguration
N99 M30
```

12.2 Achstauschbefehle

In diesem Kapitel werden NC-Befehle zum

- Anfordern von Achsen,
- Abgeben von Achsen sowie zur
- Festlegung einer Achskonfiguration beschrieben.

Bei jedem Programmstart wird die im Kanalparametersatz [1] [▷ 894]-5 angegebene Achskonfiguration wiederhergestellt. Die Achstauschbefehle sind in der aktuell angewählten Achsgruppe wirksam. In einem NC-Satz sind mehrere Anforderungen und/oder Rückgaben von Achsen möglich. Diese Operationen werden quasi-parallel ausgeführt.

Hinweis

Das Tauschen von Achsen, für die der Synchron- oder Handbetrieb aktiv ist, ist nicht zulässig.

Bei aktiver WRK dürfen die ersten beiden Hauptachsen nicht getauscht werden.

Achstauschbefehle müssen jeweils alleine in einer NC-Zeile stehen.

Achstauschbefehle gelten nur für Bahn-bzw.Mitschleppachsen. Spindelachsen werden ignoriert. Für das Abgeben/Anfordern von Spindelachsen stehen spezielle Befehle [▷ 698] zur Verfügung.

Die bisher zur Verfügung stehenden Achstauschbefehle wurden in ihrer Funktionalität und Wirkungsweise ab Version 2.6 überarbeitet und erweitert. Im Kapitel "Standard-Achstauschbefehle" wird die bis Version V2.6 verfügbare Standardsyntax beschrieben. Diese kann auch weiterhin verwendet werden.

Im Kapitel "Erweiterte Achstauschbefehle" wird die neue Syntax beschrieben. Diese ist vollständig abwärtskompatibel zum bisherigen Funktionsumfang, bietet aber durch s.g. Logikschalter und zusätzliche Möglichkeiten der Definition der Achstauschsequenzen eine flexiblere Programmierung.

12.2.1 Standardsyntax

12.2.1.1 Anfordern von Achsen (#CALL AX)

Hinweis

Wird eine Achse angefordert, die bereits in der Achsgruppe des NC-Kanals vorhanden ist, so wird für diese Achse keine Anforderung ausgelöst.

Mit folgendem NC-Befehl werden Achsen von der Achsverwaltung angefordert:

Syntax:

#CALL AX [<*Modus*>] **[**<*Achsname*>,<*Achsnummer*>,<*Achsindex*> {,<*Optionen*>} **]**
{ **[**<*Achsname*>,<*Achsnummer*>,<*Achsindex*> {,<*Optionen*>} **]** }

<*Modus*>	Mit / Ohne Anforderung der Achspositionen vom Interpolator und einer Positionsinitialisierung des NC-Kanals beim Eintauschen von Achsen. ---: Mit Anforderung der Sollwerte vom Interpolator und einer Positionsinitialisierung des NC-Kanals (Standard). FAST: Ohne Anforderung der Sollwerte vom Interpolator. Positionsinitialisierung des NC-Kanals.
<*Achsname*>	Für die Achsbezeichnung sind Strings mit den Anfangsbuchstaben A, B, C, Q, U, V, W, X, Y und Z zulässig. Die mehrfache Zuweisung der gleichen Bezeichnung für verschiedene Achsen (Identifikation durch die logische Achsnummer) bewirkt eine Fehlermeldung und den Abbruch des NC-Programms.
<*Achsnummer*>	Die physikalische Zuordnung der Achsen erfolgt über die logische Achsnummer. Zulässig sind mathematische Ausdrücke. Die logische Achsnummer muss in der Achsverwaltung bekannt sein. Bei Anforderung einer unbekannten logischen Achsnummer oder mehrerer gleicher logischer Achsnummern erfolgt eine Fehlermeldung und der Abbruch des NC-Programms.
<*Achsindex*>	Der Achsindex legt den Platz der Achse innerhalb der Achsgruppe des NC-Kanals fest. Er definiert damit die Haupt- und Mitschleppachsen (siehe folgende Tabelle). Zulässig sind mathematische Ausdrücke, deren Ergebnisse im Wertebereich [0 ... Maximale Achsanzahl-1] liegen. Der Achsindex darf noch nicht mit einer Achse belegt sein. Bei Anforderung einer Achse auf einen Index, der von einer anderen Achse belegt ist, erfolgt eine Fehlermeldung und der Abbruch des NC-Programms. 0: 1. Hauptachse in der Bearbeitungsebene. 1: 2. Hauptachse in der Bearbeitungsebene. 2: 3. Hauptachse, i.a. senkrecht zur Bearbeitungsebene. 3: 1. Mitschleppachse. …n: (n-2). Mitschleppachse.

Achtung

Zur Erleichterung der Programmierung ist es möglich, für **Mitschleppachsen** (nur bei #CALL AX...) die Angabe des Achsindex leer zu lassen. In diesem Fall wird dann automatisch der nächste freie Platz ab Index 3 an diese Mitschleppachse vergeben.

Für **Hauptachsen** muss der Index **immer** explizit angegeben werden.

Zu beachten ist jedoch, dass für verschiedene Funktionalitäten der Index einer Mitschleppachse eine Bedeutung hat. Zum Beispiel müssen bei der kinematischen Transformation (RTCP) alle Trafo.- Achsen lückenlos nach den Hauptachsen angeordnet werden. In diesen Fällen ist es deshalb notwendig, den Achsindex auch für die Mitschleppachsen explizit zu programmieren.

<Optionen>

Versatzmaße werden achsspezifisch gehalten. Mit den folgenden Kennungen kann beim Anfordern von Achsen die Übernahme der verschiedenen Versätze gesteuert werden:

---: keine Übernahme der Versätze (Standard)

ALL: Übernahme aller Versätze *

BPV: Übernahme der Bezugspunktverschiebung

PZV: Übernahme des Platzversatzes

WZV: Übernahme der Werkzeugverschiebung *

NPV: Übernahme der Nullpunktverschiebung

MOFFS: Übernahme des Messoffsets

SOFFS: Übernahme des Sollwert/Handbetriebsoffsets

PSET: Übernahme der Istwertverschiebung

Hinweis

Mit Ausnahme der Werkzeugversätze sind alle anderen Versätze beim Abgeben und Anfordern stets an die logische Achsnummer gebunden (siehe untenstehendes Programmierbeispiel "Zero_offset").

Achtung

* Die Übernahme von Werkzeugverschiebungen ist bei #CALL AX nur bei abgewähltem Werkzeug sinnvoll. Sobald ein Werkzeug im Kanal bereits aktiv ist oder nach dem Achstausch mit #CALL AX angewählt wird, werden die übernommenen Werkzeugversätze durch die Versätze des aktuellen Werkzeuges ersetzt!

Es wird daher empfohlen, #CALL AX bei abgewähltem Werkzeug durchzuführen!

Bei der Werkzeuganwahl ist zu beachten, dass die Versätze immer gemäß der, in den Werkzeugdaten indizierten Reihenfolge in die Achsen eingerechnet werden.

Programmierbeispiel

Anfordern von Achsen

```
%Zero_Offset
N010 X200
N015 G54
N015 V.G.NP_AKT.V.X = 11
N016 X0                          (Endposition Maschinen-X-Achse ist 11)
N020 #PUT AX [X]                 (Achse X mit log. Achsnr. 1 abgeben)
.....
N130 #CALL AX [X1,1,0,NPV]       (Log. Achsnr. 1 unter neuem Name X1 holen)
N140 X1=100                      (Endposition Maschinen-X1-Achse ist 111)
M30
```

Beispiel:

Zuordnung der Achsbezeichnungen, logischen Achsnummern und Achsindices bei Programmstart:

Achsbezeichnung	Logische Achsnummer	Achsindex
X	1	0
Y	2	1
Z	3	2

Programmierbeispiel

%Achstausch1

```
N10 #CALL AX FAST [X1,7,4]     (X1-Achse ohne Sollwerte-Anforderung)
                               (und Ausgabe des Init.-Funktionssatzes)
```

Zuordnung der Achsbezeichnungen, logischen Achsnummern und Achsindices nach der Achsanforderungen:

Achsbezeichnung	Logische Achsnummer	Achsindex
X	**1**	**0**
Y	**2**	**1**
Z	**3**	**2**
		3
X1	**7**	**4**

Fortsetzung des Programmierbeispiels:

```
:
N100 #CALL AX [Y1,8,6] [C,9, ]  ;Y1- und C-Achse anfordern, Achsindex
                                ;von C-Achse wird automatisch bestimmt
```

Zuordnung der Achsbezeichnungen, logischen Achsnummern und Achsindices nach der zweiten Achsanforderung:

Achsbezeichnung	Logische Achsnummer	Achsindex
X	1	0
Y	2	1
Z	3	2
C	9	3
X1	7	4
		5
Y1	8	6

Fortsetzung des Programmierbeispiels:

```
:
N1000 #CALL AX FAST [Z1,13,5,ALL]  (Übernahme sämtlicher Versätze)
N1010 #CALL AX [C1,11,7,NPV MOFFS] (Übernahme der Nullpunktver-)
                                   (schiebung und des Messoffsets)
```

Zuordnung der Achsbezeichnungen, logischen Achsnummern und Achsindices nach der dritten Achsanforderung:

Achsbezeichnung	Logische Achsnummer	Achsindex
X	1	0
Y	2	1
Z	3	2
C	9	3
X1	7	4
Z1	13	5
Y1	8	6
C1	11	7

12.2.1.2 Abgeben von Achsen (#PUT AX, #PUT AX ALL)

Mit diesem NC-Befehl können Achsen der Achsgruppe des NC-Kanals an die Achsverwaltung zurückgegeben werden. Die Rückgabe nicht bzw. nicht mehr vorhandener Achsen ist zulässig und führt zu keiner Fehlermeldung.

Syntax:

#PUT AX [<*Achsname*> {,<*Achsname*> } **]**

<*Achsname*> Für die Achsbezeichnungen sind Strings mit den Anfangsbuchstaben A, B, C, Q, U, V, W, X, Y und Z zulässig.

Mit diesem NC-Befehl können alle in der Achsgruppe des NC-Kanals vorhandenen Achsen an die Achsverwaltung zurückgegeben werden.

Syntax:

#PUT AX ALL

Beispiel

Zuordnung der Achsbezeichnungen, logischen Achsnummern und Achsindices bei Programmstart:

Achsbezeichnung	Logische Achsnummer	Achsindex
X	1	0
Y	2	1
Z	3	2

Programmierbeispiel

Abgeben von Achsen

```
N10 #PUT AX [ X, B]    (X-Achse abgeben; B-Achse nicht vorhanden)
                       (Es wird keine Fehlermeldung ausgegeben)
```

Zuordnung der Achsbezeichnungen, logischen Achsnummern und Achsindices nach der Achsabgabe:

Achsbezeichnung	Logische Achsnummer	Achsindex
		0
Y	2	1
Z	3	2

Fortsetzung des Programmierbeispiels:

```
...
N100 #PUT AX ALL (Alle Achsen dieser Gruppe abgeben.)
```

Zuordnung der Achsbezeichnungen, logischen Achsnummern und Achsindices nach der zweiten Achsabgabe:

Achsbezeichnung	Logische Achsnummer	Achsindex
		0
		1
		2

12.2.1.3 Definition einer Achskonfiguration (#SET AX)

Mit diesem NC-Befehl kann eine neue Achskonfiguration festgelegt werden, welche die vorhandene Achskonfiguration ersetzt. Nur genau die Achsen, die im NC-Befehl programmiert sind, bilden also die neue Achskonfiguration im NC-Kanal.

Syntax:

#SET AX [<*Modus*>] **[**<*Achsname*>,<*Achsnummer*>,<*Achsindex*> {,<*Optionen*>} **]**

{ **[**<*Achsname*>,<*Achsnummer*>,<*Achsindex*>{,<*Optionen*>} **]** }

<*Modus*>	Mit / Ohne Anforderung der Achspositionen vom Interpolator und einer Positionsinitialisierung des NC-Kanals beim Eintauschen von Achsen. ---: Mit Anforderung der Sollwerte vom Interpolator und einer Positionsinitialisierung des NC-Kanals (Standard). FAST: Ohne Anforderung der Sollwerte vom Interpolator. Positionsinitialisierung des NC-Kanals
<*Achsname*>	Für die Achsbezeichnung sind Strings mit den Anfangsbuchstaben A, B, C, Q, U, V, W, X, Y und Z zulässig. Die mehrfache Zuweisung der gleichen Bezeichnung für verschiedene Achsen (Identifikation durch die logische Achsnummer) bewirkt eine Fehlermeldung und den Abbruch des NC-Programms.
<*Achsnummer*>	Die physikalische Zuordnung der Achsen erfolgt über die logische Achsnummer. Zulässig sind mathematische Ausdrücke. Die logische Achsnummer muss in der Achsverwaltung bekannt sein Bei Anforderung einer unbekannten logischen Achsnummer oder mehrerer gleicher logischer Achsnummern erfolgt eine Fehlermeldung und der Abbruch des NC-Programms.
<*Achsindex*>	Der Achsindex legt den Platz der Achse innerhalb der Achsgruppe des NC-Kanals fest. Er definiert damit die Haupt- und Mitschleppachsen (siehe folgende Tabelle). Zulässig sind mathematische Ausdrücke, deren Ergebnisse im Wertebereich [0 ... Maximale Achsanzahl-1] liegen. Der Achsindex darf noch nicht mit einer Achse belegt sein. Bei Anforderung einer Achse auf einen Index, der von einer anderen Achse belegt ist, erfolgt eine Fehlermeldung und der Abbruch des NC-Programms. 0: 1. Hauptachse in der Bearbeitungsebene. 1: 2. Hauptachse in der Bearbeitungsebene. 2: 3. Hauptachse, i.a. senkrecht zur Bearbeitungsebene. 3: 1. Mitschleppachse. …n: (n-2). Mitschleppachse.

<Optionen>

Versatzmaße werden achsspezifisch gehalten. Mit den folgenden Kennungen kann beim Anfordern von Achsen die Übernahme der verschiedenen Versätze gesteuert werden:

---: keine Übernahme der Versätze (Standard)

ALL: Übernahme aller Versätze *

BPV: Übernahme der Bezugspunktverschiebung

PZV: Übernahme des Platzversatzes

WZV: Übernahme der Werkzeugverschiebung *

NPV: Übernahme der Nullpunktverschiebung

MOFFS: Übernahme des Messoffsets

SOFFS: Übernahme des Sollwert/Handbetriebsoffsets

PSET: Übernahme der Istwertverschiebung

Achtung

* Bei **angewähltem Werkzeug** ist bei der Übernahme von Werkzeugverschiebungen bei #SET AX folgendes zu beachten:

- Werden Achsen durch den #SET AX nur vertauscht (interner Achstausch) und ansonsten keine zusätzlichen Achsen abgegeben oder angefordert, so werden alle Versätze (auch die Werkzeugversätze) ebenfalls mitgetauscht und bleiben weiterhin aktiv. Die Angabe von Schlüsselworten zur Übernahme von Versätzen ist ohne Wirkung.
  Wird danach ein neues Werkzeug angewählt, so werden die mitgetauschten Versätze durch die Versätze des neuen Werkzeuges ersetzt.
- Sobald durch den #SET AX eine Achsabgabe bzw. Achsanforderung ausgelöst wird (externer Achstausch), werden die Werkzeugversätze erneut in der, in den Werkzeugdaten indizierten Reihenfolge in die Achsen eingerechnet. Eventuell übernommene Werkzeugversätze werden also durch die Versätze des aktuellen Werkzeuges ersetzt! Sollen die ursprünglichen Werkzeugversätze in den entsprechenden Achsen weiterhin gelten, so muss ein neues Werkzeug angewählt werden, bei dem die Versätze an die neue Achsanordnung angepasst sind.

Es wird daher empfohlen, #SET AX bei abgewähltem Werkzeug durchzuführen und die korrekte Zuordnung von Werkzeugversätzen durch die entsprechende Parametrierung im Datensatz eines neu anzuwählenden Werkzeuges sicherzustellen!

Beispiel:

Index der Werkzeugversätze in den Werkzeugdaten	[0]	[1]	[2]	[3]
Parametrierte Werkzeugversätze z.B. für T1	50	0	70	20
Achskonfiguration bei Programmstart	X	Y	Z	---
Eingerechnete Werkzeugversätze nach Anwahl T1	50	0	70	---
"Interner" #SET AX {Z, X, Y}:	**Z**	**X**	**Y**	---
Werkzeugversätze werden mitgetauscht oder	*70*	*50*	*0*	---
"Externer" #SET AX {Z, X, Y, B}:	**Z**	**X**	**Y**	**B**
Werkzeugversätze werden gemäß Werkzeug T1 neu eingerechnet	*50*	*0*	*70*	*20*

Beispiel:

Zuordnung der Achsbezeichnungen, logischen Achsnummern und Achsindizes bei Programmstart:

Achsbezeichnung	Logische Achsnummer	Achsindex
X	1	0
Y	2	1
Z	3	2

Programmierbeispiel

Setzen der Achskonfiguration:

```
(X-Achse bleibt auf ihrem Platz;)
(Y-Achse wird abgegeben;)
(Z-Achse wird nach Index 4 umsortiert;)
(Y1- und Z1-Achse werden angefordert;)

%ACHSTAUSCH1
N10 #SET AX [X,1,0][Y1,4,2][Z1,5,3][Z,3,4]
```

Zuordnung der Achsbezeichnungen, logischen Achsnummern und Achsindices nach N10:

Achsbezeichnung	Logische Achsnummer	Achsindex
X	1	0
		1
Y1	4	2
Z1	5	3
Z	3	4

12.2.2 Erweiterte Syntax

Versionshinweis

Die Verfügbarkeit dieser Funktionalität ist von der Konfiguration und dem Versionsumfang abhängig

Die erweiterte Syntax ermöglicht die Programmierung der Achstauschsequenzen über Makrodefinitionen (Kapitel Makros [▶ 729]) oder externe Variablen vom Typ String (V.E...). Das ist insbesondere bei mehrkanaligen Maschinen und Anlagen sinnvoll, wenn statische Achsgruppen zwischen den Kanälen getauscht werden. Diese Achsgruppen können dann z.B. in Makros abgelegt und in den Achstauschbefehlen verwendet werden.

Weiterhin ermöglicht die erweiterte Syntax durch entsprechendes Setzen von Logikschaltern die interne Behandlung von Konflikten ohne Ausgabe von Fehlermeldungen bzw. Warnings. Die Logikschalter sind zusätzlich optional im Befehl programmierbar. Falls keine zusätzliche Logikauswertung programmiert wird, gilt das Standardverhalten wie bisher, d.h. bei Plausibilitätskonflikten wird die Auswertung der Achstauschsequenzen mit Fehlermeldungen abgebrochen.

Die Logikauswertung ist für alle Achstauschbefehle gleich und prüft sowohl die Plausibilität **innerhalb der programmierten Achstauschsequenz** als auch die Plausibilität **zu bereits im NC-Kanal vorhandenen Achsen.**

Insbesondere sind die Logikschalter bei der Nutzung von Achstauschsequenzen sinnvoll, die über Makros oder Stringvariablen definiert sind, da hier eventuell auftretende Überschneidungen und Doppelprogrammierungen intern bereinigt werden können.

12.2.2.1 Anfordern von Achsen (#AX REQUEST)

Mit folgendem NC-Befehl werden Achsen von der Achsverwaltung angefordert.

Syntax:

#AX REQUEST [**NAM**, **NBR**, **IDX**] **[**<*Achstauschsequenz*> {,<*Optionen*>} **]**

{ **[**<*Achstauschsequenz*> {,<*Optionen*>} **]** }

NAM, NBR, IDX	Logikschalter zur Behandlung von Konflikten: NAM: Behandlung redundanter Achsnamen NBR: Behandlung redundanter Achsnummern IDX: Behandlung redundanter Achsindizes

Hinweis

Die Logikschalter können einzeln oder in Kombination programmiert werden!

<*Achstauschsequenz*>
bestehend aus:

<*Achsname*>	Für die Achsbezeichnung sind Strings mit den Anfangsbuchstaben A, B, C, Q, U, V, W, X, Y und Z zulässig.

Hinweis

Bei Konflikten innerhalb der programmierten Achstauschsequenz:

Redundante Achsnamen → FEHLER, Programmabbruch.

Bei Konflikten zu bereits im NC-Kanal vorhandenen Achsen:

Achsnamen gleich, Achsnummern verschieden → FEHLER, Programmabbruch

Bei gesetztem Logikschalter **NAM** wird Konflikt wie folgt bereinigt:

Die Achse bekommt den Standardachsnamen aus ihrer Achsparameterliste P-AXIS-00297. Die eindeutige Vergabe der Standardachsnamen in den Listen ist durch den Anwender sicherzustellen.

<*Achsnummer*>	Die physikalische Zuordnung der Achsen erfolgt über die logische Achsnummer. Zulässig sind mathematische Ausdrücke. Die logische Achsnummer muss in der Achsverwaltung bekannt sein.

Hinweis

Bei Konflikten innerhalb der programmierten Achstauschsequenz:

Redundante Achsnummern → FEHLER, Programmabbruch.

Bei Konflikten zu bereits im NC-Kanal vorhandenen Achsen:

Achsnummer im Kanal bereits vorhanden → WARNUNG

Bei gesetztem Logikschalter **NBR** wird Konflikt wie folgt bereinigt:

Die Achsanforderung wird ignoriert, d.h. sie wird nicht ausgeführt!

<Achsindex>

Der Achsindex legt den Platz der Achse innerhalb der Achsgruppe des NC-Kanals fest. Er definiert damit die Haupt- und Mitschleppachsen (siehe folgende Tabelle). Zulässig sind mathematische Ausdrücke, deren Ergebnisse im Wertebereich [0 ... Maximale Achsanzahl-1] liegen. Der Achsindex darf noch nicht mit einer Achse belegt sein.

0: 1. Hauptachse in der Bearbeitungsebene.

1: 2. Hauptachse in der Bearbeitungsebene.

2: 3. Hauptachse, i.a. senkrecht zur Bearbeitungsebene.

3: 1. Mitschleppachse.

…n: (n-2). Mitschleppachse.

Achtung

Zur Erleichterung der Programmierung ist es möglich, für **Mitschleppachsen** die Angabe des Achsindex leer zu lassen. In diesem Fall wird dann automatisch der nächste freie Platz ab Index 3 an diese Mitschleppachse vergeben.

Für **Hauptachsen** muss der Index **immer** explizit angegeben werden.

Zu beachten ist jedoch, dass für verschiedene Funktionalitäten der Index einer Mitschleppachse eine Bedeutung hat. Zum Beispiel müssen bei der kinematischen Transformation (RTCP) alle Trafo-Achsen lückenlos nach den Hauptachsen angeordnet werden. In diesen Fällen ist es deshalb notwendig, den Achsindex auch für die Mitschleppachsen explizit zu programmieren.

Hinweis

Bei Konflikten innerhalb der programmierten Achstauschsequenz:

Redundante Achsnummern → FEHLER, Programmabbruch.

Bei Konflikten zu bereits im NC-Kanal vorhandenen Achsen:

Achsindex im Kanal bereits belegt, Achsnamen verschieden → FEHLER, Programmabbruch.

Bei gesetztem Logikschalter **IDX** wird Konflikt wie folgt bereinigt:

Für die Achse wird automatisch der nächste freie Index in der Achskonfiguration des NC-Kanals bestimmt.

<Optionen>

Versatzmaße werden achsspezifisch gehalten. Mit den folgenden Kennungen kann beim Anfordern von Achsen die Übernahme der verschiedenen Versätze gesteuert werden:

---: keine Übernahme der Versätze (Standard)

ALL: Übernahme aller Versätze *

BPV: Übernahme der Bezugspunktverschiebung

PZV: Übernahme des Platzversatzes

WZV: Übernahme der Werkzeugverschiebung *

NPV: Übernahme der Nullpunktverschiebung

MOFFS: Übernahme des Messoffsets

SOFFS: Übernahme des Sollwert/Handbetriebsoffsets

PSET: Übernahme der Istwertverschiebung

Hinweis

Mit Ausnahme der Werkzeugversätze sind alle anderen Versätze beim Abgeben und Anfordern stets an die logische Achsnummer gebunden.

Achtung

* Die Übernahme von Werkzeugverschiebungen ist bei #AX REQUEST nur bei abgewähltem Werkzeug sinnvoll. Sobald ein Werkzeug im Kanal bereits aktiv ist oder nach dem Achstausch mit #AX REQUEST angewählt wird, werden die übernommenen Werkzeugversätze durch die Versätze des aktuellen Werkzeuges ersetzt!

Es wird daher empfohlen, #AX REQUEST bei abgewähltem Werkzeug durchzuführen!

Bei der Werkzeuganwahl ist zu beachten, dass die Versätze immer gemäß der, in den Werkzeugdaten indizierten Reihenfolge in die Achsen eingerechnet werden.

Beispiel

Anwenden der Standardfunktionalität:

Zuordnung der Achsbezeichnungen, logischen Achsnummern und Achsindizes bei Programmstart:

Achsbezeichnung	Logische Achsnummer	Achsindex
X	1	0
Y	2	1
Z	3	2

Programmierbeispiel

Anfordern von Achsen

```
N10 #AX REQUEST [X1,7,4]      ;X1-Achse anfordern
```

Zuordnung der Achsbezeichnungen, logischen Achsnummern und Achsindizes nach der Achsanforderung:

Achsbezeichnung	Logische Achsnummer	Achsindex
X	**1**	**0**
Y	**2**	**1**
Z	**3**	**2**
		3
X1	**7**	4

Fortsetzung des Programmierbeispiels:

```
N100 #AX REQUEST  [Y1,8,6] [C,9, ] ; Y1- und C-Achse anfordern,
                                    ; C-Achse wird automatisch auf
                                    ; Index 3 gelegt
```

Zuordnung der Achsbezeichnungen, logischen Achsnummern und Achsindizes nach der zweiten Achsanforderung:

Achsbezeichnung	Logische Achsnummer	Achsindex
X	1	0
Y	2	1
Z	3	2
C	9	3
X1	7	4
		5
Y1	8	6

Fortsetzung des Programmierbeispiels:

```
N1000 #AX REQUEST FAST [Z1,13,5,ALL]  ;Übernahme sämtlicher Versätze
N1010 #AX REQUEST [C1,11,7,NPV MOFFS] ;Übernahme Nullpunktver-
                                      ;schiebung und Messoffset
```

Zuordnung der Achsbezeichnungen, logischen Achsnummern und Achsindizes nach der dritten Achsanforderung:

Achsbezeichnung	Logische Achsnummer	Achsindex
X	1	0
Y	2	1
Z	3	2
C	9	3
X1	7	4
Z1	13	5
Y1	8	6
C1	11	7

Beispiel 2:

Nutzen der zusätzlichen Syntax (Logikschalter):

Zuordnung der Achsbezeichnungen, logischen Achsnummern und Achsindizes bei Programmstart im Kanal 1:

Achsbezeichnung	Logische Achsnummer	Achsindex
X	1	0
Y	2	1
Z	3	2

Zuordnung der Achsbezeichnungen, logischen Achsnummern und Achsindizes bei Programmstart im Kanal 2:

Achsbezeichnung	Logische Achsnummer	Achsindex
X	7	0
Y	8	1
Z	9	2

Programmierbeispiel

In Kanal 1: Anfordern der X- und Y-Achse aus Kanal 2 (Logische Achsnummern 7 und 8).

```
N10 #AX REQUEST NAM [X,7,3] [Y,8,4]   ;X/Y-Achsen anfordern
```

Durch den Logikschalter NAM werden die zusätzlichen neuen Achsen X und Y im Kanal 1 mit ihren Standardachsnamen aus den Achslisten (z.B. X2 und Y2) belegt.

Zuordnung der Achsbezeichnungen, logischen Achsnummern und Achsindizes nach den Achsanforderungen:

Achsbezeichnung	Logische Achsnummer	Achsindex
X	**1**	**0**
Y	**2**	**1**
Z	**3**	**2**
X2	**7**	**3**
Y2	**8**	**4**

Fortsetzung des Programmierbeispiels:

In Kanal 1: Anfordern der Z-Achse aus Kanal 2 (Logische Achsnummer 9).

```
N100 #AX REQUEST NAM IDX [Z,9,2]  ;Z-Achse anfordern
```

Durch die Logikschalter NAM und IDX wird im Kanal 1 die zusätzliche neue Z-Achse mit ihrem Standardachsnamen aus der Achsliste (z.B. Z2) auf einen noch freien Index (z.B. Index 5) übernommen.

Zuordnung der Achsbezeichnungen, logischen Achsnummern und Achsindizes nach der zweiten Achsanforderung:

Achsbezeichnung	Logische Achsnummer	Achsindex
X	**1**	**0**
Y	**2**	**1**
Z	**3**	**2**
X2	**7**	**3**
Y2	**8**	**4**
Z2	**9**	**5**

Beispiel 3:

Nutzen der zusätzlichen Syntax (Logikschalter und Achstauschsequenz als String):

Zuordnung der Achsbezeichnungen, logischen Achsnummern und Achsindizes bei Programmstart im Kanal 1:

Achsbezeichnung	Logische Achsnummer	Achsindex
X	1	0
Y	2	1
Z	3	2

Zuordnung der Achsbezeichnungen, logischen Achsnummern und Achsindizes bei Programmstart im Kanal 2:

Achsbezeichnung	Logische Achsnummer	Achsindex
X_1	7	0
Y	8	1
Z	9	2
A	10	3
B	11	4

Programmierbeispiel

In Kanal 1: Anfordern der X_1/ Y und B-Achse aus Kanal 2 (Logische Achsnummern 7, 8 und 11). Die Achstauschsequenz ist in einem Makro abgelegt.

```
N05 "ACHSEN_KANAL2" = "[X_1,7,0] [Y,8,1] [B,11,2]"
:
N10 #AX REQUEST NAM IDX "ACHSEN_KANAL2" ;Achsen anfordern
```

Durch die Logikschalter NAM und IDX werden im Kanal 1 die zusätzlichen neuen Achsen korrekt übernommen.

Zuordnung der Achsbezeichnungen, logischen Achsnummern und Achsindizes nach den Achsanforderungen:

Achsbezeichnung	Logische Achsnummer	Achsindex
X	**1**	**0**
Y	**2**	**1**
Z	**3**	**2**
X_1	**7**	**3**
Y2	**8**	**4**
B	**11**	**5**

12.2.2.2 Abgeben von Achsen (#AX RELEASE, #AX RELEASE ALL)

Mit diesen NC-Befehlen können Achsen der Achsgruppe des NC-Kanals an die Achsverwaltung zurückgegeben werden. Die Rückgabe nicht bzw. nicht mehr vorhandener Achsen ist zulässig und führt zu keiner Fehlermeldung.

Syntax:

#AX RELEASE [<*Achsname*> {,<*Achsname*> } **]**

<*Achsname*> Achsbezeichnungen der momentan im NC-Kanal vorhandenen Achsen.

Mit dem Logikschalter NBR kann auf die Auswertung von logischen Achsnummern anstatt von Achsnamen umgeschaltet werden (z.B. wenn die Achsnamen zum Zeitpunkt der Abgabe nicht bekannt sind).

Syntax:

#AX RELEASE [**NBR**] **[** <*ax_nr*> {,<*ax_nr*> } **]**

<*ax_nr*> Logische Achsnummer der Achse.

Mit diesem NC-Befehl können alle in der Achsgruppe des NC-Kanals vorhandenen Achsen an die Achsverwaltung zurückgegeben werden.

Syntax:

#AX RELEASE ALL

Beispiel

Zuordnung der Achsbezeichnungen, logischen Achsnummern und Achsindizes bei Programmstart:

Achsbezeichnung	Logische Achsnummer	Achsindex
X	1	0
Y	2	1
Z	3	2
A	4	3
B	5	4

Programmierbeispiel

Abgeben von Achsen

```
N10 #AX RELEASE[X, A]      ; X/A-Achsen abgeben
```

Zuordnung der Achsbezeichnungen, logischen Achsnummern und Achsindizes nach der Achsabgabe:

Achsbezeichnung	Logische Achsnummer	Achsindex
Y	2	1
Z	3	2
B	5	4

Fortsetzung des Programmierbeispiels:

```
...
N100 #AX RELEASE NBR[2]      ; Y-Achse abgeben
```

Zuordnung der Achsbezeichnungen, logischen Achsnummern und Achsindizes nach der zweiten Achsabgabe:

Achsbezeichnung	Logische Achsnummer	Achsindex
Z	3	2
B	5	4

Fortsetzung des Programmierbeispiels:

```
...
N100 #AX RELEASE ALL      ;Alle vorhandenen Achsen dieses
                          ;Kanals abgeben
```

Zuordnung der Achsbezeichnungen, logischen Achsnummern und Achsindizes nach der dritten Achsabgabe:

Achsbezeichnung	Logische Achsnummer	Achsindex

12.2.2.3 Definition einer Achskonfiguration (#AX DEF)

Mit diesem NC-Befehl kann eine neue Achskonfiguration festgelegt werden, welche die vorhandene Achskonfiguration ersetzt. **Nur genau die Achsen, die im NC-Befehl programmiert sind, bilden also die neue Achskonfiguration im NC-Kanal.**

Syntax:

#AX DEF [**NAM**, **NBR**, **IDX**] **[**<*Achstauschsequenz*> {,<*Optionen*>} **]**
{ **[**<*Achstauschsequenz*> {,<*Optionen*>} **]** }

NAM, NBR, IDX — Logikschalter zur Behandlung von Konflikten:
NAM: Behandlung redundanter Achsnamen
NBR: Behandlung redundanter Achsnummern
IDX: Behandlung redundanter Achsindizes

Hinweis

Die Logikschalter können einzeln oder in Kombination programmiert werden!

<*Achstauschsequenz*>
bestehend aus:

<*Achsname*> — Für die Achsbezeichnung sind Strings mit den Anfangsbuchstaben A, B, C, Q, U, V, W, X, Y und Z zulässig.

Hinweis

Bei Konflikten innerhalb der programmierten Achstauschsequenz:

Redundante Achsnamen → FEHLER, Programmabbruch.

Bei Konflikten zu bereits im NC-Kanal vorhandenen Achsen:

Achsnamen gleich, Achsnummern verschieden → FEHLER, Programmabbruch

Bei gesetztem Logikschalter **NAM** wird Konflikt wie folgt bereinigt:

Die Achse bekommt den Standardachsnamen aus ihrer Achsparameterliste P-AXIS-00297. Die eindeutige Vergabe der Standardachsnamen in den Listen ist durch den Anwender sicherzustellen.

<*Achsnummer*> — Die physikalische Zuordnung der Achsen erfolgt über die logische Achsnummer. Zulässig sind mathematische Ausdrücke. Die logische Achsnummer muss in der Achsverwaltung bekannt sein.

Hinweis

Bei Konflikten innerhalb der programmierten Achstauschsequenz:

Redundante Achsnummern → FEHLER, Programmabbruch.

Bei Konflikten zu bereits im NC-Kanal vorhandenen Achsen:

Achsnummer im Kanal bereits vorhanden → WARNING

Bei gesetztem Logikschalter **NBR** wird Konflikt wie folgt bereinigt:

Die Achsanforderung wird ignoriert, d.h. sie wird nicht ausgeführt!

<Achsindex>

Der Achsindex legt den Platz der Achse innerhalb der Achsgruppe des NC-Kanals fest. Er definiert damit die Haupt- und Mitschleppachsen (siehe folgende Tabelle). Zulässig sind mathematische Ausdrücke, deren Ergebnisse im Wertebereich [0 ... Maximale Achsanzahl-1] liegen. Der Achsindex darf noch nicht mit einer Achse belegt sein.

0: 1. Hauptachse in der Bearbeitungsebene.

1: 2. Hauptachse in der Bearbeitungsebene.

2: 3. Hauptachse, i.a. senkrecht zur Bearbeitungsebene.

3: 1. Mitschleppachse.

…n: (n-2). Mitschleppachse.

Achtung

Zur Erleichterung der Programmierung ist es möglich, für Mitschleppachsen die Angabe des Achsindex leer zu lassen. In diesem Fall wird dann automatisch der nächste freie Platz ab Index 3 an diese Mitschleppachse vergeben. Zu beachten ist jedoch, dass für verschiedene Funktionalitäten der Index einer Mitschleppachse eine Bedeutung hat. Zum Beispiel müssen bei der kinematischen Transformation (RTCP) alle Trafo-Achsen lückenlos nach den Hauptachsen angeordnet werden. In diesen Fällen ist es deshalb notwendig, den Achsindex auch für die Mitschleppachsen explizit zu programmieren.

Hinweis

Bei Konflikten innerhalb der programmierten Achstauschsequenz:

Redundante Achsindizes → FEHLER, Programmabbruch.

Bei Konflikten zu bereits im NC-Kanal vorhandenen Achsen:

Achsindex im Kanal bereits belegt, Achsnamen verschieden → FEHLER, Programmabbruch.

Bei gesetztem Logikschalter **IDX** wird Konflikt wie folgt bereinigt:

Für die Achse wird automatisch der nächste freie Index in der Achskonfiguration des NC-Kanals bestimmt.

<Optionen>

Versatzmaße werden achsspezifisch gehalten. Mit den folgenden Kennungen kann beim Anfordern von Achsen die Übernahme der verschiedenen Versätze gesteuert werden:

---: keine Übernahme der Versätze (Standard)

ALL: Übernahme aller Versätze *

BPV: Übernahme der Bezugspunktverschiebung

PZV: Übernahme des Platzversatzes

WZV: Übernahme der Werkzeugverschiebung *

NPV: Übernahme der Nullpunktverschiebung

MOFFS: Übernahme des Messoffsets

SOFFS: Übernahme des Sollwert/Handbetriebsoffsets

PSET: Übernahme der Istwertverschiebung

Achtung

* Bei **angewähltem Werkzeug ist** bei der Übernahme von Werkzeugverschiebungen bei #AX DEF folgendes zu beachten:

- Werden Achsen durch den #AX DEF nur vertauscht (interner Achstausch) und ansonsten keine zusätzlichen Achsen angegeben oder angefordert, so werden alle Versätze (auch die Werkzeugversätze) ebenfalls mitgetauscht und bleiben weiterhin aktiv. Die Angabe von Schlüsselworten zur Übernahme von Versätzen ist ohne Wirkung.
  Wird danach ein neues Werkzeug angewählt, so werden die mitgetauschten Versätze durch die Versätze des neuen Werkzeuges ersetzt.
- Sobald durch den #AX DEF eine Achsabgabe bzw. Achsanforderung ausgelöst wird (externer Achstausch), werden die Werkzeugversätze erneut in der in den Werkzeugdaten indizierten Reihenfolge in die Achsen eingerechnet. Eventuell übernommene Werkzeugversätze werden also durch die Versätze des aktuellen Werkzeuges ersetzt! Sollen die ursprünglichen Werkzeugversätze in den entsprechenden Achsen weiterhin gelten, so muss ein neues Werkzeug angewählt werden, bei dem die Versätze an die neue Achsanordnung angepasst sind.

Es wird daher empfohlen, #AX DEF bei abgewähltem Werkzeug durchzuführen und die korrekte Zuordnung von Werkzeugversätzen durch die entsprechende Parametrierung im Datensatz eines neu anzuwählenden Werkzeuges sicherzustellen!

Beispiel

Index der Werkzeugversätze in den Werkzeugdaten	[0]	[1]	[2]	[3]
Parametrierte Werkzeugversätze z.B. für T1	50	0	70	20
Achskonfiguration bei Programmstart	X	Y	Z	---
Eingerechnete Werkzeugversätze nach Anwahl T1	50	0	70	---
"Interner" #AX DEF {Z, X, Y}:	**Z**	**X**	**Y**	---
Werkzeugversätze werden mitgetauscht oder	*70*	*50*	*0*	---
"Externer" #AX DEF {Z, X, Y, B}:	**Z**	**X**	**Y**	**B**
Werkzeugversätze werden wieder gemäß T1 neu eingerechnet	*50*	*0*	*70*	*20*

12.2.2.4 Laden der Defaultachskonfiguration (#AX DEF DEFAULT)

Durch die Angabe von DEFAULT kann die Grundkonfiguration entsprechend der Kanalparameterliste [1] [▶ 894]-5 hergestellt werden. Auch die Kombination mit Logikschaltern und zusätzlichen Achsanforderungen ist möglich:

Syntax:

#AX DEF DEFAULT

oder

#AX DEF DEFAULT [**NAM**, **NBR**, **IDX**] { **[**<*Achstauschsequenz*> {,<*Optionen*>} **]** }

Beispiel

Zuordnung der Achsbezeichnungen, logischen Achsnummern und Achsindizes bei Programmstart:

Ausgangslage für das nachfolgende NC-Programm ist die in der Tabelle aufgeführte Standardachskonfiguration.

Achsbezeichnung	Logische Achsnummer	Achsindex
X	1	0
Y	2	1
Z	3	2

```
;Setzen der Achskonfiguration
;X-Achse bleibt auf ihrem Platz,
;Y-Achse wird abgegeben, weil diese Achse nicht in der neuen
;                  Definition enthalten ist
;Z-Achse wird nach Index 4 umsortiert,
;Y1- und Z1-Achse werden angefordert

%ACHSTAUSCH
N10 #AX DEF [X,1,0][Y1,4,2][Z1,5,3][Z,3,4]
:
```

Zuordnung der Achsbezeichnungen, logischen Achsnummern und Achsindizes nach N10:

Achsbezeichnung	Logische Achsnummer	Achsindex
X	1	0
		1
Y1	4	2
Z1	5	3
Z	3	4

Wiederherstellen der Standardachskonfiguration:

```
%ACHSTAUSCH
N10 #AX DEF DEFAULT
:
```

Zuordnung der Achsbezeichnungen, logischen Achsnummern und Achsindizes nach N10:

Achsbezeichnung	Logische Achsnummer	Achsindex
X	1	0
Y	2	1
Z	3	2

12.3 Verweilzeit

Syntax:

#TIME <*time*> nicht modal

Siehe hierzu Kapitel Verweilzeit (G04), (#TIME) [▶ 89].

12.4 NC-Kanal leeren (#FLUSH, #FLUSH CONTINUE, #FLUSH WAIT)

Die einzelnen NC-Sätze eines NC-Programmes werden durch das Interpretermodul fortlaufend eingelesen, in eine interne Darstellung umgesetzt und zur weiteren Verarbeitung an den NC-Kanal ausgegeben (Programmdecodierung). Im NC-Kanal werden die Daten dann von weiteren Modulen (Werkzeugradiuskorrektur, Bahnvorbereitung, etc.) bearbeitet und bis zum Interpolator durchgeschleust.

Durch diese Verarbeitung entsteht eine Pufferwirkung des NC-Kanals, die dazu führt, dass die Programmdecodierung der tatsächlichen bzw. aktuellen Ausführung der Befehle im Interpolator im Allgemeinen vorauseilt.

Diese Pufferwirkung des NC-Kanals kann nun zu einer verzögerten Versorgung des Interpolators mit Verfahrsätzen und somit zu einem Stop von Achsbewegungen führen.

Mit Hilfe der Befehle **#FLUSH** und **#FLUSH CONTINUE** wird diese Pufferwirkung des NC-Kanals aufgehoben. Alle gepufferten NC-Sätze werden sofort verarbeitet und entsprechend ihrer programmierten Reihenfolge ausgeführt.

Die Befehle sind insbesondere immer dann hilfreich, wenn es darum geht, große nicht bewegungsrelevante NC-Programmbereiche zwischen zwei Verfahrsätzen zeitlich zu überbrücken, um überflüssiges Stoppen von Achsbewegungen zu vermeiden.

Achtung

Die Befehle zum Leeren des Kanals (#FLUSH, #FLUSH CONTINUE, #FLUSH WAIT) können **nicht** bei aktiver WRK, Polynomüberschleifen sowie aktivem HSC-Modus verwendet werden.

Versionshinweis

Ab der Version V2.10.1503.08 ist es zulässig, bei aktivem Polynomüberschleifen die Befehle zum Leeren des Kanals zu verwenden. In diesem Fall wird dann im Satz vor und nach einem Befehl zum Leeren des Kanals das Überschleifen unterdrückt.

Syntax:

#FLUSH

Die Anweisung **#FLUSH** bewirkt, dass alle bis zu diesem Befehl an den NC-Kanal ausgegebenen NC-Sätze (Steuerkommandos, Verfahrbewegungen) an den Interpolator sofort weitergereicht werden. Die Programmdecodierung läuft hierbei ohne Unterbrechung weiter.

Am Ende der letzten Verfahrbewegung vor dem **#FLUSH** wird im Interpolator **generell** kurz angehalten, auch wenn eventuell der nächste Verfahrsatz durch die weiterlaufende Programmdecodierung bereits vorliegt.

Programmierbeispiel

NC-Kanal leeren

Zwischen den Verfahrsätzen N10 und N210 ist eine WHILE-Schleife mit vielen nicht bewegungsrelevanten NC-Sätzen programmiert (Parameterrechnungen, Variablenzugriffe etc.). Dadurch wird der Verfahrsatz N10 im Kanal zurückgehalten und die Bereitstellung des Verfahrsatzes N210 verzögert. Mit **#FLUSH** wird der NC-Kanal gezwungen, alle Verfahrsätze vor der WHILE-Schleife an den Interpolator weiterzureichen. Dieser kann dann schon mit der Verfahrbewegung bis einschließlich N10 starten.

```
N05  G01 F1000 G90 X150
N10  X200
N20  #FLUSH
N30  $WHILE
N40    P1 = [P2 * P3] + V.E...
N50    V.L....
...
N200 $ENDWHILE
N210 X250
...
M30
```

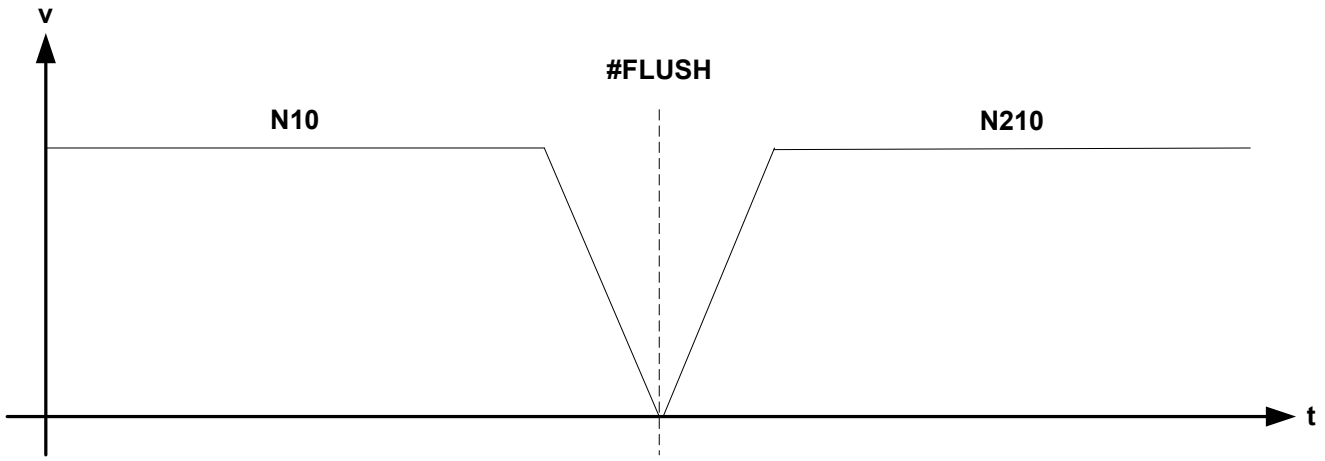


Abb. 99: Wirkungsweise #FLUSH zwischen 2 Verfahrsätzen

Syntax:

#FLUSH CONTINUE

Durch den Zusatz CONTINUE kann das generelle Anhalten am Ende des letzten Verfahrsatzes vermieden werden. Wenn dem Interpolator bereits der nächste Verfahrsatz vorliegt, dann wird ohne Stop die Bewegung fortgesetzt. Nur wenn noch kein weiterer Verfahrsatz vorliegt, dann wird im Interpolator gewartet.

Programmierbeispiel

#Flush Continue

Zwischen den Verfahrsätzen N10 und N210 ist eine WHILE-Schleife mit vielen nicht bewegungsrelevanten NC-Sätzen programmiert (Parameterrechnungen, Variablenzugriffe etc.). Dadurch wird der Verfahrsatz N10 im Kanal zurückgehalten und die Bereitstellung des Verfahrsatzes N210 verzögert. Mit **#FLUSH CONTINUE** wird der NC-Kanal gezwungen, alle Verfahrsätze vor der WHILE-Schleife an den Interpolator weiterzureichen. Dieser kann dann schon mit der Verfahrbewegung bis einschließlich N10 starten. Liegt im Interpolator vor Ende von N10 der nächste Bewegungssatz N210 bereits vor, so wird ohne anzuhalten die Bewegung fortgesetzt.

```
N05  G01 F1000 G90 X150
N10  X200
N20  #FLUSH CONTINUE
N30  $WHILE ...
N40    P1 = [P2 * P3] + V.E...
N50    V.L....
...
N200 $ENDWHILE
N210 X250
...
M30
```

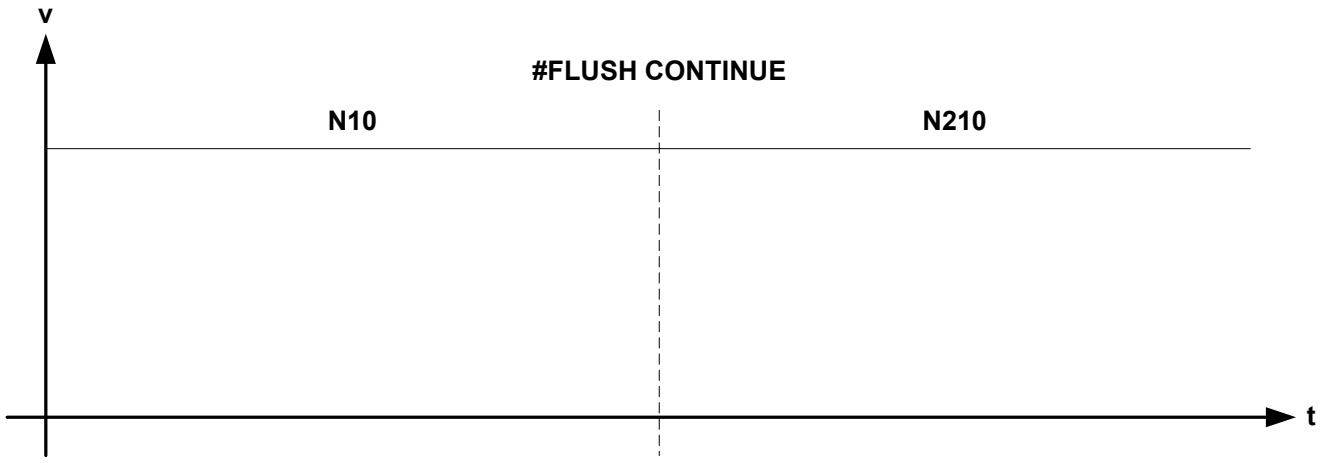


Abb. 100: Wirkungsweise #FLUSH CONTINUE zwischen 2 Verfahrsätzen

Syntax:

#FLUSH WAIT

Im Gegensatz zu **#FLUSH** bzw. **#FLUSH CONTINUE** wird durch den Befehl **#FLUSH WAIT** auch die Programmdecodierung unterbrochen. Alle bis zu diesem Befehl an den NC-Kanal ausgegebenen NC-Sätze werden im Interpolator abgearbeitet. Danach ist der NC-Kanal "leergelaufen", die Achsen stehen und Interpolator und wartende Programmdecodierung haben den gleichen Stand, d.h. sind synchronisiert. Erst wenn dieser Zustand erreicht ist, wird die weitere Programmdecodierung automatisch wieder fortgesetzt.

Achtung

Wurden zuvor synchronisierte **oder unsynchronisierte (MOS)** M/H-Funktionen bzw. #MSG-Befehle an die SPS ausgegeben, so wartet die Steuerung bis zu deren Annahme bzw. Synchronisation durch die SPS. Die Programmdecodierung ist also solange unterbrochen!

Programmierbeispiel

#Flush Wait

Mit **#FLUSH WAIT** wird die Programmdecodierung angehalten und der NC-Kanal gezwungen, alle Verfahrsätze vor der WHILE-Schleife an den Interpolator weiterzureichen. Wenn der Interpolator den Satz N10 ausgeführt und die Position X200 erreicht hat, wird die Programmdecodierung fortgesetzt.

```
N05  G01 F1000 G90 X150
N10  X200
N20  #FLUSH WAIT
N30  $WHILE ...
N40    P1 = [P2 * P3] + V.E...
N50    V.L....
...
N200 $ENDWHILE
N210 X250
...
M30
```

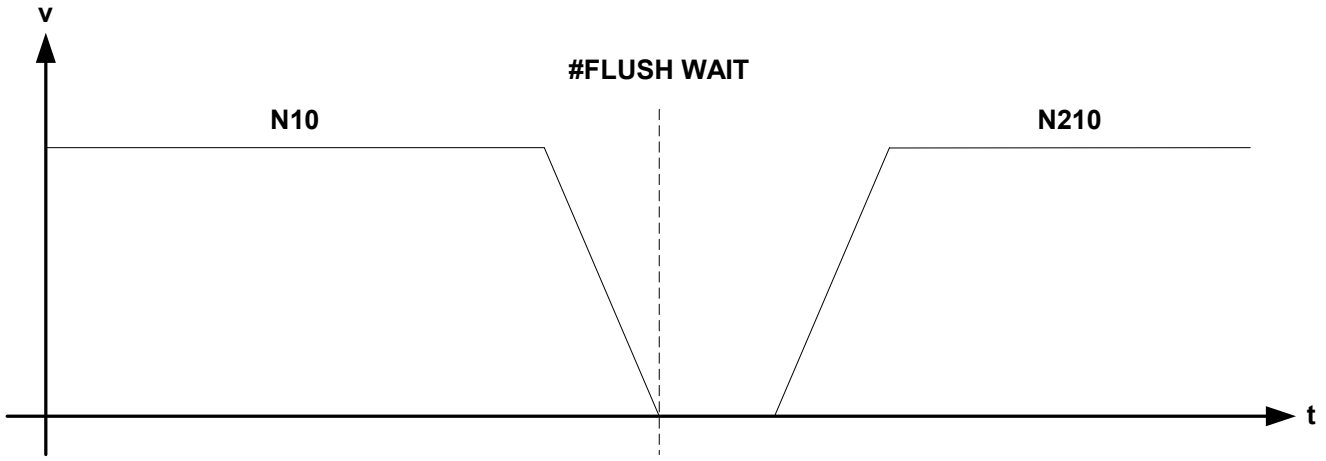


Abb. 101: Wirkungsweise #FLUSH WAIT zwischen 2 Verfahrsätzen

12.5 Satzübergreifender Kommentar (#COMMENT BEGIN/END)

Syntax:

#COMMENT BEGIN	Beginn des Kommentarblockes
:	Auskommentierter Bereich. Sämtliche Zeilen werden vom NC-Kern ohne
:	Auswertung überlesen.
#COMMENT END	Ende des Kommentarblockes

Beide Befehle müssen jeweils in einer eigenen NC-Zeile stehen, d.h. weitere NC-Befehle im selben Satz sind nicht zulässig. Ausnahme ist zeilenweiser Kommentar, der mit "(" und ")" einzurahmen ist.

Kommentarblöcke können innerhalb oder außerhalb von NC-Programmen definiert werden, sind jedoch auf den aktuellen File beschränkt, da zum einen Aufrufe von und zum anderen Rücksprünge aus globalen Unterprogrammen, die im Kommentar eingeschlossen sind, nicht ausgewertet werden.

Applikationsabhängig kann die Schachtelung von Kommentaren freigeschaltet werden.

Programmierbeispiel

Satzübergreifender Kommentar

```
:
#COMMENT BEGIN     Beginn Kommentarbereich 1
...
#COMMENT BEGIN     Beginn Kommentarbereich 2
...
#COMMENT END       Ende Kommentarbereich 2
...
#COMMENT END       Ende Kommentarbereich 1
:
Schachtelung nicht freigeschaltet
:
#COMMENT BEGIN     Beginn Kommentarbereich 1
...
#COMMENT BEGIN     Beginn Kommentarbereich 2
...
#COMMENT END       Ende Kommentarbereich 1 und 2
:
```

In folgenden Fällen erzeugt der NC-Kern Fehlermeldungen:

- Dateiende wird innerhalb eines Kommentarblocks eingelesen
- #COMMENT END wird ohne ein zuvor programmiertes #COMMENT BEGIN eingelesen.
- Nach den Comment-Befehlen sind weitere NC-Befehle im **gleichen** Satz programmiert.

12.6 Warten auf Ereignis (#WAIT FOR)

Syntax:
#WAIT FOR <*wait_condition*>

Bei diesem Befehl wird die Decodierung des NC-Programms so lange angehalten, bis der arithmetische Ausdruck erfüllt ist (TRUE bzw. ≥0,5).

Programmierbeispiel

Warten auf Ereignis

```
N10 #WAIT FOR V.E.EXT1 == 5              ;Die Decodierung des NC-Programms
                                         ;wird an dieser Stelle angehalten,
                                         ;bis der Wert der externen
....                                     ;Variable 5 ist.
N50 #WAIT FOR V.E.EXT2 == TRUE           ;Die Decodierung des NC-Programms
                                         ;wird an dieser Stelle angehalten,
                                         ;bis der Wert der externen
....                                     ;Variable ≥ 0,5 ist.
```

Achtung

Die Look-Ahead Mechanismen der Steuerung können dazu führen, dass eine bestimmte Anzahl bewegungsrelevanter NC-Sätze, die vor dem #WAIT FOR programmiert sind, im NC-Kanal zurückgehalten werden. Sind diese zurückgehaltenen Bewegungssätze für die Generierung des Ereignisses selbst relevant, entsteht eine Verklemmung und das NC-Programm bleibt ohne erkennbaren Grund stehen.

Durch Programmierung von #FLUSH oder #FLUSH CONTINUE unmittelbar vor #WAIT FOR wird die Ausführung aller vorherigen Bewegungssätze erzwungen und eine Verklemmung vermieden.

Programmierbeispiel

Warten auf Ereignis

```
....
Nxx G01 X.. Y.. Z.. F..
Nxx G01 X.. Y.. Z.. F..
Nxx G01 X.. Y.. Z.. F..
Nxx G01 X.. Y.. Z.. F..
Nxx G01 X.. Y.. Z.. F..
Nxx G01 X.. Y.. Z.. F..
Nxx G01 X.. Y.. Z.. F..
Nxx #FLUSH CONTINUE                      ;Zwangsweises Ausführen aller
                                         ;gepufferten Bewegungssätze
Nxx #WAIT FOR V.E.XX > 123               ;Zwangsweises Ausführen aller
                                         ;gepufferten Bewegungssätze
....
```

12.7 Mindestradius für tangentiale Vorschubanpassung (#TANGFEED)

Versionshinweis

Ab Version **V2.11.2010.02** ersetzt der Befehl **#TANGFEED [...]** den Befehl **#SET TANGFEED RMIN [...]**. Dieser ist aus Kompatibilitätsgründen weiterhin verfügbar, es wird aber empfohlen, diesen in neuen NC-Programmen nicht mehr zu verwenden.

Dieser Befehl ergänzt die tangentiale Vorschubanpassung, die mit G10/G11 [▶ 567] programmiert wird. Der Mindestradius wird bei Zirkularsätzen mit der Kennung für Vorschubanpassung berücksichtigt. Nur wenn der programmierte Zirkularradius größer oder gleich dem Minimalradius ist, wird der Vorschub korrigiert.

Syntax:

#TANGFEED [RMIN=.. **]**

RMIN=.. Minimaler Konturradius Rmin in [mm, inch], bis zu dem der Tangentialvorschub korrigiert wird.

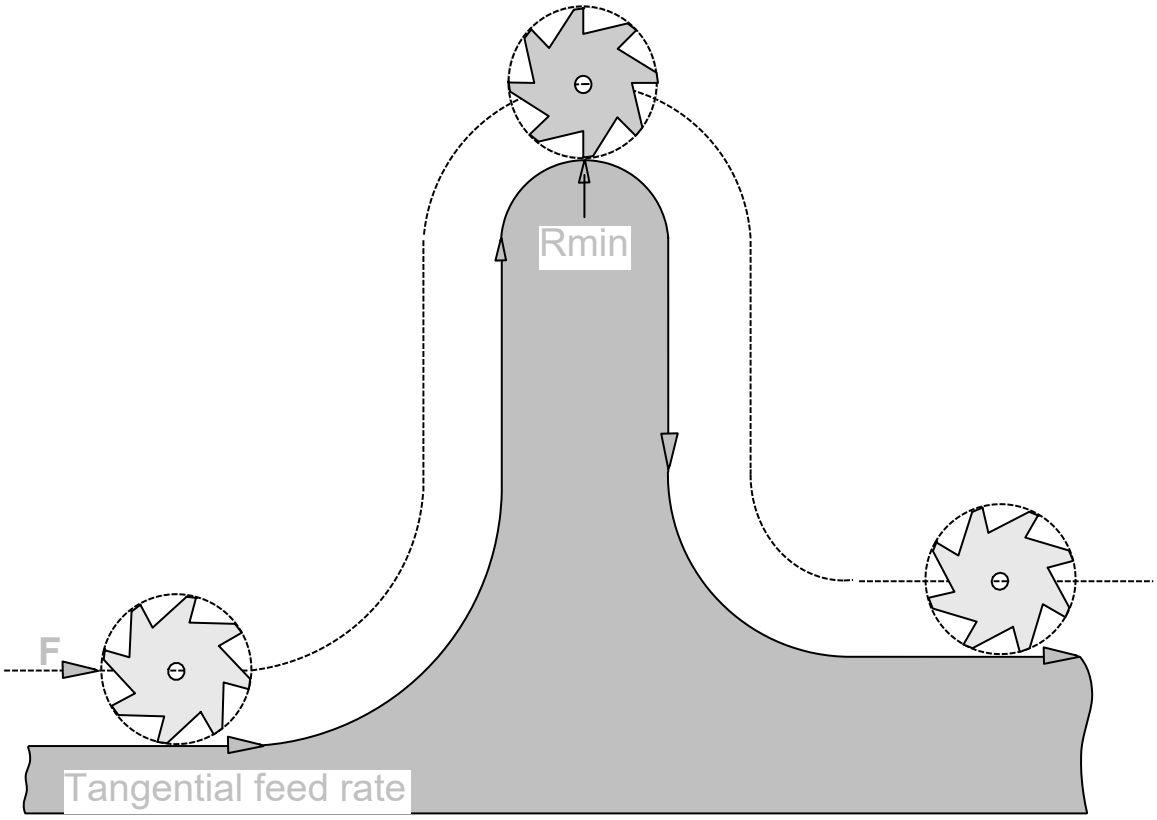


Abb. 102: Programmierung des tangentialen Vorschubes.

- Ein neu programmierter Minimalradius wird mit dem nächsten relevanten Zirkularbewegungssatz berücksichtigt.
- Die Angabe des Minimalradius ist optional. Die tangentiale Vorschubanpassung mit G11 [▶ 567] wird auch dann durchgeführt, wenn #TANGFEED [RMIN…] nicht programmiert wurde oder der Radius mit Null belegt ist.

Programmierbeispiel

Mindestradius für tangentiale Vorschubanpassung

```
(Kontur mit WRK und tangentialer Vorschubanpassung)
N10 V.G.WZR=5
N20 #TANGFEED [RMIN=3]          (Definiere Minimalradius rmin = 3mm)
N30 G41 G01 X0 Y20 G11 F600     (Anwahl WRK und Vorschubanpassung)
N40 X20 Y20
N50 G03 X40 R10                 (Vorschubanpassung)
N60 G01 Y40
N70 G02 X44 R2                  (Keine Vorschubanpassung → rprg < rmin)
N80 G01 Y20
N20 #TANGFEED [RMIN=6]          (Definiere neuen Minimalradius rmin =
6mm)
N90 G03 X54 R5                  (Keine Vorschubanpassung → rprg < rmin)
N100 G01 Y50
N110 X60 G40 G10                (Abwahl WRK und Vorschubanpassung)
N120 X0 Y0
```

12.8 Unterdrückung von Verschiebungen (#SUPPRESS OFFSETS)

Der Befehl bewirkt in Kombination mit einem Bewegungssatz das Anfahren der programmierten Achspositionen ohne Berücksichtigung aktiver Verschiebungen.

Ohne Angabe einer bestimmten Verschiebungsart werden alle Verschiebungen **im NC-Satz** unterdrückt.

Versionshinweis

Ab Version **V2.11.2032.07** kann zusätzlich angewählt werden, welche aktiven Verschiebungen (Nullpunkte, Bezugspunkte, PSET, Messen usw.) im NC-Satz unterdrückt werden sollen.

Verschiebungen aufgrund aktiver kinematischer und / oder kartesischer Transformationen (CS, ACS, ROTATION) werden durch den Befehl **nicht** unterdrückt. Zur Unterdrückung sämtlicher Arten von Verschiebungen muss dann alternativ der **Befehl** #MCS ON/OFF [▶ 773] verwendet werden.

Syntax:

#SUPPRESS OFFSETS [**[ZERO ADD_ZERO PSET CLAMP TOOL MEASURE MANUAL]**] nicht modal
<Achsname>.. {*<Achsname>*..}

ZERO	Nullpunktverschiebungen
ADD_ZERO	Additive Nullpunktverschiebungen bzw. Bezugspunktverschiebungen
PSET	Istwertverschiebungen
CLAMP	Platzversätze
TOOL	Werkzeugversätze
MEASURE	Messverschiebungen
MANUAL	Handbetriebverschiebungen
<Achsname>..	Achspositionen, die ohne Verschiebungen angefahren werden sollen.

Programmierbeispiel

Unterdrückung von Verschiebungen

```
%suppress_offsets
;Nullpunktverschiebungen für G54 definieren
N05 V.G.NP[1].V.X=11
N10 V.G.NP[1].V.Y=22

;Anwahl Nullpunktverschiebungen X11 Y22
N15 G54

;Anwahl und Definition additive Nullpunktverschiebungen X10 Y20
N20 G92 X10 Y20

N25 X100 Y150                                   ;Position X=121 Y=192

;Alle aktiven Verschiebungen unterdrücken
N30 #SUPPRESS OFFSETS X50 Y100                  ;Position X=50 Y=100

;Additive Nullpunktverschiebungen unterdrücken
N35 #SUPPRESS OFFSETS [ADD_ZERO] X50 Y100       ;Position X=61 Y=122

N40 X200 Y250                                   ;Position X=221 Y=292
N99 M30
```

12.9 Einstellungen für das Messen

12.9.1 Umschalten des Messtyps (#MEAS MODE)

Mit dem folgenden Befehl kann der Messtyp über das NC-Programm festgelegt werden.

Syntax:

#MEAS MODE [**[***<meas_type>***]**

<meas_type> Messtyp gemäß Kapitel Umschalten des Messtyps [▶ 96].

Über die Programmierung von #MEAS MODE ohne Parameter wird der Default-Messtyp angewählt, der in dem Kanalparameter P-CHAN-00057 angegeben ist.

Programmierbeispiel

Umschalten des Messtyps

```
N10 #MEAS MODE[3]   (Messtyp 3)
N20 G100 X150       (Messbewegungssatz)
..
N100 #MEAS MODE     (Default-Messtyp)
..
M30
```

12.9.2 Erweiterte Programmierung (#MEAS, #MEAS DEFAULT)

Alternativ zum #MEAS MODE Befehl bietet der folgende Befehl die Möglichkeit, noch weitere Messparameter festzulegen. Die gewählten Parametereinstellungen bleiben bis zum Programmende wirksam. Bei einem neuen Programmstart gelten wieder die Default-Einstellungen aus den Konfigurationslisten. Damit die Messparameter einer Achse geändert werden dürfen, muss sie als Messachse gekennzeichnet sein (d.h. der Achsparameter P-AXIS-00118 muss auf 1:

Syntax:

#MEAS [[**TYPE=..**] [**ERR_NO_SIGNAL=..**] [[**SIMU_OFFSET=..**] | [**TRIGGER**]] |
[{**AX**=<*Achsname*> | **AXNR=..**} [**SIGNAL=**<*ident*>] [**EDGE=**<*ident*>] [**INPUT=..**]
[**G107** | **G108**]] **]**

TYPE=..	Neuer Messtyp gemäß Kapitel Messfunktionen [▶ 96]. Dieser Messtyp ist bis zu einer erneuten Änderung oder bis Programmende gültig.
ERR_NO_SIGNAL=..	Verhalten bei nicht erfasstem Messsignal: 0: keine Fehlermeldung 1: Fehlermeldung bei Abwahl der Messfahrt (Default)
SIMU_OFFSET=..	Dieses Schlüsselwort ist nur speziell bei der Messsimulation im Zusammenhang mit dem Achsparameter P-AXIS-00112=4 wirksam. Der Wert in [mm, inch] verschiebt den simulierten Defaultmesspunkt bezogen auf die Bahnbewegung ausgehend von den programmierten Zielpunkten. Bei **Messtyp 2** kann der Defaultmesspunkt in positiver bzw. negativer Richtung durch SIMU_OFFSET verschoben werden. Der eventuell zusätzliche Offset durch den Achsparameter [2] [▶ 894]P-AXIS-00114 wird hierbei nicht berücksichtigt.
TRIGGER	Auslösen eines programmierten Messsignals. Wird nur verwendet im Zusammenhang mit der Funktion satzübergreifendes Kantenstoßen G107/G108 [▶ 112]. Ist nur wirksam, wenn P-CHAN-00257 aktiv ist.

Hinweis

SIMU_OFFSET und TRIGGER sind exklusiv und dürfen nicht in Kombination mit den achsspezifischen Schlüsselwörtern programmiert werden.

AX=<*Achsname*>	Name der Achse, deren Messparameter geändert werden sollen. Die Achse muss als Messachse konfiguriert sein.
AXNR=..	Logische Nummer der Achse, deren Messparameter geändert werden sollen. Die Achse muss als Messachse konfiguriert sein, positive Ganzzahl
SIGNAL=<*ident*>	Name der Messsignalquelle, die für die Messung verwendet werden soll (s.P-AXIS-00516). Gültige Kennungen:
	PLC:Messsignal über PLC
	DRIVE: Messsignal über Positionslatch im Antrieb
	FIXED_STOP: Messsignal durch Fahren auf Festanschlag
	PLC_FIRST_EVENT: Messsignal über PLC, die Messfahrt beendet sich, sobald eine Achse das Messereignis erhalten hat
	PLC_EXT_LATCH_CONTROL: Messen mit Messinterface für externe Hardware (s. [HLI// Messen mit externer Messhardware])
	EXT_PROBE_WITH_DRIVE: Messen mit Messinterface für externe Hardware, Messposition über Antriebsschnittstellen (s. [HLI// Messen mit externer Messhardware])
EDGE=<*ident*>	Relevante Messflanke (s. P-AXIS-00518). Gültige Kennungen:
	POS: Positive (steigende) Messflanke
	NEG: Negative (fallende) Messflanke
INPUT=..	Nummer des Messeingangs am Antrieb, der für die Messung verwendet werden soll (s. P-AXIS-00517).
	Für Messsignal DRIVE:
	1: 1. Messeingang
	2: 2. Messeingang
	Für Messsignal PLC_EXT_LATCH_CONTROL:
	1 .. 255: Nummer des Messeingangs der externen Messhardware
G107	Abwahl der Funktionalität Kantenstoßen für diese Achse, d.h. für diese Achse wird beim Kantenstoßen kein Messwert erfasst.
G108	Anwahl der Funktionalität Kantenstoßen für diese Achse. Voraussetzung ist, dass für die Achse in den Achsparametern die Funktionalität „Kantenstoßen“ aktiviert ist (siehe P-AXIS-00098).

Für die haltende Messfahrt (über mehrere Bewegungssätze) in Verbindung mit der Funktion Kantenanleimen (Messtyp 8) stehen im #MEAS-Befehl folgende Erweiterungen zur Verfügung.

Syntax:

#MEAS [**ON** | **OFF**] **[** [<Messparameter>]]

ON	Anwahl der Messfahrt für Messtyp 8. In anschliessenden Bewegungssätzen wird dann in allen gesetzten bzw. programmierten Achsen gemessen.
OFF	Abwahl der Messfahrt für Messtyp 8

Syntax:

#MEAS DEFAULT [**[** {**AX**=<*Achsname*> | **AXNR**=..} **]**

DEFAULT	Zurücksetzen der über den #MEAS-Befehl geänderten *achsbezogenen* (AX, AXNR) Parametereinstellungen (SIGNAL, EDGE, INPUT, G107/G108). Es werden wieder die Messeinstellungen aus den Achsparameterlisten wirksam.

Hinweis

Für SERCOS-Antriebe mit Positionslatch im Antrieb (SIGNAL=DRIVE) können Messflanke (EDGE) und Messeingang (INPUT) nicht geändert werden, da dazu ebenfalls Parameteränderungen im Antrieb notwendig sind.

Programmierbeispiel

Setzen von Messparametern:

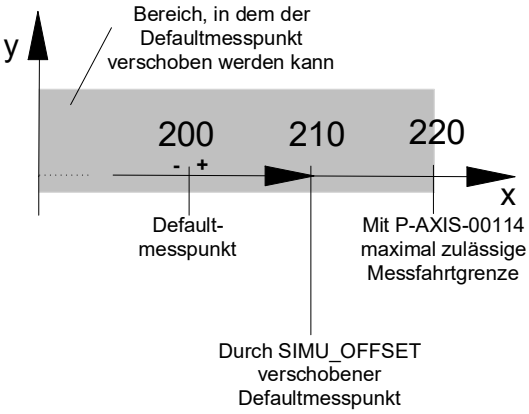
Anwahl eines anderen Messtyps:

```
N100 #MEAS [TYPE=2]
```

Setzen des Messpunkts für die Messsimulation bei Messtyp 2:

```
N10  G00 X0 Y0
N20  #MEAS [TYPE=2 SIMU_OFFSET=10]
N30  G100 X200
```

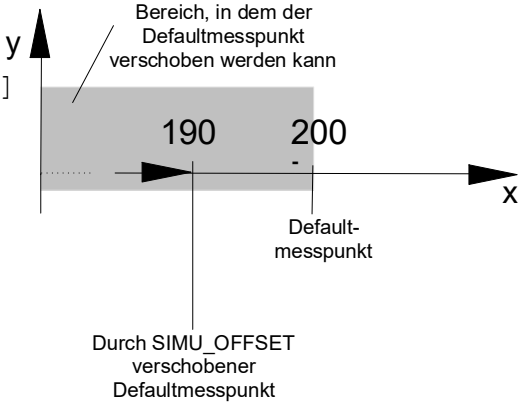
Zielpunkt der Messfahrt (Defaultmesspunkt)



Bei allen anderen Messtypen ist nur eine Verschiebung in negativer Richtung (entgegen der Bahnbewegung) möglich.

```
N10  G00 X0 Y0
N20  #MEAS [TYPE=1 SIMU_OFFSET=-10]
N30  G100 X200
```

Zielpunkt der Messfahrt (Defaultmesspunkt)



Aktivieren Messen mit Fahren auf Festanschlag für die X- und Y-Achse:

```
N100 #MEAS [AX=X AX=Y SIGNAL=FIXED_STOP]
```

Aktivieren Messsignal über PLC auf negative Flanke:

```
N100 #MEAS [AXNR=1 SIGNAL=PLC EDGE=NEG]
```

Deaktivieren der Kantenstoßfunktionalität für Y- und Z-Achse:

```
N100 #MEAS [AX=Y AX=Z G107]
```

Wiederherstellen der Messeinstellungen aus den Achsparametern für alle Bahnachsen:

```
N100 #MEAS DEFAULT
```

Wiederherstellen der Messeinstellungen aus den Achsparametern für die X-Achse:

```
N100 #MEAS DEFAULT [AX=X]
```

Haltende Messfahrt mit allen Messachsen:

```
N5 #MEAS ON [TYPE=8]
N10 G01 X100 Y100 F1000
N20 G01 Z200
N30 G01 X200 Y200
N40 #MEAS OFF
```

Haltende Messfahrt mit Messen in X- und Y-Achse, Messsignal über PLC, positive Flanke, keine Fehlermeldung bei nicht erfasstem Messsignal

```
N5 #MEAS ON [TYPE=8 AX=X AX=Z SIGNAL=PLC EDGE=POS ERR_NO_SIGNAL=0]
N10 G01 X100 Y100 Z10 F1000
N20 G01 X200 Y150 Z25
N30 #MEAS OFF
```

12.10 Anwahl der Istwertverschiebung (#PSET)

Die Anwahl der Istwertverschiebung erfolgt über:

Syntax:

#PSET <*Achsname*><*expr*> {<*Achsname*><*expr*>} nicht modal

<*Achsname*><*expr*> Neue Istposition der Achse in [mm, inch]

Die Istpositionen werden absolut programmiert. Mit diesem Befehl ist keine Verfahrbewegung verbunden.

Achsen, für die kein neuer Istwert programmiert wurde, behalten ihren aktuellen Istwert, d.h. der entsprechende Offset ändert sich für die betreffende Achse nicht.

Der #PSET-Befehl kann beliebig oft wiederholt werden.

Die vom #PSET-Befehl verursachte Verschiebung des Koordinatensystems ist nicht programmübergreifend.

Durch eine Referenzpunktfahrt (G74) wird die vom #PSET-Befehl verursachte Verschiebung des Koordinatensystems aufgehoben.

12.10.1 Abwahl der Istwertverschiebung (#PRESET)

Syntax:

#PRESET {<*Achsname*> <*value*>} nicht modal

<*Achsname*><*value*> Istwertverschiebung der Achse wird zurückgesetzt. Der Koordinatenwert in [mm, inch] ist nur aus syntaktischen Gründen erforderlich, ansonsten jedoch nicht relevant.

Wird #PRESET ohne Achsangabe programmiert, so wird in allen Achsen die Istwertverschiebung aufgehoben.

Hinweis

Bei angewählter WRK, Spiegelung oder Durchmesserprogrammierung darf #PSET oder #PRESET nicht programmiert werden.

("nicht modal" gilt nur für die Befehle #PSET bzw. #PRESET. Die Istwertverschiebung selbst ist natürlich bis zur erneuten An- bzw. Abwahl mit #PRESET gültig.)

Programmierbeispiel

Abwahl der Istwertverschiebung

```
N10 X50 Y10 Z0
N20 #PSET X20 Y20        (Position setzen für X und Y)
N30 X10
N40 #PRESET X10          (X-Position zurücksetzen; Wert 10 nicht rele-
vant)
N50 X30
N60 #PRESET              (Alle Achsen zurücksetzen)
N70 M30
```

Die folgende Abbildung zeigt die Position der x-Achse nach Ausführung des jeweiligen NC-Satzes in Maschinenkoordinaten:

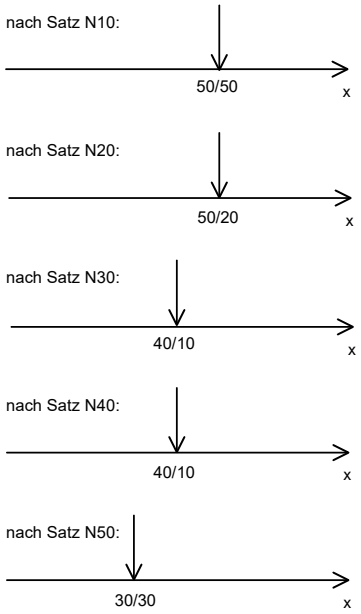


Abb. 103: Positionen der x-Achse in Maschinenkoordinaten / Programmierkoordinaten.

(In diesem Beispiel sind keine anderen Koordinatentransformationen angewählt).

12.11 Synchronbetrieb

12.11.1 Programmierung von Achskopplungen (#SET AX LINK, #AX LINK)

Im NC-Programm können Achskopplungen mit folgendem NC-Befehl festgelegt werden.

Syntax:

#SET AX LINK [*<Kopplungsgruppe>*, *<Slave>*=*<Master>* {,*<Slave>*=*<Master>*} **]**

oder alternativ

#AX LINK [**NBR**] **[** *<Kopplungsgruppe>*, *<Slave>*=*<Master>* {,*<Slave>*=*<Master>*} **]**

<Kopplungsgruppe>	Nummer der Kopplungsgruppe (1)
<Slave>	Achsbezeichnung oder logische Achsnummer der Slaveachse des Kopplungspaares i (2)
<Master>	Achsbezeichnung oder logische Achsnummer der Masterachse des Kopplungspaares i (2)
NBR	Mit dem Logikschalter NBR wird auf die Auswertung von logischen Achsnummern anstatt von Achsnamen umgeschaltet. Die Achskopplungen müssen dann über die logischen Achsnummern definiert werden. Die Achsen müssen noch nicht im Kanal vorhanden sein. Erst bei der Aktivierung der Kopplungsgruppe wird ihr Vorhandensein geprüft!

Hinweis

Es muss mindestens ein Master-Slave-Kopplungspaar pro Kopplungsgruppe definiert werden.

Die Kopplung von Spindeln wird im Kapitel Spindelsynchronbetrieb [▶ 708] genauer beschrieben!

Allgemeine Handhabung und Wirkungsweise:

- Die Definitionen von Kopplungsgruppen sind programmübergreifend gültig. D.h. sie können in einem nachfolgenden NC-Programm wieder angewählt werden.
- Die Kopplungsgruppe mit der Kopplungsnummer "0" kann nicht im NC-Programm definiert werden. Sie wird in den Kanalparameter [1] [▶ 894]-2 festgelegt.
- Eine nicht aktive Kopplungsgruppe kann jederzeit geändert werden. Die bisher definierten Kopplungen werden überschrieben.
- Eine aktive Kopplungsgruppe darf nicht geändert werden.
- Rekursive Achskopplungen sind nicht zulässig. Eine Masterachse kann in einer Kopplungsgruppe nicht gleichzeitig Slaveachse sein und umgekehrt. Eine Slaveachse kann in einer anderen Kopplungsgruppe nicht gleichzeitig Masterachse sein und umgekehrt.
- Master- und Slaveachse eines Kopplungspaares dürfen nicht identisch sein.
- Eine Hauptachse darf keine Slaveachse sein.
- Eine Slaveachse kann nur einer Masterachse zugeordnet sein, aber eine Masterachse kann mehrere Slaveachsen besitzen.
- Bei Aktivierung des Synchronbetriebs wird geprüft, ob alle Master- und Slaveachsen in der Achsgruppe des NC-Kanals vorhanden sind.
- Master- und Slaveachsen müssen vom gleichen Achstyp sein und im gleichen Achsmode betrieben werden.
- Der NC-Befehl zur Programmierung der Kopplungsgruppe muss alleine im NC-Satz stehen.
- Die Nummer der Kopplungsgruppe kann auch über mathematische Ausdrücke programmiert werden. Dabei muss das Ergebnis ein positiver Integerwert sein.

- Eine Slaveachse darf nicht gleichzeitig eine aktive Nachführachse (#CAXTRACK) sein.

1 ... [Max. Anzahl Kopplungsgruppen–1], siehe [6] [▶ 894] -2.11

Max. Anzahl Kopplungspaare, siehe [6] [▶ 894] -2.12

Hinweis

Die Aktivierung der entsprechenden Achskopplung muss explizit mit #ENABLE AX LINK [▶ 365] erfolgen.

Programmierbeispiel

Programmierung von Achskopplungen

```
#SET AX LINK[1, Z2=Z]                    ;Z2 wird als Slave an Hauptachse Z ge-
                                         koppelt
#SET AX LINK[2, Y2=Y, X2=X]              ;Kopplung von Y2 mit Y und X2 mit X
#SET AX LINK[3, X1=X, X2=X, X3=X]        ;Kopplung von X1,X2,X3 als Slaves an
                                         gleiche Masterachse X
:
oder alternativ
#AX LINK[1, Z2=Z]
#AX LINK NBR[2, 8 = 2, 9 = 3]            ;Kopplung über log. Achsnummern
```

12.11.2 Erweiterte Programmierung von Achskopplungen („SOFT-GANTRY“) (#SET AX LINK, #AX LINK)

Versionshinweis

Die Verfügbarkeit dieser Funktionalität ist von der Konfiguration und dem Versionsumfang abhängig.

Bahnachsen können auch als s.g. Gantryachsen betrieben werden. Im Gegensatz zum normalen Synchronbetrieb sind hierbei zusätzliche Überwachungsmechanismen bzgl. Positionsabweichung aktiv und es gelten spezifische Fehlerreaktionen. Beim mechanischen (auch statischen) Gantrybetrieb sind die Achsen bedingt durch den Maschinenaufbau fest miteinander gekoppelt und durch die Konfiguration der Maschine festgelegt. Ein dynamischer Wechsel der Gantrykopplung ist nach Hochlauf der Steuerung nicht möglich (siehe nachfolgende Abbildung).

Neben Bahnachsen können auch Spindeln im Synchronbetrieb gefahren werden. Eine genaue Beschreibung ist in Kapitel Spindelsynchronbetrieb [▶ 708] zu finden.

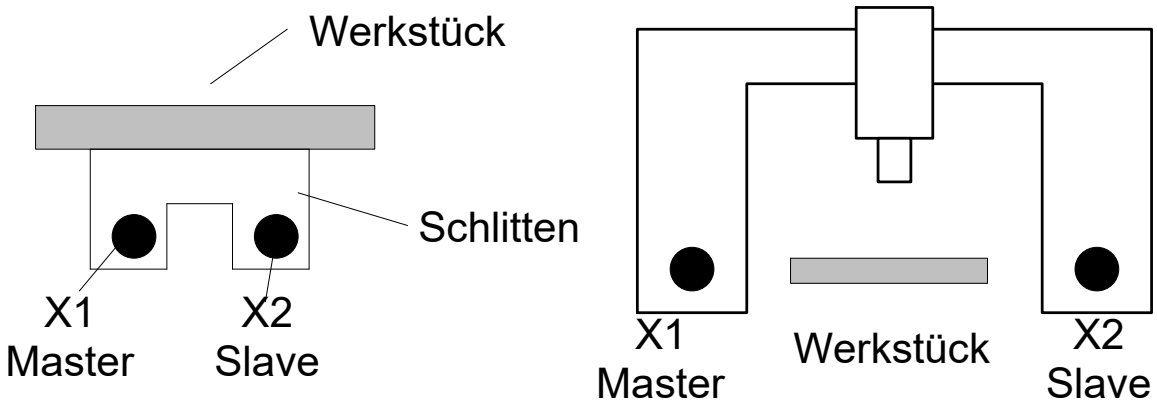


Abb. 104: Mechanischer Gantrybetrieb

Maschinen, die von ihrem Grundaufbau her keinen mechanischen Gantrybetrieb ermöglichen, z.B. Fräsmaschinen mit zwei unabhängigen Schlitten, können durch Programmierung im Gantrybetrieb gefahren werden. Dies ist z.B. dann notwendig, wenn zum Spannen und Bearbeiten großer Werkstücke solche Schlitten miteinander gekoppelt werden müssen (siehe nachfolgende Abbildung)

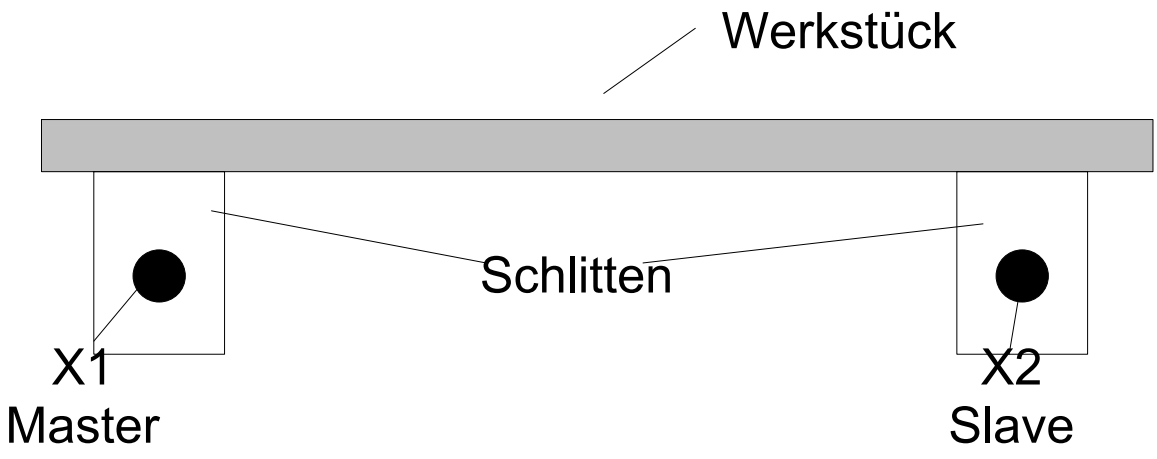


Abb. 105: Programmierbarer Gantrybetrieb („Soft-Gantry“)

Dazu stehen die in Kapitel Programmierung von Achskopplungen [▶ 359] beschriebenen Befehle #SET AX LINK bzw. #AX LINK in einer erweiterten Syntax zur Verfügung:

Syntax:

#SET AX LINK [*<Kopplungsgruppe>*, **[** *<Slave>*=*<Master>*,**G** [,*<limit_1>*, *limit_2>*] **]**
{, **[** *<Slave>*=*<Master>*,**G** [,*<limit_1>*, *limit_2>*] **]** } **]**

oder alternativ

#AX LINK [**NBR**] **[** *<Kopplungsgruppe>*, **[** *<Slave>*=*<Master>*,**G** [,*<limit_1>*, *limit_2>*] **]**
{, **[** *<Slave>*=*<Master>*,**G** [,*<limit_1>*, *limit_2>*] **]** } **]**

<Kopplungsgruppe>	Nummer der Kopplungsgruppe (1)
<Slave>	Achsbezeichnung oder logische Achsnummer der Slaveachse des Kopplungspaares i (2)
<Master>	Achsbezeichnung oder logische Achsnummer der Masterachse des Kopplungspaares i (2)
NBR	Mit dem Logikschalter NBR wird auf die Auswertung von logischen Achsnummern anstatt von Achsnamen umgeschaltet. Die Achskopplungen müssen dann über die logischen Achsnummern definiert werden. Die Achsen müssen noch nicht im Kanal vorhanden sein. Erst bei der Aktivierung der Koppelgruppe wird ihr Vorhandensein geprüft!
G	Schlüsselwort für "Gantrykopplung"

(1) 1 ... [Max. Anzahl Kopplungsgruppen–1], siehe [6] [▶ 894] -2.11

(2) Max. Anzahl Kopplungspaare, siehe [6] [▶ 894] -2.12

Bei einer Gantrykopplung dienen die folgenden Werte zur zweistufigen Überwachung der zulässigen Positionsdifferenz der Gantryachsen. Angabe in [mm]. Positive Realzahl:

<limit_1>	Erste Überwachungsgrenze in [mm, inch]. Wird diese Grenze überschritten, wird die Bewegung abgebrochen und die Steuerung geht in den Fehlerzustand. Die Positionsdifferenz wird im Standardfall während RESET abgebaut. Applikationsspezifisch kann das Verfahren auch abweichend realisiert sein.
<limit_2>	Zweite Überwachungsgrenze in [mm, inch]. Bei Überschreitung dieser Grenze erfolgt die Ausgabe eines nicht RESET-fähigen Fehlers. Die Steuerung muss ausgeschaltet und die Positionsdifferenz manuell behoben werden.

Hinweis

Werden die Überwachungsgrenzen nicht programmiert, so gelten die Defaultwerte aus dem Achsparameterdatensatz P-AXIS-00072 und P-AXIS-00071 der Slaveachse.

Ergänzungen zur allgemeinen Handhabung und Wirkungsweise:

- Die Gantrykopplung erfolgt genau in den Positionen, in denen die Achsen zum Zeitpunkt der Anwahl der Kopplung sind. Die Angabe eines Offsets im NC-Befehl ist nicht notwendig, da die Berechnung des Offsets intern im Lageregler über die Sollpositionen erfolgt.
- Bei der Bahnbewegung werden die Dynamikdaten der Slaveachse berücksichtigt.
- Eine bei RESET oder am Programmende noch aktive Kopplung wird aus Sicherheitsgründen implizit beim nächsten Programmstart wiederhergestellt. Dieses Verhalten ist parametrierbar (P-CHAN-00104, P-CHAN-00105).

Programmierbeispiel

Erweiterte Programmierung von Achskopplungen („SOFT-GANTRY“)

```
N10 #SET AX LINK[1, [Y2=Y1,G,0.01,0.25]]      Gantrykopplung von Y1 als Mas-
                                              terachse und Y2 als Slaveach-
                                              se. Erste Grenze ist 10µm,
                                              Zweite Grenze ist 250 µm.
N20 #SET AX LINK[2, [Y2=Y1,G]]                Gantrykopplung von Y1 (Master)
                                              und Y2 (Slave). Es gelten die
                                              Überwachungsgrenzen des Ach-
                                              sparameterdatensatzes der Y2-
                                              Achse.
N30 #SET AX LINK [3,[Y2=Y1]]                  Standardkopplung von Y2 mit
                                              Y1. Kein Gantrybetrieb.
oder alternativ
N10 #AX LINK[1, [Y2=Y1,G,0.01,0.25]]
N20 #AX LINK NBR[2, [8=2,G]]                  Gantrykopplung über log. Achs-
                                              nummern
```

Das parallele Bearbeiten von Werkstücken mit symmetrischer bzw. skalierter Kontur kann ebenfalls durch eine erweiterte Syntax des #SET AX LINK bzw. #AX LINK Befehls programmiert werden. In diesen Modi (Spiegeln bzw. Skalieren) erfolgt keine Überwachung von Positionsdifferenzen.

Syntax:

#SET AX LINK [*<Kopplungsgruppe>*, **[** *<Slave>*=*<Master>*,*<zähler>*, *<nenner>* **]**
{, **[** *<Slave>*=*<Master>*,*<zähler>*, *<nenner>* **]** } **]**

oder alternativ

#SET AX LINK [**NBR**] **[** *<Kopplungsgruppe>*, **[** *<Slave>*=*<Master>*,*<zähler>*, *<nenner>* **]**
{, **[** *<Slave>*=*<Master>*,*<zähler>*, *<nenner>* **]** } **]**

<Kopplungsgruppe>	Nummer der Kopplungsgruppe (1)
<Slave>	Achsbezeichnung oder logische Achsnummer der Slaveachse des Kopplungspaares i (2)
<Master>	Achsbezeichnung oder logische Achsnummer der Masterachse des Kopplungspaares i (2)

NBR	Mit dem Logikschalter NBR wird auf die Auswertung von logischen Achsnummern anstatt von Achsnamen umgeschaltet. Die Achskopplungen müssen dann über die logischen Achsnummern definiert werden. Die Achsen müssen noch nicht im Kanal vorhanden sein. Erst bei der Aktivierung der Koppelgruppe wird ihr Vorhandensein geprüft!
<zähler>, *<nenner>*	Ganzzahlen. Dienen zur Berechnung eines Kopplungsfaktors zwischen Master- und Slaveachse. Für den **resultierenden** Kopplungsfaktor gilt: -1 : Spiegelungskopplung. 1 : Standardkopplung; ist äquivalent zur bisherigen Syntax. 0 : Ausgabe einer Fehlermeldung.

Achtung

Faktoren, die eine reine Skalierung (Faktor ≠ 1) bzw. Skalierungen mit gleichzeitiger Spiegelung (Faktor ≠ -1) bewirken, sind momentan nicht zulässig. Es wird eine Warning ausgegeben und die Kopplung wird als Standardkopplung behandelt. D.h., es sind nur die **Kopplungsfaktoren 1 und –1** zulässig (siehe Beispiele)

(1) 1 ... [Max. Anzahl Kopplungsgruppen–1], siehe [6] [▷ 894] -2.11

(2) Max. Anzahl Kopplungspaare, siehe [6] [▷ 894] -2.12

Programmierbeispiel

```
N10 #SET AX LINK[1, [Y2 = Y1,1,-1]]      Spiegelungskopplung (Faktor -1)
N20 #SET AX LINK[1, [Y2 = Y1,-1,1]]      Spiegelungskopplung (Faktor -1)
N30 #SET AX LINK[1, [Y2 = Y1,-2,2]]      Spiegelungskopplung (Faktor -1)
N40 #SET AX LINK[1, [Y2 = Y1,1,1]]       Standardkopplung
N50 #SET AX LINK[1, [Y2 = Y1,2,2]]       Standardkopplung
N60 #SET AX LINK[1, [Y2 = Y1,0,1]]       Fehlermeldung, Programmabbruch
N70 #SET AX LINK[1, [Y2 = Y1,1,0]]       Fehlermeldung, Programmabbruch
N80 #SET AX LINK[1, [Y2 = Y1,1,2]]       Warning (Faktor 0.5), Standardkpl.
N90 #SET AX LINK[1, [Y2 = Y1,2,3]]       Warning(Faktor 0.666), Standardkpl.
N100 #SET AX LINK[1, [Y2 = Y1,3,2]]      Warning (Faktor 1.5), Standardkpl.
N110 #SET AX LINK[1, [Y2 = Y1,-1,2]]     Warning (Faktor -0.5), Standardkpl.
N120 #SET AX LINK[1, [Y2 = Y1,-3,2]]     Warning(Faktor -1.5), Standardkpl.
```

oder alternativ

```
N40 #AX LINK[1, [Y2 = Y1,1,1]]           Standardkopplung
N50 #AX LINK NBR[1, [8 = 2,2,2]]         Standardkoppl. über log. Achsnummern
```

12.11.3 An-/Abwahl von Achskopplungen (#ENABLE AX LINK, #DISABLE AX LINK)

Eine Kopplungsgruppe kann mit folgendem NC-Befehl aktiviert werden:

Syntax:

#ENABLE AX LINK [<Kopplungsgruppe>]

oder

#ENABLE AX LINK (Kopplungsgruppe 0, definiert in den Kanalparametern)

oder alternativ

#AX LINK ON [<Kopplungsgruppe>]

oder

#AX LINK ON (Kopplungsgruppe 0, definiert in den Kanalparametern)

Eine aktive Kopplungsgruppe kann mit folgenden NC-Befehlen deaktiviert werden:

#DISABLE AX LINK [<Kopplungsgruppe>]

oder

#DISABLE AX LINK (Abwahl der zuletzt aktivierten Kopplungsgruppe)

oder alternativ

#AX LINK OFF [<Kopplungsgruppe>]

oder

#AX LINK OFF (Abwahl der zuletzt aktivierten Kopplungsgruppe)
#AX LINK OFF ALL (Abwahl aller aktiven Kopplungsgruppen)

Handhabung und Wirkungsweise:

- Nach dem Hochlauf ist in der Grundstellung des NC-Kerns keine Kopplungsgruppe aktiv. Die Aktivierung von Achskopplungen beginnt mit der Programmierung im NC-Programm und endet, wenn keine Abwahl erfolgt, mit Programmende (M30, M02). Bei entsprechender Parametrierung mit P-CHAN-00105 können noch aktive Achskopplungen auch über das Programmende hinaus, d.h. programmübergreifend wirksam bleiben.
- Es können mehrere Kopplungsgruppen gleichzeitig aktiviert sein.
- Nicht belegte Kopplungsgruppen können nicht aktiviert werden. Eine Kopplungsgruppe gilt dann als belegt, wenn mindestens eine zulässige Master-Slave-Kopplung definiert wurde.
- Der NC-Befehl muss alleine im NC-Satz stehen.
- Die Nummer der Kopplungsgruppe kann auch über mathematische Ausdrücke programmiert werden.
- Bei An- oder Abwahl des Synchronbetriebs darf die WRK nicht angewählt sein.
- Bei Anwahl des Synchronbetriebs darf der Handbetrieb mit paralleler Interpolation (G201) für die Slaveachsen nicht aktiv sein.
- Bei aktivem Synchronbetrieb können die Slaveachsen im NC-Programm nicht angesprochen werden.

Programmierbeispiel

Verwendete Achsbezeichnungen: Masterachssystem X, Y, Z, C
Slaveachssystem Y_S, Z_S, C_S

```
(Initialisierungsprogramm)
L UP_INIT_ACHS_KOPPL
(Achskopplung 1 initialisieren)
N10 #SET AX LINK[1, Y_S=Y, Z_S=Z, C_S=C]
(oder #AX LINK[1, Y_S=Y, Z_S=Z, C_S=C]
N20 M17
(Werkzeugwechselprogramm)
L UP_WZ
N30 #DISABLE AX LINK (oder #AX LINK OFF)
(Werkzeugwechselposition anfahren)
N40 G01 G90 Y1000 Z100 C0 Y_S=1000 Z_S=100 C_S=0
(Werkzeugwechsel; T10 enthält bereits alle Werkzeugachsversätze und
Werkzeuglänge von Master- und Slavewerkzeug oder diese werden explizit
eingerechnet. )
N50 T10 D10
.....................
(Weitere Befehle für physikalischen Werkzeugwechsel)
...................
(Alte Koppelposition anfahren; die Koppelposition kann auch über Parame-
terprogrammmierung festgelegt und dann vom Unterprogramm verwendet wer-
den.)
N80 G01 G90 X20 Y20 Z40 C50 Y_S=20 Z_S=40 C_S=50
N90 #ENABLE AX LINK[1] (oder #AX LINK ON[1])
N110 M17

(Unterprogramm für Konturbearbeitung)
%L UP1
N150 G01 G91 X10 Y10 Z-20 C90
N160 G02 X20 Y20 I10 J10
N170 LL UP_WZ
N180 G01 G91 X10 Y10 Z-20 C90
N190 G02 X20 Y20 I10 J10
N200 M17


(Hauptprogramm; Ausgangsbed.: Beide Werkzeuge sind eingewechselt.)
(Zunächst beide Achssysteme auf Koppelposition fahren.)
N300 G01 G91 X20 Y20 Z40 C50 Y_S=20 Z_S=40 C_S=50 F300
(Synchronbetrieb starten)
N310 #ENABLE AX LINK[1] (oder #AX LINK ON[1])
N320 LL UP1
.....................................
.....................................
N400 #DISABLE AX LINK (oder #AX LINK OFF)
N410 M30
```

12.11.4 Abfrage von Kopplungszustand/ Kopplungsnummer über Variablen

Über Variablen im NC-Programm kann der aktuelle Kopplungszustand und die aktuell bzw. zuletzt aktive Kopplungsgruppe abgefragt werden. Dazu werden folgende achsgruppenspezifischen Variablen neu eingeführt:

```
Syntax Abfrage der aktuellen bzw. zuletzt aktiven Kopplungsgruppe:
```
V.G.AX_LINK.NR

```
Syntax Abfrage des aktuellen Kopplungszustandes:
```
V.G.AX_LINK.ACTIVE

Bei diesen Variablen sind nur Lesezugriffe zulässig. Schreiboperationen führen zu einer Fehlermeldung mit Programmabbruch.

12.12 Meldungen aus dem NC-Programm

Mit dieser Funktionalität können Meldungen aus dem NC-Programm heraus an beliebige Systemteilnehmer abgesetzt werden. Diese Funktionalität entspricht der von Betriebssystemen her bekannten "printf"-Funktion.

Meldungen im NC-Programm können z.B. zur Anzeige für den Bediener (Beobachter) programmiert werden und ihm Hinweise geben, in welchem Zustand sich der Prozess oder die Steuerung im Moment befindet. Die Meldungsanzeige kann deshalb wahlweise mit dem aktuellen Bearbeitungszustand synchronisiert werden. D.h., die Anzeige der Meldung erfolgt erst dann, wenn der Interpolator die Stelle im NC-Programm erreicht hat.

Ein Meldungstext kann aus

- ASCII-Zeichen,
- Parametern und
- Variablen

zusammengesetzt werden.

Eine Meldung kann nicht zusammen mit anderen NC-Befehlen im gleichen Satz programmiert werden (Ausnahme #ADD).

Verschickt wird die Meldung im ASCII-Format.

12.12.1 Programmierung einer Meldung (#MSG)

Syntax:

#MSG [<*Modus*>] [<*Empfänger*>] **["**<*Meldungstext*>**"]**

<*Modus*>

Der Modus legt den Zeitpunkt der Meldungsausgabe fest. Soll die Meldung synchron zum Bearbeitungsstand im Interpolator ausgegeben werden, so ist der Modus **SYN** bzw. **SYN_ACK** anzugeben. Ohne Angabe des Modus oder mit **ACK** wird die Meldung direkt nach der Decodierung ausgegeben.

---: Direkt nach der Decodierung (Standard).

SYN: Synchron zum Bearbeitungsstand im Interpolator.

SYN_ACK: Synchron zum Bearbeitungsstand im Interpolator mit Quittierung durch den Empfänger (z.B. SPS). Der Interpolator setzt erst nach Empfang der Quittierung die Bearbeitung fort.

ACK: Direkt nach der Decodierung mit Quittierung durch den Empfänger (z.B. SPS). Die Decodierung wird erst nach Empfang der Quittierung fortgesetzt.

<*Empfänger*>

Der Empfänger der Meldung wird durch seine Empfänger-ID angegeben. Ohne Angabe des Empfängers wird der Standardempfänger verwendet.

ISG_DIAG_BED: Meldung geht an CNC-Diagnoseoberfläche (Standard)

DIAG: Meldung geht an Diagnose-Daten, siehe P-CHAN-00514

HMI: Meldung geht an systemspezifische Bedienoberfläche

PLC: Meldung geht an systemspezifische SPS

EVENT_LOGGER: Meldung geht an TwinCAT-Eventlogger

Achtung

Meldungen müssen beim jeweiligen Empfänger abgeholt werden, ansonsten stoppt die CNC nach dem Versenden von 10 Meldungen.

<Meldungstext> Der Meldungstext ist in Anführungszeichen "..." einzuschließen.

Programmierbeispiel

Programmierung einer Meldung

```
#MSG HMI ["Text_1"]

#MSG SYN HMI ["Text_2"]

#MSG ["Text_3"] (Nachricht an den Standardempfänger)

#MSG SYN ["Text_4"]

#MSG SYN_ACK ["Text_5"]
```

Text_1 wird direkt nach der Decodierung und String Text_2 synchron zur Bearbeitung im Interpolator an die systemspezifische Bedienoberfläche verschickt.

Text_3 wird direkt nach der Decodierung und Text_4 synchron zur Bearbeitung im Interpolator an den Standardempfänger verschickt.

Text_5 wird ebenfalls synchron zur Bearbeitung im Interpolator an den Standardempfänger verschickt, die Bearbeitung wird jedoch bis zum Empfang der Quittierung angehalten.

Folgende Formatelemente zur Ausgabe von Zahlenwerten bzw. zur Ausgabe von Strings stehen zur Verfügung:

%d oder **%D**	Ausgabe von Dezimalzahlen mit Vorzeichen (signed integer)
%u oder **%U**	Ausgabe von Dezimalzahlen ohne Vorzeichen (unsigned integer)
%f oder **%F**	Ausgabe von Realzahlen (double)
%s oder **%S**	Ausgabe von Strings (<String>, Makros, V.E.STRING)

Hinweis

Es können maximal 10 Parameter mit %xx ausgegeben werden. Die Anzahl der Formatzeichen muss mit der Anzahl der nachfolgenden Parameter übereinstimmen.

Programmierbeispiel

Programmierung einer Meldung mit Formatelementen:

```
#MSG [ "Meldungstext_%d und Meldungstext_2", 1 ]
```

Gesendeter String: Meldungstext_1 und Meldungstext_2

```
#MSG [ "Aktueller Messwert: %f", V.A.MEAS.ACS.VALUE.X]
```

Gesendeter String: Aktueller Messwert: 3.4567800000E+001

```
#MSG [ "Aktueller Zustand: %s", "Schruppen beendet"]
```

Gesendeter String: Aktueller Zustand: Schruppen beendet

Zur Ausgabe des "%"-Zeichens muss die Steuerfolge "%%" programmiert werden. Für ein Hochkomma muss das "\"-Zeichen vorangestellt werden.

Programmierbeispiel

Programmierung einer Meldung mit "%"-Zeichen und Hochkomma:

```
#MSG [ "Text mit %%-Angabe"]
#MSG [ "Text in Hochkomma: \"TEXT\" "]
```

Gesendeter String: Text mit %-Angabe und Text in Hochkomma: "TEXT"

Ein Meldungstext kann auch mit Parametern und Variablen angegeben werden.

Programmierbeispiel

Programmierung einer Meldung mit Parametern und Variablen:

```
P10 = 1
V.P.BSP = 2
#MSG SYN ["Text_%D und Text_%D", P10, V.P.BSP]
```

In diesem Beispiel wird der String Text_1 und Text_2 synchron zur Bearbeitung im Interpolator an den Standardempfänger verschickt.

12.12.2 Programmierung der Meldungsinformation (#MSG INFO)

Benutzerspezifische Informationen und Anzeigeparameter für die Meldungsverarbeitung können mit dem folgenden Befehl programmiert werden. Die Informationen und Parameter werden ebenfalls als ASCII-String verschickt.

Syntax:

#MSG INFO [<*Modus*>] [<*Empfänger*>] **["**<*Meldungsinformation*>**"]**

Im Vergleich zur Meldung, bei der der Meldungsempfänger den ASCII-String in der Regel nur anzuzeigen hat, zeigt die Meldungsinformation an, dass der ASCII-String eine benutzerspezifisch codierte Information darstellt, die der Empfänger weiterzuverarbeiten hat.

Soll die Meldungsinformation synchron zum Bearbeitungsstand im Interpolator an den Empfänger geschickt werden, so ist der Modus SYN anzugeben. Ohne Modusangabe wird die Meldung direkt nach der Decodierung ausgegeben (siehe vorheriges Kapitel).

Bei der Angabe des Empfängers gelten die gleichen Bedingungen wie beim #MSG-Befehl (siehe vorheriges Kapitel).

Die Meldungsinformation ist in Anführungszeichen "..." einzuschließen. Zur Ausgabe von Zahlenwerten stehen die gleichen Formatelemente wie beim #MSG-Befehl zur Verfügung (siehe vorheriges Kapitel).

Programmierbeispiel

Programmierung der Meldungsinformation

```
P1 = 20
#MSG INFO ["ROT,%d,FETT",P1]
```

In diesem Beispiel wird der String ROT,20,FETT als Meldungsinformation an den Standardempfänger verschickt.

12.12.3 Einbeziehung der Funktionalität 'Makros'

Versionshinweis

Die Verfügbarkeit dieser Funktionalität ist von der Konfiguration und dem Versionsumfang abhängig.

Zur programmübergreifenden Verwaltung von Meldungen können diese als Makro definiert werden. Dies ist vor allem sinnvoll für Meldungen, die mehrfach oder von mehreren Programmen verwendet werden.

Beispiel eines Makroinhaltes als Meldung in den Kanalparametern [1] [▶ 894]**:**

```
makro_def[1].symbol Meldung_1
makro_def[1].nc_code #MSG ["Meldungstext"]
```

Dadurch kann eine Meldung im NC-Programm mit folgendem Makroaufruf abgesetzt werden:

```
:
"Meldung_1"              (Der String "Meldungstext" wird an den)
                         (Defaultempfänger für Meldungen geschickt)
:
```

12.12.4 Schreiben von Meldungen in eine Datei (#MSG SAVE)

Mit dem Befehl #MSG SAVE werden Daten aus dem NC-Programm direkt auf einem Speichermedium (z.B. Festplatte) in einer Datei gespeichert. Die Möglichkeiten zum Aufbau des Meldungstextes und der Protokollierung von Daten stimmen hierbei in vollem Umfang mit denen des Befehls #MSG bzw. #MSG INFO überein.

Syntax:

#MSG SAVE [**EXCLUSIVE**] [**CONTINUE**] [**"**<*Meldungstext*>**"**]

Die Angabe eines Modus (SYN) oder eines Empfängers (z.B. HMI) ist nicht notwendig und führt zur Ausgabe einer Fehlermeldung, da die Meldung direkt im NC-Programm nach der Auswertung in die Ausgabedatei geschrieben wird.

Der Meldungstext ist in Anführungszeichen "..." einzuschließen. Bei jedem Aufruf von #MSG SAVE wird der Meldungstext bei bereits bestehender Ausgabedatei an das Dateiende angefügt. Um eine neue Ausgabedatei zu beginnen, sollte der Anwender deshalb vor Programmstart eine bereits vorhandene Datei löschen.

Mit #MSG SAVE ["Meldungstext"] wird der Meldungstext in der Ausgabedatei in der Form

< Zeitstempel > “Meldungstext“

geschrieben.

Mit #MSG SAVE EXCLUSIVE ["Meldungstext"] wird der Meldungstext in der Ausgabedatei **ohne** Zeitstempel in der Form

“Meldungstext“

geschrieben.

Mit #MSG SAVE CONTINUE ["Meldungstext"] kann ein Zeilenumbruch am Ende des Meldungstextes unterdrückt werden. Der nächste #MSG SAVE-Befehl setzt dann in derselben Zeile wie der vorherige #MSG SAVE-Befehl auf.

Der Dateiname kann zuvor über den NC-Befehl #FILE NAME [▶ 438] festgelegt werden. Wird von dieser Möglichkeit nicht Gebrauch gemacht, schreibt #MSG SAVE automatisch in eine Datei mit dem Namen **message.txt** .

Der Pfad zur Ausgabedatei wird in der Hochlaufliste mit P-STUP-00018 und über den Pfadtyp P-STUP-00020 festgelegt. Ist hier kein Pfad für die Ausgabedatei markiert, wird abhängig von der Steuerungsplattform ein Defaultpfad verwendet bzw. die Ausgabedatei im Hauptverzeichnis der NC-Steuerung angelegt.

Programmierbeispiel

Schreiben von Meldungen in eine Datei

```
:
#FILE NAME [MSG="example.txt"]    ;Name der Ausgabedatei
:
#MSG SAVE["Hello World            ;Text "Hello World 12345" in Ausgabedatei exam-
%d",12345]                        ple.txt schreiben
:
```

12.12.5 Ausgabe von Zusatzinformationen am Satzende (#ADD)

Mit dem Befehl #ADD können im NC-Satz Zusatzinformationen erzeugt werden. Die Möglichkeiten zum Aufbau dieser Zusatzinformationen (Meldungstexte) stimmen hierbei in vollem Umfang mit denen der #MSG-Befehle überein. Im Gegensatz zu den #MSG-Befehlen dürfen jedoch **vor** #ADD andere NC-Befehle programmiert werden. Somit muss #ADD stets als letzter Befehl am NC-Satzende programmiert sein. Nachfolgende Kommentare sind erlaubt.

Syntax:

#ADD [<*Empfänger*>] **["**<*Zusatzinformationen*>**"]**

Die Angabe eines Modus (SYN) ist nicht notwendig und führt zur Ausgabe einer Fehlermeldung, da die Meldung automatisch immer synchron zum Bearbeitungsstand im Interpolator ausgegeben wird.

Bei der Angabe des Empfängers gelten die gleichen Bedingungen wie bei den #MSG-Befehlen.

Die Zusatzinformationen sind in Anführungszeichen "..." einzuschließen.

Programmierbeispiel

Ausgabe von Zusatzinformationen am Satzende

```
%add_block_info
N05 P1=20
N10 G00 X0 Y0 Z0
N15 T1             #ADD["Werkzeug T=%d aktiv", V.G.T_AKT]
N20 G01 F2000 X10 #ADD["X-Position anfahren"] (Kommentar)
N30 YP1            #ADD["Y-position=%d", P1]
N40 Z30            #ADD["Z-position"]
N50 Z33            #ADD["Z-position"] X11 Y22 ->Fehler 21509
N999 M30
```

12.13 Ruckbegrenzender Slope

Der Slope bestimmt die Geschwindigkeit auf der programmierten Bahn unter Einhaltung der vorgegebenen zulässigen Geschwindigkeiten, Beschleunigungen und Rücke [2] [▶ 894]-1. Folgende Modi sind verfügbar:

- sprungförmiges Beschleunigungsprofil mit Begrenzung der Beschleunigung ohne Überwachung des Rucks,
- trapezförmiges Beschleunigungsprofil mit Rucküberwachung,
- sinusquadratförmiges Beschleunigungsprofil mit Rucküberwachung.

Je nach Wahl der Slope-Funktion stellt sich der Beschleunigungsverlauf ein:

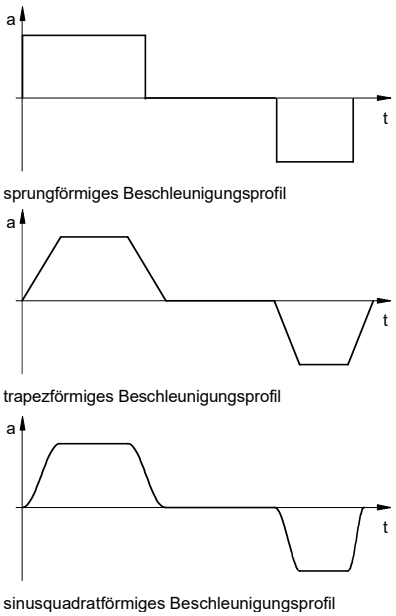


Abb. 106: Beschleunigung auf der programmierten Bahn

Das Beschleunigungsprofil wird über Beschleunigungen und Rampenzeiten achsspezifisch parametriert [2] [▶ 894]-1:

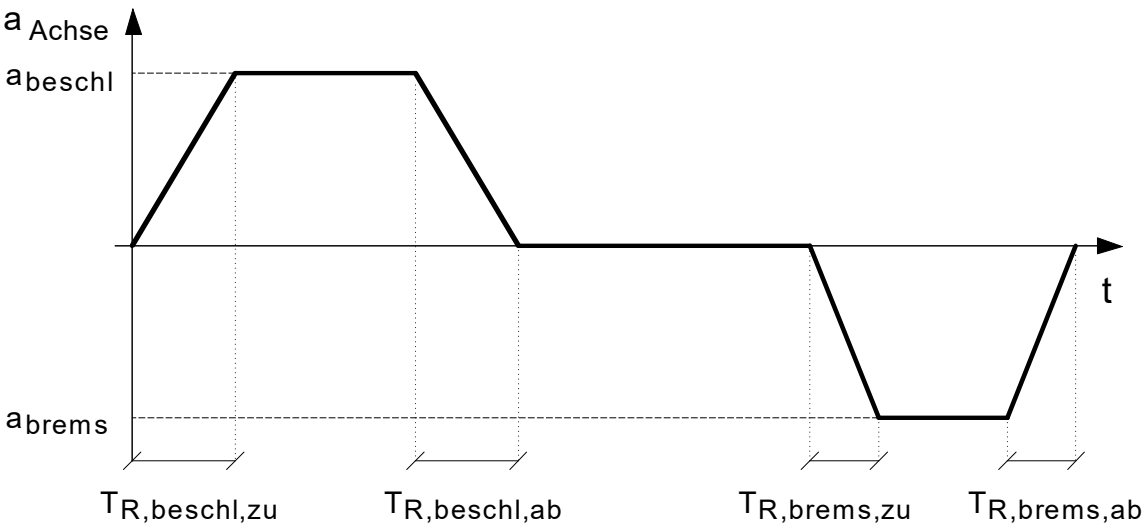


Abb. 107: Parameter des Beschleunigungsprofils

12.13.1 Wahl des Betriebsmodus (#SLOPE, #SLOPE DEFAULT)

Versionshinweis

Ab Version **V2.11.2010.02** ersetzt der Befehl **#SLOPE [...]** den Befehl **#SET SLOPE PROFIL [...]**. Dieser ist aus Kompatibilitätsgründen weiterhin verfügbar, es wird aber empfohlen, diesen in neuen NC-Programmen nicht mehr zu verwenden.

Syntax:

#SLOPE [TYPE=<*ident*> [**NO_OPT=..**] **]**

TYPE<*ident*>	Art des Beschleunigungsprofils. Zulässige Kennungen:
	STEP: sprungförmiges Beschleunigungsprofil (Grundzustand)
	TRAPEZ: trapezförmiges Beschleunigungsprofil
	SIN2: sinusquadratförmiges Beschleunigungsprofil
	HSC: HSC-Slope, empfohlen für "Erweiterte HSC-Programmierung" [▶ 252] *
NO_OPT**=..**	Schalten der optimierten Ruckausnutzung:
	0: Optimierte Ruckausnutzung ist wirksam. Dadurch kann die Bearbeitungszeit verkürzt werden, es ist jedoch ein höherer Rechenaufwand erforderlich. Es muss geprüft werden, ob die vorhandene Steuerungshardware geeignet ist.
	1: Optimierte Ruckausnutzung ist **nicht** wirksam (Grundzustand).

Hinweis

* Die Nutzung dieser Funktionalität zur Anwahl des Profiltyps HSC-Slope erfordert die Lizenzierung des Erweiterungspaketes "HSC". Sie ist nicht im Umfang der Standardlizenz enthalten.

Hinweis

Die spezifische Anpassung der Gewichtungswirkung von Rampenzeit (G132/G133) bzw. Beschleunigung (G130/G131) wird durch #SLOPE [...] nicht mehr unterstützt. Die Gewichtungen wirken immer auf alle Rampenzeiten bzw. Beschleunigungen (Grundzustand).

Syntax:

#SLOPE DEFAULT

Die Programmierung von #SLOPE DEFAULT stellt den Grundzustand (wie nach Hochlauf) wieder her, d.h. es wird der Slopetyp aus dem Kanalparametersatz P-CHAN-00071 gesetzt.

Programmierbeispiel

Wahl des Betriebsmodus

```
N10 G01 X50 Y10 Z0 F1000        (sprungförmiges Beschl.profil, Default)
N20 #SLOPE [TYPE=TRAPEZ]        (trapezförmiges Beschl.profil)
N30 X10 Y30
N40 #SLOPE [TYPE=SIN2]          (sinusquadratförmiges Beschl.profil)
N50 X15
N60 Y50
N70 M30
```

Auf der programmierten Bahn ergibt sich folgender Geschwindigkeitsverlauf:

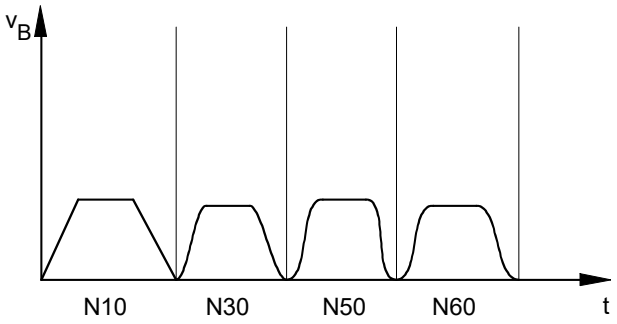


Abb. 108: Geschwindigkeitsverlauf auf der programmierten Bahn.

12.14 Schreiben und Lesen von Antriebsparametern und Kommandos

12.14.1 Antriebsparameter (#IDENT)

Die folgenden NC-Befehle dienen zur Beschreibung bzw. zum Lesen von SERCOS-Parametern. Zur einfacheren Programmierung wird das original SERCOS-Format [4] [▶ 894] verwendet.

Ab CNC-Version V3.1.3081.4 können Antriebsparameter für CANopen gelesen und beschrieben werden.

Zur korrekten Verarbeitung der Parameter-IDENT-Nummer im Antrieb sind noch weitere Angaben, wie z.B. Achsname bzw. logische Achsnummer, Kennungen und ein Attribut erforderlich. Diese werden zusammen mit der IDENT-Nummer im gleichen NC-Befehl programmiert.

Hinweis

„Nicht synchron“ bedeutet die Ausführung des Befehles im Decodierkontext.

„Synchron“ bedeutet die Ausführung des Befehles synchron zur Bearbeitung, d.h. auf Interpolatorebene.

12.14.1.1 Nicht synchronisiertes Schreiben (#IDENT WR)

Syntax:

#IDENT WR [AX=<*Achsname*> | **AXNR**=.. **ID=**<*Ident_nr*> **VAL**=.. **TYP**=.. **DEC=**.. <*Drive_type*> **]**

AX=<*Achsname*>	Name der Achse
AXNR=..	Logische Achsnummer der Achse, Positive Ganzzahl
ID=<*Ident_nr*>	Identnummer im SERCOS-Format, z.B. *S-0-0047* oder *P-0-0129* bzw. im CANopen-Format, z.B. *0x6072_00*
VAL=..	Zu schreibender Wert; Realzahl
TYP=..	Datentyp des Wertes (2 oder 4 Byte Länge): 2: 2 Byte Datenlänge 4: 4 Byte Datenlänge
DEC=..	Anzahl der Nachkommastellen; Positive Ganzzahl
<*Drive_type*>	Antriebstyp SERC: SERCOS-Antrieb CAN: CANopen-Antrieb (ab V3.1.3081.4)

Programmierbeispiel

Nicht synchronisiertes Schreiben (IDENT)

```
:
(SERCOS-Antrieb)
#IDENT WR [AX X ID S-0-0104 VAL 655.35 TYP 4 DEC 2 SERC]
#IDENT WR [AXNR 1 ID S-0-0104 VAL 655.35 TYP 4 DEC 2 SERC]

(CANopen-Antrieb)
#IDENT WR [AX X ID 0x6072_00 VAL 655.35 TYP 4 DEC 2 CAN]
#IDENT WR [AXNR 1 ID 0x6072_00 VAL 655.35 TYP 4 DEC 2 CAN]
:
```

Achtung

Es erfolgt keine Plausibilitätsprüfung bzgl. logischer Achsnummer, Identnummer und den programmierten Attributen Datentyp und Nachkommastellen. Für die richtige Angabe ist ausschließlich der Bediener verantwortlich.

12.14.1.2 Nicht synchronisiertes Lesen (#IDENT RD)

Syntax:
#IDENT RD [AX=<*Achsname*> | **AXNR**=.. **ID=**<*Ident_nr*> **P**=<*Variable*> **TYP**=.. **DEC**=.. <*Drive_type*> **]**

AX=<*Achsname*>	Name der Achse
AXNR=..	Logische Achsnummer der Achse, Positive Ganzzahl
ID=<*Ident_nr*>	Identnummer im SERCOS-Format, z.B. *S-0-0047* oder *P-0-0129* bzw. CANopen-Format, z.B. *0x6072_00*
<*Variable*>	Variable in die der zu lesende Wert abgelegt wird. z.B. P-Parameter oder V.P.-, V.L.- oder V.S.-Variablen.
TYP=..	Datentyp des Wertes (2 oder 4 Byte Länge): 2: 2 Byte Datenlänge 4: 4 Byte Datenlänge
DEC=..	Anzahl der Nachkommastellen; Positive Ganzzahl
<*Drive_type*>	Antriebstyp SERC: SERCOS-Antrieb CAN: CANopen-Antrieb (ab V3.1.3081.4)

Programmierbeispiel

Nicht synchronisiertes Lesen (IDENT)

```
:
(SERCOS-Antrieb)
#IDENT RD [AX X ID S-0-0104 P=P1 TYP 4 DEC 2 SERC]
#IDENT RD [AXNR 1 ID S-0-0104 P=V.P.KV_WERT TYP 4 DEC 2 SERC]

(CANopen-Antrieb)
#IDENT RD [AX X ID 0x6072_00 P=P1 TYP 4 DEC 2 CAN]
#IDENT RD [AXNR 1 ID 0x6072_00 P=V.P.KV_WERT TYP 4 DEC 2 CAN]
:
```

Achtung

Es erfolgt keine Plausibilitätsprüfung bzgl. logischer Achsnummer, Identnummer und den programmierten Attributen Datentyp und Nachkommastellen. Für die richtige Angabe ist ausschließlich der Bediener verantwortlich.

12.14.1.3 Synchronisiertes Schreiben (#IDENT WR SYN)

Syntax:

#IDENT WR SYN [AX=<*Achsname*> | **AXNR=.. ID=**<*Ident_nr*> **VAL=.. TYP=.. DEC=..** <*Drive_type*>
[**NO_WAIT**] **]**

AX=<*Achsname*>	Name der Achse
AXNR=..	Logische Achsnummer der Achse, Positive Ganzzahl
ID=<*Ident_nr*>	Identnummer im SERCOS-Format, z.B. *S-0-0047* oder *P-0-0129*
	bzw. CANopen-Format, z.B. *0x6072_00*
VAL=..	Zu schreibender Wert; Realzahl
TYP=..	Datentyp des Wertes (2 oder 4 Byte Länge):
	2: 2 Byte Datenlänge
	4: 4 Byte Datenlänge
DEC=..	Anzahl der Nachkommastellen; Positive Ganzzahl
<*Drive_type*>	Antriebstyp
	SERC: SERCOS-Antrieb
	CAN: CANopen-Antrieb (ab V3.1.3081.4)
NO_WAIT	Kein Warten auf erfolgreiches Schreiben des Parameters.
	Ohne Angabe dieses Schlüsselwortes wird generell auf die Ausführung des Schreibens des Parameters gewartet.

Programmierbeispiel

Synchronisiertes Schreiben (IDENT)

```
:
(SERCOS-Antrieb)
#IDENT WR SYN [AX=X ID S-0-0104 VAL 655.35 TYP 4 DEC 2 SERC]
#IDENT WR SYN [AXNR=1 ID S-0-0104 VAL 655.35 TYP 4 DEC 2 SERC]
:
#IDENT WR SYN [AX Y ID S-0-0104 VAL655.35 TYP 4 DEC 2 SERC NO_WAIT]

(CANopen-Antrieb)
#IDENT WR SYN [AX=X ID 0x6072_00 VAL 655.35 TYP 4 DEC 2 CAN]
#IDENT WR SYN [AXNR=1 ID 0x6072_00 VAL 655.35 TYP 4 DEC 2 CAN]
#IDENT WR SYN [AX Y ID 0x6072_00 VAL655.35 TYP 4 DEC 2 CAN NO_WAIT]
:
```

Achtung

Es erfolgt keine Plausibilitätsprüfung bzgl. logischer Achsnummer, Identnummer und den programmierten Attributen Datentyp und Nachkommastellen. Für die richtige Angabe ist ausschließlich der Bediener verantwortlich.

12.14.2 SERCOS-Kommandos (COMMAND)

Die folgenden NC-Befehle dienen zum Starten bzw. Warten auf die Ausführung von SERCOS-Kommandos. Zur einfacheren Programmierung wird das originale SERCOS-Format [4] [▶ 894] verwendet.

Zur korrekten Verarbeitung der Kommando-IDENT-Nummer im Antrieb sind noch weitere Angaben, wie Achsname bzw logische Achsnummer und Kennung erforderlich. Diese werden zusammen mit der IDENT-Nummer im gleichen NC-Befehl programmiert.

Hinweis

„Nicht synchron“ bedeutet die Ausführung des Befehles im Decodierkontext.

„Synchron“ bedeutet die Ausführung des Befehles synchron zur Bearbeitung, d.h. auf Interpolatorebene.

12.14.2.1 Nicht synchronisiertes Schreiben (#COMMAND WR)

Syntax:

#COMMAND WR [AX=<*Achsname*> | **AXNR**=.. **ID**=<*Ident_nr*> <*Drive_type*> **]**

AX=<*Achsname*>	Name der SERCOS-Achse
AXNR=..	Logische Achsnummer der SERCOS-Achse, Positive Ganzzahl
ID=<*Ident_nr*>	Identnummer des Kommandos im SERCOS-Format, z.B. *S-0-0148* (Antriebsgeführtes Referenzieren) oder *S-0-0170* (Messtasterzyklus)
<*Drive_type*>	Antriebstyp **SERC:** SERCOS-Antrieb (momentan als einziges zulässig)

Programmierbeispiel

Nicht synchronisiertes Schreiben (COMMAND)

```
:
#COMMAND WR [AX=X ID S-0-0148 SERC]

#COMMAND WR [AXNR 1, ID S-0-0148, SERC]
:
```

Achtung

Es erfolgt keine Plausibilitätsprüfung bzgl. logischer Achsnummer und Identnummer. Für die richtige Angabe ist ausschließlich der Bediener verantwortlich.

12.14.2.2 Synchronisiertes Schreiben (#COMMAND WR SYN)

Syntax:

#COMMAND WR SYN [AX=<*Achsname*> | **AXNR=.. ID**=<*Ident_nr*> <*Drive_type*> **]**

AX=<*Achsname*>	Name der SERCOS-Achse
AXNR=..	Logische Achsnummer der SERCOS-Achse, Positive Ganzzahl
ID=<*Ident_nr*>	Identnummer des Kommandos im SERCOS-Format, z.B. *S-0-0148* (Antriebsgeführtes Referenzieren) oder *S-0-0170* (Messtasterzyklus)
<*Drive_type*>	Antriebstyp SERC: SERCOS-Antrieb (momentan als einziges zulässig)

Programmierbeispiel

Synchronisiertes Schreiben (COMMAND)

```
:
#COMMAND WR SYN [AX Y, ID S-0-0170, SERC]

#COMMAND WR SYN [AXNR 2 ID S-0-0170 SERC]
:
```

Achtung

Es erfolgt keine Plausibilitätsprüfung bzgl. logischer Achsnummer und Identnummer. Für die richtige Angabe ist ausschließlich der Bediener verantwortlich.

12.14.2.3 Nicht synchronisiertes Warten (#COMMAND WAIT)

Syntax:

#COMMAND WAIT [AX=*<Achsname>* | **AXNR=.. ID**=*<Ident_nr>* *<Drive_type>* **]**

AX=*<Achsname>*	Name der SERCOS-Achse
	ALL: Prüfung aller im System vorhandener SERCOS-Achsen
AXNR=..	Logische Achsnummer der SERCOS-Achse, Positive Ganzzahl
ID=*<Ident_nr>*	Identnummer des Kommandos im SERCOS-Format,
	z.B. *S-0-0148* (Antriebsgeführtes Referenzieren)
	oder *S-0-0170* (Messtasterzyklus)
	Ist keine Identnummer angegeben, so wird auf alle offenen Kommandos gewartet.
<Drive_type>	Antriebstyp
	SERC: SERCOS-Antrieb (momentan als einziges zulässig)

Programmierbeispiel

Nicht synchronisiertes Warten (COMMAND)

```
...
#COMMAND WAIT [AX X, ID S-0-0148, SERC]
:
#COMMAND WAIT [AX ALL ID S-0-0148 SERC]
...
```

Achtung

Es erfolgt keine Plausibilitätsprüfung bzgl. logischer Achsnummer und Identnummer. Für die richtige Angabe ist ausschließlich der Bediener verantwortlich.

12.14.2.4 Synchronisiertes Warten (#COMMAND WAIT SYN)

Syntax:

#COMMAND WAIT SYN [AX=<*Achsname*> | **AXNR**=.. **ID**=<*Ident_nr*> <*Drive_type*> **]**

AX=<*Achsname*>	Name der SERCOS-Achse ALL: Prüfung aller im System vorhandener SERCOS-Achsen
AXNR=..	Logische Achsnummer der SERCOS-Achse, Positive Ganzzahl
ID=<*Ident_nr*>	Identnummer des Kommandos im SERCOS-Format, z.B. S-0-0148 (Antriebsgeführtes Referenzieren) oder S-0-0170 (Messtasterzyklus). Ist keine Identnummer angegeben, so wird auf alle offenen Kommandos gewartet.
<*Drive_type*>	Antriebstyp SERC: SERCOS-Antrieb (momentan als einziges zulässig)

Programmierbeispiel

Synchronisiertes Warten (COMMAND)

```
...
#COMMAND WAIT SYN [AX=X, ID=S-0-0148, SERC]
:
#COMMAND WAIT SYN [AX ALL, ID S-0-0148, SERC]
...
```

Achtung

Es erfolgt keine Plausibilitätsprüfung bzgl. logischer Achsnummer und Identnummer. Für die richtige Angabe ist ausschließlich der Bediener verantwortlich.

Hinweis

Zum Warten auf ein SERCOS-Kommando wird die Bewegung nicht zwangsläufig angehalten. Ist das Kommando am Ende des Bewegungssatzes aber noch nicht beendet, so werden keine weiteren NC-Sätze bearbeitet und somit angehalten.

Programmierbeispiel

Am Ende von Satz N120 wird gewartet bis das Kommando S-0-0148 beendet ist.

```
..
N100 #COMMAND WR SYN [AX Y ID S-0-0148 SERC]
N110 G01 X1000 F100
N120 #COMMAND WAIT SYN [AX Y ID S-0-0148 SERC]
N130 G01 X2000
...
```

Achtung

Mit den („WAIT“)-Befehlen können nur Kommandos geprüft werden, welche auch zuvor von der gleichen Ebene (Decodierkontext bzw. synchron zur Bearbeitung auf Interpolatorebene) gestartet wurden. Ein zum Beispiel synchron beauftragtes Kommando („SYN“) kann auch nur synchron auf Interpolatorebene geprüft werden.

Programmierbeispiel

Im Satz N110 wird auf Decodierebene kein aktives Kommando S-0-0148 gefunden und somit auch nicht gewartet, obwohl dieses in Wirklichkeit noch aktiv sein könnte. Erst in Satz N120 kann der wirkliche Status des Kommandos auf Interpolatorebene festgestellt werden.

```
...
N100 #COMMAND WR SYN [AX Y ID S-0-0148 SERC]
N110 #COMMAND WAIT [AX Y ID S-0-0148 SERC]
N120 #COMMAND WAIT SYN [AX Y ID S-0-0148 SERC]
...
```

12.15 Kanalsynchronisation

Beim Betrieb einer mehrkanaligen (besonders bei n > 2) Steuerung können Situationen auftreten, in denen bestimmte Ablaufreihenfolgen zwischen Kanälen zwingend einzuhalten sind.

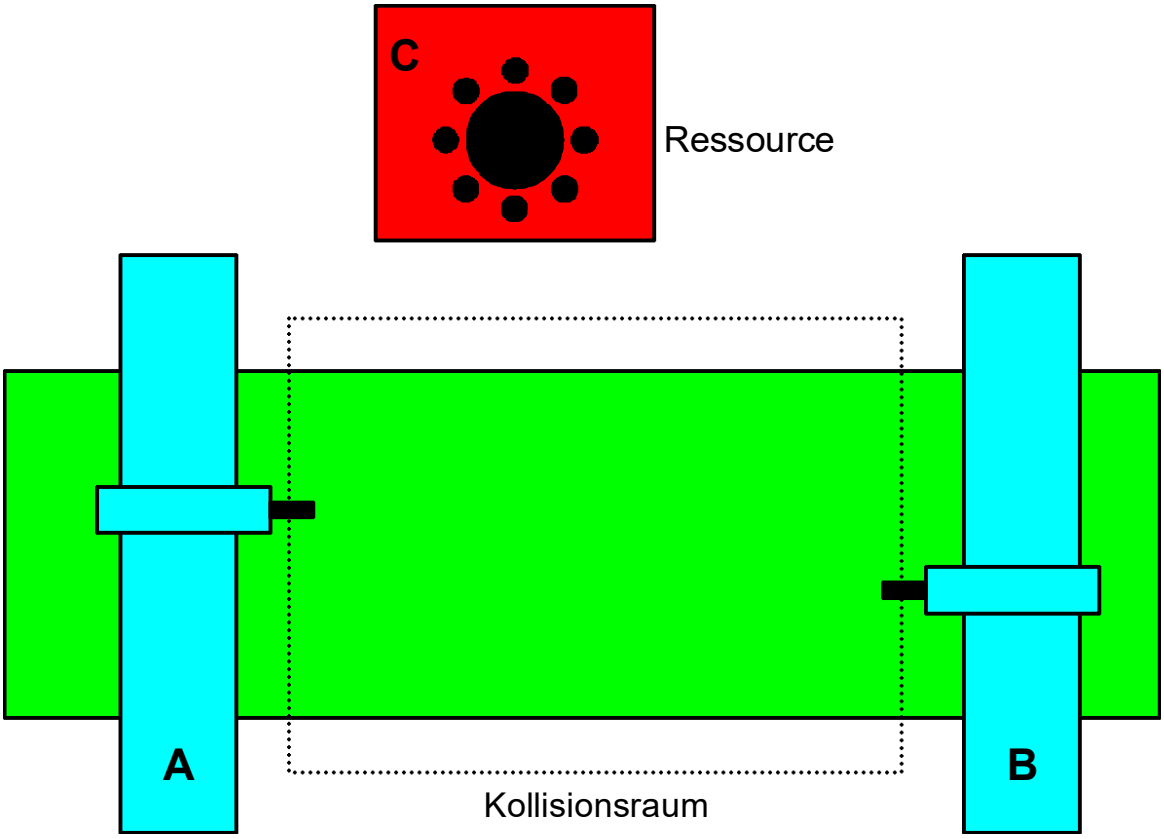


Abb. 109: Anwendungsbeispiel Zweiständermaschine mit Werkzeugwechsler

Das Beispiel in oben stehender Abbildung zeigt einen solchen Anwendungsfall, bei dem sich zwei Bearbeitungseinheiten (A und B) einen gemeinsamen Bearbeitungsraum teilen. Ebenso nutzen beide die Ressource Werkzeugwechsler (C). Um bei einem derartigen Maschinenaufbau Kollisionen zu vermeiden, müssen die NC-Programme der verschiedenen Kanäle der Steuerung an bestimmten Stellen miteinander synchronisiert werden. In obigem Fall darf beispielsweise der Ständer A nicht in den Kollisionsraum einfahren, solange sich Ständer B noch darin befindet. Ebenso darf Ständer B nicht den Werkzeugwechsler benutzen, wenn Ständer A gerade darauf zugreift.

Für den Zugriff auf die Ressource Werkzeugwechsler ergibt sich z.B. der in der folgenden Abbildung dargestellte zeitliche Ablauf in den zwei Kanälen der Steuerung.

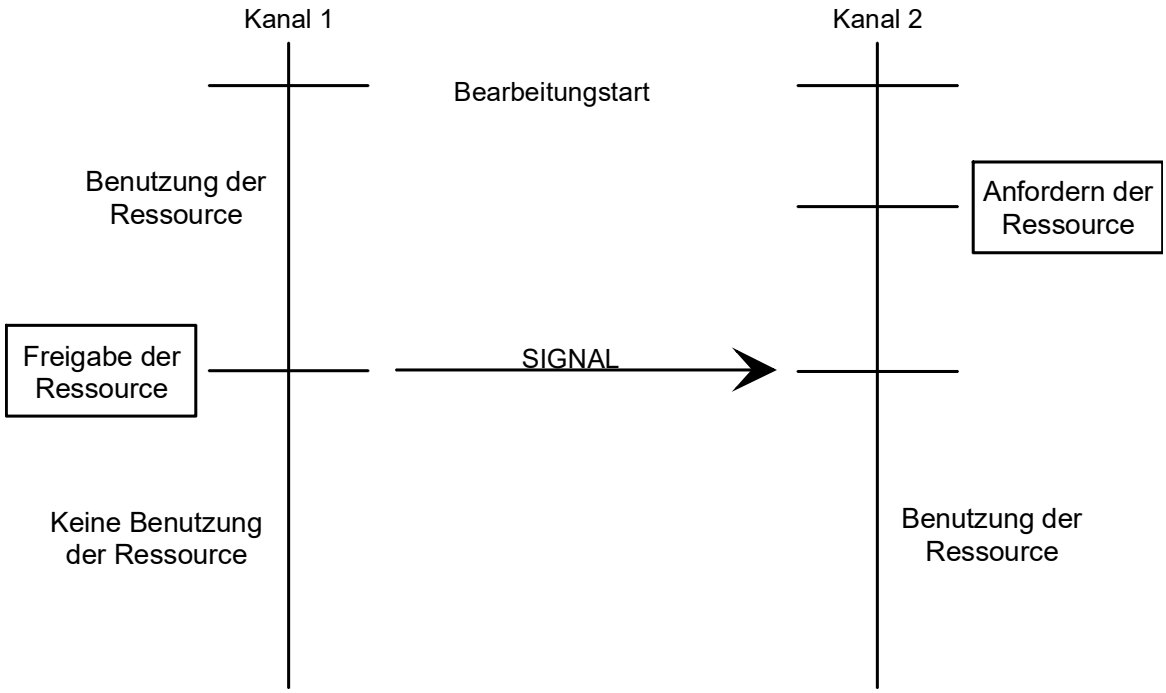


Abb. 110: Ablauf beim gemeinsamen Zugriff auf eine Ressource

Die erforderliche Synchronisation basiert auf dem Versenden und Warten auf Signale und erfolgt über die nachfolgend beschriebenen NC-Befehle in den NC-Programmen.

Werden in einem Kanal sehr viele Signale versendet, welche erst spät in einem anderen Kanal verarbeitet werden, kann es passieren, dass der Speicher für die Anzahl an Synchronisationsereignissen nicht ausreicht.

Ab V3.1.3081.02 bzw. V3.1.3108.02 kann über P-STUP-00119 die Anzahl der zu speichernden Synchronisationsereignisse festgelegt werden. Hierbei stellt jeder Signal- oder Wartebefehl ein Ereignis dar. Wird der Parameter nicht gesetzt, stehen als Standard 150 Ereignisse zur Verfügung. Zusätzlich kann mit P-STUP-00118 festgelegt werden, ob bei Standardsignalen auf Decoder-Ebene die Decodierung auf eine Quittierung beim Speichern des Signalbefehls wartet. Dadurch kann ein Überschreiten der Grenze der Anzahl von Synchronisationsereignissen verhindert werden.

12.15.1 Synchronisationsszenarien

Synchronisation von 2 Decodern in 2 Kanälen

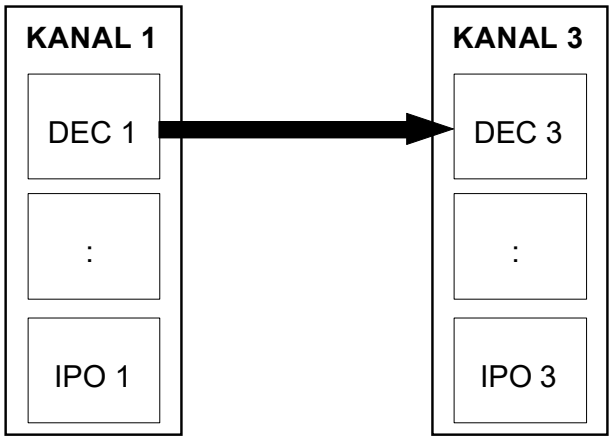


Abb. 111: Synchronisation von 2 Decodern in 2 Kanälen

- Decoder 3 wartet auf Decoder 1, Decoder 1 arbeitet ohne Unterbrechung weiter

Programmierbeispiel

Synchronisation von 2 Decodern in 2 Kanälen

```
% kanal_1
...
(Signal P100)
(Synchronisation auf DEC-Ebene)
(Synchronisation mit Kanal 3)
(Parameter V.P.SYNC)

V.P.SYNC = 1000
P100 = 814

#SIGNAL [IDP100 P[0]= V.P.SYNC CH3]
:
```

```
% kanal_3
...
(Warteanforderung 814)
(Synchronisation auf DEC-Ebene)
(Synchronisation mit Kanal 1)
(Parameter V.P.SIGNAL)

#WAIT [ID814 P[0]= V.P.SIGNAL CH1]
:
```

Synchronisation zwischen Decoder und Interpolatoren in 3 Kanälen

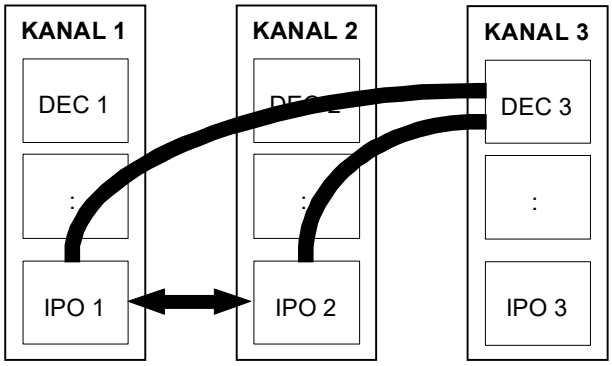


Abb. 112: Synchronisation zwischen Decoder und Interpolatoren in 3 Kanälen

- Interpolator 1 wartet auf Interpolator 2 und Decoder 3,
- Interpolator 2 wartet auf Interpolator 1 und Decoder 3,
- Decoder 3 signalisiert an Interpolator 1 und Interpolator 2.

Programmierbeispiel

Synchronisation zwischen Decoder und Interpolatoren in 3 Kanälen

```
% kanal_1
...
(Warteanforderung 968)
(Sync. auf Interp.-Ebene)
(Sync. mit Kanal 2 und 3)

#WAIT SYN [ID968 CH2 CH3]
:
```

```
%kanal_2
...
(Wartenaforderung 968)
(Sync. auf Interp.-Ebene)
(Sync. mit Kanal 3 und 1)

#WAIT SYN [ID968 CH3 CH1]
:
```

```
% kanal_3
...
(Signal 968)
(Sync. auf Decoder-Ebene)
(Sync. mit Kanal 1 und 2)

#SIGNAL [ID968 CH1 CH2]
:
```

Synchronisation zwischen Interpolatoren in 3 Kanälen

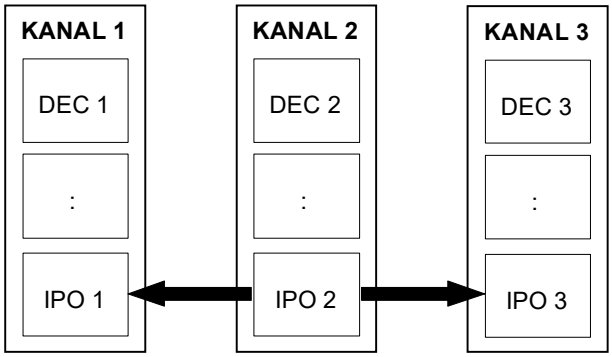


Abb. 113: Synchronisation zwischen Interpolatoren in 3 Kanälen

- Bahn 1 wartet auf Bahn 2,
- Bahn 3 wartet auf Bahn 2,
- Bahn 2 signalisiert an Bahn 1 und Bahn 3.

Programmierbeispiel

Synchronisation zwischen Interpolatoren in drei Kanälen

% kanal_1 ... (Warteanforderung 100) (Sync. auf Interp.-Ebene) (Sync. mit Kanal 2) **#WAIT SYN [ID100 CH2]** :	%kanal_2 ... (Signal 100) (Sync. auf Interp.-Ebene) (Sync. mit Kanal 1 und 3) **#SIGNAL SYN [ID100 CH1 CH3]** :	% kanal_3 ... (Warteanforderung 100) (Sync. auf Interp.-Ebene) (Sync. mit Kanal 2) **#WAIT SYN [ID100 CH2]** :

Synchronisation zwischen Decoder und Interpolator eines Kanals

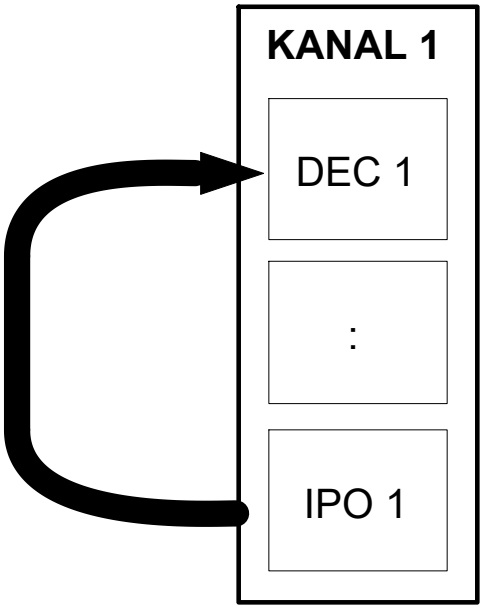


Abb. 114: Synchronisation zwischen Decoder und Interpolator eines Kanals

- Decoder wartet bis Interpolator Position X 250 erreicht hat.
- Bewegungssatz "G01 X370 Z200 F80" ist bereits im NC-Kanal und wird nach Signalisierung abgearbeitet.
- Bewegungssatz "G01 X900" wird erst nach Synchronisation decodiert.

Achtung

Bei Synchronisationsanforderungen zwischen Decoder und Interpolator kann es zu Zuständen kommen, in denen das NC-Programm nicht weiter decodiert werden kann, da noch keine Quittierung eingetroffen ist. Die Quittierung wird vom Interpolator jedoch auch nicht weggeschickt, da der Signalsatz den Interpolator aufgrund der Bufferwirkung des NC-Kanals nicht erreicht. Um mögliche Verklemmungen zu vermeiden, ist in solchen Fällen ein #FLUSH vorzusehen, durch den der NC-Kanal leergeräumt wird.

Programmierbeispiel

Synchronisation zwischen Decoder und Interpolator eines Kanals

```
% kanal_1
G00 X100 Y500
G01 X250 F300
(Signal 88)
(Synchronisation auf Interp.-Ebene, Synchronisation mit Kanal 1)
#SIGNAL SYN [ID88 CH1]
(Bearbeitung)
G01 X370 Z200 F80
(Warteanforderung 88)
(Synchronisation auf Decoder-Ebene, Synchronisation mit Kanal 1)
#FLUSH
#WAIT [ID88 CH1]
G01 X900
:
```

12.15.2 Senden von Signalen (#SIGNAL)

Grundsätzlich wird zwischen Signalen ohne Empfängerangabe (s.g. Broadcast-Signalen) und Standardsignalen, bei denen explizit ein Kanal als Empfänger angegeben ist, unterschieden.

Die Signale sind durch eine eindeutige Nummer identifiziert, wobei das Senden von Signalen mit gleichen Signalnummern erlaubt ist.

Bei Standardsignalen sind explizit ein oder mehrere NC-Kanäle als Empfänger anzugeben. Werden bei einem Signal mehrere Empfänger angegeben, so wirkt dies wie das mehrfache Senden der gleichen Signale an die einzelnen Kanäle.

Beispiel

Senden von Signalen

```
#SIGNAL [ID4711 CH1 CH2 CH3]
```

wirkt wie

```
#SIGNAL [ID4711 CH1]
#SIGNAL [ID4711 CH2]
#SIGNAL [ID4711 CH3]
```

Diese Signale sind nur für ein #WAIT des adressierten Empfängers gültig und werden ohne Angabe des Verbrauchszählers COUNT bei einem #WAIT des Empfängerkanals verbraucht. Wird ein Verbrauchszähler COUNT angegeben, so sind genauso viele #WAIT-Anforderungen bis zum Verbrauchen des Signals möglich.

Im Unterschied dazu können Broadcast-Signale durch ein #WAIT eines beliebigen Kanals empfangen werden.

Wird bei Broadcast-Signalen kein Verbrauchszähler COUNT angegeben, so werden diese durch ein #WAIT nicht verbraucht. D.h. sie bleiben bis zum expliziten Löschen (s. #SIGNAL REMOVE) bestehen. Wird ein Verbrauchszähler COUNT angegeben, so sind, wie bei Standardsignalen, genauso viele #WAIT-Anforderungen bis zum Verbrauchen des Broadcast-Signals möglich.

Syntax:

#SIGNAL [<*Modus*>] **[ID**=.. [**COUNT**=..] { **P[**<*idx*>**]=**<*param*> } { **CH**=.. } [**KEEP_AT_RESET**] **]**

<*Modus*>	Synchronisationsart. Zulässige Kennungen: ---: Synchronisation auf Decoder-Ebene (Grundeinstellung). Diese Synchronisation ist z.B. erforderlich, wenn auf Parameter oder Variablen synchronisiert werden soll. SYN: Synchronisation auf Interpolator-Ebene. Diese Synchronisation ist bei Echtzeitanforderungen notwendig, z.B. Synchronisation zweier Bearbeitungseinheiten einer Mehrständermaschine
ID=..	Signalnummer, muss systemweit eindeutig sein. Positive Ganzzahl.
COUNT=..	Verbrauchszähler, definiert wie oft ein Signal mit #WAIT [▶ 398] abgerufen werden kann. Positive Ganzzahl.
P[<*idx*>] = <*param*>	Signalparameter Beim Senden von Signalen können Signalparameter an den mit #WAIT wartenden Empfänger übergeben werden. Hierzu wird der Realwert <param> dem Signalparameter an Index <idx> zugewiesen. <*idx*>: Bereich für die maximal mögliche Parameteranzahl: 0 .. 11 (max. Anzahl Signalparameter [(1)])

CH=..	Nummer des Kanals, für den das Signal bestimmt ist. 1...max. Anzahl Kanäle (2) Wird keine Kanalnummer angegeben, so wird ein s.g. **Broadcast-Signal** an alle verfügbaren Signalempfänger des Systems gesendet.
KEEP_AT_RESET	Markiertes Signal (Standardsignal mit Empfänger) wird bei Reset nicht entfernt. Damit bleibt das Signal auch nach Reset des sendenden Kanals erhalten, bis es durch ein #WAIT [▶ 398] mit der richtigen ID verbraucht oder durch explizite Programmierung (#SIGNAL REMOVE [▶ 396]) entfernt wird

(1) siehe [6] [▶ 894]-6.45, (2) siehe [6] [▶ 894]-2.4

Programmierbeispiel

Senden von Signalen

```
(Signal 812, Synchronisation auf DEC-Ebene, Broadcast)
N500 #SIGNAL [ID812]

(Signal 4711, Synchronisation auf DEC-Ebene, an Kanal 2)
N100 #SIGNAL [ ID4711 CH2 ]

(Signal 4711, Synchronisation auf DEC-Ebene,
fuer 10 #WAIT-Abfragen, Broadcast)
N100 #SIGNAL [ ID4711 COUNT10 ]

(Signal 815, Synchronisation auf Interpolator-Ebene,
zweimal an Kanal 2 und einmal an 3)
N200 #SIGNAL SYN [ ID815 CH2 CH2 CH3 ]

(Signal 911, Synchronisation auf Decoder-Ebene, an Kanal 3)
(1.Signalparameter V.A.MEAS.ACS.VALUE.X, 2.Signalparameter P200,
3.Signalparameter 94.4)
N260 P200 = 924
N300 #SIGNAL [ IDP100 CH3 P[0]=V.A.MEAS.ACS.VALUE.X P[1]=P200
P[2]=94.4 ]
```

12.15.3 Löschen von (Broadcast-) Signalen (#SIGNAL REMOVE)

Signale werden i.a. nach dem Verbrauchen durch ein zugeordnetes WAIT gelöscht. Zusätzlich werden Standardsignale implizit bei einem NC-Reset des Empfängerkanals gelöscht (s. a. Kapitel Reset-Behandlung). Da Broadcast-Signale ohne Angabe des Verbrauchszählers nicht durch ein WAIT gelöscht werden, müssen diese explizit entfernt werden. Hierzu existiert ein zusätzlicher NC-Befehl. Dieser NC-Befehl kann ebenso für das Löschen von Standardsignalen angewandt werden, wobei hier die Identnummer und der adressierte Kanal übereinstimmen müssen.

Wird ein einzelnes Signal zum Löschen angegeben, so wird, falls evtl. mehrere gleiche Signale vorhanden sind, auch nur eines davon entfernt. Dagegen werden bei Angabe eines Signalbereichs [ID; IDMAX] alle Signale, d.h. auch evtl. mehrere gleiche Signalnummern, innerhalb dieses Bereichs entfernt.

Syntax:

#SIGNAL REMOVE [<*Modus*>] **[ID=..** | **IDMIN**=.. [**IDMAX**=..] { **CH=..** } **]**

<*Modus*>	Synchronisationsart. Zulässige Kennungen: ---: Synchronisation auf Decoder-Ebene (Grundeinstellung). Diese Synchronisation ist z.B. erforderlich, wenn auf Parameter oder Variablen synchronisiert werden soll. SYN: Synchronisation auf Interpolator-Ebene. Diese Synchronisation ist bei Echtzeitanforderungen notwendig, z.B. Synchronisation zweier Bearbeitungseinheiten einer Mehrständermaschine
ID=..	Nummer des zu löschenden Broadcast-Signals. Positive Ganzzahl.
IDMIN=..	Erstes zu löschende Broadcast-Signal eines Bereiches. Auch alternativ zu ID=... Positive Ganzzahl
IDMAX=..	Letztes zu löschende Broadcast-Signal eines Bereiches. Positive Ganzzahl
CH=..	Nummer des Kanals, für den das zu löschende Signal bestimmt ist. 1...max. Anzahl Kanäle (1) Wird keine Kanalnummer angegeben, so wird das entsprechende **Broadcast** –Signal gelöscht

(1) siehe [6] [▶ 894]-2.4

Programmierbeispiel

Löschen von (Broadcast-) Signalen

(Löschen eines Broadcast-Signals 812, Synchronisation auf DEC-Ebene)
N500 #SIGNAL REMOVE [ID812] oder
#SIGNAL REMOVE [IDMIN812]

(Löschen eines Signals 812 an Kanal2, Synchronisation auf DEC-Ebene)
N500 #SIGNAL REMOVE [ID812 CH2]

(Löschen aller Broadcast-Signale innerhalb 812-820, Synchronisation auf DEC-Ebene)
N500 #SIGNAL REMOVE [IDMIN812 IDMAX820] oder
#SIGNAL REMOVE [ID812 IDMAX820]

(Löschen aller Signale 812 an Kanal 1, Synchronisation auf DEC-Ebene)
N500 #SIGNAL REMOVE [ID812 IDMAX812 CH1]

(Löschen eines Broadcast-Signals 813, Synchronisation auf Interpolator-Ebene)
N600 #SIGNAL REMOVE SYN [ID813]

12.15.4 Warten auf Signale (#WAIT)

Entsprechend dem Versenden von Signalen kann durch den #WAIT-Befehl auf ein entsprechendes #SIGNAL gewartet werden. Ein Broadcast-WAIT wartet dabei nur auf ein Broadcast-SIGNAL mit gleicher Signalnummer. Jedes #WAIT verbraucht jeweils ein eigenes #SIGNAL.

Syntax:

#WAIT [<*Modus*>] **[ID**=.. { **P[**<*idx*>**] =** <*param*> } { **CH**=.. } [**AHEAD**] **]**

<*Modus*>	Synchronisationsart. Zulässige Kennungen: ---: Synchronisation auf Decoder-Ebene (Grundeinstellung). Diese Synchronisation ist z.B. erforderlich, wenn auf Parameter oder Variablen synchronisiert werden soll. SYN: Synchronisation auf Interpolator-Ebene. Diese Synchronisation ist bei Echtzeitanforderungen notwendig, z.B. Synchronisation zweier Bearbeitungseinheiten einer Mehrständermaschine
ID=..	Nummer des Signals, auf das gewartet wird. Positive Ganzzahl.
P[<*idx*>] = <*param*>	Signalparameter als Realzahl. Beim Senden von Signalen können Signalparameter an den mit #WAIT wartenden Empfänger übergeben werden. Sie können beim erfolgreichen #WAIT ausgelesen werden, wobei der Wert der Variablen (<param>) zugewiesen wird. (**Achtung**: In der Gleichung wird der linke Wert dem rechten Wert zugewiesen). Beim Lesen des Signalparameters wird überprüft, ob der übertragene Signalparameter an Index <idx> in die „Zielvariable “ (<param>) übernommen wurde. Ist dies nicht der Fall, so wird der Fehler ID 21549 ausgegeben. <*idx*>: Bereich für die maximal mögliche Parameteranzahl: 0 .. 11 (max. Anzahl Signalparameter (1))

Achtung

Signalparameter können nur auf Decoder-Ebene ausgewertet werden, d.h. z.B. ein #WAIT **SYN** [... P[0] = ...] ist nicht erlaubt.

CH=..	Nummer des Kanals, von dem ein Signal erwartet wird. 1...max. Anzahl Kanäle (2) Wird keine Kanalnummer angegeben, so wird auf ein Broadcast-Signal eines beliebigen Teilnehmers gewartet.
AHEAD	Kennung für die Ausführung eines „fliegenden“ WAIT. Dient der Minimierung von Wartezeiten aufgrund der Pufferwirkung des Look-Ahead (bis zu 70 Sätze im Voraus). Bei Synchronisation auf Interpolator-Ebene sofortige Ausgabe von WAIT, damit bei der nachfolgenden Prüfung auf Quittierung (SIGNAL) fliegend, d.h. ohne anzuhalten sofort in den nächsten Bewegungssatz gewechselt werden kann.

(1) siehe [6] [▷ 894]-6.45

(2) siehe [6] [▷ 894]-2.4

Programmierbeispiel

Warten auf Signale

```
(Wartemarke 4711, Synchronisation auf DEC-Ebene, SIGNAL 4711 von
beliebigem Kanal)
N200 #WAIT [ID4711]

(Wartemarke 815, Synchronisation auf Interpolator-Ebene,
SIGNAL 815 von Kanal 2 und 3)
N100 #WAIT SYN [ID815 CH2 CH3]

(Wartemarke 911, Synchronisation auf Decoder-Ebene, von Kanal 3)
N250 P100 = 911
( P[0] wird V.P.SIGNAL , P[1] wird P200 zugewiesen)
N300 #WAIT [IDP100 P[0]=V.P.SIGNAL P[1]=P200 CH3]
(Nachfolgende Berechnung findet erst nach Eintreffen)
(des Signals statt)
N350 P20 = 10 * P200
```

Programmierbeispiel

Warten auf Signale mit Übernahme von Parametern (in Kanal 3):

```
%channel1
N10 #SIGNAL [ID 110014 P[0] = 1234 CH3]
N20 M30

%channel2
N10 #SIGNAL [ID 110014 P[1] = 200 CH3]
N20 M30

%channel3
N10 P1 = 1 (Speichert Wert von Kanal 1
N20 P2 = 1 (Speichert Wert von Kanal 2
N30 XP1 YP2
N40 #WAIT [ID 110014 P[0] = P1 P[1] = P2 CH1 CH2]
N50 XP1 YP2
N60 M30
```

12.15.5 Lesen von Signalen ohne Warten (#SIGNAL READ)

Versionshinweis

Diese Funktionalität ist verfügbar ab der Version V2.11.2820.00

Der NC-Befehl #WAIT stoppt die Programmdekodierung des Interpreters, falls das abgefragte Signal nicht vorhanden ist. Somit ist eine weitere Programmabarbeitung blockiert, falls das Signal nie eintrifft.

Der NC-Befehl #SIGNAL READ ermöglicht flexible Programmabläufe ohne Programmstopp. Bei diesem Befehl wird das Ergebnis des Signal-Lesevorgangs in der kanalspezifischen Variable V.G.SIGNAL_READ gespeichert. Die anschließende Auswertung dieser Variable erlaubt dann eine entsprechende Reaktion.

Bzgl. dem Verbrauch von Signalen sowie der Programmierung und Nutzung von Parametern und Broadcast-Signalen verhält sich #SIGNAL READ analog zu #WAIT.

Achtung

Der NC-Befehl #SIGNAL READ ist nur auf Interpreterebene zulässig, Ein #SIGNAL READ SYN [..] ist nicht erlaubt und wird durch eine Fehlermeldung angezeigt

Der Status des Lesezugriffs von #SIGNAL READ wird durch die Variable V.G.SIGNAL_READ angezeigt. Sie ist TRUE, wenn das entsprechende Signal vorhanden war. Der Wert der Variable bleibt bis zum nächsten Lesezugriff mit #SIGNAL READ erhalten.

V.G.SIGNAL_READ	Status des Lesezugriffs von #SIGNAL READ TRUE: Signal vorhanden und gelesen FALSE: Kein Signal vorhanden, Default

Syntax:

#SIGNAL READ [ID=.. { **P[***<idx>***] =** *<param>* } { **CH**=.. } **]**

ID=..	Nummer des Signals, das gelesen werden soll. Positive Ganzzahl.
P[<idx>] = <param>	Signalparameter als Realzahl. Beim Lesen von Signalen können auch Parameter durch den Signalsender mit übertragen werden. Die Parameter können dabei auch von unterschiedlichen Kanälen stammen. Sie werden den angegebenen Parametern bzw. Variablen (*<param>*) zugewiesen. *<idx>:* Bereich für die maximal mögliche Parameteranzahl: 0...max. Anzahl Signalparameter (1) Nach vollständiger Quittierung aller notwendigen Signale wird geprüft, ob alle programmierten Parameter geschrieben worden sind. Ist dies nicht der Fall, so wird mit einer Fehlermeldung angehalten
CH=..	Nummer des Kanals, von dem ein Signal erwartet wird. 1...max. Anzahl Kanäle (2) Wird keine Kanalnummer angegeben, so wird auf ein Broadcast-Signal eines beliebigen Teilnehmers gewartet

(1) siehe [6] [▶ 894]-6.45

(2) siehe [6] [▶ 894]-2.4

Programmierbeispiel

Lesen von Signalen und Abfrage des Ergebnisses

```
(Wartemarke 4711, Synchronisation auf Interpreter-Ebene, SIGNAL 4711 von
beliebigem Kanal)
:
N100 #SIGNAL READ [ID=4711]
N110 $IF V.G.SIGNAL_READ == TRUE
N120   LL UP1
N130 $ELSE
N140   #ERROR [..]
N150 $ENDIF
:

(Wartemarke 815, Synchronisation auf Interpreter-Ebene,
SIGNAL 815 von Kanal 1)
:
N100 #SIGNAL READ [ID=815 CH=1]
N110 $IF V.G.SIGNAL_READ == TRUE
N120   LL UP1
N130 $ELSE
N140   L Init.nc
N150 $ENDIF
:

(Wartemarke 911, Synchronisation auf Interpreter-Ebene, von Kanal 3)
(1.Signalparameter V.P.SIGNAL, 2.Signalparameter P200)
:
N100 #SIGNAL READ [ID=911 P[0]=V.P.SIGNAL P[1]=P200 CH=3]
N110 $IF V.G.SIGNAL_READ == TRUE
N120   P20 = [10 * P200]-V.P.SIGNAL
N130 $ELSE
N140   P20 = 0
N150 $ENDIF
```

12.15.6 RESET-Behandlung

Beim Reset eines einzelnen Kanals werden die Synchronisationsereignisse des betroffenen Kanals gelöscht, d.h. alle Warteanforderungen (#WAIT), die der betroffene Kanal geschickt hat, und alle Standardsignale (#SIGNAL), die für diesen bestimmt sind.

Broadcast-Signale werden bei einem Reset eines Kanals nicht gelöscht, da diese evtl. noch von Kanälen erwartet werden. Diese müssen explizit (#SIGNAL REMOVE) gelöscht werden.

12.16 Drehung des Koordinatensystems in der Ebene (#ROTATION ON/OFF)

Mit dieser Funktion kann ein Koordinatensystem in der aktuellen Ebene (G17/G18/G19) gedreht werden. Konturen, die im Maschinenkoordinatensystem programmiert sind, können so schnell und einfach an verdreht positionierte Werkstücke angepasst werden.

Die Konturrotation wirkt direkt auf die programmierten Achskoordinaten (Kontur) **vor** allen anderen konturbeeinflussenden Funktionalitäten, d.h. alle Verschiebungen und Spiegelungen werden von der Rotation nicht beeinflusst und können wie bisher benutzt werden (*).

Die Rotation kann auch innerhalb eines bereits gedrehten Koordinatensystems (#(A)CS) angewandt werden.

Ein Ebenenwechsel mit G17/G18/G19 wählt automatisch eine aktive Konturrotation ab und es wird eine Warnung ausgegeben.

Alternativ zu #ROTATION kann die Konturrotation auch mit G68/G69 [▷ 189] programmiert werden.

Syntax:

#ROTATION ON [**[** [**ANGLE**=..] [**CENTER1**=..] [**CENTER2**=..] **]**]
#ROTATION OFF

ANGLE=..	Rotationswinkel in [°]
CENTER1=..	Versatz der ersten Hauptachse zum Drehpunkt in [mm, inch]
CENTER2=..	Versatz der zweiten Hauptachse zum Drehpunkt in [mm, inch]

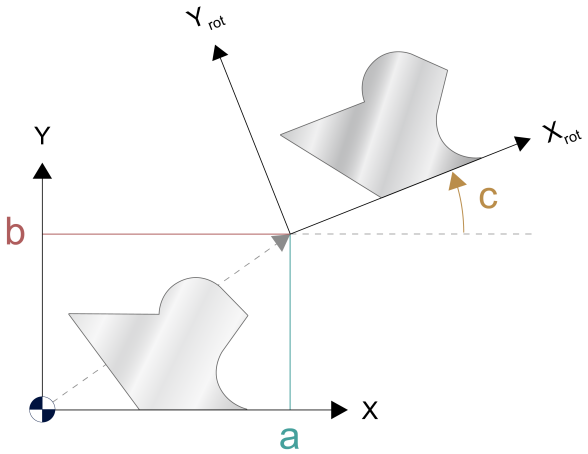


Abb. 115: Bedeutung der Rotationsparameter in der Hauptebene (Bsp. G17):

a: CENTER1	b: CENTER2	c: ANGLE

Mit nachfolgenden Variablen können die programmierten Rotationsparameter gelesen werden:

V.G.ROT_ACTIVE	Liefert den Wert 1, wenn eine Rotation aktiv ist
V.G.ROT_ANGLE	Rotationswinkel
V.G.ROT_CENTER1	Versatz der ersten Hauptachse zum Drehpunkt
V.G.ROT_CENTER2	Versatz der zweiten Hauptachse zum Drehpunkt

Hinweis

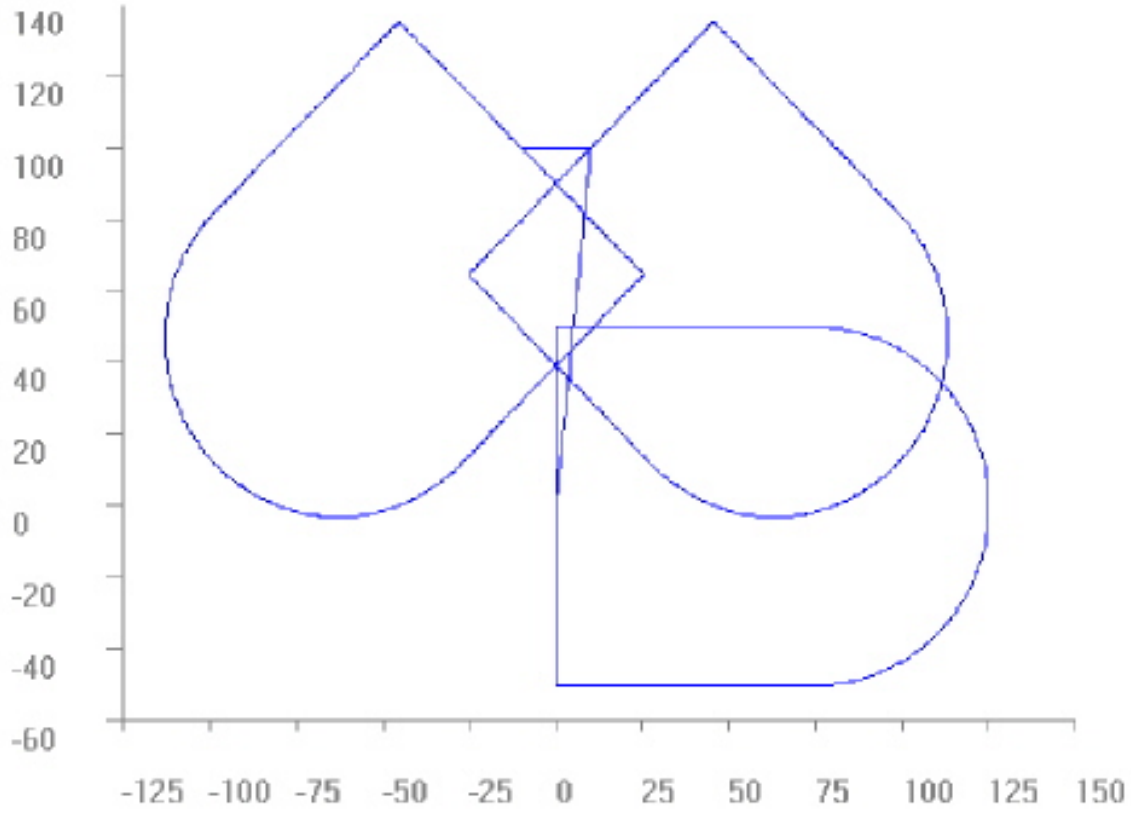
(*) Unabhängig davon, ob die Verschiebungen (z.B. G54, G92 etc.) vor oder nach dem #ROTATION-Befehl programmiert wurden, wirken diese immer in den Achsrichtungen des Grundkoordinatensystems der Maschine (MKS).

Auch die Werkzeugversätze wirken immer unabhängig von P-TOOL-00010 in den Achsrichtungen des MKS.

Programmierbeispiel

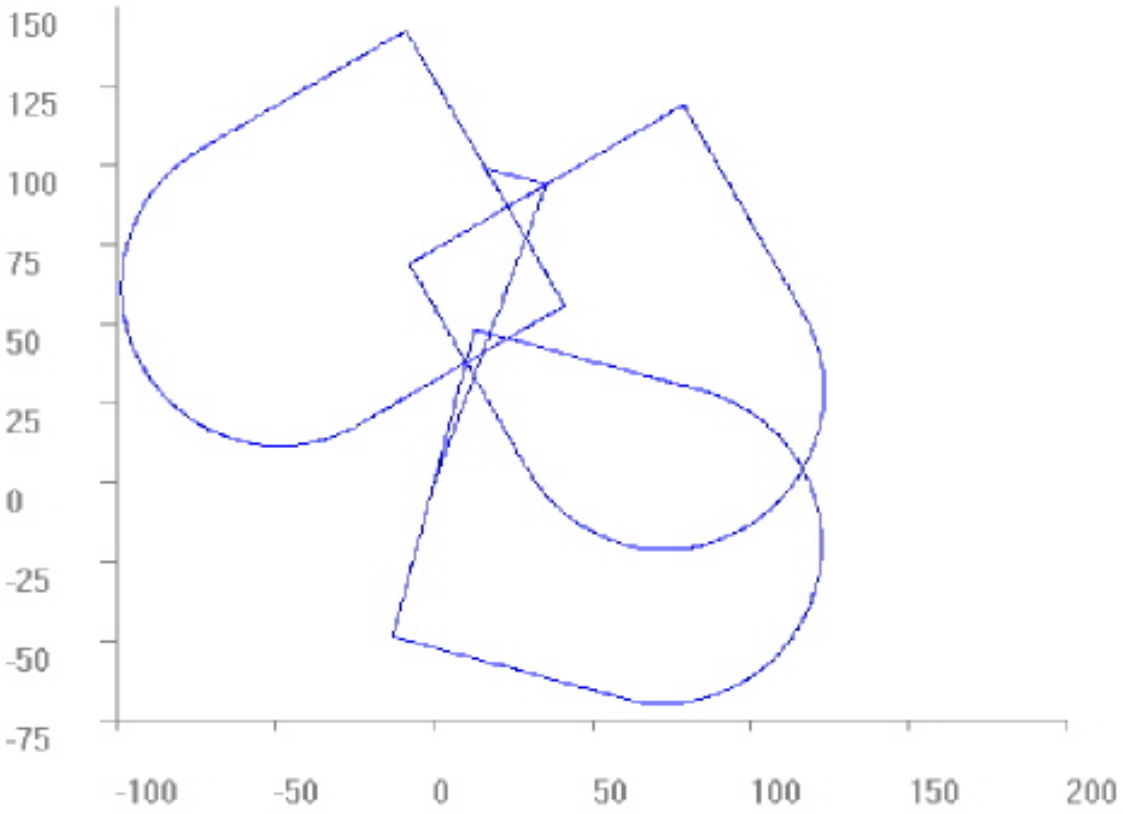
Drehung in der Ebene (Konturrotation)

```
%L part
N10 G0 G90 X0 Y0
N30 G1 F5000 Y50
N40 X75
N50 G2 Y-50 R50
N60 G1 X0
N70 Y0
N80 M29

%ang1.nc
N100 G53 G17
N110 LL part
N130 #ROTATION ON [ANGLE -45 CENTER1=10 CENTER2=100]
N140 LL part
N150 G21 (mirroring of X coordinates)
N160 LL part
N170 G18 (warning expected)
N190
M30
```

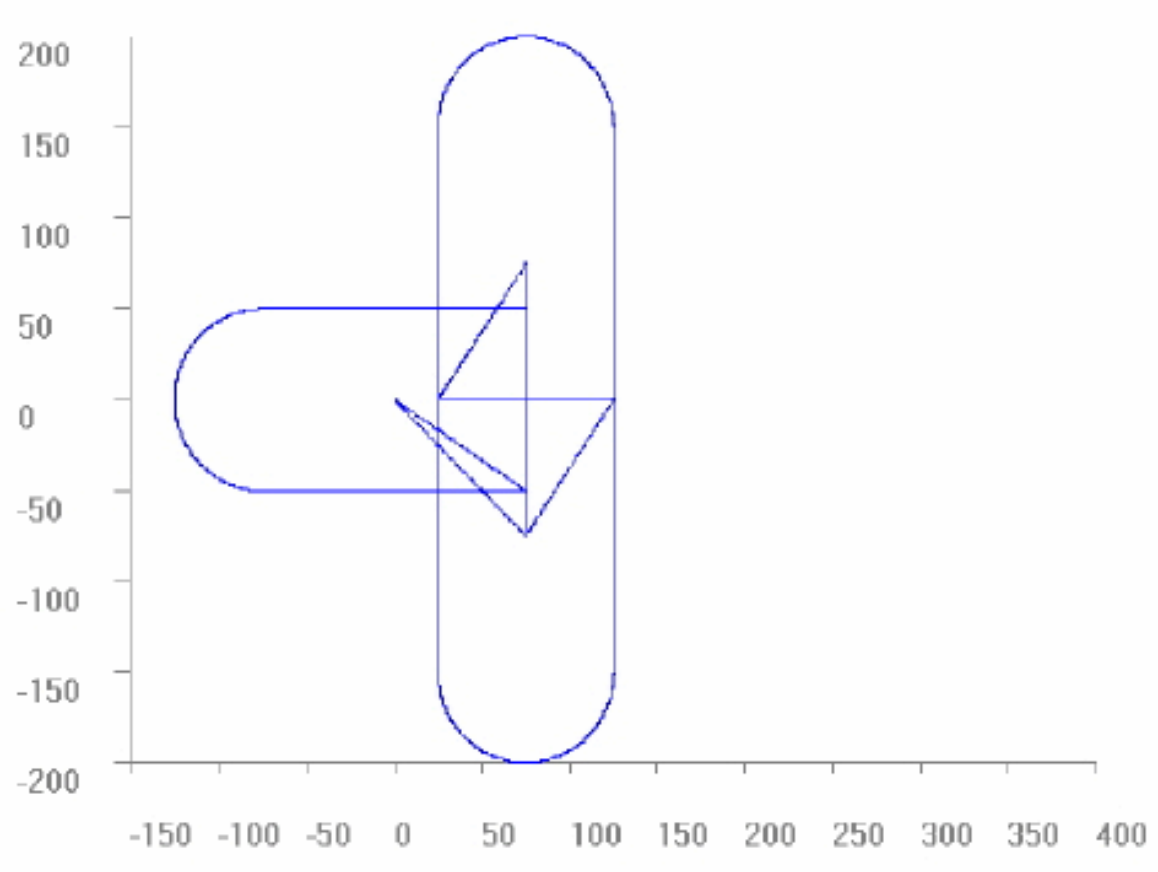

140
120
100
80
60
40
20
0
-20
-40
-60
-125
-100
-75
-50
-25
0
25
50
75
100
125
150

Gleiche Kontur wie im vorherigen Programm, jedoch innerhalb #CS von -15°.

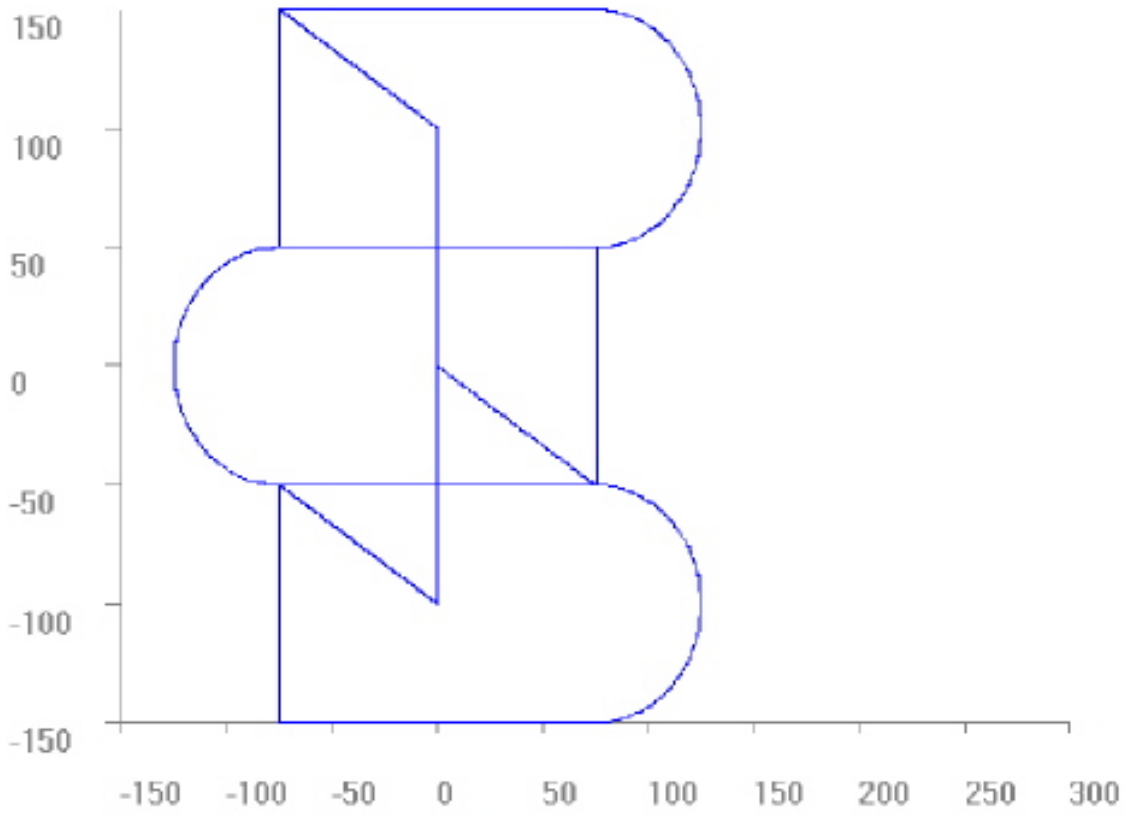
```
%L part
N10 G0 G90 X0 Y0
N30 G1 F5000 Y50
N40 X75
N50 G2 Y-50 R50
N60 G1 X0
N70 Y0
N80 M29

% ang1cs.nc
N99 #CS ON[0,0,0,0,0,-15]
N100 G53 G17
N110 LL part
N130 #ROTATION ON [ANGLE=-45 CENTER1 10 CENTER2 100]
N140 LL part
N150 G21 (mirroring of X coordinates)
N160 LL part
N190 #CS OFF
M30
```



```
%L Trajectory0
N10 G54 G90 X0 Y0
N20 G0 X75 Y-50
N30 Y50
N40 X-75
N50 G3 X-75 Y-50 R50
N60 G0 X75
N70 X0 Y0
N80 M29

%ang2.nc
F1000
N100 LL Trajectory0
N200 G92 G90 Y-25
N400 #ROTATION ON [ANGLE 90 CENTER1 75 CENTER2=-50]
N600 LL Trajectory0
N700 G92 G90 Y25
N900 #ROTATION ON [ANGLE=-90 CENTER1 75 CENTER2 50]
N60 LL Trajectory0
N70 M30
```

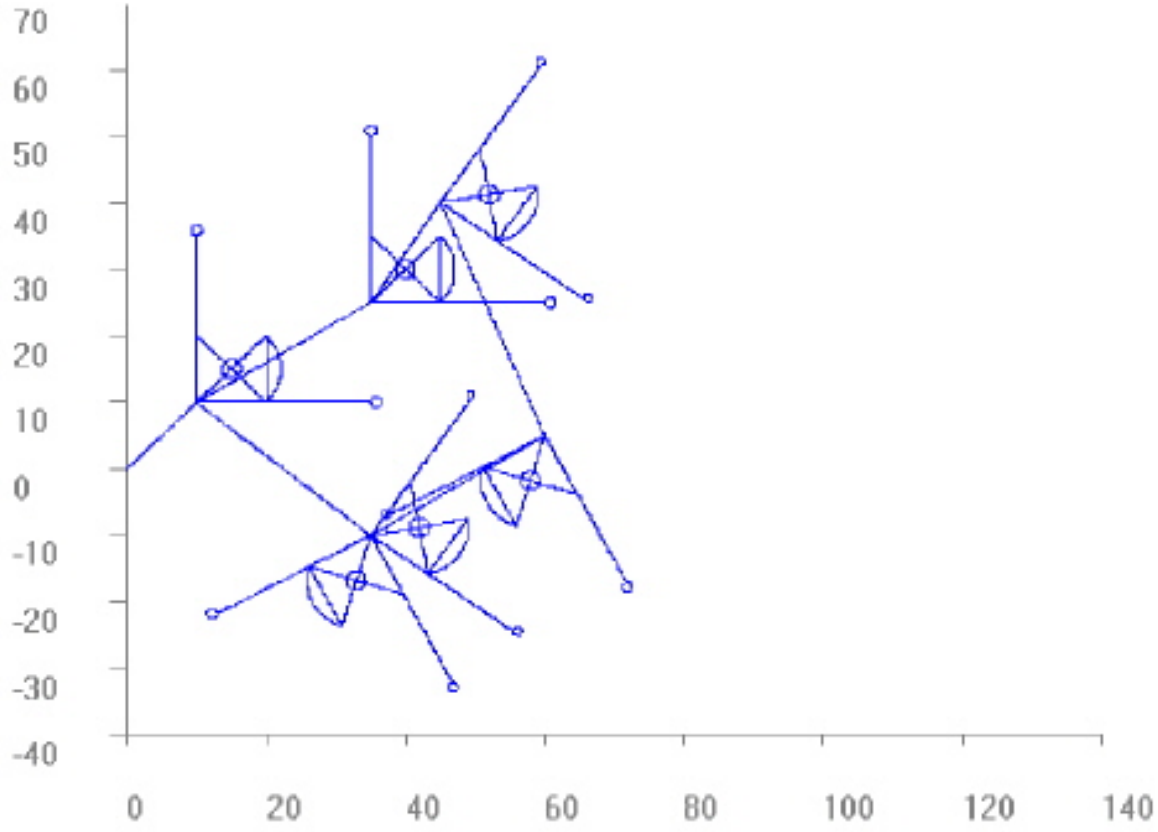


```
%L Trajectory0
N10 G54 G90 X0 Y0
N20 G1 X75 Y-50
N30 Y50
N40 X-75
N50 G3 X-75 Y-50 R50
N60 G1 X75
N70 X0 Y0
N80 M29

%ang3.nc
N10 F4000 G90
N15 #ROTATION ON
N20 LL Trajectory0
N30 G90 G92 Y100
N35 #ROTATION ON [ANGLE 180]
N40 LL Trajectory0
N50 G90 G92 Y-100
N55 #ROTATION ON [ANGLE 180]
N60 LL Trajectory0
N70 M30
```



```
%L UPRG1
N1 X0 Y0 Z0
N10 X25
N30 X0
N40 Y25
N50 Y0
N60 X10
N70 Y10
N80 X0 Y0
N90 Y10
N100 X10 Y0
N110 G03 I-5 J5 Y10
N120 G1 X0 Y0
M17

%L UPRG2
N2 X0 Y0 Z0
N10 X25
N20 G02 I0.8
N30 G1 X0
N40 Y25
N45 G02 J0.8
N50 G1 Y0
N120 G1 X0 Y0
M17

%L UPRG3
N3 G1 X0 Y0 Z0
N10 X4 Y4
N20 G02 I1 J1
N30 G1 X0 Y0 Z0
M17

%ang4.nc
N1 G1 X0 Y0 Z0 F1000

N500 G92 X10 Y10
N510 LL UPRG1
N520 #ROTATION ON [ANGLE 0 CENTER1 25 CENTER2 15]
N540 LL UPRG1
N550 G92 X20 Y25
N560 #ROTATION ON [ANGLE -35]
N570 LL UPRG1
N580 G92 X35 Y-10
N590 #ROTATION ON [ANGLE=V.G.ROT_ANGLE-117]
N600 LL UPRG1
N610 #ROTATION ON [CENTER1 0 CENTER2 0]
N620 LL UPRG1
N630 #ROTATION ON [ANGLE=V.G.ROT_ANGLE+117]
N640 LL UPRG1
N650 #ROTATION ON [ANGLE=V.G.ROT_ANGLE+35]

N500 G92 X10 Y10
N510 LL UPRG2
N520 #ROTATION ON [ANGLE 0 CENTER1 25 CENTER2 15]
N540 LL UPRG2
N550 G92 X20 Y25
N560 #ROTATION ON [ANGLE -35]
N570 LL UPRG2
N580 G92 X35 Y-10
N590 #ROTATION ON [ANGLE=V.G.ROT_ANGLE-117]
N600 LL UPRG2
N610 #ROTATION ON [CENTER1 0 CENTER2 0]
```

```
N620 LL UPRG2
N630 #ROTATION ON [ANGLE=V.G.ROT_ANGLE+117]
N640 LL UPRG2
N650 #ROTATION ON [ANGLE=V.G.ROT_ANGLE+35]
N500 G92 X10 Y10
N510 LL UPRG3
N520 #ROTATION ON [ANGLE 0 CENTER1 25 CENTER2 15]
N540 LL UPRG3
N550 G92 X20 Y25
N560 #ROTATION ON [ANGLE -35]
N570 LL UPRG3
N580 G92 X35 Y-10
N590 #ROTATION ON [ANGLE=V.G.ROT_ANGLE-117]
N600 LL UPRG3
N610 #ROTATION ON [CENTER1 0 CENTER2 0]
N620 LL UPRG3
N630 #ROTATION ON [ANGLE=V.G.ROT_ANGLE+117]
N640 LL UPRG3
N650 #ROTATION ON [ANGLE=V.G.ROT_ANGLE+35]
M30
```

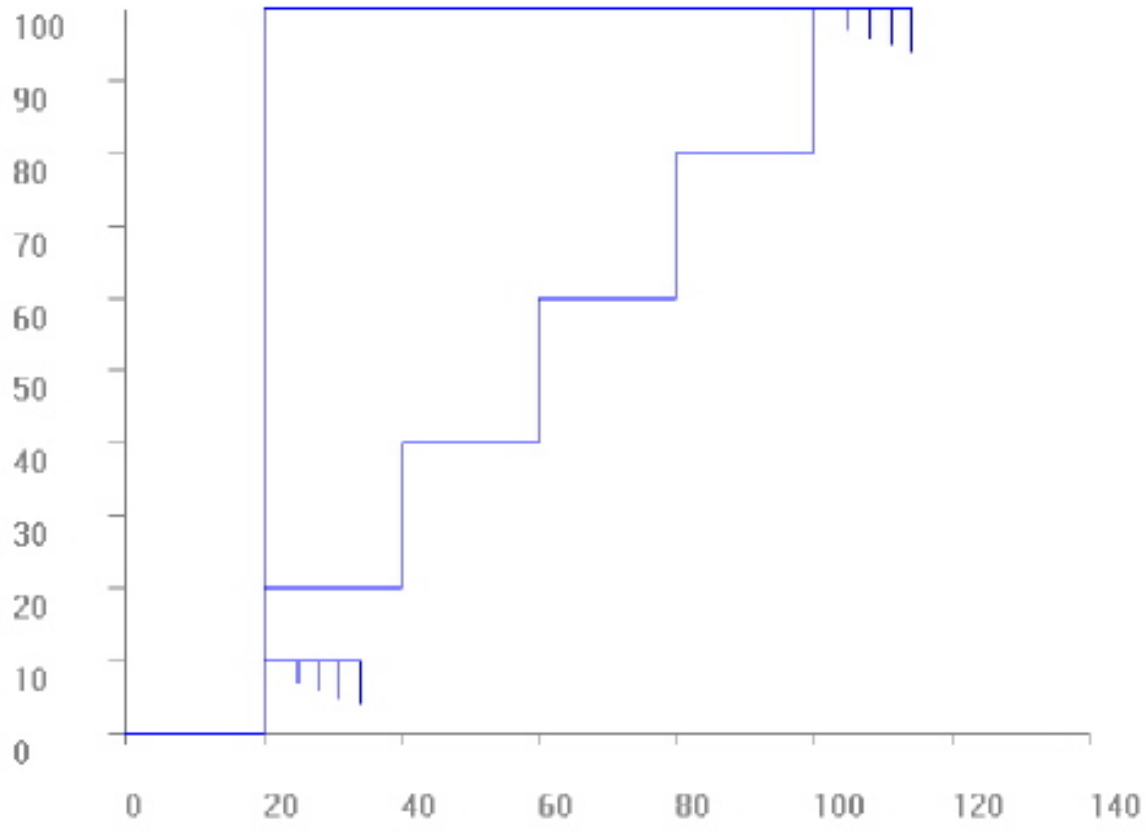


Test von relativer und absoluter Programmierung:

```
%L contour_1
N1  G1 G91          (all positions with G91)
N2  X20
N3      Y20
N4  X20
N5      Y20
N6  X20
N7      Y20
N8  X20
N9      Y20
N10 X20
N11     Y20
N12         X5
N13         Y-3
N14         Y3
N15         X-5
N16 G90 X0
N17     Y0
N18         X5
N19         Y-3
N20         Y0
N21         X0
#MSG SYN["contour_1 finished"]
M17
%L contour_2
N100 G1           (same contour, X with G91, Y with G90)
N101 G91 X20
N102 G90     Y10 (transl. offset in Y is 10)
N103 G91 X20
N104 G90     Y30
N105 G91 X20
N106 G90     Y50
N107 G91 X20
N108 G90     Y70
N109 G91 X20
N110 G90     Y90
N111              G91 X8
N112              G91 Y-4
N113              G91 Y4
N114              G91 X-8
N115 G90 X0
N116 Y0
N117     X8
N118     Y-4
N119     Y0
N119     Y0
N120     X0
#MSG SYN["contour_2 finished"]
M17
%L contour_3
N200 G1           (same contour, Y with G91, X with G90)
N201 G90 X0       (transl. offset in X is 20)
N202 G91     Y20
N203 G90 X20
N204 G91     Y20
N205 G90 X40
N206 G91     Y20
N207 G90 X60
N208 G91     Y20
N209 G90 X80
N210 G91     Y20
```

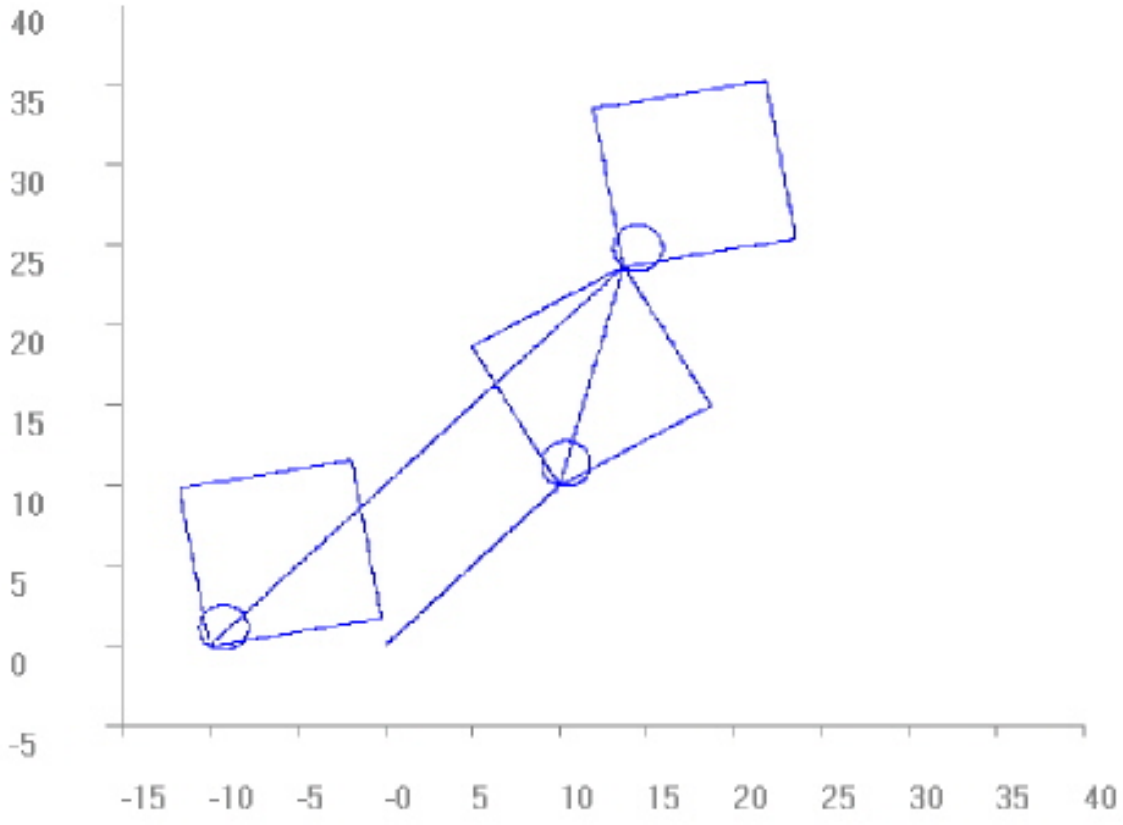
```
N211              G91 X11
N212              G91 Y-5
N213              G91 Y5
N214              G91 X-11
N215 G90 X0
N216 Y0
N217     X11
N218     Y-5
N219     Y0
N220     X0
#MSG SYN["contour_3 finished"]
M17
%L contour_4
N300 G1 G90          (same contour with G90)
N301 X0              (transl. offset in X is 20)
N302     Y10         (transl. offset in Y is 10)
N303 X20
N304     Y30
N305 X40
N306     Y50
N307 X60
N308     Y70
N309 X80
N310     Y90
N111              G91 X14
N312              G91 Y-6
N313              G91 Y6
N314              G91 X-14
N315 G90 X0
N316 Y0
N317     X14
N318     Y-6
N319     Y0
N320     X0
#MSG SYN["contour_4 finished"]
M17
%ang5.nc
N5001 G0 G90 X0 Y0 F5000

N501 #ROTATION ON [ANGLE 0 CENTER1 20 CENTER2 10]
(Note: with angle != 0 the contours are
(     not congruent because of difference
(     of absolute and incremental movement !)

N502 #ROTATION ON
N503 LL contour_1
N504 #ROTATION OFF
N5002 G0 G90    Y0
N5003         X0

N505 #ROTATION ON
N506 LL contour_2
N507 #ROTATION OFF
N5004 G0 G90    Y0
N5005         X0

N508 #ROTATION ON
N509 LL contour_3
N510 #ROTATION OFF
N5006 G0 G90 Y0
N5007         X0

N511 #ROTATION ON
```

```
N512 LL contour_4
N513 #ROTATION OFF
N5005 G0 G90    Y0
N5006           X0

N210 M2
```



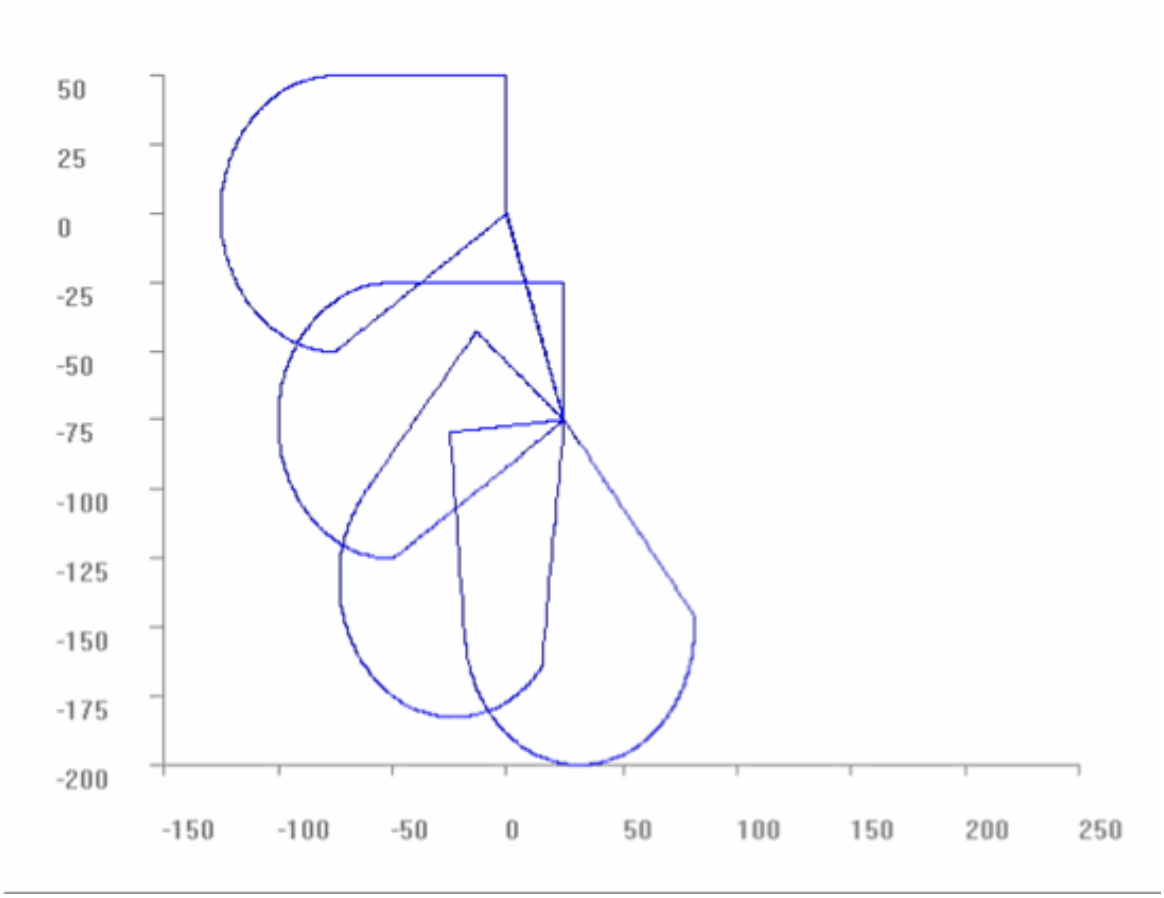
Gültiger Drehwinkel nach Anwahl. Die Verschiebungen für den Drehpunkt dürfen erst mit der ersten absoluten Programmierung (G90) berücksichtigt werden.

```
%ang6.nc
N10  G90 X0Y0Z0 G1 F200
N20  #ROTATION ON [ANGLE 30 CENTER1 10 CENTER2 10]
N30  X0 Y0
N40  G3  I1 J1 F500
N50  G01 X10
N60      Y10
N70 G90 X0
N80 G90 Y0
N90 X10 Y10
(New rotation parameters.
(Note: Center offset has no effect until an absolute (G90) position
(      has been programmed, the angle is effective however.
N100 #ROTATION ON [ANGLE 10 CENTER1 -10 CENTER2 0]
N110 G3 I1 J1 F500
N120 G01 G91 X10
N130 G91 Y10
N140 G91 X-10
N150 G91 Y-10
(Make the new center effective by first absolute position:
N200 G90 X0 Y0
N210 G3 I1 J1 F500
N220 G01 X10
N230 Y10
N240 X0
N250 Y0

M30
```



Transformierung des absolut oder relativ programmierten Kreismittelpunktes:

```
%ang_cent.nc
N10 F2000 G53
N11 G0 X0 Y0 G90
(------------------------------------------------------------------------)
(4 times same circle with different programming of circle center point)
(------------------------------------------------------------------------)

N12 G0 X0 Y0 G90
N13 Y50
N14 X-75
N15 G3 X-75 Y-50 G161 I-75 J0 (absolute center)
N16 G0 X0 Y0 G90
N17 Y50
N18 X-75
N19 G3 X-75 Y-50 G162 I0 J-50 (relative center)
N20 G0 X0 Y0 G90
N28 #ROTATION ON [ANGLE 0 CENTER1 25 CENTER2 -75]
(------------------------------------------------------------------------)
(The same with LIN and ANG offset active (ED=0) )
(------------------------------------------------------------------------)

N80 G0 X0 Y0 G90
N90 Y50
N100 X-75
N110 G3 X-75 Y-50 G161 I-75 J0 (absolute center)
N120 G0 X0 Y0 G90
N130 Y50
N140 X-75
N150 G3 X-75 Y-50 G162 I0 J-50 (relative center)
N360 G0 X0 Y0 G90

(------------------------------------------------------------------------)
(The same rotated by 50° (unnecessary I / J omitted) )
(------------------------------------------------------------------------)
N370 #ROTATION ON [ANGLE 50]
N380 G0 X0 Y0 G90
N390 Y50
N400 X-75
N410 G3 X-75 Y-50 G161 I-75 (J0) (absolute center, not prog. is 0)
N420 G0 X0 Y0 G90
N630 Y50
N640 X-75
N650 G3 X-75 Y-50 G162 I0 J-50 (relative center)
N655 G0 X0 Y0 G90

(------------------------------------------------------------------------)
(The same rotated by 95° )
(------------------------------------------------------------------------)
N660 #ROTATION ON [ANGLE 95]
N670 G0 X0 Y0 G90
N680 Y50
N690 X-75
N700 G3 X-75 Y-50 G161 I-75 J0 (absolute center)
N710 G0 X0 Y0 G90
N730 Y50
N740 X-75
N750 G3 X-75 Y-50 G162 I0 J-50 (relative center)
N760 G0 X0 Y0 G90
M30
```


50
25
0
-25
-50
-75
-100
-125
-150
-175
-200
-150
-100
-50
0
50
100
150
200
250

12.17 Automatische Achsnachführung (C-Achsnachführung) (#CAXTRACK)

Bearbeitungsvorgänge wie z.B. das Schneiden verschiedener Materialien erfordern das Führen des benutzen Werkzeugs, welches i.a. mit der C-Achse verbunden ist, derart, dass dieses stets tangential zur gefahrenen Bahn ausgerichtet ist.

Dabei ist zu beachten, dass die Tangente nicht an jedem Punkt der Bahn eindeutig ist (Knickstellen). Eine Lösung erfordert daher Strategien zur Behandlung nicht tangentenstetiger Satzübergänge.

Ein typischer Anwendungsfall ist im Technologiebereich der Glasschneidtechnik zu finden. Dort werden mit Hilfe von CNC-Maschinen ebene Konturen mit Hilfe von Schneidwerkzeugen in Form von Hartmetallschneidrädchen bearbeitet. Entsprechend dem programmierten Konturverlauf (geschlossene Kontur, z.B. Ellipse) wird das ebene Werkstück an der Bearbeitungsstelle angeritzt. Die gewünschte Kontur kann dann aus dem Glaswerkstück ausgebrochen werden.

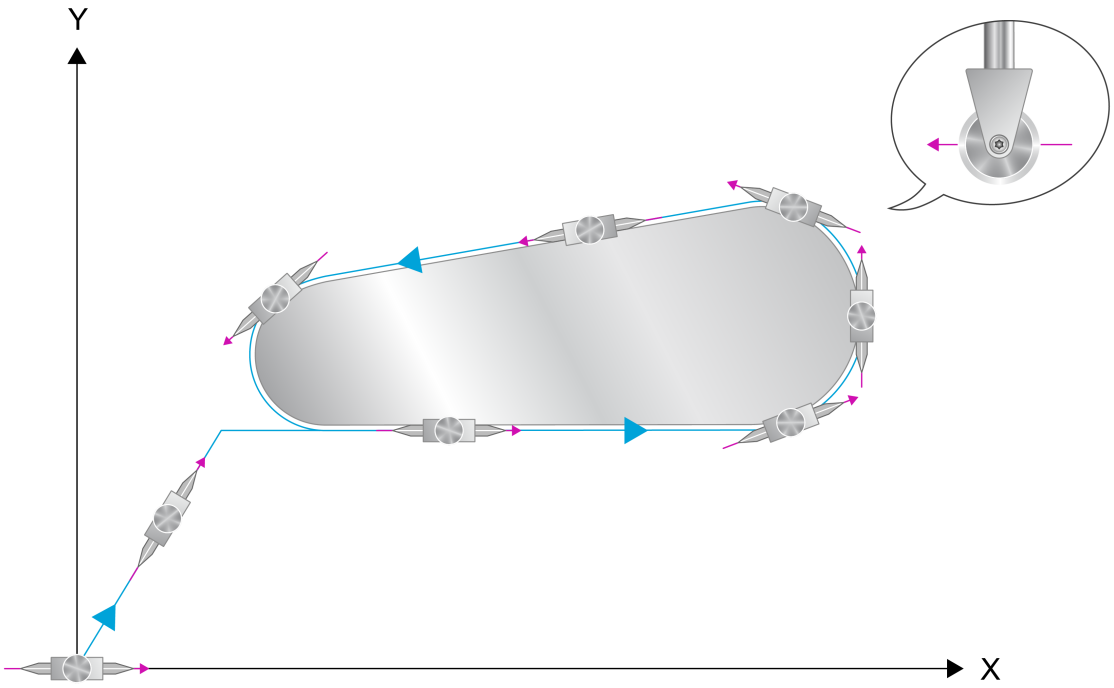


Abb. 116: Nachführen der rotatorischen C-Achse tangential zur x-y-Kontur

Die C-Achse kann auch durch explizite Programmierung tangential zur Bahn geführt werden. Die im Folgenden beschriebenen NC-Befehle erleichtern jedoch die Programmierung erheblich.

Syntax:

#CAXTRACK ON [**[** [**ANGLIMIT**=..] [**OFFSET**=..] [**OPTALIGN**=..] [**ROTMODE**=..] [**SCALEFACT**=..]
[**AX**=<*Achsname*> | **AXNR**=..] [**START_STROKE**] **]**
#CAXTRACK OFF [**[ANGPOS=..]**]

ANGLIMIT=..

Grenzwinkel in [°].

Dieser Parameter wird nur bei nicht tangentenstetigen Konturabschnitten berücksichtigt. Tangentenstetige Konturabschnitte entstehen z.B. durch die Überschleiffunktion G61.

Überschreitet der Winkel zwischen den Tangenten an die Kontur beim Satzübergang den Grenzwinkel, so wird die Bahnbewegung angehalten und die Richtbewegung durch einen eingefügten Bewegungssatz mit Eilganggeschwindigkeit durchgeführt. Die eingefügte Bewegung bildet hierbei eine Einheit mit dem folgenden, zweiten Satz. Dies bedeutet insbesondere, dass SPS-Synchronisationsereignisse in Zusammenhang mit der Bewegung (M-Funktionen etc.) nur vor oder hinter dieser Bewegungseinheit möglich sind.

Ist der Übergangswinkel kleiner als der Grenzwinkel, so erfolgt die Richtbewegung sofort mit Beginn des zweiten Satzes, wenn die dynamischen Betrachtungen keine Reduktion der Bahngeschwindigkeit erfordern. Im Allgemeinen führen die begrenzten Achsbeschleunigungen zu einer niedrigen Vorschubgeschwindigkeit am Satzübergang. Ist dies nicht erwünscht, so kann die Nachführachse von der dynamischen Betrachtung ausgenommen werden. (z.B. G116 C1 [▶ 184])

OFFSET=..

Winkeloffset in [°].

Dient zur Angabe eines Winkeloffsets um das Werkzeug gegenüber der Tangente an die Kontur zu orientieren.

OPTALIGN=..

Bei Anwahl wird automatisch optimiert ausgerichtet, wenn der Ausrichtweg größer als der angebene Winkelwert 'OPTALIGN' in [°] ist.

Dieser Parameter wird nur bei aktivem automatischem Ausrichten entsprechend P-CHAN-00101 und rotatorischen Linearachsen mit begrenztem Verfahrbereich (keine Moduloachse) berücksichtigt und ist nur während des automatischen Ausrichtvorgangs auf das erste Konturelement wirksam.

Nach Anwahl der tangentialen Nachführfunktion mit automatischem Ausrichten liegt die Position der Nachführachse ohne Offset im Bereich von -180..+180°. Über den Parameter kann gesteuert werden, dass die Stellung der Nachführachse vor Anwahl der tangentialen Nachführfunktion beim automatischen Ausrichtvorgang berücksichtigt wird.

Die Verwendung der Funktion ist dann sinnvoll, wenn die Nachführachse bereits vor Anwahl des automatischen Ausrichtens näherungsweise die korrekte Position zum ersten Konturelement hat, aber z.B. mit einer Vorverdrehung von +-360° beaufschlagt ist. Überschreitet der intern berechnete Ausrichtwinkelwert den angegebenen Winkelwert 'OPTALIGN', so werden alternative Lösungen für den Ausrichtwinkel untersucht. Der kleinste Ausrichtweg der Lösungen legt dann den tatsächlichen Ausrichtwinkel fest. (*)

Hinweis

(*) Bei Moduloachsen erfolgt der automatische Ausrichtvorgang immer auf dem kürzesten Weg.

ROTMODE=..

Bool'scher Wert, der die Lage der Nachführachse bestimmt:

0: Die Nachführachse ist eine Achse im Werkzeug (Default).

1: Die Nachführachse ist eine Achse im Werkstück.

Die Werkzeugachse muss dabei immer senkrecht zur XY Bearbeitungsebene stehen. Alternativ kann die Lage der Nachführachse auch in den Kanalparametern definiert werden (P-CHAN-00185).

SCALEFACT=..	Skalierungsfaktor für den Nachführwinkel mit Werten >0.0 ... ≤1.0.
	Bei Werten außerhalb der zulässigen Grenzen wird der Skalierungsfaktor auf 1.0 (Default) gesetzt. Nur für Sonderapplikationen.
ANGPOS=..	Position bei Abwahl in [°].
	Während der Abwahl kann die Nachführachse zusätzlich positioniert werden. Die Positionierbewegung erfolgt bei rotatorischen Achsen auf kürzestem Weg.
AX=<*Achsname*>	Vorgabe der Nachführachse über Namen. Diese ist bis Programmende (M30) gültig.
	Nach Programmstart bzw. wenn keine Nachführachse programmiert ist, ist die im Kanalparameter P-CHAN-00095 festgelegte Standardachse gültig.
AXNR=..	Vorgabe der Nachführachse über logische Achsnummer, Positive Ganzzahl. Diese ist bis Programmende (M30) gültig.
	Nach Programmstart bzw. wenn keine Nachführachse programmiert ist, ist die im Kanalparameter P-CHAN-00095 festgelegte Standardachse gültig.
START_STROKE	Einmalige Ausführung einer reduzierten Richtsequenz, definiert ab #STROKE DEF CAXTRACK ALIGN BLOCK bei #CAXTRACK ON. Wird automatisches Ausrichten mit P-CHAN-00101 verwendet, so wird die Richtbewegung zusätzlich am Anfang der Richtsequenz ausgeführt. START_STROKE ist nicht haltend und muss bei jeder Aktivierung von #CAXTRACK ON programmiert werden. START_STROKE ist nur bei Verwendung einer Richtsequenz relevant.

Hinweis

Die verwendete Tracking-Achse muss eine Zusatzachse sein. Diese darf nicht auf einem Hauptachsenindex liegen.

Die automatische Nachführung der Achse erfolgt vorzeichenrichtig, relativ zur letzten Position entsprechend dem sich ergebenden Konturübergangswinkel.

Hinweis

Abhängig von der Parametrierung P-CHAN-00101 erfolgt die **Ausrichtung der Nachführachse** in die gewünschte Orientierung (i.a. parallel zur Kontur) wie folgt:

- Programmiertes Ausrichten vor Anwahl der automatischen Nachführung. Es erfolgt keine Prüfung der Position. Die aktuelle Winkelstellung wird eingefroren und die Nachführachse wird mit diesem Winkel nachgeführt.
- Automatisches tangentiales Ausrichten (P-CHAN-00101) der Nachführachse zum ersten programmierten Konturelement bei Anwahl der automatischen Nachführung.

Achtung: Die Aktivierung des automatischen Ausrichtens ist zwingend erforderlich, wenn Konturabschnitte mit Polynomen (z.B. G261) programmiert sind!

Die Nachführung beginnt mit dem Übergang vom ersten zum zweiten relevanten Bewegungssatz nach Aktivierung mit #CAXTRACK ON. Die automatische Achsnachführung arbeitet in der Hauptebene der Zirkularinterpolation (1. + 2.HA). Diese muss vor Aktivierung festgelegt werden (G17 / 18 / 19, #PUT AX / #CALL AX / #SET AX).

Besitzt die Nachführachse bei Anwahl des Nachführbetriebes bereits die erforderliche Orientierung, so wird bei gesetztem Parameter P-CHAN-00109 ohne Anhalten in den ersten relevanten Bewegungssatz eingefahren.

Bei bereits aktiver Überschleiffunktion (G261) und Parameter ANGLIMIT > 0 ist für einen günstigen Bewegungsübergang folgende Bedingung erforderlich:

- Konturelemente vor und nach #CAXTRACK ON [..] müssen zueinander tangentiale Linearsätze sein.

Hinweis

Eine aktive Nachführachse darf im Synchronbetrieb nicht als Slaveachse betrieben werden!

Programmierbeispiel

Automatische Achsnachführung (C-Achsnachführung)

```
Beispiel 1: Anwahl Achsnachführung
N10 G00 G90 X0 Y0 Z0 C0
N20 X5 Y5 C45        ;Gerade 45° zur X-Achse, Nachführachse C
                     ;parallel zur Kontur ausgerichtet
N20 #CAXTRACK ON [ANGLIMIT 3, OFFSET 0] ;Aktivierung der Achsnach-
                                        ;führung,Grenzwinkel 3°,
                                        ;Winkeloffset 0°
N30 X10 Y10          ;Primärer Bewegungssatz, C-Achse ist
                     ;bereits ausgerichtet
N40 X20              ;Winkel zum Vorhergehenden Satz: -45° >
                     ;Grenzwinkel -> Satz wird eingefügt: End-
                     ;position von C = 0
N50 M99 X30          ;Wenn M-Funktion Synchronisation vor
                     ;Bewegung -> Erst Synch. dann Bewegung C
                     ;auf 0, dann X auf 30.
                     ;Wenn Sync. Nach Satz-> Bewegung C auf 0
                     ;dann X auf 30, dann Sync.
N60 X40              ;Winkel C-Achse 0°
N70 X30              ;Winkel C-Achse 180°
N80 Y0               ;Winkel C-Achse -90°
N90 #CAXTRACK OFF    ;Deaktivierung der Achsnachführung
M30
```

```
Beispiel 2: Kopplung einer Slaveachse (C2) an Master-Nachführachse (C)
N20 G00 X0 Y0 Z0 C0 A0 C2=0 A2=0
N50 #SET AX LINK[1, C2=C]
N70 #ENABLE AX LINK[1]
N140 G01 X0 Y0 Z0 A0 C0 F2000
N170 #CAXTRACK ON [AX=C ANGLIMIT 0.1]
N190 LL SUB_1
N220 #CAXTRACK OFF
N250 #DISABLE AX LINK[1]
M30
```

12.18 Benutzerdefinierte Fehlerausgabe (#ERROR)

Der NC–Befehl #ERROR erlaubt die Ausgabe anwenderdefinierter Fehlermeldungen, die von der übergeordneten Bedienung (GUI = Graphical User Interface) weiterverarbeitet werden. Zusätzliche Parameter bieten die Möglichkeit zur genaueren Spezifizierung des Fehlers.

Die Zuordnung zwischen Fehlernummer (ID) und Fehlertext erfolgt in einer anwenderspezifischen Datei (FCT-M7// Ausgeben eigener Fehlermeldungen). Speicherort (Pfad) und Name dieser Datei werden im Parameter P-STUP-00169 eingetragen.

Syntax:

#ERROR [[**ID**=..] [**RC**=..] [**MID**=..] {**PV**<*i*>=..} {**PM**<*i*>=..} {**PIV**<*i*>=..} **]**

ID**=..**	Fehlernummer:
	1...1000: Der Zahlenwert bestimmt die auszugebende kundenspezifische Fehlernummer.
RC**=..**	Fehlerbehebungsklasse:
	0: Warnung, Kein Übergang in Fehlerzustand. Fortsetzung der Programmbearbeitung.
	2: Fehler, Übergang in Fehlerzustand. Per NC-RESET behebbar.
	7: Fataler Fehler, Übergang in Zustand 'Systemfehler'. Neustart der Steuerung erforderlich.
MID=..	Mehrfach-ID. Zähler dient als Unterscheidungsmerkmal, wenn in einem NC-Programm mehrmals der #ERROR–Befehl mit der gleichen Fehlernummer (ID) verwendet wird. MID muss eine positive Ganzzahl sein.
PV<*i*>=..	Es können maximal 5 (1 <= i <= 5) kundenspezifische Zahlenwerte (PV1...PV5) im Realformat in der Fehlermeldung mit ausgegeben werden.
	Ab V3.1.3080.14 bzw. V3.1.3107.48 können auch Strings ausgegeben werden, z.B. PV1="Test". Die maximale Länge ist auf 23 Zeichen beschränkt.
PM<*i*>=..	Die maximal 5 (1 <= i <= 5) PM-Parameter (PM1...PM5) dienen dazu, die Bedeutung der PV–Parameter genauer zu spezifizieren.
	0: IGNORE, Wert ohne Bedeutung
	1: Grenzwert
	2: Aktueller Wert
	3: Fehlerhafter Wert
	4: Erwarteter Wert
	5: Korrigierter Wert
	6: Logische Achsnummer
	7: Antriebstyp
	8: Logische Bedienelementnummer
	9: Zustand
	10: Transition
	11: Sender
	12: Klasse
	13: Instanz
	14: Identifikationsnummer
	15: Status
	16: Ringnummer
	17: Satznummer
	18: Unterer Grenzwert
	19: Oberer Grenzwert
	20: Startwert
	21: Endwert
PIV<*i*>=..	Die maximal 4 (1 <= i <= 4) PIV-Parameter (PIV1...PIV4) dienen zur Übermittlung zusätzlicher Informationen im Realformat.

Für nicht programmierte Parameter sind folgende Standardwerte gültig:

ID	1
RC	0
MID	0
PV1...PV5	0.0
PM1…PM5	1
PIV1...PIV4	0.0

Programmierbeispiel

Benutzerdefinierte Fehlerausgabe

```
; -------
; Ausgabe der Warnung mit ID 100, Mehrfachkennung 10
#ERROR [ID100 RC0 MID10]
; ..
; -------
; Ausgabe des Fehlers mit ID 455 mit Parametern
; Fehler 455 mit Parametern
; Parameter 1 - aktueller Wert ist 1
; Parameter 2 - fehlerhafte Wert ist 4.999
#ERROR [ID455 RC2 PV1=5 PV2=4.999 PM1=2 PM2=3]
; ..
; Fehlerausgabe mit String ab V3.1.3080.14
; Erwarteter Wert: Text-A
; Fehlerhafter Wert Text-B
#ERROR [ID123 RC2 PV1="Text-A" PM1=4 PV2="Text-B" PM2=3]
;…
; -------
; Fataler Fehler 999
#ERROR [ID999 RC7]
```

12.19 Zeitmessung (#TIMER)

Der NC-Befehl #TIMER bietet die Möglichkeit der Zeitmessung im NC-Programm. Die erfasste Zeit wird in der Einheit Millisekunden (ms) dargestellt.

Achtung

Die Timerzähler stehen **kanalübergreifend (global)** zur Verfügung. Dies ermöglicht z.B. die Messung von Signallaufzeiten zwischen Kanälen.

Bei parallelen unabhängigen Zeitmessungen in verschiedenen Kanälen ist darauf zu achten, unterschiedliche Zählernummern (ID's) zu verwenden. Ansonsten beeinflussen sich die Messungen gegenseitig!

Syntax:

#TIMER *<Aktion>* [*<Modus>*] **[ID**=..**]**

<Aktion>

Bestimmt die Aktion mit dem bezeichneten Zähler (ID).

START: Starten des Zählers (ID).

STOP**:** Stoppen des Zählers (ID).

READ**:** Auslesen des Zählers (ID).

Der Zählerstand wird gelatcht und in der zugeordneten **V.G.TIMER[ID]**-Variable in Millisekunden (ms) ab-gelegt.

CLEAR**:** Rücksetzen und Stoppen des Zählers (ID).

Die zugeordnete V.G.TIMER-Variable wird dabei **nicht gelöscht** sondern bleibt bis zu einer erneuten READ-Aktion des zugehörigen Zählers erhalten.

Achtung

Mit der Timerfunktion können Zeiten bis maximal 1193 Stunden erfasst werden.

<Modus>

Synchronisationsart:

---: Zeitmessung asynchron zum Interpolator auf Decodier-Ebene (Grundeinstellung). Die Zeitmessung beginnt direkt nach der Decodierung.

SYN: Zeitmessung auf Interpolator-Ebene. Der bezeichnete Zähler wird synchron zu Bearbeitungsvorgängen der NC–Maschine gesetzt. Beim synchronen Lesen (<SYN>) im Interpolator wird die Decodierung unterbrochen, bis der Zählerstand auf Decodier-Ebene in die Timervariable übernommen worden ist.

Hinweis

Zur Messung von Programmlaufzeiten müssen die Timer immer mit dem Schlüsselwort SYN verwendet werden.

ID=.. Zählernummer:

0...127: Es können maximal 128 kanalübergreifende Zähler programmiert werden. Pro Timerbefehl kann jedoch immer nur ein Zähler (ID) programmiert werden.

Programmierbeispiel

Zeitmessung

```
:
#FILE NAME[ MSG="C:\timer.txt" ] Dateiname für Zeitprotokollierung
:
#TIMER START [ID=10]                  Zähler 10 (Decodier-Ebene) starten
#TIMER START SYN [ID11]               Zähler 11 (Interpolator-Ebene)
starten
:
:
#TIMER READ [ID10]                    Zählerstand ablegen in V.G.TI-
MER[10]
#TIMER READ SYN [ID11]                Zählerstand ablegen in V.G.TI-
MER[11]
#MSG SAVE["T10 = %d",V.G.TIMER[10]]   Zählerstand in Datei protokollieren
#MSG SAVE["T11 = %d",V.G.TIMER[11]]   Zählerstand in Datei protokollieren
#TIMER STOP [ID10]                    Zähler 10 stoppen
#TIMER CLEAR [ID10]                   Zähler 10 zurücksetzen
:
:
#TIMER READ SYN [ID11]                Zählerstand ablegen in V.G.TI-
MER[11]
#MSG SAVE["T11 = %d",V.G.TIMER[11]]   Zählerstand in Datei protokollieren
:
:
#TIMER READ SYN [ID11]                Zählerstand ablegen in V.G.TI-
MER[11]
#MSG SAVE["T11 = %d",V.G.TIMER[11]]   Zählerstand in Datei protokollieren
:
:
#TIMER READ SYN [ID11]                Zählerstand ablegen in V.G.TI-
MER[11]
#MSG SAVE["T11 = %d",V.G.TIMER[11]]   Zählerstand in Datei protokollieren
#TIMER STOP SYN [ID11]                Zähler 11 stoppen
#TIMER CLEAR SYN [ID11]               Zähler 11 zurücksetzen
:
:
:
#TIMER START [ID=10, ID11]            Fehler, nur ein Zähler pro Timerbe-
fehl
                                      erlaubt!
:
:
```

12.20 Definition von Vorschubachsen (#FGROUP, #FGROUP ROT, #FGROUP WAXIS)

Mit #FGROUP wird festgelegt, auf welche Achsen sich der programmierte Vorschub (F-Wort) bezieht. Für die im Befehl #FGROUP programmierten Achsen wird eine Bahn im Raum bestimmt, die mit dem programmierten Vorschub abgefahren wird. Alle anderen Achsen werden wie Mitschleppachsen behandelt, erreichen also gleichzeitig mit den Bahnachsen ihr Ziel.

Eine Bahnachse ist dadurch gekennzeichnet, dass der zu verfahrende Weg beim Vorschub berücksichtigt wird. Dagegen hat der zu verfahrende Weg einer Mitschleppachse auf die Geschwindigkeit auf der Bahn keinen direkten Einfluss.

Syntax Festlegung von Vorschubachsen:

#FGROUP [<*Achsname*> {,<*Achsname*> } **]**

<*Achsname*> Namen der Achsen, die die Vorschubgruppe bilden.

Syntax Anwahl Grundeinstellung:

#FGROUP

Wenn keine Vorschubachsen programmiert sind, so ist die in den Kanalparametern P-CHAN-00096 und P-CHAN-00011 festgelegte Grundeinstellung gültig. Sind dort keine Vorschubachsen konfiguriert, so bilden automatisch alle Hauptachsen (Index 0, 1, 2) die Vorschubgruppe des Kanals. Dies wird durch die Meldung 21209 angezeigt.

Vorschubachsen sind bei Linearinterpolation beliebig definierbar.

Für Zirkular- und Polynominterpolation gelten folgende Ausnahmen:

- Bei der Zirkularinterpolation müssen alle Hauptachsen Vorschubachsen sein oder alle definierten Vorschubachsen müssen Mitschleppachsen sein.
- Bei der Polynominterpolation bilden alle Hauptachsen die Vorschubgruppe unabhängig vom #FGROUP-Befehl. Hiervon ausgenommen ist das Polynomüberschleifen im DIST_SOFT-Modus. Hier wirkt die programmierte #FGROUP.

Programmierbeispiel

Definition von Vorschubachsen

```
N10 #FGROUP [X,Y]          ;X und Y sind Vorschubachsen
N20 #FGROUP [X,Y,U,V]      ;X,Y,U und V sind Vorschubachsen
:
N50 #FGROUP [A]            ;Mitschleppachse A ist Vorschubachse
N60 #FGROUP [X,Y,U,V,A,B]  ;X,Y,U,V,A und B sind Vorschubachsen
:
N100 #FGROUP               ;Vorschubachsen gemäß Grundeinstellung
:                          ;in den Kanalparametern
N999 M30
```

In der Grundeinstellung werden alle mit #FGROUP festgelegten Vorschubachsen bei der Bestimmung des effektiven Vorschubes gleichgewichtet berücksichtigt. Durch die optionale Angabe eines Faktors bei der Festlegung der Vorschubachsen kann für jeder Achse eine Gewichtung und somit ihr Einfluss auf die Bestimmung des effektiven Vorschubes definiert werden. Die Erweiterung von #FGROUP ist nachfolgend beschrieben.

Syntax Festlegung von Vorschubachsen mit Vorschubfaktoren:

#FGROUP [<*Achsname*>=.. {,<*Achsname*>=..} **]**

<*Achsname*>=.. Namen der Achsen, die die Vorschubgruppe bilden mit zugehörigen Vorschubfaktoren. Wird bei einer Achse kein Faktor angegeben, so gilt der Faktor 1.0.

Wird eine Achse per Achstausch in den Kanal geholt, so wird ihr Vorschubfaktor mit 1.0 initialisiert.

Zulässiger Wertebereich für den Vorschubfaktor ist [0.001..1000]

Syntax Anwahl Grundeinstellung, alle Vorschubfaktoren auf 1.0:

#FGROUP

Programmierbeispiel

Definition von Vorschubachsen mit Vorschubgewichtung

```
N10 #FGROUP [X=0.5,Y=0.75] ;X und Y sind Vorschubachsen mit
                           ;geänderten Vorschubfaktoren
:
N50 #FGROUP [A=0.85]       ;Mitschleppachse A ist Vorschubachse
                           ;mit geändertem Vorschubfaktor
N60 #FGROUP [X,Y,U,V,A=0.85,B=1.7] ;X,Y,U,V sind Vorschubachsen
                                   ;mit Vorschubfaktor 1.0
                                   ,A und B sind Vorschubachsen
                                   ;mit geänderten Vorschubfaktoren
:
N100 #FGROUP           ;Alle Vorschubachsen mit Vorschubfaktor 1.0
:
N999 M30
```

Bei der Bearbeitung von zylindrischen Werkstücken auf einer rotatorischen Werkstückachse soll der real programmierte Vorschub [mm/min] am Eingriffspunkt des Werkzeugs wirksam sein. Dies kann zum einen durch Anwahl einer geeigneten kinematischen Transformation sichergestellt werden (z.B. Mantelflächentransformation) zum anderen durch Verwendung des Befehls #FGROUP ROT[...]. Nach Programmierung dieses Befehls wird der Vorschub der rotatorischen Achse in [°/min] in Abhängigkeit eines Bezugsradius umgerechnet. Bei Programmierung der rotatorischen Achse alleine oder gemeinsam mit Linearachsen ergibt sich am Bezugradius der geforderte programmierte Vorschub.

Syntax Vorschubanpassung bzgl. einer Rundachse:

#FGROUP ROT [AX=<*Achsname*> **REF**=.. **]**

AX=<*Achsname*>	Name der Achse, auf die der Bezugsradius wirken soll.
REF=..	Wirksamer Bezugsradius der rotatorischen Achse in [mm, inch].

Syntax Abwahl Vorschubanpassung bzgl. einer Rundachse:

#FGROUP

Achtung

Es wird nicht geprüft ob es sich bei der Achse "AX.." tatsächlich um eine Rundachse handelt! Die Funktion kann nur bei Vorschubsätzen (G01) und in Verbindung mit G94 verwendet werden!

Hinweis

Diese Funktion wird typischerweise bei der Fräsbearbeitung eingesetzt.

Vorschubanpassungen bei der Drehbearbeitung werden über G95 und G96 programmiert.

Programmierbeispiel

Werkstück mit Bezugradius R=10mm

```
N05 G00 C0
N10 G01 C180 F1000 ;Drehgeschwindigkeit des Werkstücks 1000 °/min
                   ;Vorschub am Werkstückumfang 174.67 mm/min

N20 #FGROUP ROT[AX=C REF=10]
N30 G01 C360 F1000 ;Vorschub am Umfang des Werkstücks 1000 mm/min
                   ;Drehgeschw. des Werkstücks 5727.6 °/min
:
Nxx #FGROUP ROT    (Abwahl)


:
N10 G00 X0 Y0 Z0
N15 #FGROUP ROT[AX=C REF=10] ;Vorschub am Fraesereingriffspunkt 1000 mm/
min

N60 G01 G91 X10 C57.325 F1000 ;Diagonale auf Mantelfläche
N70 G90 X0 C0
Nxx #FGROUP ROT               ;Abwahl
:
```

Mit #FGROUP WAXIS wird unabhängig von der Standardeinstellung in den Kanalparametern festgelegt, dass automatisch die Achse mit der längsten Fahrzeit („schwächste Achse“) als Vorschubachse mit dem programmierten Vorschub (F-Wort) bewegt wird. Alle anderen Achsen werden als Mitschleppachsen behandelt.

Syntax Anwahl „schwächste“ Achse als Vorschubachse:

#FGROUP WAXIS

Programmierbeispiel

```
N10 #FGROUP [X, Y]      ;X und Y sind Vorschubachsen
N20 G00 X0 Y0
N30 #FGROUP WAXIS       ;Schwächste Achse ist Vorschubachse
N40 G01 F1000 X10 Y200  ;Y-Achse ist Achse mit längster Fahrzeit
:
N999 M30
```

12.21 Anpassung der Bahndynamikgrenzwerte (#VECTOR LIMIT ON/ OFF)

Versionshinweis

Ab Version **V2.10.1507.02** ersetzt der Befehl **#VECTOR LIMIT ON/OFF...** die Befehle **#VECTORACC ON/OFF...** und **#VECTORVEL ON/OFF...** .Diese sind aus Kompatibilitätsgründen weiterhin verfügbar, es wird aber empfohlen, sie in neuen NC-Programmen nicht mehr zu verwenden.

Die maximal zulässige Geschwindigkeit, Beschleunigung und Verzögerung auf der Bahn ergibt sich in Abhängigkeit von den dynamischen Kenngrößen in den achsspezifischen Parameterlisten und der programmierten Kontur.

Bestimmte Anwendungen (z.B. Hochleistungslaserschneiden, Plasmastrahlschneiden) erfordern zur Gewährleistung optimaler Bearbeitungsergebnisse die Anpassung der dynamischen Kenngrößen auf der Bahn.

Diese Bahngrenzwerte können während der dynamischen Phasen der Bearbeitung durch den nachfolgenden NC-Befehl angepasst werden. Er ermöglicht im NC-Programm die Aktivierung/ Deaktivierung von selbst definierten Grenzwerten bzw. Standardgrenzwerten.

Zusätzlich ist es möglich, die auftretende Radialbeschleunigung sowie den Radialruck in gekrümmten Konturelementen (Polynome, Kreise) zu begrenzen.

Syntax Anwahl „schwächste" Achse als Vorschubachse:

#VECTOR LIMIT ON [[**ACC**=..] [**DEC**=..] [**RADIAL_ACC**=..] [**RADIAL_JERK**=..] [**JERK**=..]
[**TRANS_ACC**=..] [**VEL**=..] [**FEED**] [**RAPID**] { \ } **]**

oder mit Standardgrenzwerten

#VECTOR LIMIT ON [[**ACC**] [**DEC**] [**RADIAL_ACC**] [**RADIAL_JERK**] [**JERK**]
[**TRANS_ACC**] [**VEL**] [**FEED**] [**RAPID**] { \ } **]**

oder

#VECTOR LIMIT ON ALL

ACC=..	Grenzwert für die Bahnbeschleunigung in [mm/min², inch/min²]. Mit P-CHAN-00351 kann die Einheit auf [mm/sec², inch/sec²] umgeschaltet werden.
DEC=..	Grenzwert für die Bahnverzögerung in [mm/min², inch/min²]. Mit P-CHAN-00351 kann die Einheit auf [mm/sec², inch/sec²] umgeschaltet werden.

Hinweis

DEC ist wirksam, wenn das angewählte Slopeprofil eine getrennte Definition von Beschleunigungs- und Verzögerungsparametern zulässt (z.B. Slopetyp TRAPEZ [▶ 377]).

RADIAL_ACC=..	Grenzwert für die Radialbeschleunigung in [mm/min², inch/min²]. Mit P-CHAN-00351 kann die Einheit auf [mm/sec², inch/sec²] umgeschaltet werden. Die Programmierung dieses Parameters ist ab V2.11.2033.05 verfügbar.
RADIAL_JERK=..	Grenzwert für den Radialruck in [mm/min³, inch/min³]. Mit P-CHAN-00351 kann die Einheit auf [mm/sec³, inch/sec³] umgeschaltet werden. Die Programmierung dieses Parameters ist ab V3.1.3076.02 verfügbar.

JERK=..	Grenzwert für den Bahnruck. Die Einheit ist immer in [mm/sec³, inch/sec³] und kann nicht umgeschaltet werden. Die Programmierung dieses Parameters ist ab V3.1.3079.12 verfügbar.
TRANS_ACC=..	Grenzwert für die Beschleunigung im Satzübergang in [mm/min², inch/min²]. Mit P-CHAN-00351 kann die Einheit auf [mm/sec², inch/sec²] umgeschaltet werden. Die Programmierung dieses Parameters ist ab V3.1.3072.02 verfügbar.
VEL=..	Grenzwert für Geschwindigkeit in [mm/min, inch/min].
	Werden die Schlüsselworte ACC, DEC, RADIAL_ACC und VEL ohne Grenzwert programmiert oder wird der Befehl #VECTOR LIMIT ON ALL verwendet, gelten die Standardgrenzwerte aus der Kanalparameterliste (P-CHAN-00002, P-CHAN-00208, P-CHAN-00361, P-CHAN-00090). Für RADIAL_JERK, JERK und TRANS_ACC wird ein interner Grenzwert gesetzt.
	Der Befehl #VECTOR LIMIT OFF ... bewirkt das Umschalten auf die Berechnung der Dynamikbegrenzung durch den Look-Ahead. Die Umschaltung kann für bestimmte oder für alle Grenzwerte programmiert werden.
FEED	Dynamikbegrenzungen wirken nur auf Vorschubsätze (G01, G02, G03).
RAPID	Dynamikbegrenzungen wirken nur auf Eilgangsätze (G00).
	Sind die Schlüsselworte FEED und RAPID nicht programmiert oder beide gleichzeitig programmiert, wirken die Dynamikbegrenzungen auf alle Bewegungssätze (G00, G01, G02, G03).
\	Trennzeichen ("Backslash") für übersichtliche Programmierung des Befehls über mehrere Zeilen

```
Syntax Anpassung der dynamischen Bahngrenzwerte abwählen:
```

#VECTOR LIMIT OFF [[**ACC**] [**DEC**] [**RADIAL_ACC**] [**RADIAL_JERK**] [**JERK**]
[**TRANS_ACC**] [**VEL**] { \ } **]**

oder

#VECTOR LIMIT OFF ALL

Hinweis

Die Bahndynamikgrenzwerte werden nur verwendet, wenn sie unterhalb der im Look-Ahead gültigen Grenzwerte liegen. Sie haben keinen Einfluss auf achsspezifische Bewegungen wie z.B. Referenzpunktfahrt, Handbetrieb oder unabhängige Achsen und wirken sowohl bei G01 als auch bei G00.

Achtung

Im Zusammenhang mit dem Kanalparameter P-CHAN-00097 muss der Anwender berücksichtigen, dass die Maschine abhängig vom programmierten Grenzwert bei einem Feedhold auch langsamer verzögert.

Hinweis

Die Bahndynamikgrenzwerte bei G00 können auch über eine Gewichtungstabelle mit Fahrwegabhängigkeit in den Kanalparametern [1] [▶ 894]-7 beeinflusst werden.

Programmierbeispiel

Anpassung der Bahndynamikgrenzwerte

```
%vec_limit
(Dynamikgrenzwerte mit Wert setzen)
N10 #VECTOR LIMIT ON [ACC=3600000 DEC=4000000 VEL=3000 FEED]
N11 #VECTOR LIMIT ON [ACC=3600000 DEC=4000000 TRANS_ACC=3000000]
N12 #VECTOR LIMIT ON [ACC=3600000 VEL=3000 RADIAL_ACC=2000000 RAPID]
N13 #VECTOR LIMIT ON [ACC=3600000]
N14 #VECTOR LIMIT ON [DEC=4000000 VEL=3000 FEED RAPID]
N15 #VECTOR LIMIT ON [DEC=4000000]
N16 #VECTOR LIMIT ON [VEL=3000]

(Dynamikgrenzwerte auf Standard setzen)
N20 #VECTOR LIMIT ON [ACC DEC RADIAL_ACC TRANS_ACC VEL RAPID]
N21 #VECTOR LIMIT ON [ACC DEC RADIAL_JERK JERK]
N22 #VECTOR LIMIT ON [ACC VEL FEED]
N23 #VECTOR LIMIT ON [ACC]
N24 #VECTOR LIMIT ON [DEC VEL FEED RAPID]
N25 #VECTOR LIMIT ON [DEC]
N26 #VECTOR LIMIT ON [VEL]

(Gemischte Belegung der Dynamikgrenzwerte)
N27 #VECTOR LIMIT ON [ACC=3600000 DEC]
N28 #VECTOR LIMIT ON [ACC VEL=3000]

(Alle Dynamikgrenzwerte auf Standard setzen)
N30 #VECTOR LIMIT ON ALL
(:= #VECTOR LIMIT ON [ACC DEC RADIAL_ACC RADIAL_JERK JERK TRANS_ACC VEL
FEED RAPID])

(Dynamikgrenzwerte durch LOOK_AHEAD bestimmen lassen)
N40 #VECTOR LIMIT OFF [ACC DEC VEL]
N41 #VECTOR LIMIT OFF [ACC DEC]
N42 #VECTOR LIMIT OFF [ACC VEL]
N43 #VECTOR LIMIT OFF [ACC]
N44 #VECTOR LIMIT OFF [DEC VEL]
N45 #VECTOR LIMIT OFF [DEC]
N46 #VECTOR LIMIT OFF [VEL]

(Alle Dynamikgrenzwerte durch LOOK_AHEAD bestimmen lassen)
N50 #VECTOR LIMIT OFF ALL
(:= #VECTOR LIMIT OFF [ACC DEC RADIAL_ACC TRANS_ACC VEL])
N999 M30
```

12.22 Festlegen einer minimalen Satzübergangsgeschwindigkeit (#TRANSVELMIN ON/OFF)

An unstetigen Satzübergängen (Konturknicken) wird die Bahngeschwindigkeit soweit reduziert, dass die dabei auftretenden Achsbeschleunigungen die eingestellen Werte in den Achsparametern nicht überschreiten. Aus technologischer Sicht (z.B. beim Brennschneiden) kann diese Geschwindigkeitsabsenkung an Konturknicken unerwünscht sein. In diesem Fall kann über den nachfolgenden Befehl eine Minimalgeschwindigkeit angegeben werden, die an Konturknicken nicht unterschritten werden darf.

Achtung

Die im Befehl programmierte Minimalgeschwindigkeit wirkt nur bei unstetigen Satzübergängen bei Zirkular und Linearsätzen. Im NC-Programm darf somit kein Überschleifverfahren oder Spline angewählt sein (z.B. G261, #HSC, G151).

Syntax:

#TRANSVELMIN ON [**[** <*min_vel*> **]**]
#TRANSVELMIN OFF

<*min_vel*> Minimale Satzübergangsgeschwindigkeit in [mm/min, inch/min].

Wenn nach #TRANSVELMIN ON kein Grenzwert programmiert ist, wird 0 als minimale Übergangsgeschwindigkeit ausgegeben. Der Befehl #TRANSVELMIN OFF bewirkt das Umschalten auf die freie Berechnung der Geschwindigkeitsbegrenzung durch den Look Ahead.

Die Funktion wirkt bei linearem und nichtlinearem Slope, es darf aber bei nichtlinearem Slope nicht die Ruckbegrenzung an unstetigen Satzübergängen über Kanalparameter P-CHAN-00009 (corr_v_trans_jerk=1) aktiviert sein. In diesem Fall ist die Ruckbegrenzung priorisiert.

Die programmierte Minimalgeschwindigkeit ist nur bis Programmende gültig. Sie wird beim nächsten Programmstart bzw. Reset abgenullt.

Eine Vorbelegung über die Kanalparameterliste ist nicht möglich.

12.23 Schreiben von Maschinendaten (#MACHINE DATA)

Versionshinweis

Die Verfügbarkeit dieser Funktionalität ist von der Konfiguration und dem Versionsumfang abhängig.

Dieser Befehl ermöglicht die Änderung von achsspezifischen Parametern im NC-Programm. Die neuen Werte sind programmübergreifend gültig. Sie werden durch eine Listenaktualisierung im Hochlauf wieder überschrieben.

Hinweis

Eine evtl. aktive Bahninterpolation wird gestoppt, bis der neue Parameterwert übernommen und wirksam ist. Evtl. drehende Spindeln werden **nicht** angehalten.

Syntax:

#MACHINE DATA [*<Modus>*] **[AX**=*<Name>* | **AXNR**=.. *<Param_ID>* =.. | *<Param_ID>{.<idx>}<value>* |
AXPARAM "*<string>***"** [**WAIT**] **]**

<Modus>	Synchronisationsart ---: Synchronisation auf Decoder-Ebene (Grundeinstellung). SYN: Synchronisation auf Interpolator-(Echtzeit) Ebene.
AX=*<Name>*	Name der Bahnachse oder Spindel, bei der ein Parameter neu geschrieben werden soll.
AXNR=..	Logische Achsnummer der Bahnachse oder Spindel, bei der ein Parameter neu geschrieben werden soll. Positive Ganzzahl.

Achtung

Es erfolgt keine Plausibilitätsprüfung bzgl. der logischen Achsnummer. Für die richtige Angabe ist ausschließlich der Bediener verantwortlich. Die Änderung eines beliebigen Parameterwertes mit #MACHINE DATA bewirkt die erneute Übernahme der kompletten Achsparameterliste in den NC-Kanal. Zuvor über andere NC-Befehle geänderte Achsparameterwerte (z.B. Softwareendschalter über G98/G99) werden überschrieben und sind im NC-Kanal nicht mehr gültig.

<Param_ID>=..	Achsparameter in ISG-Schreibweise (P-AXIS-xxxxx) mit neuem Wert in der Einheit der Achsparameterliste [AXIS].
<Param_ID>{.<idx>}<value>	Bei Achsparametern, die ein Array adressieren, kann das entsprechende Element durch die erweiterte Angabe von Punkt und Index geschrieben werden (z.B. P-AXIS-00209.0).

Hinweis

Eine Slaveachse in einem Hard-Gantry-Verbund kann nur über die logische Achsnummer angesprochen werden.

Folgende Achsparameter sind über vordefinierte Schlüsselworte (Param_ID) verfügbar:

Param_ID	Bedeutung
P-AXIS-00001	Nichtlineares Geschwindigkeitsprofil: Achsbeschleunigung bei Geschwindigkeitszunahme
P-AXIS-00002	Nichtlineares Geschwindigkeitsprofil: Achsbeschleunigung bei Geschwindigkeitsabnahme
P-AXIS-00004	Beschleunigung im Eilgang (G00)
P-AXIS-00005	Lineares Geschwindigkeitsprofil: Beschleunigung der Stufe 1 im Eilgang (G00)
P-AXIS-00006	Lineares Geschwindigkeitsprofil: Beschleunigung der Stufe 2 im Eilgang (G00)
P-AXIS-00008	Zulässige Achsdynamik: Maximal zulässige Achsbeschleunigung
P-AXIS-00011	Lineares Geschwindigkeitsprofil: Beschleunigung der Stufe 1 im Vorschubbetrieb (G01, G02, G03)
P-AXIS-00012	Lineares Geschwindigkeitsprofil: Beschleunigung der Stufe 2 im Vorschubbetrieb (G01, G02, G03)
P-AXIS-00045	Minimale Distanz (Sicherheitsabstand) zwischen zwei Kollisionsachsen
P-AXIS-00056	Maximale Abweichung nach Abwahl Nachführbetrieb
P-AXIS-00075	Gantrybetrieb: Korrekturgeschwindigkeit zum Ausfahren des statischen Offsets zwischen der Master- und der Slaveachse bei Differenzen der Messsysteme
P-AXIS-00099	Verstärkungsfaktor k_v für P-Lageregelung
P-AXIS-00103	Größe der Lose
P-AXIS-00109	Maximal zulässiger Geschwindigkeitsoverride bei unabhängigen Achsen und bei Spindeln
P-AXIS-00151	Maximal zulässige Einschwingzeit zur Erreichung des Genauhaltfensters
P-AXIS-00152	Absolutposition des Referenzpunktes
P-AXIS-00166	Bleibende Abweichung bei nichtlinearer Schleppabstandsüberwachung
P-AXIS-00167	Faktor zur Parametrierung der dyn. Schleppabstandsüberwachung
P-AXIS-00168	Maximaler Schleppabstand
P-AXIS-00169	Minimaler Schleppabstand
P-AXIS-00172	Art der Schleppabstandsüberwachung
P-AXIS-00195	Nichtlineares Geschwindigkeitsprofil: Rampenzeit für Beschleunigungsabbau
P-AXIS-00196	Nichtlineares Geschwindigkeitsprofil: Rampenzeit für Beschleunigungsaufbau

P-AXIS-00197	Nichtlineares Geschwindigkeitsprofil: Rampenzeit für Verzögerungsabbau
P-AXIS-00198	Nichtlineares Geschwindigkeitsprofil: Rampenzeit für Verzögerungsaufbau
P-AXIS-00200	Nichtlineares Geschwindigkeitsprofil: Rampenzeit im Eilgang (G00)
P-AXIS-00201	Minimal zulässige Rampenzeit des Antriebs zur Begrenzung des Achsruckes
P-AXIS-00208	Maximale Geschwindigkeit der Ausgleichsbewegung nach Abwahl des Nach-führbe-triebs
P-AXIS-00209	Eilganggeschwindigkeit G00
P-AXIS-00211	Lineares Geschwindigkeitsprofil: Umschaltgeschwindigkeit im Eilgang (G00) zwischen Geschwindigkeits-rampe 1 und Geschwindigkeitsrampe 2
P-AXIS-00212	Zulässige Achsdynamik: Maximal zulässige Achsgeschwindigkeit
P-AXIS-00216	Minimal zulässige Achsgeschwindigkeit bei Spindeln. Unterhalb dieser Geschwindigkeit liefert die Drehzahlüberwachung im Lageregler "Drehzahl null"
P-AXIS-00217	Faktor zur Berechnung der Istdrehzahl, bei der der Zustand "Drehzahl erreicht" gemel-det wird.
P-AXIS-00218	Minimale Referenzpunktfahrtgeschwindigkeit
P-AXIS-00219	Maximale Referenzpunktfahrtgeschwindigkeit
P-AXIS-00221	Lineares Geschwindigkeitsprofil: Umschaltgeschwindigkeit im Vorschubbetrieb (G01, G02, G03) zwischen Geschwindig-keitsrampe 1 und Geschwindigkeitsrampe 2
P-AXIS-00236	Größe des Regelfensters für Genauhalt
P-AXIS-00414	Maximaler Positionsoffset bei Abstandsregelung

AXPARAM "<*String*>" Alternative Schreibweise: Achsparameter mit komplettem Strukturpfad und Wert in der internen Schreibweise der Achsparameterliste [AXIS] (siehe Beispiel). Es können **alle** Achsparameter geschrieben werden.

WAIT Dieses Schlüsselwort darf nur in Verbindung mit einem synchronisierten Setzen eines Achsparameters verwendet werden (SYN). Bei programmiertem WAIT wird zusätzlich die Programmdecodierung unterbrochen (implizites FLUSH) und gewartet, bis im NC-Kanal der neue Parameterwert übernommen und wirksam ist.

Programmierbeispiel

Schreiben von Maschinendaten

```
N10 G00 X100                  (Positionierung mit V-Eilgang gemäß)
                              (Standardeinstellung nach Hochlauf)

N20 #MACHINE DATA SYN [AX=X P-AXIS-00209=80000]    (Neue V-Eilgang)

N30 ...      (Im weiteren Programmverlauf gilt die neue V-Eilgang).
:

Alternativ mit Achslistenschreibweise:
:
N20 #MACHINE DATA SYN [AX=X AXPARAM="getriebe[0].vb_eilgang 80000"]
                                                    (Neue V-Eilgang)
:

Für Spindelachse:
Alternativ mit Achslistenschreibweise:
N20 #MACHINE DATA SYN [AX=S P-AXIS-00109=1200]   (Neuer V-Override)


Setzen eines Softwareendschalters mit Warten im Kanal:
N20 #MACHINE DATA SYN [AX=X AXPARAM="kenngr.swe_pos 15000000" WAIT]


Setzen der Eilganggeschwindigkeit für Getriebestufe 1 mit Index:
N20 #MACHINE DATA SYN [AX=X P-AXIS-00209.1=80000]
```

12.24 Dateioperationen

12.24.1 Definition von Dateinamen (#FILE NAME)

Mit dem Befehl #FILE NAME können im NC-Programm Dateinamen definiert werden, die von speziellen NC-Funktionen z.B. zur Anlage einer Ausgabedatei oder zum Aufruf eines NC-Programms verwendet werden.

Syntax:

#FILE NAME [<*File_ID*>="<*Dateiname*>" { <*File_ID*>="<*Dateiname*>" } **]**

<*File_ID*>

MSG: Name der Ausgabedatei für #MSG SAVE [▶ 373].

Standardname ist "*message.txt*".

M6: Name des globalen Unterprogramms bei implizitem Programmaufruf über M6 im NC-Programm. Der Name ist bis M30 gültig. M6 wird nicht mehr als M-Funktion behandelt!

Der Standardname wird in P-CHAN-00118 gesetzt. M6 wird immer als letzte Aktion am Satzende ausgeführt.

D: Name des globalen Unterprogramms bei implizitem Programmaufruf durch ein D-Wort im NC-Programm. Der Name ist bis M30 gültig.

Der Standardname wird in P-CHAN-00429 gesetzt. Der Unterprogrammaufruf wird immer als letzte Aktion am Satzende ausgeführt.

G80 – G89: Namen der globalen Unterprogramme bei impliziten Programmaufrufen über G80 - G89 im NC-Programm. Die Namen sind bis M30 gültig.

Die Standardnamen werden in P-CHAN-00160 - P-CHAN-00169 gesetzt. G80 - G89 wird immer als letzte Aktion am Satzende ausgeführt.

G800- G819 bzw. **G800- G839**:** Namen der globalen Unterprogramme bei zusätzlichen impliziten Programmaufrufen über G800 - G819** im NC-Programm. Die Namen sind bis M30 gültig.

Die Standardnamen werden in P-CHAN-00187 gesetzt. G800 - G819** wird immer als letzte Aktion am Satzende ausgeführt.

** Erweitert auf 40 Aufrufe (G800 – G839) ab V3.1.3079.23

Das Gleichheitszeichen (=) ist optional.

Die Dateinamen können jederzeit im NC-Programm mit #FILE NAME definiert bzw. geändert werden.

Bei RESET und bei Programmstart werden die Dateinamen auf ihren Standardnamen zurückgesetzt.

Programmierbeispiel

Definition von Dateinamen

```
%example1
N10 #FILE NAME[ MSG="prog_flow.txt" ]
N20 $IF V.E.PLC_START_HOME == 1
N30 G74 X1 Y2 Z3
N40 #MSG SAVE["Homing executed"]   ;Ausgabe in prog_flow.txt
N50 $ENDIF
;…
N120 #MSG SAVE["Roughing OK"]
;…
N950 #MSG SAVE["Finishing OK"]
N985 V.E.WP_CNTR = V.E.WP_CNTR+1
N990 #MSG SAVE["Workpiece No. %d OK", V.E.WP_CNTR]
M1000 M30


%example2
N10 #FILE NAME[ M6="tool_change.nc" ]
N20 G00 X100 Y100 Z0
N30 M6          ; Aufruf Werkzeugwechselprogramm tool_change.nc
;…
M1000 M30


%example3
N10 #FILE NAME[ D="tool_data.nc" ]
N20 G00 X100 Y100 Z50
N30 D1          ;1:Daten anfordern von Werkzeug 1
                ;2:Impliziten Aufruf tool_data.nc
;…
M1000 M30


%example4
N10 #FILE NAME[ G80="g80_up_test.nc" ]
N20 G00 X100 Y100 Z50
N30 G80         ; Aufruf Unterprogramm g80_up_test.nc
;…
M1000 M30


%example5
N10 #FILE NAME[ G800="g800_up_test.nc" G815="g815_up_test.nc"]
N20 G00 X100 Y100 Z50
N30 G800         ; Aufruf Unterprogramm g800_up_test.nc
;…
N90 G815         ; Aufruf Unterprogramm g815_up_test.nc
;…
M1000 M30
```

12.24.2 Umbenennen einer Datei (#FILE RENAME)

Der Befehl #FILE RENAME dient zum Umbenennen einer bereits vorhandenen Datei. Alle Parameter müssen zwingend angegeben werden. Das Weglassen eines Parameters führt zu einer entsprechenden Fehlermeldung.

Syntax:

#FILE RENAME [PATHOLD="<*Dateiname*>" PATHNEW="<*Dateiname*>" OVRMODE=..]

PATHOLD= “<*Dateiname*>“	Datei, die umbenannt werden soll, ggf. mit Verzeichnisangabe. Existiert dieses Verzeichnis bzw. die Datei nicht, wird das Programm mit einer Fehlermeldung abgebrochen.
PATHNEW= “<*Dateiname*>“	(Ziel)Datei, in die umbenannt werden soll, ggf. mit Verzeichnisangabe. Wird hier ein anderes Verzeichnis als bei PATHOLD angegeben, wird die Datei mit dem neuen Namen in das Verzeichnis verschoben, sofern dieses existiert. Ist das Verzeichnis nicht vorhanden, wird eine entsprechende Fehlermeldung ausgegeben.
OVRMODE=..	Bool’scher Wert, der angibt ob eine durch PATHNEW angegebene Datei überschrieben werden soll, wenn diese bereits vorhanden ist. 0: Datei darf nicht überschrieben werden. Ausgabe einer Fehlermeldung. 1: Bereits vorhandene Datei darf überschrieben werden.

Das Gleichheitszeichen (=) ist optional.

Hinweis

Der Anwender muss in den Verzeichnissen PATHNEW und PATHOLD Schreibrechte haben, sonst führt das Umbenennen zu einem Fehler.

Achtung

SCHREIBSCHUTZ:

Ein Fehler wird erzeugt, wenn die Datei, die umbenannt werden soll, schreibgeschützt ist oder die (Ziel)Datei bereits existiert und schreibgeschützt ist.

Achtung

RELATIVE VERZEICHNISSE:

Ist bei PATHOLD nur der Dateiname angegeben, wird die Datei in den Ordnern der Hochlauf/Kanalparameterliste gesucht.

Gesucht wird in der Reihenfolge Hauptprogramm - Unterprogramm - Arbeitsverzeichnis.

Ist PATHNEW relativ angegeben, so wird das Verzeichnis von PATHOLD verwendet.

Hinweis

Unter TwinCAT wird zusätzlich das Standardverzeichnis für die Dateioperation verwendet. Dieses wird im System Manager eingestellt.

Programmierbeispiel

Umbenennen einer Datei

```
%FileRename

N10 #FILE NAME[MSG="C:\Test.txt"] ;Datei anlegen
...
N40 #MSG SAVE["Write me into file"] ;Schreibt Text in die Datei

N60 #FILE RENAME[PATHOLD="C:\Test.txt" PATHNEW = "C:\NewName.txt" OVRMO-
DE=1]
N70 M30
```

12.24.3 Löschen einer Datei (#FILE DELETE)

Der Befehl #FILE DELETE dient zum Löschen einer Datei. Der Parameter muss zwingend angegeben werden. Ansonsten erfolgt die Ausgabe einer entsprechenden Fehlermeldung.

Syntax:

#FILE DELETE [PATH="*<Dateiname>***"]**

PATH= “*<Dateiname>*“	Datei, die gelöscht werden soll, mit Verzeichnisangabe.

Das Gleichheitszeichen (=) ist optional.

Hinweis

Der Anwender muss im Verzeichnis PATH Schreibrechte haben um die Datei erfolgreich Löschen zu können.

Achtung

SCHREIBSCHUTZ:

Ist die Datei schreibgeschützt wird der Fehler mit der ID 21627 erzeugt.

Achtung

RELATIVE VERZEICHNISSE:

Ist Parameter PATH relativ angegeben, wird die Datei in den Ordnern der Hochlauf/Kanalparameterliste gesucht.

Gesucht wird in der Reihenfolge Hauptprogramm - Unterprogramm - Arbeitsverzeichnis.

Hinweis

Unter TwinCAT wird zusätzlich das Standardverzeichnis für die Dateioperation verwendet. Dieses wird im System Manager eingestellt.

Programmierbeispiel

Löschen einer Datei

```
%FileDelete

N10 #FILE NAME[MSG="C:\Test.txt"] ;Datei anlegen
...
N40 #MSG SAVE["Write me into file"] ;Schreibt Text in die Datei
…
N60 #FILE DELETE[PATH="C:\Test.txt" ] ;Datei löschen
N70 M30
```

12.24.4 Prüfen der Existenz einer Datei (#FILE EXIST)

Mit dem Befehl #FILE EXIST wird geprüft, ob eine Datei auf dem Dateisystem vorhanden ist und geöffnet werden kann. Nach dem Aufruf von #FILE EXIST wird das Suchergebnis in V.G.-Variablen abgelegt. Es bleibt bis zum nächsten Aufruf von #FILE EXIST erhalten.

V.G.<var_name>	**Bedeutung**	**Datentyp**	**Einheit der Ein/ Ausgabe**	**Erlaubter Zugriff** Lesen / Schreiben
FILE_EXIST	Ergebnis der Dateisuche. Wenn Datei gefunden wurde, dann 1	Boolean	0 , 1	L
FILE_EXIST_PATH_NAME *	Pfad mit Dateiname *	String	-	L
FILE_EXIST_DIRECTORY *	Pfad ohne Dateiname. Der Pfad endet mit '\' *	String	-	L

* Pfade, welche in der Tabelle der Suchpfade in der Hochlaufliste (P-STUP-00018) oder Kanalliste (P-CHAN-00401) eingetragen sind.

Syntax:

#FILE EXIST [PATH="<*Dateiname*>" **]**

PATH= “<*Dateiname*>“ Name der Datei, deren Existenz geprüft werden soll, mit oder ohne Verzeichnisangabe.

Das Gleichheitszeichen (=) ist optional.

Achtung

RELATIVE VERZEICHNISSE:

Ist PATH relativ angegeben, wird die Datei in den Ordnern der Hochlauf/Kanalparameterliste gesucht.

Gesucht wird in der Reihenfolge Hauptprogramm - Unterprogramm - Arbeitsverzeichnis.

Programmierbeispiel

Prüfen der Existenz einer Datei

```
%FileExist

N010 #FILE EXIST[PATH = "C:\TestDir\test.nc"]

N030 $IF V.G.FILE_EXIST == TRUE
N040 #MSG ["FILE EXISTS"]
N050 $ELSE
N060 #MSG ["FILE DOES NOT EXIST"]
N070 $ENDIF

N090 M30
```

12.24.5 Anlegen und Verwalten von Backup-Dateien

Daten protokollieren

Im NC-Programm kann der Anwender mit #MSG SAVE […] Texte in eine Datei schreiben. Hierbei werden neue Daten an die Datei angehängt, wodurch die Dateigröße stetig ansteigt.

Größe und Anzahl der Protokolldateien festlegen

Um die Größe und den Zugriff bei grösser werdenden Dateien zu kontrollieren, kann der Anwender die maximale Anzahl der Zeilen der Datei beschränken. Wird diese maximale Größe erreicht, so wird automatisch ein Backup der Datei angelegt und die ursprüngliche Datei gelöscht.

Für die Verfolgung des zeitlichen Verlaufs der Dateiänderung können mehrere Backups (Protokolldateien) gespeichert werden. Der Anwender kann dazu die maximale Anzahl der Backups definieren. Ist die maximale Anzahl der Backupdateien erreicht, so wird die älteste Backupdatei verworfen.

Die Größe und die Anzahl der Backupdateien wird mit dem Befehl #FILE NAME […] festgelegt und bezieht sich auf die Datei, in die mit #MSG SAVE geschrieben wird:

`Syntax:`

#FILE NAME [BACKUP_LINES_MAX=.. **BACKUP_COUNT_MAX**=.. **]**

BACKUP_LINES_MAX=..	Anzahl der Zeilen in der Datei seit Steuerungshochlauf. Wird diese überschritten, so wird ein Backup angelegt. Dies gilt für die mit <*Anzahl*> zuletzt programmierten Dateien.
BACKUP_COUNT_MAX=..	Maximale Anzahl der parallel gehaltenen Backups. Ist beim Anlegen eines neuen Backups die maximale Anzahl erreicht, so wird das älteste Backup überschrieben.

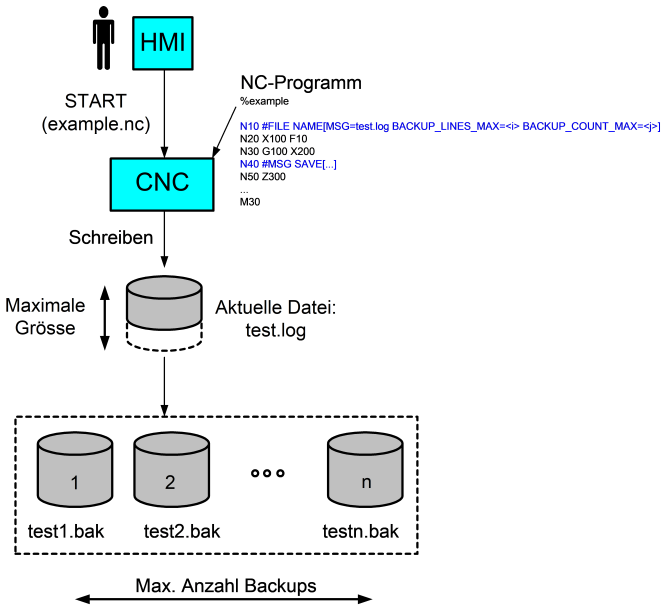


Abb. 117: Schema der Backupfunktion

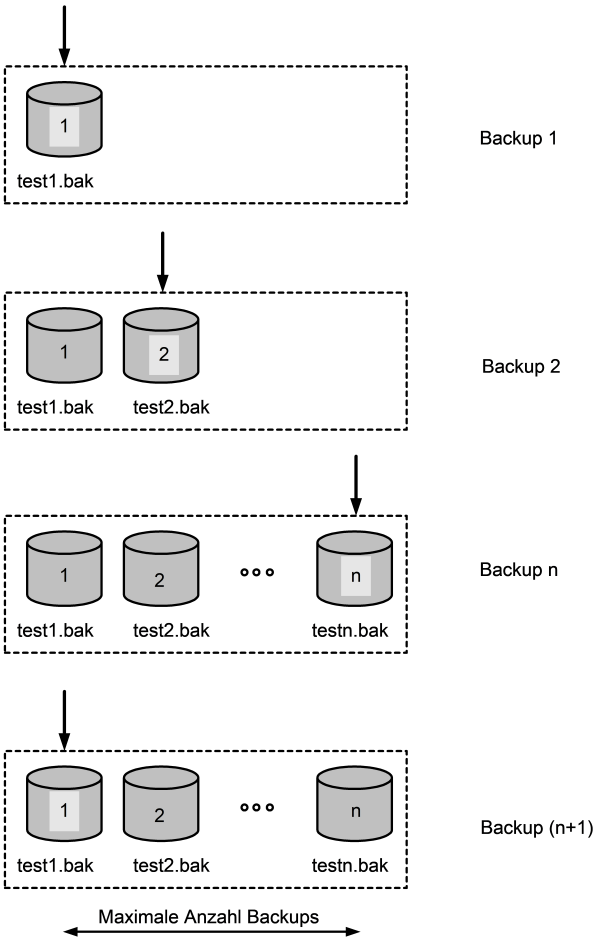


Abb. 118: Anlegen der Backupdateien

Aktivieren und Deaktivieren der Backupfunktion

Die Backupfunktion wird bei Angabe einer maximalen Zeilenanzahl > 0 und einer maximalen Dateianzahl > 0 aktiviert.

Die Deaktivierung erfolgt, wenn einer dieser Parameter mit 0 programmiert wird oder keiner dieser Parameter programmiert wird oder bei Programmende M30.

Hinweis

Aktuell entspricht die maximale Zeilenanzahl der Anzahl der Schreibzugriffe per #MSG SAVE -Befehl. D.h. es wird nicht die tatsächliche Zahl der Zeilen ermittelt. Insbesondere, wenn eine mehrzeilige Message geschrieben wird, ergibt sich hier ein Unterschied.

Der Originalname der Backupdatei wird über #FILE NAME [MSG=<name>] angegeben. Der Name der Backups setzt sich aus diesem Originalnamen, der Zählnummer <nr> und der Dateierweiterung .bak zusammen.

Beispiel: Test.log (Originaldatei)
Test1.bak, Test2.bak, Test3.bak, usw.

Verhalten bei Steuerungsneustart

Beim Start der Steuerung können Backupdateien bereits vorhanden sein. Beim ersten Anlegen eines neuen Backups erfolgt keine Prüfung, welches Backup das älteste ist. D.h. nach Steuerungsstart wird ein neues Backup immer ab 1 angefangen.

Hinweis

Das Löschen eines Backups findet nur statt, wenn dies beim Anlegen eines neuen Backups erforderlich ist.

Beim Start der Steuerung, der Definition des Backupverhaltens über den NC-Befehl, im Fehlerfall oder bei NC-Reset wird kein Backup automatisch gelöscht!

Der Name der letzten Backupdatei sowie die aktuelle Zeile werden gespeichert. Somit kann bei erneuter Aktivierung der Backupfunktion immer auf dem aktuellen Stand aufgesetzt werden. Ein Löschen dieser Daten findet nur bei Steuerungsneustart statt. Alle anderen Werte für zuvor programmierte Dateinamen werden verworfen.

12.25 Arbeits-/ Schutzraumüberwachung

Durch Kontrollbereiche kann verhindert werden, dass bestimmte Achskonstellationen zu einer Kollision des Werkzeuges mit zu schützenden Teilen führen. Ein Kontrollbereich kann ein Arbeits- oder ein Schutzbereich sein.

Kontrollbereiche können dreidimensional in zylindrischer oder polygonaler Form definiert werden. Die dritte Dimension wird durch konstante Minimal- und Maximalwerte festgelegt. Die Zuordnung der Achsen ist abhängig von der aktuell ausgewählten Arbeitsebene (z.B. G17).

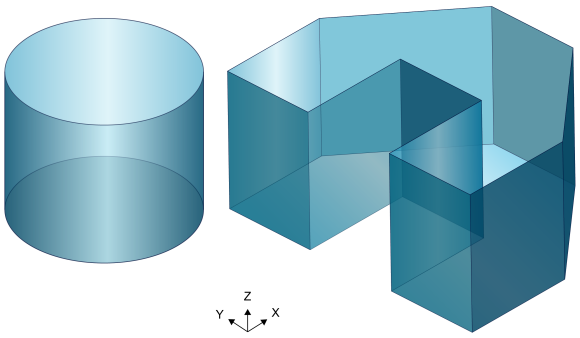


Abb. 119: Definition von 3D Kontrollbereichen (Zylindrisch, Polygonal)

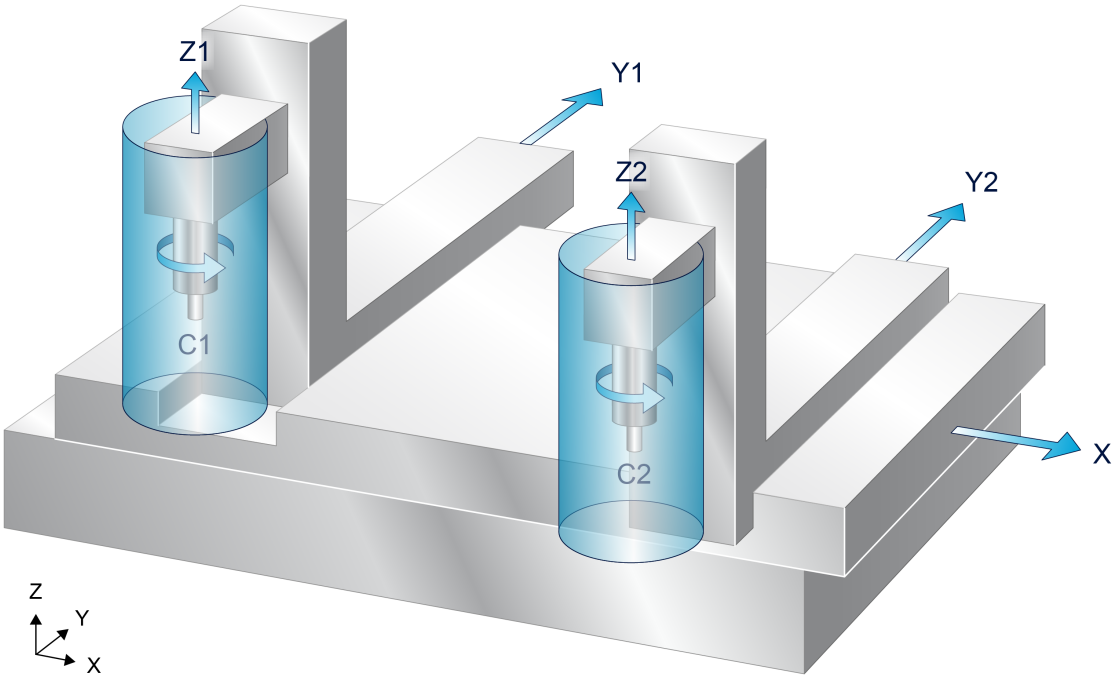


Abb. 120: Beispiel zylindrischer Arbeitsräume einer Applikation

Aus einem Arbeitsbereich kann nicht herausgefahren und in einen Schutzbereich nicht hineingefahren werden. Die aktuellen Werkzeugdaten werden mit einbezogen, so dass die Werkzeugspitze auf ihre Gültigkeit im Kontrollbereich geprüft werden kann.

Arbeits- oder Schutzbereiche werden im NC Programm definiert, aktiviert und deaktiviert, wobei auch mehrere Kontrollbereiche gleichzeitig aktiv sein können. Kontrollbereiche können im deaktivierten Zustand auch neu definiert werden.

Die Arbeits-/Schutzraumüberwachung mit Überwachung des Werkzeugmittelpunktes ist möglich bei

- Automatikbetrieb in Verbindung mit:
  - Linearen Bewegungssätzen
  - Zirkularen Bewegungssätzen (gleich welcher Orientierung G17/G18/ G19).
  - Kinematischen Transformationen
  - Polynomkonturen (Überschleifen, HSC)
  - Helikalen Bewegungen
  - Bezugspunktverschiebungen mit G92, G54
  - Kartesischen Transformationen #(A)CS ab CNC-Version V2.11.2015

- aktivem Handbetrieb ab CNC-Version V3.1.3068.9 in Verbindung mit:
  - Exklusivem (G200) oder inklusivem Modus (G201/G202)
  - Kinematischen Transformationen

12.25.1 Definition eines Kontrollbereiches (#CONTROL AREA BEGIN/END)

Zeitpunkt der Definition

Bei Hochlauf der Steuerung sind keine Arbeits-/ Schutzräume vordefiniert. Eine Definition in den Konfigurationslisten ist nicht möglich.

Arbeits-/ Schutzräume werden ausschließlich direkt im NC-Programm in einer von Klartextbefehlen eingeschlossenen Sequenz von Verfahrbewegungen definiert.

Die Verfahrbewegungen sind hierbei immer absolut zu programmieren. Die Kontur des Kontrollbereiches in der Ebene wird entweder über ein durch Linearsätze beliebig gebildetes geschlossenes Polygon (Endpunkt der Satzfolge muss gleich dem Startpunkt sein) oder einen Vollkreis definiert. Die Ausdehnung in der dritten Dimension sowie weitere Merkmale des Kontrollraumes werden im zugehörigen Klartextbefehl mit festgelegt.

Syntax Beginn einer Kontrollbereichsdefinition:

#CONTROL AREA BEGIN [ID=.. WORK | **PROT POLY** | **CIRC MIN_EXCUR**=.. **MAX_EXCUR**=..
[**EXCUR_AX**=.. |**EXCUR_AXNR**=..] [**MONITOR_LVL**=..] **]**

ID=..	Identifikationsnummer des Kontrollbereiches (ID). Die Definition ist über Programmende und RESET hinaus gültig. Es können bis zu 20 Kontrollbereiche definiert werden.
WORK	Kontrollbereich ist ein Arbeitsraum.
PROT	Kontrollbereich ist ein Schutzraum.
POLY	Kontur des Kontrollbereichs wird über einen Polygonzug definiert.
CIRC	Kontur des Kontrollbereichs wird über einen Vollkreis definiert.

MIN_EXCUR=..	Begrenzung des Kontrollbereiches in der dritten Dimension in negativer Richtung in [mm, inch].
MAX_EXCUR=..	Begrenzung des Kontrollbereiches in der dritten Dimension in positiver Richtung in [mm, inch].
EXCUR_AX=..	Optionale Angabe eines Achsbezeichners für die dritte Richtung der Ausdehnung des Arbeits- oder Schutzraums (ab CNC-Version V2.11.2025.00). Standardmäßig wird die dritte Hauptachse verwendet.
EXCUR_AXNR=..	Optionale Angabe einer logischen Achsnummer für die dritte Richtung der Ausdehnung des Arbeits- oder Schutzraums (ab CNC-Version V2.11.2025.00). Standardmäßig wird die dritte Hauptachse verwendet.
MONITOR_LVL=..	Angabe der Überwachungsebene, siehe Überwachungsebenen (ab CNC-Version V3.1.3079.42) IMCS: Zwischenkoordinatensystem (nur bei mehrstufigen Transformationen sinnvoll) MCS: Maschinenkoordinatensystem (Standard)

Syntax Ende einer Kontrollbereichsdefinition:

#CONTROL AREA END

Jeder Kontrollbereich muss mit #CONTROL AREA END abgeschlossen werden. Erst danach können weitere Kontrollbereiche definiert werden.

Hinweis

Ein nicht aktiver Überwachungsraum kann durch erneutes Programmieren der gleichen ID überschrieben werden.

Achtung

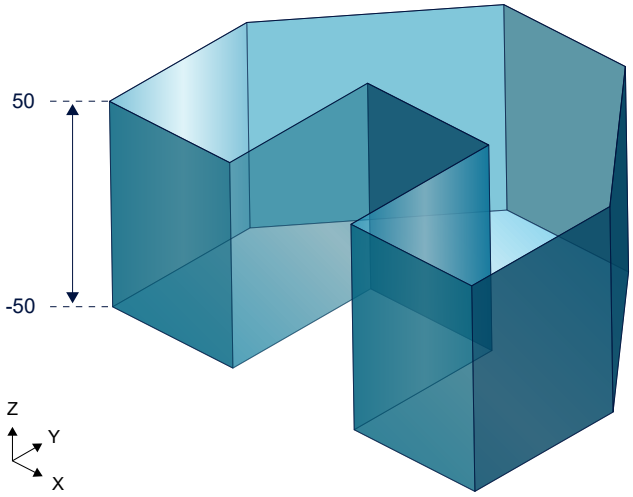
Bei der Definition der Überwachungsräume werden aktive kartesische Transformationen #(A)CS nicht berücksichtigt. Die Angabe der Arbeits- und Schutzräume erfolgt immer kartesisch im MCS-Koordinatensystem.

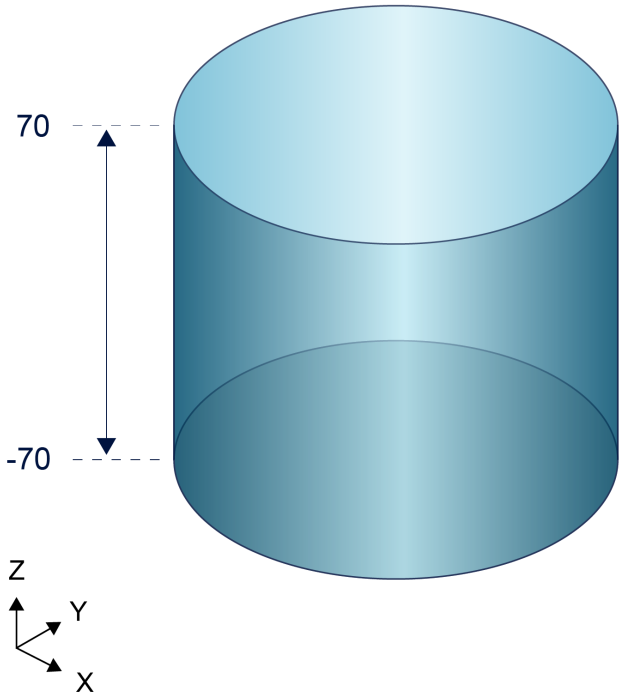
Programmierbeispiel

Definition eines Kontrollbereiches

```
(Definition eines polygonförmigen Arbeitsraumes:
:
N10 #CONTROL AREA BEGIN [ID1 WORK POLY MIN_EXCUR=-50 MAX_EXCUR=50]
N20 G01 F1000 G90 X-150 Y75 (Startpunkt)
N30 X-50  Y150
N40 X50   Y150
N50 X150  Y75
N60 X150  Y0
N70 X50   Y0
N80 X50   Y75
N90 X-50  Y75
N100 X-50  Y0
```

```
N120 X-150 Y0
N130 X-150 Y75   (Endpunkt identisch mit Startpunkt)
N140 #CONTROL AREA END
:
(Definition eines zylindrischen Schutzraumes:
:
N210 #CONTROL AREA BEGIN [ID2 PROT CIRC MIN_EXCUR=-70 MAX_EXCUR=70]
N220 G01 X100 Y0 F10000 (Startpunkt für zylindrischen Schutzraum)
N230 G02 G162 I50 J0
N240 #CONTROL AREA END
:
```





12.25.2 Kontrollbereiche an-/abwählen (#CONTROL AREA ON/OFF)

Beim Aktivieren von Arbeitsräumen muss sich der TCP bereits im gültigen Arbeitsraum befinden. Ebenso darf beim Aktivieren eines Schutzraumes der TCP auf der aktuellen Position keine Verletzung hervorrufen.

Syntax Anwahl eines Kontrollbereiches:

#CONTROL AREA ON [**ALL**] | **[ID**=.. **]**

ID=..	Identifikationsnummer des Kontrollbereiches (ID).
ALL	Alle derzeit definierten Kontrollbereiche aktivieren.

Programmierbeispiel

Kontrollbereiche an-/abwählen

```
#CONTROL AREA ON [ID3] (Aktiviere spezifischen Kontrollbereich)
#CONTROL AREA ON ALL (Aktiviere alle definierten Kontrollbereiche)
```

Syntax Abwahl eines Kontrollbereiches:

#CONTROL AREA OFF [**ALL**] | **[ID**=..**]**

ID=..	Identifikationsnummer des Kontrollbereiches (ID).
ALL	Alle derzeit definierten Kontrollbereiche deaktivieren.

Programmierbeispiel

```
#CONTROL AREA OFF [ID3]    (Deaktiviere spezifischen Kontrollbereich)
#CONTROL AREA OFF          (Deaktiviere zuletzt angewählten Kontrollbe-
reich)
#CONTROL AREA OFF ALL      (Deaktiviere alle aktiven Kontrollbereiche)
```

12.25.3 Kontrollbereiche löschen (#CONTROL AREA CLEAR)

Syntax Löschen eines Kontrollbereiches:
#CONTROL AREA CLEAR [**ALL**] | **[ID**=..**]**

ID=..	Identifikationsnummer des Kontrollbereiches (ID).
ALL	Alle derzeit definierten Kontrollbereiche löschen.

Programmierbeispiel

Kontrollbereiche löschen

```
#CONTROL AREA CLEAR [ID3] (Lösche spezifischen Kontrollbereich)
#CONTROL AREA CLEAR ALL   (Lösche alle definierten Kontrollbereiche)
```

12.25.4 Überwachung zusätzlicher Achsen

Neben den Hauptachsen. X, Y, Z können weitere Achsen in die Überwachung von Arbeits- und Schutzräumen aufgenommen werden. Hierbei ist die Definition der zugeordneten Kontrollbereiche auf Polygonzüge beschränkt. Die Definition der Kontrollbereiche erfolgt über die entsprechenden Achsbezeichner.

Programmierbeispiel

Definition eines Arbeitsraumes für die zusätzliche Achsen X2, Y2 und Z2

```
:
N10 #CONTROL AREA BEGIN [ID4 WORK POLY MIN_EXCUR=-50 MAX_EXCUR=50]
N20 G01 F1000 G90 X2=100 Y2=100   ;Startpunkt
N30 X2=-100
N40 Y2=-100
N50 X2=100
N60 X2=100 Y2=100   ;Endpunkt identisch mit Startpunkt
N70 #CONTROL AREA END
:
N500 #CONTROL AREA ON  ALL
:
N1000 M30
```

12.25.5 Besonderheiten beim Handbetrieb

Die Überwachung wird im Handbetrieb in Echzeitteil der CNC auf Basis der definierten und aktivierten Überwachungsräume durchgeführt.

Die Fehlerreaktion ist identisch zur ACS Begrenzung bzw. dem Auflaufen auf die Handbetriebsoffsetgrenzen. Wird eine IMCS / MCS Grenze erreicht stoppen alle Achsen ohne Fehler vor der Grenze.

Der Beginn des Bremsvorganges vor der Grenze ist abhängig von Handbetriebsgeschwindigkeit und -beschleunigung.

Ausgabe einer Warnmeldung

Durch die Ausgabe einer Warnmeldung kann dem Anwender der Grund für den Bewegungsstopp angezeigt werden. Dazu muss P-MANU-00014 gesetzt sein.

Exklusiver Handbetrieb (G200)

Als Reaktion auf einen Fehler erfolgt beim Automatikbetrieb die Ausgabe einer Fehlermeldung und ein Programmabbruch. Im Gegensatz dazu erfolgt beim Handbetrieb vor Eintritt in den Schutzraum bzw. Verlassen des Arbeitsraums als Fehlerreaktion nur ein Bewegungsstopp der Achse(n).

Inklusiver Handbetrieb (G201/G202)

Bei der Überlagerung von Bewegungen aus dem Automatik- und Handbetrieb (parallele Interpolation) kann es zu Verletzungen von Arbeitsräumen und Schutzräumen kommen.

Unterdrückung der Arbeitsraumüberwachung

Über den Parameter P-CHAN-00442 kann die Arbeitsraumüberwachung im Handbetrieb beeinflusst oder gar unterdrückt werden.

Beispiel

Unterdrückung Arbeitsraumüberwachung mit P-CHAN-00442

Ausgangslage für alle Fälle:
Es wird im Automatik- oder Handbetrieb gefahren. Vor Aktivierung des Handbetriebs wird die Arbeitsraumüberwachung z.B. über ein Unterprogramm aktiviert.

Fall 1:

P-CHAN-00442 ist mit 1 belegt.
Keine aktive Arbeitsraumüberwachung im Handbetrieb obwohl Definition und Aktivierung im NC-Programm durchgeführt wurde. Es kann beliebig über die Arbeitsraumgrenzen hinaus und zurück gefahren werden.

Fall 2:

P-CHAN-00442 ist mit 0 belegt.

Aktive Arbeitsraumüberwachung im Handbetrieb in Verbindung mit der Control Unit Unterdrückung Fehlerausgabe in Arbeitsraumüberwachung im Handbetrieb.

Bei aktivem Handbetrieb kann die Arbeitsraumüberwachung über das gesetzte Signal der der Control Unit Unterdrückung Fehlerausgabe in Arbeitsraumüberwachung im Handbetrieb deaktiviert werden. Es kann dann über die Arbeitsraumgrenzen hinaus- und zurückgefahren werden.

Bei Aktivierung des Handbetriebs muss der TCP innerhalb des erlaubten Bereichs stehen, ist dies nicht der Fall wird ein Fehler ausgegeben.

Fehler ID 50961, wenn der Arbeitsraum verlassen wurde

Fehler ID 50962, wenn der Schutzraum verlassen wurde.

Fall 3:

P-CHAN-00442 ist mit 2 belegt.

Aktivierung der Arbeitsraumüberwachung im Handbetrieb in Verbindung mit der Control Unit Unterdrückung Fehlerausgabe in Arbeitsraumüberwachung im Handbetrieb.

Bei Aktivierung des Handbetriebs darf der TCP außerhalb des erlaubten Bereichs stehen, es erfolgt keine Prüfung bezüglich der Position des TCPs. Die Arbeitsraumüberwachung kann mit der Control Unit deaktiviert werden, es kann in den erlaubten Bereich zurückgefahren werden.

12.26 Beeinflussung des Vorwärts-/Rückwärtsfahrbetriebes

Versionshinweis

Die Verfügbarkeit dieser Funktionalität ist von der Konfiguration und dem Versionsumfang abhängig.

Weitere Informationen zu diesen Befehlen und zum Thema Vorwärts-/ Rückwärtsfahren auf der Bahn siehe Funktionsbeschreibung [FCT-C7].

12.26.1 Ausblenden von Programmsequenzen (#OPTIONAL EXECUTION)

Im NC-Programm kann über den Programmierbefehl #OPTIONAL EXECUTION ON/OFF Sequenz markiert werden, die beim Rückwärtsfahren oder simulierten Fahren ausgelassen werden soll.

Aktiviert wird das Auslassen über die SPS. Der markierte Programmabschnitt wird ausgelassen, wenn

- Rückwärtsrichtung aktiv ist (Control Unit „Rückwärtsfahren“)
- Oder bei simulierter Fahrt (Control Unit „Simuliertes Fahren“)

Der markierte Bereich wird dann auf Interpolatorebene ausgelassen. Es erfolgt jedoch keine Neuberechnung von Übergangsbedingungen zwischen den Sätzen vor und nach dem ausgelassenen Bereich.

Hinweis

Für die Nutzung der Funktionalität muss der P-STUP-00033 parametriert sein

Syntax:

#OPTIONAL EXECUTION [[**ON**] [**[SIMULATE** | **SIMULATE MASK**=.. **]**]] | **OFF**

ON	Ausblenden aktivieren
OFF	Ausblenden deaktivieren
Nachfolgende Syntax ist verfügbar ab CNC-Version V3.3107.12	
SIMULATE	Die programmierte Sequenz zwischen #OPTIONAL EXECUTION ON und OFF wird nur ausgelassen, wenn das Signal der Control Unit „Simuliertes Fahren“ aktiv ist.
SIMULATE MASK=..	64-Bit Maske für die Angabe. Die Sequenz zwischen #OPTIONAL EXECUTION ON und OFF wird nur ausgelassen, wenn das Signal der Control Unit „Simuliertes Fahren“ aktiv ist und die programmierte Maske mit der Maske der Control Unit „Maske für simulierte Fahrt“ bitweise Übereinstimmungen aufweist.

Um einen diskontinuierlichen Übergang von Weg, Geschwindigkeit und Beschleunigung zwischen diesen Sätzen zu vermeiden, muss der Interpolatorkontext insbesondere in Bezug auf Achspositionen unverändert sein.

Hinweis

Achspositionen müssen vor und nach der auszublendenden Sequenz identisch sein.

Bei veränderten Achspositionen wird der Fehler ID 50452 ausgegeben.

Synchronisation von M-/ H-Funktionen bei #OPTIONAL EXECUTION

Die mit #OPTIONAL EXECUTION ON/OFF markierten Sequenzen werden nur bei aktivem Rückwärtsfahren oder bei „Simuliertem Fahren“ ausgelassen. Eine Ausgabe der M/H-Funktionen erfolgt nicht.

Das Verhalten/ die Möglichkeiten bei M- oder H-Funktionen **außerhalb** der Sequenz werden in [FCT-C7//„M-/ H-Funktion Handshake mit der SPS“] beschrieben.

Bei Verwendung des Befehls #OPTIONAL EXECUTION ON [SIMULATE] muss für das Auslassen der Sequenz zwingend die Control Unit „Simuliertes Fahren“ aktiv sein. Ein Auslassen der Sequenz bei Rückwärtsfahrt erfordert somit ein aktives Rückwärtsfahren und „Simuliertes Fahren“.

Programmierbeispiel

Ausblenden von Programmsequenzen

```
%t_storag.nc
X10 Y0
N10 G91 G00 X10 F1000

N11 #OPTIONAL EXECUTION ON
N12 Z123
N13 S1000 M3
N14 Z-123
N15 M101
N16 #OPTIONAL EXECUTION OFF

N20 G90 G01 X0
N30 G02 I10
N40 G03 J10
M30
```

Die CNC prüft und überwacht nur den kontinuierlichen Positionsverlauf der Achsen mit oder ohne Ausblenden von Sätzen. Die Einhaltung aller anderen Bedingungen muss vom Anwender gewährleistet sein, da diese von der CNC nicht geprüft werden.

Die Schachtelung von ausgeblendeten Bereichen wird nicht berücksichtigt.

Ein OPTIONAL EXECUTION muss vor dem Verlassen (M17, M29) der Programmebene, in der es angewählt wurde (ON), auch wieder abgewählt (OFF) werden. Dies gilt auch beim Verlassen der Hauptprogrammebene (M30). Wird die Programmebene ohne die Abwahl verlassen, so wird der Fehler ID 21719 ausgegeben.

Es können nur komplette Bereiche ausgelassen werden. Erfolgt bei Programmausführung innerhalb eines OPTIONAL EXECUTION-Bereichs die Aktivierung von Rückwärtsfahren oder „Simuliertem Fahren“, so wird der Bereich nicht ausgelassen.

Hinweis

Der NC-Befehl #OPTIONAL EXECUTION ist mit konturverändernden Funktionalitäten wie z.B. Werkzeugradiuskorrektur oder Polynomüberschleifen nicht sinnvoll nutzbar.

Programmierbeispiel

Ausblenden von Programmsequenzen mit „SIMULATE MASK“

Im nachfolgenden NC-Programm werden 3 auszublendende Sequenzen markiert, die jeweils mit einer Kennung in Form einer binären Bitmaske versehen sind. Diese Sequenzen werden jeweils nur bei aktiver Control Unit „Simuliertem Fahren“ ausgelassen, wenn zusätzlich die programmierte Maske mit der Maske der Control Unit „Maske für das simulierte Fahren“ bitweise Übereinstimmungen aufweist.

```
N010 X10 Y0
N020 G91 G00 X10 F1000
N030 #OPTIONAL EXECUTION ON [SIMULATE MASK='2#000001']
N040 X20
N050 M3
N060 X0
N070 M101
N080 #OPTIONAL EXECUTION OFF

N090 #OPTIONAL EXECUTION ON [SIMULATE MASK='2#000010']
N100 X30
N110 M3
N120 X0
N130 M102
N140 #OPTIONAL EXECUTION OFF

N150 #OPTIONAL EXECUTION ON [SIMULATE MASK='2#000100']
N160 X40
N170 M3
N180 X0
N190 M103
N200 #OPTIONAL EXECUTION OFF

N210 X50
N220 X0
N230 M30
```

12.26.2 Rückwärtsfahrspeicher löschen (#BACKWARD STORAGE CLEAR)

Durch den NC-Befehl #BACKWARD STORAGE CLEAR kann der seitherige Rückwärtsfahrspeicher explizit gelöscht werden. Hierdurch wird sichergestellt, dass nach Überfahren dieser Programmposition angehalten wird.

Syntax:

#BACKWARD STORAGE CLEAR

Programmierbeispiel

Rückwärtsfahrspeicher löschen

```
%backward-storage

N000 G01 X0 F10000
N010 X100 Y123
N020 X100
N030 X200 Y10
N040 X300 Y20

N050 #BACKWARD STORAGE CLEAR

N060 X400 Y-20
N070 X500 Y-3

N060 #BACKWARD STORAGE CLEAR

N080 X444 Y10
N090 X333 Y3
N100 X222 Y10
N110 X111 Y3
N120 X000 Y10
N130 X-111 Y3

N140 #BACKWARD STORAGE CLEAR

N1000 M30
```

12.27 Werkzeugwechsel bei aktivem Synchronbetrieb (#FREE TOOL CHANGE)

Generell ist die Übernahme neuer Werkzeugdaten (T (mit implizitem D), D bzw. #TOOL DATA) bei aktivem Synchronbetrieb aus Sicherheitsgründen nicht zulässig und wird durch die Fehlermeldung 20169 angezeigt. Diese Verriegelung ist durch den nachfolgenden NC-Befehl schaltbar.

Syntax:

#FREE TOOL CHANGE ON | OFF

Der NC-Befehl muss spätestens vor dem ersten, im aktiven Synchronbetrieb angewählten Werkzeugwechsel programmiert sein. Die Aufhebung der Verriegelung erfolgt mit dem Schlüsselwort "ON".

Die Reaktivierung der Verriegelung von Synchronbetrieb und Werkzeugwechsel wird mit "OFF" erreicht. Weiterhin wird der NC-Befehl bei RESET und implizit bei Hauptprogrammende abgewählt.

Bei Programmstart ist die Verriegelung von Synchronbetrieb und Werkzeugwechsel aus Kompatibilitäts- und Sicherheitsgründen ebenfalls immer aktiv (Default).

Hinweis

Ist gleichzeitig der Kanalparameter P-CHAN-00100 (move_tool_offsets_directly) aktiv, so wird die Fehlermeldung 20169 weiterhin erzeugt, da das direkte Ausfahren der Werkzeugoffsets in den Slaveachsen bei aktivem Synchronbetrieb sonst zu Maschinenschäden führen könnte. D.h. bei Verwendung von #FREE TOOL CHANGE sollte P-CHAN-00100 deaktiviert sein.

Programmierbeispiel

Werkzeugwechsel bei aktivem Synchronbetrieb

```
%TOOL_AXLINK
N05 X0 Y0 Z0 C100 G53 D0
N10 #AX LINK [1,C=X]
N15 #FREE TOOL CHANGE ON
N20 #AX LINK ON [1]
N30 X100 Y50 Z30
N40 D2
N50 X200 Y75 Z40
N60 D1
N65 X300 Y100
N70 #AX LINK OFF [1]
N75 #FREE TOOL CHANGE OFF
N70 X25 Y25 Z25 C25
N80 M30
```

12.28 Sperren von Programmbereichen für den Einzelschrittbetrieb (#SINGLE STEP)

Im NC-Programm können durch den Befehl #SINGLE STEP [DISABLE / ENABLE] beliebige Programmbereiche für den Einzelschrittbetrieb gesperrt werden. Dadurch kann der Bediener diese markierten Programmbereiche in einem Schritt übergehen und so schneller zu den zu untersuchenden NC-Zeilen springen.

Bei einer Schachtelung von gesperrten Bereichen umfasst die Einzelschrittsperre den Bereich von der ersten Aktivierung bis zur ersten Deaktivierung (siehe Beispiel 2).

Weiterhin kann der Anwender mit einer Satznummernspezifischen Einzelschrittauflösung #SINGLE STEP [RESOLUTION…] eine Schrittweite definieren, bei der der Einzelschrittstopp wirken soll (siehe Beispiel 3).

Weitere Informationen zum Einzelschrittbetrieb siehe Funktionsbeschreibung [FCT-M2].

Syntax:

#SINGLE STEP [DISABLE | **ENABLE** | **RESOLUTION**=.. **]**

DISABLE	Beginn der Einzelschrittsperre.
ENABLE	Ende der Einzelschrittsperre.
RESOLUTION=..	Satznummernspezifische Schrittweite, bei der der Einzelschrittstopp wirkt.

Programmierbeispiel

Sperren von Programmbereichen für den Einzelschrittbetrieb

Beispiel 1:

Im Bereich der NC-Sätze N40–N100 einschließlich der darin aufgerufenen Unterprogramme ist der Einzelschrittbetrieb nicht wirksam.

```
%SINGLE_STEP
N10 X0 Y0 Z0
N20 X10
N30 Y10
N40 #SINGLE STEP [DISABLE]
N50 X20
N60 Y20
N65 L GSP.nc
N70 Z20
N80 X30
N90 Z30
N100 #SINGLE STEP [ENABLE]
N110 Y30
N120 X40
N130 Z40
N999 M30
```

Beispiel 2:

Bereich der Einzelschrittsperre bei Schachtelung umfasst N40-N75

```
%SINGLE_STEP
N10 X0 Y0 Z0
N20 X10
N30 Y10
N40 #SINGLE STEP [DISABLE]
N50 X20
N55 #SINGLE STEP [DISABLE]
N60 Y20
N65 L GSP.nc
N70 Z20
N75 #SINGLE STEP [ENABLE]
N80 X30
N90 Z30
N100 #SINGLE STEP [ENABLE]
N110 Y30
N120 X40
N130 Z40
N999 M30
```

Beispiel 3:

Satznummerierung mit anwenderspezifischer Einzelschrittauflösung (10er Schritte) und interner Nummerierung (1er Schrittweite). Die schwarzen Linien stellen den Einzelschrittstopp dar. Im grauen Bereich wird nicht angehalten.

```
%SINGLE_STEP

N010 #SINGLE STEP [RESOLUTION = 10]

N090 Y0
N091 Y1
N092 Y2
N093 Y3
N094 Y4

N100 Y5
N101 Y6
N102 Y7

N110 Y8

...
```

12.29 Programmierbarer Bahnoverride (#OVERRIDE)

Mit dieser Funktion kann der Bahnvorschub im NC-Programm, wenn erforderlich für Vorschub und Eilgangsätze unterschiedlich, beeinflusst werden. Der programmierte Overridewert ist bei Bahnbewegungen wirksam, wenn sich mindestens eine Achse bewegt. Die Wirkungsweise der Echtzeitbeeinflussungen des Vorschubs über die PLC bleibt davon unberührt.

Als weitere Funktion steht auch ein programmierbarer Achsoverride [▶ 842] zur Verfügung.

Ist für eine, an der Bahnbewegung, beteiligte Achse zusätzlich auch ein Achsoverride definiert, so ergibt sich der wirksame Bahnoverride aus der Multiplikation der beiden Overridewerte.

Hinweis

Die Funktion G166 [▶ 187] unterdrückt die Wirkung der programmierten Overridewerte.

Syntax:

#OVERRIDE [FEED_FACT=.. **RAPID_FACT**=.. **]**

FEED_FACT=..	Overridefaktor für Vorschubsätze [0,1%-200%].
RAPID_FACT=..	Overridefaktor für Eilgangsätze [0,1%-200%].

Programmierbeispiel

Programmierbarer Bahnoverride

```
%path_override

N10 G00 G90 X0 Y0 Z0
   (Bahnoverride G01 122.765%, G00 155.7%)
N20 #OVERRIDE [FEED_FACT=122.765 RAPID_FACT=155.7]
N30 G01 X100 Y100 Z100 F1000              Wirksamer Vorschub=1227.65
   (Bahnoverride G01, G00 100%)
N40 #OVERRIDE [FEED_FACT=100 RAPID_FACT=100]
N50 G01 X200 Y200 Z200 F1000              Wirksamer Vorschub=1000

M30
```

12.30 Antriebsunabhängiges Schalten von Antriebsfunktionen

Im Gegensatz zu den SERCOS-spezifischen Befehlen gemäß Kapitel Schreiben und Lesen von Antriebsparametern und Kommandos [▶ 379] ermöglichen die nachfolgenden Befehle das **antriebsunabhängige** Setzen von Antriebsparametern. Die Parameter können ausschließlich synchron geschrieben werden, d.h. die Ausführung des Befehles erfolgt zum Zeitpunkt der Bearbeitung auf Interpolatorebene. In der Grundeinstellung erfolgt die Ausführung des #DRIVE-Befehls parallel zur weiteren Abarbeitung des NC-Programms, d.h. die Programmabarbeitung wird nicht angehalten. Um die Interpolation bis zum erfolgreichen Schalten der Antriebsfunktion anzuhalten, kann der Parameter „WAIT“ benutzt werden.

Zur Prüfung des erfolgreichen Schaltens der Antriebsfunktion zu einem **späteren** Zeitpunkt kann der Befehl #DRIVE WAIT SYN verwendet werden.

12.30.1 Synchrones Schreiben (#DRIVE WR SYN)

Syntax:

#DRIVE WR SYN [AX=<*Achsname*> | **AXNR**=.. [**MOTOR**=..] [**PARAM_SET**=..] **KEY=**<*string*> **VAL**=.. [**WAIT**] **]**

AX=<*Achsname*>	Name der (Antriebs-)achse
AXNR=..	Logische Achsnummer der (Antriebs-)achse, Positive Ganzzahl
MOTOR=..	Im Antriebsverstärker zu wählende Motornummer bei Motorumschaltung; Positive Ganzzahl
PARAM_SET=..	Umzuschaltender Parametersatz im Antriebsverstärker; Positive Ganzzahl
KEY=<*string*>	Kennung der anzusprechenden Funktion gemäß [2] [▶ 894]
VAL=..	Zu übertragender Wert, Negative oder positive Ganzzahl
WAIT	Anhalten der Interpolation und Warten auf erfolgreiches Schalten der Antriebsfunktion. Ohne Angabe dieses Schlüsselwortes kann zur Prüfung des erfolgreichen Schaltens der Antriebsfunktion im weiteren Verlauf des NC-Programms ein #DRIVE WAIT SYN programmiert werden.

Achtung

Für Spindeln darf während aktivem Endlosdrehen keine Motorumschaltung und keine Parametersatzumschaltung (MOTOR=<expr>, PARAM_SET=<expr>) programmiert werden.

12.30.2 Synchrones Warten auf Quittierung (#DRIVE WAIT SYN)

Mit dem folgenden Befehl wird geprüft, ob vorhergehende #DRIVE WR SYN für eine Achse abgeschlossen sind. Der Interpolator wird angehalten, bis alle #DRIVE WR SYN im Antrieb ausgeführt wurden. Dies gilt sowohl für Bahnachsen als auch für Spindelachsen.

Syntax:

#DRIVE WAIT SYN [AX=<*Achsname*> | **AXNR**=.. **SWITCH_OK]**

AX=<*Achsname*>	Name der (Antriebs-)achse
AXNR=..	Logische Achsnummer der (Antriebs-)achse, Positive Ganzzahl
SWITCH_OK	Prüfen, ob alle vorhergehenden #DRIVE WR SYN abgeschlossen sind.

Programmierbeispiel

Synchrones Warten auf Quittierung

Synchrones Schreiben mit sofortigem Warten auf Quittierung:

```
%TOOL_AXLINK1
N05 X0 Y0 Z0
N10 #DRIVE WR SYN [AX=X MOTOR=2 PARAM_SET=4 KEY=Drehmomentgrenze VAL=400
WAIT]
N20 X100 Y50 Z30 G01 F3000
N30 X200 Y75 Z40
N65 X300 Y100
N70 X25 Y25 Z25 C25
Nxx
N80 M30
```

Synchrones Schreiben mit spaeterem Warten auf Quittierung:

```
%TOOL_AXLINK2
N05 X0 Y0 Z0
N10 #DRIVE WR SYN [AX=X MOTOR=2 PARAM_SET=4 KEY=Drehmomentgrenze
VAL=400]
N20 X100 Y50 Z30 G01 F3000
N30 X200 Y75 Z40
N60 #DRIVE WAIT SYN [AX=X SWITCH_OK]
N65 X300 Y100
N70 X25 Y25 Z25 C25
Nxx
N80 M30
```

12.31 Geschwindigkeitsoptimierte Bewegungsführung durch Segmentierung (#SEGMENTATION)

Für bestimmte Applikationen (z.B. Kinematiken, bei denen singuläre Bereiche auftreten können) kann es von Vorteil sein, die programmierte Satzaufteilung CNC-seitig zu verfeinern bzw. Kreissätze (G2/G3) durch Segmentierung in Linearsätze (G1) umzuwandeln. Zusätzlich können Kreissätze auch in Kreissegmente zerlegt werden um die Ausnutzung der Maschinendynamik zu verbessern. Dies kann durch folgenden Befehl bereichsweise im NC-Programm erreicht werden.

Syntax:

#SEGMENTATION [**ON** | **OFF**] [**ALL**] [**[** [**LIN**] [**LENGTH**=..] [**CIR**] [**OPMODE**=..] [**PARAM**=..] **]**]

ON	Segmentierung aktivieren
OFF	Segmentierung deaktivieren
ALL	Segmentierung von Linear- und Zirkularsätzen
LIN	Segmentierung von Linearsätzen
LENGTH=..	Länge der entstehenden Linearsätze in [mm, inch]
CIR	Segmentierung von Zirkularsätzen
OPMODE=..	Modus der Kreissegmentierung: 0: Vorgabe der gewünschten Satzlänge. 1: Angabe des gewünschten Sehnenfehlers. Satzlänge wird automatisch berechnet (Default). 2: Vorgabe der gewünschten Satzlänge, Ausgabe als Kreissegmente. Sehnenfehler bzw. Satzlänge werden im Schlüsselwort PARAM angegeben.
PARAM=..	Entweder Sehnenfehler oder Länge der entstehenden Linearsätze, je nach gewähltem OPMODE in [mm, inch]

Wird bei einer Aktivierung der Segmentierung keine Parametrierung ausser LIN und/oder CIR vorgenommen, ist folgender Grundzustand gültig:

LENGTH	1 mm
OPMODE	0
PARAM	0.1 mm

Programmierbeispiel

Geschwindigkeitsoptimierte Bewegungsführung durch Segmentierung

Anwahl Linearsegmentierung mit Defaultparametrierung:

```
N20 G00 X0 Y0 Z0 F10000
N30 #SEGMENTATION ON [LIN]
N40 X3     Y25
N50 X15    Y15
N60 X23     Y12
N70 X25     Y25
N80 X30     Y35
N90 #SEGMENTATION OFF [LIN] ;Abwahl
N100 M30
```

Anwahl Linearsegmentierung + Parametrierung:

```
N20 G00 X0 Y0 Z0 F10000
N30 #SEGMENTATION ON [LIN LENGTH 0.5]
N40 X3     Y25
N50 X15    Y15
N60 X23    Y12
N70 X25    Y25
N80 X30    Y35
N90 #SEGMENTATION OFF [LIN] ;Abwahl
N100 M30
```

Anwahl Linear- und Zirkularsegmentierung + Parametrierung:

```
N30 #SEGMENTATION ON [LIN LENGTH 0.5 CIR OPMODE 1 PARAM 0.5]
N40 X3     Y25
N50 X15    Y15
N60 X23    Y12
N70 X25    Y25
N80 X30    Y35
N90 #SEGMENTATION OFF ALL ;Abwahl
N100 M30 ;Alternativ: #SEGMENTATION OFF [LIN CIR]
```

Kombinierte Anwahl Linear- und Zirkularsegmentierung:

```
N30 #SEGMENTATION ON ALL
N40 X3     Y25
N50 X15    Y15
N60 X23    Y12
N70 X25    Y25
N80 X30    Y35
N90 #SEGMENTATION OFF ALL ;Abwahl
N100 M30
```

12.32 Vergrößern/Verkleinern von Konturen (#SCALE ON/OFF)

Mit dem Befehl #SCALE können Konturen bzw. Positionen durch die Angabe achsspezifischer Faktoren maßstäblich vergrößert bzw. verkleinert werden. Die Angabe unterschiedlicher Faktoren erlaubt hierbei auch Dehnungen bzw. Stauchungen von Konturen.

Die Skalierung wirkt auf

- Bahnachsen (Linearachsen, z.B. X, Y, Z, U, V, W)
- Positionierachsen (unabhängige Achsen, Pendelachsen).

Verschiebungen werden generell nicht skaliert, unabhängig davon, ob die Anwahl oder Programmierung der Verschiebung vor oder nach #SCALE ON erfolgt.

Die skalierten Werte werden immer auf den Nullpunkt des aktuell aktiven Koordinatensystems bezogen. Insbesondere bei der Beschreibung der Kontur in Absolutkoordinaten wird empfohlen, vor Anwahl der Skalierung den Nullpunkt durch die entsprechenden G-Funktionen G53...G57, G159, G92 oder durch die Funktionen zur Definition von gedrehten Koordinatensystemen #ROTATION und #CS auf den Startpunkt der Kontur zu legen.

Syntax:

#SCALE [ON] <*Achsname*>.. { <*Achsname*>.. }
#SCALE OFF

ON	Skalierung aktivieren
OFF	Skalierung deaktivieren
<*Achsname*>..	Achsspezifische Skalierungsfaktoren. Der Faktor muss > 0 sein.
	Bei Faktoren ≤ 0 erfolgt die Ausgabe einer Fehlermeldung.

Hinweis

Die Festlegung der Skalierungsfaktoren und die Anwahl können gemeinsam oder in separaten Schritten angegeben werden. Das heißt, es ist z.B. möglich, zunächst die Skalierungsfaktoren zu definieren und in einem zweiten Befehl dann die Skalierung zu aktivieren.

Die programmierten Skalierungsfaktoren bleiben bis zum Programmende mit M30 gespeichert und können bei mehrfachem #SCALE ON / OFF erneut verwendet werden.

Achtung

Die Skalierungsfaktoren können bei abgewählter Skalierung geändert werden.

Eine Änderung der Skalierungsfaktoren bei aktiver Skalierung führt zur Ausgabe einer Fehlermeldung.

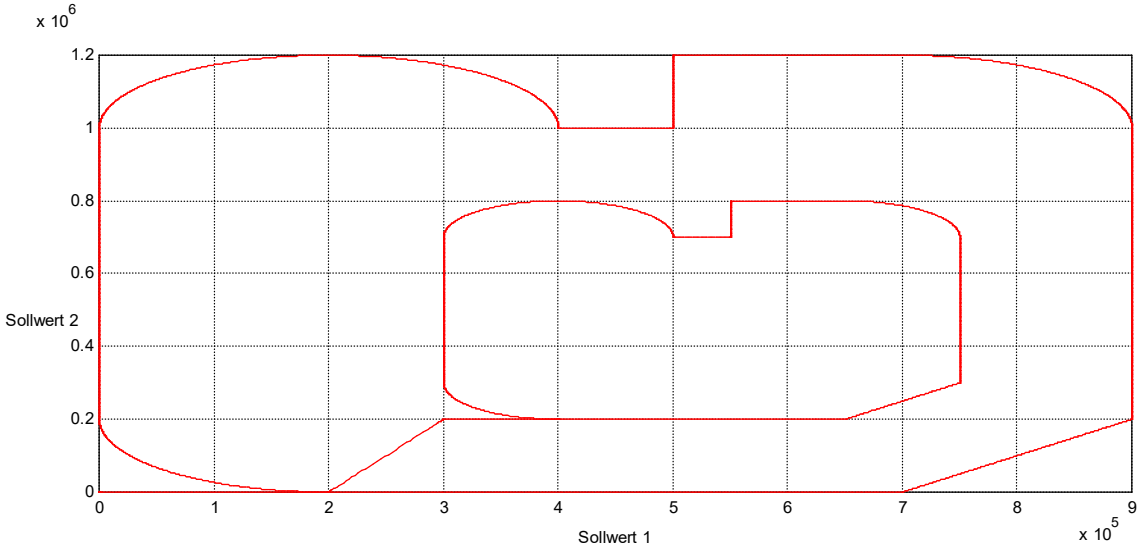
Bei der Kombination von Skalierung und Kreisprogrammierung ist folgendes zu beachten:

- Die Skalierung von Kreisen ist nur sinnvoll, wenn die Skalierungsfaktoren der an der Kreisinterpolation beteiligten Achsen gleich sind. Dies gilt insbesondere für die Programmierung von Kreisen mir Radiusangabe über R oder G163, da hier der Faktor für den Radius auf dem mit "X" programmierten Skalierungsfaktor basiert.
- Bei der Kreisprogrammierung mit I, J, K sind auch unterschiedliche Skalierungsfaktoren möglich; - diese führen im Normalfall jedoch zu Fehlern in der Kreismittelpunktskorrektur bzw. zu Kreissegmenten ohne praktische Bedeutung.

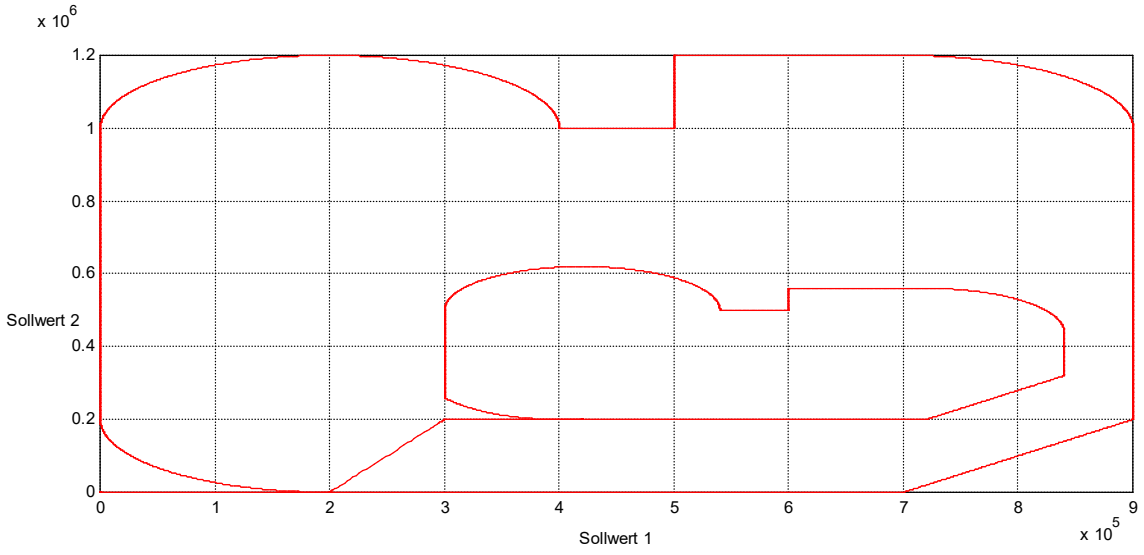
Programmierbeispiel

Vergrößern und Verkleinern von Konturen

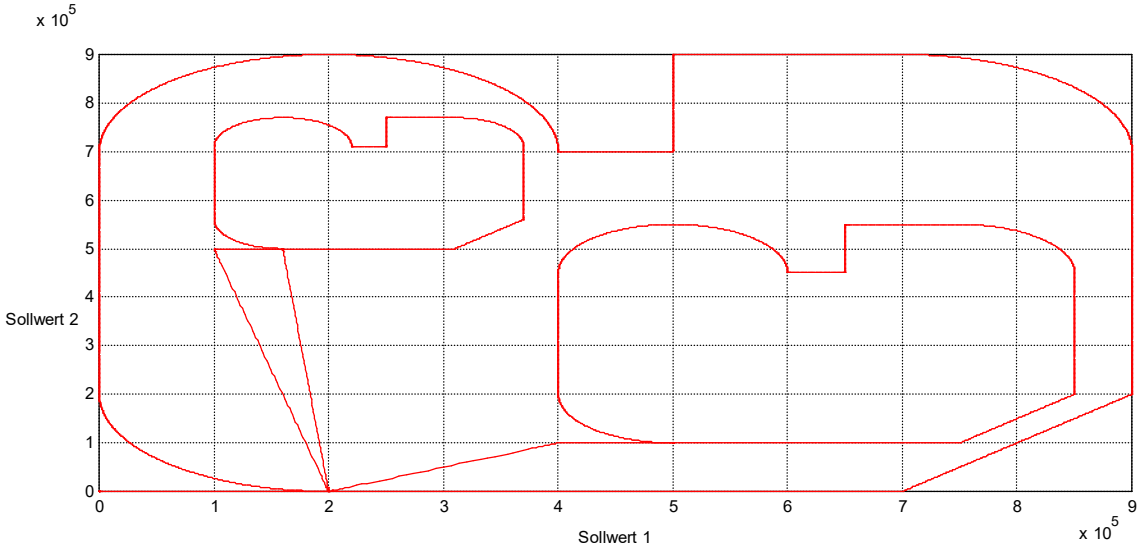
```
;Skalieren einer absolut programmierten Kontur mit gleichen Faktoren

%L Cont1_abs
N01 G01 G90 F2000
N02 X90 Y0
N03 G301 I20
N04 X90 Y120
N05 G302 I20
N06 X50 Y120
N07 X50 Y100
N08 X40 Y100
N09 G03 X0 Y100 I-20 J0
N10 G01 X0 Y20
N11 G03 X20 Y0 R20
N13 M17

%scale
N015 G53
N020 G00 X0 Y0 Z0
N065 LL Cont1_abs ; Basiskontur
N075 #SCALE X0.5 Y0.5 ;Definition der Skalierungsfaktoren
;Definition des Nullpunktes der skalierten Kontur
N085 V.G.NP[1].V.X = 30
N090 V.G.NP[1].V.Y = 20
N095 G54
N100 G90 G0 X0 Y0 Z0
N105 #SCALE ON
N110 LL Cont1_abs
N115 #SCALE OFF
N140 M30
```



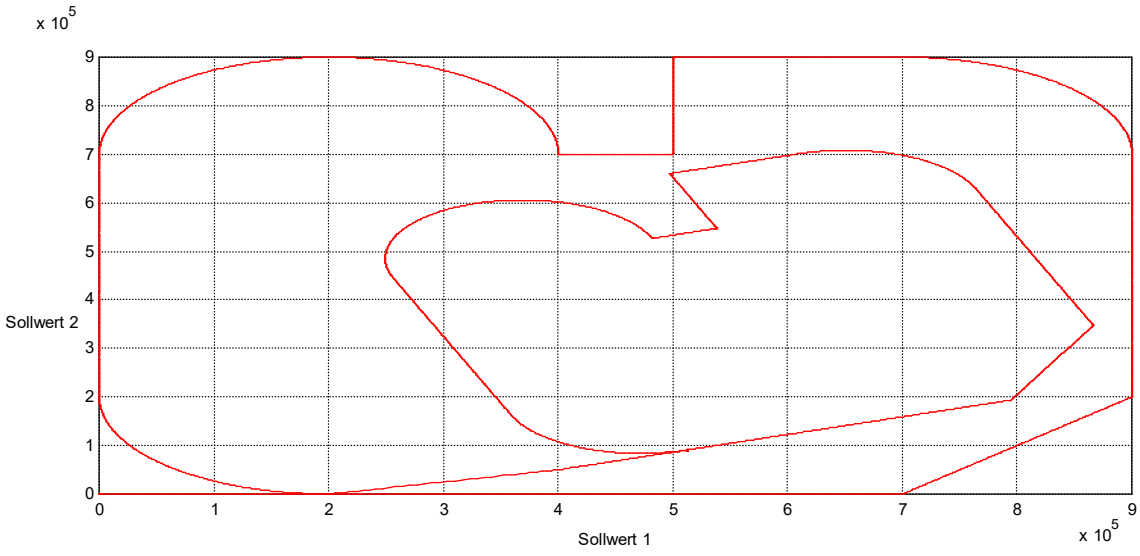
```
;Skalieren einer absolut programmierten Kontur mit unterschiedli-
chen ;Faktoren

%L Cont1_abs
N01 G01 G90 F2000
N02 X90 Y0
N03 G301 I20
N04 X90 Y120
N05 G302 I20
N06 X50 Y120
N07 X50 Y100
N08 X40 Y100
N09 G03 X0 Y100 I-20 J0
N10 G01 X0 Y20
N11 G03 X20 Y0 R20
N13 M17

%scale

N015 G53
N020 G00 X0 Y0 Z0
N065 LL Cont1_abs ; Basiskontur
N075 #SCALE X0.6 Y0.3 ;Definition der Skalierungsfaktoren
;Definition des Nullpunktes der skalierten Kontur
N085 V.G.NP[1].V.X = 30
N090 V.G.NP[1].V.Y = 20
N095 G54
N100 G90 G0 X0 Y0 Z0
N105 #SCALE ON
N110 LL Cont1_abs
N115 #SCALE OFF
N140 M30
```



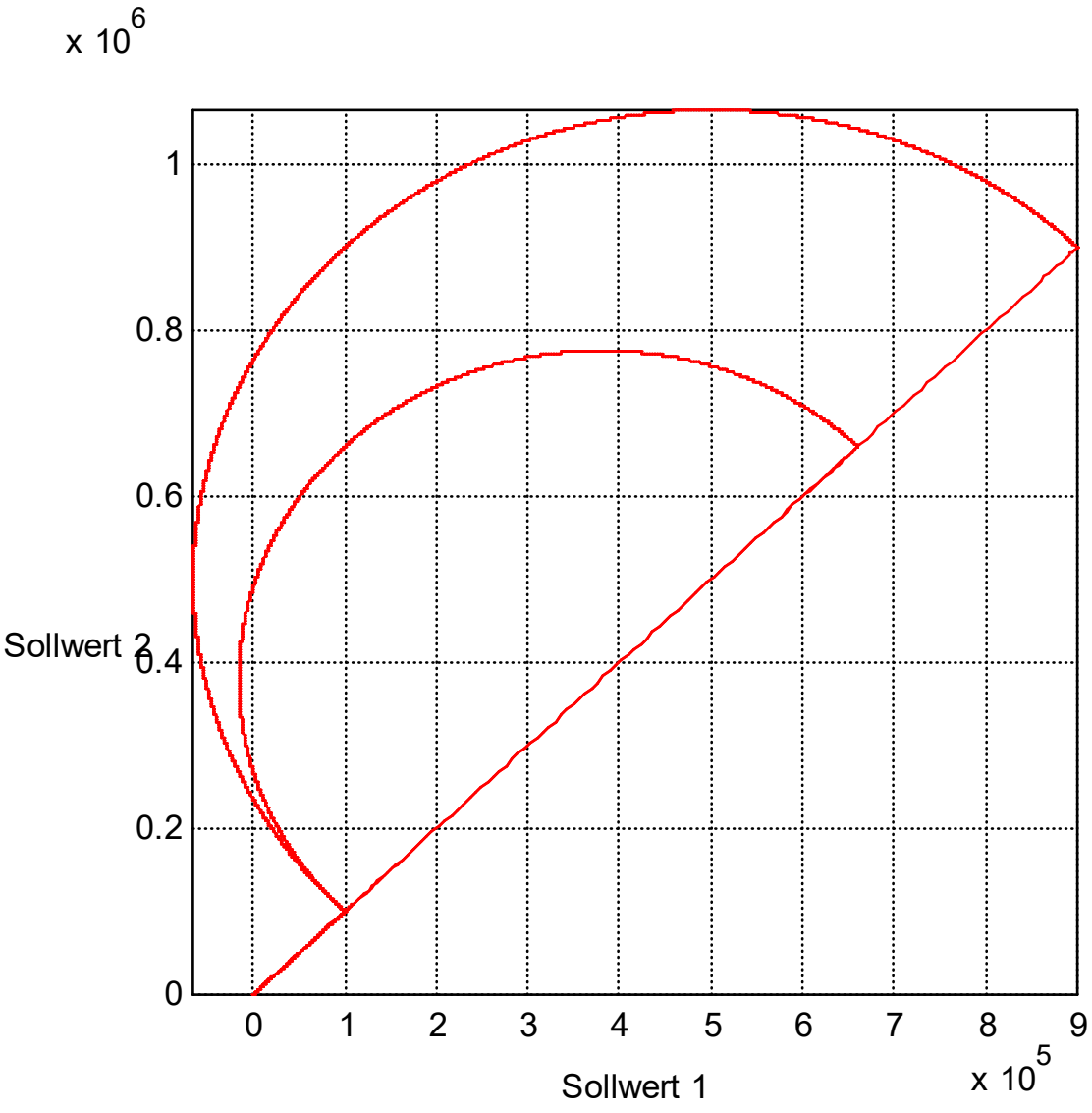
```
;Mehrfaches unterschiedliches Skalieren einer relativ programmierten
;Kontur

%L Cont1_rel
N01 G01 G91 F2000
N02 X90 Y0
N03 G301 I20
N04 X0 Y90
N05 G302 I20
N06 X-40 Y0
N07 X0 Y-20
N08 X-10 Y0
N09 G03 X-40 Y0 I-20 J0
N10 G01 X0 Y-50
N11 G03 X20 Y-20 R20
N13 M17

%scale

N015 G53
N020 G00 X0 Y0 Z0
; Basiskontur
N030 LL Cont1_rel
N040 #SCALE X0.3 Y0.3 ;Definition 1 der Skalierungsfaktoren
;Definition 1 des Startpunktes der skalierten Kontur
N055 G90 G0 X10 Y50
N060 #SCALE ON
N065 LL Cont1_rel
N070 #SCALE OFF
N075 G90 G00 X20 Y0
N085 #SCALE X0.5 Y0.5 ;Definition 2 der Skalierungsfaktoren
;Definition 1 des Startpunktes der skalierten Kontur
N100 G90 G0 X40 Y10
N105 #SCALE ON
N110 LL Cont1_rel
N115 #SCALE OFF
N125 M30
```



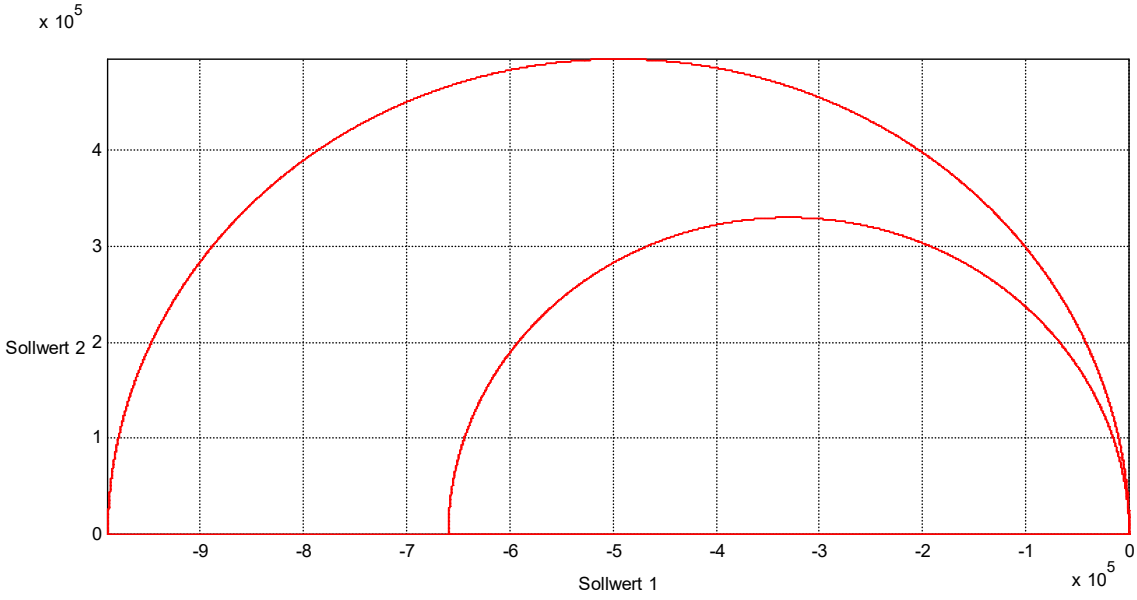
```
;Skalieren einer Kontur in einem mit #CS verschobenem und gedrehten
;Koordinatensystem

%L Cont1_rel
N01 G01 G91 F2000
N02 X90 Y0
N03 G301 I20
N04 X0 Y90
N05 G302 I20
N06 X-40 Y0
N07 X0 Y-20
N08 X-10 Y0
N09 G03 X-40 Y0 I-20 J0
N10 G01 X0 Y-50
N11 G03 X20 Y-20 R20
N13 M17

%scale

N015 G53
N020 G00 X0 Y0 Z0
; Basiskontur
N030 LL Cont1_rel
N040 #SCALE X0.6 Y0.6 ;Definition der Skalierungsfaktoren
N045 #CS ON [40,5,0,0,0,20]
N055 G90 G0 X0 Y0
N060 #SCALE ON
N065 LL Cont1_rel
N070 #SCALE OFF
N0525 #CS OFF
N125 M30
```



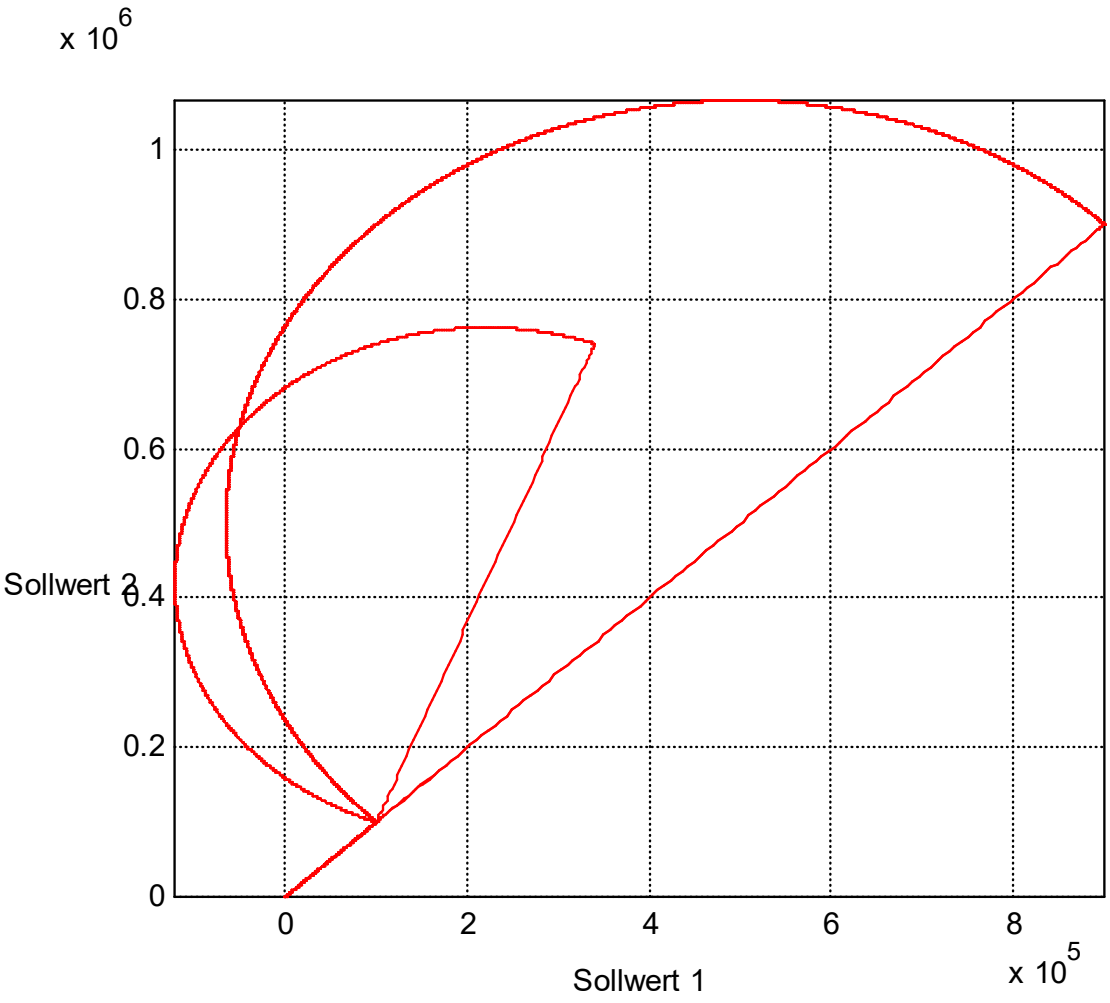
```
;Skalieren eines Halbkreises, programmiert mit I, J

%L Circle1
G02 G91 X80 Y80 I40 J40
M17

%scale

N015 G53
N020 G00 X0 Y0 Z0
N025 G01 X10 Y10 F1000
; Basishalbkreis
N030 LL Circle1
N040 #SCALE X0.7 Y0.7 ;Definition der Skalierungsfaktoren
N055 G90 G0 X10 Y10
N060 #SCALE ON
N065 LL Circle1
N070 #SCALE OFF
N075 G90 G0 X0 Y0

M30
```



```
;Skalieren eines Halbkreises, programmiert mit R

%L Circle2
G03 R33 X-66 Y0
M17
%scale

N015 G53
N020 G00 X0 Y0 Z0 F1000
; Basishalbkreis
N030 LL Circle2
N040 #SCALE X1.5 Y1.5 ;Definition der Skalierungsfaktoren
N055 G90 G0 X10 Y10
N060 #SCALE ON
N065 LL Circle2
N070 #SCALE OFF
N075 G90 G0 X0 Y0

M30
```



```
;Unsymmetrisches Skalieren eines Halbkreises, programmiert mit I, J

%L Circle1
G02 G91 X80 Y80 I40 J40
M17

%scale

N015 G53
N020 G00 X0 Y0 Z0
N025 G01 X10 Y10 F1000
; Basishalbkreis
N030 LL Circle1
N040 #SCALE X0.3 Y0.8 ;Definition der Skalierungsfaktoren
N055 G90 G0 X10 Y10
N060 #SCALE ON
N065 LL Circle1
N070 #SCALE OFF
N075 G90 G0 X10 Y10

M30
```



12.33 Stanzen und Nibbeln

Beim Stanzen und Nibbeln werden automatische Hubbewegungen in einer programmierbaren Anzahl von Teilschritten ausgeführt. Die Sequenz der Hubbewegung ist hierbei vom Anwender frei definierbar. Die Hubbewegung wird nur bei Verfahrbewegungen in der aktiven Bearbeitungsebene ausgelöst.

12.33.1 Aufteilung des Fahrwegs und Programmierung (#STROKE DEF, #PUNCH ON/OFF, #NIBBLE ON/OFF)

Die Aufteilung des Fahrwegs erfolgt entweder durch Angabe der **Länge** eines Teilsegmentes oder der **Anzahl** von Teilsegmenten. Die Programmierung ist modal wirksam.

Die Aufteilung durch Angabe der Länge eines Teilsegmentes erfolgt so, dass der Verfahrsatz gleichmässig in Teilsegmente unterteilt wird. Dabei ist die Länge jedes realen Teilsegmentes kleiner/gleich der Länge des programmierten Teilsegmentes.

Bei der Aufteilung durch Angabe der Anzahl von Teilsegmenten wird die Segmentlänge automatisch berechnet.

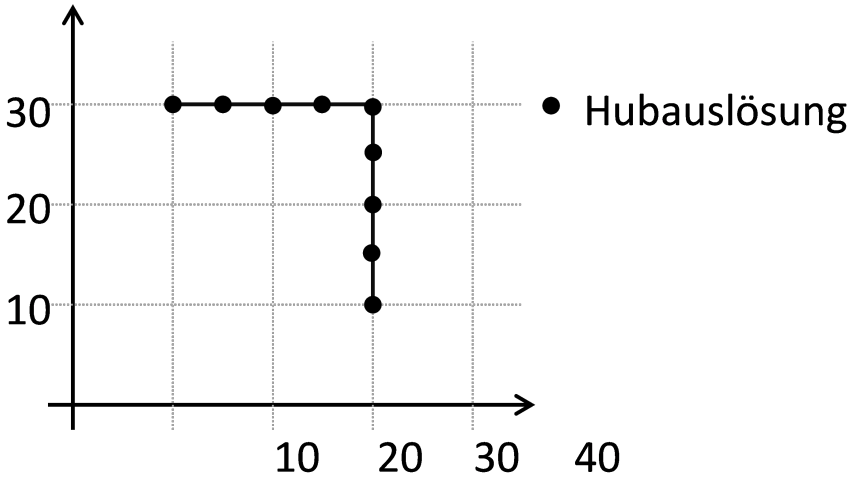


Abb. 121: Aufteilung von Linearsätzen

Programmierte Zirkularsätze werden in lineare Teilbewegungungen umgesetzt, wobei sich die Aufteilung des Fahrwegs auf die Bogenlänge des Kreises bezieht.

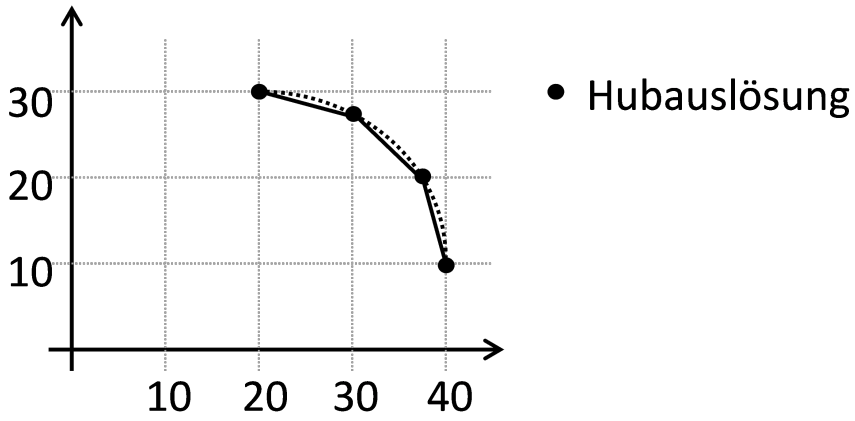


Abb. 122: Aufteilung von Zirkularsätzen

Zur Definition der Hubbewegung wird eine auf maximal *10 Sätze* begrenzte Satzsequenz innerhalb eines Blocks programmiert, der mit folgenden beiden Befehlen begrenzt wird:

Syntax Definition der Hubbewegung:

#STROKE DEF BEGIN	Beginn der Hubdefinition
#STROKE DEF END	Ende der Hubdefinition

Für die Hubdefinition ist nur folgender eingeschränkter Funktionsumfang erlaubt:

#STROKE DEF BEGIN

G0, G01, M, H, G261, G61, G260, G60, G04, #TIME, G90, G91

#STROKE DEF END

Hinweis

Die Verwendung anderer NC-Befehle innerhalb der Hubdefinition führt zu einem Fehler.

Syntax Aktivierung/Deaktivierung Stanzen und Nibbeln:

#PUNCH ON | **OFF [** [**LENGTH**=.. | **NUMBER**=..] **]**

#NIBBLE ON | **OFF [** [**LENGTH**=.. | **NUMBER**=..]

ON	Stanzen/Nibbeln aktivieren
OFF	Stanzen/Nibbeln deaktivieren
LENGTH=..	Länge eines Teilsegmentes, nach dem automatisch eine Hubbewegung eingefügt wird in [mm, inch]. Bei #NIBBLE ON wird bei der ersten Verfahrbewegung in der aktiven Bearbeitungsebene auch am Start eine Hubbewegung ausgeführt.
NUMBER=..	Anzahl der zu erzeugenden Teilsegmente innerhalb eines Verfahrbefehls. Nach jedem Teilsegment wird eine Hubbewegung ausgelöst.

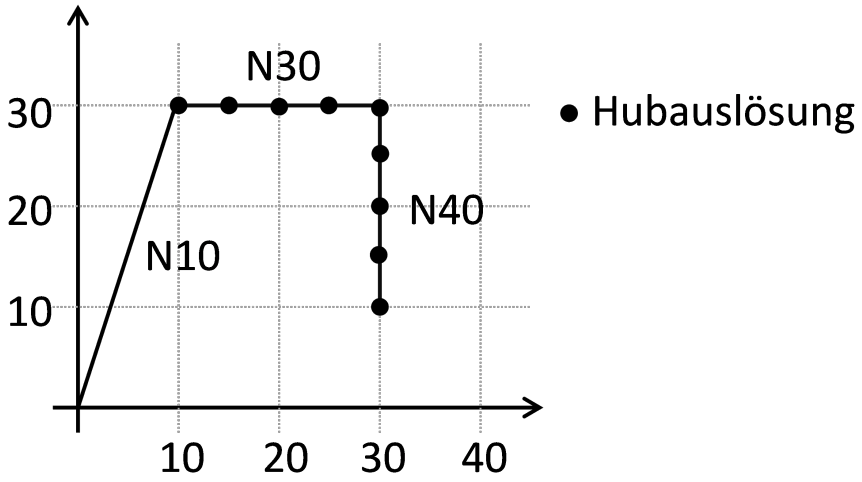
Hinweis

LENGTH und NUMBER sind exklusiv, d.h. es wird entweder nach Teilsegmentlänge oder nach Teilsegmentanzahl aufgeteilt.

Programmierbeispiel

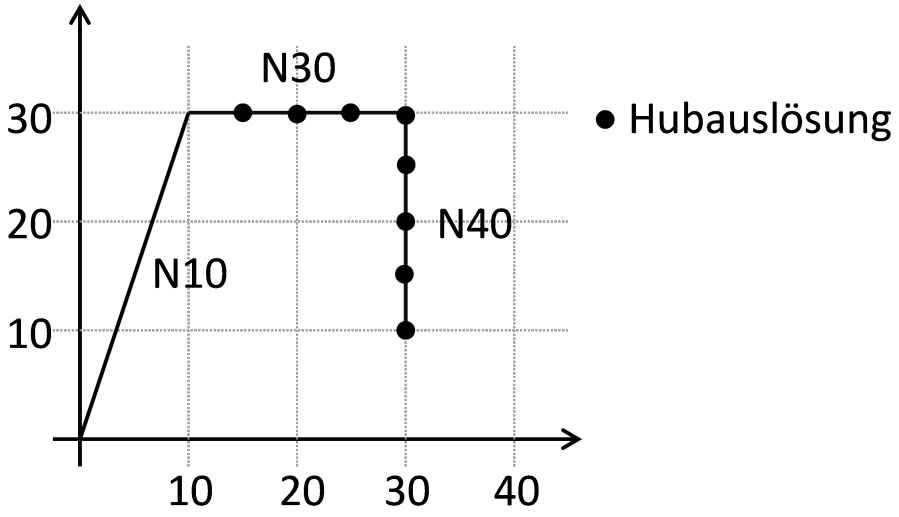
Aufteilung des Fahrwegs und Programmierung

```
%Nibble
N10 G90 G17
N20 #STROKE DEF BEGIN
N30 G04 0.01
N40 G91 Z10
N50 Z-10
N60 #STROKE DEF END

N70   X10    Y30 C0
N80 #NIBBLE ON [LENGTH 5]
N90   X30    Y30 C180
N100  X30    Y10 C360
N110 #NIBBLE OFF
N120 M30
```



Programmierbeispiel

```
%Punch
N10 G90 G17
N20 #STROKE DEF BEGIN
N30 G04 0.01
N40 G91 Z10
N50     Z-10
N60 #STROKE DEF END

N70   X10    Y30
N80 #PUNCH ON [LENGTH 5]
N90   X30    Y30
N100  X30    Y10
N110 #PUNCH OFF
N120 M30
```



Konfiguration:

Für die Benutzung der Funktionalität muss in der Hochlaufliste folgende Einstellung für P-STUP-00060 vorgenommen werden:

configuration.channel[0].path_preparation.function FCT_DEFAULT | **FCT_PUNCHING**

configuration.channel[0].path_preparation.function FCT_DEFAULT | **FCT_NIBBLING**

12.33.2 Weiterführende Funktionen

Beeinflussung der Hubauslösezeit:

Für ein gutes Bearbeitungsergebnis kann der Auslösezeitpunkt des Hubes beeinflusst werden. So ermöglicht eine Hub**vor**auslösung den Ausgleich der konstanten Totzeit der Signalverarbeitung. Durch eine Hub**nach**auslösung kann das Einschwingverhalten der Achsen oder zum Beispiel die Zeit zum Niederhalten beim Stanzen mit Niederhalten ausgeglichen werden.

Die Beeinflussung der Hubauslösezeit bezieht sich je nach Einstellung auf eines der beiden folgenden Ereignisse:

- Interpolationsendpunkt erreicht →G00,G01,G02,G03
- In-Pos Fenster der langsamsten Achsen → G60

Für die Hubnachauslösung wird die Verweilzeit G04 zusammen mit einer M-Funktion eingesetzt.

Für die Vorauslösung wird eine M-Funktion vom Typ MET_SVS verwendet.

Hinweis

Die zeitliche Auflösung der Verschiebung richtet sich nach der Zykluszeit der CNC-Steuerung als kleinste Einheit.

Beispiel

Hubvorauslösung:

Die anwenderspezifische M-Funktion M97 soll 40 Millisekunden vor Erreichen des Synchronisationszeitpunktes in der Satzsequenz an die PLC ausgegeben werden.

Auszug aus Kanalparameterliste:

```
# Festlegung der M-Funktionen und Synchronisationsarten
# ====================================================
m_synch[97] 0x02000000 MET_SVS

# Einstellung von Vorausgabezeit, Vorausgabeweg mit Typ MET_SVS
# ============================================================
m_pre_outp[97] 40000 in us
```

```
N10 #STROKE DEF BEGIN
N30 M97
N40 #STROKE DEF END
```

Beispiel

Hubnachauslösung:

M96 ist die anwenderspezifische M-Funktion zum Auslösen der Hubbewegung. Die Hubbewegung soll erst 50 Millisekunden nach erreichen der Hubpositionen erfolgen.

```
N10 #STROKE DEF BEGIN
N20 G04 0.05
N30 M96
N40 #STROKE DEF END
```

Beispiel

Ausgabe der M-Funktion bezogen auf In-Pos Fenster:

M96 ist die anwenderspezifische M-Funktion zum Auslösen der Hubbewegung. Die Hubbewegung soll erst 50 Millisekunden nach erreichen der physikalischen Hubpositionen aller Achsen (Lage-Istwert ist im Fenster) erfolgen.

```
N10 #STROKE DEF BEGIN
N20 G04 0.05
N30 M96
N40 #STROKE DEF END
```

```
G90 G17
N10   X10     Y30
N20 #PUNCH ON [LENGTH 5]
N30   G60 X30    Y30
N40   G60 X30    Y10
N90 #PUNCH OFF
```

12.33.3 Einschränkungen

Die Verwendung von Rotationen, welche die aktive Bearbeitungsebene kippen, ist nicht erlaubt. Rotationen bei aktiver XY-Ebene um die Z-Achse sind hingegen erlaubt. Diese Einschränkung gilt nur, falls in der Hubsequenz Achsbewegungen programmiert wurden. Eine Ausgabe z.B. von M-Funktionen ist weiterhin möglich.

12.34 Steuerung der Eckenbearbeitung (#EDGE MACHINING)

Abhängig von der Bearbeitungstechnologie kann es erforderlich sein, dass der Bearbeitungsprozess an spitzen Konturverläufen (Ecken) speziell gesteuert werden muss. Im Fall einer spitzen Ecke (definiert durch Winkeldifferenz zwischen zwei Konturelementen) wird der Bahngeschwindigkeitsverlauf an der Ecke abhängig von vordefinierten Parametern modifiziert. Als Konturelemente können Linear- oder Zirkularsätze programmiert werden, dabei wird nicht geprüft, ob es sich um eine Außen- oder Innenkontur handelt!

Die Funktion Eckenbearbeitung wird in der Kanalparameterliste konfiguriert und wirkt ab Programmstart. Die genaue Beschreibung der Parametrierung kann der Dokumentation [MDS-CHAN//Kapitel Einstellungen für die Eckenbearbeitung] entnommen werden.

Zusätzlich kann die Eckenbearbeitung auch im NC-Programm durch folgenden NC-Befehl an-/abgewählt und umparametriert werden.

Syntax Anwahl der Eckenbearbeitung:

#EDGE MACHINING ON [[**ANGLE_LIMIT**=..] [**WAIT_TIME**=..] [**PRE_DIST**=..] [**PRE_FEED**=..]
[**POST_DIST**=..] [**POST_FEED**=..] [**DISABLE_FEED_ADAPTION**=..] **]**

oder mit Setzen der Standardwerte

#EDGE MACHINING ON [[**ANGLE_LIMIT**] [**WAIT_TIME**] [**PRE_DIST**] [**PRE_FEED**]
[**POST_DIST**] [**POST_FEED**] [**DISABLE_FEED_ADAPTION**] **]**

oder

#EDGE MACHINING ON DEFAULT

ON	Eckenbearbeitung aktivieren
ANGLE_LIMIT=..	Grenzknickwinkel in [°]. Unterschreitet der Knickwinkel zwischen zwei Konturelementen diesen Grenzwinkel, so wird die spezielle Eckenbehandlung durchgeführt.
WAIT_TIME=..	Wartezeit in der Ecke in [s]
PRE_DIST=..	Abstand vor der Ecke in [mm; inch], bei dem ein PLC-Signal erzeugt wird.
PRE_FEED=..	Vorschub vor Ecke in [mm/min, m/min, inch/min], mit dem ab PRE_DIST bis zum Stop in der Ecke gefahren wird.
POST_DIST=..	Abstand nach der Ecke in [mm, inch], bei dem ein PLC-Signal erzeugt wird.
POST_FEED=..	Vorschub nach Ecke in [mm/min, m/min, inch/min], mit dem bis POST_DIST gefahren wird. Danach wird mit dem ursprünglich programmierten Vorschub weitergefahren.
DISABLE_FEED_ADAPTION=..	Schalten der Vorschubanpassung, Bool'scher Wert: 0: Vorschubanpassung ist wirksam 1: Vorschubanpassung aus, PRE_FEED und POST_FEED wirken nicht.
DEFAULT	In Verbindung mit der Anwahl ON gelten die Standardwerte aus der Kanalparameterliste [P-CHAN-00221 - P-CHAN-00226, P-CHAN-00300].

Werden alle oder einzelne Schlüsselworte ANGLE_LIMIT, WAIT_TIME, PRE_DIST, PRE_FEED, POST_DIST, POST_FEED oder DISABLE_FEED_ADAPTION ohne Wert programmiert, gelten die entsprechenden Standardwerte aus der Kanalparameterliste [P-CHAN-00221 - P-CHAN-00226, P-CHAN-00300].

Der Befehl #EDGE MACHINING OFF bewirkt die Abwahl der Eckenbearbeitung. Zusammen mit der Abwahl dürfen keine weiteren Parameter angegeben werden.

Syntax Abwahl der Eckenbearbeitung:

#EDGE MACHINING OFF

Konfiguration:

Für die Benutzung der Funktionalität muss in der Hochlaufliste ([STUP]) folgende Einstellung vorgenommen werden:

configuration.channel[0].path_preparation.function FCT_DEFAULT | **FCT_EMF**

Achtung

Das Umparametrieren während aktiver Eckenbearbeitung ist zulässig. Es wird jedoch empfohlen, die Änderung erst nach Abschluss relevanter Konturelemente zu programmieren, um ein undefiniertes Verhalten der Eckenbearbeitung zu vermeiden.

Programmierbeispiel

Steuerung der Eckenbearbeitung

```
%edge_machining

(Anwahl Eckenbearbeitung und Parameter mit Wert setzen)
#EDGE MACHINING ON [ANGLE_LIMIT=90 WAIT_TIME=0.2 PRE_DIST=5 PRE_FEED=800
POST_DIST=10 POST_FEED=1600]

(Anwahl Eckenbearbeitung und alle Parameter auf Default setzen)
#EDGE MACHINING ON [ANGLE_LIMIT PRE_DIST PRE_FEED POST_DIST POST_FEED
WAIT_TIME DISABLE_FEED_ADAPTION]

oder

(Anwahl Eckenbearbeitung und alle Parameter auf Default setzen)
#EDGE MACHINING ON DEFAULT

(Abwahl Eckenbearbeitung)
#EDGE MACHINING OFF
```

12.35 Schalten der Dynamikgewichtung (#DYNAMIC WEIGHT)

Die weg- bzw. radiusabhängige Gewichtung der Dynamikgrenzwerte werden in der Kanalliste konfiguriert und über die Parameter P-CHAN-00190 und P-CHAN-00230 aktiviert. In Konfigurationsbeispiele für wegabhängige Gewichtung und radiusabhängige Gewichtung sind in [CHAN] zu finden.

Nach Hochlauf der Steuerung sind diese Gewichtungen dann wirksam.

Mit nachfolgendem NC-Befehl sind die Dynamikgewichtungen zur Laufzeit im NC-Programm schaltbar, d.h. sie können beliebig aktiviert und deaktiviert werden.

Syntax:

#DYNAMIC WEIGHT [**ON** | **OFF**] **[** [**PATH** | **CURVE**] **]**

ON	Gewichtung aktivieren
OFF	Gewichtung deaktivieren
PATH	Weglängenabhängige Gewichtung (siehe P-CHAN-00190)
CURVE	Kreisradiusabhängige Gewichtung (siehe P-CHAN-00230)

Hinweis

Bei An-/ und Abwahl muss mindestens ein Kennwort PATH oder CURVE programmiert sein.

Programmierbeispiel

Schalten der Dynamikgewichtung

```
N20 G00 X0 Y0 Z0 F10000
N30 #DYNAMIC WEIGHT ON [PATH]   ;Anwahl wegabhängige Dynamikgewichtung
N40 X3 Y25
N50 X15 Y15
N60 X23 Y12
N70 X25 Y25
N80 X30 Y35
N90 #DYNAMIC WEIGHT OFF [PATH]  ;Abwahl
N100 M30
```

12.36 Gewichtung des externen Vorschubes (#FF)

Im NC-Kanal besteht die Möglichkeit, über die SPS einen externen Bahnvorschub [HLI// Steuerkommandos eines Kanals] zu kommandieren. Dieser kann im NC-Programm durch den Vorschubfaktor #FF zusätzlich gewichtet werden.

Die Wirkung von #FF wird durch Setzen des Parameters P-CHAN-00422 gesteuert.

Syntax:

#FF = *<feed_factor>*

<feed_factor> Gewichtung des externen Vorschubs in [%]

Hinweis

Der Vorschubfaktor #FF wirkt nur auf einen extern vorgegebenen Vorschub.
Vor der Version V3.01.3079.0 ist der Vorschubfaktor #FF auf maximal 10000% und minimal auf >0 begrenzt und ab dieser Version auf maximal 1000000% und minimal auf 0.1%.

Der NC-Befehl ist verfügbar ab CNC-Version V3.01.3064.02.

12.37 Klemmen und Überwachen von Achsen (#CLAMP MONITORING)

Zur Gewährleistung einer hohen Qualität und Genauigkeit eines Bearbeitungsprozesses (Fräsen, Drehen, Erodieren etc.) kann die werkstücktragende Achse physikalisch durch die Antriebsbremse fixiert werden. Dieses mechanische Klemmen der Achse wird durch synchronisierte M/H-Funktionen im NC-Programm beauftragt und in der SPS geschaltet. Dadurch ist sichergestellt, dass eine Achse geklemmt ist, bevor der nächste Bewegungssatz ausgeführt wird.

Der Status 'geklemmte Achse' ist auf Achstreiberebene wirksam und bleibt auch bei Achstauschoperationen kanalübergreifend bis zur Abwahl gültig.

Während eine Achse geklemmt ist, darf ihr Antrieb weder vom Interpolator noch von einer Zusatzschnittstelle mit Sollwerten kommandiert werden. Die dazu notwendige Überwachung wird mit dem nachfolgenden NC-Befehl aktiviert bzw. deaktiviert. Wenn eine geklemmte Achse bewegt werden soll, erfolgt die Ausgabe der Fehlermeldung P-ERR-70525 und die CNC wechselt in den Fehlerzustand.

Der Überwachungsstatus ist modal und bleibt auch nach NC-Programmende oder RESET aktiv.

Syntax Überwachen bestimmter geklemmter Achsen:

#CLAMP MONITORING ON | **OFF [** { **AX**=<*Achsname*> | **AXNR=..** } **]**

AX=<*Achsname*>	Name der geklemmten Achse, die überwacht werden soll.
AXNR=..	Logische Nummer der Achse, die überwacht werden soll, positive Ganzzahl

Syntax Überwachen aller geklemmten Achsen:

#CLAMP MONITORING ON | **OFF ALL**

Programmierbeispiel

Klemmen und Überwachen von Achsen

```
%clamp.nc
N010 X0 Y0 Z0 A0 B0 C0
N020 A[M300] C[M300] ;Mechanisches Klemmen Achsen A + C an
N030 #CLAMP MONITORING ON ALL ;Ueberwachung Achsen A + C aktivieren
..
N100 #CLAMP MONITORING OFF ALL ;Ueberwachung Achsen A + C deaktivieren
..
N110 #CLAMP MONITORING ON [AX=A AX=C] ;Ueberwachung Achsen A + C akti-
vieren
N120 Y10
N130 X15
N140 #CLAMP MONITORING OFF [AX=A AX=C] ;Ueberwachung Achsen A + C deak-
tivieren
N150 X10
N160 #CLAMP MONITORING ON [AXNR=4 AXNR=6] ;Ueberwachung Achsen A + C ak-
tivieren
N120 Y20
N130 X25
N150 A[M301] C[M301] ;Mechanisches Klemmen Achsen A + C lösen
N140 #CLAMP MONITORING OFF [AXNR=4 AXNR=6] ;Ueberwachung Achsen A + C
deaktivieren
..
N999 M30
```

Hinweis

Der Befehl bewirkt keine mechanische Klemmung einer Achse, sondern schaltet die Überwachung bzgl. unzulässiger Achsbewegungen. Bei der Funktion 'Rückwärtsfahren auf der Bahn' wird der Status der Überwachung entsprechend invertiert.

Hinweis

Achsbewegungen, die durch aktive Kompensationsfunktionen im Achstreiber erzeugt werden (z.B. Kreuzkompensation, Volumetrische Kompensation etc.), werden nicht überwacht.

Der Überwachungsstatus einer Achse kann über den ADS-Zugriff..

Index group: 0x20300

Index offset: 0x10189 (Achse 1)

.. gelesen werden.

12.38 Gantryinbetriebnahme (#GANTRY ON/OFF)

Achtung

Der #GANTRY Befehl darf nur zur Inbetriebnahme verwendet werden

Mögliche Maschinenschäden bei falscher Anwendung des Befehls.

Weitere Informationen siehe [FCT-C11// Gantryinbetriebnahme]

Syntax Lösen eines Gantryverbunds:

#GANTRY OFF [{ **AXNR**=.. | **AX**=<*Achsname*> } **]**

AXNR=..	Logische Achsnummer (P-AXIS-00016) der Masterachse
AX=<*Achsname*>	Name der Masterachse des Gantryverbunds

Syntax Lösen aller Gantryverbunde:

#GANTRY OFF ALL

Syntax Wiederherstellen eines Gantryverbunds:

#GANTRY ON [{ **AXNR=..** | **AX=**<*Achsname*> } **]**

AXNR=..	Logische Achsnummer (P-AXIS-00016) der Masterachse
AX=<*Achsname*>	Name der Masterachse des Gantryverbunds

Syntax Wiederherstellen aller Gantryverbunde:

#GANTRY ON ALL

12.39 Lagereglerbasierte Achskopplungen (#GEAR LINK)

Durch die lagereglerbasierte Achskopplung wird die Bewegung einer s.g. **Zielachse** additiv oder auch exklusiv von den Bewegungen anderer Achsen beeinflusst. Bei der Zielachse handelt es sich entweder um eine weitere Interpolatorachse oder auch eine Spindel.

Achsen, die die Zielachse über die Kopplungsvorschriften beeinflussen, werden als **Quellachsen** bezeichnet. Durch die Definition von Faktoren kann der Bewegungsanteil der Quellachsen entsprechend gewichtet werden. Es lässt sich somit ein elektronisches Getriebe realisieren.

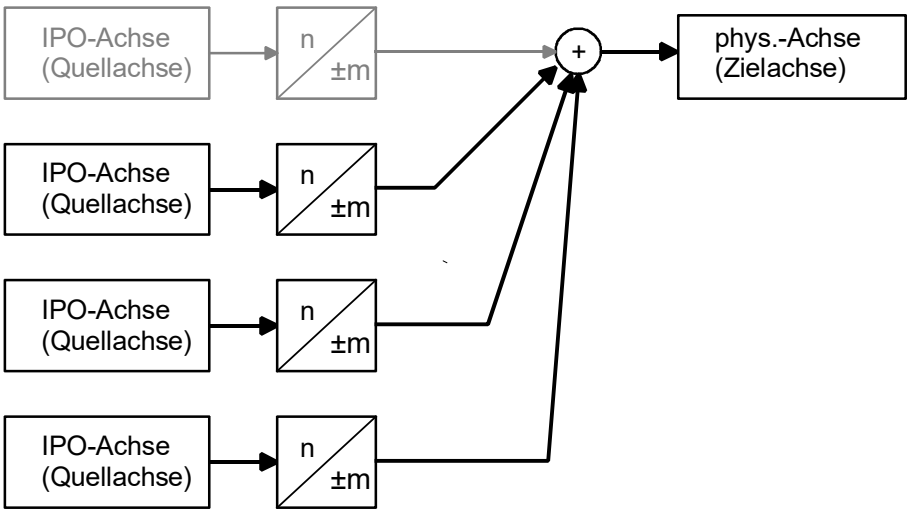


Abb. 123: Ansicht Unterscheidung Quell- und Zielachse

Diese Funktion kann zum einen über das HLI gesteuert werden und ist genauer in [FCT-A9] beschrieben. Durch den NC-Befehl #GEAR LINK kann eine solche Achskopplung aber auch im Teileprogramm parametriert und geschaltet werden.

`Syntax der GEAR-LINK-Programmierung, Parametrierung über Achsnamen:`

#GEAR LINK [TARGET=<*Achsname*> **AX**<*i*>**=**<*Achsname*> **NUM**<*i*>=.. **DENOM**<*i*>=.. {**AX**<*i*>**=**<*Achsname*>
NUM<*i*>=.. **DENOM**<*i*>=..} [**MODE**=<*Modus*>] [**ACC**=..] { \ } **]**

TARGET=<*Achsname*>	Name der Zielachse der Kopplung
AX<*i*>=<*Achsname*>	Name der Quellachse i mit i =1 .. 4
	Achtung: Es dürfen nur kanalspezifische Achsnamen programmiert werden, da nur diese eindeutig sind!
NUM<*i*>=..	Kopplungsfaktor NUM<i> / DENOM<i> für die Quellachse i,
DENOM<*i*>=..	Zähler und Nenner sind negative oder positive Ganzzahlen
MODE=<*Modus*>	Modus beim Ein-/Auskoppeln:
	DIRECT: Kopplung erfordert Stillstand aller beteiligten Achsen (Default). Ist dies nicht der Fall, so wird mit der Fehlermeldung P-ERR-70200 angehalten
	SOFT: Kopplung weich über Slope, Achsen können fahren
ACC=..	Beschleunigungslimit auf der Achse beim Ein-/Auskoppeln in [mm/sec², inch/sec²]. Ohne Angabe von ACC wird P-AXIS-00053 als Default-Beschleunigungslimit verwendet.
\	Trennzeichen ("Backslash") für übersichtliche Programmierung des Befehls über mehrere Zeilen

Syntax der GEAR-LINK-Programmierung, Parametrierung über Achsnummern:

#GEAR LINK [TARGETNR=.. **AXNR**<*i*>**=.. NUM**<*i*>=.. **DENOM**<*i*>=.. {**AXNR**<*i*>**=..**
NUM<*i*>=.. **DENOM**<*i*>=..} [**MODE**=<*Modus*>] [**ACC**=..] [**MCH**] { \ } **]**

TARGETNR=..	Logische Nummer der Zielachse der Kopplung, Positive Ganzzahl
AXNR<*i*>=..	Logische Nummer der Quellachse i mit i =1 .. 4, Positive Ganzzahl
	Achtung: Bei kanalübergreifenden Achskopplungen mit Schlüsselwort MCH dürfen alle im System bekannten logischen Achsnummern programmiert werden, da diese in mehrkanaligen Systemen immer eindeutig sind!
NUM<*i*>=..	Kopplungsfaktor NUM<i> / DENOM<i> für die Quellachse i;
DENOM<*i*>=..	Zähler und Nenner sind negative oder positive Ganzzahlen
MODE=<*Modus*>	Modus beim Einkoppeln:
	DIRECT: Kopplung erfordert Stillstand aller beteiligten Achsen (Default). Ist dies nicht der Fall, so wird mit der Fehlermeldung P-ERR-70200 angehalten
	SOFT: Kopplung weich über Slope, Achsen können fahren. Ohne Angabe von ACC wird P-AXIS-00053 als Default-Beschleunigungslimit verwendet.
ACC=..	Beschleunigungslimit auf der Achse beim Ein-/Auskoppeln in [mm/sec², inch/sec²]
MCH	Wenn MCH gesetzt ist, dürfen kanalübergreifende Achskopplungen definiert werden. Ziel- und Quellachsen können dann aus unterschiedlichen Kanälen stammen. Kanalspezifische Überwachungen (aktive Kopplung bei Programmende bzw. Achsabgabe) sind nicht aktiv.
	Bei Kopplungsmodus DIRECT muss der Anwender sicherstellen, dass Achsen außerhalb des beauftragenden Kanals im Stillstand sind. Kanalspezifische und kanalübergreifende Achskopplungen werden getrennt verwaltet.
\	Trennzeichen ("Backslash") für übersichtliche Programmierung des Befehls über mehrere Zeilen

Syntax Anwahl und Parametrierung in einem Schritt:

#GEAR LINK ON [.. **]**

oder nur Anwahl:

Eine zuvor parametrierte Kopplung wird aktiviert. Wurde keine Kopplung parametriert, erfolgt die Ausgabe einer Fehlermeldung.

#GEAR LINK ON [TARGET=<*Achsname*> | **TARGETNR**=.. [**MCH**] **]**

oder nur Abwahl:

#GEAR LINK OFF [TARGET=<*Achsname*> | **TARGETNR**=.. [**MCH**] **]**

Im NC-Programm wird der Koppelzustand einer Achse mit folgenden Variablen ermittelt::

V.A.GEAR_LINK_ACTIVE.*X* oder **V.A.GEAR_LINK_ACTIVE[***i***]** Achse an einer Kopplung beteiligt?

V.G.GEAR_LINK_ACTIVE Wurde im Kanal eine Achskopplung beauftragt?

Verhalten bei Programmende und Reset:

Sind bei Programmende kanalspezifische Achskopplungen aktiv, erfolgt die Ausgabe der Fehlermeldung P-ERR-70554.

Bei Reset werden alle kanalspezifischen aktiven Achskopplungen deaktiviert.

Kanalübergreifende Achskopplungen werden bei Reset deaktiviert, falls sich die Zielachse im Resetkanal befindet oder zu diesem Zeitpunkt keinem Kanal zugeordnet ist.

Programmierbeispiel

Definition und Anwahl einer kanalspezifischen Kopplungsvorschrift:

```
Parametrierung über Achsnamen:
#GEAR LINK [TARGET=U AX1=X AX2=Y NUM1=1 DENOM1=2 NUM2=-1 DENOM2=1\
            MODE=SOFT ACC=400]
:
TARGET=U          ;Zielachse ist U-ACHSE
AX1=X             ;Quellachse 1 ist X-Achse
NUM1=1 DENOM1=2   ;Kopplungsfaktor für X-Achse ist NUM1/DENOM1=0.5
AX2=Y             ;Quellachse 2 ist Y-Achse
NUM2=-1 DENOM2=1  ;Kopplungsfaktor für Y-Achse ist NUM2/DENOM2=-1
MODE=SOFT         ;Weiches Ein-/Auskoppeln
ACC=400           ;Beschleunigungslimit auf Achse bei 400 mm/sec²

Anwahl der Kopplung:
#GEAR LINK ON [TARGET=U]
Abwahl der Kopplung:
#GEAR LINK OFF [TARGET=U]

Alternativ Parametrierung über Achsnummern:
#GEAR LINK [TARGETNR=4 AXNR1=1 AXNR2=2 NUM1=1 DENOM1=1 NUM2=-1 \
            DENOM2=1]
Anwahl der Kopplung:
#GEAR LINK ON [TARGETNR=4]
Abwahl der Kopplung:
#GEAR LINK OFF [TARGETNR=4]
```

Hinweis

Falls die Zielachse weiterhin durch den Kanal selbst bewegt werden soll, muss sie zusätzlich auch als Quellachse angegeben werden:

```
#GEAR LINK [TARGET=U AX1=U AX2=Y NUM1=1 DENOM1=1 NUM2=-1 DENOM2=1]
```

Programmierbeispiel

Definition und Anwahl einer kanalübergreifenden Kopplungsvorschrift:

Parametrierung über logische Achsnummern:

```
#GEAR LINK [TARGETNR=25 AXNR1=1 AXNR2=40 NUM1=1 DENOM1=2 NUM2=-1\
           DENOM2=1 MODE=SOFT ACC=400 MCH]
:
TARGETNR=25       ;Zielachse ist Achse 25 in Kanal 2
AXNR1=1           ;Quellachse 1 ist Achse 1 im akt. Kanal 1
NUM1=1 DENOM1=2   ;Kopplungsfaktor für Achse 1 ist NUM1/DENOM1=0.5
AXNR2=1           ;Quellachse 2 ist Achse 40 in Kanal 4
NUM2=-1 DENOM2=1  ;Kopplungsfaktor für Achse 40 ist NUM2/DENOM2=-1
MODE=SOFT         ;Weiches Ein-/Auskoppeln
ACC=400           ;Beschleunigungslimit auf Achse 25 bei 400 mm/sec²
MCH               ;Kennung für kanalübergreifende Kopplung
```

Anwahl der Kopplung:

```
#GEAR LINK ON [TARGETNR=25 MCH]
```

Abwahl der Kopplung:

```
#GEAR LINK OFF [TARGETNR=25 MCH]
```

12.40 Einstellungen für Drehfunktionen (#TURN)

Versionshinweis

Diese Funktionalität ist verfügbar ab CNC-Version V3.1.3079.03

Mit dem Befehl #TURN können Drehfunktionalitäten beeinflusst werden.

Syntax:

#TURN [[**ROT_FEED_CPL=**..] [**TAPPING_ACT_POS=**..] [**TAPPING_N_CYCLES=**..]
[**THREAD_CUT_ACT_SPEED =**..] [**THREAD_CUT_N_CYCLES =**..]**]**

ROT_FEED_CPL=..	Einfluss von Achskopplungen auf den Umdrehungsvorschub bei G95 [▶ 654]
	0 : Achskopplungen werden nicht berücksichtigt (Standard)
	1 : Achskopplungen werden berücksichtigt
TAPPING_ACT_POS=..	Aktivieren des Gewindebohrens mit Istpositionen der Spindel. (analog dazu P-CHAN-00761)
	0: Spindel wird an Sollpositionen der Linearachse gekoppelt.
	1: Die Linearachsen werden an die Istpositionen der Spindel gekoppelt.
	Verfügbar ab V3.1.3080.04
TAPPING_N_CYCLES=..	Anzahl der Filtertakte für die Filterung der Istpositionen der Spindel. (analog dazu P-CHAN-00762)
	Wertebereich: 0 … 20
	0: Filter ist deaktiviert.
	Verfügbar ab V3.1.3080.04
THREAD_CUT_ACT_SPEED	Aktivieren des Gewindeschneidens mit Istdrehzahl der Spindel. (analog dazu P-CHAN-00834)
	0: Die Linearachsen werden an die Solldrehzahl der Spindel gekoppelt.
	1: Die Linearachsen werden an die Istdrehzahl der Spindel gekoppelt.
	Verfügbar ab V3.1.3080.16
THREAD_CUT_N_CYCLES	Anzahl der Filtertakte für die Filterung der Istdrehzahl der Spindel. (analog dazu P-CHAN-00835)
	Wertebereich: 0 … 20
	0: Filter ist deaktiviert.
	Verfügbar ab V3.1.3080.16

12.41 Restweganzeige in einem Programmabschnitt (#DIST TO GO)

Versionshinweis

Diese Funktionalität ist verfügbar ab CNC-Version V3.1.3079.27

Mit den NC-Befehlen **#DIST TO GO BEGIN/END** kann eine Geometriesequenz im NC Programm markiert werden. Innerhalb dieser Sequenz kann die verbleibende Distanz der jeweiligen Achse bis zum Ende der Geometriesequenz über dist_to_geom_end auf dem HLI angezeigt werden.

Voraussetzung für die Nutzung der Anzeigefunktionalität ist die Parametrierung von P-STUP-00033.

Syntax:

#DIST TO GO BEGIN

#DIST TO GO END

Eigenschaften:

- Die Distanz der Achse wird immer in die Vorwärtsrichtung angezeigt. Durch Rückwärtsfahren auf der Bahn wird die angezeigte Distanz somit wieder größer.
- Durch eine Geometrieänderung mit der Funktionalität "Restweg verwerfen" sind die angezeigten Werte der Anzeige nicht mehr korrekt. Dieses Fehlverhalten wird durch die Warnung ID 51047 signalisiert.
- Für eine gültige Anzeige der Achsdistanzen bis #DIST TO GO END ist die gesamte Geometriesequenz erforderlich.
  Besonders bei großen Programmsequenzen und klein dimensioniertem P-STUP-00033 kann dies zu einer Verzögerung des Anzeigedatums dist_to_geom_end und der Warnung ID 51048 führen.
  Ist P-STUP-00033 zu klein dimensioniert, so wird ohne Stopp die Sequenz ausgegeben und das Anzeigedatum dist_to_geom_end bleibt solange bei 0 bis ein gültiger Wert für die Anzeige ermittelt werden kann.
- Werden NC-Befehle innerhalb der markierten Geometriesequenz programmiert, die zu einer Kanalsynchronisation führen, so verhindern diese die vorausschauende Restwegberechnung bis zum #DIST TO GO END. Die Restweganzeige wird somit erst nach diesem NC-Befehl korrekt angezeigt.
  Beispiele für diese NC-Befehle:
  - #CHANNEL INIT [▶ 177]
  - #FLUSH WAIT [▶ 341]
  - oder das Lesen einer synchronen V.E.-Variablen.

Programmierbeispiel

Restweganzeige in einem Programmabschnitt

```
N010 G01 X0 Y0 F5000
N020 #DIST TO GO BEGIN
N030 G01 X10
N040 G01 Y10
N050 G01 X5
N060 #DIST TO GO END
;…
```

Die Anzeige würde in dem Beispiel am Anfang der Sequenz für X 15 und Y 10 anzeigen und am Ende jeweils 0.

Der Verlauf der Anzeigenwerte ist in folgender Abbildung dargestellt:

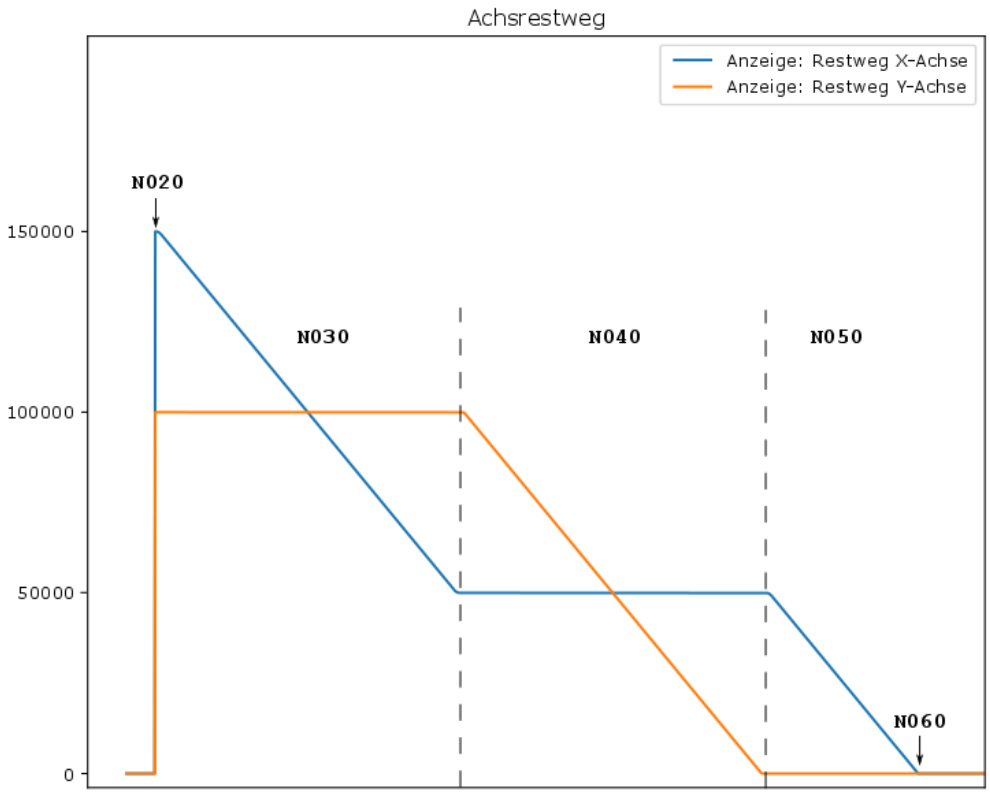


Abb. 124: Restweganzeige in einem Programmabschnitt

12.42 Umschalten der Auflösung an der externen Geschwindigkeitsschnittstelle der PLC (#EDM ON/OFF)

Standardmäßig hat die externe Geschwindigkeitsschnittstelle eine Auflösung von [µm/s]. Anspruchsvollere Technologien wie z.B. Draht-/Senkerodieren erfordern jedoch höhere Auflösungen. Mit diesem NC-Befehl kann die Auflösung deshalb auf nm/s umgestellt werden.

Syntax:

#EDM ON | **OFF**

ON	Auf höhere Geschwindigkeitsauflösung [nm/s] umschalten
OFF	Auf Standard-Geschwindigkeitsauflösung [µm/s] zurückschalten

13 Werkzeuggeometriekorrektur (D)

Werkzeuggeometrien werden in Länge, Radius und Achsversatz korrigiert. Die entsprechenden Korrekturdaten für die Werkzeuggeometrie werden zur Verfügung gestellt entweder durch:

Werkzeuglisten, die entsprechende Datensätze für jedes Werkzeug enthalten und bei Steuerungshochlauf geladen werden (siehe [TOOL]), oder eine externe Werkzeugverwaltung.

Werkzeugkorrekturdaten

Werkzeugkorrekturdaten werden mit dem D-Wort bzw. bei entsprechender Parametrierung automatisch mit dem T-Wort (P-CHAN-00014) ausgewählt:

Syntax:

D.. Werkzeugkorrekturanwahl mit Nummer des Korrekturdatensatzes. Positive Ganzzahl. modal

Das D-Wort definiert einen Werkzeugdatensatz, der folgende Werte enthält:

- Werkzeuglänge (senkrecht zur Hauptebene)
- Werkzeugradius (in der Hauptebene)
- Achsversatzkoordinaten (in allen Achsrichtungen)
- Maßeinheit der Zahlenangaben (mm/Inch)
- Werkzeug-Gültigkeitskennung
- Werkzeugdynamikdaten

Achtung

Der Zeitpunkt der Wirksamkeit der Werkzeugkorrekturdaten für die Werkzeuglänge (in der Senkrechten zur Hauptebene) und für die Achsversatzkoordinaten (in den Achsrichtungen) wird über den Kanalparameter P-CHAN-00100 bestimmt. Die folgenden Zeilen gilt es ebenfalls zu beachten:

2 Modi werden hierbei unterschieden:

Die Ausgleichsbewegung wird direkt mit dem D-Wort ohne Angabe einer Wegbedingung ausgeführt.

Die Steuerung führt die Ausgleichsbewegung aus Sicherheitsgründen erst mit dem nächsten **absoluten** Verfahrsatz in der entsprechenden Achse aus (Defaultmodus).

Dabei gilt für die Einrechnung der Werkzeugkorrekturdaten im G91-Modus:

Die Programmierung von...

```
N10 D16
N20 G0 X0 G91
```

...darf keine Bewegung der X-Achse hervorrufen (entspricht Bewegung relativ um 0). Somit wirken sich die Werkzeugkorrekturdaten für diese Achse erst dann aus, wenn die nächste Verfahrinformation absolut (G90) programmiert wird.

Für beide Modi gemeinsam gilt: Der Werkzeugradius wird an die Werkzeugradius-korrektur übergeben und beeinflusst hier die Äquidistanten-Berechnung. Die Ausgleichsbewegung erfolgt hier immer mit einer Wegbedingung.

Die Regeln für die Ausführung der Ausgleichsbewegung gelten entsprechend auch bei der Abwahl der Werkzeugkorrektur:

Syntax:

D00 Werkzeugkorrekturabwahl modal

13.1 Werkzeuglängenkorrektur

Die Werkzeuglängenkorrektur (WLK) wirkt bei kartesischen Maschinen immer in Richtung der 3. Hauptachse. Dies ist bei G17 im Allgemeinen die werkzeugtragende Z-Achse.

Wenn das Werkzeug ausgehend vom Einspannpunkt in die negative Richtung der Z-Achse zeigt, dann erfolgt die Ausgleichsbewegung in positiver Richtung der Z-Achse.

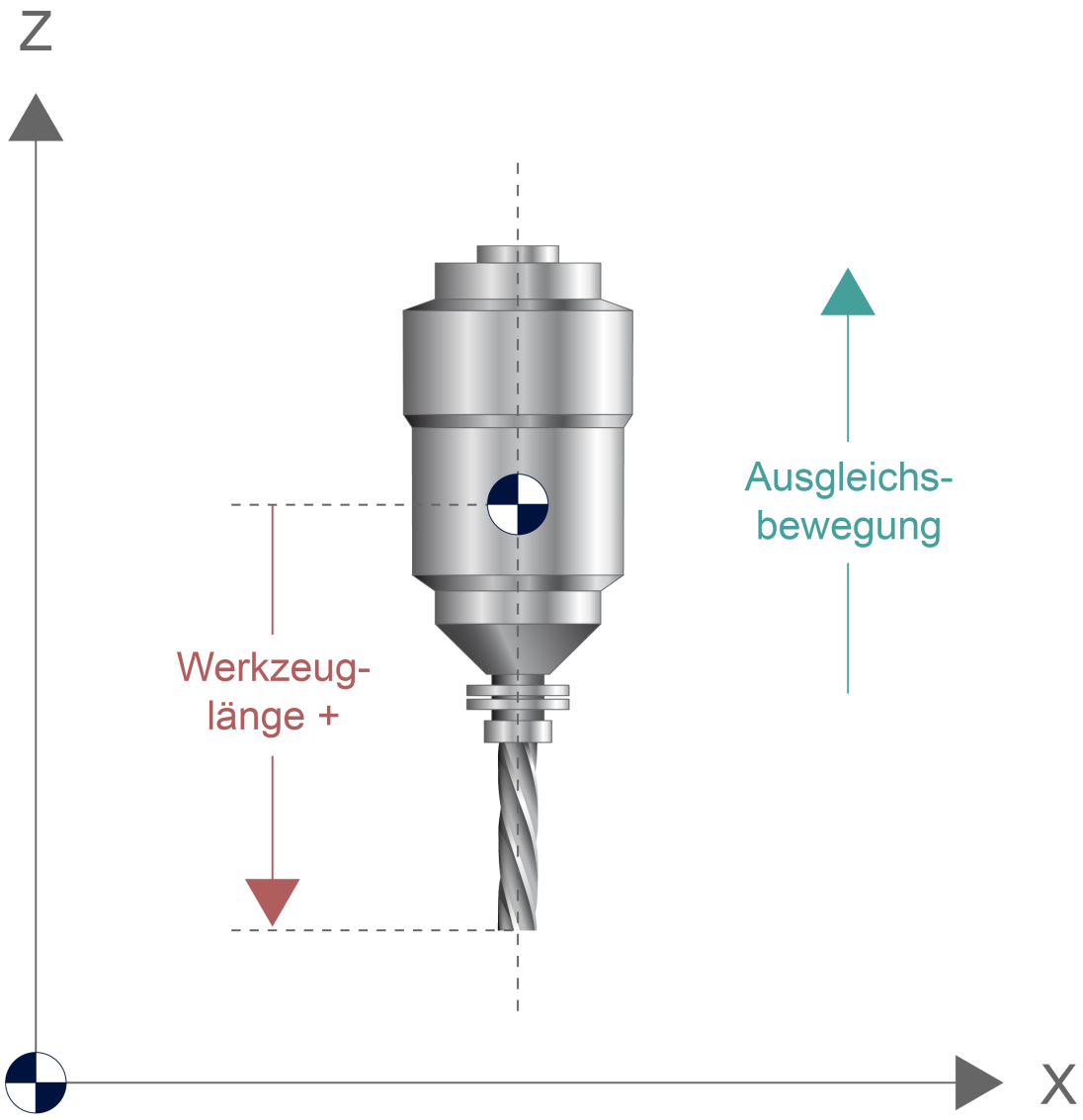


Abb. 125: Korrektur der Werkzeuglänge durch Ausgleichsbewegung

Achtung

Ist durch den mechanischen Maschinenaufbau eine andere werkzeugtragende Achse vorgegeben, so ist vom Anwender ein geeignetes Programmierkoordinatensystem (PCS) zu wählen (G18, G19, #CS, #TRAFO), damit die Werkzeuglänge richtig eingerechnet wird.

Im folgenden Beispiel wird die Werkzeuglängenkorrektur in Z-Richtung durchgeführt. Bei der Anwahl des Korrekturdatensatzes D16 in Satz N30 erfolgt die Ausgleichsbewegung in Z-Richtung gemeinsam mit dem Verfahrdatensatz in N30.

Programmierbeispiel

```
N10 G01 F900 G17 ;X-Y-Ebene, Längenkorrektur in Z+ (Default)
N20 X150 Y10 Z10
N30 D16 Y40 Z15 ;Anwahl der Längenkorrektur D16. Die
:               ;Ausgleichsbewegung wird durchgeführt.
:
N100 D20 ;Anwahl der Längenkorrektur D20. Die
:        ;Ausgleichsbewegung erfolgt erst mit dem
:        ;nächsten absoluten Verfahrsatz in Z-Richtung.
:
N200 G0 D0 X0 Y0 Z0 ;Abwahl der WLK

Korrekturdatensatz
D0 : Länge = 0 Radius = 0
D16: Länge = 5 Radius = 5
D20: Länge = 12,5 Radius = 5
```

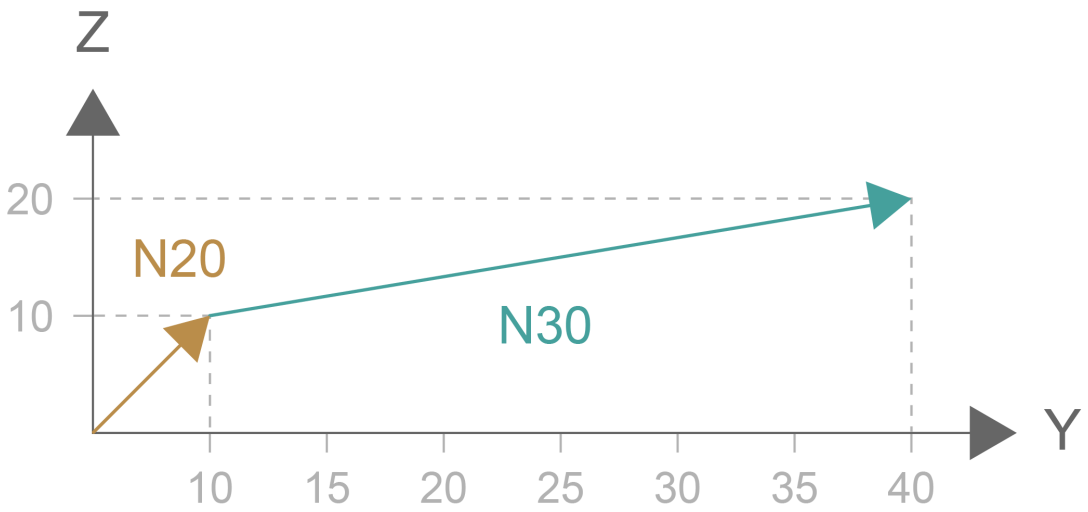


Abb. 126: Beispiel zur Werkzeuglängenkorrektur

13.1.1 Achsspezifische Zuordnung der Werkzeuglängenkorrektur (#TLAX, #TLAX DEFAULT)

In der Grundeinstellung wird bei einem Ebenenwechsel (G17, G18, G19) die Werkzeuglänge immer in der dritten Hauptachse der neuen Ebene berücksichtigt. Das ist dann sinnvoll, wenn der Werkzeugkopf (bzw. die Werkzeugachse) orientierbar ist und in der jeweiligen Ebene eine Bearbeitung stattfindet.

Dieses Verhalten ist bei nicht orientierbaren (feststehenden) Werkzeugköpfen nicht erforderlich. Dient ein Wechsel der Ebene nur zur Programmierung einer Anfahrbewegung, so kann die Werkzeuglänge und ihre Orientierung in der ursprünglichen Hauptachse verbleiben bzw. durch folgenden NC-Befehl einer bestimmten Hauptachse fest zugeordnet werden:

Syntax:

#TLAX [<*Achsname*> **+** | **-** **]**

<*Achsname*> Name der Hauptachse, in die die Werkzeuglänge eingerechnet werden soll mit Angabe der Orientierung + oder -

Es gilt folgende Vereinbarung der Orientierungszuordnung:

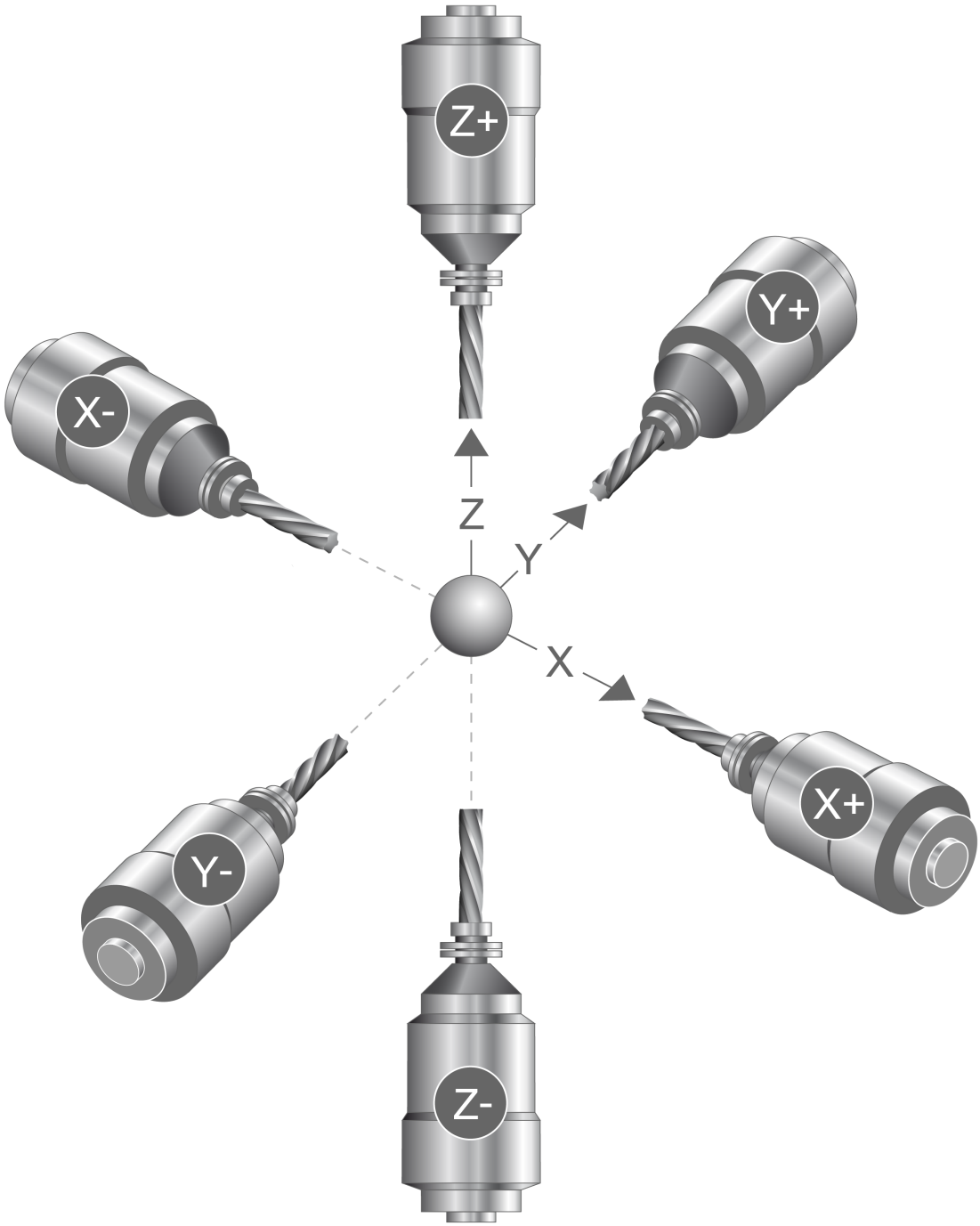


Abb. 127: Zuordnungsregel der WZ-Längenkorrektur

Im Grundzustand (G17) wird die Werkzeuglänge in die dritte Hauptachse (Z) eingerechnet. Die Richtung (Orientierung) wird durch das Vorzeichen '+'angegeben. Das entspricht z.B. in der X-Y Ebene dem Befehl #TLAX [Z+].

Bei Maschinen mit nicht orientierbaren Werkzeugköpfen, die überwiegend in der 3D-Bearbeitung in G17 eingesetzt werden, kann dieses Verhalten auch durch den Kanalparameter P-CHAN-00420 voreingestellt werden; die Programmierung von #TLAX ist dann nicht notwendig.

Der nachfolgende Befehl hebt die feste Zuordnung der Werkzeuglängenkorrektur zu einer bestimmten Achse auf. Die Werkzeuglänge ist dann wieder in der dritten Hauptachse der jeweils aktuellen Ebene berücksichtigt.

Syntax:

#TLAX DEFAULT

Programmierbeispiel

```
Überschleifen zirkularer Anfahrbewegungen in verschiedenen Ebenen mit
konstanter Orientierung der Werkzeuglängenkorrektur
%tlax
N010 G0 X0 Y0 Z0
N020 V.G.WZL=33 G161
N030 X0 Y0 Z0
N040 #TLAX [Z+]  ;Werkzeuglängenkorrektur in Z+
N050 G18
N060 G01 X0 Z50 F2000
N070 G02 X50 Z0 I50 K50 F1000
N080 G17
N090 G03 X100 Y50 I50 J50
N100 G19
N110 G03 Y0 Z50 J50 K50 F1000
N120 G18
N130 G02 X0 Z50 I50 K50 F1000
N140 #TLAX DEFAULT  ;Abwahl Werkzeuglängenkorrektur in Z+,
                    ;Werkzeuglänge wird in Y (G18) eingerechnet
N150 G17
N160 M30
```

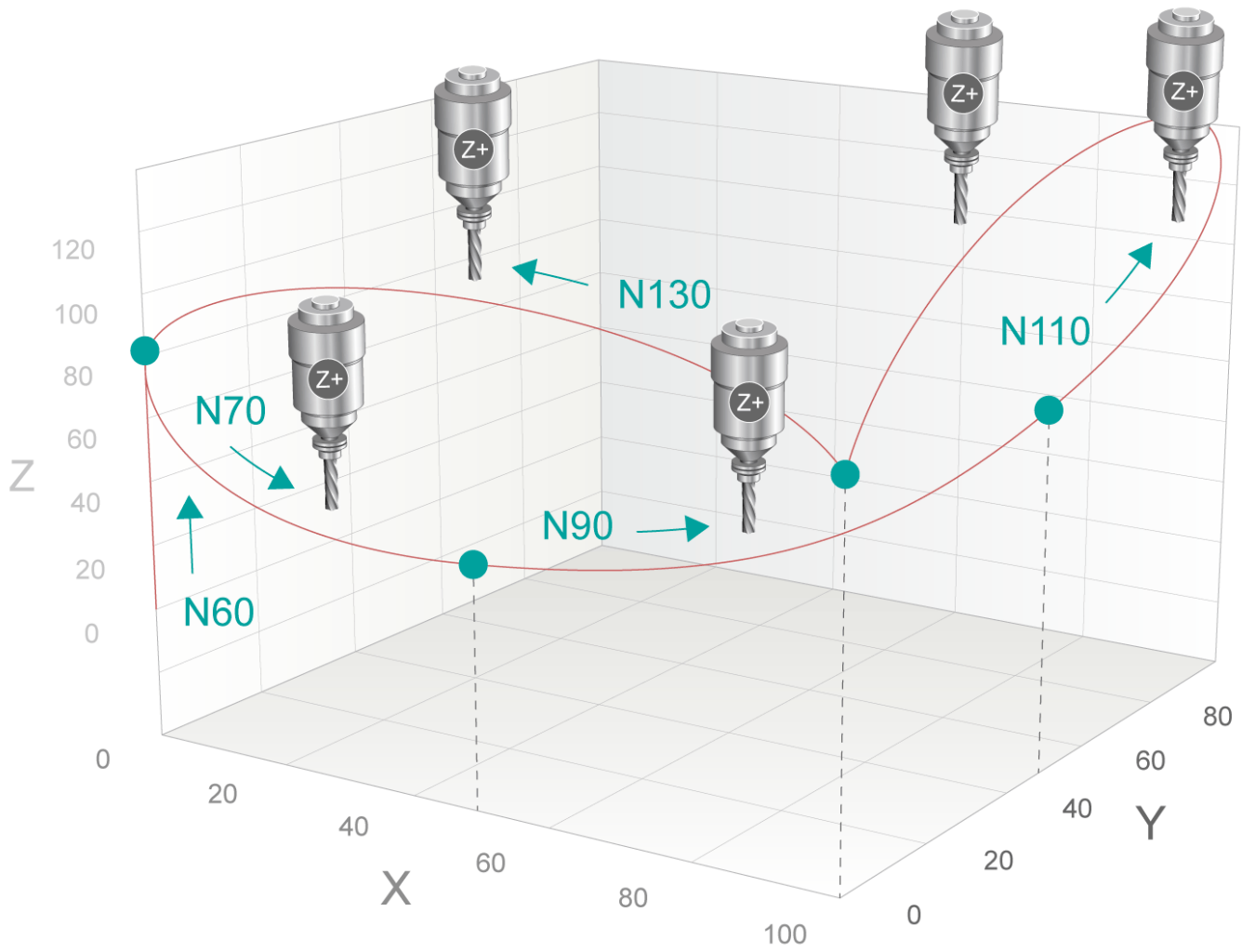


Abb. 128: Überschleifen von Anfahrbewegungen in G17, G18, G19 mit konstanter Orientierung der Werkzeuglängenkorrektur

13.2 Werkzeugradiuskorrektur (WRK)

Die Werkzeugradiuskorrektur (WRK) erlaubt die Programmierung der Werkstückkontur unabhängig von der Geometrie des Werkzeugs. Bei angewählter WRK (G41, G42) wird eine zu dieser programmierten Werkstückkontur äquidistante Werkzeugbahn im Abstand "Werkzeugradius" berechnet.

Die Werkzeugradiuskorrektur wirkt in der mit G17, G18 oder G19 angewählten Ebene. Als Werkzeugkorrekturwerte werden die unter den D-Worten abgelegten Werkzeugkorrektursätze verwendet (s. Werkzeuggeometriekorrektur [▶ 499]).

Durch Anwahl eines neuen D-Wortes oder Beschreiben der Variablen V.G.WZ_AKT.R kann der Werkzeugradius auch bei aktiver WRK geändert werden.

Bei Verwendung eines negativen Werkzeugradius' wird automatisch die Anwahlseite der WRK gewechselt.

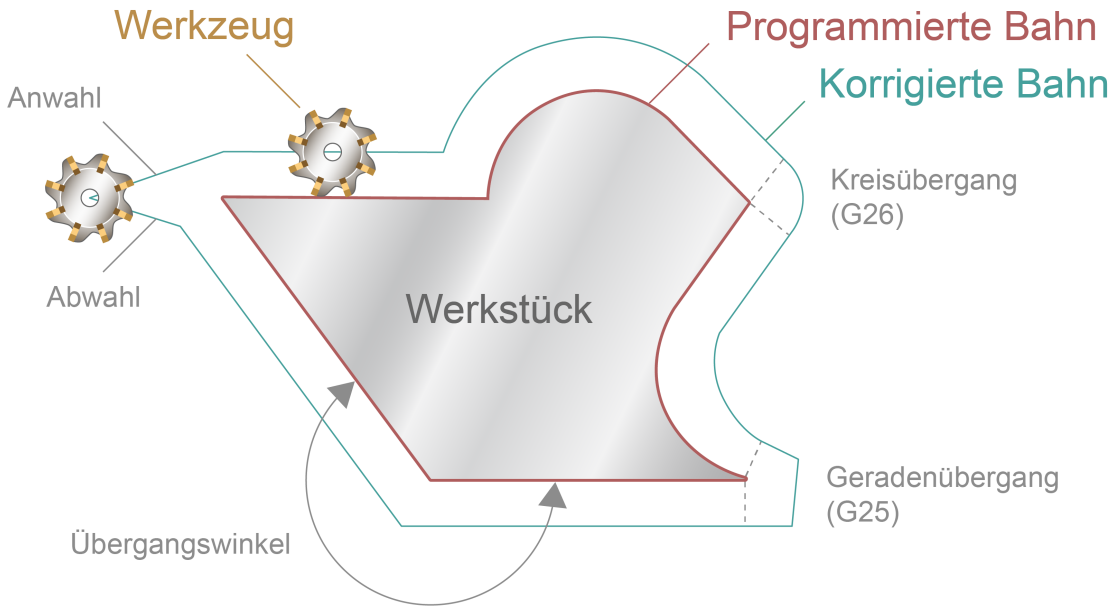


Abb. 129: Wirkungsweise und Begriffe der Werkzeugradiuskorrektur

Übersicht aller WRK-relevanten G-Funktionen:

An-/Abwahl der WRK

G40	WRK-Abwahl	(modal, Grundzustand)
G41	Anwahl WRK links der Kontur	(modal)
G42	Anwahl WRK rechts der Kontur	(modal)

An-/Abwahlverfahren der WRK

G138 [▶ 515]	Direkte An-/Abwahl der WRK	(modal)
G139 [▶ 521]	Indirekte An-/Abwahl der WRK	(modal, Grundzustand)
G236 [▶ 548]	Direkte An-/Abwahl der WRK auf die Bahn	(modal)
G237 [▶ 533]	Lotrechte An-/Abwahl der WRK	(modal)
G238 [▶ 540]	Inneneckanwahl der WRK	(modal)
G239 [▶ 543]	Direkte An-/Abwahl der WRK ohne Satz	(modal)
G05 [▶ 564]	Tangentiale An-/Abwahl der WRK	(nicht modal)

Achtung

Ein Wechsel des modalen Anwahlverfahrens zu G238 bei angewählter WRK ist nicht zulässig.

Vorschubanpassung:

G10 [▶ 567]	Vorschub konstant	(modal, Grundzustand)
G11 [▶ 567]	Vorschub angepasst	(modal)

Konturausblendung:

G140 [▶ 568]	Abwahl der Konturausblendung	(modal, Grundzustand)
G141 [▶ 568]	Anwahl der Konturausblendung	(modal)

Auswahl der Außenübergänge:

Bei angewählter Werkzeugradiuskorrektur müssen Außenecken umfahren werden. Dazu fügt die Werkzeugradiuskorrektur sogenannte Übergangssätze ein. Mit G25 und G26 kann zwischen dem Einfügen von Linearsätzen und dem Einfügen von Zirkularsätzen ausgewählt werden.

G25	Geradenübergang	(modal, Grundzustand)
G26	Kreisübergänge	(modal)

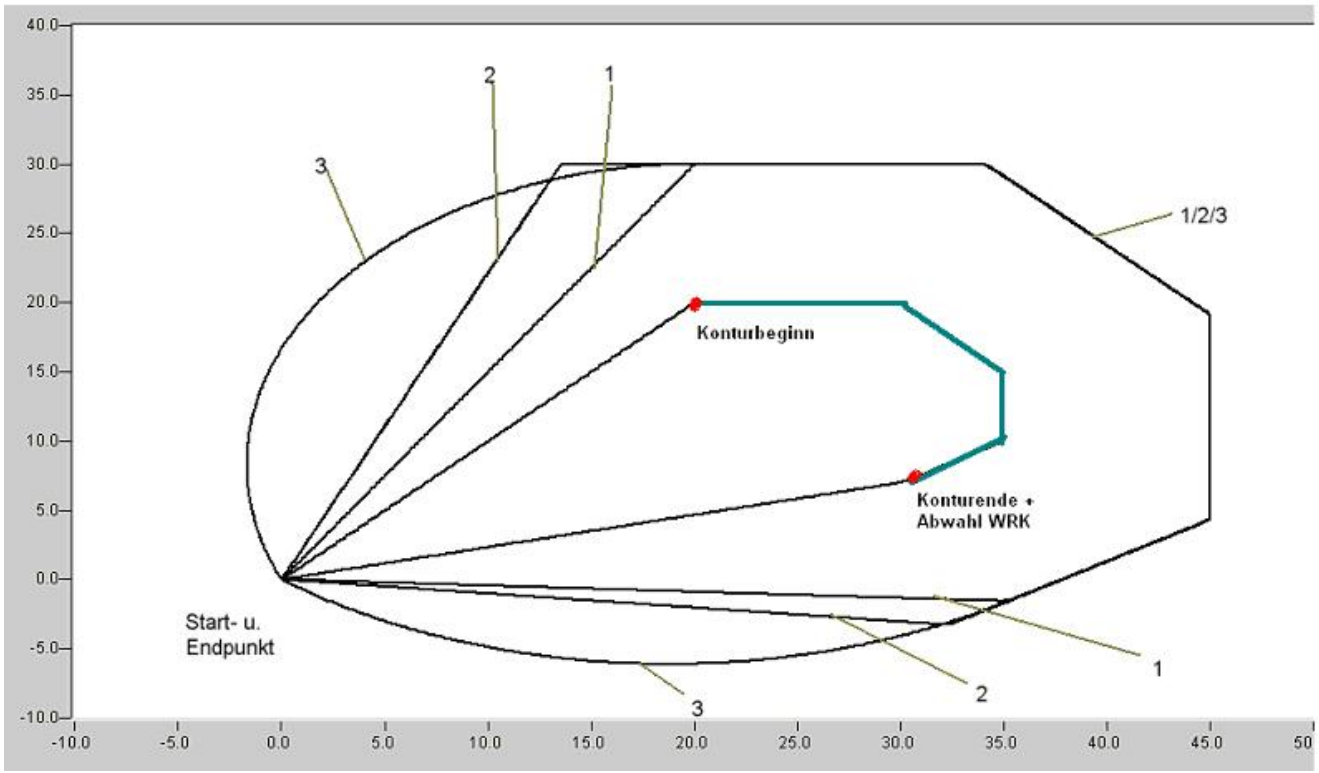
Programmierbeispiel

Darstellung der unterschiedlichen Anwahlmöglichkeiten bei Einsatz der Werkzeugradiuskorrektur. Es werden die Befehle der direkten, indirekten und tangentialen An-/Abwahl der WRK (G139/ G138/G05) in Kombination mit den WRK-Übergangsarten für Geraden- und Kreisübergang (G25/ G26) veranschaulicht.

Die Testprogramme werden mit einem Werkzeugradius von 10mm gefahren.

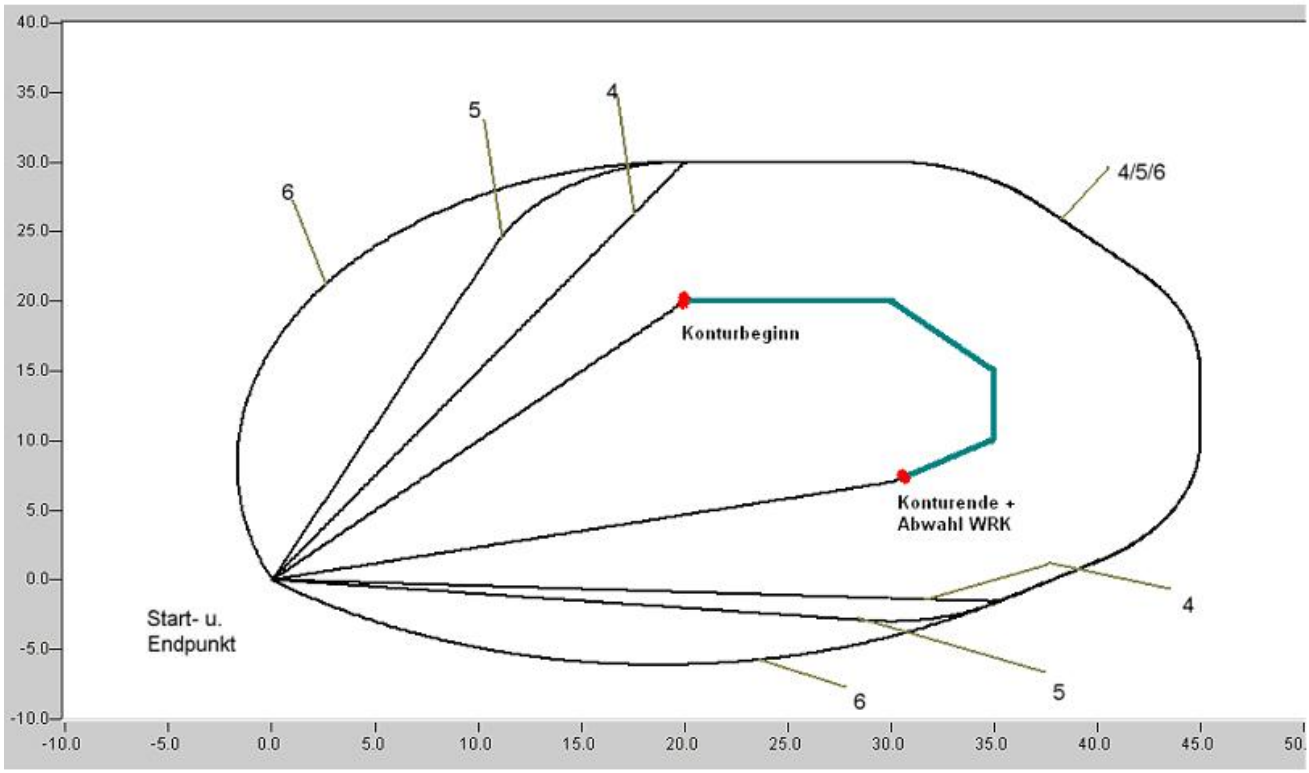
Beispiel 1	G138	G139	G05
G25	**Bahn 1**	**Bahn 2**	**Bahn 3**

```
%WZKG25 (Muster Kontur für G25)
N1 G00 G90 T1 D1 X0 Y0 Z0 G17
(Kontur Darstellung)
N15 G01 X20 Y20 F1000
N20 G91
N25 G1 X10
N30 X5 Y-5
N35 Y-5
N40 X-5 Y-3
N45 G01 G90 X0 Y0 F1000

(Bahn 1)

N100 G138 G41            (Direkte Anwahl u. WRK links der Kontur)
N105 G01 X20 Y20 F1000   (Notwendige Ausgleichsbewegung nach G41)
N110 G25                 (G25 Geradenübergänge)
N115 G1 G91 X10
N120 X5 Y-5
N125 Y-5
N130 X-5 Y-3
N135 G138 G40            (Direkte Abwahl u. WRK-Abwahl)
N140 G01 G90 X0 Y0 F1000 (Notwendige Ausgleichsbewegung nach G40)

(Bahn 2)

N200 G139 G41            (Indirekte Anwahl u. WRK links der Kontur)
N205 G01 X20 Y20 F1000   (Notwendige Ausgleichsbewegung nach G41)
N210 G25                 (G25 Geradenübergänge)
N215 G1 G91 X10
N220 X5 Y-5
N225 Y-5
N230 X-5 Y-3
N235 G139 G40            (Indirekte Abwahl u. WRK-Abwahl)
N240 G01 G90 X0 Y0 F1000 (Notwendige Ausgleichsbewegung nach G40)

(Bahn 3)

N300 G05 G41             (Tangentiale Anwahl u. WRK links der Kontur)
N305 G01 X20 Y20 F1000   (Notwendige Ausgleichsbewegung nach G41)
N310 G25                 (G25 Geradenübergänge)
N315 G1 G91X10
N320X5 Y-5
N325 Y-5
N330 X-5 Y-3
N335 G05 G40             (Tangentiale Abwahl u. WRK-Abwahl)
N340 G01 X20 Y20 F1000   (Notwendige Ausgleichsbewegung nach G41)

N999 M30
```


40.0
35.0
30.0
25.0
20.0
15.0
10.0
5.0
0.0
-5.0
-10.0
-10.0
-5.0
0.0
5.0
10.0
15.0
20.0
25.0
30.0
35.0
40.0
45.0
50
2
1
3
1/2/3
Konturbeginn
Konturende +
Abwahl WRK
Start- u.
Endpunkt
1
2
3

Beispiel 2	G138	G139	G05
G26	**Bahn 4**	*Bahn 5*	**Bahn 6**

```
%WZKG26 (Muster Kontur für G26)
N10 G00 G90 T1 D1 X0 Y0 Z0 G17
(Kontur Darstellung)
N15 G01 X20 Y20 F1000
N20 G91
N25 G1 X10
N30 X5 Y-5
N35 Y-5
N40 X-5 Y-3
N45 G01 G90 X0 Y0 F1000

(Bahn 4)

N400 G138 G41            (Direkte Anwahl u. WRK links der Kontur)
N405 G01 X20 Y20 F1000   (Notwendige Ausgleichsbewegung nach G41)
N410 G26                 (G26 Kreisübergänge)
N415 G1 G91 X10
N420 X5 Y-5
N425 Y-5
N430 X-5 Y-3
N435 G138 G40            (Direkte Abwahl u. WRK-Abwahl)
N440 G01 G90 X0 Y0 F1000 (Notwendige Ausgleichsbewegung nach G40)

(Bahn 5)

N500 G139 G41            (Indirekte Anwahl u. WRK links der Kontur)
N505 G01 X20 Y20 F1000   (Notwendige Ausgleichsbewegung nach G41)
N510 G26                 (G26 Kreisübergänge)
N515 G1 G91 X10
N520 X5 Y-5
N525 Y-5
N530 X-5 Y-3
N535 G139 G40            (Indirekte Abwahl u. WRK-Abwahl)
N540 G01 G90 X0 Y0 F1000 (Notwendige Ausgleichsbewegung nach G40)

(Bahn 6)

N600 G05 G41             (Tangentiale Anwahl u. WRK links der Kontur)
N605 G01 X20 Y20 F1000   (Notwendige Ausgleichsbewegung nach G41)
N610 G26                 (G26 Kreisübergänge)
N615 G1 G91X10
N620 X5 Y-5
N625 Y-5
N630 X-5 Y-3
N635 G05 G40             (Tangentiale Abwahl u. WRK-Abwahl)
N640 G01 X20 Y20 F1000   (Notwendige Ausgleichsbewegung nach G41)

N999 M30
```


40.0
35.0
30.0
25.0
20.0
15.0
10.0
5.0
0.0
-5.0
-10.0
-10.0
-5.0
0.0
5.0
10.0
15.0
20.0
25.0
30.0
35.0
40.0
45.0
50.
6
5
4
4/5/6
Konturbeginn
Konturende +
Abwahl WRK
Start- u.
Endpunkt
4
5
6

Änderung von Werkzeugdaten

Programmierbeispiel

Änderung von Werkzeugdaten

```
:
N30 G0 D0 X0 Y0 Z0 G17  (X-Y-Ebene)
N40 G0 D100 X10 Y10     (Anwahl der WLK)
N50 G1 Z0
N60 G0 Z100
N70 G41                 (Anwahl der WRK mit Datensatz D100)
N80 G1 Z0
N90 G2 X10 Y10 I-15     (Vollkreis mit Radius 15)
N100 G0 Z100
N110 D2 Z200            (Anderer Korrekturdatensatz, d.h)
(andere Werte für WLK und WRK)
N120 G1 Z0              (Hier erfolgt die Ausgleichs-)
                        (bewegung der WLK)
N130 G1 X20 Y20         (Ausgleichsbewegung der WRK)
N140 G02 X10 Y10 I-15   (Die WRK wird jetzt mit Datensatz)
                        (D2 durchgeführt)
N150 G0 Z100
N160 G40                (Abwahl der WRK)
N170 X0
:
```

Dynamische Änderung des Werkzeugradius

Eine weitere Möglichkeit, den Werkzeugradius zu ändern, ist über die Zuweisung durch Variablen gegeben (s. a. Kap. 13 [▶ 571]). Dadurch kann z.B. die Abnutzung von Schleifwerkzeugen während der Verfahrsätze berücksichtigt werden.

Programmierbeispiel

Dynamische Änderung des Werkzeugradius

```
N00 G1 G90 X0 D6 F5
N10 G41 G26                      (Anwahl WRK)
N20 X0 Y250                      (Startpunkt)
N30 V.P.VERSCHLEISS = 0.010
N100 $FOR V.P.LAUF = 0,100,10    (Schleifzyklus)
N110 X300
N120 Y200
N130 X0
N140 Y250                        (Werkzeugradius wird immer kleiner)
N150 V.G.WZ_AKT.R = V.G.WZ_AKT.R - V.P.VERSCHLEISS
N160 $ENDFOR

N200 G40 X300                    (Abwahl WRK)
N999 M30
```

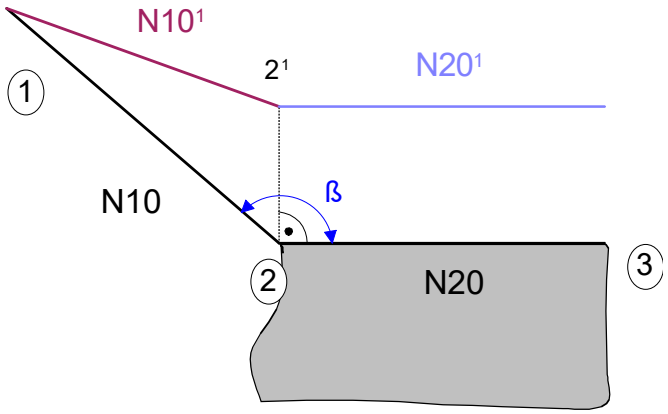
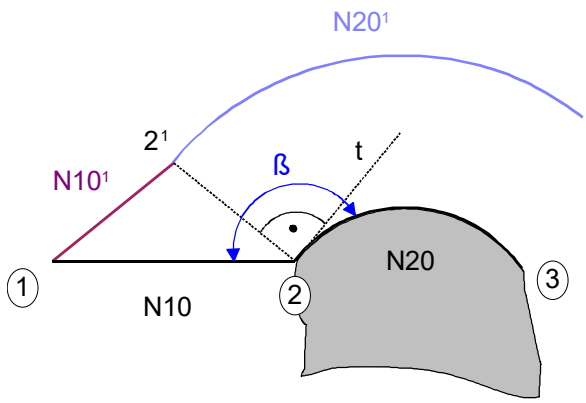
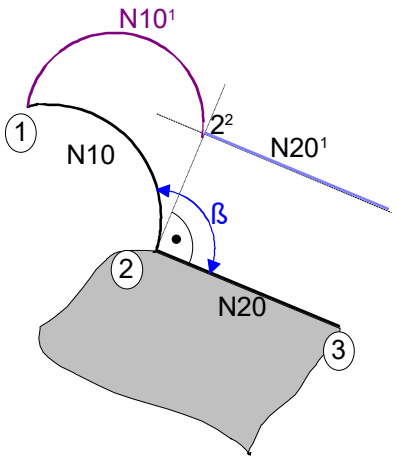
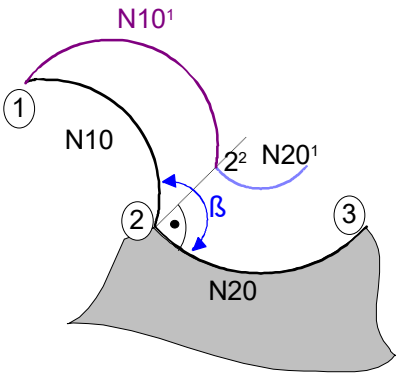
13.2.1 Direkte/ indirekte Anwahl der WRK

Bei direkt oder indirekt angewählter WRK wird mit dem nachfolgenden Verfahrsatz im NC-Programm in der angewählten Korrektur-Ebene ein Anfahrsatz generiert.

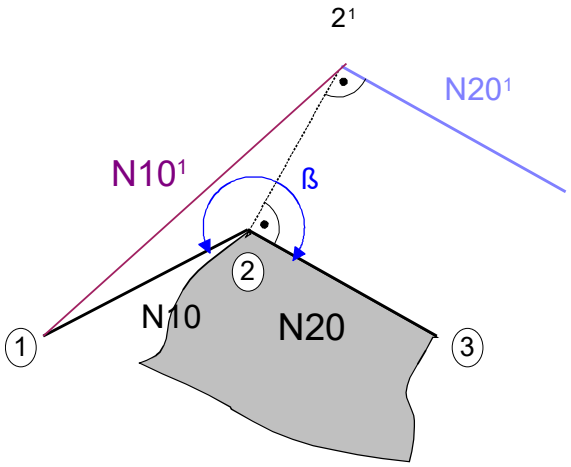
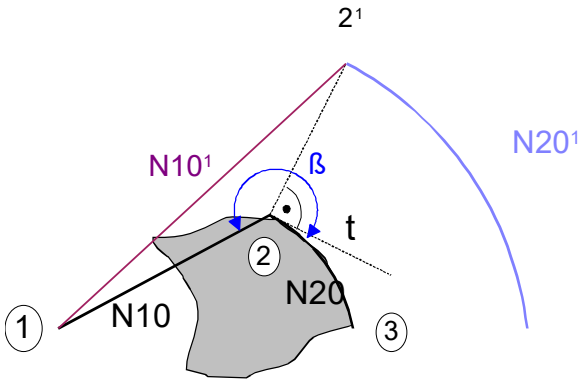
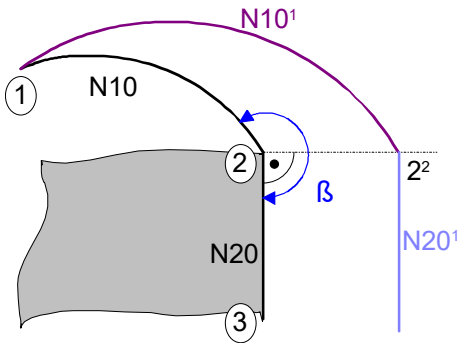
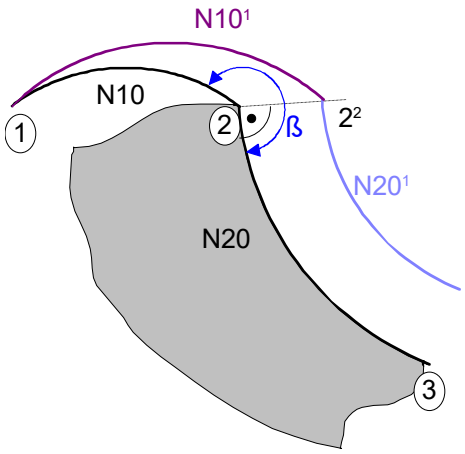
Bei der WRK-Anwahl kann mit G138 neben der Standardanfahrstrategie (indirekte Anwahl der WRK) eine alternative Anfahrstrategie (direkte Anwahl der WRK) angewählt werden.

Die folgenden Bilder zeigen alle möglichen Anfahrsätze bei den zugelassenen Konturübergängen. Dargestellt sind jeweils 2 NC-Sätze N10 und N20 für die drei relevanten Konturübergangswinkel.

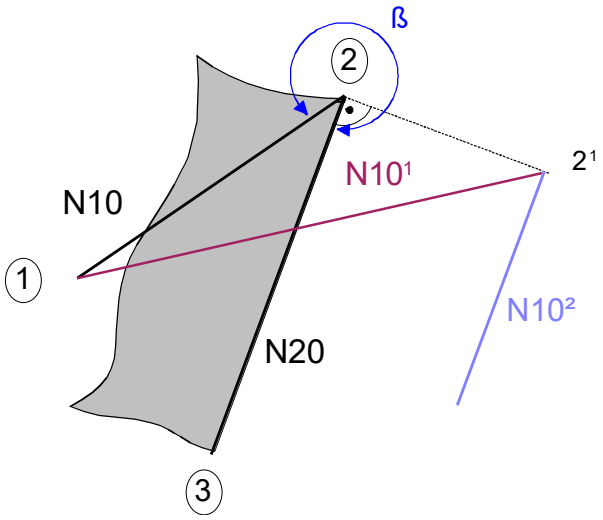
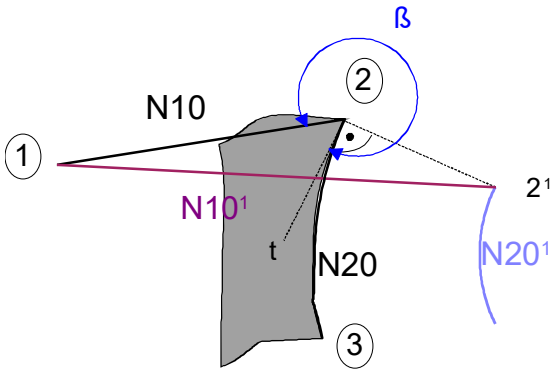
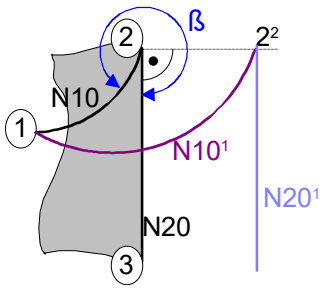
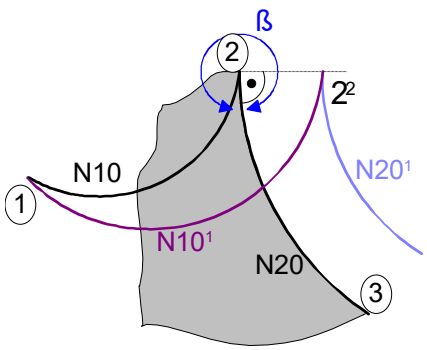
13.2.1.1 Direkte Anwahl (G138/G41/G42)

Kombinationstyp 1	Kombinationstyp 2
	
Kombinationstyp 3	**Kombinationstyp 4**
	

Konturübergangsbereich 0° < ß ≤ 180°

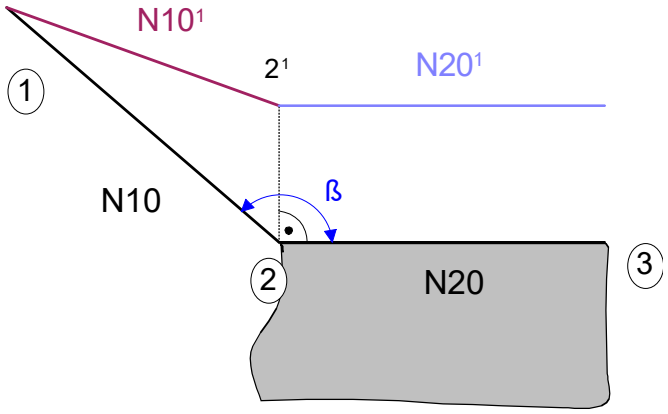
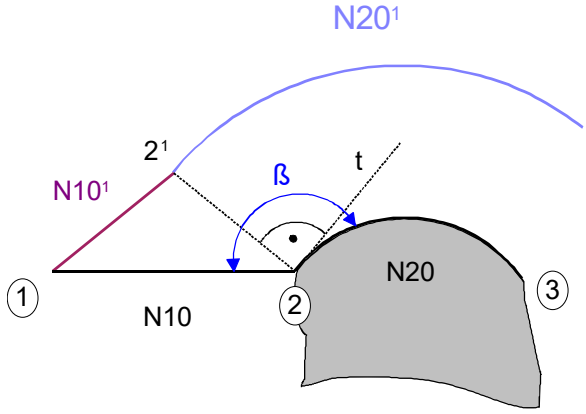
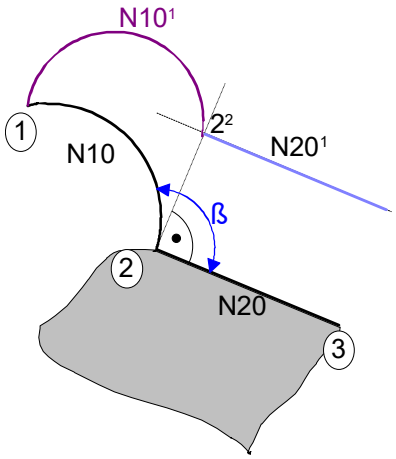
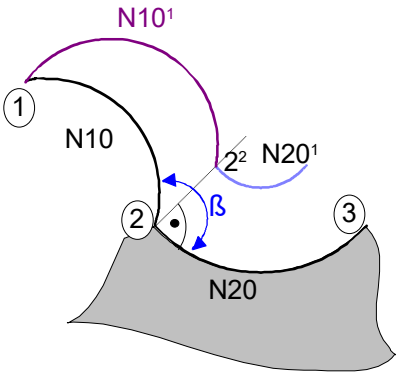
Kombinationstyp 1	Kombinationstyp 2
	
Kombinationstyp 3	**Kombinationstyp 4**
	

Konturübergangsbereich 180° < ß ≤ 270°

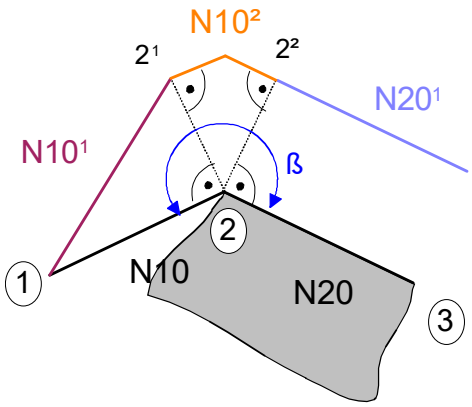
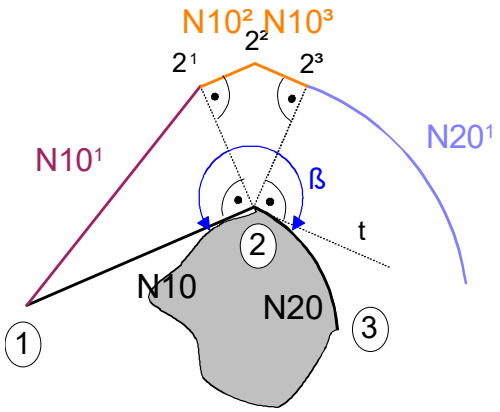
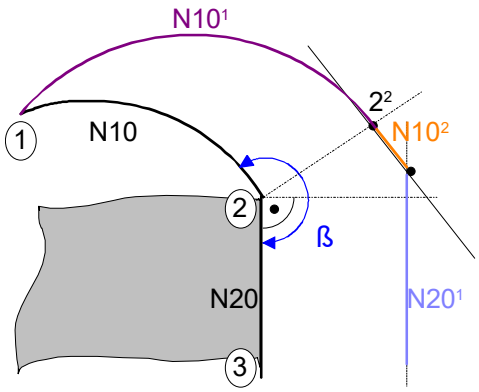
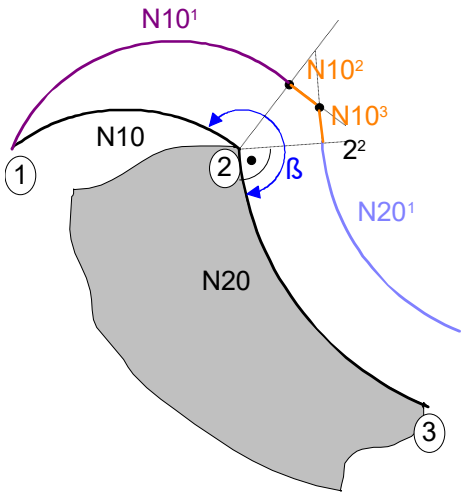
Kombinationstyp 1	Kombinationstyp 2
	
Kombinationstyp 3	**Kombinationstyp 4**
	

Konturübergangsbereich 270° < ß ≤ 360°

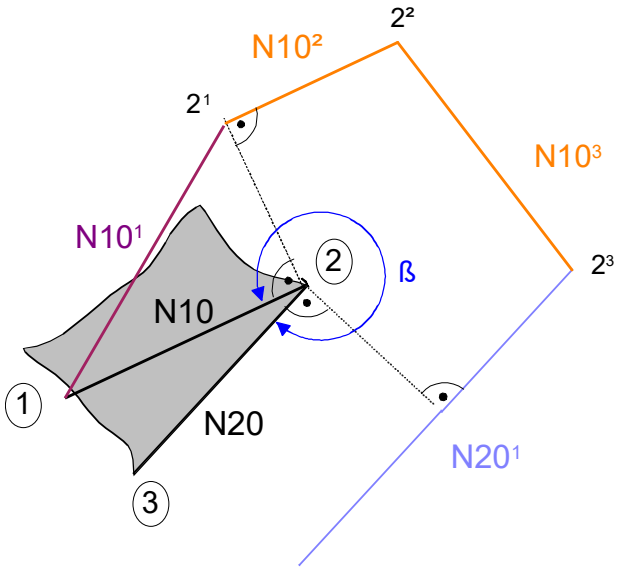
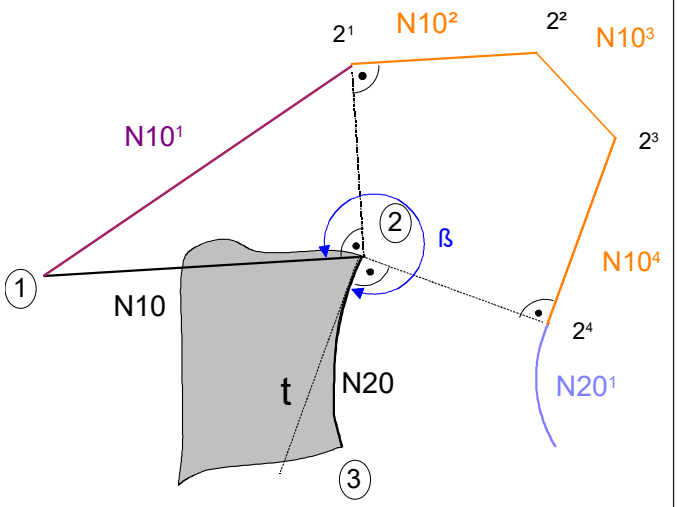
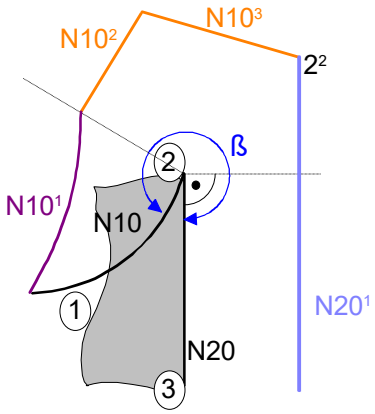
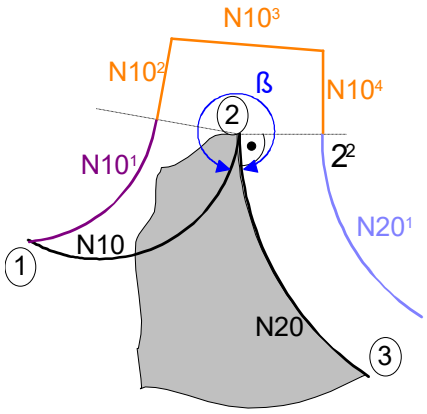
13.2.1.2 Indirekte Anwahl (G139/G41/G42) mit G25

Kombinationstyp 1	Kombinationstyp 2
	
Kombinationstyp 3	**Kombinationstyp 4**
	

Konturübergangsbereich 0° < ß ≤ 180°

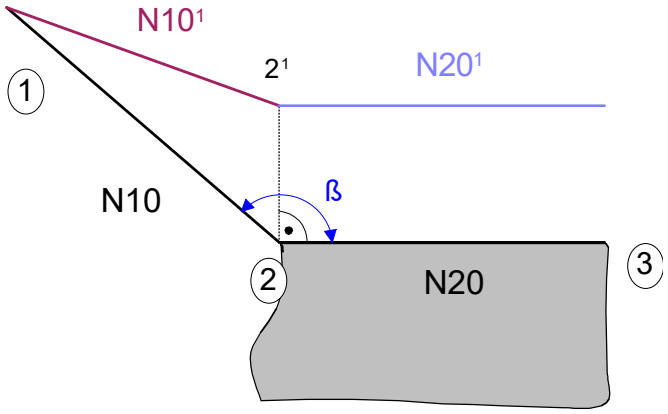
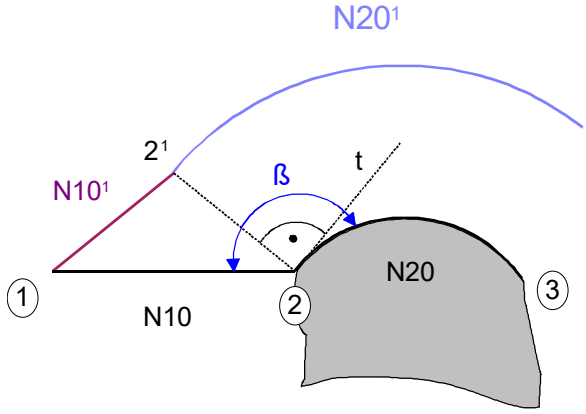
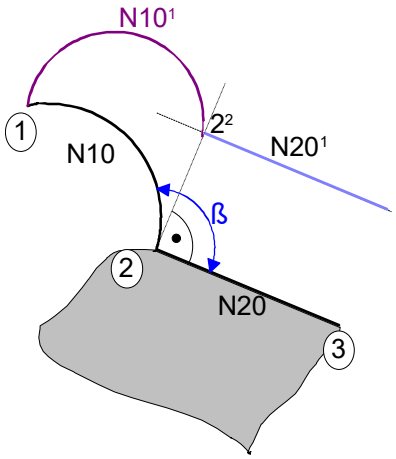
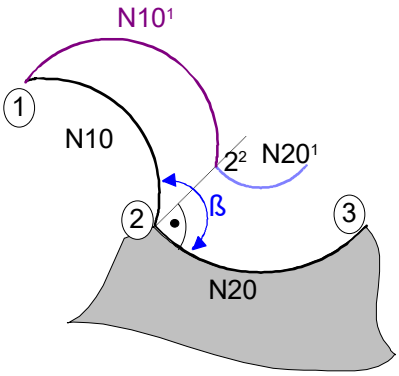
Kombinationstyp 1	Kombinationstyp 2
	
Kombinationstyp 3	**Kombinationstyp 4**
	

Konturübergangsbereich 180° < ß ≤ 270°

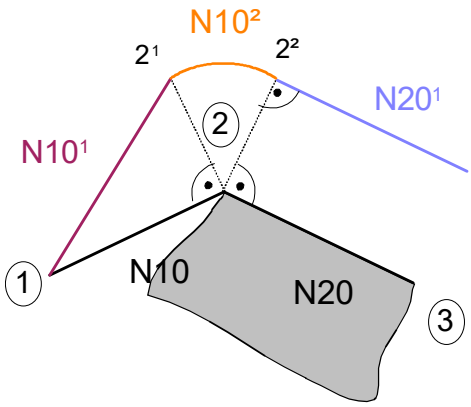
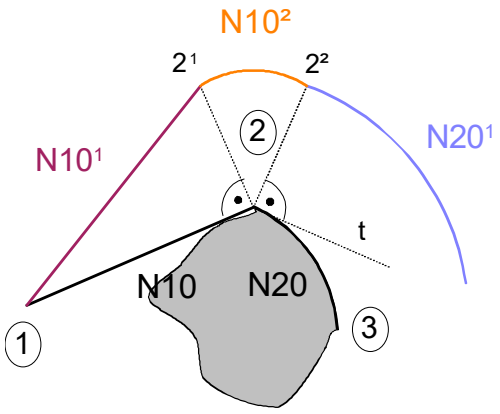
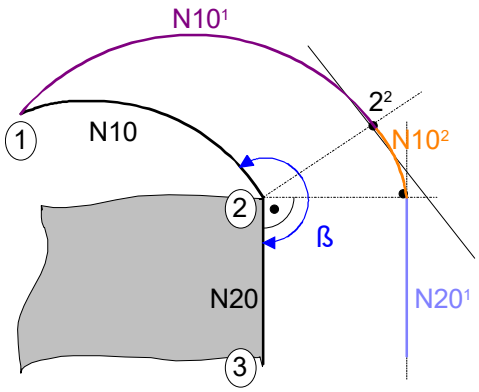
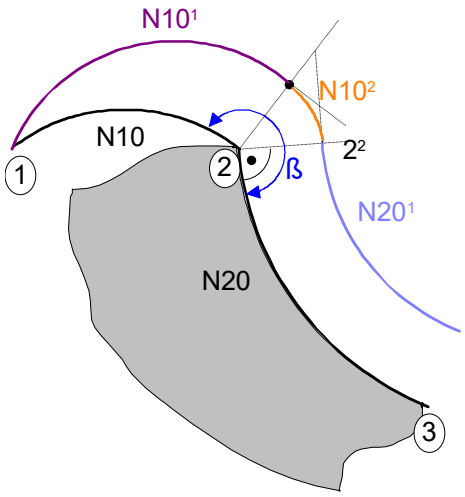
Kombinationstyp 1	Kombinationstyp 2
	
Kombinationstyp 3	**Kombinationstyp 4**
	

Konturübergangsbereich 270° < ß ≤ 360°

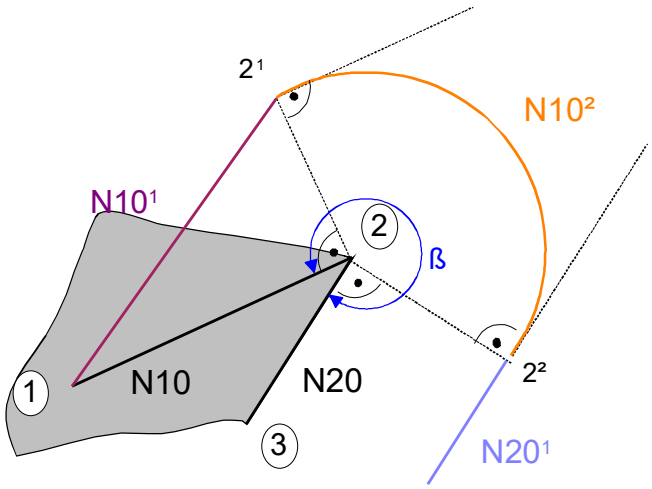
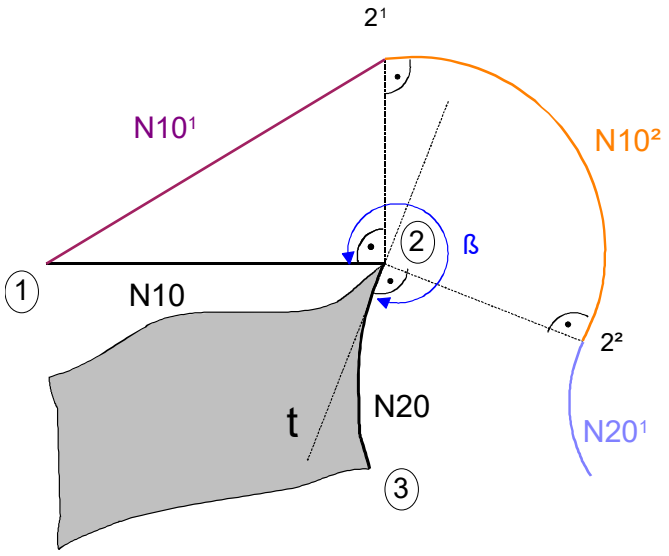
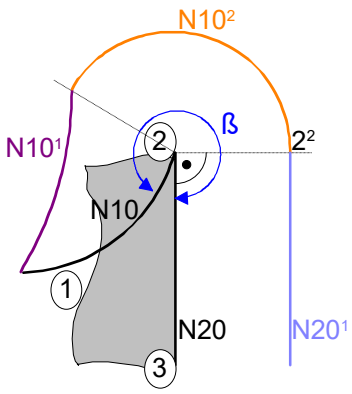
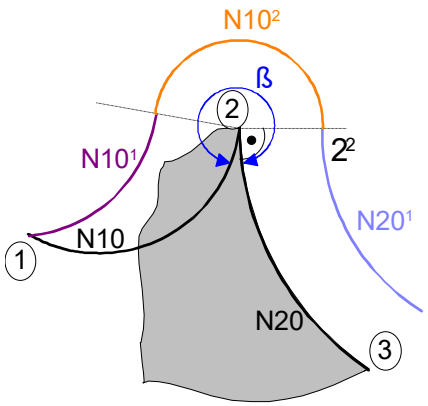
13.2.1.3 Indirekte Anwahl (G139/G41/G42) mit G26

Kombinationstyp 1	Kombinationstyp 2
	
Kombinationstyp 3	**Kombinationstyp 4**
	

Konturübergangsbereich 0° < ß ≤ 180°

Kombinationstyp 1	Kombinationstyp 2
	
Kombinationstyp 3	**Kombinationstyp 4**
	

Konturübergangsbereich 180° < ß ≤ 270°

Kombinationstyp 1	Kombinationstyp 2
	
Kombinationstyp 3	**Kombinationstyp 4**
	

Konturübergangsbereich 270° < ß ≤ 360°

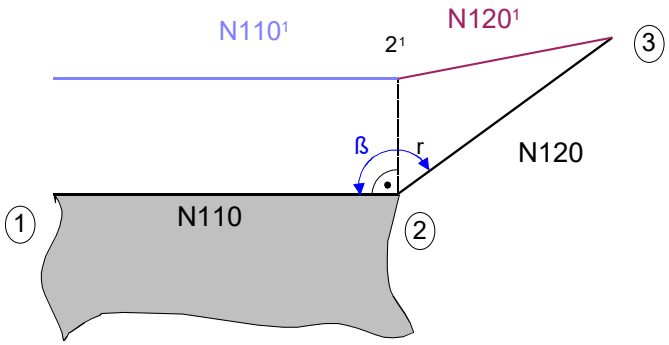
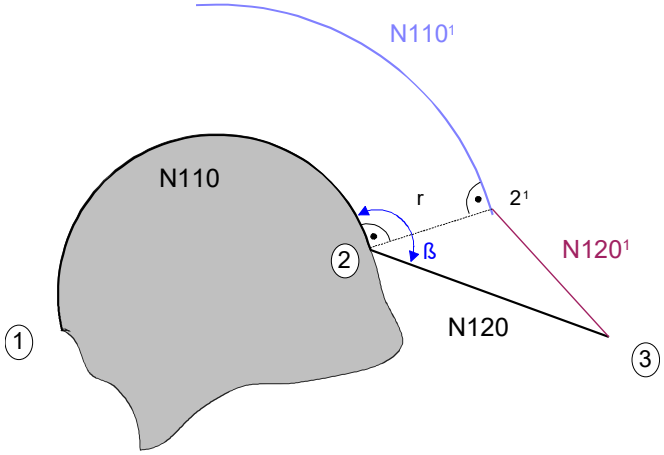
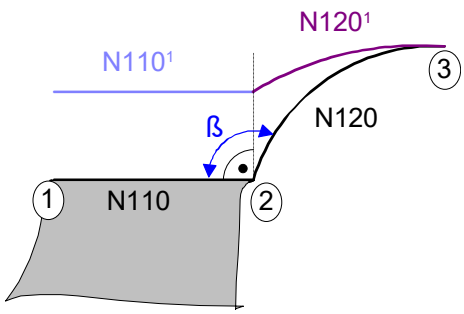
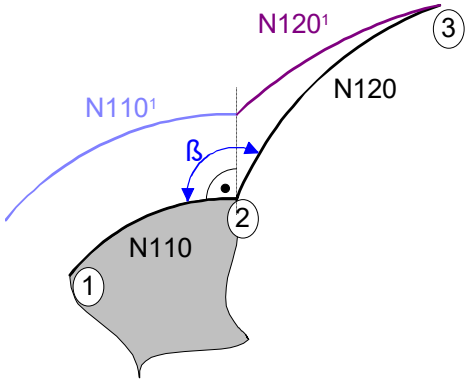
13.2.2 Direkte/ indirekte Abwahl der WRK

Bei direkt oder indirekt angewählter WRK wird mit dem nachfolgenden Verfahrsatz nach programmiertem G40 ein Abfahrsatz in der angewählten Korrektur-Ebene generiert.

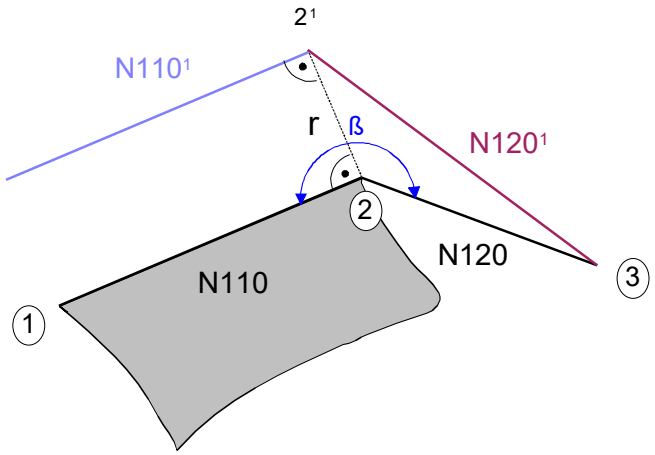
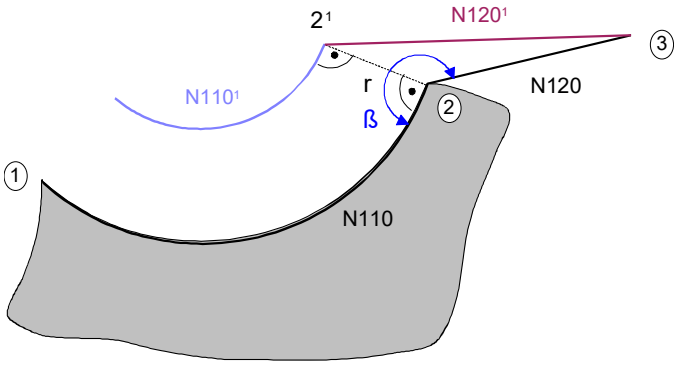
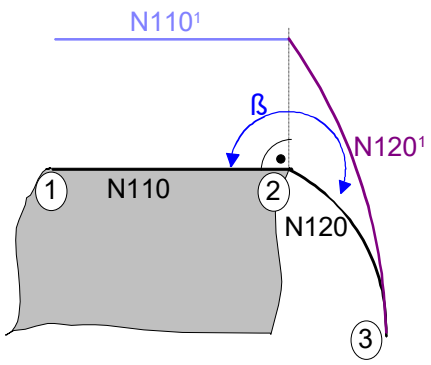
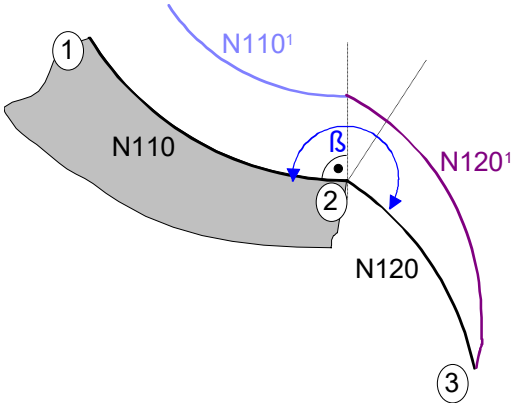
Bei der WRK-Abwahl kann mit G138 neben der Standardabfahrstrategie (indirekte Abwahl der WRK) eine alternative Abfahrstrategie (direkte Abwahl der WRK) angewählt werden.

Die Bilder der folgenden Unterkapitel zeigen alle möglichen Abfahrsätze bei allen Konturübergängen. Dargestellt sind jeweils zwei NC-Sätze N10 und N20 für die drei relevanten Konturübergangswinkel.

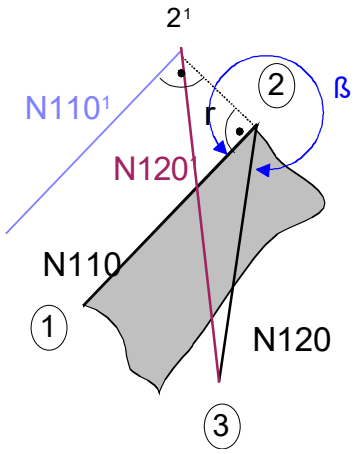
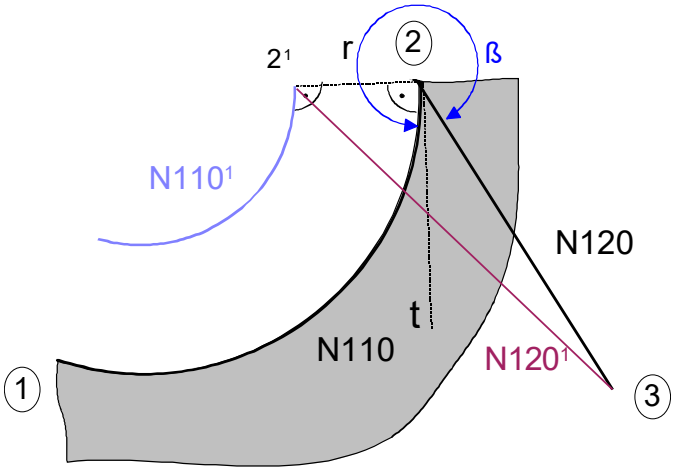
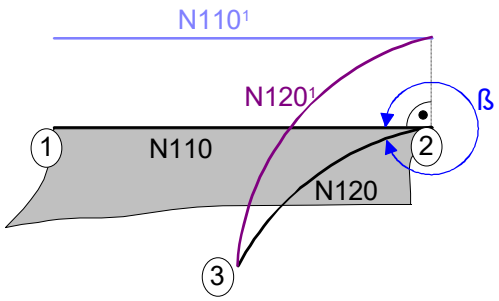
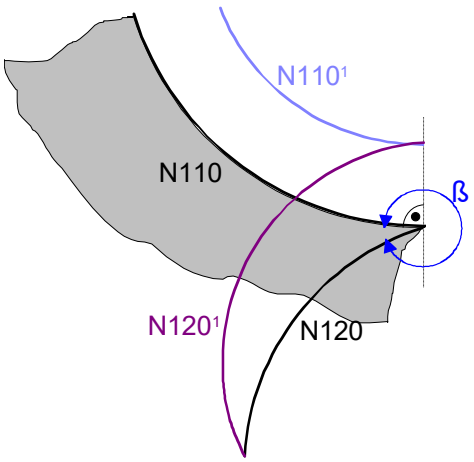
13.2.2.1 Direkte Abwahl (G138/G40)

Kombinationstyp 1	Kombinationstyp 2
	
Kombinationstyp 3	**Kombinationstyp 4**
	

Konturübergangsbereich 0° < ß ≤ 180°

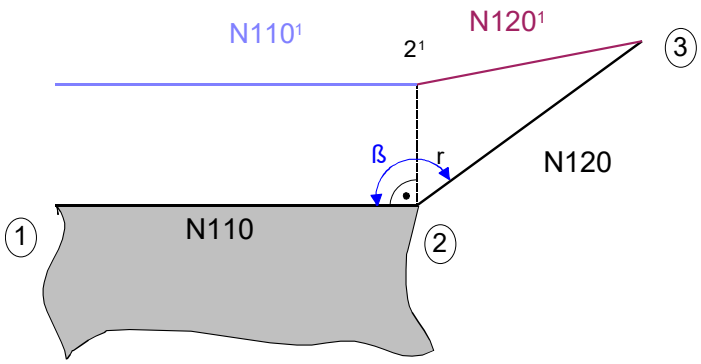
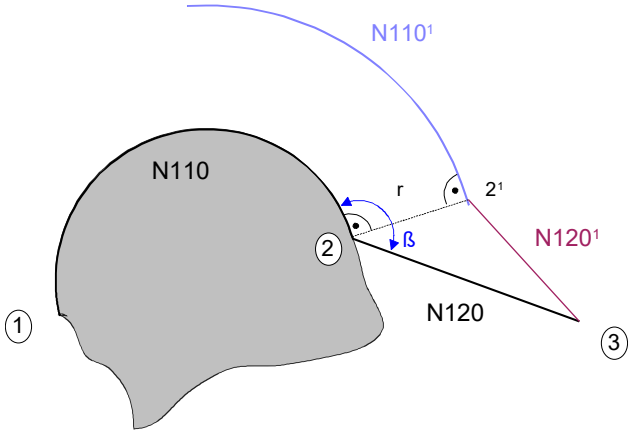
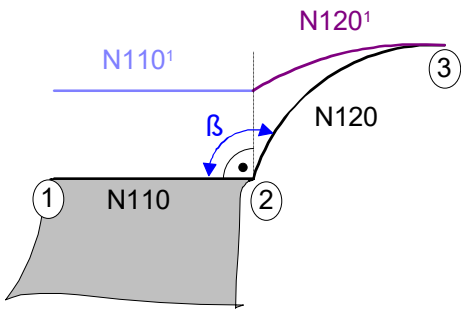
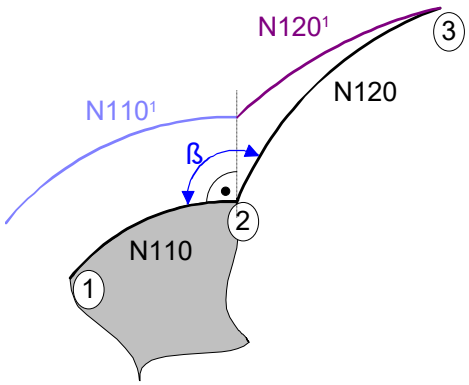
Kombinationstyp 1	Kombinationstyp 2
	
Kombinationstyp 3	**Kombinationstyp 4**
	

Konturübergangsbereich 180° < ß ≤ 270°

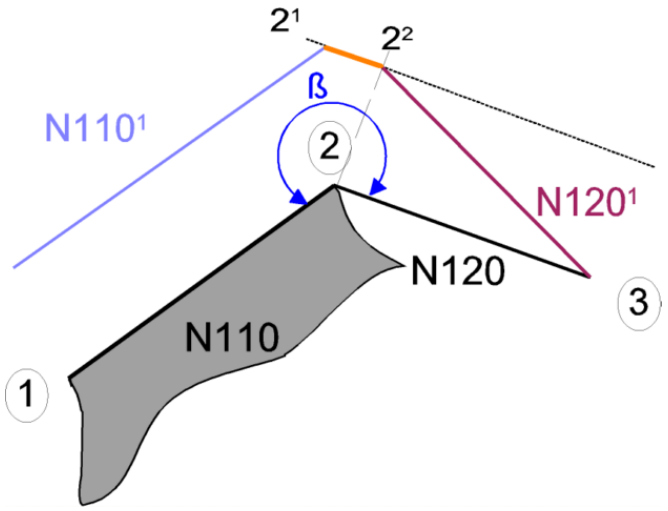
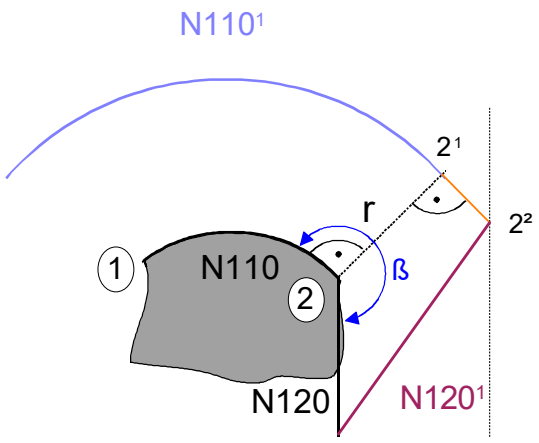
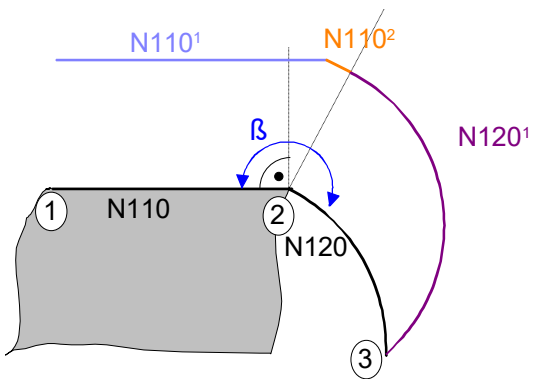
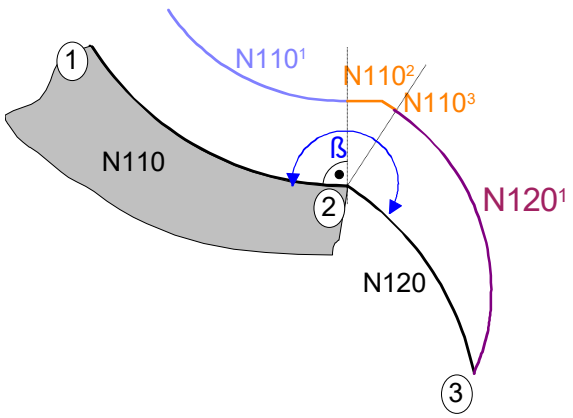
Kombinationstyp 1	Kombinationstyp 2
	
Kombinationstyp 3	**Kombinationstyp 4**
	

Konturübergangsbereich 270° < ß ≤ 360°

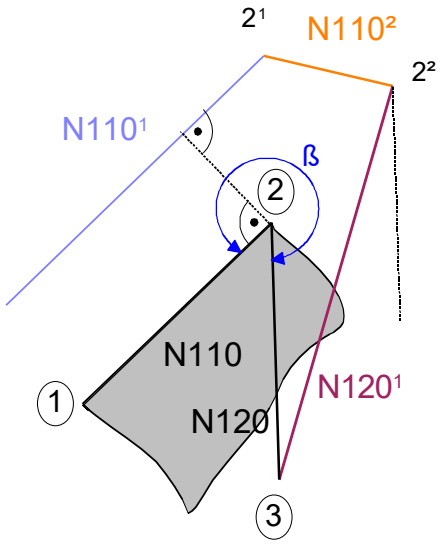
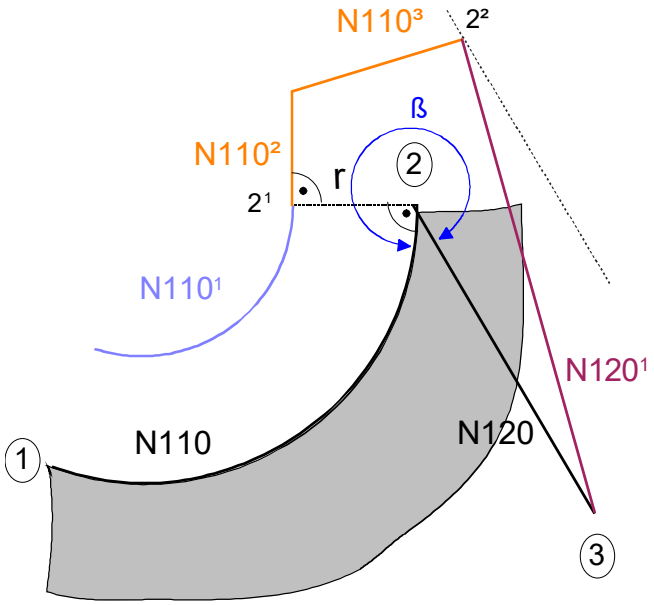
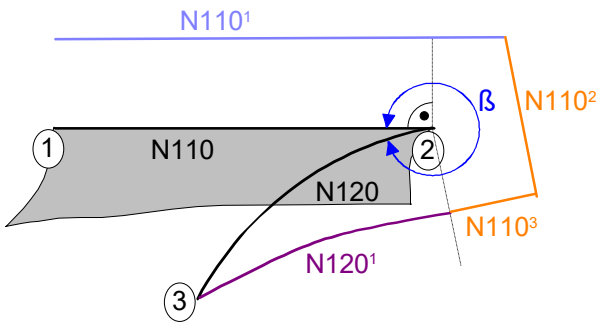
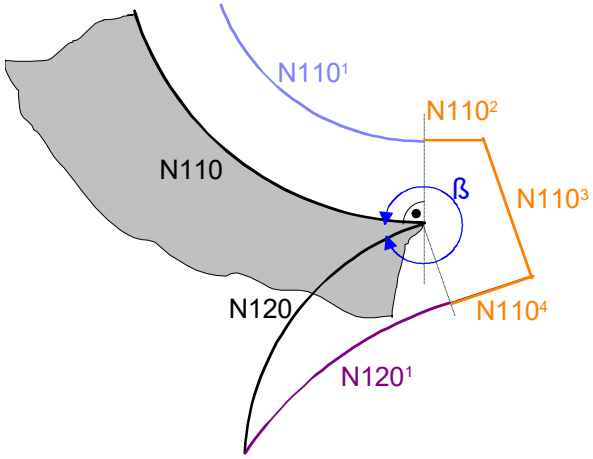
13.2.2.2 Indirekte Abwahl (G139/G40) mit G25

Kombinationstyp 1	Kombinationstyp 2
	
Kombinationstyp 3	**Kombinationstyp 4**
	

Konturübergangsbereich 0° < ß ≤ 180°

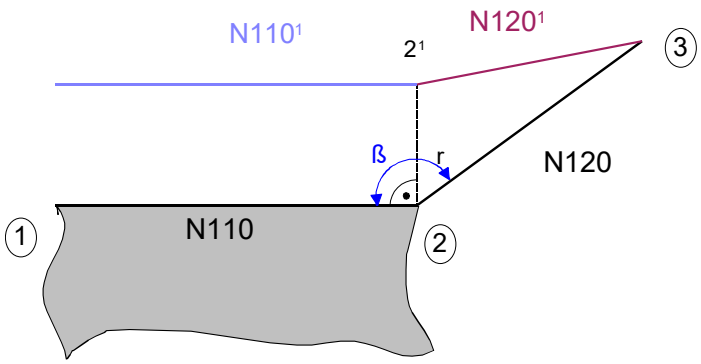
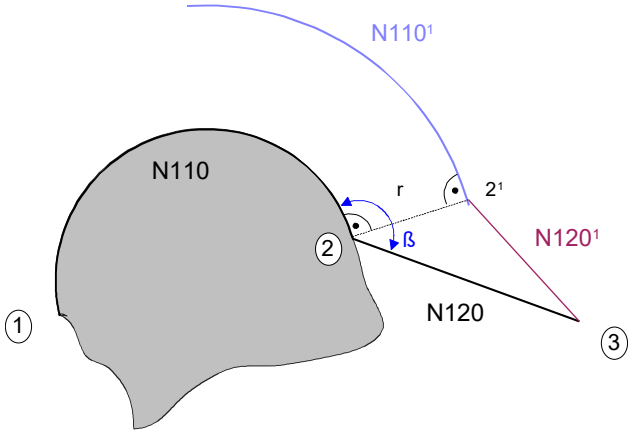
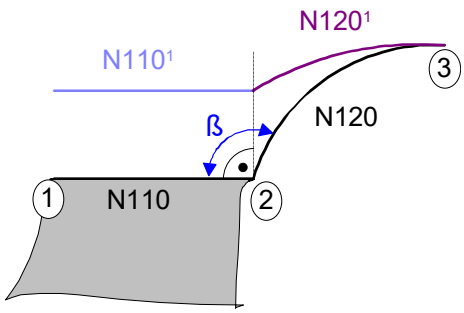
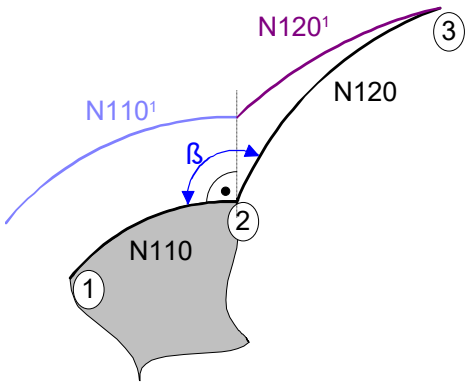
Kombinationstyp 1	Kombinationstyp 2
	
Kombinationstyp 3	**Kombinationstyp 4**
	

Konturübergangsbereich 180° < ß ≤ 270°

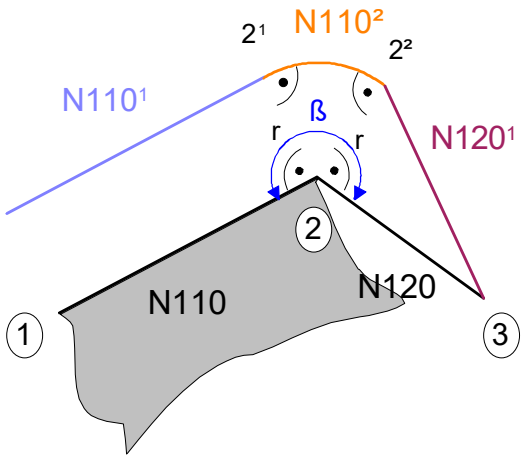
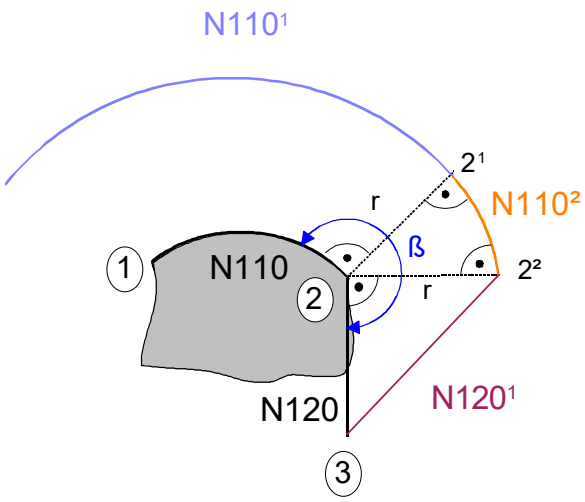
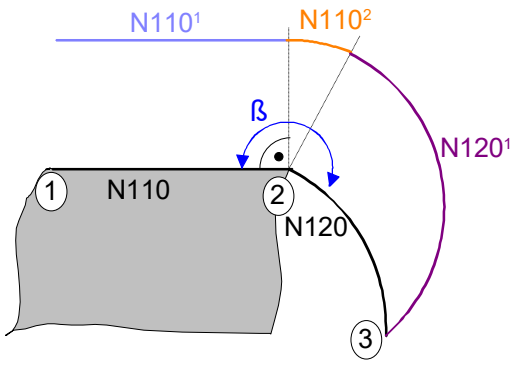
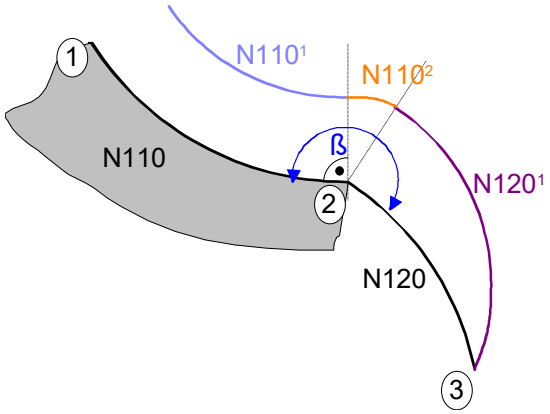
Kombinationstyp 1	Kombinationstyp 2
	
Kombinationstyp 3	**Kombinationstyp 4**
	

Konturübergangsbereich 270° < ß ≤ 360°

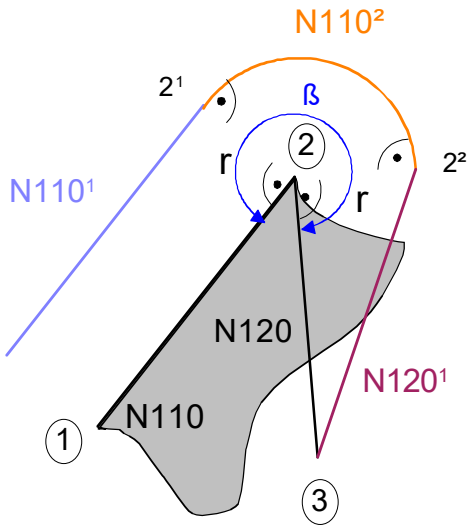
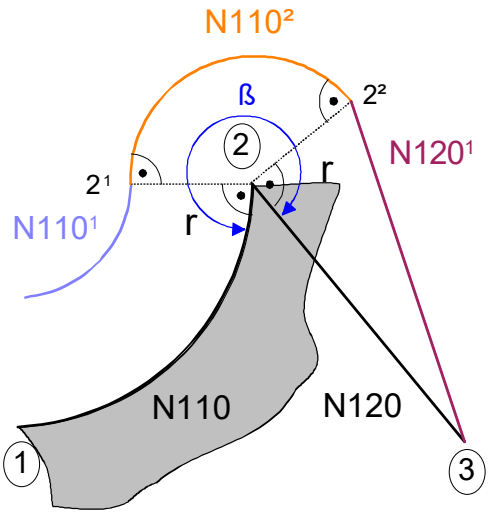
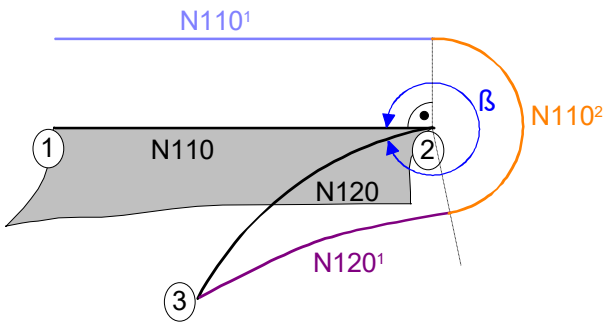
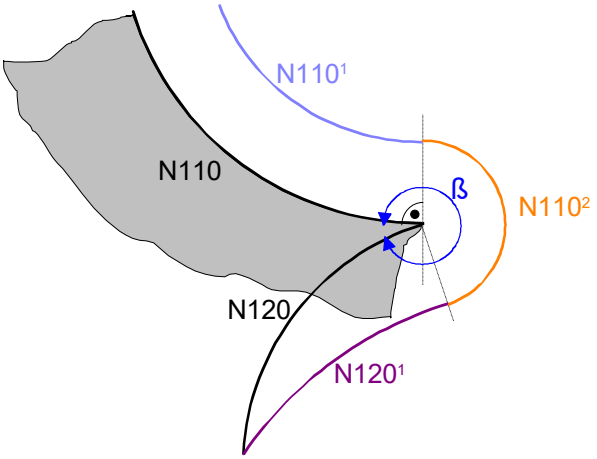
13.2.2.3 Indirekte Abwahl (G139/G40) mit G26

Kombinationstyp 1	Kombinationstyp 2
	
Kombinationstyp 3	**Kombinationstyp 4**
	

Konturübergangsbereich 0° < ß ≤ 180°

Kombinationstyp 1	Kombinationstyp 2
	
Kombinationstyp 3	**Kombinationstyp 4**
	

Konturübergangsbereich 180° < ß ≤ 270°

Kombinationstyp 1	Kombinationstyp 2
	
Kombinationstyp 3	**Kombinationstyp 4**
	

Konturübergangsbereich 270° < ß ≤ 360°

13.2.3 Lotrechte An-/Abwahl der WRK (G237)

Bei Verwendung des lotrechten An- und Abwahlverfahrens gibt es keine Einschränkungen bezüglich der Satzfolge, wie sie bei der direkten oder indirekten An-und Abwahl gelten.

Es ist ebenfalls möglich, die WRK für einzelne Sätze zu aktivieren.

Bei der lotrechten Anwahl wird ein Satz eingefügt, der orthogonal zur programmierten Bahn verläuft. Dieser Satz wird vor dem ersten Bewegungssatz ausgegeben und stellt den Abstand (Werkzeugradius) der korrigierten Bahn zur programmierten Bahn her.

Bei Abwahl der WRK mit lotrechtem Verfahren wird nach dem Abwahlsatz ein Satz eingefügt, der orthogonal zur programmierten Bahn verläuft. Dieser Satz macht den hergestellten Abstand zur programmierten Bahn wieder rückgängig.

Programmierbeispiel

Lotrechte An- und Abwahl (G237) der WRK

```
%bsp01.nc
N10 G00 X0 Y0 Z0 G17
N20 X10Y10
N30 V.G.WZ_AKT.R=5      (Radius des Werkzeuges)
N40 G237                (lotrechte Anwahl aktiviert)

; Korrigierte Bahn
N50 G42                 (Anwahl der WRK rechts)
N60 G03 X60Y60J50F1000
N70 G01 Y100
N80 G03 X10Y150J50
N90 G40                 (Abwahl der WRK)

; Darstellung der originalen Kontur
N100 G00 X0Y0
N110 X10Y10
N120 G03 X60Y60J50F1000
N130 G01 Y100
N140 G03 X10Y150J50
N150 G00 X0Y0
N999 M30
```

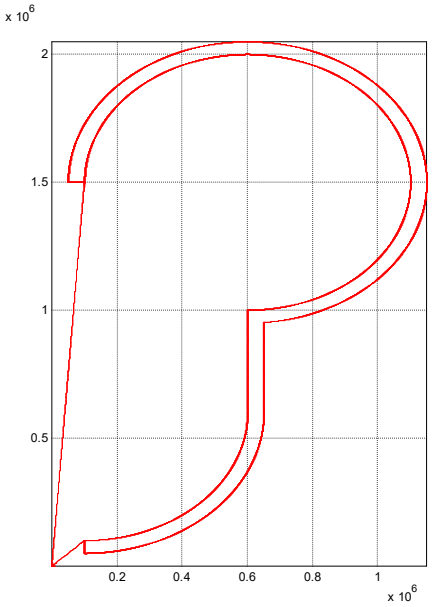


Abb. 130: Konturbeispiel bei G237

13.2.3.1 Technologiefunktionen

Die Platzierung von Technologiefunktionen im NC-Programm ist bei lotrechter An- und Abwahl von besonderer Bedeutung, da diese den Ausgabezeitpunkt bestimmt.

Bei lotrechter Anwahl ist die Anordnung von Anwahlsatz, Technologiefunktion und des ersten Bewegungssatzes ausschlaggebend für den Ausgabezeitpunkt.

Bei lotrechter Abwahl ist die Anordnung von Technologiefunktion und des letzten zu korrigierenden Bewegungssatzes ausschlaggebend für den Ausgabezeitpunkt.

Die nachfolgenden Beispielprogramme mit den zugehörigen Bildern verdeutlichen dies.

Programmierbeispiel

Technologiefunktion 1

```
%bsp02.nc

N10 G00 X0 Y0 Z0 G17
N20 F9000
N30 V.G.WZ_AKT.R=5      (Radius des Werkzeuges)
N40 G237                (lotrechte Anwahl aktiviert)
N50 G91                 (Programmierung relativ)
N60 G01 X30 Y10

; Korrigierte Bahn
N50 G41 M7 X20 Y70      (Anwahl der WRK links)
N55 M8
N60 G03 X60 I30
N70 G01 X30
N80 X25 Y-20
N85 M9
N90 G40                 (Abwahl der WRK)
N100 G90 X200 Y0        (Programmierung absolut)
N110 X0

; Darstellung der originalen Kontur
N200 G91              (Programmierung relativ)
N210 G01 X30 Y10
N220 X20 Y70
N230 G03 X60 I30
N240 G01 X30
N250 X25 Y-20
N260 G90 X200 Y0      (Programmierung absolut)
N270 G00 X0
N999 M30
```

Die Ausgabe der Technologiefunktion M9 in nachfolgendem Bild erfolgt unmittelbar vor der Ausgabe des erzeugten lotrechten Abwahlsatzes. Die Technologiefunktion muss zwischen dem letzten zu korrigierenden Satz und der lotrechten Abwahl der WRK platziert werden.

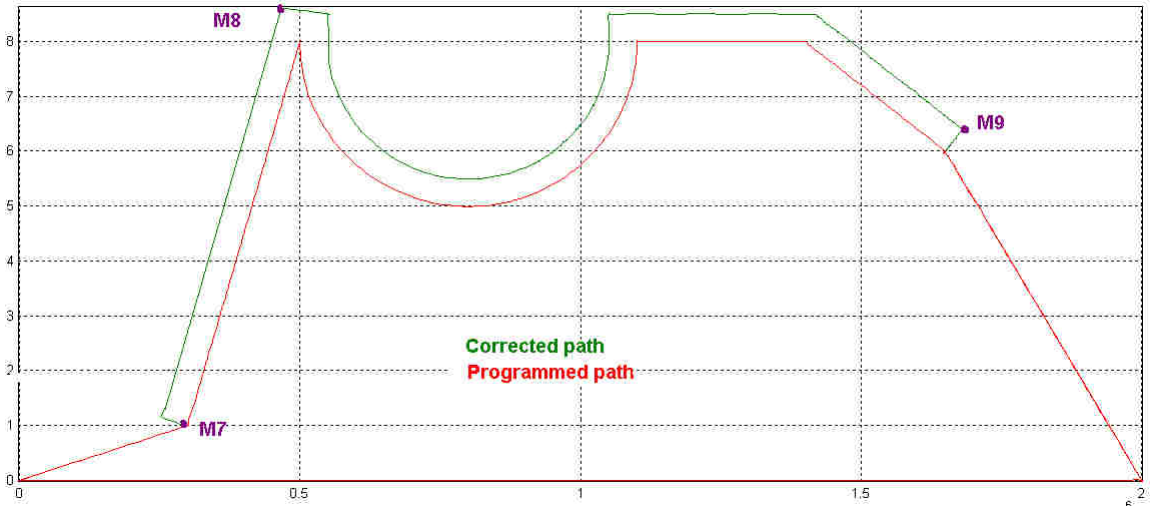


Abb. 131: Konturbeispiel bei Technologiefunktion 1

Programmierbeispiel

Technologiefunktion 2

```
%bsp03.nc

N10 G00 X0 Y0 Z0 G17
N20 F9000
N30 V.G.WZ_AKT.R=5      (Radius des Werkzeuges)
N40 G237                (lotrechte Anwahl aktiviert)
N50 G91                 (Programmierung relativ)
N60 G01 X30 Y10
; Korrigierte Bahn
N50 G41 M7              (Anwahl der WRK links)
N55 M8 X20 Y70
N60 M9 G03 X60 I30
N70 G01 X30
N80 X25 Y-20
N90 G40                 (Abwahl der WRK)
N100 G90 X200 Y0        (Programmierung absolut)
N110 X0
; Darstellung der originalen Kontur
N200 G91                (Programmierung relativ)
N210 G01 X30 Y10
N220 X20 Y70
N230 G03 X60 I30
N240 G01 X30
N250 X25 Y-20
N260 G90 X200 Y0        (Programmierung absolut)
N270 G00 X0
N999 M30
```

Die Technologiefunktion M8 wird nach dem erzeugten Anwahlsatz ausgeführt. Für diese Platzierung ist es zwingend erforderlich, dass sie nicht im selben Satz wie die Anwahl der WRK steht, und dass kein Bewegungssatz zwischen der Anwahl und der Technologiefunktion steht.

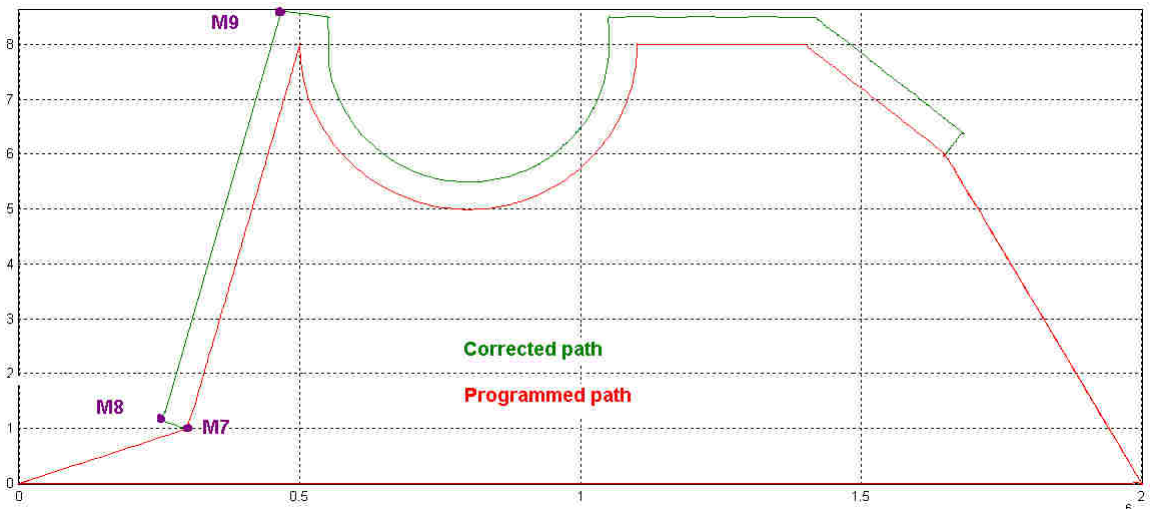


Abb. 132: Konturbeispiel bei Technologiefunktion 2

Programmierbeispiel

Technologiefunktion 3

```
%bsp04.nc

N10 G00 X0 Y0 Z0 G17
N20 F9000
N30 V.G.WZ_AKT.R=5      (Radius des Werkzeuges)
N40 G237                (lotrechte Anwahl aktiviert)
N50 G91                 (Programmierung relativ)
N60 G01 X30 Y10
; Korrigierte Bahn
N50 G41                 (Anwahl der WRK links)
N51 M7
N52 M8
N53 M8
N55 X20 Y70
N60 G03 X60 I30
N70 G01 X30
N80 X25 Y-20
N90 G40                 (Abwahl der WRK)
N100 G90 X200 Y0        (Programmierung absolut)
N110 X0
; Darstellung der originalen Kontur
N200 G91                (Programmierung relativ)
N210 G01 X30 Y10
N220 X20 Y70
N230 G03 X60 I30
N240 G01 X30
N250 X25 Y-20
N260 G90 X200 Y0        (Programmierung absolut)
N270 G00 X0
N999 M30
```

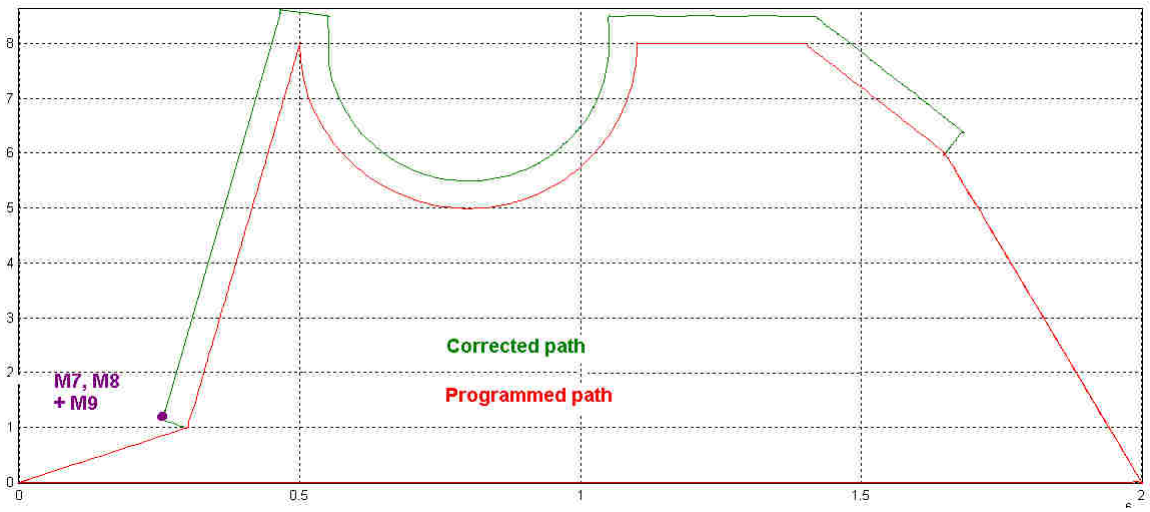


Abb. 133: Konturbeispiel bei Technologiefunktion 3

13.2.3.2 Technologiefunktionalität bei Einzelsatz

Programmierbeispiel

Technologiefunktionalität bei Einzelsatz

```
%bsp05.nc

N10 G00 X0 Y0 Z0 G17
N20 F9000
N30 V.G.WZ_AKT.R=5      (Radius des Werkzeuges)
N40 G237                (lotrechte Anwahl aktiviert)
N60 G01 X10 Y10
; Korrigierte Bahn
N50 G41 M7              (Anwahl der WRK links)
N55 M8 X70 Y30
N60 M7
N90 G40                 (Abwahl der WRK)
N100 G90 X100 Y0
N110 X0
; Darstellung der originalen Kontur
N210 G01 X10 Y10
N220 X70 Y30
N240 X100 Y0
N270 G00 X0
N999 M30
```

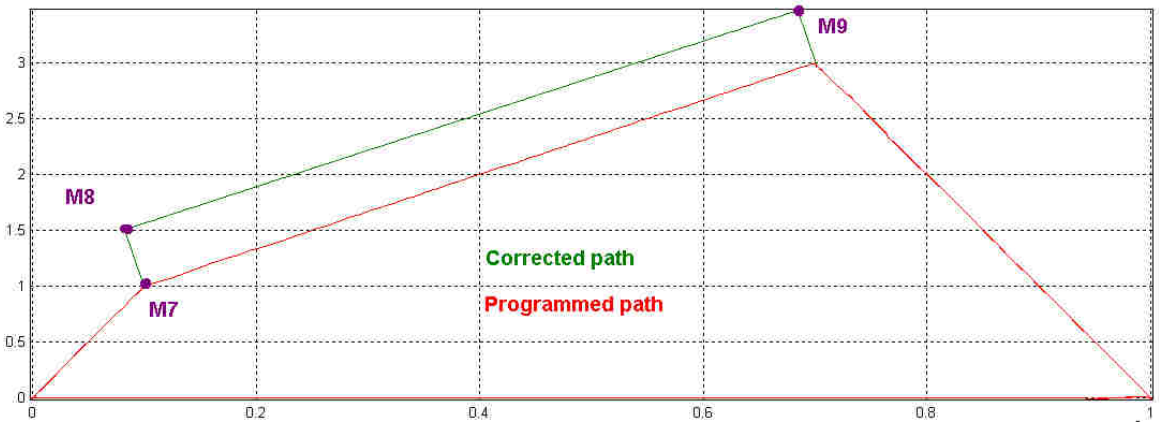


Abb. 134: Konturbeispiel bei Technologiefunktionalität bei Einzelsatz

13.2.4 Inneneckanwahl der WRK (G238)

Inneneckanwahl bedeutet, dass die WRK an einem Inneneck einer geschlossenen Kontur angewählt werden soll.

Im Zusammenhang mit der Inneneckwahl werden folgende Bezeichnungen verwendet:

- **Anfahrsatz**: linearer Bewegungssatz, der vom Punkt der Anwahl ausgeht
- **1.Bewegungssatz**: Bewegungssatz, der vom Bewegungsverlauf an den Anfahrsatz anknüpft, es ist auch der 2.Bewegungssatz nach der Anwahl der WRK mit Bewegungsinformationen
- **letzter Bewegungssatz:** letzter Bewegungssatz vor der Abwahl der WRK mit Bewegungsinformationen

Programmierbeispiel

Inneneckanwahl (G238) der WRK

Bei einer sternförmigen Kontur soll an einem spitzen Inneneck die WRK angewählt werden.

```
%musterstern.nc
N1 G74 X1Y2Z3
N2 G17 G00 X0Y0Z0 G90
N4 F10000

(Kontur Darstellung)
N100 G01 X0 Y100
N110 X-20 Y20
N120 X-100 Y0
N130 X-20 Y-20
N140 X0 Y-100
N150 X20 Y-20
N160 X100 Y0
N170 X20 Y20
N180 G01 X0 Y100

N200 G00 X0Y0
( Bahndarstellung )
N210 G238          ( Inneneckanwahl aktiviert )
N220 V.G.WZR = 10  ( Werkzeugradius festlegen )
N230 G41      ( Anwahl der WRK links der Kontur)

N240 G01 X0 Y100   ( Anfahrsatz )

N310 X-20 Y20
N320 X-100 Y0
N330 X-20 Y-20
N340 X0 Y-100
N350 X20 Y-20
N360 X100 Y0
N370 X20 Y20
N380 G01 X0 Y100

N390 G40          ( Abwahl der WRK )
N400 G00 X0Y0
N999 M30
```

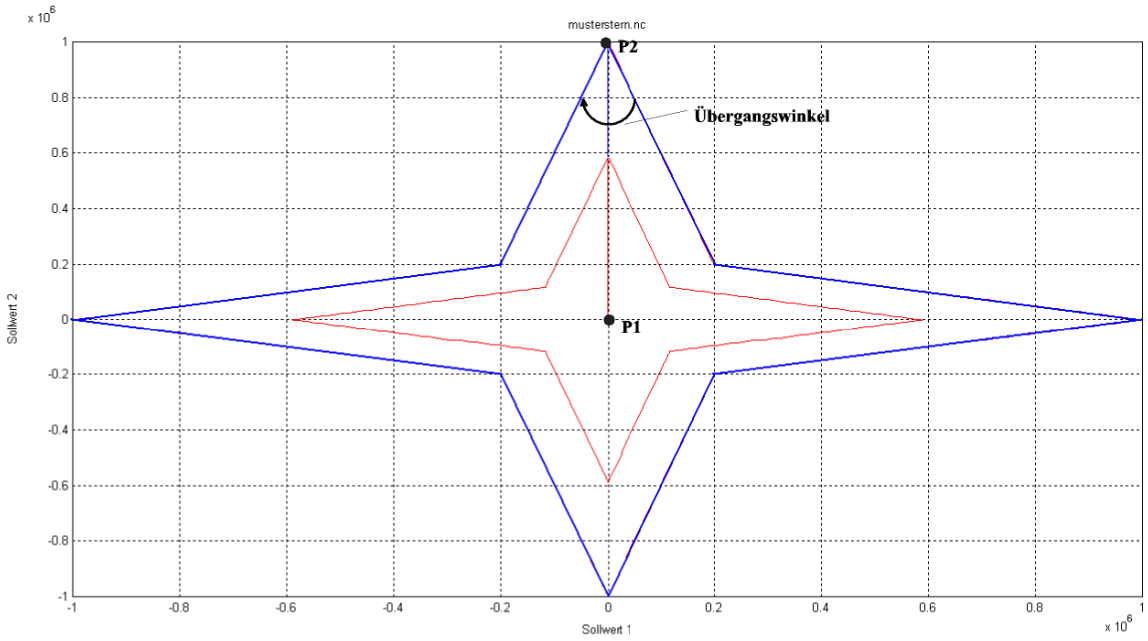


Abb. 135: Konturbeispiel bei Inneneckanwahl (G238)

Bemerkung:

Der Anwahlbefehl G41 erfolgt am Punkt P1, der Punkt P2 ist der Endpunkt des ersten Linearsatz nach G41 und muss gleichzeitig der Zielpunkt des letzten Bewegungssatzes vor G40 sein.

Die äußere (bzw. blaue) Linie zeigt die programmierte Kontur, die innere (bzw. rote) Linie zeigt den Verlauf des Werkzeuges.

Geschlossene Kontur bedeutet, dass der Endpunkt des letzten Bewegungssatzes vor der Abwahl der WRK und der Endpunkt des Linearsatzes nach der Anwahl identisch sind.

Für den Satzübergang zwischen dem letzten und dem ersten Bewegungssatz der Kontur gibt es weder beim Übergangswinkel noch bei den Satzkombinationen Einschränkungen.

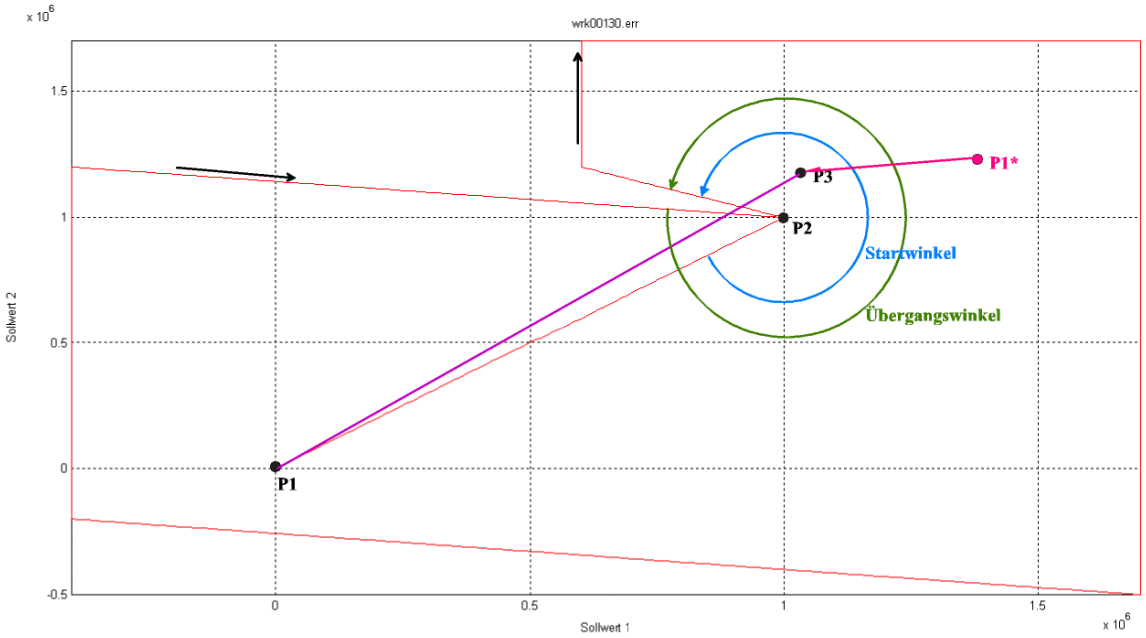
Hinweis

Nach dem programmierten G40 zur Abwahl der WRK erfolgt beim nachfolgenden Bewegungssatz keine Überwachung auf Konturverletzung.

13.2.4.1 Einschränkungen der Inneneckanwahl

- Die programmierte Kontur muss geschlossen sein.
- Der letzte Bewegungssatz darf kein Vollkreis sein.
- Der erste Bewegungssatz muss ein Linearsatz sein, der erste Bewegungssatz nach der Abwahl der WRK ebenso.
- Der Punkt P1 muss zu allen Elementen der Kontur mindestens den Abstand des Werkzeugradius haben.
- Der erste Bewegungssatz darf kein Konturelement kreuzen, mit Ausnahme des ersten und des letzten Bewegungssatzes.
- Ein Wechsel der Anwahlseite ist im angewählten Zustand nicht zulässig.
- Änderungen am Werkzeugradius sind im angewählten Zustand nicht zulässig.
- Die Anwahlseite der WRK muss von der Position der Anwahl so gewählt werden, dass das Werkzeug die programmierte Kontur nicht kreuzt.
- Der eingeschlossene Winkel des ersten und des zweiten Bewegungssatzes nach der Anwahl darf 180° nicht übersteigen.

Ursache für die Winkeleinschränkung:



Bei obiger Grafik soll die WRK rechts der Kontur angewählt werden, ab einem Übergangswinkel zwischen dem letzten und dem ersten Satz der Kontur von über 180° wird der erste Punkt der äquidistanten Bahn mit der direkten Anwahlmethode bestimmt. Dies ist P3. Die Grafik verdeutlicht, dass die direkte Verbindung P1 zu P3 die in rot gezeichnete Kontur verletzt.

Der in der Grafik mit Startwinkel bezeichnete Winkel darf maximal 180° betragen um die Kontur nicht zu beschädigen. Der alternativ eingezeichnete Anwahlpunkt P1* verhindert diese Konturbeschädigung.

Achtung

Bei Verletzung einer der Einschränkungen wird ein entsprechender Fehler ausgegeben.

13.2.5 Direkte An-/Abwahl der WRK ohne Satz (G239)

Die Anfahrbewegung bei G239 erfolgt direkt wie mit G138.

Mit Abwahl der WRK (G40) wird auf der zuletzt äquidistanten Position aufgesetzt. Ein nachfolgender Bewegungssatz zum Abbau des Werkzeugradius ist nicht erforderlich.

Achtung

Wird unmittelbar nach G40 ein Kreisbogen programmiert, so führt dies zu einem Fehler, da die reale Abwahlposition und der programmierte Startpunkt des Kreises dann nicht mehr identisch sind!

In den nachfolgenden Beispielen wird die programmierte Bahn in schwarz und die äquidistante Bahn in blau dargestellt.

Programmierbeispiel

Beispiel 1

Nachfolgend wird nach der WRK-Abwahl eine Position in beiden Hauptachsen programmiert und auch so ausgegeben.

```
%G239_Demo1.nc
N010 G0 X0 Y0 Z0 F1000
N020 G239               (Mode der WRK )
N030 V.G.WZ_AKT.R = 10
N040 G41 G01 X30 Y20    (Anwahl der WRK )
N050 G01 X60
N060 G01 X90
N070 G01 X120
N080 G40                (Abwahl der WRK ohne Satz)

N100 G00 X150 Y0
N110 M30
```

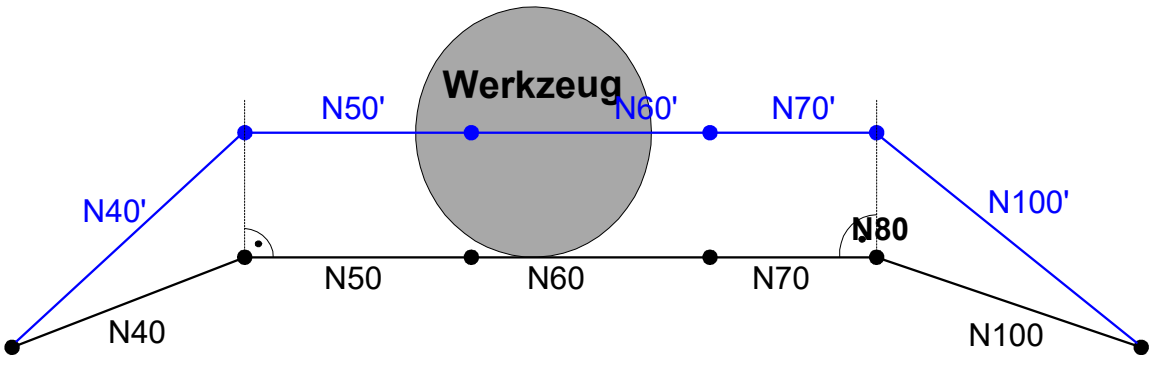


Abb. 136: Bewegungsverlauf zu G239_eine Position in beiden Hauptachsen

Programmierbeispiel

Beispiel 2:

Hier wird nach Abwahl der WRK nur eine der Hauptachsen programmiert. Die Position der 2. Hauptachse bleibt gleich.

```
%G239_Demo2.nc
N010 G0 X0 Y0 Z0 F1000
N020 G239                (Mode der WRK )
N030 V.G.WZ_AKT.R = 10
N040 G41 G01 X30 Y20     (Anwahl der WRK )
N050 G01 X60
N060 G01 X90
N070 G01 X120
N080 G40                  (Abwahl der WRK ohne Satz)

N090 G00 X150
N100 G00 X200

N110 M30
```

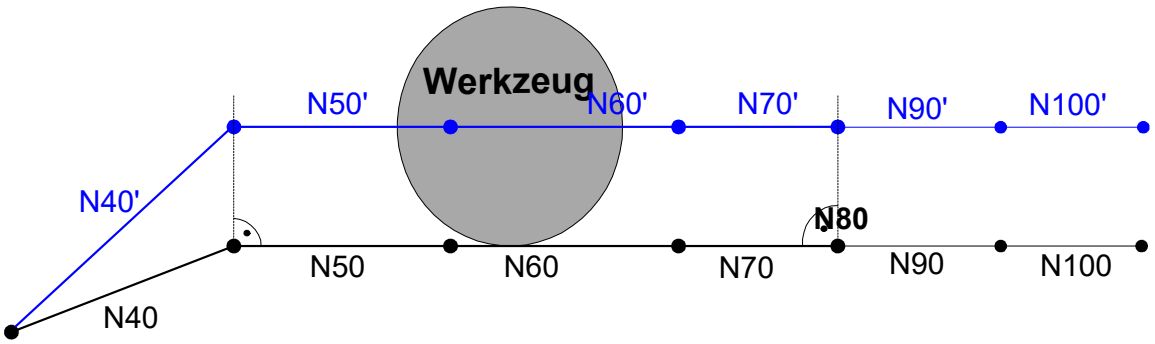


Abb. 137: Bewegungsverlauf zu G239_nur eine Hauptachse programmiert

Programmierbeispiel

Beispiel 3:

Bei erneuter Anwahl der WRK unmittelbar nach G40 wird nur eine der beiden Hauptachsen programmiert.

```
%G239_Demo3.nc
N010 G0 X0 Y0 Z0 F1000
N020 G239                (Mode der WRK )
N030 V.G.WZ_AKT.R = 10
N040 G41 G01 X30 Y20     (Anwahl der WRK )
N050 G01 X60
N060 G01 X90
N070 G01 X120
N080 G40                 (Abwahl der WRK ohne Satz)
( Keine erneute Programmierung von Y )
N090 G41 G01 X150        (Erneute Anwahl der WRK )
N100 G00 X200

N110 M30
```

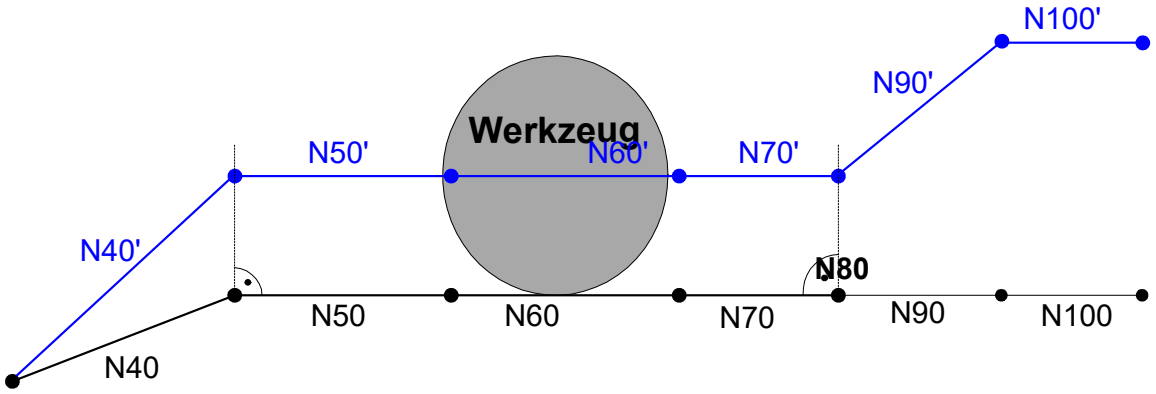


Abb. 138: Bewegungsverlauf zu G239_nach G40 nur eine Hauptachse programmiert

Programmierbeispiel

Beispiel 4:

Hier wird erst im 2. Bewegungssatz nach erneuter Wiederanwahl der WRK die 2. Hauptachse programmiert.

```
%G239_Demo4.nc
N010 G0 X0 Y0 Z0 F1000
N020 G239                (Mode der WRK )
N030 V.G.WZ_AKT.R = 10
N040 G41 G01 X30 Y20     (Anwahl der WRK )
N050 G01 X60
N060 G01 X90
N070 G01 X120
N080 G40                 (Abwahl der WRK ohne Satz)

N090 G41 G01 X150        (Erneute Anwahl der WRK )
N100 G00 X200 Y20

N110 M30
```

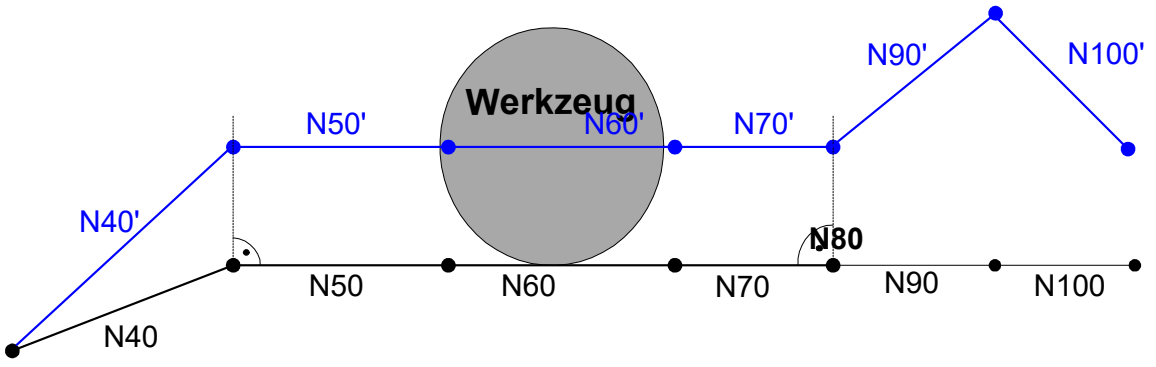


Abb. 139: Bewegungsverlauf zu G239_Programmierung 2. Hauptachse im 2. Bewegungssatz

Programmierbeispiel

Beispiel 5:

In diesem Beispiel wird bei erneuter Wiederanwahl der WRK nach G40 die Position der 2.Hauptachse wie bei der ersten Anwahl (G41) programmiert.

```
%G239_Demo5.nc
N010 G0 X0 Y0 Z0 F1000
N020 G239                (Mode der WRK )
N030 V.G.WZ_AKT.R = 10
N040 G41 G01 X30 Y20     (Anwahl der WRK )
N050 G01 X60
N060 G01 X90
N070 G01 X120
N080 G40                 (Abwahl der WRK ohne Satz)

N090 G41 G01 X150 Y20    (Erneute Anwahl der WRK )
N100 G00 X200

N110 G40
N120 M30
```

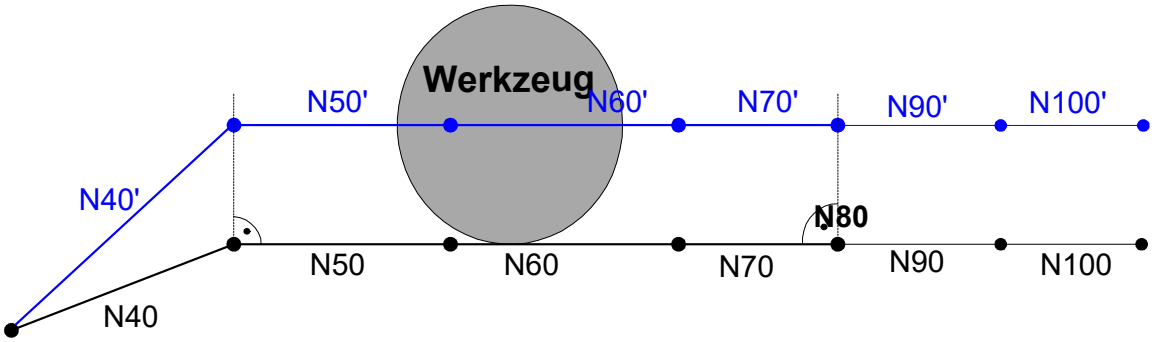


Abb. 140: Bewegungsverlauf zu G239_Programmierung der 2. Hauptachse wie bei 1. Anwahl

Programmierbeispiel

Beispiel 6:

Der nach der WRK-Abwahl programmierte Kreisbogen kann nicht gefahren werden, weil der angegebene Mittelpunkt (M) nicht mehr zum (neuen) Start (S)- und (programmierten) Endpunkt (E) passt.

```
%G239_Demo_Kreis.nc
N005 G162
N010 G0 X0 Y0 Z0 F1000
N020 G239                  (Mode der WRK =
N030 V.G.WZ_AKT.R = 30
N040 G41 G01 X30 Y20       (Anwahl der WRK )
N050 G01 X60
N060 G01 X90
N070 G40                   (Abwahl der WRK ohne Satz)
N080 G02 X140 Y-30 J-50    (Fehler, Kreismittelpunkt nicht korrekt)
N100 G00 X200
N110 G40
N120 M30
```

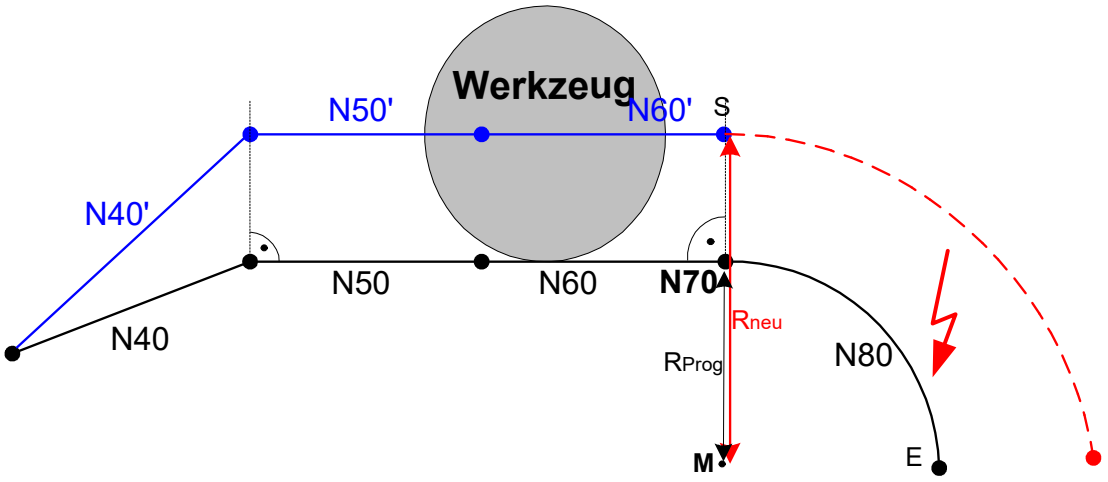


Abb. 141: Bewegungsverlauf zu G239_Mittelpunkt passt nicht zu Start- und Endpunkt

Es wird eine Meldung mit der ID 20035 ausgegeben.

13.2.6 Direkte An-/Abwahl der WRK auf die Bahn (G236)

Für die An- bzw. Abwahl von G236 sind jeweils 2 Bewegungssätze mit Bewegungsinformationen erforderlich.

Die programmierten Bewegungssätze in den folgenden Abbildungen verlaufen von S nach E1 und von E1 nach E2.

Bei An- bzw. Abwahl sind alle Kombinationen von Linear- und Zirkularsätzen zulässig.

Hinweis

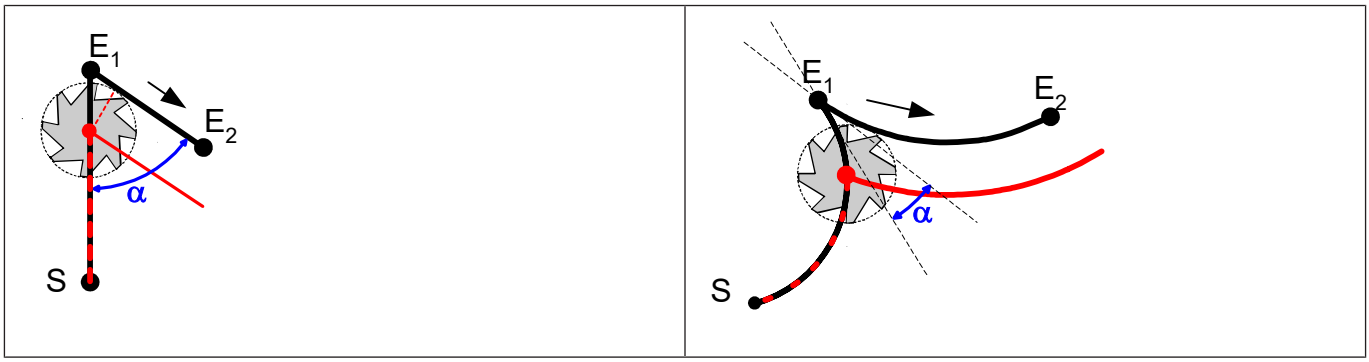
Wird bei G236 die Übergangsstrategie G25 (Einfügen von Linearsätzen) verwendet, so wird bei entsprechendem Übergangswinkel bei An- oder Abwahl der WRK ein Kreisbogen integriert.

Hinweis

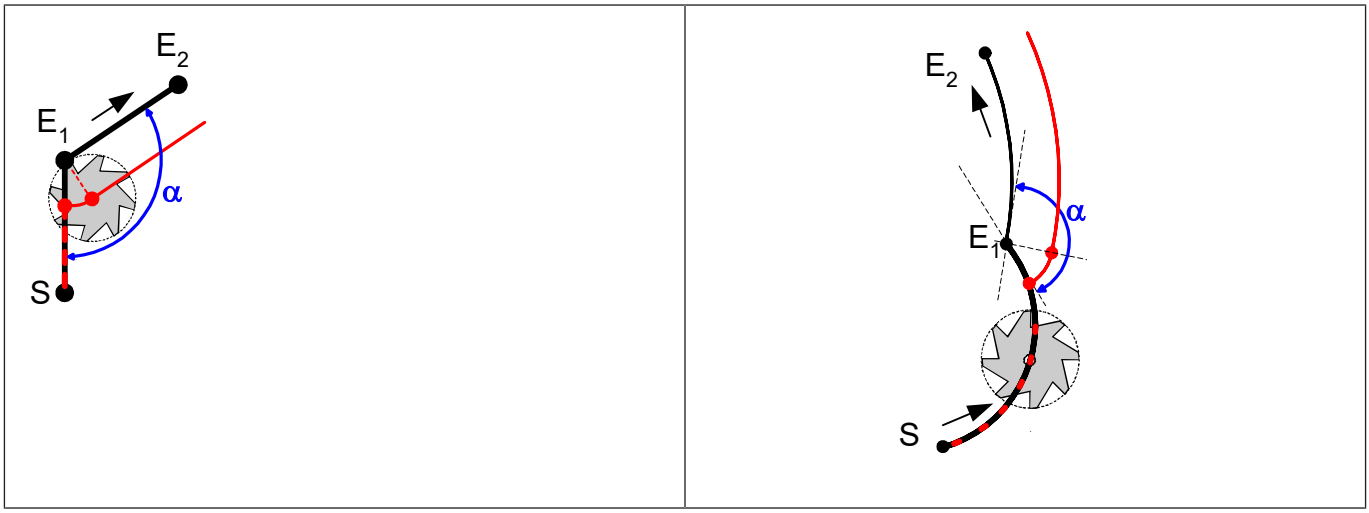
Bei Verwendung einer 2-Pfadkonfiguration ist die An- bzw. Abwahl nur mit Linearsätzen möglich. Bei zirkularer An- bzw. Abwahl wird die entsprechende Fehlermeldung 90181 bzw. 90182 ausgegeben.

Anwahl

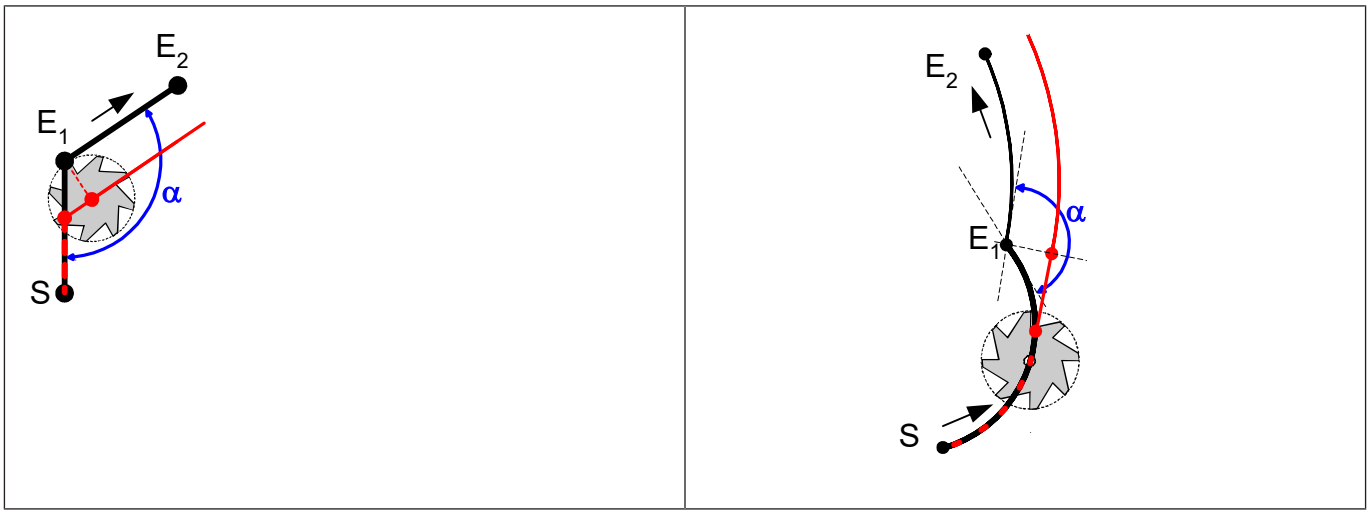
Der Werkzeugradius wird folgendermaßen aufgebaut:



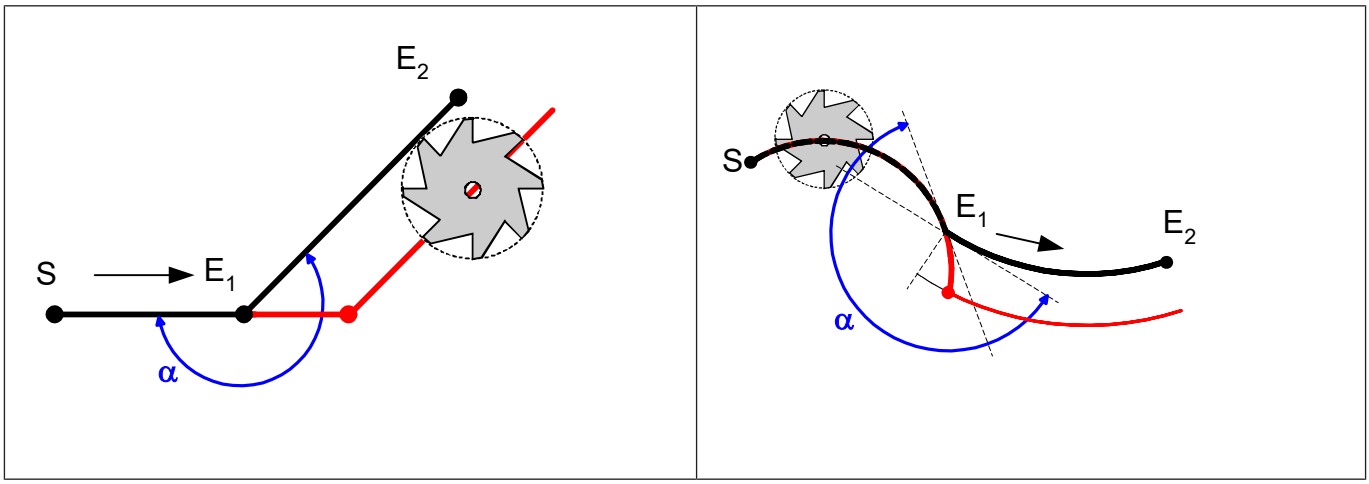
Übergangswinkel 0° < α < 90°



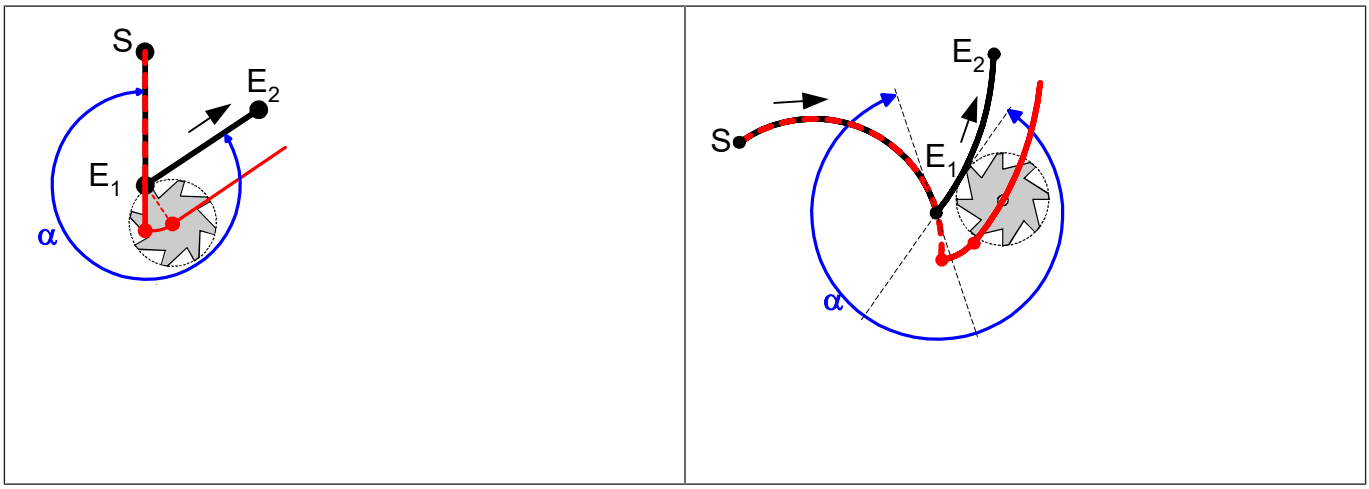
Übergangswinkel 90° <= α <= 180° , TRC-Option G236_LIN= 0 (Standard)



Übergangswinkel 90° <= α <= 180° , TRC-Option G236_LIN= 1



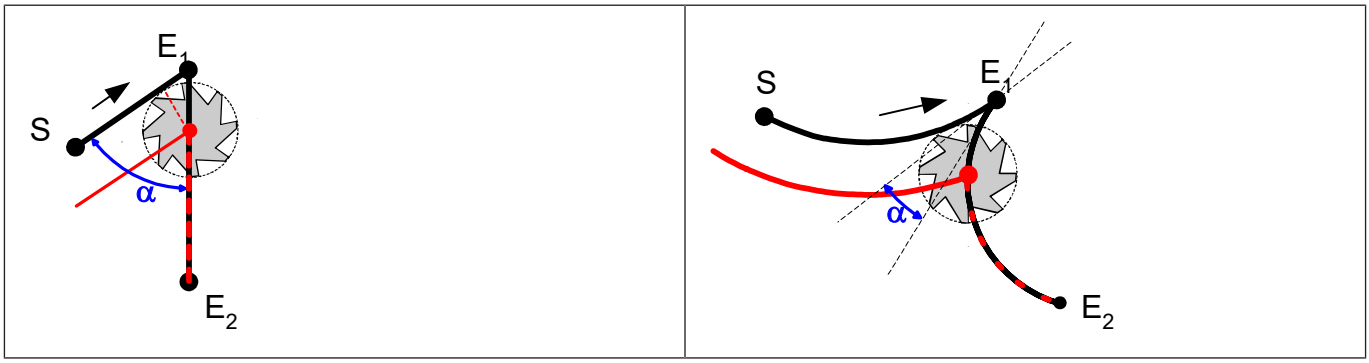
Übergangswinkel 180° < α <= 270°



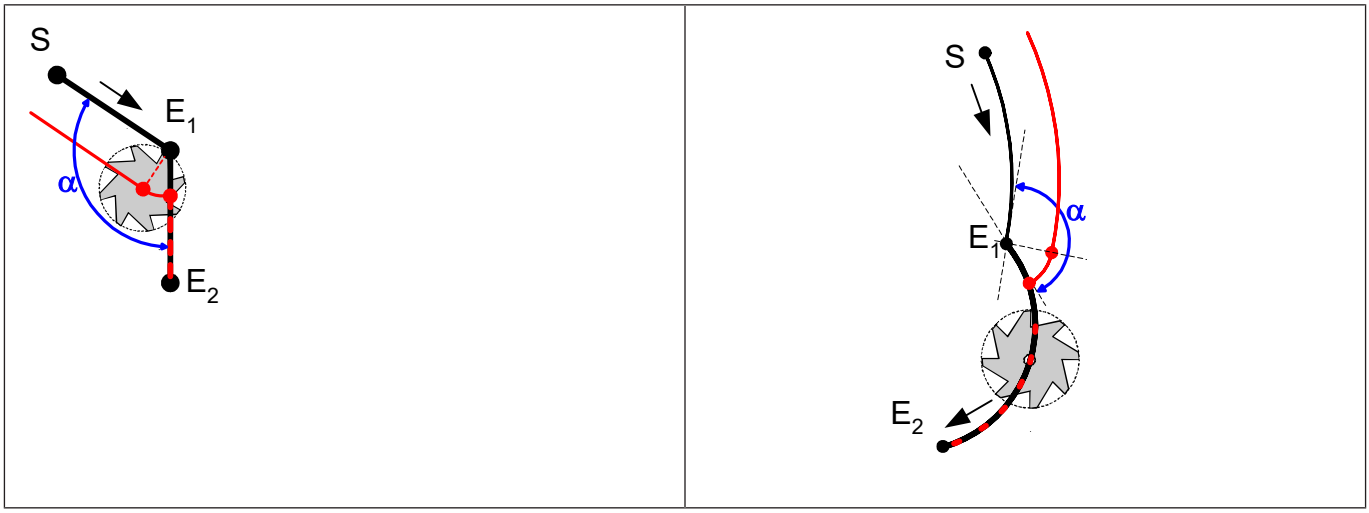
Übergangswinkel 270° < α < 360°

Abwahl

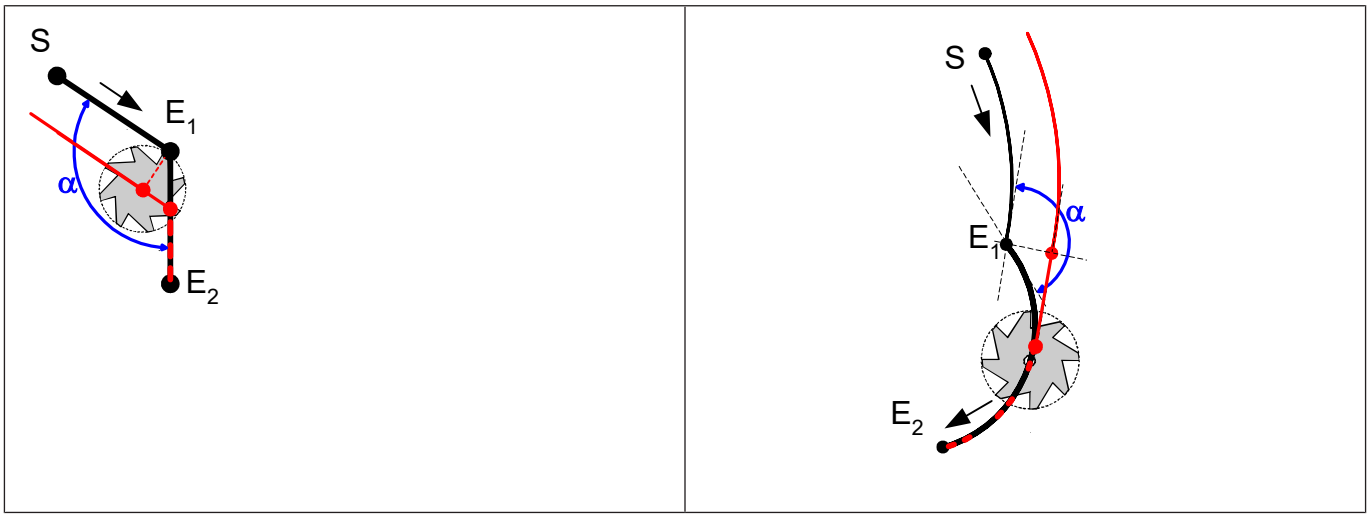
Analog zum Aufbau des Werkzeugradius wird dieser bei der Abwahl der WRK folgendermaßen abgebaut:



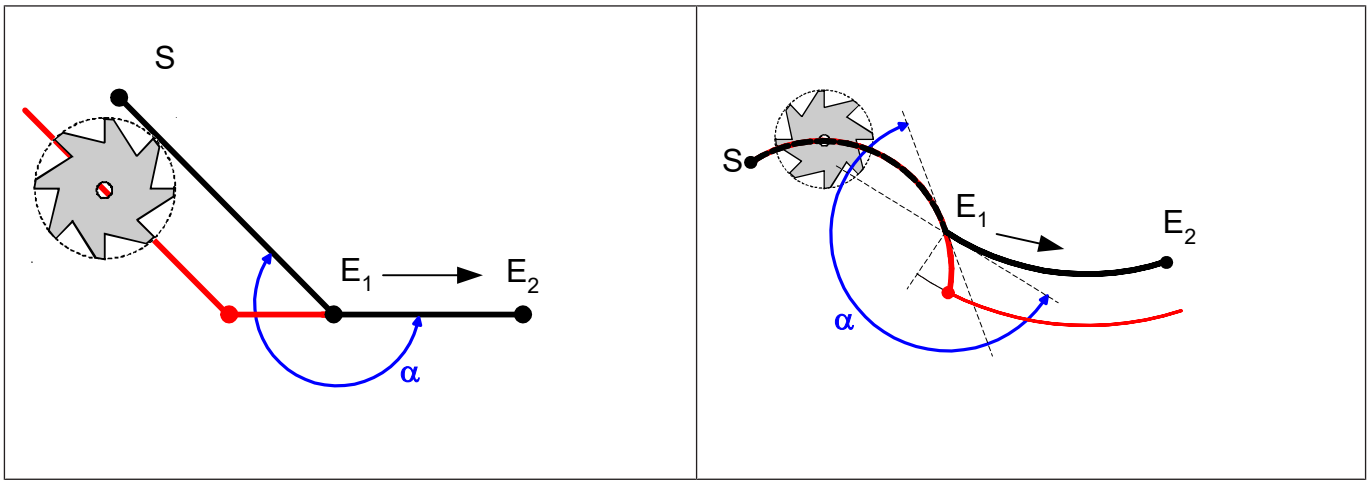
Übergangswinkel 0° < α < 90°



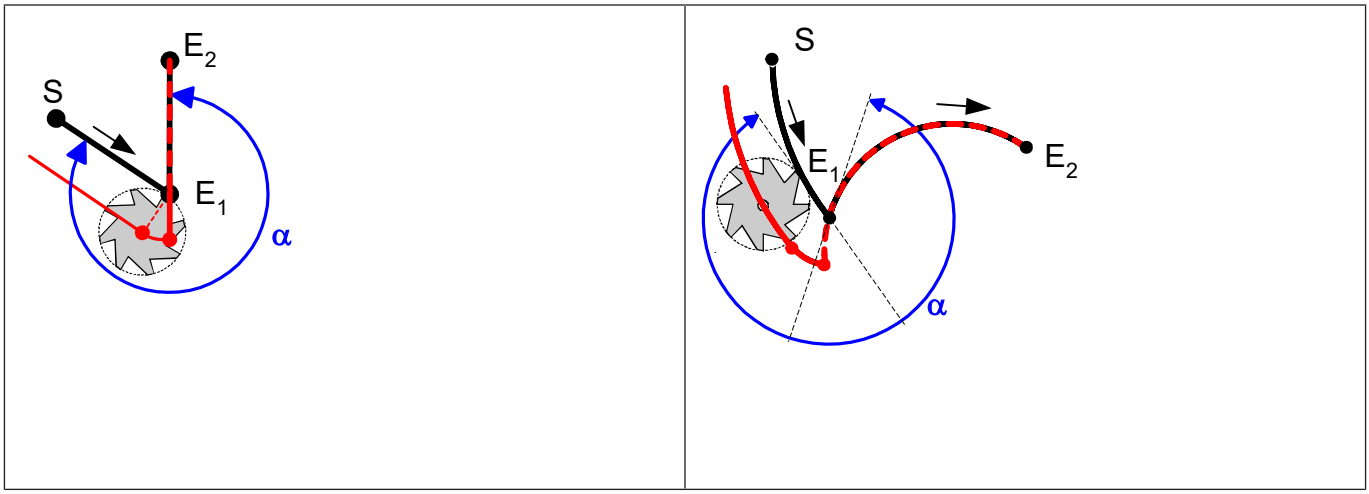
Übergangswinkel 90° <= α <= 180° , TRC-Option G236_LIN= 0 (Standard)



Übergangswinkel 90° <= α <= 180° , TRC-Option G236_LIN= 1



Übergangswinkel 180° < α <= 270°



Übergangswinkel 270° < α < 360°

13.2.6.1 An-/Abwahl G236 bei geschlossenen Konturen

13.2.6.1.1 An- und Abwahl bei Innenecken

An- und Abwahl an einer Innenecke bedeutet, dass der Übergangswinkel β zwischen dem ersten und letzten Bewegungssatz der Kontur kleiner als 180° Grad ist.

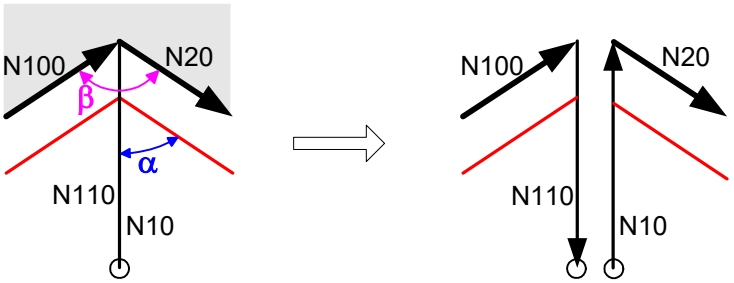
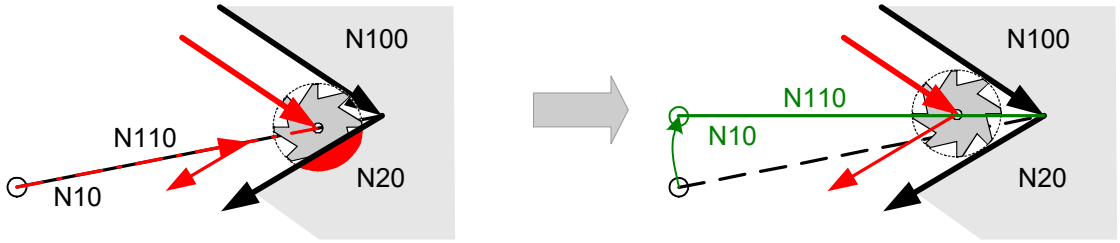


Abb. 142: An- und Abwahl geschlossener Konturen am Inneneck

Es wird empfohlen den An- und Abwahlpunkt der WRK auf der Winkelhalbierenden im Anwahleck zu platzieren um Konturschäden zu vermeiden.

Bei ungeeigneter Platzierung des An- und Abwahlpunktes wird die Kontur beschädigt.



Konturverletzung mit G236 am Inneneck

Programmierbeispiel

An- und Abwahl mit G236

```
%g236_test.nc
N01 G236 D1 F1000
N02G01 X10 Y10 (Anwahlpunkt)
N05 G42
N10 G01 X30 Y30
N20 G01 X30 Y20
;
N100 G01 X10 Y30
N105 G40
N110 G01 Y10 Y10 (Abwahlpunkt)
;
N999 M30
```

13.2.6.1.2 An- und Abwahl bei Außenecken

An- und Abwahl an einer Außenecke bedeutet, dass der Übergangswinkel β zwischen dem ersten und letzten Bewegungssatz der Kontur größer als 180° Grad ist.

Es wird empfohlen den An- und Abwahlpunkt innerhalb des grün markierten Bereichs, wie in nachfolgender Abbildung dargestellt, zu platzieren.

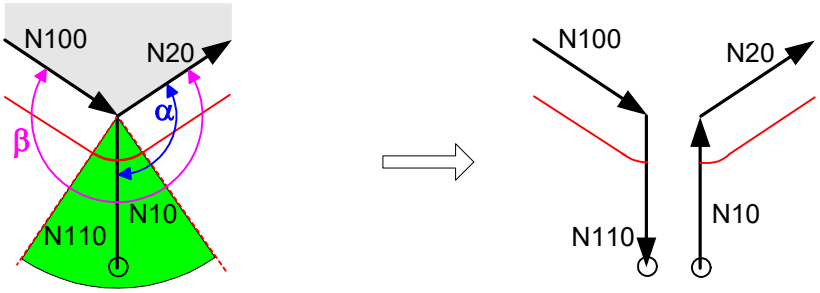


Abb. 143: An- und Abwahl geschlossener Konturen am Außeneck

13.2.7 Generierung von Korrektursätzen

Die Erzeugung von Korrektursätzen unterliegt keinen Einschränkungen bezüglich der NC-Satzfolge, wie sie bei der direkten oder indirekten WRK-Anwahl gelten. Für die Berechnung von Konturübergängen zwischen den programmierten NC-Sätzen wird jeweils der nächste zum gerade aktuellen Satz herangezogen.

Der Konturübergang erfolgt standardmässig auf Geraden (G25) – optional auf Kreisbögen (G26) – mit der Möglichkeit zur Geschwindigkeitsanpassung (G10/G11).

Die folgende Tabelle, sowie die ergänzenden Bilder zeigen alle möglichen Konturübergänge, wobei beide möglichen Übergänge (lineare bzw. zirkulare Zwischensätze) dargestellt werden.

Einfügen von Sätzen durch die WRK

LIN: Linearsatz, ZIR: Zirkularsatz

G25: Einfügen von Übergangsgeraden

G26: Einfügen von Übergangskreisen

(2 LIN : Die WRK fügt am Übergang 2 Linearsätze ein).

	Winkelbereich					
	0° bis 180°		180° bis 270°		270° bis 360°	
progr. Satzfolge	G25	G26	G25	G26	G25	G26
LIN-LIN	0	0	0	1 ZIR	1 LIN	1 ZIR
LIN-ZIR / ZIR-LIN	0	0	1 LIN	1 ZIR	2 LIN	1 ZIR
ZIR-ZIR	0	0	2 LIN	1 ZIR	3 LIN	1 ZIR

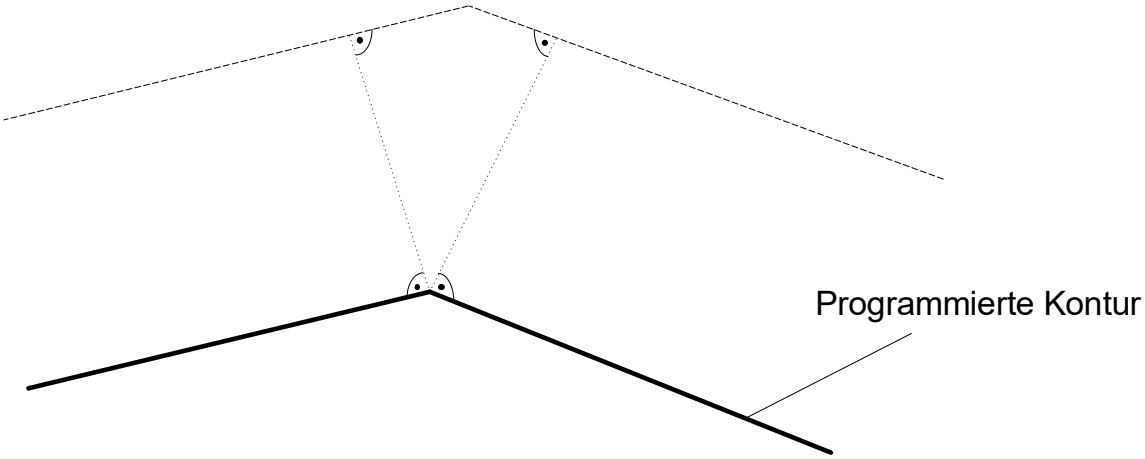


Abb. 144: Beispiel für Konturübergang auf Geraden bei Satzfolge Linear- Linear

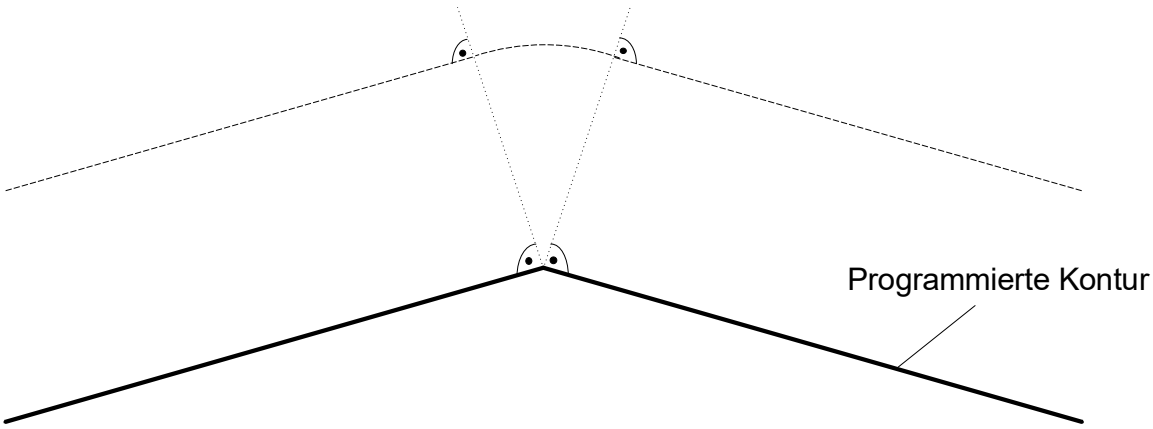
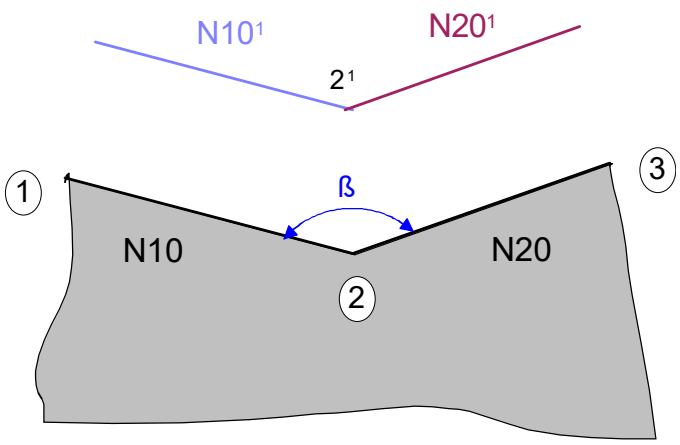
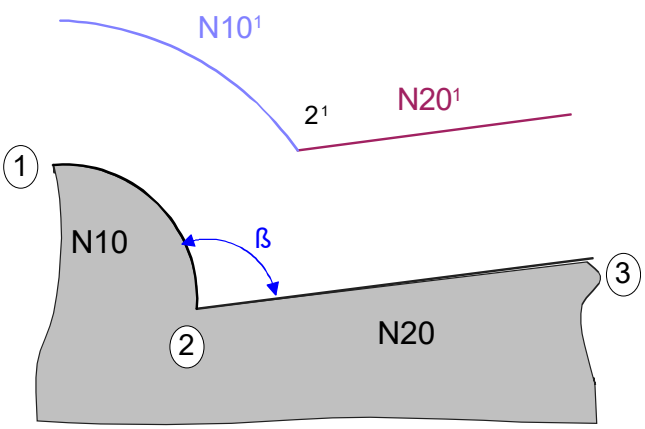
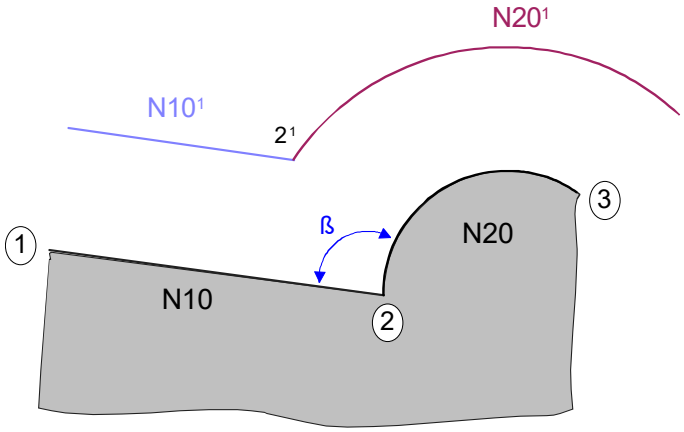
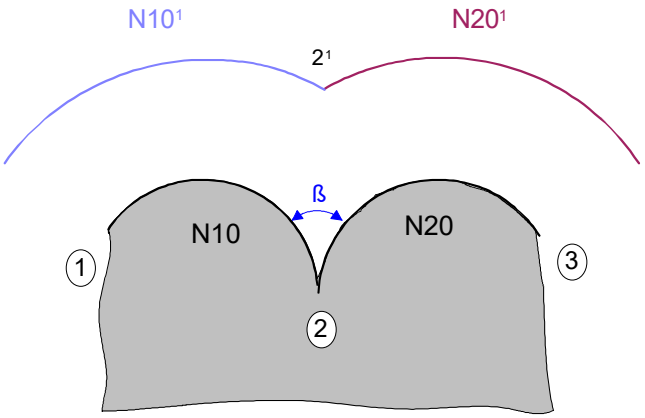
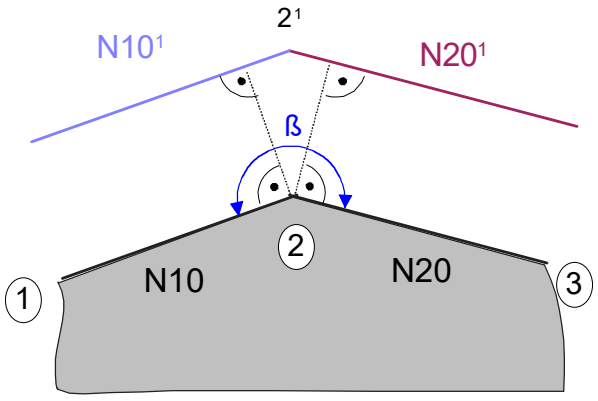
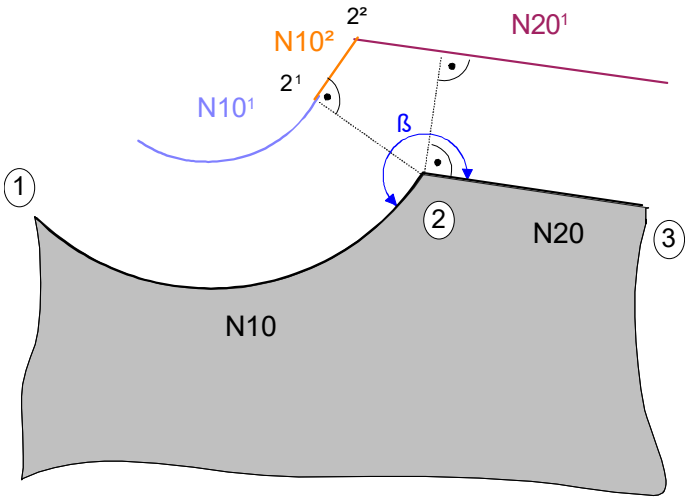
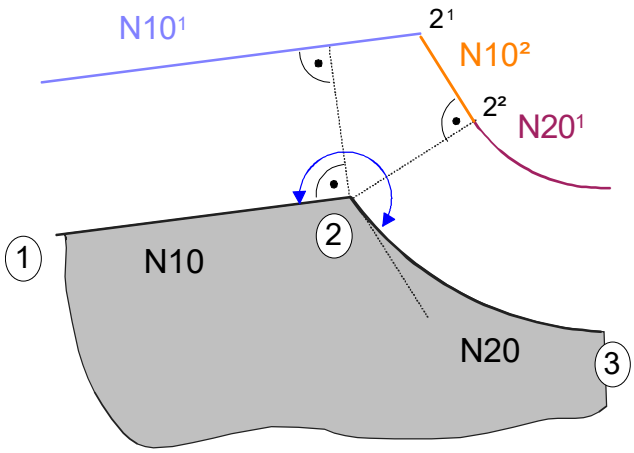
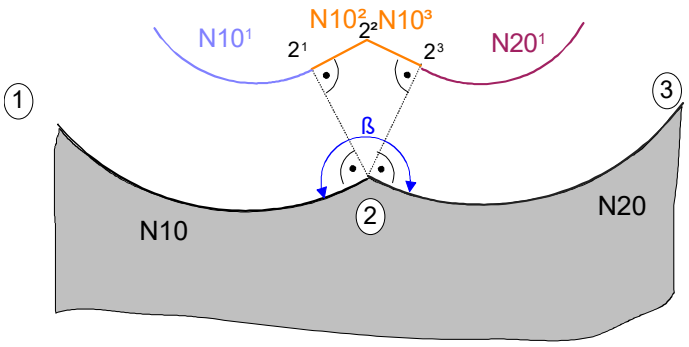


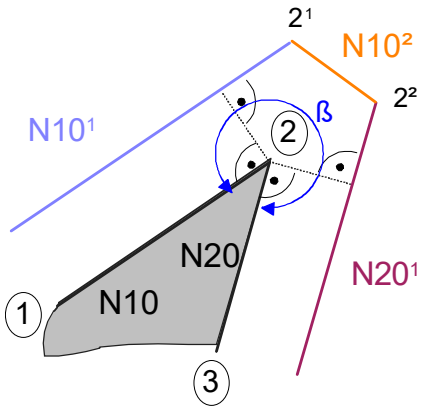
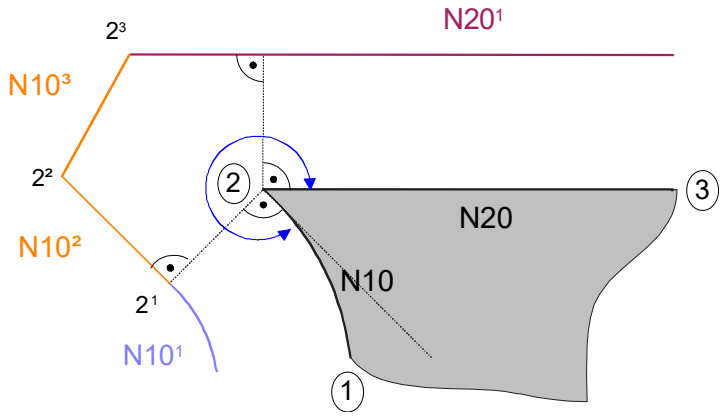
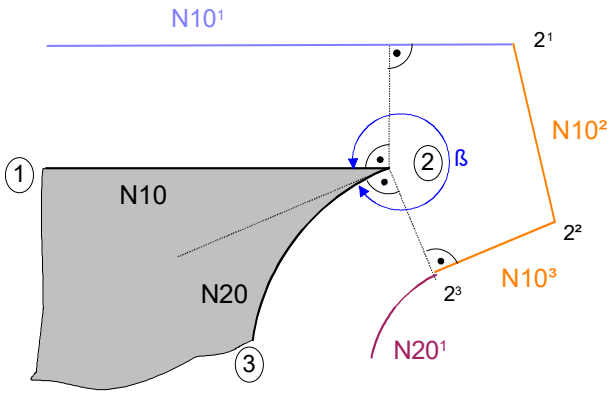
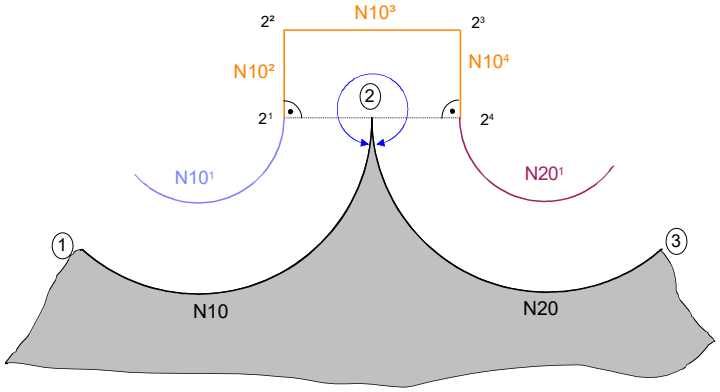
Abb. 145: Beispiel für Konturübergang auf einem Kreisbogen bei Satzfolge Linear-Linear

Kombinationstyp 1: 0° < ß ≤ 180°, G41	Kombinationstyp 3: 0° < ß ≤ 180°, G41
	
Kombinationstyp 2: 0° < ß ≤ 180°, G41	**Kombinationstyp 4: 0° < ß ≤ 180°, G41**
	

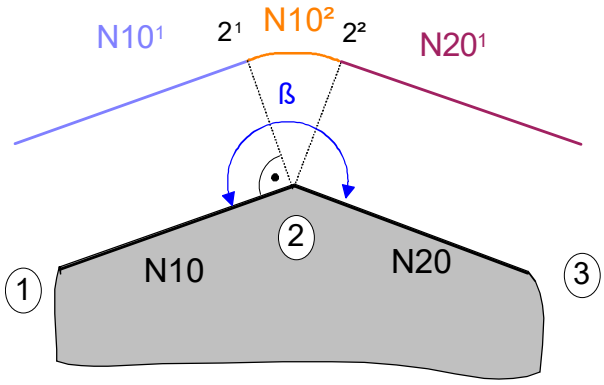
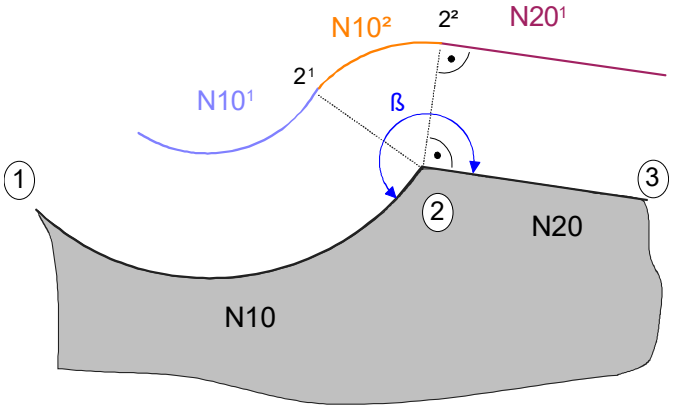
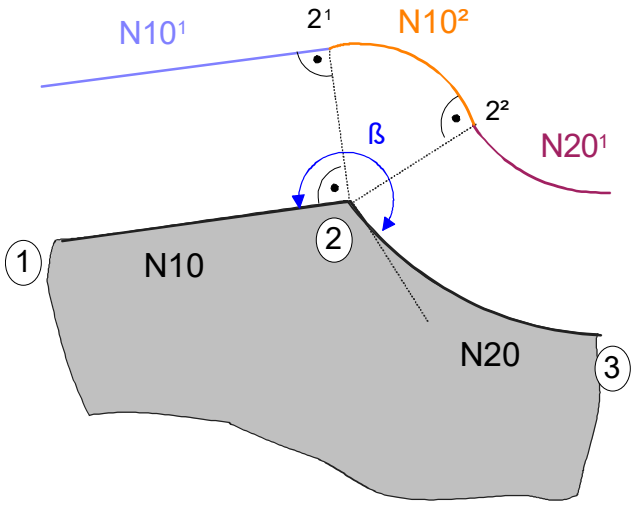
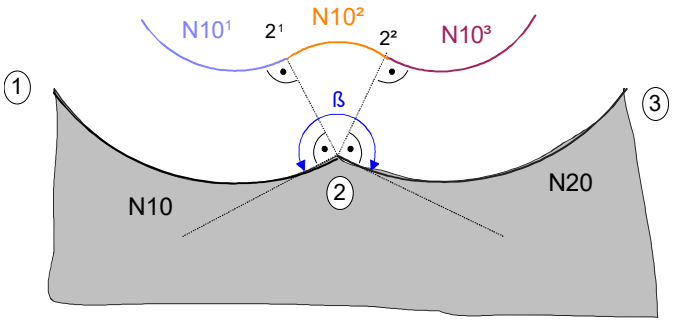
WRK-Übergänge: Kombinationsfälle für 0° < ß ≤ 180 ° (unabhängig von G25 bzw. G26, da keine Korrektursätze eingefügt werden)

Kombinationstyp 1: 180° < β ≤ 270°, G41	Kombinationstyp 3: 180° < β ≤ 270°, G41
	
Kombinationstyp 2: 180° < β ≤ 270°, G41	**Kombinationstyp 4: 180° < β ≤ 270°, G41**
	

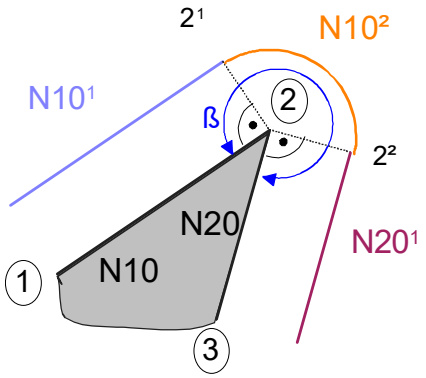
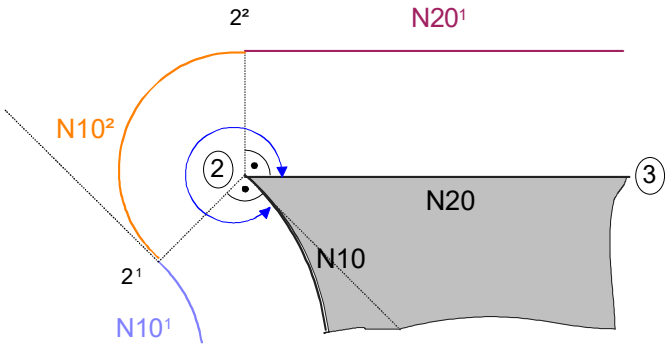
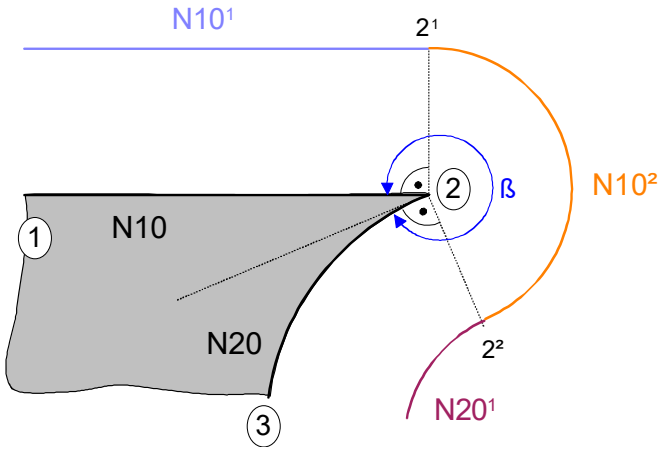
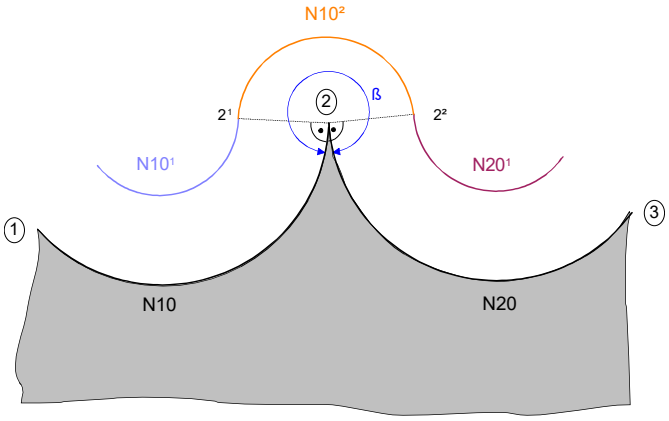
WRK-Übergänge: Kombinationsfälle für 180° < ß ≤ 270° mit linearen Zwischensätzen

Kombinationstyp 1: 270° < ß ≤ 360°, G41	Kombinationstyp 3: 270° < ß ≤ 360°, G41
	
Kombinationstyp 2: 270° < ß ≤ 360°, G41	**Kombinationstyp 4: 270° < ß ≤ 360°, G41**
	

WRK-Übergänge: Kombinationsfälle für 270° < ß ≤ 360° mit linearen Zwischensätzen

Kombinationstyp 1: 180° < β ≤ 270° , G41, G26	Kombinationstyp 3: 180° < β ≤ 270° , G41, G26
	
Kombinationstyp 2: 180° < β ≤ 270° , G41, G26	**Kombinationstyp 4: 180° < β ≤ 270° , G41, G26**
	

WRK-Übergänge: Kombinationsfälle für 180° < ß ≤ 270° mit kreisförmigen Zwischensätzen

Kombinationstyp 1: 270° < ß ≤ 360°, G41, G26	Kombinationstyp 3: 270° < ß ≤ 360°, G41, G26
	
Kombinationstyp 2: 270° < ß ≤ 360°, G41, G26	Kombinationstyp 4: 270° < ß ≤ 360°, G41, G26
	

WRK-Übergänge: Kombinationsfälle für 270° < ß ≤ 360° mit kreisförmigen Zwischensätzen

13.2.8 Verhalten bei Konturwechsel

Ein direkter Konturwechsel z.B. von G41 nach G42 ist ohne explizite Abwahl (G40) der WRK möglich. Der Konturwechsel entspricht dem einer WRK-Ab- und Anwahl. Der Konturwechsel darf wie bei den Ausfahr- und Anfahrsätzen nur bei linearen NC-Sätzen ausgeführt werden. Direkt vor und nach dem Konturwechsel sind auch Zirkularsätze erlaubt.

Hinweis

Wird beim Konturwechsel ein Zirkularsatz programmiert, so wird eine Fehlermeldung ausgegeben.

Programmierbeispiel

Verhalten bei Konturwechsel

```
N05 G01 Y10 F100    (Linearsatz mit WRK-Anwahl links der)
G41 V.G.WZR=2       (Kontur )
N10 X20 Y25
N20 X40 G42         (Linearsatz mit WRK-Anwahl rechts der)
                    (Kontur)
N30 X60 Y10
N40 X0 Y0 G40       (Abwahl der WRK)
N50 M30
```

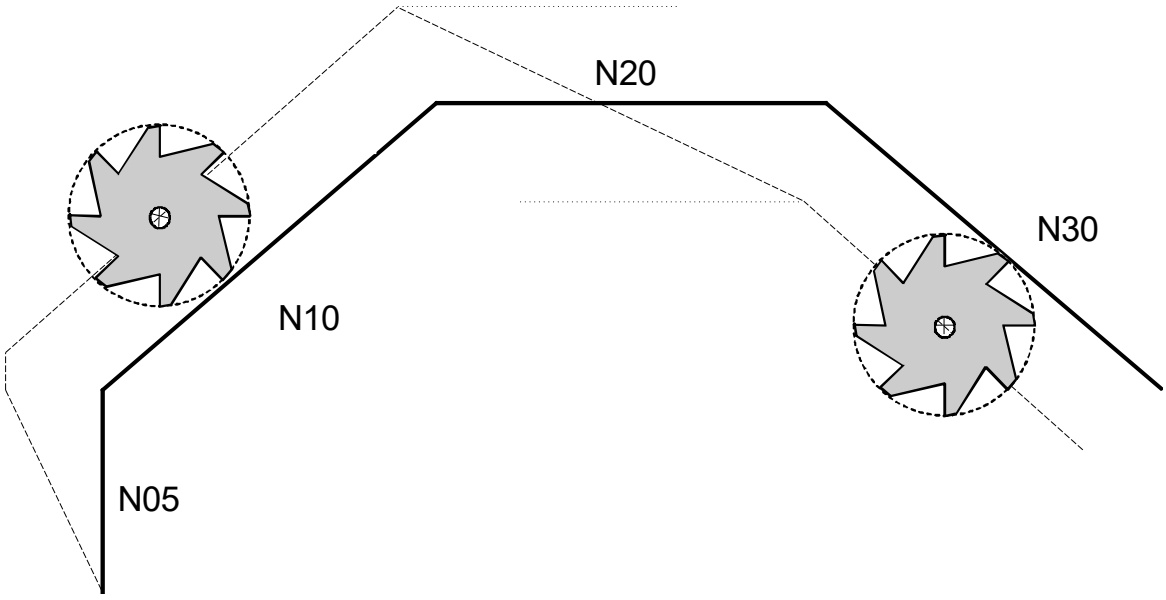


Abb. 146: Beispiel eines Anwahlwechsels ohne Abwahl

13.2.9 Verhalten bei Änderung des Werkzeugradius

Eine Änderung des Werkzeugradius ist sowohl innerhalb eines Linearsatzes als auch bei Zirkularsätzen möglich.

Programmierbeispiel

Änderung innerhalb eines Linearsatzes

```
%wr_lin.nc

N10 V.G.WZ_AKT.R = 10
N20 G00 X0 Y0 Z0 F1000
N25 G41 G1 X20 Y20
N30 G01 X100
N35 V.G.WZ_AKT.R = 20
N40 G01 X200
N50 X240 Y100
...
N200 G40 X500
N999 M30
```

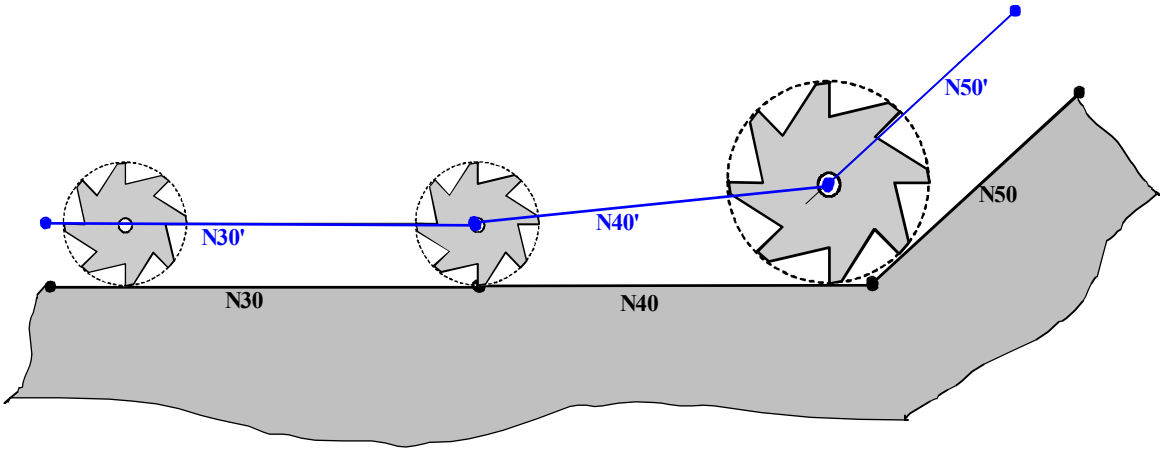


Abb. 147: Werkzeugradiusänderung im Linearsatz

Programmierbeispiel

Änderung innerhalb eines Zirkularsatzes

```
%wr_rad.nc

N10 V.G.WZ_AKT.R = 10
N20 G00 X0 Y0 Z0 F1000
N25 G41 G1 X20 Y20
N30 G01 X100
N35 V.G.WZ_AKT.R = 20
N40 G02 X180 Y100 R50
N50 X240 Y150
...
N200 G40 X500
N999 M30
```

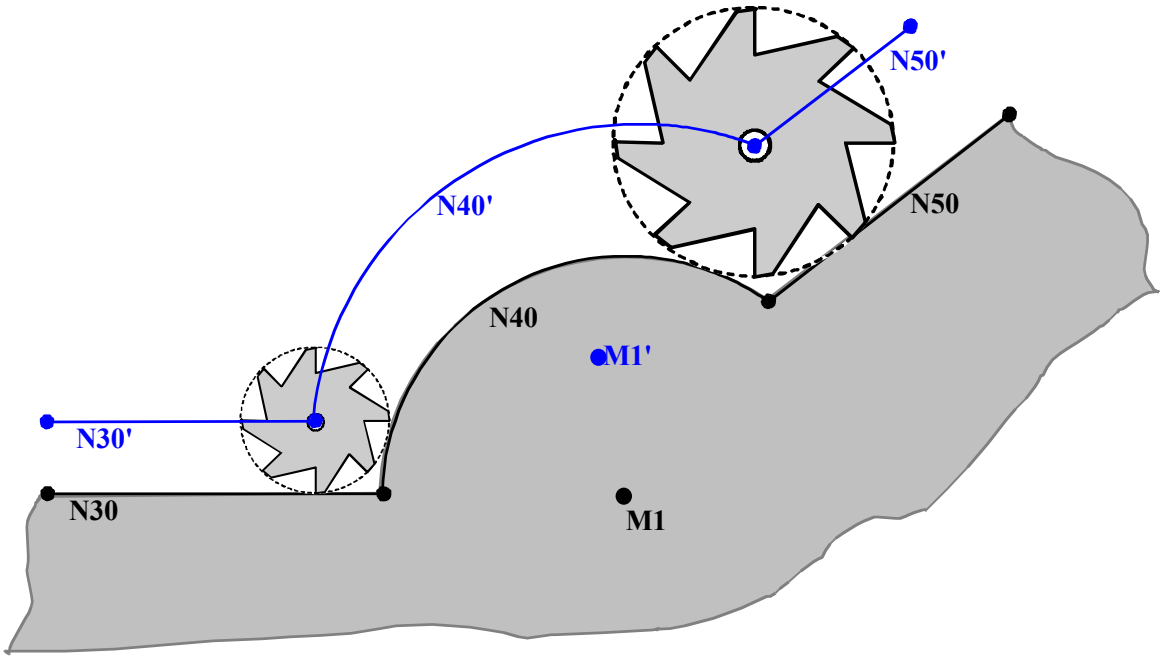


Abb. 148: Werkzeugradiusänderung im Zirkularsatz

13.2.10 Tangentiale An-/Abwahl der WRK (G05)

Bei der direkten An-/Abwahl der WRK entsteht in aller Regel ein Knick in der Bahnkontur bei Bearbeitungsbeginn. Bei Winkeln größer 180° wird zudem die Kontur des auf die Anwahl folgenden und des vor der Abwahl liegenden Satzes verletzt.

Zur Vermeidung dieser Konturverletzungen bei direkter An-/Abwahl, sowie zur Verringerung des auftretenden Ruckes an Knickpunkten auf der Trajektorie, dient das tangentiale Ein- und Ausfahren.

G05 muss in Verbindung mit G40, G41, G42 im gleichen Satz programmiert werden. Daraus wird abgeleitet, ob ein tangentialer Übergang am Anfang oder am Ende der Kontur stattfinden soll.

Vom aktuellen Standpunkt aus wird das nächste Konturelement G01, G02... tangential im Kreis mit programmiertem Vorschub angefahren; ggf. muss eine Vorschubanpassung mit G10/G11 erfolgen.

G05 in Verbindung mit G41 oder G42 bewirkt ein tangentiales Einfahren am Konturanfang, G05 mit G40 bei aktivem G41/G42 ein tangentiales Ausfahren am Konturende. Damit wird der An-/Abwahlsatz in einen Zirkularsatz umgewandelt.

Die Bewegungssätze, die durch die G05-Funktion bei der An- und Abwahl der WRK ausgelöst werden, sind auf den Abbildungen auf den beiden folgenden Seiten verdeutlicht. Insgesamt werden vier Verläufe dargestellt, die sich bei gleicher Programmierung durch unterschiedliche Startpunkte (P1, P1') ergeben.

Tangentiale Anwahl:

Der Anwahlpunkt (AnP) wird wie bei der konventionellen direkten Anwahl berechnet. Anhand der Orientierung des ersten angewählten Satzes und der Lage des Startpunktes wird die Drehrichtung des Zirkularanwahlsatzes festgelegt. Der Kreismittelpunkt (MP) ergibt sich aus dem Schnittpunkt der Mittelsenkrechten vom Startpunkt (P1 bzw. P1') und Anwahlpunkt (AnP) und der Geraden Startpunkt des Anwahlsatzes (P2) – Anwahlpunkt (AnP).

Tangentiale Abwahl:

Der letzte korrigierte Endpunkt (AbP), im Folgenden letzter Anwahlpunkt genannt, wird wie bei der konventionellen direkten Abwahl berechnet. Anhand der Orientierung des letzten angewählten Satzes und der Lage des Abwahlpunktes (P4 bzw. P4') wird die Drehrichtung des Zirkularabwahlsatzes festgelegt. Der Kreismittelpunkt ergibt sich aus dem Schnittpunkt der Mittelsenkrechten der Verbindungslinie des letzten Anwahlpunktes (AbP) und des Abwahlpunktes (P4 bzw. P4') und der Geraden letzter Anwahlpunkt (AbP) – letzter vor der Abwahl programmierter Punkt (P3).

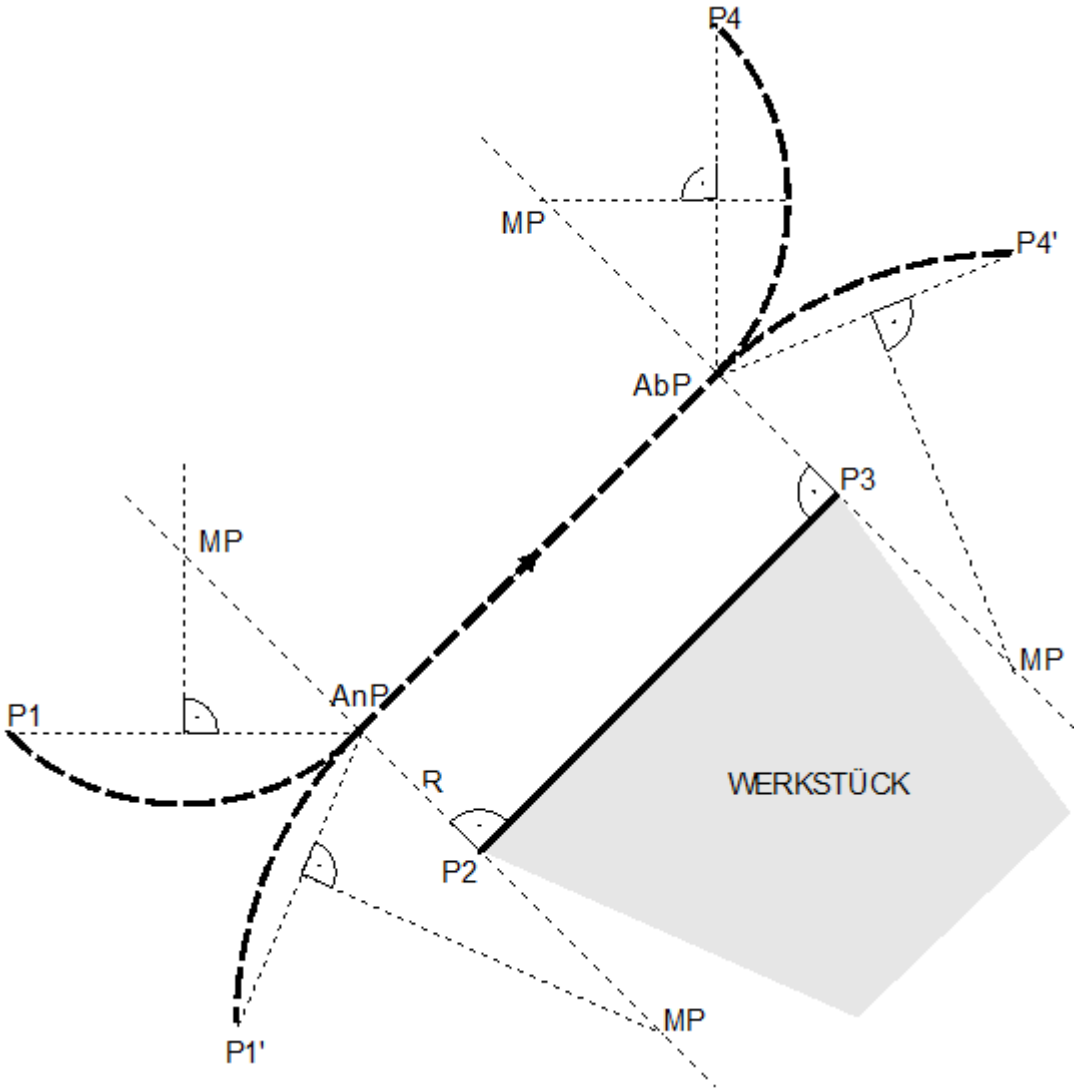


Abb. 149: Tangentiale An- und Abwahl der WRK

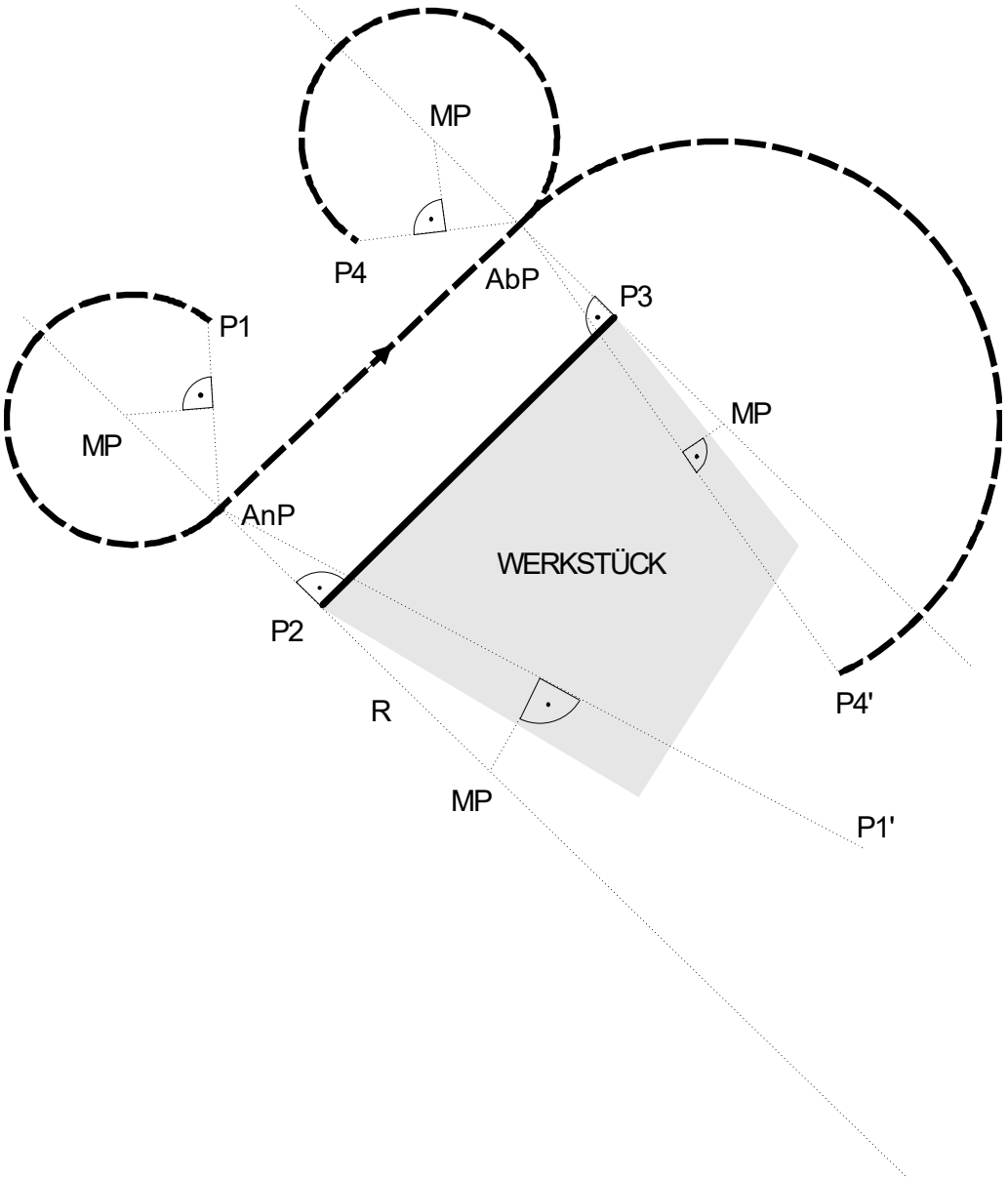


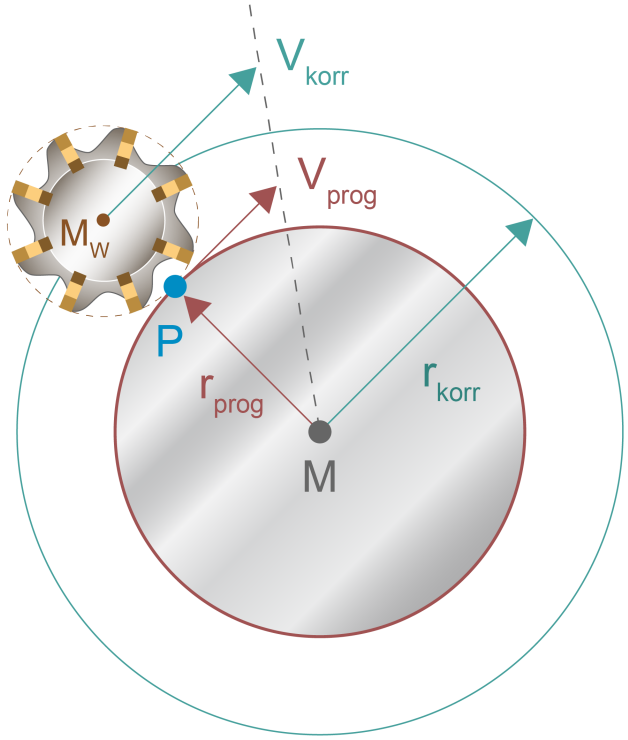
Abb. 150: Tangentiale An- und Abwahl der WRK

13.2.11 Vorschubanpassung der WRK (G10/G11)

Syntax:

G10	Vorschub konstant	modal, Grundzustand
G11	Vorschub angepasst	modal

Bei wirksamem G11 werden von der Werkzeugradiuskorrektur eingefügte Übergangssätze mit höherer Geschwindigkeit abgefahren. Außerdem wirkt dann bei Zirkularsätzen der programmierte Vorschub im Werkzeugeingriffspunkt P (siehe folgende Abbildung).



$$V_{korr} = \frac{r_{korr}}{r_{prog}} \text{ x } V_{prog}$$

P = Werkzeugeingriffspunkt

M_W = Werkzeugmittelpunkt

r_{prog} = programmierter Radius

r_{korr} = korrigierter Radius

V_{prog} = programmierte Geschwindigkeit

V_{korr} = korrigierte Geschwindigkeit

Anpassung des Vorschubes bei korrigierter Kreisinterpolation

13.2.12 An-/Abwahl der WRK-Konturausblendung (G140/G141)

Syntax:

G140	Abwahl der Konturausblendung	modal, Grundzustand
G141	Anwahl der Konturausblendung	modal

Wird vor der Endbearbeitung ein Schruppvorgang mit einem Werkzeug mit großem Durchmesser durchgeführt, so besteht die Gefahr der Konturverletzung, wenn bei einzelnen Konturelementen die Abmessungen kleiner als die der Werkzeuggeometrie sind. Um zusammengehörige Konturobjekte fahren zu können, besteht durch G140/G141 bei aktiver WRK die Möglichkeit, bei erkannten Verletzungen die entsprechenden Konturelemente auszublenden und die darauffolgende Kontur so zu behandeln, dass diese nicht beschädigt wird.

Es wird keine Fehlermeldung ausgegeben, wenn bei Programmende die Konturausblendung noch angewählt ist. Bei Programmstart wird die Grundstellung G140 angewählt.

Achtung

Bei Abwahl der WRK durch G40 bleibt die Konturausblendung weiterhin angewählt und wird bei erneuter Anwahl der WRK wieder aktiv.

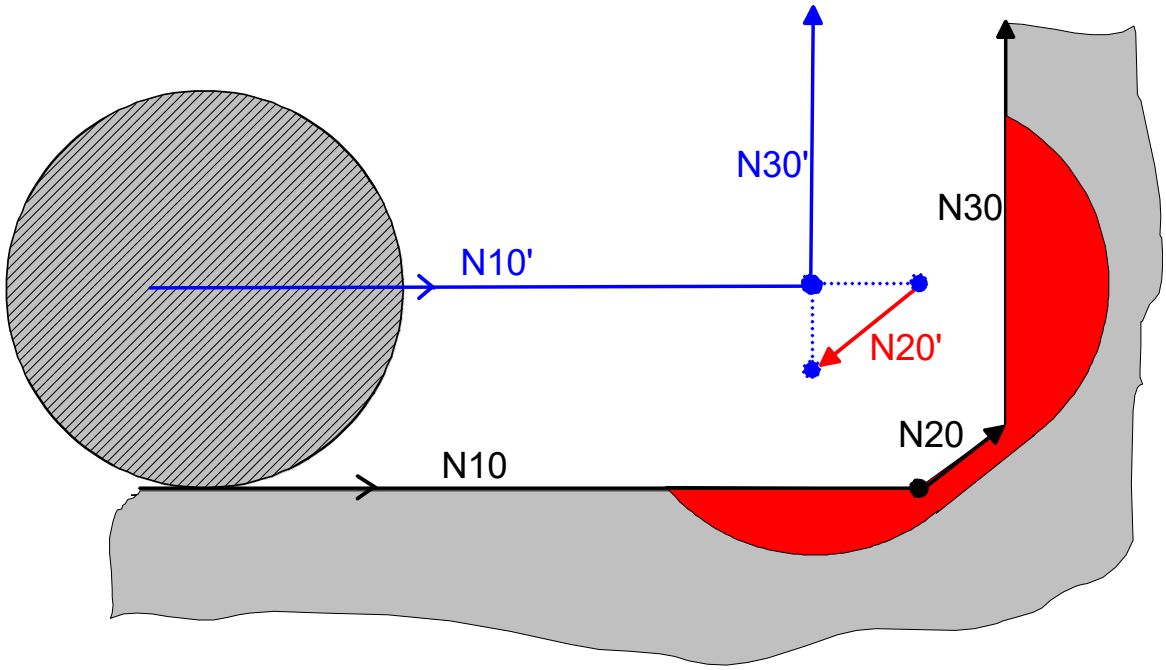


Abb. 151: Ausblenden von N20 zur Vermeidung einer Konturverletzung

13.2.13 Grenzen der WRK

Für die Berechnung von Zwischensätzen betrachtet die WRK den aktuellen und zwei weitere relevante NC-Sätze. "Relevant" bedeutet hier: Verfahrsätze in der angewählten Hauptebene, in der die Werkzeugradiuskorrektur erfolgt. "Nicht-relevant" wären z.B. Techno-Sätze, Zeitverzögerungen (G04) oder Verfahrbewegungen mit einer Achse senkrecht zur Hauptebene. Wenn innerhalb dieser 3 Sätze ein Zwischensatz errechnet wird, der in entgegengesetzter Richtung der programmierten Kontur verlaufen würde, so wird dadurch eine Konturverletzung erkannt. In unten stehender Abbildung ist dazu ein Beispiel dargestellt.

Wird eine Engstelle durch einen oder mehrere Sätze bewirkt, die weiter als 3 Sätze vom aktuellen Satz entfernt sind, so wird diese Konturverletzung von der WRK nicht erkannt. In der Abbildung auf der folgenden Seite ist dazu ein Beispiel dargestellt.

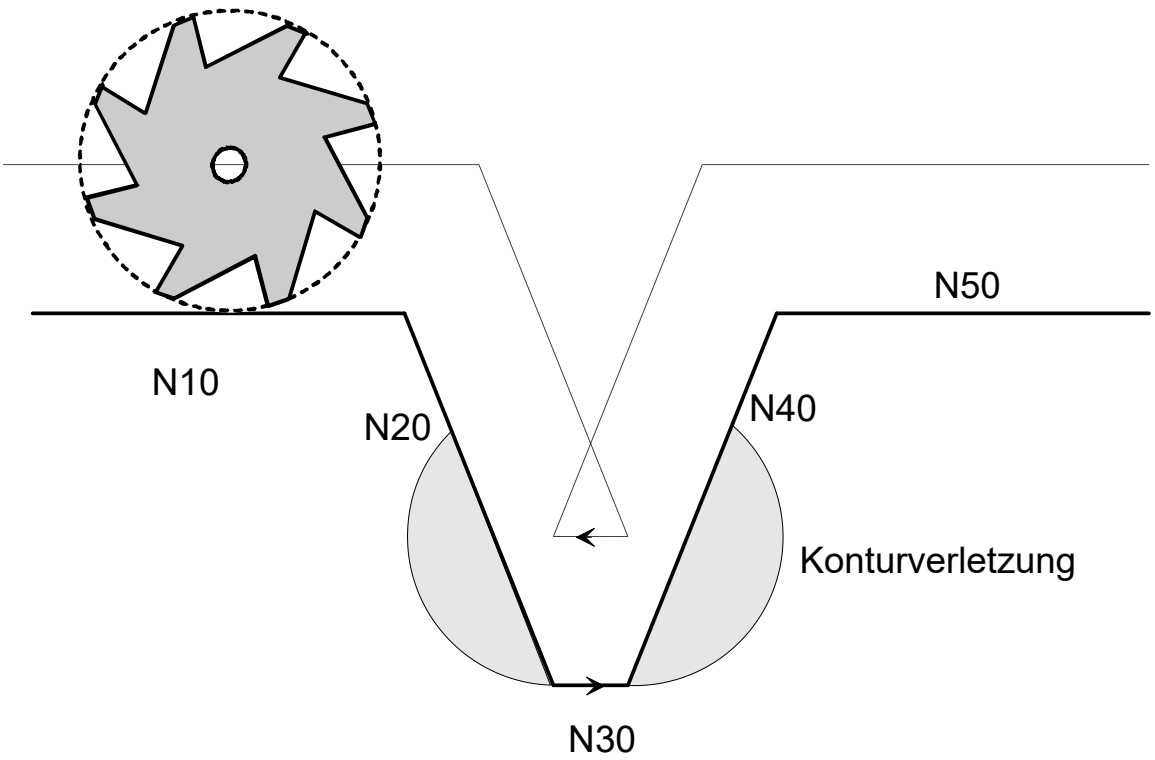


Abb. 152: Beispiel für erkannte Konturverletzung

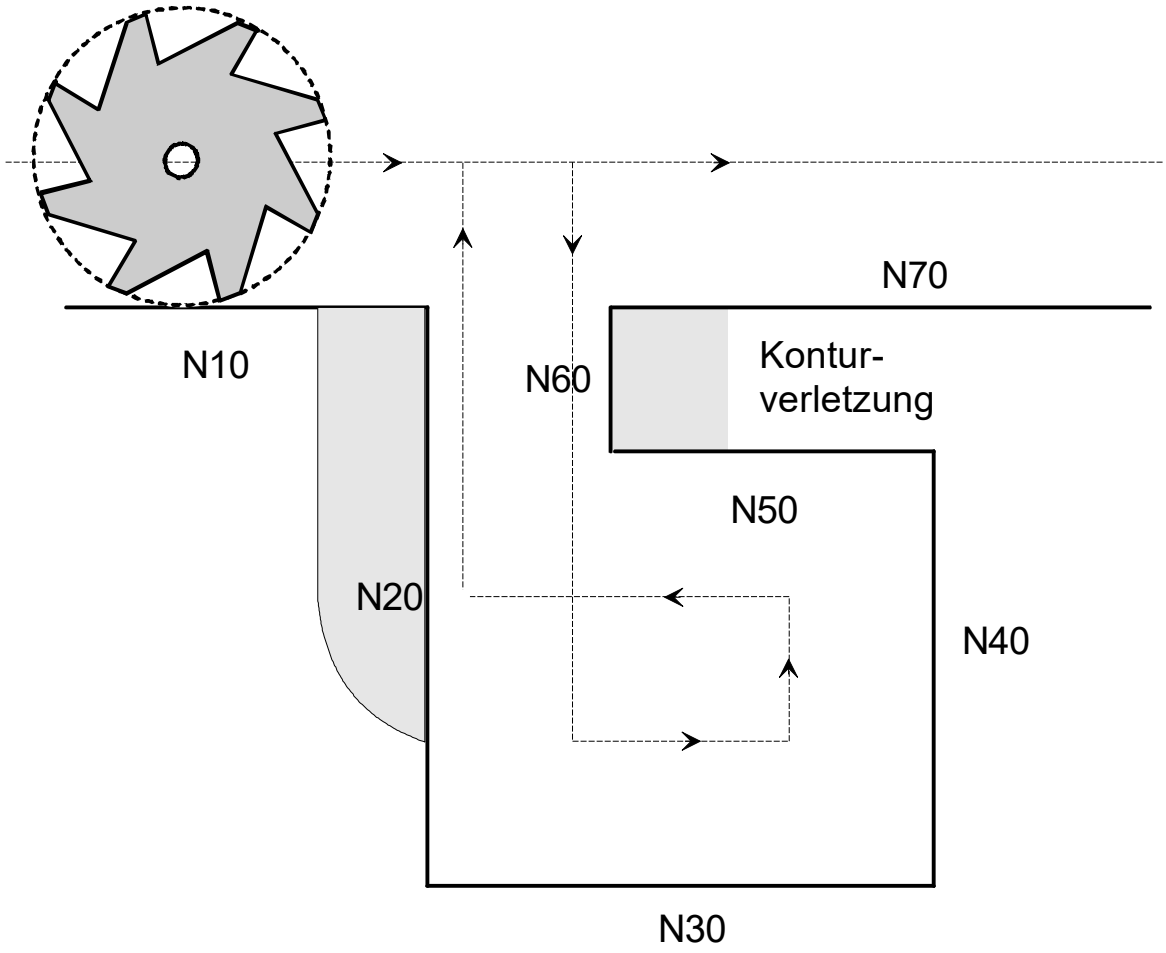


Abb. 153: Beispiel für nicht erkannte Konturverletzung

13.2.14 Programmierbare Zusatzoptionen der WRK (#TRC)

Der folgende Befehl ermöglicht die Programmierung zusätzlicher optionaler WRK-Funktionen.

Hinweis

Nach der Anwahl sind die optionalen WRK-Funktionen bis zum Hauptprogrammende bzw. RESET wirksam,- sie können aber jederzeit auch im NC-Programm wieder abgewählt werden.

Syntax:

#TRC [[CONV_CIR_TO_LIN=..] **[KERF_MASKING**=..] **[REVERSE**=..] [**IGNORE_CONT_DAMAGE**=..]
[**REMOVE_MASKED_BLOCKS**=..]] **[EXT_ANGLE_BLOCK_INTERSECTION**=..]]

CONV_CIR_TO_LIN=.. Mit diesem Parameter können Zirkularsätze, bei denen der Werkzeugradius größer als der programmierte Radius des Kreiselements ist, direkt in Linearsätze konvertiert werden. Voraussetzung für die Wirksamkeit ist, dass der Konturausblendprozesse aktiv ist (G141).

0: Keine Konvertierung von Zirkularsätzen (Standard)

1: Konvertierung von Zirkularsätzen in Linearsätze.

KERF_MASKING=.. Mit diesem Parameter wird explizit das Ausblenden von Kerben angewählt.

0: Kerbfunktionalität deaktiviert (Standard).

1: Kerbfunktionalität aktiviert.

Hinweis

Beim Konturausblenden wird diese WRK-Option implizit mit aktiviert bzw. deaktiviert.

Die Funktionsweise des Kerbausblendens basiert auf der Verschiebung eines programmierten Punktes, sobald festgestellt wird, dass mit einem Werkzeug eine Kerbe nicht fahrbar ist.

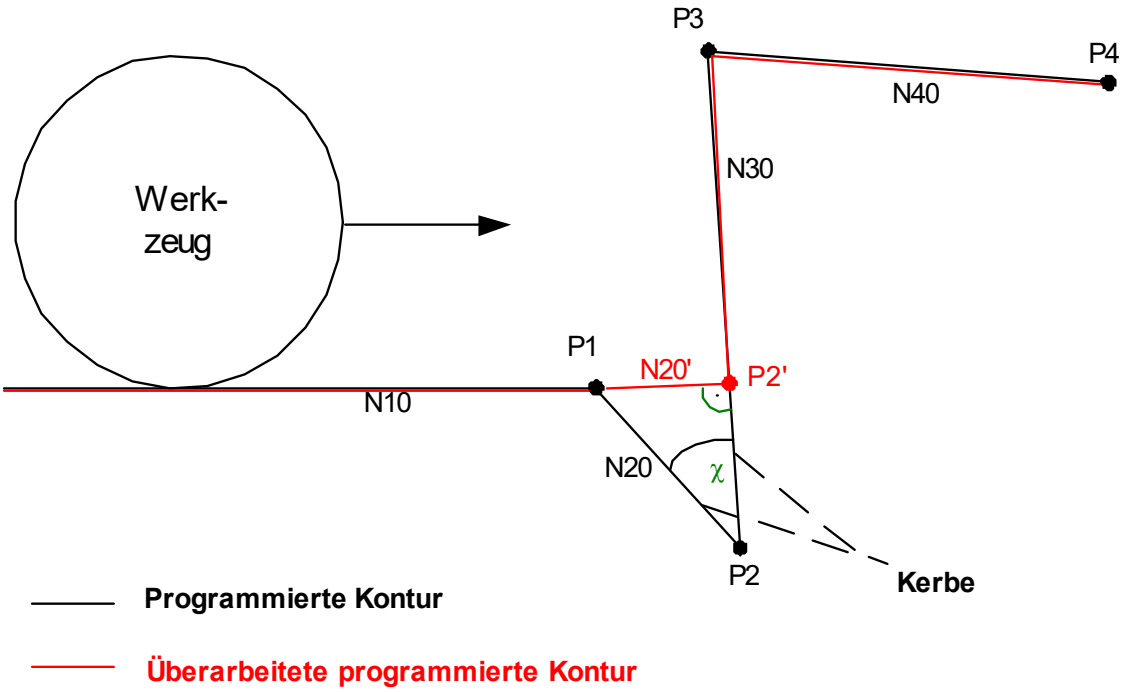


Abb. 154: Veranschaulichung des Kerbausblendens

REVERSE=..

Dieser Parameter erlaubt das direkte Wechseln (G41<->G42) der Anwahlseite bei reversierenden Bewegungen bei aktiver WRK.

0: Direkter Wechsel der Anwahlseite deaktiviert (Standard)

1: Direkter Wechsel der Anwahlseite aktiviert

Der Wechsel der Anwahlseite der WRK erfolgt stets im Umkehrpunkt. Voraussetzung hierfür sind bei Linearsätzen exakt reversierende Bewegungen.

Bei Zirkularsätzen müssen die Tangenten beider Kreiselemente im Umkehrpunkt identisch und die Kreisrichtungen beider Kreise unterschiedlich sein.

IGNORE_CONT_DAMAGE=..

Mit diesem Parameter werden explizit Konturverletzungen ignoriert.

0: Kein Ignorieren der Konturverletzungen (Standard)

1: Ignorieren der Konturverletzungen

REMOVE_MASKED_BLOCKS=..

Mit diesem Parameter werden erkannte Konturschleifen im Konturausblenden gelöscht. Reine Bewegungen von Mitschleppachsen bleiben erhalten, dazu zählt aus Sicht der WRK auch eine Bewegung der 3. Hauptachse.

Der Parameter ist speziell für Konturen mit sehr kurzen Sätzen geeignet.Voraussetzung für die Wirksamkeit ist, dass die Konturausblendung aktiv ist (G141).

0: Konturschleifen werden nicht gelöscht (Standard)

1: Konturschleifen werden gelöscht

EXT_ANGLE_BLOCK_INTERSECTION=..

Mit diesem Parameter kann der Grenzwert für den Übergangswinkel zwischen zwei Bewegungssätzen von 180° auf 181° verändert werden. Dies verhindert die Erzeugung zusätzlicher WRK-Übergangsätze in diesem Winkelbereich.

Der Grenzwert selbst kann nicht modifiziert werden.

0: Grenzwert für den Übergangswinkel ist 180° (Standard)

1: Grenzwert für den Übergangswinkel ist 181°

Programmierbeispiel

Konvertierung von Zirkularsätzen

```
N1000 V.G.WZ_AKT.R=1      (Radius des Werkzeuges)
N2300 G140 (Deaktivieren Konturausblenden)
....
N2450 #TRC [CONV_CIR_TO_LIN=1] (Aktivierung der Option)
N2500 G41 (Anwahl WRK links der Kontur)
...
(Kreiselement mit Radius kleiner als Werkzeugradius)
N2550 G03 X3557.83 Y-577.61 I0.00 J0.60
(keine unmittelbare Konvertierung zu einem Linearsatz)
...
N3000 G141                (Aktivieren Konturausblenden)
...
(Kreiselement mit Radius kleiner des Werkzeugradius)
N3550 G03 X3557.83 Y-577.61 I0.00 J0.60
(unmittelbare Konvertierung zu einem Linearsatz)

...
N3600 G40                 (Abwahl der WRK)
```

Programmierbeispiel

Direkter Wechsel der Anwahlseite

```
N090 G90
N100 #TRC[REVERSE=1]
N110 G00 X0 Y0
N120 G01 X100 Y100
N130 G41 G01 X150    (Anwahl der WRK links der Kontur)
N140 G01 X250
N150 G01 Y200        (Umkehrpunkt)
```

```
N160 G42            (Wechsel der Anwahlseite, jetzt rechts der Kontur)
N170 G01 X100
N180 G01 X100 Y150
N160 G40 G01 X0
N170 G01 Y0
…
```

13.2.14.1 TRC Option STRETCH_FACTOR

STRETCH_FACTOR=..

Mit diesem Parameter kann bei spitz zulaufenden Außenecken die Umfahrungsstrategie beeinflusst werden. Der Wert des Parameters bestimmt die maximale Satzverlängerung. Die Länge ist abhängig vom verwendeten Werkzeugradius.

Der Parameter hat nur Einfluss auf Außenecken mit einem Übergangswinkel größer 270 Grad.

1-9: Faktor für die mögliche Satzverlängerung
Standardwert: 1.

Hinweis

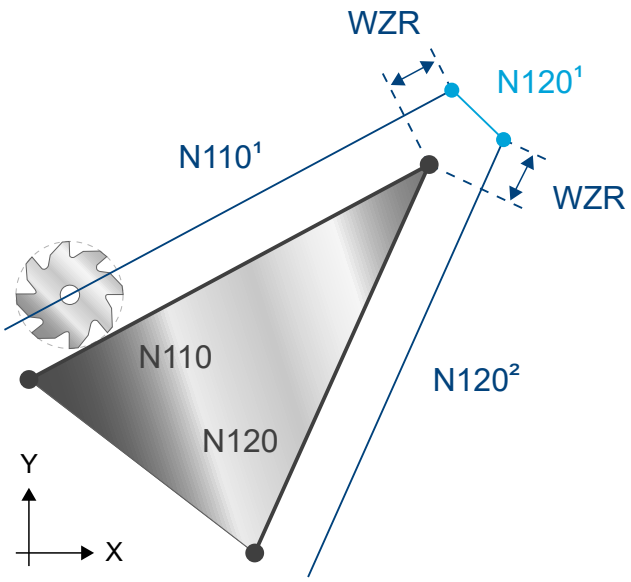
Wird der STRETCH_FACTOR kleiner 1 oder größer als 9 programmiert, so wird eine Warnung ID-22007 ausgegeben und der Standardwert 1 gesetzt.

Hinweis

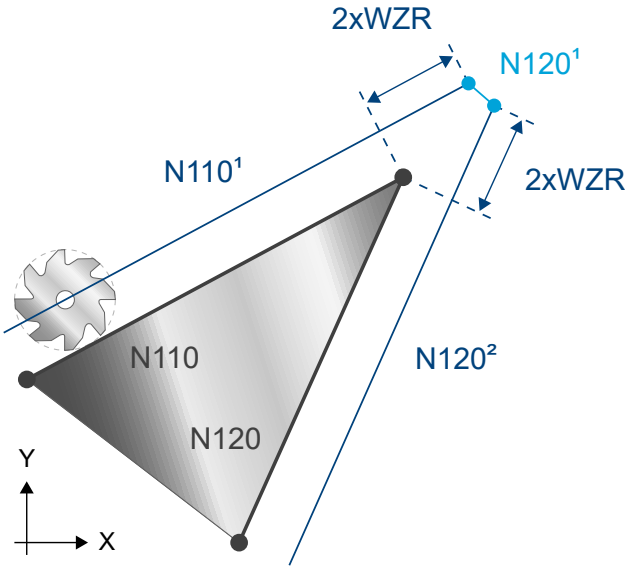
Bei sehr spitzen Außenecken und sehr groß gewähltem STRETCH_FACTOR kann die Verlängerung des jeweiligen Satzes sehr groß werden.

Extreme Satzverlängerungen sind möglich sowie mögliche Überschreitung des Softwareendschalters mit entsprechenden Fehlermeldungen (ID 120002/ ID120003)

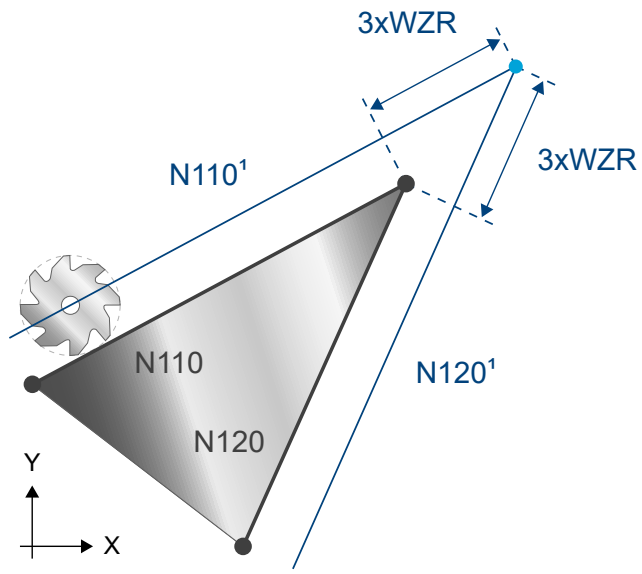
Das Verhalten der Option wird anhand der Satzübergänge Linear-Linear und Zirkular-Zirkular exemplarisch veranschaulicht. Die Funktionalität kann ebenfalls bei den Übergängen Linear-Zirkular sowie Zirkular-Linear eingesetzt werden.



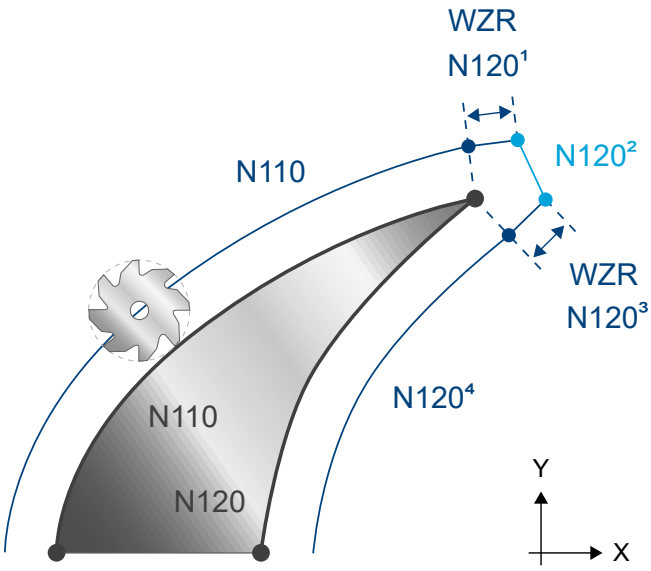

```
%stretch_test1.nc
N010 G139 G25
N020 V.G.WZ_AKT.R = 5
( Verwenden des Standardwerts )
N040 #TRC[STRETCH_FACTOR= 1 ]
N080 G0 X0 Y0 Z0
N090 G41 F10000
N100 G1 X10 Y20
N110 G1 X35 Y30
N120 G1 X30 Y20
N130 G1     Y10
N140 G40
N150 G1 X20 Y0
N160 G0 X0 Y0
N180 M30
```



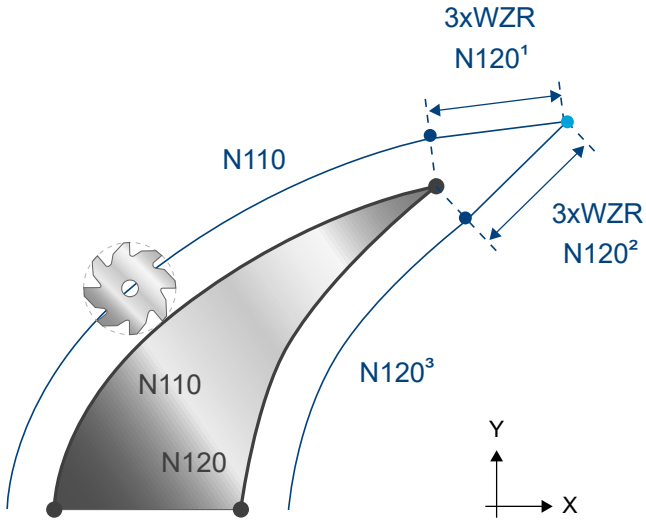

```
%stretch_test2.nc
N010 G139 G25
N020 V.G.WZ_AKT.R = 5

N040 #TRC[STRETCH_FACTOR=2 ]
N080 G0 X0 Y0 Z0
N090 G41 F10000
N100 G1 X10 Y20
N110 G1 X35 Y30
N120 G1 X30 Y20
N130 G1     Y10
N140 G40
N150 G1 X20 Y0
N160 G0 X0 Y0
N180 M30
```




```
%stretch_test3.nc
N010 G139 G25
N020 V.G.WZ_AKT.R = 5

N040 #TRC[STRETCH_FACTOR=3 ]
N080 G0 X0 Y0 Z0
N090 G41 F10000
N100 G1 X10 Y20
N110 G1 X35 Y30
N120 G1 X30 Y20
N130 G1     Y10
N140 G40
N150 G1 X20 Y0
N160 G0 X0 Y0
N180 M30
```




```
%stretch_test4.nc
N010 G139 G25
N020 V.G.WZ_AKT.R = 5
( Verwenden des Standardwerts )
N040 #TRC[STRETCH_FACTOR = 1 ]
N080 G0 X0 Y0 Z0
N090 G41 F10000
N100 G1 X10 Y20
N110 G2 X35 Y30 R50
N120 G3 X25 Y15 R40
N130 G1     Y10
N140 G40
N150 G1 X20 Y0
N160 G0 X0 Y0
N180 M30
```




```
%stretch_test4.nc
N010 G139 G25
N020 V.G.WZ_AKT.R = 5

N040 #TRC[STRETCH_FACTOR = 3 ]
N080 G0 X0 Y0 Z0
N090 G41 F10000
N100 G1 X10 Y20
N110 G2 X35 Y30 R50
N120 G3 X25 Y15 R40
N130 G1     Y10
N140 G40
N150 G1 X20 Y0
N160 G0 X0 Y0
N180 M30
```

Wird bei den vorliegenden Geometrien der STRETCH_FACTOR noch weiter vergrößert, dann ändert sich die Parallelbahn nicht. Der Eckpunkt, wie bei Faktor 3, wird verwendet.

13.2.14.2 TRC Option PERPENDICULAR_RADIUS_CHANGE

Mit dieser Option wird eine programmierte Änderung des Werkzeugradius unmittelbar durch Einfügen einer Bewegung orthogonal zur programmierten Kontur ausgefahren.

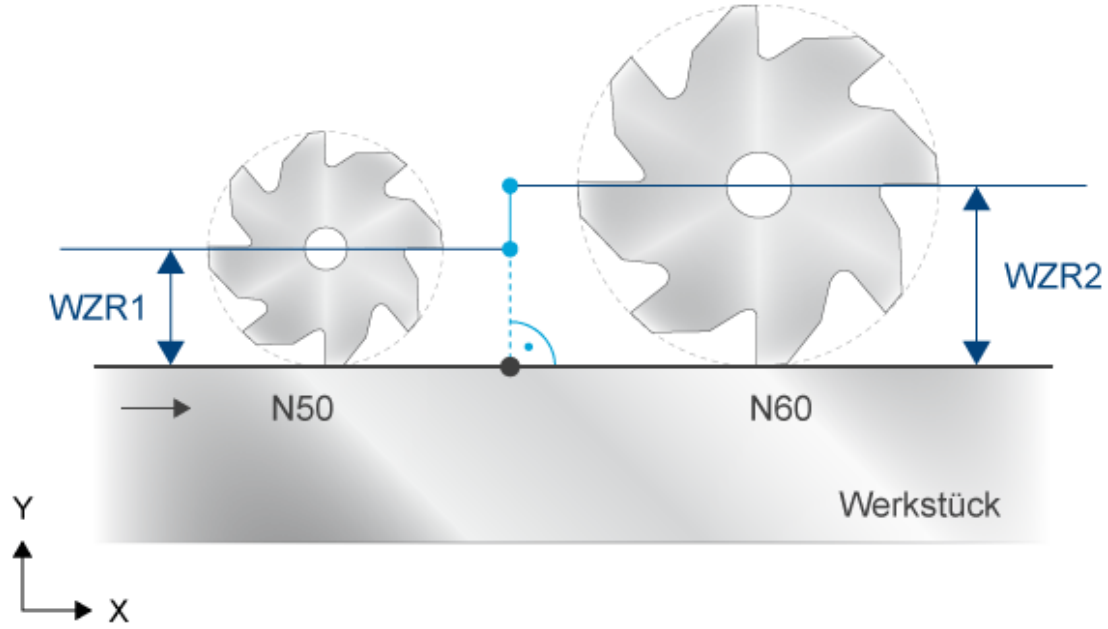


Abb. 155: Orthogonales Ausfahren der Änderung des Werkzeugradius

Syntax:

#TRC [[PERPENDICULAR_RADIUS_CHANGE=..]]

PERPENDICULAR_ RADIUS_CHANGE=.. Mit diesem Parameter kann bei Änderung des Werkzeugradius die Änderung senkrecht ausgefahren werden.

0: Kein senkrechtes Ausfahren des neuen Werkzeugradius (Standard)

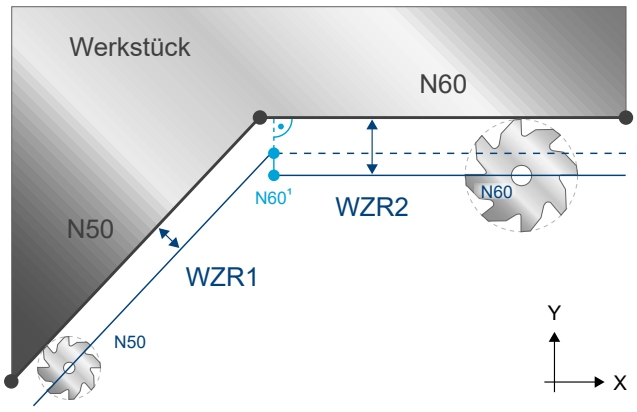
1: Senkrechtes Ausfahren des neuen Werkzeugradius

In den Programmbeispielen soll zwischen den programmierten Bewegungssätzen N50 und N60 der Werkzeugradius geändert werden.

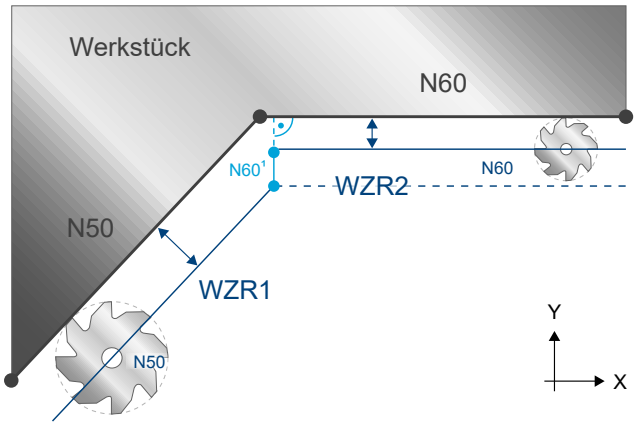
Um das Verhalten besser zu veranschaulichen, ist die Änderung des Werkzeugradius sehr groß gewählt. Üblicherweise handelt es sich bei der Änderung des Werkzeugradius um sehr kleine Korrekturen.

In den nachfolgenden Beispielen werden einige Satzübergänge exemplarisch dargestellt. Es sind alle Kombinationen aus Linear- und Zirkularsätzen zulässig.

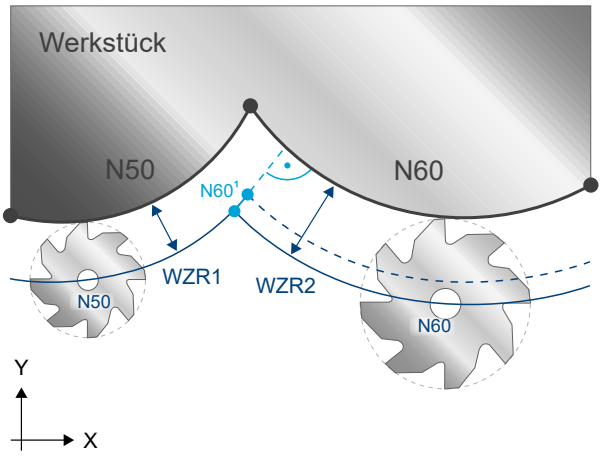
Änderung des Werkzeugradius im Inneneck



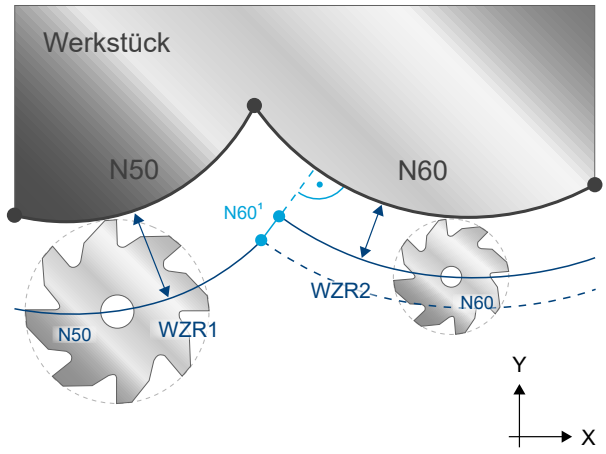
%Test.nc
N05 G0 X0 Y0 F1000
#TRC [**PERPENDICULAR_RADIUS_CHANGE=1**]
N10 **V.G.WZ_AKT.R= 5**
N20 G42
N40 G01 X20 Y0
N50 G01 X50 Y50
N55 **V.G.WZ_AKT.R = 6**
N60 G01 X100
N70 G40 X120 Y0
N99 M30



%Test.nc
N05 G0 X0 Y0 F1000
#TRC [**PERPENDICULAR_RADIUS_CHANGE=1**]
N10 **V.G.WZ_AKT.R= 6**
N20 G42
N40 G01 X20 Y0
N50 G01 X50 Y50
N55 **V.G.WZ_AKT.R = 5**
N60 G01 X100
N70 G40 X120 Y0
N99 M30

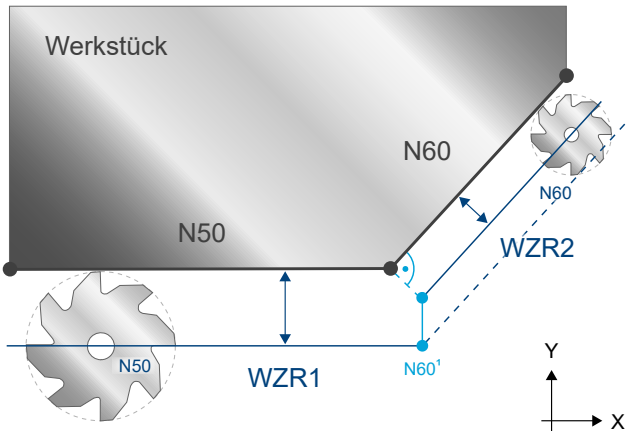


%Test.nc
N05 G0 X0 Y0 F1000
#TRC [**PERPENDICULAR_RADIUS_CHANGE=1**]
N10 **V.G.WZ_AKT.R= 5**
N20 G42
N30 G01 X10 Y20
N40 G01 X20
N50 G03 X50 Y25 R30
N55 **V.G.WZ_AKT.R = 6**
N60 G03 X90 Y18 R40
N70 G01 X100
N80 G40 X120 Y0
N99 M30

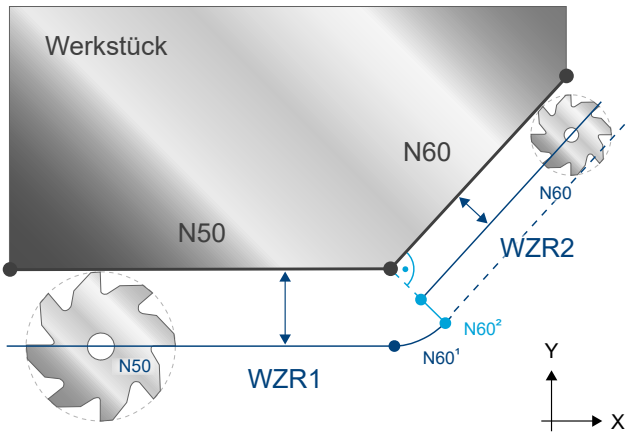



```
%Test.nc
N05 G0 X0 Y0 F1000
#TRC [PERPENDICULAR_RADIUS_CHANGE=1 ]
N10 V.G.WZ_AKT.R= 6
N20 G42
N30 G01 X10 Y20
N40 G01 X20
N50 G03 X50 Y25 R30
N55 V.G.WZ_AKT.R = 5
N60 G03 X90 Y18 R40
N70 G01 X100
N80 G40 X120 Y0
N99 M30
```

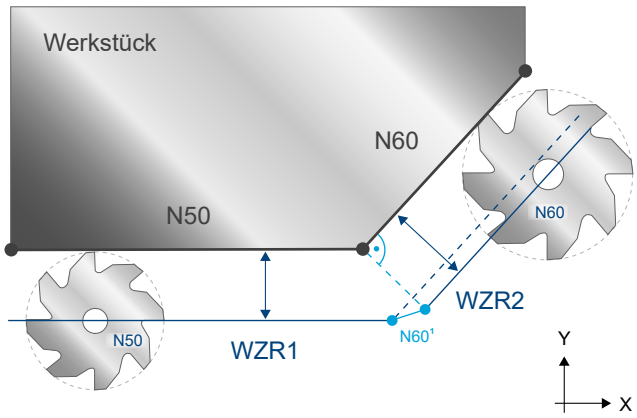
Änderung des Werkzeugradius am Außeneck



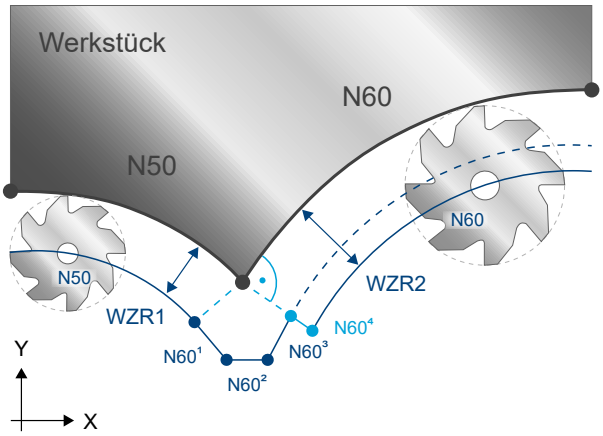

```
%Test.nc
N05 G0 X0 Y0 F1000
#TRC [PERPENDICULAR_RADIUS_CHANGE=1 ]
N10 V.G.WZ_AKT.R= 6
N20 G42 G25
N40 G01 X20 Y20
N50 G01 X50
N55 V.G.WZ_AKT.R = 5
N60 G01 X80 Y50
N70 G01 X120
N80 G40 X140 Y0
N99 M30
```



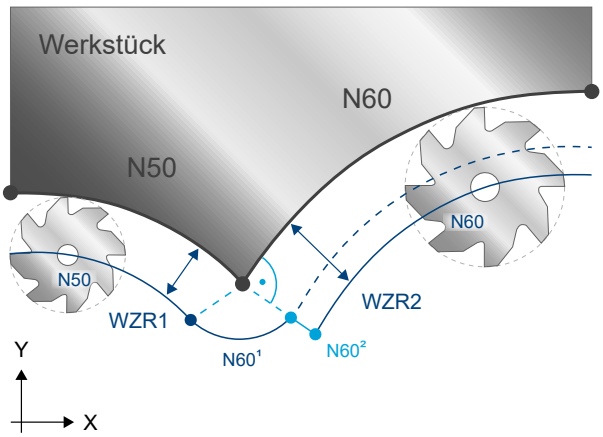

```
%Test.nc
N05 G0 X0 Y0 F1000
#TRC [PERPENDICULAR_RADIUS_CHANGE=1 ]
N10 V.G.WZ_AKT.R= 6
N20 G42 G26
N40 G01 X20 Y20
N50 G01 X50
N55 V.G.WZ_AKT.R = 5
N60 G01 X80 Y50
N70 G01 X120
N80 G40 X140 Y0
N99 M30
```



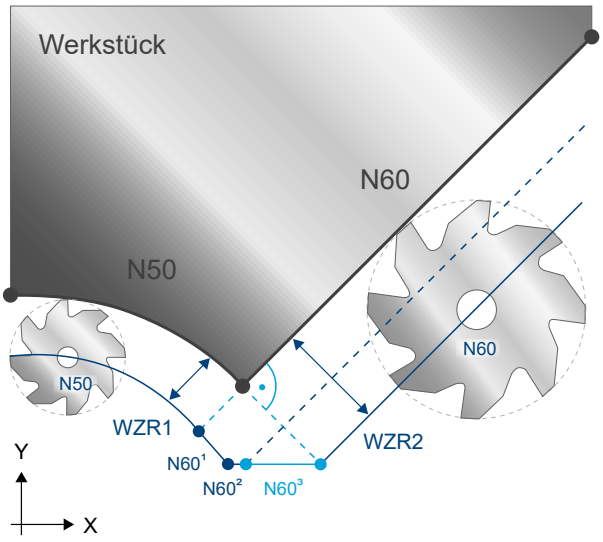

```
%Test.nc
N05 G0 X0 Y0 F1000
#TRC [PERPENDICULAR_RADIUS_CHANGE=1 ]
N10 V.G.WZ_AKT.R= 5
N20 G42 G25
N40 G01 X20 Y20
N50 G01 X50
N55 V.G.WZ_AKT.R = 6
N60 G01 X80 Y50
N70 G01 X120
N80 G40 X140 Y0
N99 M30
```



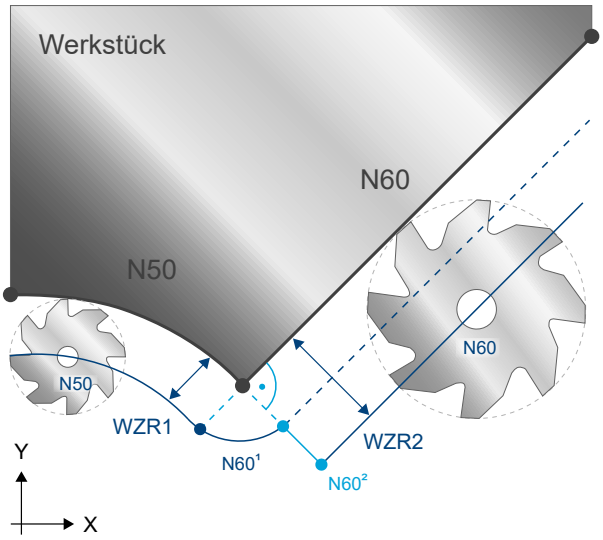

```
%Test.nc
N05 G0 X0 Y0 F1000
#TRC [PERPENDICULAR_RADIUS_CHANGE=1 ]
N10 V.G.WZ_AKT.R= 5
N20 G25
N30 G42 X10 Y10
N40 G01 X20
N50 G02 X60 Y0 R30
N55 V.G.WZ_AKT.R = 6
N60 G02 X90 Y12 R50
N70 G01 X100
N80 G40 X120 Y0
N99 M30
```




```
%Test.nc
N05 G0 X0 Y0 F1000
#TRC [PERPENDICULAR_RADIUS_CHANGE=1 ]
N10 V.G.WZ_AKT.R= 5
N20 G26
N30 G42 X10 Y10
N40 G01 X20
N50 G02 X60 Y0 R30
N55 V.G.WZ_AKT.R = 6
N60 G02 X90 Y12 R50
N70 G01 X100
N80 G40 X120 Y0
N99 M30
```




```
%Test.nc
N05 G0 X0 Y0 F1000
#TRC [PERPENDICULAR_RADIUS_CHANGE=1 ]
N10 V.G.WZ_AKT.R= 5
N20 G25
N30 G42 X10 Y10
N40 G01 X20
N50 G02 X60 Y0 R30
N55 V.G.WZ_AKT.R = 6
N60 G01X80 Y40
N70 G01 X100
N80 G40 X120 Y0
N99 M30
```




```
%Test.nc
N05 G0 X0 Y0 F1000
#TRC [PERPENDICULAR_RADIUS_CHANGE=1 ]
N10 V.G.WZ_AKT.R= 5
N20 G26
N30 G42 X10 Y10
N40 G01 X20
N50 G02 X60 Y0 R30
N55 V.G.WZ_AKT.R = 6
N60 G01X80 Y40
N70 G01 X100
N80 G40 X120 Y0
N99 M30
```

13.2.14.3 TRC Option SPLIT

Mit der Option SPLIT bzw. SPLIT_PATH können der An- und Abwahlsatz der WRK in mehrere Segmente mit jeweils eigenem Vorschub aufgeteilt werden.

Das Splitten wird wahlweise über den Weg einer Achse der Hauptebene oder den Bahnfahrweg festgelegt.

Ein Splitten ist bei An- oder Abwahl nur bei Übergangswinkeln kleiner 180 Grad möglich. Ist der Winkel größer als 180 Grad, so erfolgt keine Aufteilung des jeweiligen Satzes. Es wird dabei der programmierte Vorschub verwendet.

Versionshinweis

Diese Funktionalität ist verfügbar ab CNC-Version V3.1.3080.05.

Bei nachfolgenden Bedingungen ist kein Splitten möglich:

- Übergangswinkel größer 180 Grad
- Anwahlmodus G237, G238, G05
- Anwahlmodus G236 und Übergangswinkel größer als 90 Grad
- An- oder Abwahl mit Zirkularsatz
- Mehrpfadprogrammierung

Syntax:

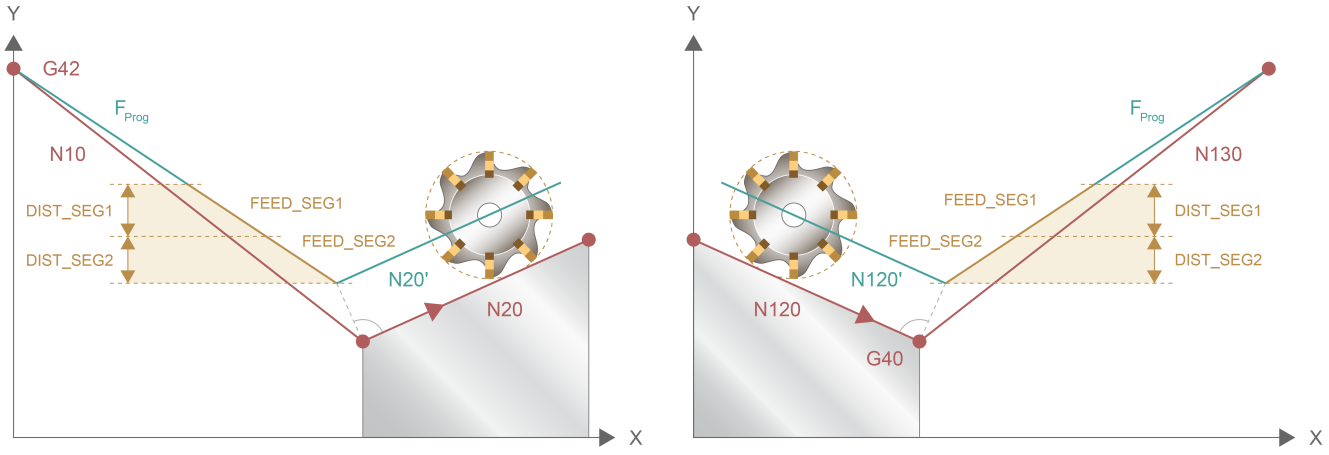
#TRC [SPLIT | **SPLIT_PATH** | **SPLIT_OFF** [**AX**=<*Achsname*> | **AXNR**=..] **COMBINED** | **POST** | **PRE**
DIST_SEG1=.. **FEED_SEG1**=.. [**DIST_SEG2**=.. **FEED_SEG2**=..] { \ } **]**

SPLIT	Splitten über Weg einer Achse
SPLIT_PATH	Splitten über Bahnfahrweg
SPLIT_OFF	Deaktivieren des Splittens
AX=<*Achsname*>	Angabe des Achsnamens bei Verwenden von SPLIT
AXNR=..	Angabe der Achsnummer bei Verwenden von SPLIT
COMBINED	Splitten bei An- und Abwahl der WRK
PRE	Splitten bei Anwahl der WRK
POST	Splitten bei Abwahl der WRK
DIST_SEG1=..	Längenangabe für erstes Segment in [mm]
FEED_SEG1=..	Vorschub für erstes Segment in [mm/min]
DIST_SEG2=..	Längenangabe für zweites zusätzliches Segment in [mm], optional
FEED_SEG2=..	Vorschub für zweites zusätzliches Segment in [mm/min] , optional
\	Trennzeichen ("Backslash") für übersichtliche Programmierung des Befehls über mehrere Zeilen

Splitten über programmierten Weg einer Achse

Splitten des An- und Abwahlsatzes der WRK über Achsvorgabe SPLIT und z.B. AX=Y

Anwahl **Abwahl**

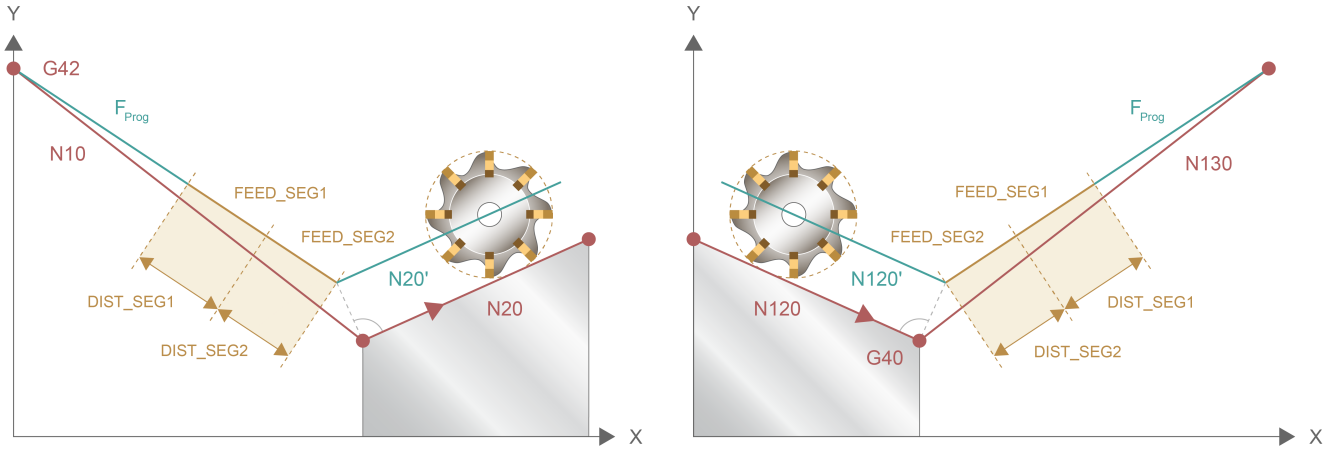


Splitten über programmierten Bahnfahrweg

Splitten des An- und Abwahlsatzes der WRK über Bahnfahrweg SPLIT_PATH

Anwahl

Abwahl



Verhalten bei kurzem Fahrweg

Ist bei Angabe von zwei Segmenten die Länge des Bahn- oder Achsfahrwegs kürzer als die Summe der Segmentlänge, so wird das 2.Segment, wie programmiert gesplittet und mit dem dafür vorgesehenen Vorschub versehen. Der verbleibende Fahrweg des An- oder Abfahrsatzes wird mit dem programmieren Vorschub des ersten Segments versehen.

Bei Angabe von nur einem Segment und zu kurzem Weg wird der gesamte Weg mit dem programmierten Vorschub des Segments gefahren.

Programmierbeispiel

Parametrierung der SPLIT-Option

```
;SPLIT mit Angabe des Achsnamens nur bei Anwahl PRE ist Standard
#TRC [SPLIT AX=X FEED_SEG1=1111 DIST_SEG1=100 FEED_SEG2=2222
DIST_SEG2=200]

;SPLIT mit Angabe der Achsnummer nur bei Abwahl POST
#TRC [SPLIT POST AXNR=2 FEED_SEG1=1000 DIST_SEG1=100 FEED_SEG2=2222
DIST_SEG2=2000]

;SPLIT mit Angabe der Achsnummer bei An- und Abwahl COMBINED
#TRC [SPLIT COMBINED AXNR=1 FEED_SEG1=1000 DIST_SEG1=100 FEED_SEG2=2222
DIST_SEG2=2000]

;Splitten über Angabe des Weges SPLIT_PATH nur bei Anwahl PRE
#TRC [SPLIT_PATH PRE FEED_SEG1=1111 DIST_SEG1=100 FEED_SEG2=2222
DIST_SEG2=200]

;Splitten über Angabe des Weges SPLIT_PATH bei An- und Abwahl COMBINED,
mit nur einem zusätzlichen Segment
#TRC [SPLIT_PATH COMBINED FEED_SEG1=1111 DIST_SEG1=100]

;Angabe des Splittens über mehrere Zeilen
#TRC [SPLIT_PATH COMBINED \
FEED_SEG1=1111 DIST_SEG1=100]

;Abwahl des Splitten
#TRC [SPLIT_OFF]
```

13.2.14.4 TRC Option G236_LIN

Diese Option ist nur wirksam bei Verwendung des Anwahlmodus G236 [▶ 548] und einem Übergangswinkel im An- oder Abwahlbereich der Werkzeugradiuskorrektur zwischen 90° und 180°.

Syntax:

#TRC [[G236_LIN =..]]

G236_LIN=.. Festlegung ob im beschriebenen Winkelbereich 90° bis 180° ein Kreis- oder Linearsatz eingefügt wird.

0: Einfügen eines Kreissatzes (Standard)

1: Einfügen eines Linearsatz

13.2.14.5 TRC Option TANGENTIAL_LIMIT

Option ist verfügbar ab CNC-Version V2.11.2835

Mit diesem Parameter kann die Wirkungsweise von eingefügten Bewegungssätzen durch die WRK bei fast tangentialen Übergängen beeinflusst werden. Eingefügte Sätze der WRK werden bis zum Übergangswinkel (180 Grad + TANGENTIAL_LIMIT) als tangential betrachtet.

Die Reduzierung der Grenze führt an den entsprechenden Übergängen zu stärkerem Bremsverhalten. Bei Verwendung von Überschleifverfahren, z.B. Polynomüberschleifen (G261) [▶ 131], wird eine Reduzierung der Grenze nicht empfohlen.

Syntax:

#TRC [[TANGENTIAL_LIMIT=..]]

TANGENTIAL_LIMIT=.. Festlegen der additiven Grenze für tangentiale Übergangsbetrachtung.

Wertebereich: 0.1 bis 3 Grad

Standardwert ist 3

```
;Übergangswinkel bis 181,5 Grad als tangentiale Übergänge betrachtet
#TRC[TANGENTIAL_LIMIT=1.5]
```

13.2.14.6 TRC Optionen für Online-TRC und 2-Pfad

Online-TRC

Mit den nachfolgenden Parametern kann Einfluss auf die Online-Werkzeugradiuskorrektur genommen werden, diese kann nur gesetzt werden mit aktiver Werkzeugradiuskorrektur (G41/ G42).

Mit dem Parameter INVERSE kann die, durch die Werkzeugradiuskorrektur entstandene, verschobene Parallelbahn auf die ursprünglich programmierte Bahn zurückgerechnet werden.

Der Parameter ONLINE ermöglicht bei gesetztem Parameter INVERSE Einstellungen für die Online-Werkzeugradiuskorrektur. Diese Online-Werkzeugradiuskorrektur kann über ein TcCOM-Objekt realisiert werden.

Achtung

Der Parameter ONLINE darf nur auf einen Wert ungleich 0 gesetzt werden, wenn der Parameter INVERSE aktiv ist. Bei inaktivem INVERSE wird der Fehler ID 22125 ausgegeben.

Syntax:

#TRC [[INVERSE=..] [ONLINE=..] [ONLINE_BY_VECTOR=..]]

INVERSE=.. Rückrechnen der durch die Werkzeugradiuskorrektur entstandenen Parallelbahn.

0: kein Rückrechnen (Standard).

1: Rückrechnen der Parallelbahn.

ONLINE=.. Einstellung der Online-Werkzeugradiuskorrektur:

0: Keine Berechnung der Online-Werkzeugradiuskorrektur.

1: Berechnung der kompensierten Bahn via TcCOM-Schnittstelle.

2: Einfache Berechnung der Parallelbahn für jede Eben.

3: Berechnung der Parallelbahn unter Berücksichtigung der Werkzeugorientierung.

ONLINE_BY_VECTOR=.. Dieser Parameter bestimmt die Definition der Werkzeugorientierung

0: Die Werkzeugorientierung wird durch die Position des oberen und unteren Pfads gegeben.

1: Die Werkzeugorientierung wird durch die normierten Vektoren gegeben.

Basisradius für Werkzeugradiuskorrektur

Mit dieser Option kann festgelegt werden, ob die Werkzeugradiuskorrektur für die einzelnen Pfade mit dem individuell für diesen Pfad festgelegten Werkzeugradius die Parallelbahn berechnen soll, oder mit dem Werkzeugradius des Pfades, der über diese Option bestimmt werden kann.

Syntax:

#TRC [[MULTI_PATH_RADIUS=..]]

MULTI_PATH_RADIUS=.. Mit diesem Parameter kann festgelegt werden, mit welchem Werkzeugradius die statische Werkzeugradiuskorrektur die Parallelbahn berechnen soll.

INDIVIDUAL/ DEFAULT	Die Werkzeugradien, die in dem jeweiligen Pfad programmiert sind, werden verwendet. (Standard)
REFERENCE	Der Werkzeugradius des Referenzpfads wird verwendet.
SECONDARY	Der Werkzeugradius des zweiten Pfads wird verwendet.

Dynamische Offsetüberlagerung

Die dynamische Offsetüberlagerung ist eine anwenderspezifische Überlagerung der TCP Bahn, die in der Online-TRC wirkt.

Die überlagerten Achsbewegungen können deshalb nur eingeschränkt bei der Planung der maximalen Geschwindigkeit und Beschleunigung in der CNC berücksichtigt werden.

Speziell an Innenecken können die Überlagerungen zu Beschleunigungsüberschreitungen führen. Um dies zu verhindern, analysiert die CNC die Tangentendrehung an Innenecken und bestimmt daraus die maximal zulässige Dynamik.

Die An/Abwahl der Funktionalität in der Online-TRC wird über die SPS geschaltet. Die Berechnung der zulässigen Dynamik bei der Offsetüberlagerung an Innenecken wird mit nachfolgendem Programmierbefehl aktiviert:

Syntax:

#TRC [[DYNAMIC_VARIATION_MAX_OFFSET=..]]

DYNAMIC_VARIATION_MAX_OFFSET =..	Definition des maximalen dynamischen Offsets in [0,1µm]] 0: Abwahl Dynamische Offsetüberlagerung (Standard) > 0: Anwahl, maximaler Offset wird für Dynamikberechnung verwendet.

TAPERLINK

Die Option TAPERLINK ermöglicht bei einer 2-Pfadkonfiguration die Synchronisation zwischen dem Referenzpfad und dem 2. Pfad um die programmierte Drahtneigung zu erhalten. Siehe [FCT-C49, Kapitel: Beschreibung].

Voraussetzung für die Funktionalität ist eine 2-Pfadkonfiguration [▶ 860] und die Anwahl der Werkzeugradiuskorrektur mit G41 oder G42.

Syntax:

#TRC [[TAPERLINK=..]]

TAPERLINK =..	Modus für Funktionalität Taper-Link festlegen. 0: Taper-Link Funktionalität inaktiv (Standard). 1: Taper-Link aktiv: Kompensation ist auf beiden Pfaden aktiv, automatische Erkennung. 2: Taper-Link aktiv: Referenzpfad kompensiert den 2. Pfad. 3: Taper-Link aktiv: 2. Pfad kompensiert den Referenzpfad.

13.2.14.7 TRC Option GEN_CIR_BLOCK_IN_CORNER

Mit dieser Option kann bei aktiver Werkzeugradiuskorrektur in sogenannten Innenecken ein virtueller Kreis (Radius 0) integriert werden. Innenecken sind Übergänge zwischen Bewegungssätzen, deren Überganswinkel kleiner als 180° ist.

Nur möglich mit einem Werkzeugradius ungleich 0.

Syntax:

#TRC [[GEN_CIR_BLOCK_IN_CORNER=..]]

GEN_CIR_BLOCK_IN_CORNER=.. Einfügen von virtuellen Kreissätzen mit Radius 0 bei Innenecken der Werkzeugradiuskorrektur.

0: Kein Einfügen von virtuellen Kreissätzen (Standard)

1: Einfügen von virtuellen Kreissätzen

13.2.14.8 TRC Option SUPPRESS_TRC

Dieser Parameter ermöglicht das Unterdrücken der Werkzeugradiuskorrektur.

Die Kerbberechnung über #TRC[KERF_MASKING=1] findet statt, es wird nur die Berechnung der Parallelbahn unterdrückt.

Syntax:

#TRC [[SUPPRESS_TRC =..]]

SUPPRESS_TRC =.. Unterdrücken der Werkzeugradiuskorrektur

0 : WRK Unterdrückung inaktiv (Standard)

1 : WRK Unterdrückung aktiv

13.2.15 Ausschlussliste von Befehlen bei aktiver WRK/SRK

Nachfolgend eine Liste der Befehle, die bei aktiver WRK/SRK (G41/G42) nicht zulässig sind:

(Es handelt sich hierbei um den Bereich im NC-Programm zwischen G41/G42 und G40.)

Hinweis

Fehler 20651 wird bei nachfolgenden Befehlen zwischen G41/G42 und G40 ausgegeben.

Hinweis

Bei Verwendung der WRK-Modi G139/G138/G236/G05 ist nach G40 ein zusätzlicher Bewegungssatz nötig um den Werkzeugradius wieder abzubauen.

Werden nachfolgende Befehle **unmittelbar** nach G40 verwendet, so wird der **Fehler 90050** ausgegeben.

G-Befehl	Bemerkung

G200/ G201	
G303	Kreise im Raum

Ein Lesezugriff von synchronen V.E.-Variablen ist ebenfalls nicht zulässig.

Zusatz-Befehl	**Bemerkung**
#AX DEF	
#AX DEF DEFAULT	
#AX LINK ON/OFF/OFF ALL	
#AX REQUEST	
#AX RELEASE	Abgeben von Achsen. Achsen, die nicht in der aktiven Ebene sind, dürfen abgegeben werden.
#AX RELEASE ALL	
#CALL AX	
#CAX	
#CAX OFF	
#CHANNEL INIT	
#CLEAR CONFIG	
#CS ON/OFF	
#CS MODE ON/OFF	
#CYL	
#CYL OFF	
#CYL ORI LATERAL	
#CYL ORI PROFILE	
#FACE	
#FACE OFF	
#FLUSH	
#FLUSH CONTINUE	Ausgabe wird in CNC verschoben, ist aber zulässig.
#FLUSH WAIT	
#GET CMDPOS	

#GET ACTPOS	
#GET MANUAL OFFSETS	
#LOAD CONFIG	
#MCS ON/OFF	
#MCS TO WCS	
#OTC OFF	
#PRESET	
#PTP ON/OFF	
#PUT AX	Abgeben von Achsen. Achsen, die nicht in der aktiven Ebene sind, dürfen abgegeben werden.
#PUT AX ALL	
#SET AX	
#SET AX LINK	
#TRACK CS ABS/ON/OFF	
#TRAFO ON / OFF	
#WCS TO MCS	

14 Variablen und Variablenrechnung

Eine vollständige Liste der Variablenprogrammierung findet sich in der Befehlsübersicht unter Variablenprogrammierung (V.) [▶ 891].

Unter Variablen sind hier zum einen interne Daten des Decoders mit einer festen Namenszuweisung zu verstehen, zum anderen selbst zu definierende Variablen, deren Bezeichnung im Wesentlichen frei wählbar ist. Mit Ausnahme der externen Variablen (V.E...) ist deren Gültigkeit und Verwendbarkeit ausschließlich auf den jeweiligen NC-Kanal begrenzt.

`Allgemeine Syntax:`

V.<*NAME_1*>**.**<*NAME_2*>**.**<*NAME_3*>.{<*NAME_n*>.}

V.	zeigt den Variablenzugriff an	
<*NAME_1*>.	Bestimmt die globale Datenbezeichnung:	
	"A."	steht für achsspezifische Variablen,
	"SPDL." "SPDL_PROG."	steht für spindelspezifische Variablen,
	"G."	steht für achsübergreifende, im Kanal global gültige Variablen,
	"E."	steht für externe Variablen,
	"P."	steht für eigendefinierte, bis Hauptprogrammende (M2, M30) gültige Variablen,
	"S."	steht für eigendefinierte, Hauptprogramm übergreifende Variablen,
	"L."	steht für eigendefinierte, lokale Variablen, gültig bis aktuelle Programmebene durch Rücksprung (M17, M29) verlassen wird,
	"CYC."	steht für Variablen, die nur innerhalb von Zyklenprogrammen (L CYCLE) verwendet werden dürfen.
<*NAME_2*>.	gibt den Datennamen an	
<*NAME_3*>.	gibt z.B. den Index an, wenn mehrere gleiche Daten zu unterscheiden sind.	

Hinweis

Programmierung der Achskennung

Die letzte Kennung der achsspezifischen und einiger gruppenspezifischen Variablen stellt die Achskennung dar. Dabei sind die Bezeichnungen

".X" bzw. "[0]",

".Y" bzw. "[1]",

".Z" bzw. "[2]"

jeweils wahlweise zu verwenden, wenn im Kanalparametersatz der Achse mit dem Index 0 der Name X, der Achse mit dem Index 1 der Name Y und der Achse mit dem Index 2 der Name Z zugewiesen ist.

Beispiel

Absoluter Wert der X-Achse:

V.A.ABS.**X** oder V.A.ABS**[0]**

Analog sind bei spindelspezifischen Variablen die Spindelnamen bzw. entsprechenden Indizes gemäß dem Kanalparametersatz zu verwenden.

".S" bzw. "[0]",

".S2" bzw. "[1]", usw.

Beispiel

Logische Achsnummer der S-Spindel:

Logische Achsnummer der **S**-Spindel:

V.SPDL.LOG_AX_NR.**S** oder V.SPDL.LOG_AX_NR**[0]**

Programmierbeispiel

Die Zeilen N20/N30 bewirken jeweils eine Geradeninterpolation in X-Richtung um den Wert der Variablen V.A.BZP.Y, also der Bezugspunktverschiebung in Y-Richtung.

```
N10 G92 X0 Y40 Z0                       ;Bezugspunktverschiebung
N20 G91 G01 F1000 XV.A.BZP.Y
N30 XV.A.BZP[1]                         ;hier: Achsindex 1 == Y
N40 M30
```

Von allen Variablen kann der Inhalt gelesen werden, einigen kann auch ein Wert zugewiesen werden. Die Zugriffsart ist für jede Variable fest vorgegeben; im allgemeinen ist nur ein lesender Zugriff zugelassen, da bei den meisten Variablen ein schreibender Zugriff nicht sinnvoll ist.

Programmierbeispiel

Hier wird dem 2. Nullpunktverschiebungsvektor für die Achse mit dem Index 1 der Wert 100 zugewiesen:

```
N10  V.G.NP[2].V[1] = 100
...
```

Mit der EXIST-Funktion (siehe Kapitel Arithmetische Ausdrücke <expr> [▶ 32]) kann geprüft werden, ob eine Variable überhaupt verfügbar ist.

Programmierbeispiel

Durch die EXIST-Abfrage auf eine achspezifische Variable wird geprüft, ob eine bestimmte Achse überhaupt im NC-Kanal vorhanden ist:

```
…
N10 G90 Y0
N20 $IF EXIST[V.A.LOG_AX_NR.X] == TRUE
N30 X-10                    ;X-Achse ist im Kanal, Position -10 anfahren
N40 $ELSE
N50 #CALL AX [X,1,0]        ;X-Achse nicht im Kanal, zuerst anfordern
N60 $ENDIF
…
M30
```

Vor dem Zugriff auf eine externe Variable wird zunächst geprüft, ob überhaupt zugegriffen werden kann:

```
…
N10 G90 Y0
N20 $IF EXIST[V.E.POS_1] == TRUE
N30 XV.E.POS_1                          ;Fahre X-Achse auf Position POS_1
N40 $ELSE
N50 #MSG ["V.E.POS_1 nicht vorhanden!"]              ;Meldung ausgeben
N55 M0                                               ;Anhalten
N60 $ENDIF
…
M30
```

14.1 Achsspezifische Variablen (V.A.)

Die Kennung für achsspezifische Variablen ist "V.A. ...". Sie kann in 2 Varianten angegeben werden:

1: Achsnamen gemäß Kanalliste (Im Folgenden exemplarisch mit "X" dargestellt)

2: Achsindex [i] gemäß Kanalliste mit <i>: 0…31

Beispiel: V.A.ABS.X oder V.A.ABS[0]

Achtung

V.A.-Variablen können für Linear- und Rundachsen, jedoch nicht für Spindeln, programmiert werden.

Hinweis

Ein lesender Zugriff auf die Variablen mit der Kennzeichnung L_{Flush} bewirkt ein Leeren des NC-Kanals.

Ein Leeren des NC-Kanals [▶ 341] kann z.B. bei aktiver Werkzeugradiuskorrektur [▶ 506] (G41/ G42) zum Fehler ID 20651 führen.

V.A.<var_name>	Bedeutung	Datentyp	Einheit der Ein/ Ausgabe	Erlaubter Zugriff Lesen / Schreiben
MENT.X	Mentale Koordinate des vorhergehenden NC-Satzes (siehe Kapitel Spiegeln G20-G23 [▶ 119])	Real	[mm, inch]	L
PROG.X	Programmierte Koordinate des vorhergehenden NC-Satzes. Bei aktiver Konturrotation (#ROTATION) liefert Variable den auf die Maschinenachsen abgebildeten Koordinatenwert.	Real	[mm, inch]	L
ABS.X	Absolute Koordinate des vorhergehenden NC-Satzes bzw. aktuelle absolute Koordinate nach NC-Befehl #CHANNEL INIT [▶ 179] jeweils im momentan aktiven Koordinatensystem	Real	[mm, inch]	L
ACT_POS.X	Aktuelle Achsposition im momentanen Koordinatensystem ohne Verschiebungen.	Real	[mm, inch]	L
ACS.ABS.X	Aktuelle absolute Achsposition, bei aktiver Transformation abgebildet auf die Maschinenachse	Real	[mm, inch]	L
-SWE.X	Aktuell wirksamer negativer Softwareendschalter	Real	[mm, inch]	L
+SWE.X	Aktuell wirksamer positiver Softwareendschalter	Real	[mm, inch]	L

-SWE_MDS.X	Konfigurierter negativer Softwareendschalter (gemäß P-AXIS-00177)	Real	[mm, inch]	L
+SWE_MDS.X	Konfigurierter positiver Softwareendschalter (gemäß P-AXIS-00178)	Real	[mm, inch]	L
REF.X	Position Referenzpunkt (wird erst nach erfolgter RPF belegt)	Real	[mm, inch]	L
BZP.X	Bezugspunktverschiebung	Real	[mm, inch]	L
PZV.X	Platzversatz	Real	[mm, inch]	L
MESS.X	Liefert nach erfolgter Messfahrt den achsspezifischen Messwert in dem Koordinatensystem, in dem gemessen wurde. Im Wert sind immer alle Verschiebungen eingerechnet Bei 2,5D: ACS-Werte bzw. bei CS / TRAFO: PCS-Werte	Real	[mm, inch]	L

Versionshinweis

Ab Version **V2.11.2020.07** ergänzen die achsspezifischen Variablen **V.A.MEAS.ACS.VALUE** und **V.A.MEAS.PCS.VALUE** die Variable **V.A.MESS**. Die zusätzlichen Variablen liefern den Messwert sowohl im Achskoordinatensystem inklusive aller Verschiebungen als auch den Messwert im Programmierkoordinatensystem ohne Verschiebungen.

MEAS.ACS.VALUE.X	Liefert nach erfolgter Messfahrt den achsspezifischen Messwert im Achskoordinatensystem (ACS). Im Wert sind alle Verschiebungen eingerechnet	Real	[mm, inch]	L
MEAS.PCS.VALUE.X	Liefert nach erfolgter Messfahrt den achsspezifischen Messwert im Programmierkoordinatensystem (PCS). Im Wert sind keine Verschiebungen eingerechnet	Real	[mm, inch]	L
MOFFS.X	Messoffset	Real	[mm, inch]	L
MERF.X	Messfahrt erfolgt? Wenn ja, dann 1	Boolean	0 , 1	L
MEIN.X	Eingerechneter Messoffset	Real	[mm, inch]	L
RERF.X	Referenzpunktfahrt erfolgt? Wenn ja, dann 1	Boolean	0 , 1	L
MANUAL_OFFSETS.X oder SOFFS.X	Verfahrweg während Handbetrieb. Nur sinnvoll in Verbindung mit NC-Befehl #GET MANUAL OFFSETS [▷ 177].	Real	[mm, inch]	L
MODE.X	Aktueller Achsmode	Integer	-	L
MODULO_VALUE.X	Modulobereich	Real	[°]	L

LOG_AX_NR.X	Logische Achsnummer einer Achse	Integer	-	L
AX_LIST_NAME.X	Konfigurierter Achsname (gemäß P-AXIS-00297)	String	-	L
AXIS_DEACTIVATED.X	Variable zeigt an, ob die Achse über das HLI geparkt wurde. **[ab V2.11.2813.00]**	Boolean	-	L_{Flush}
ENCODER2_VALUE.X	Aktueller Wert eines zweiten Encoders (optional) im Antrieb	Real	[mm, inch]	L_{Flush}
MIRROR.X	Spiegelstatus der Achse (1: keine Spiegelung -1: Spiegelung)	Integer	-	L
WCS.X MCS.X	Umrechnung zwischen Maschinenkoordinaten(MCS) und Werkstückkoordinaten (WCS). Nur sinnvoll in Verbindung mit den NC-Befehlen #WCS TO MCS [▶ 774] und #MCS TO WCS [▶ 774]	Real	[mm, inch]	L/S
DIAMETER_PROG.ABS.X	Liefert den Wert von P-AXIS-00058, wenn gilt: - Durchmesserprogrammierung (G51) ist aktiv - Achse ist als Plandrehachse konfiguriert **[ab V2.11.2051.00]**	Boolean	0 , 1	L
DIAMETER_PROG.REL.X	Liefert den Wert von P-AXIS-00059, wenn gilt: - Durchmesserprogrammierung (G51) ist aktiv - Achse ist als Plandrehachse konfiguriert **[ab V2.11.2051.00]**	Boolean	0 , 1	L
GEAR_LINK_ACTIVE.X	Ist Achse an einer programmierbaren Achskopplung (#GEAR LINK ON […] [▶ 491]) im Kanal beteiligt? Wenn ja, dann 1	Boolean	0 , 1	L_{Flush}
ANTR_TYP.X	Konfigurierter Antriebstyp einer Achse (gemäß P-AXIS-00020)	Integer	-	L
TRANSFORM.X	Liefert die berechnete Koordinate einer Achse im Ziel-CS in Verbindung mit #TRANSFORM [▶ 788]	Real	[mm, inch]	L
TORQUE_NOM.X	Liefert den Wert von P-AXIS-00392. Notwendig zur Umrechnung von Drehmomenten oder Kräften (bei Linearantrieben) ins Antriebsformat in Verbindung mit #DRIVE WR [▶ 465]	Real	[Nm, N]	L

CROSS_COMP_IN-IT.X	Kreuzkompensation für die Achse initialisiert? Wenn ja, dann 1	Boo-lean	0 , 1	L_{Flush}
PLANE_COMP_INIT.X	Flächenkompensation für die Achse initiali-siert? Wenn ja, dann 1	Boo-lean	0 , 1	L_{Flush}
LEAD_COMP_INIT.X	Spindelsteigungsfehlerkompensation für die Achse? Wenn ja, dann 1	Boo-lean	0 , 1	L_{Flush}
TEMP_COMP_INIT.X	Temperaturkompensation für die Achse initia-lisiert? Wenn ja, dann 1	Boo-lean	0 , 1	L_{Flush}
FRICT_COMP_INIT.X	Reibungskompensation für die Achse initiali-siert? Wenn ja, dann 1	Boo-lean	0, 1	L_{Flush}
CROSSTALK_COMP_INIT.X	Nickkompensation für die Achse initialisiert? Wenn ja, dann 1	Boo-lean	0, 1	L_{Flush}

CROSS_COMP_ACTI-VE.X	Kreuzkompensation für die Achse aktiv? Wenn ja, dann 1	Boo-lean	0 , 1	L_{Flush}
PLANE_COMP_ACTI-VE.X	Flächenkompensation für die Achse aktiv? Wenn ja, dann 1	Boo-lean	0 , 1	L_{Flush}
LEAD_COMP_ACTI-VE.X	Spindelsteigungsfehlerkompensation für die Achse aktiv? Wenn ja, dann 1	Boo-lean	0 , 1	L_{Flush}
TEMP_COMP_ACTI-VE.X	Temperaturkompensation für die Achse ak-tiv? Wenn ja, dann 1	Boo-lean	0 , 1	L_{Flush}
FRICT_COMP_ACTI-VE.X	Reibungskompensation für die Achse aktiv? Wenn ja, dann 1	Boo-lean	0, 1	L_{Flush}
CROSSTALK_COMP_ACTIVE.X	Nickkompensation für die Achse aktiv? Wenn ja, dann 1	Boo-lean	0, 1	L_{Flush}
BACK-LASH_COMP_ACTI-VE.X	Losekompensation für die Achse aktiv? Wenn ja, dann 1 **[ab V3.1.3081.05]**	Boo-lean	0, 1	L_{Flush}

Ab CNC-Version **V2.11.2810** sind folgende V.A.-Variablen der aktuellen Korrekturwerte verfügbar.				
LEAD_COMP_CURR.X	Aktueller Korrekturwert der SSFK für die Achse	Real	[mm, inch]	L_{Flush}
CROSS_COMP_CURR.X	Aktueller Korrekturwert der Kreuzkompensation für die Achse	Real	[mm, inch]	L_{Flush}
PLANE_COMP_CURR.X	Aktueller Korrekturwert der Flächenkompensation für die Achse	Real	[mm, inch]	L_{Flush}
TEMP_COMP_CURR.X	Aktueller Korrekturwert der Temperaturkompensation für die Achse	Real	[mm, inch]	L_{Flush}
FRICT_COMP_CURR.X	Aktueller Korrekturwert der Reibungskompensation für die Achse	Real	[mm, inch]	L_{Flush}
CROSSTALK_COMP_CURR.X	Aktueller Korrekturwert der Nickkompensation für die Achse	Real	[mm, inch]	L_{Flush}
BACKLASH_COMP_CURR.X	Aktueller Korrekturwert der Losekompensation für die Achse **[ab V3.1.3081.05]**	Real	[mm, inch]	L_{Flush}

Bei Verwendung von Strings zur Achsbezeichnung (z.B. X_SCHLITTEN, s. a. Beschreibung Achsbefehle [▶ 45]) sind diese Achsnamen zur Kennzeichnung der Variablen zu verwenden.

Beispiel: V.A.MENT.X_SCHLITTEN

Programmierbeispiel

```
N10 G90 G92 X50
N20 G100 X100                          ;Messfahrt, Messinterrupt 2mm vor
Ziel
N30 G90 G92 X0
N40 XV.A.MeESS.X YV.A.MOFFS.X          ;X auf 148 (98+50) auf 2
oder
N40 XV.A.MEAS.ACS.VALUE.X YV.A.MOFFS.X ;X auf 148 (98+50), Y auf 2
N50 XV.A.MEAS.PCS.VALUE.X              ;X au 98
```

14.2 Spindelspezifische Variablen (V.SPDL., V.SPDL_PROG.)

Die Kennung für Variablen, die den Zugriff auf konfigurationsspezifische Spindeldaten ermöglichen, ist "V.SPDL. ...".

Für Spindeldaten, die durch Programmierung belegt wurden, stehen Variablen mit der Kennung "V.SPDL_PROG. …" zur Verfügung.

V.SPDL.<name>	**Bedeutung**	**Datentyp**	**Einheit der Ein/ Ausgabe**	**Erlaubter Zugriff** **L**esen / **S**chreiben
LOG_AX_NR.S	Logische Achsnummer der Spindel	Integer	-	L
PLC_CONTROL.S	Ist Spindel eine PLC-Spindel? Wenn ja, dann 1	Boolean	0 , 1	L
NBR_IN_CHANNEL	Anzahl der im NC-Kanal verfügbaren Spindeln	Integer	-	L
M_FCT_FREE	Wie ist die Klassifizierung der M-Funktion M3, M4, M5, M19? Festlegung explizit als Spindel-M-Funktionen: 0 Frei verfügbar für andere Technofunktionen: 1	Boolean	0 , 1	L/S

V.SPDL_PROG.<name>	**Bedeutung**	**Datentyp**	**Einheit der Ein/ Ausgabe**	**Erlaubter Zugriff** **L**esen / **S**chreiben
SPEED.S	Aktuelle Drehzahl S.. der Spindel	Real	[U/min]	L
MOVE_CMD.S	Aktuelle Bewegungsart der Spindel: Für M3: 3, Für M4: 4, Für M5: 5	Integer	-	L
POSITION.S	Aktuell gesetzte Position S.POS.. der Spindel für M19	Integer	[°]	L
MAX_SPEED.S	Maximale Drehzahl G196 S.. der Spindel bei G96. Nur für Hauptspindel!	Real	[U/min]	L
CONST_CUT_SPEED.S	Konstante Schnittgeschwindigkeit G96 S.. bei Drehbearbeitung. Nur für Hauptspindel!	Real	[m/min, ft/min *]	L
DWELL_ROT_COUNT.S	Verweilzeit G04 S.. in Anzahl Spindelumdrehungen. Nur für Hauptspindel!	Real	[U]	L
GEAR_DATA_STEP.S	Aktuell gesetzte Getriebestufe G112 S.. Nur für Hauptspindel!	Integer	-	L

* [ab V2.11.2032.08 bei G70 und P-CHAN-00360 = 1]

Achtung

Schreibzugriff bewirkt bleibende Änderung in internen Kanalparametern (P-CHAN-00098)

Programmierbeispiel

Vor der Programmierung einer Spindel wird zunächst geprüft, ob diese überhaupt im Kanal bekannt ist:

```
…
N10 G90 Y0
N20 $IF EXIST[V.SPDL.LOG_AX_NR.S] == TRUE
N30   M3 S1000         (Spindel S mit Drehzahl 1000 U/min)
N40 $ELSE
N50   #MSG ["Spindel S nicht vorhanden!"] (Meldung ausgeben und anhal-
ten)
N55   M0
N60 $ENDIF
...
M30
```

14.3 Globale Variablen (V.G.)

Die Kennung für achsübergreifende, im Kanal global gültige Variablen ist "V.G. ...".

Die kanalglobale Kennung kann in 2 Varianten angegeben werden:

1: Achsnamen gemäß Kanalliste (Im Folgenden exemplarisch mit "X" dargestellt)

2: Achsindex [i] gemäß Kanalliste mit <i>: 0…31

Beispiel: V.G.NP_AKT.V.X oder V.G.NP_AKT.V[0]

V.G.<var_name>	**Bedeutung**	**Daten-typ**	**Einheit der Ein/ Ausga-be**	**Erlaub-ter Zu-griff: L**esen/ **S**chrei-ben
BLOCK_NR	Die zuletzt programmierte NC-Satz-Nummer	Integer	-	L
MASS_MM	Wenn Maßeinheit [mm], dann 0	Boolean	0 , 1	L
MASS_360	Wenn Maßeinheit [GRAD], dann 0	Boolean	0 , 1	L
I	I-Koordinate der Kreisprogrammierung. Bei aktiver Mittelpunktskorrektur (G165) Zugriff auf korrigierten Wert	Real	[mm, inch]	L
J	J-Koordinate der Kreisprogrammierung. Bei aktiver Mittelpunktskorrektur (G165) Zugriff auf korrigierten Wert	Real	[mm, inch]	L
K	K-Koordinate der Kreisprogrammierung. Bei aktiver Mittelpunktskorrektur (G165) Zugriff auf korrigierten Wert	Real	[mm, inch]	L
R	Der bei der Kreisinterpolation gefahrene Radius	Real	[mm, inch]	L
FEEDRATE	Der zuletzt programmierte Vorschub (F-Wort)	Real	[mm/ min, m/min, inch/min]	L
FEEDRATE_MODE	G-Nummer des aktuellen Vorschubmodus: 93: G93 aktiv, 94: G94 aktiv, 95: G95 aktiv, 194: G194 aktiv **[ab V2.11.2039.01]**	Integer	-	L
FEEDRATE_SCALE	Ermöglicht Anpassung eines in mm/min programmier-ten Vorschubes an die mit 'prog_start.feedrate_factor' (P-CHAN-00108) vorgegebene Einheit. Liefert den Wert 1000 für 'prog_start.feedrate_factor' = 0.1 (m/min) 1 für 'prog_start.feedrate_factor' = 100 (mm/min) Programmierbeispiel: F2000 / V.G.FEEDRATE_SCALE ergibt bei 1000: F2 (m/min) 1: F2000 (mm/min) **[ab V2.11.2024.00]**	Integer	-	L
MERR[i]	Korrekturoffset des Kreismittelpunktes in den Haupt-achsen der aktuellen Ebene mit <i> := 0 , 1	Real	[mm, inch]	L

Die „WZ[j]“-Variablen ermöglichen den lesenden Zugriff auf die Daten eines **beliebigen** Werkzeuges. Sie sind sowohl bei einer externen Werkzeugverwaltung (transparenter Zugriff) als auch bei Verwendung einer internen Werkzeugtabelle verfügbar (<j> entspricht dabei dem Index des Werkzeuges (bzw. Werkzeugnummer) aus der Werkzeugliste [5] [▶ 894]).

Der Schreibzugriff ist nur bei Verwendung einer internen Werkzeugtabelle zulässig.

WZ[j].R	Radius des Werkzeuges	Real	[mm, inch]	L/S*
WZ[j].L	Länge des Werkzeuges	Real	[mm, inch]	L/S*
WZ[j].P[i]	Parameter des Werkzeuges mit <i>: 0 … 59	Real	-	L/S*
WZ[j].V[i] oder WZ[j].V.X	Versatz in Achse <i> bzw. "X" des Werkzeuges mit <i>: 0 … 31	Real	[mm, inch]	L/S*
WZ[j].ME	Maßeinheit von Radius, Länge und Achsversätzen, liefert bei Verwendung einer Werkzeugliste immer 0 (für [mm]), ansonsten ohne Bedeutung	Boolean	0 , 1	L
WZ[j].OK	Gültigkennung des Werkzeuges; wenn gültig, dann 1	Boolean	0 , 1	L/S*
WZ[j].SPDL_AX_NR	Logische Achsnummer der zugeordneten Spindel	Integer	-	L/S*
WZ[j].KIN_PARAM[i]	Kinematikparameter des Werkzeuges in interner Einheit mit <i>: 0 … 69	Real	[0.1 µm, 10^{-4}°]	L/S*
WZ[j].KIN_ID	Kinematik-ID des Werkzeuges	Integer	-	L/S*
WZ[j].TYPE	Werkzeugtyp (0: Fräs-WZ 1: Dreh-WZ 2: Schleif-WZ)	Integer	-	L
WZ[j].TOOL_FIXED	Werkzeug ist ausrichtbar oder feststehend	Boolean	0 , 1	L/S*
WZ[j].SRK_ID	Schneidenlage bei einem Drehwerkzeug	Integer	-	L/S*
WZ[j].S_MIN_SPEED	Minimale Drehgeschwindigkeit (WZ-Dynamikdaten)	Real	[U/min]	L/S*
WZ[j].S_MAX_SPEED	Maximale Drehgeschwindigkeit (WZ-Dynamikdaten)	Real	[U/min]	L/S*
WZ[j].S_MAX_ACC	Maximale Beschleunigung (WZ-Dynamikdaten)	Real	[°/s^2]	L/S*
WZ[j].SISTER_VALID	Gültigkennung des Schwesterwerkzeuges (TOOL-ID)	Boolean	0 , 1	L/S*
WZ[j].SISTER	Nummer des gültigen Schwesterwerkzeuges	Integer	-	L/S*
WZ[j].VARIANT_VALID	Gültigkennung des Variantwerkzeuges (TOOL-ID)	Boolean	0 , 1	L/S*
WZ[j].VARIANT	Nummer des gültigen Variantwerkzeuges	Integer	-	L/S*
WZ[j].GOBJECT[i].*	Zugriff auf die Unterelemente eines bestimmten grafischen Objektes **[ab V3.01.3018.00]** mit <i>: 0 … 4	-	-	…
WZ[j].LINKPOINT.*	Zugriff auf die Unterelemente des zugehörigen Linkpoints **[ab V3.01.3018.00]**	-	-	...

S*: Schreibzugriff auf Daten der internen Werkzeugverwaltung ab CNC-Version V3.1.3079.08

Die "WZ_AKT"-Variablen sowie "T_AKT" und "D_AKT" ermöglichen den Zugriff auf Daten des **aktuell angewählten** Werkzeuges. Sie sind sowohl bei einer externen Werkzeugverwaltung als auch bei Verwendung einer internen Werkzeugtabelle verfügbar.

T_AKT	Nummer des angewählten Werkzeuges	Integer	-	L
D_AKT	Nummer des angewählten Werkzeugdatensatzes	Integer	-	L

Hinweis

Ein Schreibzugriff bewirkt die temporäre Änderung der Daten des Werkzeuges, solange dieses angewählt ist. Die geänderten Daten sind mit der Anwahl eines neuen Werkzeuges (Dxx) oder bei Werkzeugabwahl (D0) verloren!

Ausnahme:

Bei einer externen Werkzeugverwaltung werden die s.g. zusätzlichen Werkzeugparameter (V.G.WZ_AKT.P[i]) mit der Anwahl eines neuen Werkzeuges oder bei Werkzeugabwahl (P-CHAN-00103) übernommen und gesichert.

WZ_AKT.R	Radius des angewählten Werkzeuges	Real	[mm, inch]	L/S
WZ_AKT.L	Länge des angewählten Werkzeuges	Real	[mm, inch]	L/S
WZ_AKT.P[i]	Freie Parameter des angewählten Werkzeuges mit <i>: 0 … 59	Real	-	L/S
WZ_AKT.V[i] oder WZ_AKT.V.X	Versatz in Achse <i> bzw. "X" des angewählten Werkzeuges mit <i>: 0 … 31	Real	[mm, inch]	L/S
WZ_AKT.ME	Maßeinheit von Radius, Länge und Achsversätzen des angewählten Werkzeuges, liefert bei Verwendung einer Werkzeugliste immer 0 (für [mm]), ansonsten ohne Bedeutung	Boolean	0 , 1	L
WZ_AKT.OK	Gültigkennung des angewählten Werkzeuges; ist immer 1, da nur Daten gültiger Werkzeuge übernommen werden. Bei der Anforderung ungültiger Werkzeuge erfolgt die Ausgabe einer Fehlermeldung.	Boolean	0 , 1	L/S*
WZ_AKT.SPDL_AX_NR	Logische Achsnummer der zugeordneten Spindel	Integer	-	L/S*
WZ_AKT.KIN_PARAM[i]	**ACHTUNG**: Hinweis zum Schreibzugriff: Wert muss in internen Einheiten programmiert sein! Kinematikparameter des angewählten Werkzeuges mit <i>: 0 … 69	Real	[0.1 µm, 10^{-4}°]	L/S
WZ_AKT.KIN_ID	Kinematik-ID des angewählten Werkzeuges	Integer	-	L/S*
WZ_AKT.TYPE	Werkzeugtyp des angewählten Werkzeuges (0: Fräs-WZ 1: Dreh-WZ 2: Schleif-WZ)	Integer	-	L
WZ_AKT.TOOL_FIXED	Werkzeug ist ausrichtbar oder feststehend	Boolean	0 , 1	L/S

WZ_AKT.SRK_ID	Schneidenlage des angewählten Drehwerkzeuges	Integer	-	L/S*
WZ_AKT.S_MIN_SPEED	Minimale Drehgeschwindigkeit (WZ-Dynamikdaten)	Real	[U/min]	L/S*
WZ_AKT.S_MAX_SPEED	Maximale Drehgeschwindigkeit (WZ-Dynamikdaten)	Real	[U/min]	L/S*
WZ_AKT.S_MAX_ACC	Maximale Beschleunigung (WZ-Dynamikdaten)	Real	[°/s²]	L/S*
WZ_AKT.SISTER_VALID	Gültigkennung des Schwesterwerkzeuges (TOOL-ID)	Boolean	0 , 1	L/S*
WZ_AKT.SISTER	Nummer des gültigen Schwesterwerkzeuges	Integer	-	L/S*
WZ_AKT.VARIANT_VALID	Gültigkennung des Variantwerkzeuges (TOOL-ID)	Boolean	0 , 1	L/S*
WZ_AKT.VARIANT	Nummer des gültigen Variantwerkzeuges	Integer	-	L/S*
WZ_AKT.WEAR_RADIUS	Gesamter Radiusverschleiß bei Radiuskompensation (OTC) (Summe diskreter + kontinuierlicher Verschleiß)	Real	[mm, inch]	L
WZ_AKT.WEAR_RADIUS_CONT	Kontinuierlicher Radiusverschleiß bei Radiuskompensation (OTC)	Real	[mm, inch]	L
WZ_AKT.WEAR[i] oder WZ_AKT.WEAR.X	Verschleiß in Achse <i> bzw. "X" bei Längenkompensation (OTC) mit <i>: 0 … 31	Real	[mm, inch]	L
WZ_AKT.WEAR_CONST	Verschleißkonstante (OTC)	Real	[0.1 µm/m]	L/S
WZ_AKT.GOBJECT[i].*	Zugriff auf die Unterelemente eines bestimmten grafischen Objekts (siehe FCT-C15) **[ab V3.01.3018.00]** mit <i>: 0 … 4	-	-	-
WZ_AKT.LINKPOINT.*	Zugriff auf die Unterelemente des zugehörigen Linkpoints (siehe FCT-C15) **[ab V3.01.3018.00]**	-	-	-

S*: Schreibzugriff auf diese aktuellen Werkzeugdaten ab CNC-Version V3.1.3079.08

NP[j].V[i] oder NP[j].V.X	Nullpunktverschiebung einer Achse <i> bzw. "X" mit <j>: 0 … 96 und <i>: 0 … 31 **ACHTUNG**: Schreibzugriff bewirkt bleibende Änderungen in internen Nullpunktdaten!	Real	[mm, inch]	L/S
NP[j].ALL	Ansprechen aller Achsen einer Nullpunktverschiebung mit <j>: 0 … 96 **ACHTUNG**: Schreibzugriff bewirkt bleibende Änderungen in internen Nullpunktdaten!	Real	[mm, inch]	L/S
NP_AKT.V[i] oder NP_AKT.V.X	Aktuelle (momentan aktive) Nullpunktverschiebung einer Achse <i> bzw. "X" mit <i>: 0 … 31	Real	[mm, inch]	L/S
NP_AKT.ALL	Ansprechen der aktuellen (momentan aktiven) Nullpunktverschiebungen aller Achsen	Real	[mm, inch]	L/S
NP_AKT.IDX	Index der aktuellen (momentan aktiven) Nullpunktverschiebungsgruppe (z.B. 0 bei G53, 1 bei G54...)	Integer	-	L
NP_DEFAULT	Index der im Grundzustand wirksamen Nullpunktverschiebungsgruppe nach Hochlauf	Integer	-	L/S

Die folgenden Variablen können bei einem CS-Stack verwendet werden, der mit #(A,B)CS DEF oder #(A,B)CS ON ohne ID-Angabe programmiert wurde. **[ab V3.01.3080.03]**

BCS_COUNT_DEF	Anzahl definierter BCS	Integer	-	L
BCS_COUNT_FREE_DEF	Anzahl noch freier Plätze zur Definition von BCS	Integer	-	L
BCS_COUNT_ACTIVE	Anzahl aktiver (verketteter) BCS	Integer	-	L
ACS_COUNT_DEF	Anzahl definierter ACS	Integer	-	L
ACS_COUNT_FREE_DEF	Anzahl noch freier Plätze zur Definition von ACS	Integer	-	L
ACS_COUNT_ACTIVE	Anzahl aktiver (verketteter) ACS	Integer	-	L
CS_COUNT_DEF	Anzahl definierter CS	Integer	-	L
CS_COUNT_FREE_DEF	Anzahl noch freier Plätze zur Definition von CS	Integer	-	L
CS_COUNT_ACTIVE	Anzahl aktiver (verketteter) CS	Integer	-	L

Die folgenden Variablen können bei einem CS-Stack verwendet werden, der mit #(A,B)CS ADD oder #(A,B)CS SELECT programmiert wurde. **[ab V3.01.3080.03]**

COORD_SYS_BCS_COUNT_DEF	Anzahl definierter BCS	Integer	-	L
COORD_SYS_ACS_COUNT_DEF	Anzahl definierter ACS	Integer	-	L
COORD_SYS_CS_COUNT_DEF	Anzahl definierter CS	Integer	-	L
COORD_SYS_COUNT_ALL_DEF	Gesamtanzahl definierter Koordinatensysteme (BCS+ACS+CS)	Integer	-	L
COORD_SYS_COUNT_ALL_ACTIVE	Gesamtanzahl aktiver Koordinatensysteme nach #CS SELECT	Integer	-	L

CNC_CHANNEL	Kanalnummer	Integer	-	L
IPO_COUNT	Systemzeitzähler	Integer	-	L
EXECUTION_MODE	Bearbeitungsmodus der CNC, Werte siehe Übersicht Bearbeitungsmodi Funktionsbedingungen	Integer	-	L
TOOL_COMP	Aktive Werkzeugkorrektur (1: RTCP 2: TLC 3: 2.5D)	Integer	-	L
AKT_PLATZ	Aktueller Platzversatzindex	Integer	-	L
AX_LINK.NR	Aktuelle bzw. zuletzt aktive Koppelgruppennummer	Integer	-	L
AX_LINK.ACTIVE	Ist eine Achskopplung (#AX LINK ON) aktiv? Wenn aktiv, dann 1	Boolean	0 , 1	L
AX_LINK_GROUP[i].ACTIVE	Ist eine bestimmte Koppelgruppe mit der Nummer <i>: 0 … 4 aktiv? Wenn aktiv, dann 1	Boolean	0 , 1	L
ROT_ACTIVE	Konturrotation (#ROTATION...) aktiv? Wenn aktiv, dann 1	Boolean	0 , 1	L
ROT_ANGLE	Winkel der Konturrotation	Real	[°]	L
ROT_CENTER1	Versatz 1. Hauptachse zum Drehpunkt bei Konturrotation	Real	[mm, inch]	L
ROT_CENTER2	Versatz 2. Hauptachse zum Drehpunkt bei Konturrotation	Real	[mm, inch]	L
TIMER[i]	Zählerstand des Timers mit der Nummer <i>: 0 … 127	Integer	[ms]	L
PROG_ABS	Maßsystem, 0: G91 aktiv 1: G90 aktiv	Boolean	0 , 1	L
ACT_PLANE	G-Nummer der aktuell aktiven Ebene: 17: G17 aktiv, 18: G18 aktiv, 19: G19 aktiv	Integer	-	L
CNC_RELEASE	Buildnummer der CNC-Version (Alte Syntax)	Integer	-	L
CNC_VERSION.MAJOR	Majorversion der CNC **[ab V2.11.2800]** (z.B. **2** bei V**2.11.2807.42**)	Integer	-	L
CNC_VERSION.MINOR	Minorversion der CNC **[ab V2.11.2800]** (z.B. **11** bei V**2.11.2807.42**))	Integer	-	L
CNC_VERSION.BUILD	Buildnummer der CNC-Version **[ab V2.11.2800]** (z.B. **2807** bei V**2.11.2807.42**))	Integer	-	L
CNC_VERSION.PATCH	Patchnummer der CNC-Version **[ab V2.11.2800]** (z.B. **42** bei V2.11.2807.**42**)	Integer	-	L
WCS_POSLIMIT_1 WCS_POSLIMIT_2 WCS_POSLIMIT_3	Bewegungsgrenzen in den Hauptachsen im WCS. Nur sinnvoll in Verbindung mit dem NC-Befehl #GET WCS POSLIMIT [▶ 774]	Real	[mm, inch]	L

SIGNAL_READ	Lesen eines Signals ohne Warten **[ab V2.11.2820.00]** 1: Signal vorhanden und gelesen, 0: Signal nicht vorhanden Nur sinnvoll in Verbindung mit dem NC-Befehl #SIGNAL READ [▶ 400]	Boolean	0 , 1	L
EXECUTION_MODE	Bearbeitungsmodus im NC-Programmauftrag, Bedeutung des Wertes siehe HLI-Beschreibung	Integer	-	L
ACTIVE_MODE	Aktive Betriebsart, mit der ein NC-Programm gestartet wurde, z.B. Automatikbetrieb, Handsatz (Werte siehe Betriebsarten) **[ab V3.1.3080.10, V3.1.3107.43]**	Integer	-	L

RANDOM	Liefert zufälligen Zahlenwert im Bereich von 0.0 bis 1.0 Variable ist unter TwinCAT nicht verfügbar.	Real	-	L

Die nachfolgenden Variablen ermöglichen den Zugriff auf die Daten einer in den Kanalparametern definierten Kinematik.

In Versionen bis V2.11.28xx sowie für einstufige Transformationen ab V3.00 erfolgt der Zugriff mit V.G.KIN[j].* (mit j:= Kinematik-ID).

Für mehrstufige Transformationen (ab V3.00.3012.00) erfolgt der Zugriff mit V.G.KIN_STEP[i].ID[j].*

(mit i:= Transformationsstufe 0 oder 1und j:= Kinematik-ID).

Achtung

Schreibzugriff bewirkt bleibende Änderung in internen Kanalparameterdaten! Wert muss in internen Einheiten programmiert sein!

Die Änderungen bleiben bis zum nächsten Steuerungshochlauf oder Aktualisieren der Kanalparameterliste erhalten!

Schreibzugriffe auf kinematikrelevante Versatzdaten in Kanal- oder Werkzeugparametern bei aktiver Kinematik (#TRAFO ON) sind erst nach #TOOL REFRESH [▶ 818] oder nach Ab- und Wiederanwahl (#TRAFO OFF- #TRAFO ON) wirksam.

(einstufig bis V3.00.3017.01): **V.G.KIN[j].*** (mehrstufig ab V3.00.3018.00): **V.G.KIN_STEP[i].ID[j].***	**Bedeutung**	**Datentyp**	**Einheit der Ein/Ausgabe**	**L/S***
PARAM[k]	Kinematikparameter mit <k>:= 0 ... 69	Real	[0.1 µm, 10^{-4}°]	L/S
TYPE	Kinematiktyp der angegebenen Kinematik-ID	Integer	-	L/S

* Erlaubter Zugriff: L/S = Lesen/Schreiben

Zusätzlich stehen für den Zugriff auf die Daten der **Universellen Kinematik** (ID 91, [FCT-C27]) die nachfolgend aufgeführten Variablen zur Verfügung:

ZERO_ORIENTATI-ON[k]	Null-Orientierung des Werkzeuges mit <k>:= 0, 1, 2 (X, Y, Z)	Real	-	L/S
ZERO_POSITION[k]	Null-Position des Werkzeuges mit <k>:= 0, 1, 2 (X, Y, Z)	Real	[0.1 µm, 10^{-4}°]	L/S
PROGRAMMING_MO-DE	Programmiermodus, siehe P-CHAN-00112	Integer	-	L/S
RTCP	Winkeltransformation	Boolean	-	L/S
NUMBER_OF_AXES	Anzahl der Achsen in kinematischer Kette	Integer	-	L/S
AXIS[k].*	Achse der kinematischen Kette mit <k>:= 0 … 5	-	-	-
AXIS[k].TYPE	Achstyp (1: Translator, 2: Rotator)	Integer	-	L/S
AXIS[k].ORIENTATI-ON[l]	Orientierungsvektor der Achse (Richtung) mit <l>:= 0, 1, 2 (X, Y, Z)	Real	-	L/S
AXIS[k].POINT[l]	Stützpunkt auf der Achse mit <l>:= 0, 1, 2 (X, Y, Z)	Real	[0.1 µm, 10^{-4}°]	L/S
CHAIN[k]	Beschreibung der Achsreihenfolge in der kinemati-schen Kette mit <k>:= 0 … 5	Integer	-	L/S

Die folgenden Variablen ermöglichen den Lesezugriff auf die aktuell angewählte Kinematik-ID:

KIN_ID	Aktuell angewählte Kinematik-ID bei einstufigen Transformationen	Integer	**-**	L
KIN_TYPE	Aktuell angewählter Kinematiktyp bei einstufigen Transformationen **[ab V3.1.3080.09]**	Integer	**-**	L
KIN_ID_STEP[0]	Aktuell angewählte Kinematik-ID der ersten Stufe bei mehrstufigen Transformationen **[ab V3.00.3018.00]**	Integer	-	L
KIN_TYPE_STEP[0]	Aktuell angewählter Kinematiktyp der ersten Stufe bei mehrstufigen Transformationen **[ab V3.1.3080.09]**	Integer	-	L
KIN_ID_STEP[1]	Aktuell angewählte Kinematik-ID der zweiten Stufe bei mehrstufigen Transformationen **[ab V3.00.3018.00]**	Integer	-	L
KIN_TYPE_STEP[1]	Aktuell angewählter Kinematiktyp der zweiten Stufe bei mehrstufigen Transformationen **[ab V3.1.3080.09]**	Integer	-	L
TOTAL_KIN_OFF-SET[i]	Wirksamer Gesamtversatz der angewählten Kinema-tik, bestehend aus der Summe der Kinematikversätze des aktiven Werkzeuges und den Kinematikversätzen aus der Kanalparameterliste mit <i>:= 0 … 69 **[ab V2.11.2024.00]**	Real	[0.1 µm, 10^{-4}°]	L

Nachfolgende Variablen für PCS-Transformationen sind verfügbar ab V3.1.3110.

V.G.TRAFO_PCS[j].*	**Bedeutung**	**Daten-typ**	**Einheit der Ein/ Ausga-be**	**L/S***
PARAM[k]	Parameter der PCS-Transformation mit <k>:= 0 ... 69	Real	[0.1 µm, 10^{-4}°]	L/S
TYPE	Typ der PCS-Transformation mit der angegebenen Transformations-ID	Integer	-	L/S

V.G.TRAFO_PCS_ID	Aktuell angewählte PCS Transformations-ID	Integer	-	L
V.G.TRAFO_PCS_TY-PE	Aktuell angewählter PCS Transformationstyp	Integer	-	L

MAIN_FILE_NAME	Dateiname des NC-Hauptprogramms. Bei MDI oder Handsatz liefert die Variable den Dateiname "-".	String	-	L
MAIN_PROG_NAME	Name (%...) des NC-Hauptprogramms	String	-	L
MAIN_PROG_NR	Nummer des NC-Hauptprogramms, wenn der Programmname eine Zahl ist	Integer	-	L
FILE_NAME	Dateiname des momentan aktiven NC-Programms	String	-	L
PROG_NAME	Name (%...) des momentan aktiven NC-Programms	String	-	L
PROG_NR	Nummer des momentan aktiven NC-Programms, wenn der Programmname eine Zahl ist	Integer	-	L
PROG_LEVEL	Programmebene des momentan aktiven NC-Programms: 1: Hauptprogramm ≥ 2: Lokales oder globales Unterprogramm	Integer	-	L
PATH_NR	Logische Pfadnummer des momentan aktiven NC-Haupt- bzw. globalen Unterprogramms gemäß der Hochlaufliste (P-STUP-00019)	Integer	-	L
FILE_OFFSET	Dateiposition des Beginns des NC-Satzes, in dem V.G.FILE_OFFSET programmiert ist	Integer	-	L
MIRROR_ACTIVE	Spiegelung (G351 X... oder G21, G22, G23) aktiv? Wenn aktiv, dann 1	Boolean	0 , 1	L
BLOCK_SEARCH_ACTIVE	Satzvorlauf aktiv? Wenn aktiv, dann 1	Boolean	0 , 1	L
TRAFO_ACTIVE	Kinematische Transformation (#TRAFO...) aktiv? Wenn aktiv, dann 1	Boolean	0 , 1	L
MCS_ACTIVE	Temporärer Übergang in das Maschinenkoordinatensystem (#MCS...) aktiv? Wenn aktiv, dann 1	Boolean	0 , 1	L
HSC_ACTIVE	Freiformflächenbearbeitung (#HSC...) bzw. Splineinterpolation (#SPLINE... oder G151) aktiv? Wenn aktiv, dann 1	Boolean	0 , 1	L
HSC_SURFACE_ACTIVE	Freiformflächenbearbeitung (#HSC [SURFACE...]) aktiv? Wenn aktiv, dann 1	Boolean	0 , 1	L
CONT_MODE_ACTIVE	Polynomüberschleifen (G261) aktiv? Wenn aktiv, dann liefert Variable den Wert der aktuellen Überschleifart: 3 bei Überschleifart DIST 4 bei Überschleifart DEV 5 bei Überschleifart POS 6 bei Überschleifart DIST_SOFT 6 bei Überschleifart DIST_MASTER 7 bei Überschleifart PTP 0 bei G260	Integer	-	L

TRC_ACTIVE	Werkzeugradiuskorrektur (G41, G42) aktiv? Wenn aktiv, dann 1	Boolean	0 , 1	L
OTC_ACTIVE	Online-Werkzeugkorrektur (#OTC...) aktiv? Wenn aktiv, dann 1	Boolean	0 , 1	L
CAXTRACK_ACTIVE	C-Achsnachführung (#CAXTRACK…) aktiv? Wenn aktiv, dann 1	Boolean	0 , 1	L
CYCLE_ACTIVE	Ist die aktuelle Programmebene ein Zyklus? Wurde das aktuelle Programm mit L oder LL CYCLE... oder G8xx mit Parameterübergabe aufgerufen, dann 1	Boolean	0 , 1	L
@P[i].VALID	Ist Übergabeparameter im Zyklusaufruf [▷ 212] bzw. Unterprogrammaufruf mit G8xx programmiert? 0: Parameter nicht programmiert 1: Parameter ist programmiert Mit <i>:= 1 … 200 wird die Nummer des Parameters angegeben, der auf Gültigkeit geprüft werden soll. (z.B. 1 bei @P1 bzw. bei Abfrage des ersten Parameters bei Aufruf über G8xx). HINWEIS zum Lesezugriff: Die Variable kann nur innerhalb eines Zyklus oder in Unterprogrammen G8xx mit Parameterübergabe verwendet werden!	Boolean	0 , 1	L
TIME_STAMP	Aktueller Zeitstempel <Datum Zeit> im Format <*DD.MM.YYYY hh:mm:ss:zzz*> mit: D: Tag, 2-stellig M: Monat, 2-stellig Y: Jahreszahl, 4-stellig h: Stunden, 2-stellig m: Minuten, 2-stellig s: Sekunden, 2-stellig z: Millisekunden, 3-stellig (z.B. 16.06.2015 14:08:10:123) **[ab V2.10.1507.02]**	String	-	L
TIME_STAMP_FILE_NAME	Aktueller Zeitstempel <Datum Zeit> im Format gemäß ISO 6801:2004 <*YYYYMMDD*T*hhmmsszzz*> ohne '.' und ':' z.B. zur Verwendung in Namen von Protokolldateien mit: Y: Jahreszahl, immer 4-stellig M: Monat, 2-stellig D: Tag, 2-stellig h: Stunden, 2-stellig m: Minuten, 2-stellig s: Sekunden, 2-stellig z: Millisekunden, 3-stellig (z.B. 20150616T140810123) **[ab V2.11.2024.03]**	String	-	L
LIFT_ACTIVE	Abheben / Senken einer Achse (X[LIFT...]) aktiv? Wenn aktiv, dann 1	Boolean	0 , 1	L

Die nachfolgenden Variablen ermöglichen den Zugriff auf Daten der in den Kanalparametern definierten M/H-Funktionen.

Hinweis

Bei der Festlegung von M/H-Funktionen mit Vorausgabefunktion (MEP_SVS, MET_SVS) ist folgendes zu beachten:

Stimmen die Synchronisationsart der M/H-Funktion und der Schreibzugriff (S) auf die Vorausgabewerte nicht überein, so wird implizit die Synchronisationsart mit angepasst.

Stimmen die Synchronisationsart der M/H-Funktion und der Lesezugriff (L) auf die Vorausgabewerte nicht überein, so wird eine Fehlermeldung ausgegeben.

Achtung

Die Änderungen bleiben bis zum nächsten Hochlauf oder Aktualisieren der Kanalparameterliste erhalten!

M_FCT[i].SYNCH	Synchronisationsart einer in den Kanalparametern P-CHAN-00041 definierten M-Funktion <i>. Der Wert der Synchronisationsart wird in hexadezimaler Schreibweise ‘Hex‘ [▶ 22] angegeben.	Integer	-	L/S
M_FCT[i].PRE_OUTP_PATH	Vorausgabeweg *m_pre_output* einer in den Kanalparametern P-CHAN-00070 definierten M-Funktion <i> vom Typ MEP_SVS	Real	[mm, inch]	L/S
M_FCT[i].PRE_OUTP_TIME	Vorausgabezeit *m_pre_output* einer in den Kanalparametern P-CHAN-00070 definierten M-Funktion <i> vom Typ MET_SVS	Real	[s]	L/S
H_FCT[i].SYNCH	Synchronisationsart einer in den Kanalparametern P-CHAN-00027 definierten H-Funktion <i>. >. Der Synchronisationsart der H-Funktionen (P-CHAN-00027)Wert der Synchronisationsart wird in hexadezimaler Schreibweise ‘Hex‘ [▶ 22] angegeben.	Integer	-	L/S
H_FCT[i].PRE_OUTP_PATH	Vorausgabeweg *h_pre_output* einer in den Kanalparametern P-CHAN-00107 definierten H-Funktion <i> vom Typ MEP_SVS	Real	[mm, inch]	L/S
H_FCT[i].PRE_OUTP_TIME	Vorausgabezeit *h_pre_output* einer in den Kanalparametern P-CHAN-00107 definierten H-Funktion <i> vom Typ MET_SVS	Real	[s]	L/S

Die "V.G.SPEED_LIMIT" Variablen ermöglichen den Zugriff auf die in den Kanalparametern vorbelegten Parameter für den Geschwindigkeits-Look-Ahead [1] [▶ 894]-6.

Hinweis

Beim Schreibzugriff wird die geänderte Parametrierung an den Kanal ausgegeben. Eine eventuell aktive Bewegung wird unterbrochen!

Stimmen die Einheit der Abstandsparameter und der Schreibzugriff (S) auf diese Werte nicht überein, so wird implizit die Einheit mit angepasst.

Stimmen die Einheit der Abstandsparameter und der Lesezugriff (L) auf diese Werte nicht überein, so wird eine Fehlermeldung ausgegeben.

Achtung

Die Änderungen bleiben bis Programmende oder Reset erhalten! Nach Programmstart gelten wieder die Einstellungen der Kanalparameter.

SPEED_LIMIT.ENABLE	An-/Abwahl des Geschwindigkeits-Look-Ahead (0: Abwahl 1: Anwahl)	Boolean	0 , 1	L/S
SPEED_LIMIT.VEL_LIMIT	Definition Geschwindigkeitsgrenzwert	Integer	[%]	L/S
SPEED_LIMIT.TIME	Festlegung der Einheit der Abstände zur und von der "Ecke" (0: Weg 1: Zeit)	Boolean	0 , 1	L/S
SPEED_LIMIT.DIST_TO_CORNER	Wegabstand zur "Ecke"	Real	[mm, inch]	L/S
SPEED_LIMIT.DIST_FROM_CORNER	Wegabstand von der "Ecke"	Real	[mm, inch]	L/S
SPEED_LIMIT.TIME_TO_CORNER	Zeitabstand zur "Ecke"	Real	[s]	L/S
SPEED_LIMIT.TIME_FROM_CORNER	Zeitabstand von der "Ecke"	Real	[s]	L/S
SPEED_LIMIT.OVERRIDE_WEIGHT	Gewichtung des Geschwindigkeits-Grenzwertes (0: keine Gewichtung, 1: Gewichtung über Override)	Boolean	0 , 1	L/S

MAX_NC_BLOCKS_AHEAD	Satzbezogene Decodervorlaufbegrenzung: alle NC-Sätze	Integer	-	L/S
MAX_MOTION_BLOCKS_AHEAD	Satzbezogene Decodervorlaufbegrenzung: nur NC-Bewegungssätze	Integer	-	L/S
MAX_TIME_AHEAD	Zeitbezogene Decodervorlaufbegrenzung	Real	[s]	L/S

CIRCLE_CP_ABS	Angabe Kreismittelpunkt, 0: G162 aktiv 1: G161 aktiv	Boolean	0 , 1	L
CIRCLE_CPC_ACTIVE	Kreismittelpunktskorrektur (G165) aktiv? Wenn aktiv, dann 1 0 bei G164	Boolean	0 , 1	L

SEGMENTATION_ACTIVE	Segmentierung (#SEGMENTATION...) aktiv? Wenn aktiv, dann 1	Boolean	0 , 1	L
STROKE_DEF_ACTIVE	Definition einer Hubbewegung (#STROKE DEF…) aktiv? Wenn aktiv, dann 1	Boolean	0 , 1	L
PUNCH_ACTIVE	Stanzen (#PUNCH...) aktiv? Wenn aktiv, dann 1	Boolean	0 , 1	L
NIBBLE_ACTIVE	Nibbeln (#NIBBLE...) aktiv? Wenn aktiv, dann 1	Boolean	0 , 1	L
EDM_ACTIVE	Erodierbearbeitung (#EDM...) aktiv? Wenn aktiv, dann 1 **[ab CNC-Version V3.1.3019]**	Boolean	0 , 1	L
FRICTION_ACTIVE	Aufnahme Reibkompensationswerte (#FRICTION ON […] aktiv? Wenn aktiv, dann 1 **[ab CNC-Version V2.11.2022.05]**	Boolean	0 , 1	L
VECTOR_OFFSET_ACTIVE	Schnittkantenkorrektur (#VECTOR OFFSET...) aktiv? Wenn aktiv, dann 1	Boolean	0 , 1	L
DIAMETER_PROG_ACTIVE	Durchmesserprogrammierung (G51) aktiv? Wenn aktiv, dann 1 **[ab CNC-Version V2.11.2051.00]**	Boolean	0 , 1	L
DIAMETER_PROG.ZERO_OFFSET	Liefert den Wert von P-CHAN-00091, wenn gilt: - Durchmesserprogrammierung (G51) ist aktiv - Plandrehachse ist konfiguriert **[ab CNC-Version V2.11.2051.00]**	Boolean	0 , 1	L
GEAR_LINK_ACTIVE	Programmierbare Achskopplung (#GEAR LINK ON […] [▸ 491]) im Kanal aktiv? Wenn aktiv, dann 1	Boolean	0 , 1	L

FILE_EXIST	Ergebnis der Abfrage mit #FILE EXIST [▸ 443] Wenn Prüfung der Datei positiv, dann 1	Boolean	0 , 1	L

MEAS_TYPE	Wert des aktuell aktiven Messtyps [▸ 96] **[ab V2.11.2022.03]**	Integer	-	L

TRACK_CS.X TRACK_CS.Y TRACK_CS.Z TRACK_CS.A TRACK_CS.B TRACK_CS.C	Werte der Masterposition bei dynamischen Koordinatensystemen (siehe [FCT-C30]).	REAL	[mm, inch]	L

Die nachfolgenden V.G.-Variablen ermöglichen das Lesen der Anzahl aller Schreibzugriffe auf die Inhalte der jeweiligen Liste seit Steuerungshochlauf. [verfügbar ab V2.11.2037.03]

CHAN_LIST_INVOKE	Kanalparameterliste	Integer	-	L
CLAMP_LIST_INVOKE	Liste der Platzversätze	Integer	-	L
TOOL_LIST_INVOKE	Werkzeugliste	Integer	-	L
VE_VAR_LIST_INVOKE	Liste externer Variablen	Integer	-	L
ZERO_POINT_LIST_INVOKE	Nullpunktliste	Integer	-	L

Die Anzahl der Schreibzugriffe auf einzelne Nullpunktgruppen sind ebenfalls möglich [verfügbar ab V2.11.2037.03].

NP[j].INVOKE_COUNT	Anzahl aller Schreibzugriffe auf eine Nullpunktverschiebung seit Steuerungshochlauf mit <j>: 0 … 96	Integer	-	L
NP_AKT.INVOKE_COUNT	Anzahl aller Schreibzugriffe auf aktuelle Nullpunktverschiebungsgruppe seit Steuerungshochlauf	Integer	-	L

Programmierbeispiel

Lesen der Anzahl von Schreibzugriffen einer Nullpunktgruppe

```
N010 P1 = V.G.NP[1].INVOKE_COUNT ; Zähler lesen

N020 V.G.NP[1].V.X = 15
N030 V.G.NP[1].V.X = 16

N040 $IF P1+2 != V.G.NP[1].INVOKE_COUNT ; Zähler erneut lesen
N050   #ERROR [ID1]
N060 $ENDIF

N070 G54
N080 $IF V.G.NP_AKT.INVOKE_COUNT != V.G.NP[1].INVOKE_COUNT
N090   #ERROR [ID2]
N100 $ENDIF

N110 G53
N120 M30
```

Die nachfolgenden V.G.-Variablen für die TCP-Geschwindigkeitsbegrenzung stehen ab V3.1.3079.26 zur Verfügung

LIMIT.KIN[i].TOOL. LENGTH	Werkzeuglänge mit <i>:= 0, 1 Index der konfigurierten Kinematik	Real	[mm]	L/S
LIMIT.KIN[i].TOOL. KIN_PARAM[j]	Kinematikparameter des Werkzeugs mit <j>:= 0..69 Index des Kinematikparameters	Real	[0.1 µm, 10-4°]	L/S

14.3.1 Versionieren von NC-Programmen

Über die V.G.-Variable V.G.PROG_VERSION können NC-Programme mit einer Versions-Nr. versehen werden.

Hinweis

Die komplette Versions-Nr. muss zwingend das Format “<Major>.<Minor>.<Build>.<Patch>“ haben.

Wird ein anderes Format verwendet, so wird der Fehler mit der ID 22015 ausgegeben.

Die komplette Version-Nr. setzt sich wie folgt zusammen:

“Complete“=“<Major>.<Minor>.<Build>.<Patch>“

Beispielsweise können die komplette Versions-Nr. oder auch einzelne Elemente der Versions-Nr. festgelegt werden.

```
V.G.PROG_VERSION.COMPLETE = "4.1.2.3"
V.G.PROG_VERSION.PATCH = 4
```

PROG_VERSION.MA-JOR	Majorversion des NC-Programms (z.B. **4** bei **4**.1.2.3)	UNS08	-	L/S
PROG_VERSION.MI-NOR	Minorversion des NC-Programms (z.B. **1** bei 4.**1**.2.3)	UNS08	-	L/S
PROG_VERSI-ON.BUILD	Buildversion des NC-Programms (z.B. **2** bei 4.1.**2**.3)	UNS08	-	L/S
PROG_VERSI-ON.PATCH	Patchversion des NC-Programms (z.B. **3** bei 4.1.2.**3**)	UNS08	-	L/S
PROG_VERSI-ON.COMPLETE	Major.Minor.Build.Patch	STRING	-	L/S

Die Versionsinformation wird an aufgerufene Unterprogramme vererbt. In Unterprogrammen selbst kann ebenfalls eine eigene Version-Nr. vergeben werden, diese wird dann wiederum an deren aufgerufene Unterprogramme vererbt.

Beispiel

Versionieren von NC-Programmen

Beispiel 1

```
%L UP_1
N110 V.G.PROG_VERSION.COMPLETE = "5.1.2.3"
N120 #MSG ["Version UP_1: %s", V.G.PROG_VERSION.COMPLETE]
N130 M17

%MAIN
N010 V.G.PROG_VERSION.COMPLETE = "4.1.2.3"
N20 LL UP_1
N30 #MSG ["Version Main: %s", V.G.PROG_VERSION.COMPLETE]
N040 M30
```

Es wird folgendes ausgegeben:

Version UP_1: 5.1.2.3

Version Main: 4.1.2.3

Beispiel 2

```
%L UP_1
( --- keine eigene Versionsangabe ---)
N120 #MSG ["Version UP_1: %s", V.G.PROG_VERSION.COMPLETE]
N130 M17

%MAIN
N010 V.G.PROG_VERSION.COMPLETE = "4.1.2.3"
N20 LL UP_1
N30 #MSG ["Version Main: %s", V.G.PROG_VERSION.COMPLETE]
N040 M30
```

Es wird folgendes ausgegeben:

Version UP_1: 4.1.2.3

Version Main: 4.1.2.3

14.4 Eigendefinierte Variablen (#VAR, #ENDVAR, #DELETE)

Eigendefinierte Variablen werden im NC-Haupt- oder Unterprogramm **nach** dem Programmname in einem Deklarationsblock angelegt und ggf. initialisiert, der mit **#VAR** beginnt und mit **#END-VAR** abgeschlossen wird.

Hinweis

Die eigendefinierten Variablen haben die Einstiegskennungen V.P. , V.S. und V.L.. Ihnen können Werte im REAL-Format zugewiesen werden. Ab Version V2.11.2032.08 stehen auch eigendefinierte Variablen mit der Einstiegskennung V.CYC. [▶ 629] zur Verfügung.

`Syntax zum Anlegen eines Deklarationsblocks für eigendefinierte Variablen:`

#VAR	Beginn des Deklarationsblockes
:	
:	Deklarations und Initialisierungsbereich
:	
#ENDVAR	Ende des Deklarationsblockes

Mit der Einführung von V.CYC.-Variablen [▶ 629] sind für alle Arten von eigendefinierten Variablen folgende Erweiterungen verfügbar (ab: V2.11.2032.08, V2.11.2832.00, V3.1.3079.41, V3.1.3107.30)

Die Deklaration kann neben der Angabe des Variablennamens auch die Festlegung eines Datentyps und eines Initialwerts umfassen. Ohne Zuweisung eines Initialwertes wird die Variable mit dem Initialwert des jeweiligen Datentyps belegt. Ohne Angabe eines Datentyps wird die Variable grundsätzlich im REAL-Format angelegt.

`Syntax der Deklaration und Initialisierung:`

V.P | **S** | **L** | **CYC.***<Name>* **:** *<Datentyp>* **=** *<Initialwert>* | **"***<Initialstring>***"**

<Name>	Frei gewählter Name der eigendefinierten Variable
<Datentyp>	Kennungen für den Datentyp (optional): BOOLEAN SGN08, UNS08 SGN16, UNS16 SGN32, UNS32 REAL64 STRING[i] mit <i>:= 0..126
<Initialwert>, "*<Initialstring>*"	Initialwert bzw. String der Variable (optional) gemäß Datentyp

Programmierbeispiel

```
Anlegen von Variablen mit und ohne Typdeklaration
%test_var_def_1
:
#VAR
  V.P.VAR_1
  V.P.VAR_2 = 10.67
  V.P.VAR_3 : UNS32 = 10
  V.L.NAME_1 : STRING[20] = "GRUNDPLATTE"
  V.L.VAR_1 : REAL64 = 23.45
  V.L.VAR_2 : SGN08
#ENDVAR


Zur besseren Übersicht kann die Initialisierung eines Variablen-Arrays
mit dem „\"-Zeichen auch über mehrere NC-Zeilen geschrieben werden.
%test_var_def_2
:
#VAR
  V.P.ARRAY_1[3][6] = [10,11,12,13,14,15, \
                       20,21,22,23,24,25, \
                       30,31,32,33,34,35 ]
  V.L.MY_ARRAY[3][6] = [10,11,12,13,14,15, 20,21,22,23,24,25,
30,31,32,33,34,35]
#ENDVAR
```

Hinweis

Der Zugriff auf Arrayvariablen beginnt bei Index 0! Mit obigem Beispiel liefert der Zugriff V.L.MY_ARRAY[0][5] somit den Wert 15.

Eigendefinierte Variablen und Variablen-Arrays können im NC-Programm auch wieder gelöscht werden. Hierzu steht der Befehl #DELETE zur Verfügung.

Syntax:

#DELETE V.<*name*> {, **V.**<*name*>}

Programmierbeispiel

```
#DELETE V.P.ARRAY_1, V.L.MY_ARRAY, V.P.VAR_1, V.L.VAR_1, V.S.VAR_1
```

Des Weiteren stehen die SIZEOF- und die EXIST-Funktion zur Verfügung (siehe Kapitel Arithmetische Ausdrücke <expr> [▶ 32]), die die Dimensionsgröße von Variablenarrays ermitteln und die Existenz von eigendefinierten Variablen überprüfen.

Programmierbeispiel

Durch die EXIST-Abfrage auf eine eigendefinierte V.S.-Arrayvariable (mit einem beliebigen gültigen Index) wird geprüft, ob diese Variable z.B. bereits in einem vorhergehenden NC-Programm angelegt wurde oder noch definiert werden muss:

```
…
N10 $IF EXIST[V.S.EXAMPLE[0]] == TRUE
N20  V.S.EXAMPLE[2] = 10  ;V.S-Variable[2] mit einem Wert belegen
N30 $ELSE
N40   #VAR
N50     V.S.EXAMPLE[5] = [1,2,3,4,5 ]
N60   #ENDVAR
N70 $ENDIF
…
M30
```

14.4.1 Global, bis Hauptprogrammende gültig (V.P.)

Mit der Kennung "V.P. ..." ist es möglich, eigene Variablen zu definieren, die nach dem Anlegen in der aktuellen Programmebene und in allen weiteren Programmebenen bekannt sind (global), aber nicht über das Hauptprogrammende hinweg gültig sind. V.P.-Variablen können Werte im REAL-Format zugewiesen werden. Ab Version V2.11.2032.08 sind auch Deklarationen anderer Datentypen [▶ 620] mit Initialwerten möglich.

V.P.-Variablen können auch als mehrdimensionalen Arrays angelegt werden. Es sind bis zu 4 Dimensionen möglich, z.B. V.P.TEST[1][2][3][4].

Syntax:

V.P.<FREE_DEF> globale, nicht (Haupt-) programmübergreifende Variable

<FREE_DEF> Frei gewählter Name, der aus beliebigen Zeichen (außer Leerzeichen, Tabulatoren, Kommentaren, Vergleichsoperatoren, mathematischen Operatoren, eckigen Klammern) bestehen kann.

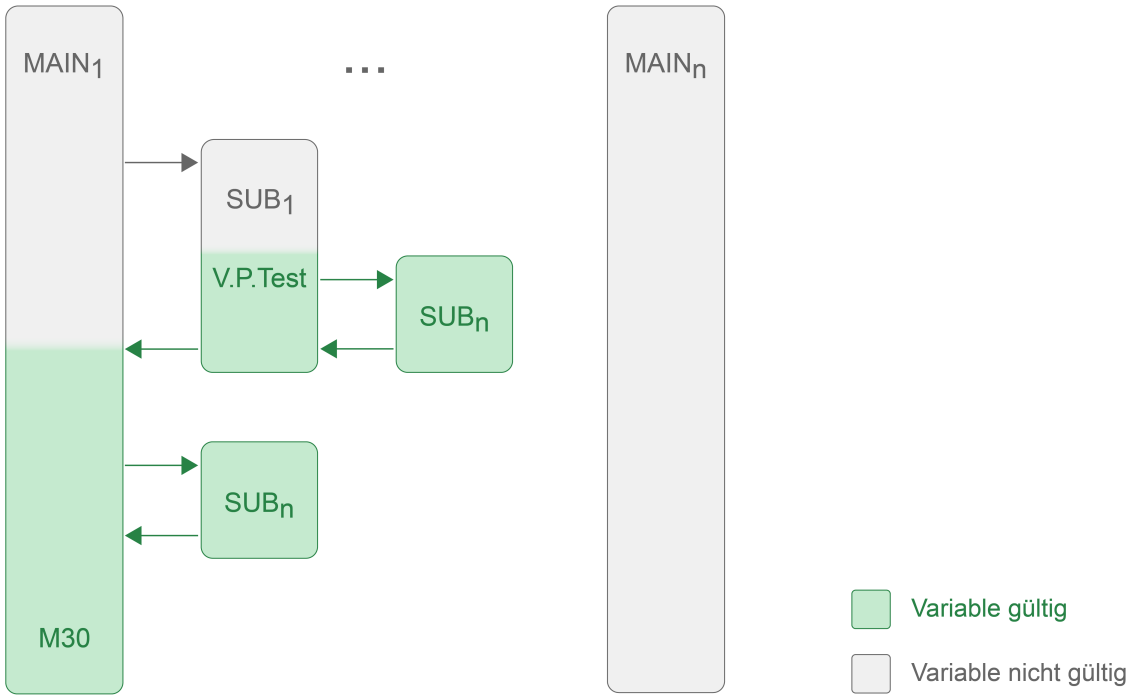
Programmierbeispiel

Global, nicht (Haupt-) programmübergreifend (V.P.)

Anlegen von V.P.-Variablen mit und ohne Zuweisung von Datentypen und Initialwerten.

```
:
#VAR
  V.P.VAR_1                              ;REAL64, 0.0
  V.P.VAR_2 : UNS32 = 200                ;UNS32, 200
  V.P.VAR_3 : REAL64 = 11.34             ;REAL64, 11.34
  V.P.VAR_4 : STRING[20] = "END_OK"      ;STRING, END_OK
  V.P.VAR_5 = 20                         ;REAL64, 20.0
#ENDVAR
:
XV.P.VAR_5         ;X20.0
:
```

Die maximale Anzahl eigendefinierter V.P.-Variablen ist fest vorgegeben [6] [▶ 894]-6-21. Beim Programmstart werden sämtliche Namen und Werte von V.P.-Variablen gelöscht.


MAIN1
...
MAINn
SUB1
V.P.Test
SUBn
SUBn
M30
Variable gültig
Variable nicht gültig

14.4.2 Global, Hauptprogramm übergreifend (V.S.)

Mit der Kennung "V.S. ..." ist es möglich, eigene Variablen zu definieren, die in allen Programmebenen und allen nachfolgenden (Haupt-) NC-Programmen noch unter dem gleichen Namen und mit den zuletzt belegten Werten bestehen. Auch nach RESET bleiben diese Variablen weiterhin gültig. Die Werte dieser Variablen können nur durch Überschreiben verändert werden, die Variablen selbst können nur durch einen Neustart der Steuerung oder mit #DELETE gelöscht werden. V.S.-Variablen können Werte im REAL-Format zugewiesen werden. Ab Version V2.11.2032.08 sind auch Deklarationen anderer Datentypen [▶ 620] mit Initialwerten möglich.

Syntax:

V.S.<FREE_DEF> globale, (Haupt-) programmübergreifende Variable

<FREE_DEF> Frei gewählter Name, der aus beliebigen Zeichen (außer Leerzeichen, Tabulatoren, Kommentaren, Vergleichsoperatoren, mathematischen Operatoren, eckigen Klammern) bestehen kann.

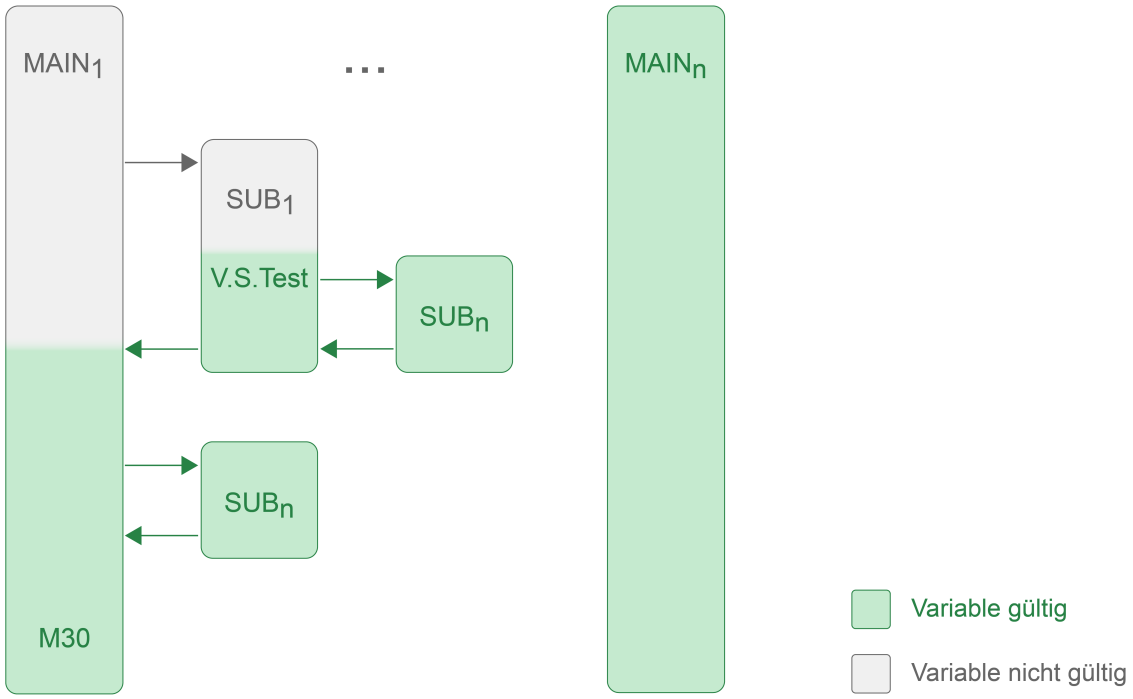
Programmierbeispiel

Global, (Haupt-) programmübergreifend (V.S.)

Anlegen von V.S.-Variablen mit und ohne Zuweisung von Datentypen und Initialwerten.

```
:
#VAR
  V.S.VAR1                        ;REAL64, 0.0
  V.S.VAR2 : UNS32 = 200          ;UNS32, 200
  V.S.VAR3 : REAL64 = 11.34       ;REAL64, 11.34
  V.S.VAR4 : BOOLEAN              ;BOOLEAN, FALSE or 0
  V.S.VAR[5] = [5,10,10,15,20]  ;Array with REAL64 values
#ENDVAR
:
XV.S.VAR[4]         ;X20.0
:
```

Die maximale Anzahl eigendefinierter V.S.-Variablen ist fest vorgegeben [6] [▶ 894]-6.22.


MAIN1
...
MAINn
SUB1
V.S.Test
SUBn
SUBn
M30
Variable gültig
Variable nicht gültig

14.4.3 Lokal, Unterprogramm übergreifend (V.L.)

Mit der Kennung "V.L. ..." ist es möglich, eigene Variablen zu definieren, die lokal in der aktuellen Programmebene und in direkt aufgerufenen Unterprogrammebenen gültig sind. Sie werden beim Verlassen (Rücksprung) der Programmebene, in der sie angelegt wurden, gelöscht.

Eine V.L.-Variable kann in einer tieferliegenden Programmebene mit gleichem Namen erneut definiert und mit einem eigenen Wert belegt werden. Dieser Wert ist dann bis zum Verlassen (Rücksprung) aus dieser Programmebene gültig. Danach besitzt die V.L.-Variable mit gleichem Namen wieder den ursprünglichen Wert.

V.L.-Variablen können Werte im REAL-Format zugewiesen werden. Ab Version V2.11.2032.08 sind auch Deklarationen anderer Datentypen [▶ 620] mit Initialwerten möglich.

V.L.-Variablen können auch als mehrdimensionalen Arrays angelegt werden. Es sind bis zu 4 Dimensionen möglich, z.B. V.L.TEST[1][2][3][4].

Syntax:

V.L.<FREE_DEF> lokale, (Unter-) programmübergreifende Variable

<FREE_DEF> Frei gewählter Name, der aus beliebigen Zeichen (außer Leerzeichen, Tabulatoren, Kommentaren, Vergleichsoperatoren, mathematischen Operatoren, eckigen Klammern) bestehen kann.

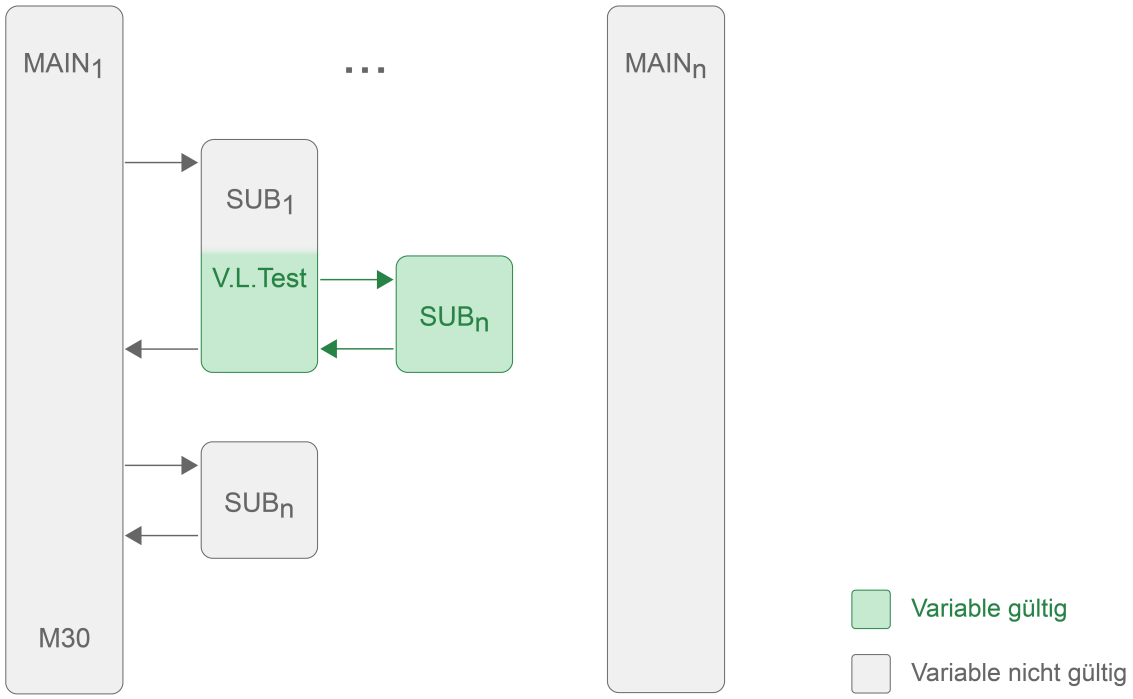
Programmierbeispiel

Lokal, (Unter-) programmübergreifend (V.L.)

Anlegen von V.L.-Variablen mit und ohne Zuweisung von Datentypen und Initialwerten.

```
:
#VAR
  V.L.LOC_VAR1                    ;REAL64, 0.0
  V.L.LOC_VAR2 : UNS32 = 200      ;UNS32, 200
  V.L.LOC_VAR3 : REAL64 = 11.34   ;REAL64, 11.34
  V.L.LOC_VAR4 : BOOLEAN          ;BOOLEAN, FALSE or 0
  V.L.LOC_VAR5 = 10               ;REAL64, 10.0
#ENDVAR
:
XV.L.LOC_VAR5        ;X10.0
:
```

Die maximale Anzahl eigendefinierter V.L.-Variablen ist fest vorgegeben [6] [▶ 894]-6.23. Beim Programmstart werden sämtliche Namen und Werte von V.L.-Variablen gelöscht.


MAIN1
...
MAINn
SUB1
V.L.Test
SUBn
SUBn
M30
Variable gültig
Variable nicht gültig

14.4.4 Zyklenvariablen (V.CYC.)

Versionshinweis

Diese Funktionalität ist verfügbar ab der Version V2.11.2032.08

Hinweis

Voraussetzung für die Nutzung der V.CYC.-Variablen ist die Reservierung von Speicher über den Kanalparameter P-CHAN-00418.

Die Kennung "V.CYC. ..." adressiert eigendefinierte Variablen, die vorzugsweise innerhalb von Zyklenprogrammen zu verwenden sind. Darüber hinaus können V.CYC.-Variablen aber auch in Standardhaupt- und Unterprogrammen genutzt werden. Die Deklaration umfasst neben der Angabe des Variablennamens auch die Festlegung des Datentyps. Die Variablen sind ab ihrer Deklaration in der aktuellen Programmebene und in allen weiteren direkt aufgerufenen Programmebenen (Unterprogrammen) gültig. Sie werden beim Verlassen (Rücksprung) der Programmebene, in der sie angelegt wurden, gelöscht (siehe Gültigkeit und Sichtbarkeit [▶ 630]).

V.CYC.-Variablen bieten auch die Möglichkeit, mehrdimensionale Arrays anzulegen. Es sind maximal 4 Dimensionen möglich, z.B. V.CYC.Test[1][2][3][4].

Syntax:

V.CYC.<FREE_DEF> zyklen- / programmspezifische Variable

<FREE_DEF> Frei gewählter Name, der aus beliebigen Zeichen (außer Leerzeichen, Tabulatoren, Kommentaren, Vergleichsoperatoren, mathematischen Operatoren, eckigen Klammern) bestehen kann.

Beispiel

Deklaration – V.CYC

```
%CYCLE_TEST.cyc
P1 = 3          ;erster Index des Arrays
P2 = 2          ;zweiter Index des Arrays
P3 = 10         ;vorgegebene maximale Stringlänge
#VAR
  V.CYC.TEST_A[P1][P2] : STRING[P3]
  V.CYC.TEST_B : STRING[P3] = "TEXT"
  V.CYC.TEST_C : REAL64 = 1.0
#ENDVAR
:
M30
```

14.4.4.1 Gültigkeit und Sichtbarkeit

Eine V.CYC.-Variable ist ab der Deklaration in der Programmebene und in den Programmebenen darunter sichtbar und nutzbar.

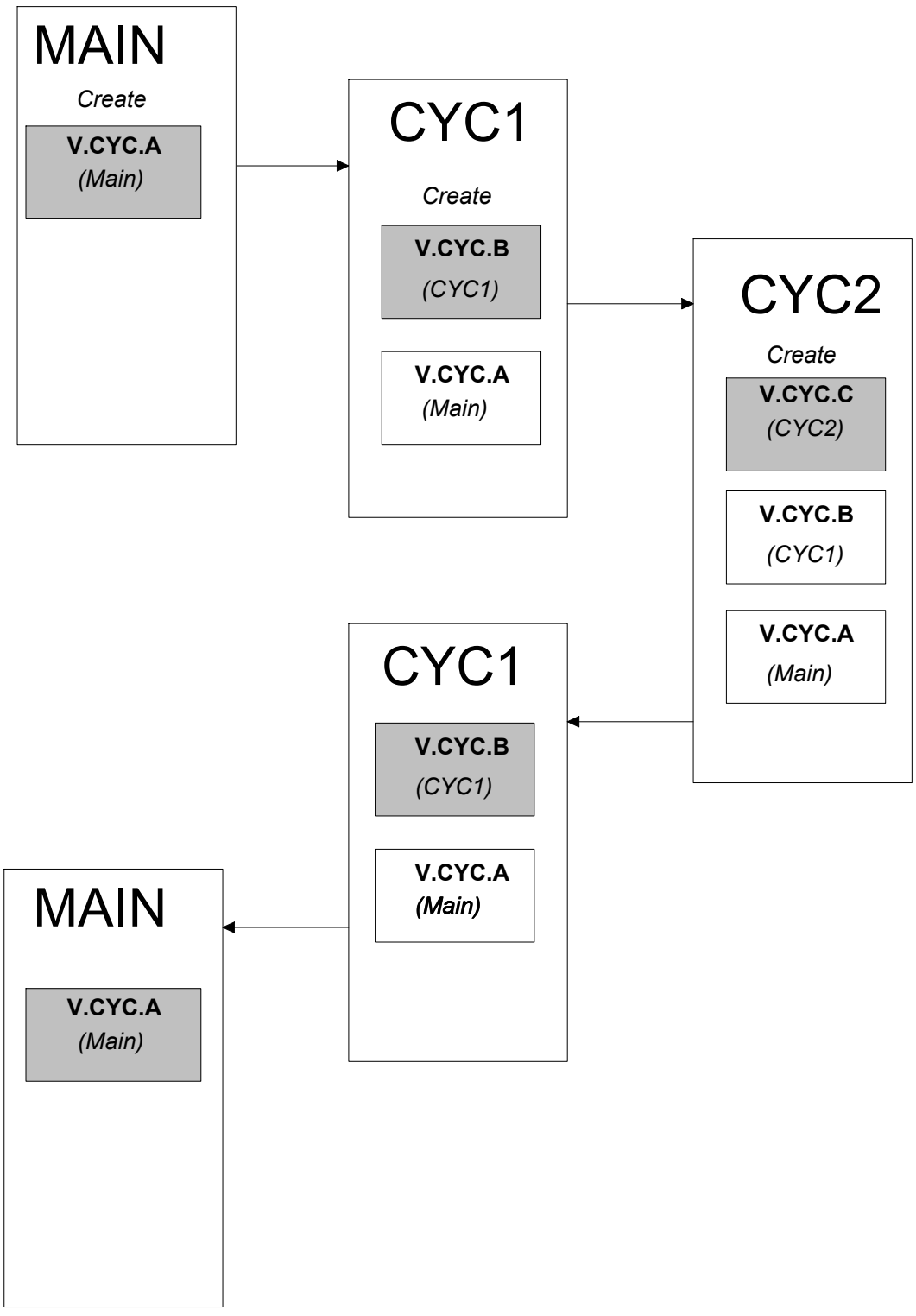


Abb. 156: Gültigkeit eigendefinierter V.CYC.-Variablen

Wird in einer tieferen Programmebene erneut eine V.CYC.-Variable mit dem gleichen Namen angelegt, so ist jeweils die 'lokalste' Deklaration sichtbar und nutzbar.

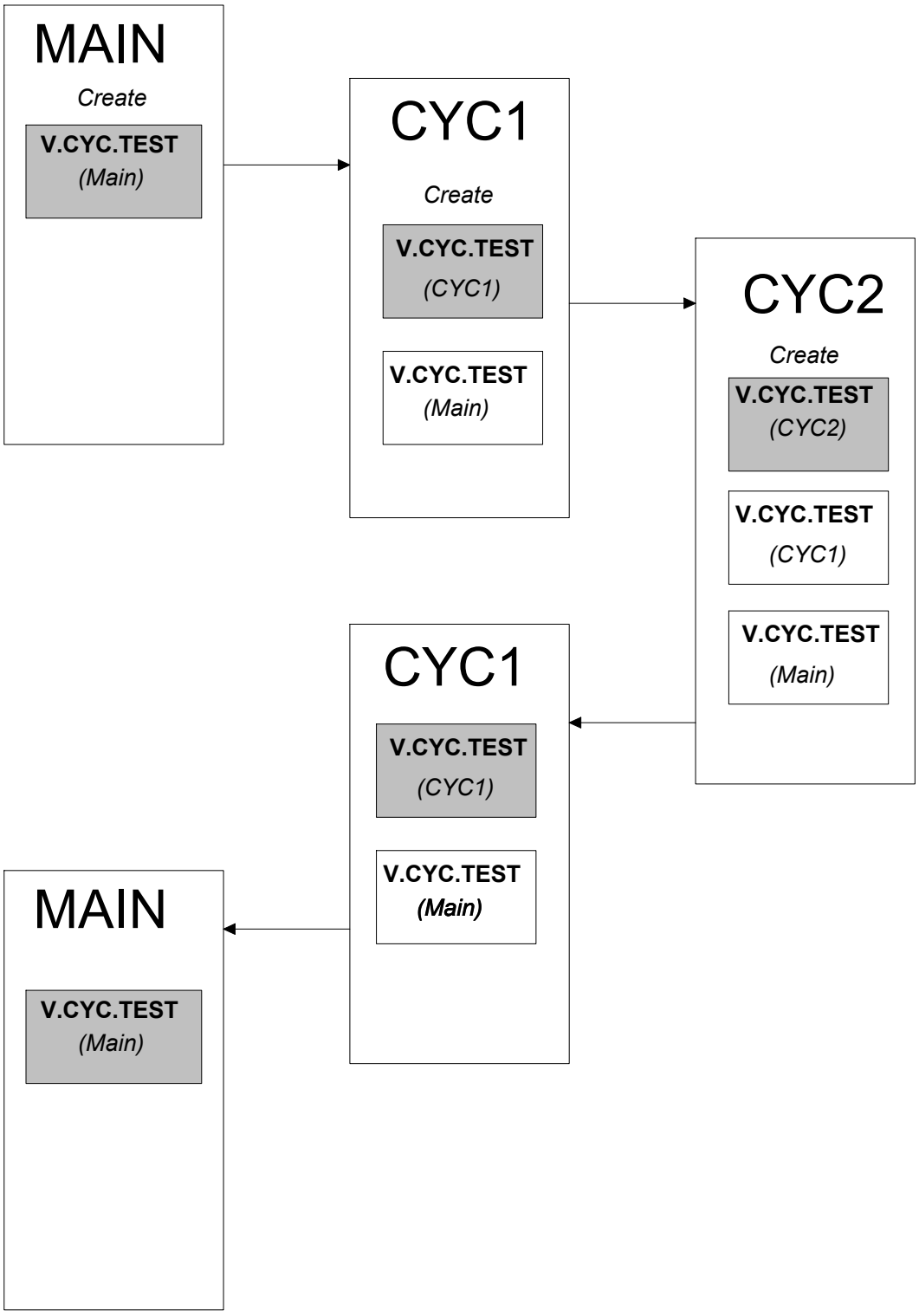


Abb. 157: Gültigkeit gleichnamiger V.CYC.-Variablen

14.4.4.2 Löschen von V.CYC.-Variablen

Eine V.CYC.-Variable wird automatisch gelöscht, wenn der Zyklus bzw. die Programmebene, in der die Variable angelegt wurde, verlassen wird.

Alternativ kann eine V.CYC.-Variable auch über den NC-Befehl #DELETE in der jeweiligen Programmebe gelöscht werden, in der sie angelegt wurde.

Z.B.

#DELETE V.CYC.MyVar

Hinweis

Mit #DELETE können nur V.CYC.-Variablen in der Programmebene gelöscht werden, in der sie angelegt wurden.

Das Löschen einer nicht existierenden V.CYC.-Variable wird ignoriert. Es erfolgt keine Warnung oder Fehlermeldung.

Wird in einer Programmebene versucht, eine existierende V.CYC.-Variable zu löschen, die in einer Programmebene höher angelegt wurde, so wird der Fehler P-ERR-20933 ausgegeben.

14.5 Externe Variablen (V.E.) (#INIT V.E.)

Mit dem Befehl "V.E. ..." kann im NC-Programm auf externe Adressen geschrieben und/oder von externen Adressen gelesen werden. Dies wird durch die direkte, speichergekoppelte Kommunikation zwischen dem NC-Kanal und externen Teilnehmern, typischerweise der SPS, ermöglicht.

Der Zugriff vom NC-Kanal aus kann synchron durch den Interpolator, oder asynchron durch den Decoder durchgeführt werden.

Hinweis

Ein Lesen einer synchronen V.E.-Variable verursacht ein Leeren des NC-Kanals.

Dies ist bei aktiver WRK, aktivem Polynomüberschleifen oder aktivem HSC-Modus nicht zulässig.

Weitere Informationen zur Konfiguration und Parametrierung sind in der Dokumentation der externen Variablen beschrieben [8] [▶ 894].

Programmierbeispiel

Externe Variablen (V.E.)

```
N100 $IF V.E.EXT1 >= 100                (Entsprechend dem Wert von
V.E.EXT1)
                                        (wird in die verschiedenen Fälle)
                                        (verzweigt.)
N110 G01 X100 Y100 F1000

N120 $ELSE
N130 G01 X100 YV.E.EXT1 F1000           (Geradeninterpolation in Y-Rich-
tung)
                                        (mit dem Wert von V.E.EXT1)
N140 $ENDIF
N150 V.E.EXT1 = V.A.ABS.X               (Der externen Variablen wird die)
                                        (absolute X-Koordinate zugewiesen)
N160 G01 XV.E.EXT2                      (Geradeninterpolation in X-Rich-
tung)
                                        (mit dem Wert von V.E.EXT2)
```

Nach Steuerungshochlauf sind die konfigurierten V.E.-Variablen mit "Null" initialisiert.

Danach können V.E.-Variablen im NC-Programm über den Befehl #INIT bei Bedarf erneut initialisiert werden. Dem Befehl können eine oder mehrere V.E.-Variablen folgen, die vollständig abgelöscht werden. Neben einzelnen V.E.-Variablen können auch komplette V.E.-Arrays und V.E.-Strukturen sowie Unterelemente von V.E.-Strukturen initialisiert werden.

Syntax:

#INIT V.E.<*name*> {**, V.E.**<*name*>}

Achtung

Zugriffsrechte:

Hat eine Variable nur Leserechte, kann diese nicht mit dem #INIT-Befehl initialisiert werden. Gleiches gilt für V.E.-Strukturen, wenn mindestens ein Unterelement enthalten ist, das nur gelesen werden kann.

Achtung

Synchrone V.E.-Variablen:

Sobald eine V.E.-Struktur eine synchrone Variable enthält, ist der gesamte Initialisierungsvorgang mit #INIT synchron, d.h. er wird erst im Interpolatorkontext ausgeführt. Mögliche asynchrone Unterelemente sind also von diesem Vorgang ebenfalls betroffen, da diese möglicherweise bei einem nachfolgenden lesenden Zugriff noch nicht neu initialisiert sind!

Um in diesen Fällen eine vollständige Synchronität zu erreichen, sollte der Benutzer daher manuell vor dem #INIT-Befehl einen #FLUSH WAIT-Befehl programmieren.

Tipp:

Bei Verwendung des #INIT-Befehls wird empfohlen, V.E.-Strukturvariablen so anzulegen, dass alle Elemente vollständig synchron oder vollständig asynchron sind!

Programmierbeispiel

Initialisieren einzelner V.E.-Variablen:

```
Nxx #INIT V.E.EXT1, V.E.EXT2, V.E.EXT3
```

Initialisieren einer V.E.-Arrayvariable:

```
Nxx #INIT V.E.ARRAY1
```

Initialisieren bestimmter V.E.-Arrayvariablen:

```
Nxx #INIT V.E.ARRAY1[5], V.E.ARRAY1[8], V.E.ARRAY1[20]
```

Initialisieren einer V.E.-Strukturvariable:

```
Nxx #INIT V.E.STRUCT1
```

Initialisieren bestimmter Elemente einer V.E.-Strukturvariable:

```
Nxx #INIT V.E.STRUCT1.NBR_POINTS, V.E.STRUCT1.POINTS
```

Kombiniertes Initialisieren von V.E.-Variablen:

```
Nxx #INIT V.E.EXT2, V.E.ARRAY1[5], V.E.STRUCT1.POINTS
...
```

15 Spindelprogrammierung

Die Spindelprogrammierung erfolgt gemäß der in ISO bzw. ISO + Erweiterungen festgelegten konventionellen Standardsyntax. Dies ist insbesondere aus Kompatibilitätsgründen und wegen bestimmter Standardfunktionalitäten wie Drehen, Gewindebohren, Getriebeschalten etc. erforderlich.

Um den Anforderungen neuer Maschinenkonzepte und Fertigungstechnologien bzgl. einer flexiblen Spindelprogrammierung gerecht zu werden, besteht für jede im Kanal vorhandene Spindel zusätzlich die Möglichkeit einer achsspezifischen Programmierung.

Durch diese Syntax können in Multispindelsystemen in einem NC-Satz gleichzeitig mehrere Spindeln unabhängig voneinander angesprochen werden (P-CHAN-00082, [6] [▷ 894]-8.8). Dabei ist zu beachten, dass immer nur eine Spindel, die s.g. „Hauptspindel“ sowohl in der Standardsyntax als auch in der spindelspezifischen Syntax programmiert werden kann.

Alle weiteren Spindeln können nur über die spindelspezifische Syntax angesprochen werden (siehe Kapitel Spindeloverride DIN-Syntax (G167) [▷ 689]).

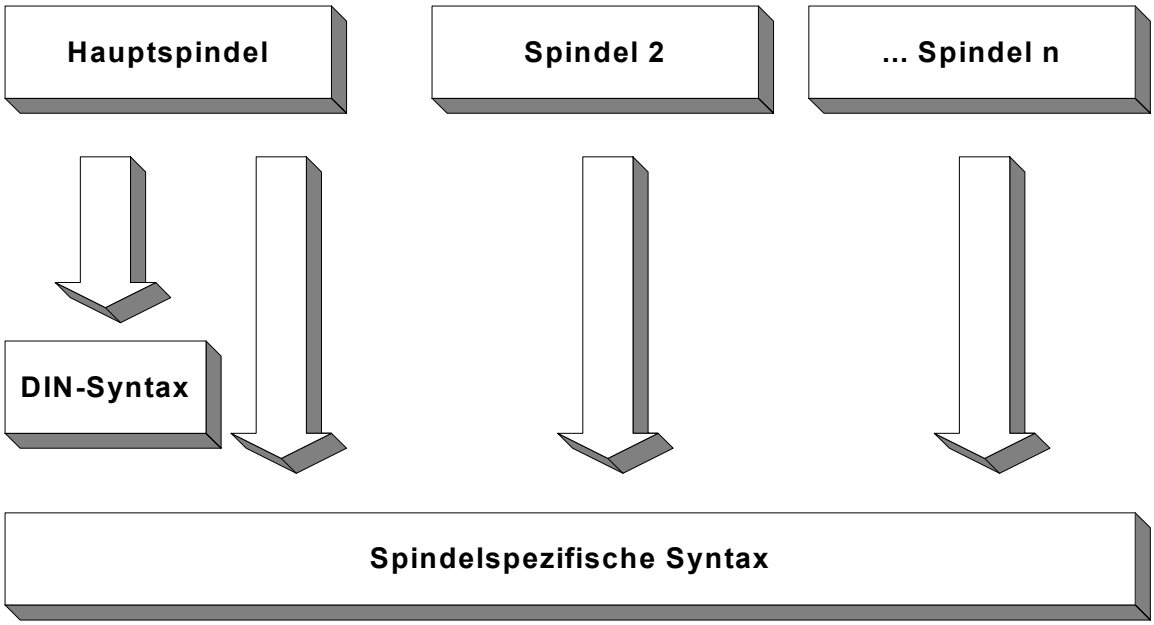


Abb. 158: Korrekte Verwendung von DIN-Syntax und spindelspezifischer Syntax

Die Definition der Spindeln und die Festlegung der Hauptspindel erfolgt über die Parametrierung von Kanalparameter [1] [▷ 894]-3. Diese Konfiguration liegt nach dem Hochlauf der Steuerung vor. Die Hauptspindel kann im NC-Programm über einen NC-Befehl (#MAIN SPINDLE, siehe Kapitel Programmierbarer Spindeloverride (OVERRIDE) [▷ 704]) geändert werden.

Die nachfolgende Tabelle zeigt, welche NC-Befehle im Zusammenhang mit der Spindelprogrammierung nur in der DIN-Syntax zu verwenden und welche auch innerhalb der erweiterten spindelspezifischen Syntax zulässig sind.

Übersicht der Spindelbefehle

Beschreibung	DIN-Syntax	Spindel.-spezif. Syntax
Spindel-M-Funktionen	M3, M4, M5, M19	M3, M4, M5, M19
Drehzahl	S	REV
Position	S.POS	POS
Anwenderspezifische M/H-Funktionen	Mxx/Hxx	Mxx/Hxx
Getriebeschalten (mechanisch)	M40-M45	
Gewindeschneiden	G33	
Gewindebohren	G63	
Drehen	G95, G96, G97, G196	
C-Achse	#CAX	
Getriebeschalten (neue Daten)	G112	
Referenzpunktfahrt	G74	G74
Override 100%	G167	G167
Explizites Anfordern einer Spindelachse		CALLAX
Explizites Abgeben einer Spindelachse		PUTAX
Übernahme Werkzeug-Dynamikdaten		GET_DYNAMIC_DATA
Vorsteuerung		G135, G136, G137
Vorschubkopplung		FEED_LINK

15.1 Parametrierung von Spindeln

15.1.1 Achsparameter

Sowohl für geregelte als auch für gesteuerte (SPS-) Spindeln sind zur Angabe der spindelspezifischen Parameter eigene Achsparameterdatensätze zu belegen [2] [▶ 894].

15.1.2 Kanalparameter

Für die Programmierung von Spindeln im NC-Programm sind spezifische Parametrierungen [1] [▶ 894] notwendig.

Jede Spindel, die von einem Kanal aus angesprochen werden soll, muss in diesem auch bekannt gemacht werden. Dazu wird für jede Spindel ein String (Achsbezeichnung) und die entsprechende logische Achsnummer definiert. Die Achsbezeichnungen der Spindeln sind frei wählbar, sie müssen jedoch immer mit “S“ beginnen (z.B. S, S_MAIN, S1, SPINDEL_1).

Für die Spindel-M-Funktionen (M3, M4, M5, M19) sowie für das S-Wort müssen zusätzlich die Synchronisationsarten spindelspezifisch parametriert werden. Dazu muss die Bedeutung von M3, M4, M5 und M19 entsprechend geschaltet sein (P-CHAN-00098).

Hinweis

Die Synchronisationsart der S-Funktion ist wirkungslos, wenn im NC-Satz eine Spindel M-Funktion programmiert wurde. Eine Synchronisation findet dann nur entsprechend der Einstellungen für die Spindel M-Funktion statt. Es gilt folgende Prioritätsreihenfolge:

M19 > M3/M4/M5 > S

Sollen die Spindeln im Simulationsmode „Fertigungszeitberechnung“ berücksichtigt werden, so können die dazu notwendigen Daten ebenfalls spindelspezifisch parametriert werden.

Um zur bisherigen Programmierung kompatibel zu bleiben muss eine Spindel als s.g. Hauptspindel deklariert werden (P-CHAN-00051, P-CHAN-00053). Diese Hauptspindel kann dann zusammen mit bestimmten Standardfunktionalitäten (z.B. Gewindebohren, Getriebeschalten etc.) in der herkömmlichen DIN-Syntax programmiert werden. Auch wenn nur eine Spindel im System vorhanden ist, muss diese als Hauptspindel konfiguriert werden.

Durch setzen eines Flags kann für die Hauptspindel zusätzlich die optionale Funktion Getriebeschalten freigeschaltet werden (P-CHAN-00052).

Die in den Kanalparametern [1] [▶ 894] definierte Defaultbelegung steht nach dem Hochlauf der Steuerung zur Verfügung.

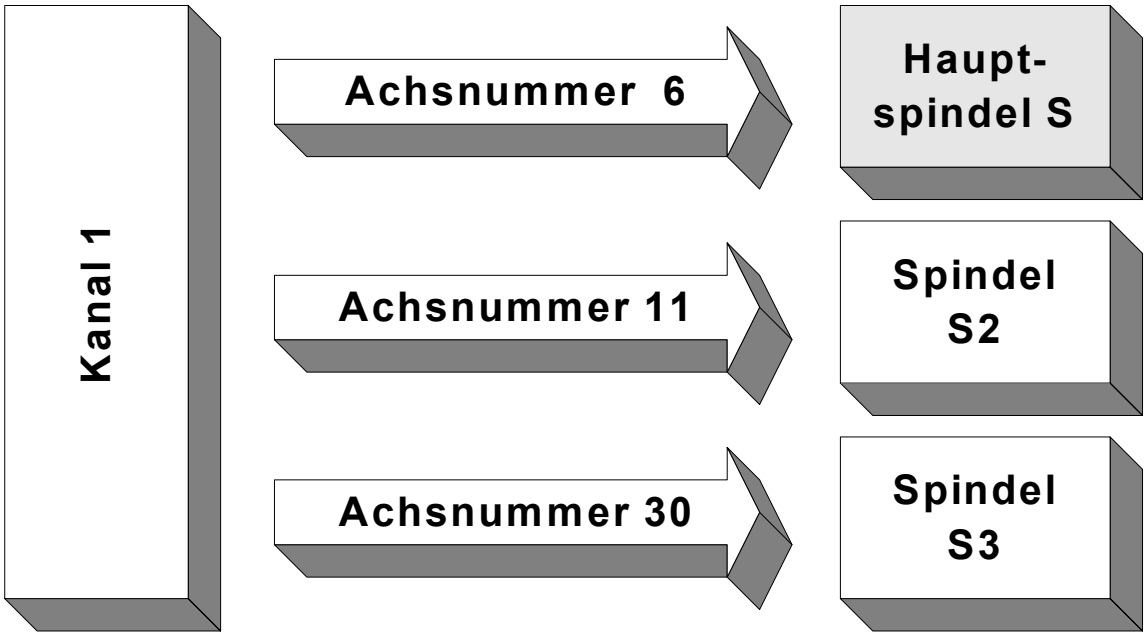
Beispiel 1:

Konfiguration eines 1-kanaligen Systems mit 3 Spindeln. Die Spindel mit der Achsnummer 6 soll Hauptspindel sein. Getriebeschalten dieser Spindel ist deaktiviert.

Kanalparametersatz [1] [▶ 894]:

```
:
spdl_anzahl                                    3
:
main_spindle_ax_nr                             6 -> -> ->-
main_spindle_name                              S          |
main_spindle_gear_change                       0          |
#                                                         |
spindel[0].bezeichnung                         S1         |
spindel[0].log_achs_nr                         6 -< -< -<-
spindel[0].s_synch                             0x00000001
spindel[0].m3_synch                            0x00000002
spindel[0].m4_synch                            0x00000004
spindel[0].m5_synch                            0x00000008
spindel[0].m19_synch                           0x00000001
spindel[1].bezeichnung                         S2
spindel[1].log_achs_nr                         11
spindel[1].s_synch                             0x00000001
spindel[1].m3_synch                            0x00000002
spindel[1].m4_synch                            0x00000004
spindel[1].m5_synch                            0x00000008
spindel[1].m19_synch                           0x00000001
spindel[2].bezeichnung                         S3
spindel[2].log_achs_nr                         30
spindel[2].s_synch                             0x00000001
spindel[2].m3_synch                            0x00000002
spindel[2].m4_synch                            0x00000004
spindel[2].m5_synch                            0x00000008
spindel[2].m19_synch                           0x00000001
:
```

Nach dem Hochlauf ist die Spindel mit der logischen Achsnummer 6 die Hauptspindel. Sie wird über den Spindelnamen „S“ angesprochen und kann in herkömmlicher DIN-Syntax oder in spindelspezifischer Syntax programmiert werden. Die Spindeln „S2“ und „S3“ können nur in spindelspezifischer Syntax programmiert werden.



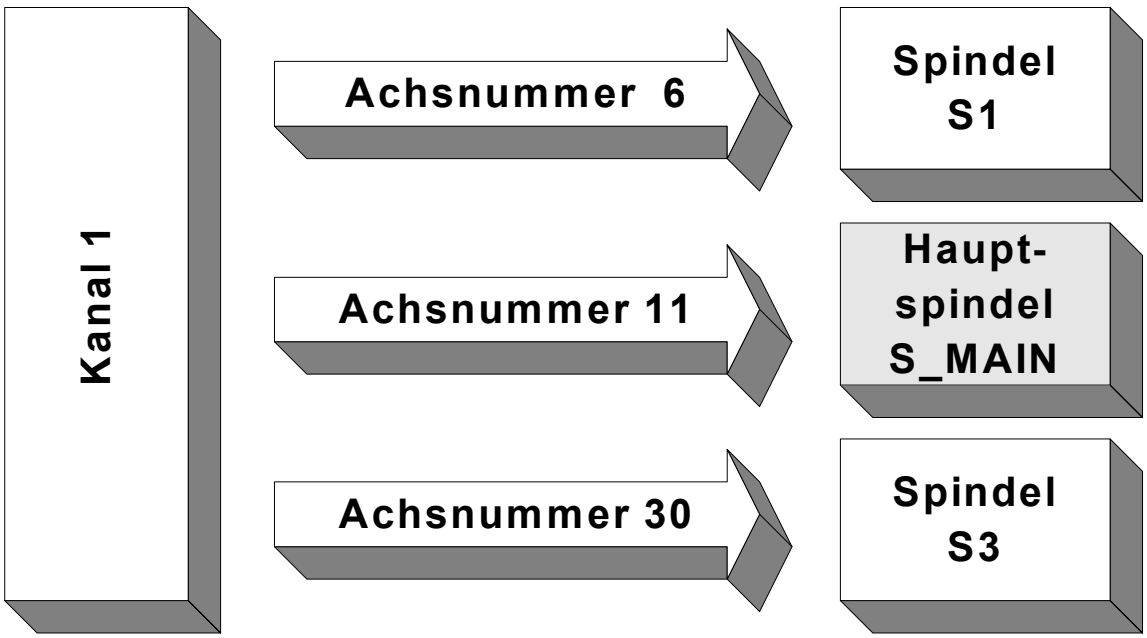
Beispiel 2:

Konfiguration eines 1-kanaligen Systems mit 3 Spindeln. Die Spindel mit der Achsnummer 11 soll Hauptspindel sein. Getriebeschalten dieser Spindel ist deaktiviert.

Kanalparametersatz [1] [▶ 894]:

```
:
spdl_anzahl                                    3
:
main_spindle_ax_nr                             11 -> -> -> ->
main_spindle_name                              S_MAIN         |
main_spindle_gear_change                       0              |
#                                                             |
spindel[0].bezeichnung                         S1             |
spindel[0].log_achs_nr                         6              |
spindel[0].s_synch                             0x00000001     |
spindel[0].m3_synch                            0x00000002     |
spindel[0].m4_synch                            0x00000004     |
spindel[0].m5_synch                            0x00000008     |
spindel[0].m19_synch                           0x00000001     |
spindel[1].bezeichnung                         S2             |
spindel[1].log_achs_nr                         11 -< -< -< -<
spindel[1].s_synch                             0x00000001
spindel[1].m3_synch                            0x00000002
spindel[1].m4_synch                            0x00000004
spindel[1].m5_synch                            0x00000008
spindel[1].m19_synch                           0x00000001
spindel[2].bezeichnung                         S3
spindel[2].log_achs_nr                         30
spindel[2].s_synch                             0x00000001
spindel[2].m3_synch                            0x00000002
spindel[2].m4_synch                            0x00000004
spindel[2].m5_synch                            0x00000008
spindel[2].m19_synch                           0x00000001
:
```

Nach dem Hochlauf ist die Spindel mit der logischen Achsnummer 11 die Hauptspindel. Sie wird über den Spindelnamen „S_MAIN“ angesprochen und kann in herkömmlicher DIN-Syntax oder in spindelspezifischer Syntax programmiert werden. Die Spindeln „S1“ und „S3“ können nur in spindelspezifischer Syntax programmiert werden.



Beispiel 3:

Konfiguration eines 2-kanaligen Systems mit insgesamt 3 Spindeln:

Kanal 1: 3 Spindeln. Spindel mit der Achsnummer 11 soll Hauptspindel sein.

Kanalparametersatz [1] [▶ 894]:

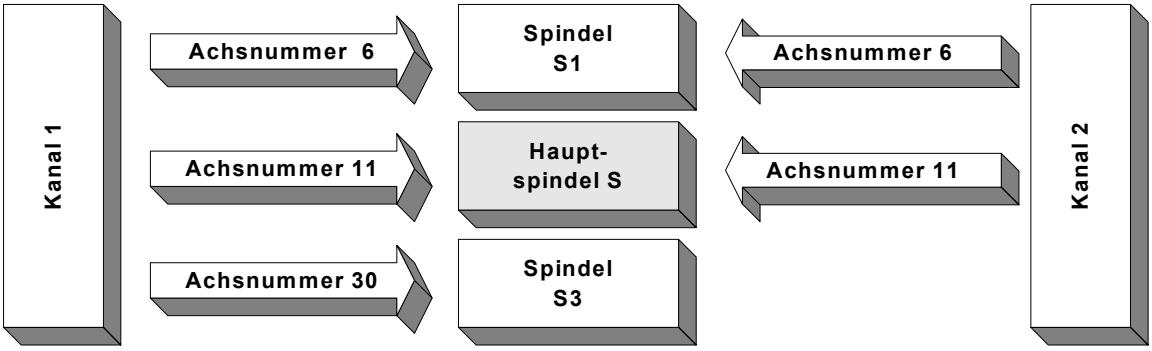
```
:
spdl_anzahl                                    3
:
main_spindle_ax_nr                             11 -> -> -> ->
main_spindle_name                              S              |
main_spindle_gear_change                       0              |
#                                                             |
spindel[0].bezeichnung                         S1             |
spindel[0].log_achs_nr                         6              |
spindel[0].s_synch                             0x00000001     |
spindel[0].m3_synch                            0x00000002     |
spindel[0].m4_synch                            0x00000004     |
spindel[0].m5_synch                            0x00000008     |
spindel[0].m19_synch                           0x00000001     |
spindel[1].bezeichnung                         S2             |
spindel[1].log_achs_nr                         11 -< -< -< -<
spindel[1].s_synch                             0x00000001
spindel[1].m3_synch                            0x00000002
spindel[1].m4_synch                            0x00000004
spindel[1].m5_synch                            0x00000008
spindel[1].m19_synch                           0x00000001
spindel[2].bezeichnung                         S3
spindel[2].log_achs_nr                         30
spindel[2].s_synch                             0x00000001
spindel[2].m3_synch                            0x00000002
spindel[2].m4_synch                            0x00000004
spindel[2].m5_synch                            0x00000008
spindel[2].m19_synch                           0x00000001
:
```

Kanal 2: 2 Spindeln. Spindel mit der Achsnummer 11 soll Hauptspindel sein.

Kanalparametersatz [1] [▶ 894]:

```
:
spdl_anzahl                                          2
:
main_spindle_ax_nr                                   11 -> -> ->-
main_spindle_name                                    S           |
main_spindle_gear_change                             0           |
#                                                                |
spindel[0].bezeichnung                               S1          |
spindel[0].log_achs_nr                               6 -< -< -<-
spindel[0].s_synch                                   0x00000001
spindel[0].m3_synch                                  0x00000002
spindel[0].m4_synch                                  0x00000004
spindel[0].m5_synch                                  0x00000008
spindel[0].m19_synch                                 0x00000001
spindel[1].bezeichnung                               S2
spindel[1].log_achs_nr                               11
spindel[1].s_synch                                   0x00000001
spindel[1].m3_synch                                  0x00000002
spindel[1].m4_synch                                  0x00000004
spindel[1].m5_synch                                  0x00000008
spindel[1].m19_synch                                 0x00000001
:
```

Nach dem Hochlauf kann von beiden Kanälen die Spindel mit der logischen Achsnummer 11 als Hauptspindel über den Spindelnamen „S“ angesprochen werden. Sie kann in herkömmlicher DIN-Syntax oder in spindelspezifischer Syntax programmiert werden. Die Spindel „S1“ kann ebenfalls von beiden Kanälen aus in spindelspezifischer Syntax programmiert werden. Spindel „S3“ ist ausschließlich in Kanal 1 verfügbar, in Kanal 2 ist diese Spindel unbekannt.



15.2 Programmierung in DIN-Syntax

15.2.1 Die Spindel-M-Funktionen

15.2.1.1 Spindel bewegen in DIN-Syntax (M3/M4/M5)

Syntax:

M03	Spindeldrehung im Uhrzeigersinn (cw)	modal
M04	Spindeldrehung im Gegenuhrzeigersinn (ccw)	modal
M05	Spindel stoppen	modal

Die Spindel-M-Funktionen M03, M04, M05 legen die Spindelbetriebsart fest und sind im Zusammenhang mit dem S-Wort zu verwenden (Kap. Spindeldrehzahl (S-Wort) [▶ 645]). Sie sind haltend wirksam und dürfen jeweils nur alleine im NC-Satz programmiert werden.

Die Spindeldrehung wird aktiviert, wenn M03 oder M04 programmiert wurde und eine gültige Drehzahl gesetzt ist.

Mit M05 wird die Spindeldrehung gestoppt. Es ist zu beachten, dass diese Spindel-M-Funktion der Default-Spindelmodus nach Steuerungshochlauf und erstem Programmstart ist.

Wird am Programmende kein M05 gesetzt, so dreht die Spindel weiter.

Programmierbeispiel

Spindel bewegen (M3/M4/M5)

```
N10          S1000      (Drehzahl 1000 U/min wird gespeichert, keine)
                        (Spindeldrehung da M05 Default)
N20    M03               (Spindeldrehung cw mit 1000 U/min)
N30    M04               (Spindeldrehung ccw mit 1000 U/min)
N40          S500       (Spindeldrehung ccw mit 500 U/min)
N50    M05   S300       (Spindelstopp, Drehzahl 300 U/min wird)
                        (gespeichert)
N60    M04               (Spindeldrehung ccw mit 300 U/min)
N70    M05               (Spindelstopp )
N80    M03   S1000       (Spindeldrehung cw mit 1000 U/min)
N90    M30               (Programmende)
```

Kanalparametersatz [1] [▶ 894]:

Für M3, M4, M5 müssen die Synchronisationsarten spindelspezifisch festgelegt werden. Bei der Synchronisationsart „0“ (NO_SYNCH) wird die M-Funktion nicht ausgeführt.

```
:
spindel[0].bezeichnung                          S1
spindel[0].log_achs_nr                          6
spindel[0].s_synch                              0x00000001
spindel[0].m3_synch                             0x00000002
spindel[0].m4_synch                             0x00000002
spindel[0].m5_synch                             0x00000008
spindel[0].m19_synch                            0x00000001
```

15.2.1.2 Spindel positionieren in DIN-Syntax (M19, *.POS)

Syntax:

M19	Spindel positionieren	nicht modal
<*Spindelname*>**.POS**=..	Spindelposition	modal

Die Positionierung der Spindel kann mit folgender Syntax dargestellt werden:

M19 [<*Spindelname*>**.POS**=..] [**M03** | **M04**] [<*Spindelname*>=..]

M19	Spindel positionieren
<*Spindelname*>.POS=..	Spindelposition in [°]. Bezeichnung der Hauptspindel gemäß P-CHAN-00053
<*Spindelname*>=..	Spindeldrehzahl in [U/min]. Bezeichnung der Hauptspindel gemäß P-CHAN-00053

M03/M04 bzw. die Spindeldrehzahl im gleichen NC-Satz sind optional. Es muss jedoch eine gültige Spindeldrehzahl (> 0) gesetzt sein.

Die Spindelposition ist haltend und muss bei einer erneuten Programmierung von M19 nicht nochmals angegeben werden. Wurde bisher noch keine Spindelposition programmiert, so wird per default auf Position „Null“ gefahren.

Rotiert die Spindel nicht, wird die Positionierung mit dem kürzesten Verfahrweg durchgeführt.

Spindelpositionierung mit M19 ist nur für lagegeregelte Spindeln erlaubt.

Programmierbeispiel

Spindel positionieren (M19, *.POS)

In den folgenden Beispielen wird die Spindel jeweils auf 180° positioniert. Das "="-Zeichen ist optional:

```
M19 S.POS180
M19 S.POS 180
M19 S.POS=180
M19 SPINDEL.POS=180
M19 S1.POS=180
```

Die Spindel rotiert beim Positionieren nicht. Es wird der kürzeste Verfahrweg berechnet.

```
N10 M05 S100        (Spindelstopp, Drehzahl 100 U/min wird)
                    (gespeichert)
N20 M19 S.POS180    (Positionieren mit 100 U/min auf Position 180)
                    (Die Drehrichtung ergibt sich aus dem kürzesten) (Verfahrweg)
N30 M19 S200        (Die Drehrichtung ergibt sich aus dem kürzesten)
S.POS90             (Verfahrweg)
                    (Positionieren mit 200 U/min ccw auf Position 90)
```

Kanalparametersatz [1] [▶ 894]:

Für M19 muss die Synchronisationsart spindelspezifisch festgelegt werden. Bei der Synchronisationsart „0“ (NO_SYNCH) wird die M-Funktion nicht ausgeführt.

```
:
spindel[0].bezeichnung                    S1
spindel[0].log_achs_nr                    6
spindel[0].s_synch                        0x00000001
spindel[0].m3_synch                       0x00000002
spindel[0].m4_synch                       0x00000002
spindel[0].m5_synch                       0x00000008
spindel[0].m19_synch                      0x00000001
```

15.2.2 Spindeldrehzahl (S)

Syntax:

<*Spindelname*> .. modal

<*Spindelname*>.. Bezeichnung der Hauptspindel gemäß P-CHAN-00053 mit Spindeldrehzahl in [U/min]

Bei der Konfigurierung kann der Hauptspindel in den Kanalparametern (P-CHAN-00053) ein String zugeordnet werden. Um Mehrdeutigkeiten zu vermeiden, muss nach allen Spindelbezeichnungen, die mehr als ein Zeichen umfassen, vor der Drehzahlangabe ein Gleichheitszeichen stehen.

Dem S-Wort können Werte direkt oder per Parameter zugewiesen werden, wobei auch Dezimalzahlen (REAL-Format) zulässig sind.

In Verbindung mit den Spindel-M-Funktionen sind beim S-Wort folgende Verwendungsarten zu unterscheiden:

1. S-Wort in Verbindung mit M03, M04, M19:
   Wird das S-Wort bzw. der dafür verwendete String in Verbindung mit M03/M04 oder M19 programmiert, so wird der dem S-Wort folgende Wert als Spindeldrehzahl interpretiert und an die Spindel ausgegeben.
2. S-Wort in Verbindung mit M05:
   In Verbindung mit M05 wird der dem S-Wort folgende Wert als Spindeldrehzahl in die Arbeitsdaten übernommen, aber nicht an die Spindel ausgegeben.

Das S-Wort alleine erzeugt noch keine Bewegung im NC-Programm. Hierfür muss ein Spindelmodus M03, M04, M19 bekannt sein. Entsprechend wird auch die Programmierung von M03, M04 und M19 erst eine Bewegung bewirken, wenn das S-Wort gesetzt wird (> 0, Analogie zu G01, G02 und G03, bei denen erst verfahren wird, wenn der Vorschub und die zu verfahrenden Achsen angegeben wurden).

Hinweis

Bei einem negativen S-Wert wird eine Fehlermeldung ausgegeben.

Nur im Zusammenhang mit G63 (Gewindebohren) ist ein negativer S-Wert erlaubt, da damit die Umkehr der Drehrichtung beim Herausfahren aus der Gewindebohrung veranlasst wird

Programmierbeispiel

Programmierung mit Spindel S

```
N10          S300     (Drehzahl 300 U/min wird gespeichert)
N20    M04             (Spindeldrehung ccw mit 300 U/min)
N30    M03   S1000     (Spindeldrehung cw mit 1000 U/min)
N40          S500     (M03 aktiv, somit Spindeldrehung cw mit 500 U/min)
N50    M05   S100     (Spindelstopp, Drehzahl 100 U/min wird gespeichert)
N60    M04             (Spindeldrehung ccw mit 100 U/min)
N70    M05             (Spindelstopp)
N80    M30             (Programmende)
```

Kanalparametersatz [1] [▶ 894]:

Für das S-Wort muss die Synchronisationsart spindelspezifisch festgelegt werden. Bei der Synchronisationsart „0“ (NO_SYNCH) wird eine Fehlermeldung erzeugt, da ein S-Wort nicht unberücksichtigt bleiben darf.

```
:
spindel[0].bezeichnung                    S1
spindel[0].log_achs_nr                    6
spindel[0].s_synch                        0x00000001
spindel[0].m3_synch                       0x00000002
spindel[0].m4_synch                       0x00000002
spindel[0].m5_synch                       0x00000008
spindel[0].m19_synch                      0x00000001
```

Programmierbeispiel

Spindeldrehzahl (S-Wort)

```
N10   M03   S100              (Spindeldrehung cw mit 100 U/min)
N20   M19   S.POS90           (Mit 100 U/min cw auf Position 90)
N30   M04                     (Spindeldrehung ccw mit 100 U/min)
N40   M19   S200   S.POS 180  (Mit 200 U/min ccw auf Position 180)
N50   M05   S150              (Spindelstopp, Drehzahl 150 U/min)
                              (wird gespeichert)
N60   M19   S.POS=135         (Positionieren mit 150 U/min auf)
                              (dem kürzesten Weg auf Position 135)
N70   M03   S300              (Spindeldrehung cw mit 300 U/min)
N80   M19   S200   S.POS270   (Mit 200 U/min cw auf Position 270)
N90   M03   S400   S.POS45    (Spindeldrehung cw mit 400 U/min,)
                              (Position 45 wird gespeichert)
N100  M19                     (Mit 400 U/min cw auf Position 45)
N110  M04   S800              (Spindeldrehung ccw mit 800 U/min)
N120  S1200                   (Spindeldrehung ccw mit 1200 U/min)
N130  M5                      (Spindelstopp)
N140  M03                     (Spindeldrehung cw mit 1200 U/min)
N150  M19                     (Mit 1200 U/min cw auf Position 45)
```

15.2.3 Getriebeschalten von Spindeln (M40 - M45)

Die Programmierung eines Schaltens des Spindelgetriebes erfolgt mit M40...M45. Diese M-Funktionen definieren maximal 6 Getriebestufen. Für jede Getriebestufe muss in der zugehörigen Spindelachsliste [AXIS] ein eigener Getriebedatensatz parametriert sein.

Die M-Funktionen können zusammen mit der Spindeldrehzahl und der M-Funktion für die Drehrichtung im gleichen NC-Satz programmiert werden. Die Funktionen M40 bis M45 werden zur Anwahl der Getriebestufe und zum Anstoß des mechanischen Getriebeschaltens verwendet.

Syntax:

M40 | M41 | M42 | M43 | M44 | M45 [<*Spindelname*>..] **[M03 | M04]** modal

M40 bis M45	Getriebestufen 1 bis 6
<Spindelname>=<*expr*>	Spindeldrehzahl bestehend aus Spindelbezeichnung gemäß P-CHAN-00053 und Drehzahlwert in [U/min].

Programmierbeispiel

Getriebeschalten von Spindeln (M40 - M45)

```
S800 M41 M03 ;Spindeldrehzahl 800, Getriebestufe 2, Drehrichtung cw
```

- Die dekodierten Funktionen M40 bis M45 sind modal und werden am Satzanfang aktiviert. M40 bis M45 heben sich gegenseitig auf.
- Das System ermöglicht die Definition von maximal 6 Spindelgetriebestufen (M40... M45). Die minimalen und maximalen Drehzahlen werden für jede Getriebestufe in einer "Tabelle der Drehzahlbereiche" in [1] [▶ 894]-4 parametriert (Einheit = U/min).
- Die Festlegung der maximalen Geschwindigkeit für die 10-Volt Ausgabe bei lagegeregelten Spindeln erfolgt bei analogen Antrieben über den Multi-Gain-Faktor P-AXIS-00128 und P-CHAN-00129.
- Bei einem System mit automatischer Wahl der Getriebestufe wird diese allein durch die Programmierung der Drehzahl S bestimmt. M40 bis M45 müssen dann nicht mit programmiert werden.
- M40 bis M45 sind nur bei lagegeregelten Spindeln programmierbar.
- Der NC-Kern versucht immer, Getriebeschaltvorgänge zu minimieren (Wenn z.B. eine neue Drehzahl mit der aktuellen Getriebestufe gefahren werden kann, wird ein Getriebeschalten unterdrückt, auch wenn dieses mit M40 bis M45 explizit programmiert wurde).

Kanalparametersatz [1] [▶ 894]:

- Definition der M-Funktionen M40 - M45 und Festlegung der Synchronisationsarten.

```
:
m_synch[1]          0x00000001          MOS
m_synch[2]          0x00000002          MVS_SVS
:
m_synch[40]         0x00000002          MVS_SVS
m_synch[41]         0x00000002          MVS_SVS
m_synch[42]         0x00000002          MVS_SVS
m_synch[43]         0x00000002          MVS_SVS
m_synch[44]         0x00000002          MVS_SVS
m_synch[45]         0x00000002          MVS_SVS
m_synch[48]         0x00000008          MNS_SNS
m_synch[49]         0x00000002          MVS_SVS
```

- Freischalten des Getriebeschaltens:

```
main_spindle_gear_change    1     0:OFF        1:ON
```

- Parametrierung des Spindelgetriebes (Suchrichtung, Drehzahlbereiche):

```
:
spindel[0].range_way        0     0:bottom up    1: top down

#
spindel[0].range_table[0].min_speed      50      (M40)
spindel[0].range_table[0].max_speed      560     (M40)
spindel[0].range_table[1].min_speed      400                (M41)
spindel[0].range_table[1].max_speed      800                (M41)
spindel[0].range_table[2].min_speed      700     (M42)
spindel[0].range_table[2].max_speed      3500    (M42)
spindel[0].range_table[3].min_speed      3501               (M43)
spindel[0].range_table[3].max_speed      4000               (M43)
spindel[0].range_table[4].min_speed      3800    (M44)
spindel[0].range_table[4].max_speed      5500    (M44)
spindel[0].range_table[5].min_speed      5400               (M45)
spindel[0].range_table[5].max_speed      7000               (M45)
#
:
```

Programmierbeispiel

Getriebeschalten von Spindeln (M40 - M45)

Automatische Getriebestufenbestimmung: EIN

```
:
spindel[0].autom_range 1
:

NC-Programm:
S650 M03                    OK, M41 => SPS
S750                        OK, kein Schalten, M41 bereits angewählt
S950                        OK, automatisches Schalten, M42 => SPS
S1050                       OK, kein Schalten, M42 bereits angewählt
S750                        OK, automatisches Schalten, M41 => SPS
S500                        OK, kein Schalten, M41 bereits angewählt

S8000                       Fehler, zu große Drehzahl

Eine programmierte Getriebestufe wird immer geprüft:
M41 S750                    OK, 'automatisches' Schalten, M41 => SPS
...aber
M40 S750                    Fehler, falsche Getriebestufe
```

Programmierbeispiel

Getriebeschalten von Spindeln (M40 - M45)

Automatische Getriebestufenbestimmung: AUS

```
:
spindel[0].autom_range 0
:

NC-Programm:
M41 S650 M03                OK, M41 => SPS
M41 S750                    OK, kein Schalten, M41 bereits angewählt
M42 S950                    OK, Schalten, M42 => SPS
M42 S1050                   OK, kein Schalten, M42 bereits angewählt
M41 S750                    OK, Schalten, M41 => SPS
M41 S500                    OK, kein Schalten, M41 bereits angewählt

M41 S200                    Fehler, andere Getriebestufe (M40) program-
mieren
S950                        Fehler, keine Getriebestufe (M42) program-
miert
```

15.2.4 Drehfunktionen

15.2.4.1 Durchmesserprogrammierung (G51/G52)

Syntax:

G51	Anwahl der Durchmesserangabe	modal
G52	Abwahl der Durchmesserangabe	modal, Grundzustand

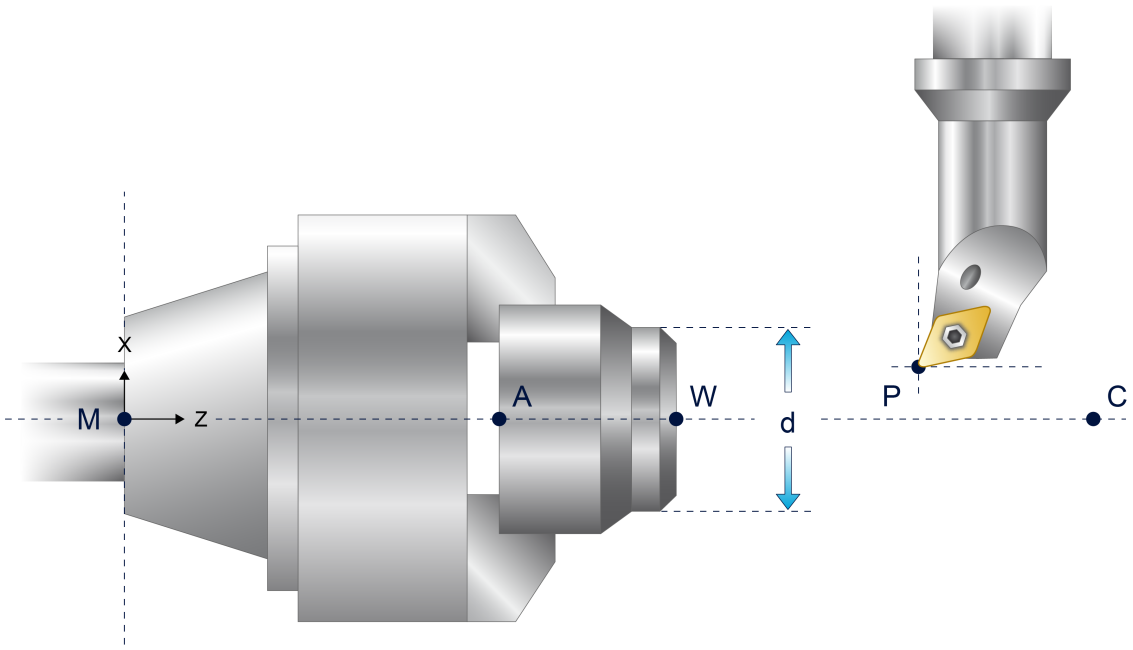


Abb. 159: Bezugspunkte und Durchmesserprogrammierung beim Drehen

W Werkstücknullpunkt
X Plandrehachse
M Maschinennullpunkt
C Steuerungsnullpunkt
P Schneidenpunkt
Z Längsdrehachse
A Anschlagpunkt
d Programmiertes Maß bei Durchmesserprogrammierung

Bei angewählter Durchmesserprogrammierung werden die Positionsangaben für die Plandrehachse in einem Verfahrsatz als Durchmesserangaben bezüglich der Drehmitte interpretiert.

Es ist zu beachten, dass die programmierten Koordinaten der Plandrehachse nur dann tatsächlich dem Werkstückdurchmesser entsprechen, wenn sich der Nullpunkt der Plandrehachse in der Drehmitte befindet (unabhängig davon, ob Verschiebungen im Durchmesser wirken; siehe Programmierbeispiel unten).

Über die Achsparameter können für Achsen im Modus "Plandrehen" parametriert werden:

- G51 bei absoluter Programmierung (G90) (P-AXIS-00058)
- … und/oder G51 bei relativer Programmierung (G91) (P-AXIS-00059)
- Bezugspunktprogrammierung (G92) und Nullpunktprogrammierung (G53 – G59) im Durchmesser (P-CHAN-00091)

G51 wirkt auf die Achse, die im Modus "Plandrehen" betrieben wird. Diese Plandrehachse muss bei Anwahl in der Bearbeitungsebene (G17, G18, G19) vorhanden sein.

Die Koordinaten des Kreismittelpunktes (I, J, K) bzw. die Kreisradiusprogrammierung (R) werden nicht im Durchmesser programmiert.

Die Abwahl der Durchmesserprogrammierung erfolgt durch G52.

Programmierbeispiel

Durchmesserprogrammierung (G51/G52) in G18

```
;Allgemeine Einstellungen (optional):
;Anzeige Positionswerte im Durchmesser: P-CHAN-00256 (TRUE, 1)

;Einstellungen X-Achse:
;Plandrehachse, translatorisch: P-AXIS-00015 (0x41)
;G51 bei G90: P-AXIS-00058 (TRUE, 1)
;G51 bei G91: P-AXIS-00059 (FALSE, 0)(optional)
;G92, G53-G59 im Durchmesser: P-CHAN-00091 (TRUE, 1) (optional)

;Einstellungen Z-Achse:
;Längsdrehachse, translatorisch: P-AXIS-00015 (0x81)

:
N05 G18
N10 G90 G01 F1000
N20 G51 X80         ;Durchmesser 80mm
N30 G92 X10         ;G92 um 10mm im Durchmesser
N40 X0              ;Position 0 + G92 => Durchmesser 10mm
N50 G91 X50         ;X relativ +50mm, nicht im Durchmesser
N80 G52             ;Abwahl Durchmesserprogrammierung
:
```

15.2.4.2 Schneidenradiuskorrektur (G40/G41/G42)

Syntax:

G40	SRK-Abwahl	modal, Grundzustand
G41	SRK links der Kontur	modal
G42	SRK rechts der Kontur	modal

Die Schneidenradiuskorrektur (SRK) wirkt bei der Drehbearbeitung in der mit G17, G18, G19 angewählten Bearbeitungsebene. In dieser Ebene muss eine der Achsen im Modus "Plandrehen", die andere im Modus "Längsdrehen" betrieben werden. (Achsmode: P-AXIS-00015)

Als Werkzeugkorrekturwerte werden die unter den D-Worten abgelegten Datensätze verwendet. Bei Drehwerkzeugen ist die Lage der Werkzeugschneide im Parameter P-TOOL-00002 bezüglich der Bearbeitungsebene (Plan-, Längsdrehachse) über eine zusätzliche Kennung 1...9 anzugeben (siehe Abbildung).

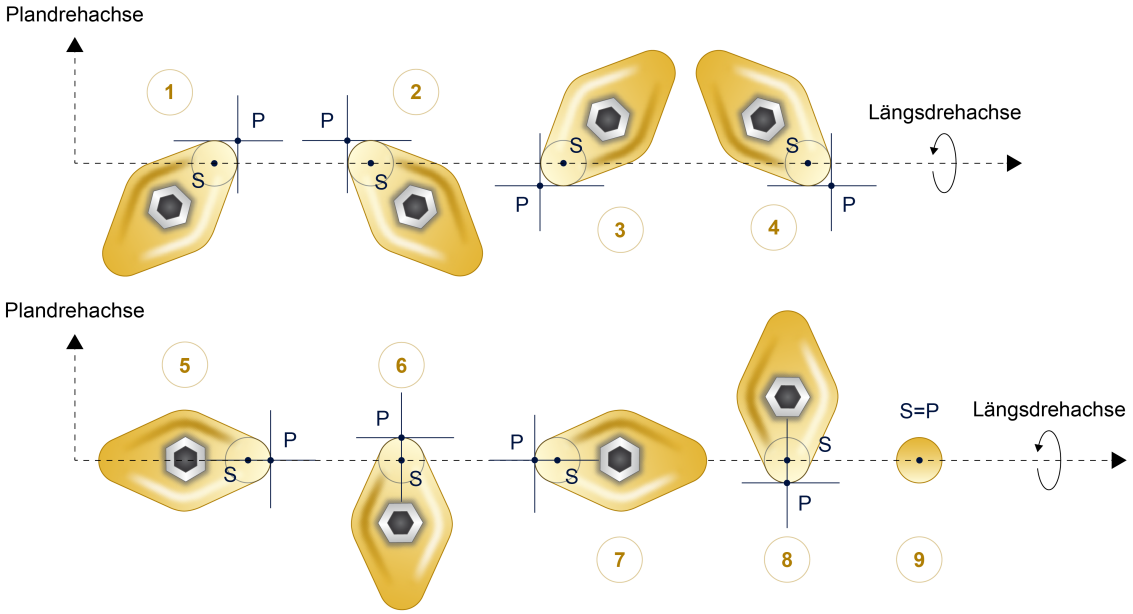


Abb. 160: Lage der Drehwerkzeugschneide in der Bearbeitungsebene.

Ein typisches Drehwerkzeug wird durch folgende Werte / Parameter charakterisiert:

• Werkzeugtyp	1 (Drehwerkzeug)
• SRK-Lage	1...9
• Werkzeugradius	Schneidenradius
• Werkzeuglänge	--
• Werkzeugversatzmaße (siehe nochfolgende Abbildung)	

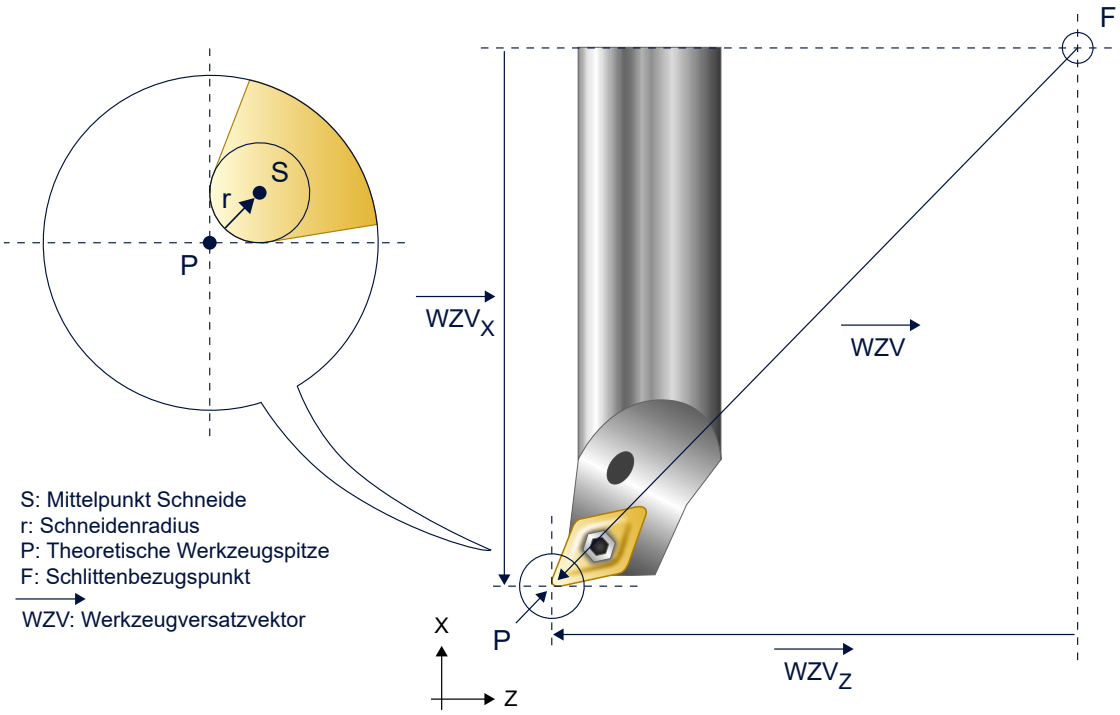


Abb. 161: Werkzeugvermessung für Werkzeugversatzkorrektur.

Bei der Angabe von Werkzeugachsversätzen ist deren Vorzeichen zu beachten, da es sich um Komponenten des Werkzeugversatzvektors in der Bearbeitungsebene handelt. Für das in der Abbildung oben dargestellte Beispiel eines Drehwerkzeugs haben die Versätze in Richtung der X- und Z-Achse jeweils negatives Vorzeichen.

Die Angabe der Werkzeugversätze müssen bis zur theoretischen Werkzeugspitze (Punkt P) erfolgen.

Ein Wechsel zwischen Drehwerkzeug und Fräswerkzeug ist bei angewähltem G41, G42 erlaubt. Bei absoluter Programmierung (G90) werden die aktuellen Achsversatzmaße des neuen Werkzeugs entsprechend dem Werkzeugtyp im nächsten Bewegungssatz verrechnet.

15.2.4.3 Umdrehungsvorschub (G95)

Syntax:

G95	Vorschub in mm/Umdrehung	modal

Beim Drehen mit aktivem G95 lässt sich über das F-Wort in [mm/U, inch/U] unabhängig von der Spindeldrehzahl eine konstante Spandicke festlegen.

Der Vorschub der Bahnachsen wird dabei an die Drehzahl der lagegeregelten Spindel gekoppelt. Er ist jeweils nur in Verbindung mit der Vorschubvereinbarung gültig, bei der er programmiert wurde. Bei einem Wechsel von G95 zu G94 oder G93 (Unterkapitel "Bearbeitungszeit/Vorschubgeschwindigkeit" [▶ 155]) wird also das bei G95 gültige F-Wort nicht übernommen.

G95 kann auch in Kombination mit einer gesteuerten (PLC-)Hauptspindel (P-CHAN-00069) programmiert werden. Hierbei ist zu beachten, dass der resultierende Umdrehungsvorschub nur von den programmierten Werten (F-Wort, S-Wort) abhängt. Echtzeiteinflüsse auf die Spindeldrehzahl, wie z.B. Overrideänderungen, werden nicht berücksichtigt. Diese Art der Programmierung ist nur in einem Kanal in Verbindung mit der zugeordneten Hauptspindel ab der Version V3.1.3066.02 zulässig.

Programmierbeispiel

Umdrehungsvorschub (G95)

```
N10 F1000 G01 X0 Z100 M3 S1200  ;Vorschub 1000 mm/min (G94)
N20 G95 F1.5          ;Umdrehungsvorschub 1.5 mm/U,
                      ;Drehzahl 1200 U/min
N30 Z50               ;Vorschub 1800 mm/min
N40 G94 X50           ;Vorschub 1000 mm/min aus N10 gültig
N50 G93 F20 X20       ;Bearbeitungszeit 20 s
N60 G95 Y200 S2000    ;Umdrehungsvorschub 1.5 mm/U aus N20 gültig,
                      ;Drehzahl 2000 U/min
N70 M30
```

Umdrehungsgeschwindigkeit (G95) und Achskopplung mit variabler Getriebeübersetzung (Getriebekopplung)

Ab Version 3.1.3079.03 kann mit #TURN [ROT_FEED_CPL=1] der Vorschub der Bahnachsen an die Geschwindigkeit der Hauptspindel gekoppelt werden. Die Vorschubanpassung erfolgt entsprechend der eingestellten Getriebeübersetzungen der Achskopplungen. Dies ermöglicht das Aktivieren des Umdrehungsvorschubs (G95) bei bereits drehender Spindel.

Achskopplungen mit variabler Getriebeübersetzung können über

- den NC-Befehl #GEAR LINK [▶ 491] oder
- über das HLI [FCT-A9// Achskopplung über HLI]

erfolgen.

Im Standardfall #TURN [ROT_FEED_CPL=0] wird die Geschwindigkeit der Hauptspindel ohne Berücksichtigung von Achskopplungen verwendet.

Bei einem Wechsel der Hauptspindel wird die Einstellung von ROT_FEED_CPL automatisch für die neue Hauptspindel übernommen und für die bisherige Hauptspindel deaktiviert.

Programmierbeispiel

G95 - Spindelgeschwindigkeit mit Getriebekopplung

```
;Achse 5 ist als Hauptspindel, Achse 6 (S2) als Nebenspindel konfigu-
riert.
; -----------------------------------------------------
; Achse 5 wird an Achse 6 und an sich selbst gekoppelt.
N10 #GEAR LINK ON [TARGETNR=5 AXNR1=6 AXNR2=5 NUM1=1 DENOM1=1 NUM2=1 DE-
NOM2=1]
N20 M3 S500          ; Solldrehzahl Achse 5 = 500 U/min.
N30 S2[M3 REV1500]   ; Solldrehzahl Achse 6 = 1500 U/min.
; Drehzahl Achse 5 = 500 U/min + 1500 U/min = 2000 U/min
; Drehzahl Achse 6 = 1500 U/min
; Vorschub = 1000 mm/min
N40 F1000 G01 X100
; Spindeldrehzahl entsprechend Achskopplung wird für G95 verwendet
N50 #TURN [ROT_FEED_CPL=1]
N60 G95 F1.5         ; Umdrehungsvorschub = 1.5 mm/U
; Vorschub = 2000 U/min * 1.5mm/U = 3000 mm/min
N70 X200
; Spindeldrehzahl ohne Achskopplung wird für G95 verwendet
N80 #TURN [ROT_FEED_CPL=0]
; Vorschub = 500 U/min * 1.5mm/U = 750 mm/min
N90 X400
N100 M05
N110 SC[M05]
N120 #GEAR LINK OFF [TARGETNR=5]
M30
```

15.2.4.4 Konstante Schnittgeschwindigkeit (G96/G97/G196)

Syntax:

G96	Anwahl konstante Schnittgeschwindigkeit	modal
G97	Abwahl konstante Schnittgeschwindigkeit, Anwahl Spindeldrehzahl	modal, Grundzustand
G196	Maximale Spindeldrehzahl bei G96	G196 nicht modal max. Drehzahl modal

Über die G-Funktionen G96, G97 und G196 kann die Interpretation des S-Wortes wahlweise umgeschaltet werden:

G96 S in [m/min bzw. ft/min *] (Schnittgeschwindigkeit)
G97 S in [U/min] (Spindeldrehzahl)
G196 S in [U/min] (max. Spindeldrehzahl während G96)

* [ab V2.11.2032.08 bei G70 und P-CHAN-00360 = 1]

Bei Anwahl mit G96 berechnet sich die Startdrehzahl der Spindel aus der programmierten Schnittgeschwindigkeit und dem Abstand der Werkzeugspitze zur Drehmitte. Dieser Abstand ergibt sich aus der zuletzt (nicht im aktuellen NC-Satz) programmierten Position und den Bezugspunktverschiebungen der Plandrehachse. In der aktuellen Bearbeitungsebene (G17, G18, G19) muss genau eine Plandrehachse vorhanden sein.

Eine bei G96 über das S-Wort programmierte Schnittgeschwindigkeit ist nur bis zur Abwahl mit G97 gültig. Die Aktivierung der konstanten Schnittgeschwindigkeit erfolgt bei G96 also erst wenn auch das S-Wort programmiert wurde.

Die Angabe einer maximalen Spindeldrehzahl mit G196 in Kombination mit dem S-Wort ist optional und nur während G96 wirksam. Die Programmierung der Drehzahlbegrenzung muss vor Anwahl von G96 erfolgen!

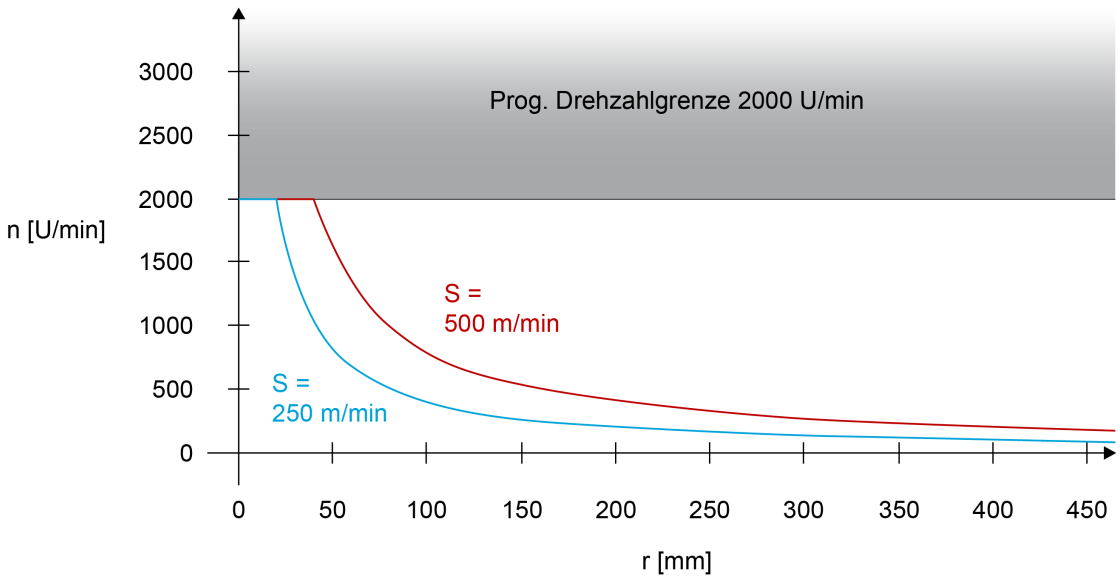


Abb. 162: Spindeldrehzahl bei aktivem G96

Versionshinweis

Erweiterte G-Funktion G196

Ab V3.1.3057.04

Die maximale Spindeldrehzahl in [U/min] kann alternativ auch als Zusatzwert in Kombination mit G196 programmiert werden. Sie ist haltend wirksam.

Mit dieser Syntax darf G196 dann im gleichen Satz wie G96 programmiert werden, ein eigener NC-Satz ist nicht erforderlich.

Syntax:

G196 = <*Max_Spindeldrehzahl*> G196 nicht modal, max. Drehzahl modal

In der Nähe der Drehmitte legt die programmierte Maximaldrehzahl (G196) bzw. die in den zugehörigen Achsparametern festgelegte maximale Spindeldrehzahl P-AXIS-00212 die Grenze der konstanten Schnittgeschwindigkeit fest.

Bei Abwahl mit G97 wird die zuletzt eingestellte Spindeldrehzahl beibehalten.

Verfahrbewegungen der Plandrehachse im Eilgang (G00) führen zu einer Unterbrechung von G96, so dass unerwünschte Drehzahländerungen beim Positionieren des Werkzeugs verhindert werden können. Der nächste Verfahrsatz mit G01, G02 oder G03 hebt die Unterdrückung von G96 wieder auf.

Programmierbeispiel

Konstante Schnittgeschwindigkeit (G96/G97/G196)

```
; X ist Plandrehachse

N10 M03 S1000 G01 F1500 X100
N20 G196 S6000    ;max. Drehzahl 6000 U/min
N30 G96  S63  ;Anwahl konst. Schnittgeschw. 63 m/min,
              ;Werkstückradius 100mm entsprechend X-Koordinate
N40 X80
N50 S4 X50    ;neue Schnittgeschw. 4m/min; Werkstückradius 80mm,
              ;am Satzende 50mm
N60 G97       ;max. Drehzahl 6000 U/min hier nicht wirksam!
N80 G92 X-10  ;Bezugspunktverschiebung in X um -10mm
N90 G96 X60   ;Schnittgeschw. aus N50 nicht gültig: konst.
              ;Schnittgeschw. nicht aktiv, Drehzahl 8000 U/min
N100 S25 X70  ;Schnittgeschw. 25m/min, Werkstückradius 50mm,
              ;(=60mm+BPV), konst. Schnittgeschw. aktiv
N110 G00 X450 ;Eilgang: Drehzahl bleibt konstant
N115 X70
N120 G01 X40  ;Unterdrückung von G96 aufgehoben
N110 M30
```

15.2.4.5 Gewindeschneiden mit endlos drehender Spindel (G33)

Ein-/mehrgängige Gewinde

Beim Gewindeschneiden mit endlos drehender Spindel (G33) wird die Bahnbewegung auf den Nulldurchgang der Spindeldrehung synchronisiert. Der Gewindeschnitt kann deshalb auch in mehreren, aufeinanderfolgenden Durchgängen erfolgen. Durch optionale Angabe eines Versatzwinkels können auch mehrgängige Gewinde gefertigt werden.

Für das Erzielen eines guten Bearbeitungsergebnisses und zur Minimierung von Konturfehlern kann für die Spindel sowie für die Bahnachsen eine Vorsteuerung angewählt werden.

Programmierung

```
Syntaxbeispiel für ZX-Ebene (Z Längsachse, X Zustellachse):
```

G33 Z.. K.. [<*Spindelname*>**.OFFSET=..**] modal

G33	Gewindeschneiden mit endlos drehender Spindel. Die G33-Funktion ist haltend wirksam. Der nächste Bewegungssatz mit einer haltenden Satzart (G00, G01, G02, G03, Spline, Polynom) wählt das Gewindeschneiden ab.
Z..	Zielposition ("Gewindelänge") in [mm, inch]
K..	Die Gewindesteigung wird bei aktivem Gewindeschneiden in der Einheit [mm/U, inch/U] ohne Vorzeichen über die Adressbuchstaben I, J und K programmiert. Diese sind gemäß DIN 66025 der X-, Y-, und Z-Achse zugeordnet. Die Gewindesteigung ist bis zum Programmende haltend wirksam und darf bei Anwahl von G33 nicht Null sein. Der Vorschub wird nicht über das F-Wort programmiert, sondern ergibt sich aus der Spindeldrehzahl und der Gewindesteigung. Die Steigung von Längsgewinden bzw. Kegelgewinden mit einem Neigungswinkel kleiner als 45° wird über den Adressbuchstaben K angegeben, wenn die Z-Achse Längsdrehachse ist. Bei Plangewinden bzw. Kegelgewinden mit einer Steigung größer oder gleich 45° erfolgt die Angabe der Steigung über I, wenn als Plandrehachse die X-Achse verwendet wird und über J, wenn die Y-Achse verwendet wird. In der nachfolgenden Abbildung sind Beispiele für die Angabe der Gewindesteigung über die Adressbuchstaben in der Z-X-Ebene dargestellt.
<*Spindelname*>.OFFSET=..	Gewindeversatzwinkel in [°] im Modulobereich der Spindel. Optional, ist nur bei mehrgängigen Gewinden erforderlich. Der Versatzwinkel ist bis Programmende haltend wirksam. Spindelbezeichnung gemäß P-CHAN-00053. Das "="-Zeichen ist optional.

Steigungsangaben I, K bei Längsgewinde

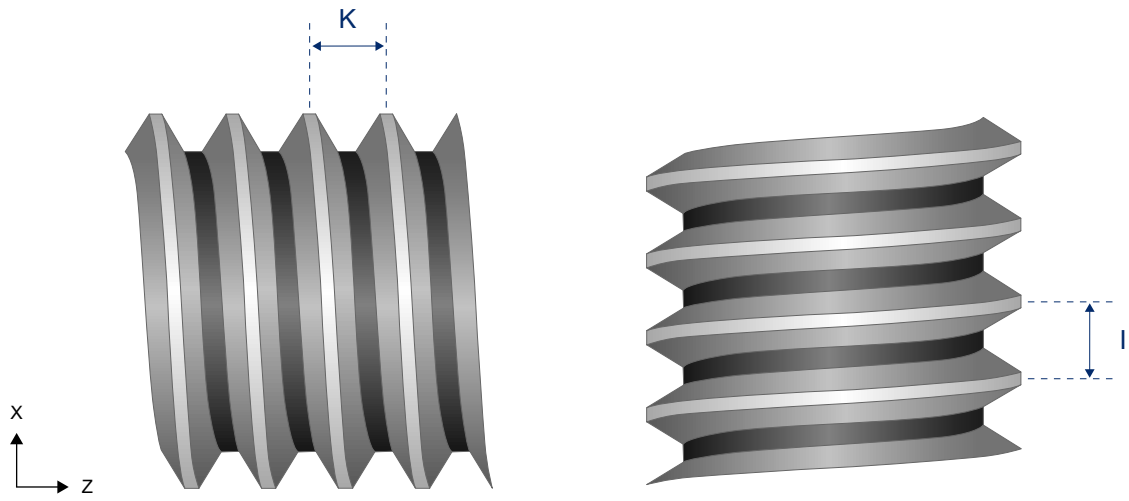


Abb. 163: Angabe der Gewindesteigung bei Längsgewinde

Steigungsangaben I, K bei Kegelgewinde

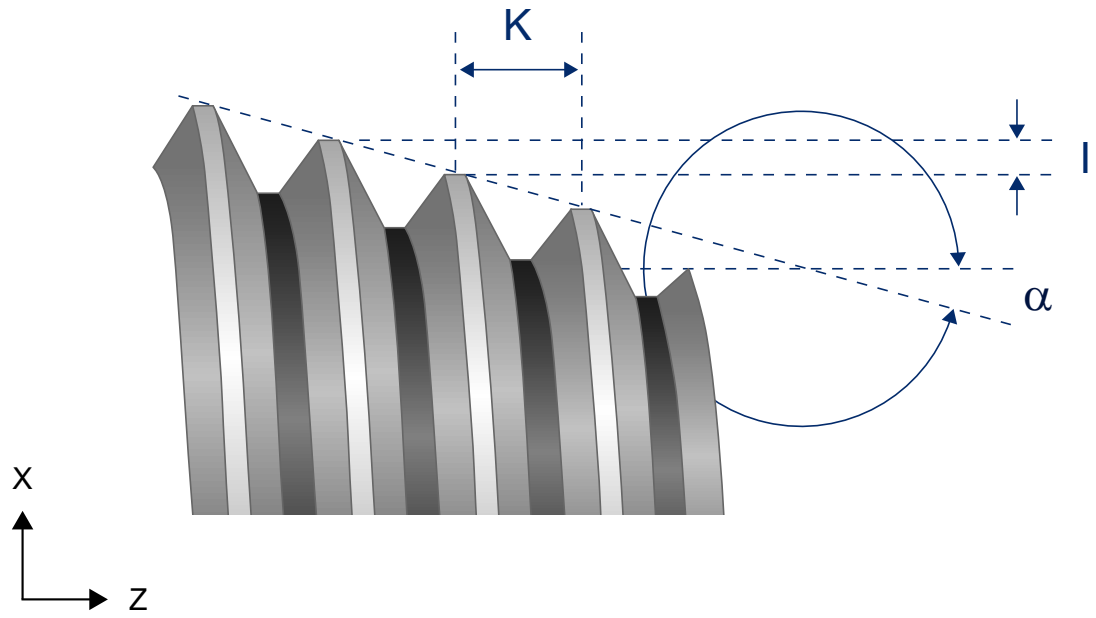


Abb. 164: Angabe der Gewindesteigung bei Kegelgewinde

Programmierbeispiel

Gewindeschneiden mit endlos drehender Spindel (G33)

G33 Z.. K.. [S.OFFSET=..]

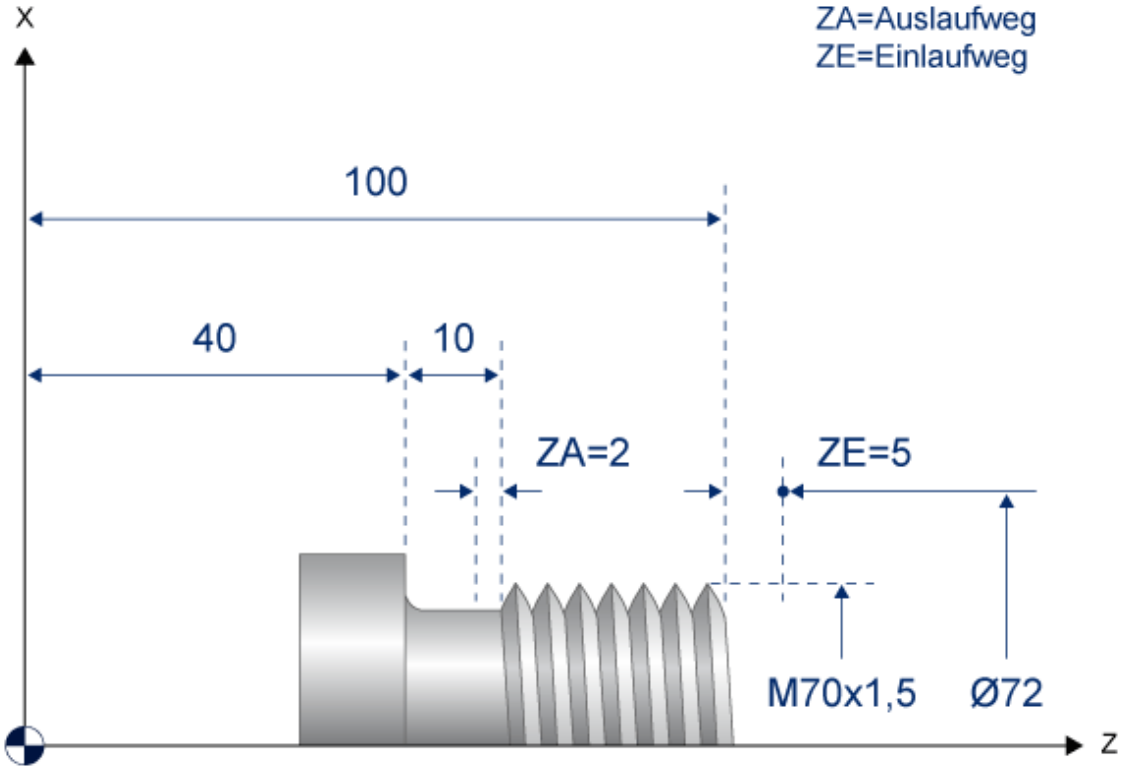


Abb. 165: Darstellung der Beispielgeometrie

Schneiden eines Längsgewindes (M70x1.5) in mehreren Schnitten:

```
%L Laengsgewinde

N100 G33 Z48 K1.5                ;Gewindegang schneiden
N110 G00 X72                     ;Rückzug und Fahren
N120 Z105                        ;auf Startposition
N130 M29                         ;Unterprogrammende

%G33 (Gewindetiefe 0.92 mm)
N10 G51                          ;Anwahl Durchmesserprogrammierung
N15 T1 D1 M03 S400               ;Werkzeuganwahl, Spindel starten
N20 G00 X72 Z105                 ;Anfahren

N25 G01 X69.54 F1000             ;Auf 1. Schnitttiefe positionieren
N30 LL Längsgewinde              ;1. Gewindeschnitt

N35 G01 X69.08                   ;Auf 2. Schnitttiefe positionieren
N30 LL Längsgewinde              ;2. Gewindeschnitt

N35 G01 X68.62                   ;Auf 3. Schnitttiefe positionieren
N30 LL Längsgewinde              ;3. Gewindeschnitt

N35 G01 X68.16                   ;Auf Endtiefe positionieren
N30 LL Längsgewinde              ;4. Gewindeschnitt

N35 G01 X68.16                   ;Erneut auf Endtiefe positionieren
N30 LL Längsgewinde              ;Leerschnitt

N60 M05 X150 Z200                ;Fahren auf Endposition
N65 M30                          ;Programmende
```

Schneiden eines 2-gängigen Längsgewindes (M70x1.5)

```
%G33_2 (2 Gewindegänge, Gewindetiefe 0.92 mm)
N10 G51                          ;Anwahl Durchmesserprogrammierung
N15 T1 D1 M03 S40                ;Werkzeuganwahl, Spindel starten
N20 G00 X72 Z105                 ;Anfahren
N25 G01 X68.16 F1000             ;Auf Gewindetiefe positionieren
N30 G33 Z48 K1.5                 ;1. Gewindegang schneiden
N35 G00 X72                      ;Rückzug und Fahren
N40 Z105                         ;auf nächste
N45 G01 X68.16                   ;Startposition
N50 G33 Z48 K1.5 S.OFFSET=180    ;2. Gewindegang schneiden bei 180°
N55 G00 X72                      ;Rückzug und Fahren
N60 M05 X150 Z200                ;auf Endposition
N65 M30                          ;Programmende
```

Schneiden eines Kegelgewindes

```
%L Kegelgewinde
N010 G33 Z90 X1 I5.0              ;Gewindegang schneiden (Bezug I)
; N010 G33 Z90 X1 K5.0            ;Gewindegang schneiden (Bezug K)
N020 G00 X72                      ;Rückzug und Fahren
N030 Z105                         ;auf Startposition
N040 M29                          ;Unterprogrammende

%G33
N050 G00 X0 Y0 Z0
N060 G18
N070 G51                          ;Anwahl Durchmesserprogrammierung
N080 D1 M03 S1                    ;Werkzeuganwahl, Spindel starten
N090 G00 X105 Z105                ;Anfahren
N100 G01 X100 F1000               ;Auf 1. Schnittiefe positionieren
N110 LL Kegelgewinde              ;1. Gewindeschnitt
N120 M05 X150 Z200                ;Fahren auf Endposition
N130 M30                          ;Programmende
```

15.2.4.6 Gewindeschneiden mit Istdrehzahl der Spindel

Beim Gewindeschneiden kann die Spindel unter Last zu Drehzahlabweichungen neigen. Um auch in diesem Fall Gewindeschneiden zu können, ist es möglich, die Bewegung der Linearachsen direkt an die Istdrehzahl der Spindel zu koppeln. Das heißt, dass auch die Bahnbewegung auf den Nulldurchgang der Istpositionen synchronisiert wird.

Um Sollwertsprünge der Bahnachsen zu vermeiden, kann eine stark verrauschte Spindel-Istdrehzahl über einen Mittelwertfilter geglättet werden.

Versionshinweis

Funktionalität verfügbar ab V3.01.3080.16

Die Aktivierung der Funktionalität erfolgt über P-CHAN-00834 oder über den NC-Befehl #TURN [▷ 495] [THREAD_CUT_ACT_SPEED=1 …].

Beispiel

Gewindeschneiden mit Spindel Istdrehzahl]

Festlegen über Kanalparameter:

```
thread_cutting.use_actual_speed 1   ( P-CHAN-00834)
thread_cutting.n_cycles         5   ( P-CHAN-00835)
```

Alternative Festlegung über NC-Programm:

```
#TURN[THREAD_CUT_ACT_SPEED=1 THREAD_CUT_N_CYCLES=5]
```

Programmierbeispiel

Gewindeschneiden mit Spindel Istdrehzahl

```
%spindle_test.nc
(Gewindeschneiden mit Istdrehzahl)
N15 M03 S40           ; Spindel starten
N20 G00 X72 Z105      ; Anfahren
N25 G01 X68.16 F1000  ; Auf Gewindetiefe positionieren
N30 G33 Z48 K1.5      ; 1. Gewindegang schneiden mit Solldrehzahl
N35 G00 X72           ; Rückzug
N40 Z105              ; und Fahren
N45 G01 X68.16        ; nächste Startposition

; Gewindeschneiden mit Istdrehzahl und gefilterten Werten
N50 #TURN[THREAD_CUT_ACT_SPEED=1 THREAD_CUT_N_CYCLES=20]
; 2. Gewindegang schneiden bei 180° mit Istdrehzahl
N55 G33 Z10 K1.5 S.OFFSET=180

N60 G00 X72           ; Rückzug und Fahren
N65 M05 X150 Z200     ; auf Endposition
N70 M30
```

15.2.5 Gewindebohren (G63)

```
Syntaxbeispiel Gewindebohrung in Z-Richtung:
```

G63 Z.. **F**.. <*Spindelname*>.. modal

G63	Gewindebohren
Z..	Gewindetiefe (Zielposition) in der Bohrachse in [mm, inch]
F..	Vorschub in [mm/min, m/min, inch/min]
<*Spindelname*>..	Spindeldrehzahl bestehend aus Spindelbezeichnung gemäß P-CHAN-00053 und Drehzahlwert in [U/min]

Diese Art des Gewindebohrens (G63) erfordert eine lagegeregelte Spindel, die durch die CNC synchron zur Bahnbewegung mitgeführt wird. Hierbei erfolgt eine genaue dynamische Abstimmung von Spindel und den an der Zustellbewegung beteiligten Achsen. Ein Ausgleichsfutter ist nicht erforderlich. Der programmierte Vorschub muss zur gewählten Spindeldrehzahl und zur Gewindesteigung des Bohrers passen und wird wie folgt berechnet:

Vorschub F [mm/min] = Drehzahl S [U/min] * Gewindesteigung [mm/U]

G63 wird durch die Anwahl einer anderen modalen Satzart (z.B. Linearbewegung G01) abgewählt. Eine nicht-haltende Satzart (z.B. Verweilzeit mit G04) bewirkt keine Deaktivierung von G63.

Der Bahnvorschub (F-Wort) und die Spindeldrehzahl (S-Wort) müssen nicht zwingend im selben NC-Satz wie G63 angegeben werden. Die Berechnung des Vorschubes muss immer auf den zuletzt programmierten Werten basieren.

Eine Fehlermeldung wird ausgegeben, wenn Bahnvorschub oder Spindeldrehzahl bei der G63 - Anwahl gleich Null sind.

In Kombination mit G331/G332 können M03, M04, M05, M19 nicht programmiert werden.

Achtung

Die Spindel (bzw. der Gewindebohrer) muss beim Aufruf von G63 stehen. Dies kann durch die vorhergehende Programmierung von M05 (Spindel stoppen) oder M19 mit S.POS (Spindel positionieren) erreicht werden.

Das Schneiden eines Linksgewindes oder das Herausfahren aus der Gewindebohrung wird mit **negativem S-Wert** programmiert.

Bei C-Achsbetrieb kann über den Parameter P-AXIS-00052 die Getriebestufe festgelegt werden.

Programmierbeispiel

Gewindebohren (G63)

Bohren eines Rechtsgewindes mit Steigung 1.25 mm, Gewindetiefe 50 mm. Bei einer Spindeldrehzahl S von 200 U/min ergibt sich somit ein Vorschub von:

F = 200*1.25 = 250 mm/min

```
;…
G01 F2000 G90 X0 Y0 Z0 ; Achsen positionieren
M19 S.POS=0 M3 S100    ; Spindel anhalten und positionieren
;…
G63 Z-50 F250 S200     ; Gewindebohren
    Z0 S-200           ; Herausfahren aus Gewindebohrung
G01 F1000 X100         ; Neupositionieren, Abwahl Gewindebohren
:
```

Programmierbeispiel

Gewindebohren (G63)

```
%Gewindebohren_G63

N05 X0 Y0 Z0
N10 G91 Z100
N20 M19 S.POS180 M3 S100    ; Spindel positionieren

N30 G63 Z-50 F300 S200      ; Gewindebohren
N40 Z100 S-200              ; Herausfahren aus Gewindebohrung

N50 G01 X200 F3000     ; Neupositionieren, Abwahl Gewindebohren

N60 G63 Z-70 F300 S200      ; Gewindebohren
N70 Z100 S-200              ; Herausfahren aus Gewindebohrung

N80 M05 G01 X300 F1000
N90 M30
```

15.2.6 Gewindebohren (G331/G332)

Versionshinweis

Diese Funktionalität ist verfügbar ab CNC-Version V3.1.3067.01

Syntaxbeispiel für Gewindebohrung in Z-Richtung:

G331 Z.. **K**.. <*Spindelname*>..	Gewindebohren	modal
G332 Z.. [**K**..] [<*Spindelname*>..]	Gewindebohren Rückzug	modal

G331	Gewindebohren
Z..	Gewindetiefe (Zielposition) in der Bohrachse in [mm, inch]
K..	Gewindesteigung im zugeordneten Interpolationsparameter in [mm/U, inch/U]
<*Spindelname*>..	Spindeldrehzahl bestehend aus Spindelbezeichnung gemäß P-CHAN-00053 und Drehzahlwert in [U/min]
G332	Herausfahren aus Gewindebohrung (Rückzug). G332 bewirkt die automatische Richtungsumkehr der Spindel bei der Rückzugsbewegung.
Z..	Rückzugsposition der Bohrachse nach der Gewindebohrung in [mm, inch]
K..	Gewindesteigung im zugeordneten Interpolationsparameter in [mm/U, inch/U]. Die Gewindesteigung muss die gleiche Steigung sein wie die bei der zugehörigen Gewindebohrung mit G331. Die Angabe ist optional, wenn nicht programmiert, gilt die Steigung aus G331-Satz.
<*Spindelname*>..	Spindeldrehzahl bestehend aus Spindelbezeichnung gemäß P-CHAN-00053 und Drehzahlwert in [U/min]. Die Angabe ist optional, wenn nicht programmiert, gilt die Drehzahl aus G331-Satz.

Diese Art des Gewindebohrens (G331/G332) erfordert eine lagegeregelte Spindel, die durch die CNC synchron zur Bahnbewegung mitgeführt wird. Hierbei erfolgt eine genaue dynamische Abstimmung von Spindel und den an der Zustellbewegung beteiligten Achsen. Ein Ausgleichsfutter ist nicht erforderlich.

Durch die Angabe eines Vorzeichens bei der Gewindesteigung wird die Art des Gewindes festgelegt:

- Steigung ohne oder mit positivem Vorzeichen (+): Rechtsgewinde, z.B. K2 oder K+2
- Steigung mit negativem Vorzeichen (-): Linksgewinde, z.B. K-2

Der Vorschub der Gewindebohrachse ergibt sich aus der programmierten Steigung und der Spindeldrehzahl. Bei seiner internen Berechnung gelten die zulässigen Geschwindigkeitsgrenzwerte. Bei Verletzung dieser Grenzwerte erfolgt die Ausgabe einer Fehlermeldung.

Nach Beenden der Gewindebohrung ist der Vorschub weiterhin gültig. Bei folgenden G331/G332 wird der Vorschub aber wieder erneut aus den entsprechenden programmierten bzw. gespeicherten Werten von Steigung und Spindeldrehzahl berechnet.

G331/G332 wird durch die Anwahl einer anderen modalen Satzart (z.B. Linearbewegung G01) abgewählt und die Spindel aus dem Bahnverbund abgegeben. Eine nicht-haltende Satzart (z.B. Verweilzeit mit G04) bewirkt keine Deaktivierung von G331/G332.

Eine Fehlermeldung wird ausgegeben, wenn Steigung oder Spindeldrehzahl bei G331/G332 gleich Null sind oder Bohrachse und Steigungsparameter nicht zusammenpassen. Gültige Kombinationen sind X mit I, Y mit J und Z mit K.

In Kombination mit G331/G332 können M03, M04, M05, M19 nicht programmiert werden.

Achtung

Die Spindel (bzw. der Gewindebohrer) muss beim Aufruf von G331 stehen. Dies kann durch die vorhergehende Programmierung von M05 (Spindel stoppen) oder M19 mit S.POS (Spindel positionieren) erreicht werden.

Programmierbeispiel

Gewindebohren (G331/G332)

Bohren von Rechtsgewinden mit Steigung 2 mm, Gewindetiefe 50 mm, Spindeldrehzahl S 200 U/min, Z ist Bohrachse:

```
;…
G01 F2000 G90 X0 Y0 Z0 ; Achsen positionieren
M19 S.POS=0 M3 S100    ; Spindel anhalten und positionieren
;…
G331 Z-50 K2 S200      ; Gewindebohren in Z
G332 Z10  K2 S200      ; Rückzug
G01 F1000 X50          ; Neupositionieren, Abwahl Gewindebohren
G331 Z-50 K2 S200      ; Gewindebohren in Z
G332 Z10  K2 S400      ; Rückzug mit erhöhter Drehzahl
G01 F1000 X100         ; Neupositionieren, Abwahl Gewindebohren
G331 Z-50  K2 S200     ; Gewindebohren in Z
G332 Z10               ; Rückzug, K und S von G331
G01 F1000 X150         ; Neupositionieren, Abwahl Gewindebohren
;…
```

Bohren eines Rechtsgewindes mit Steigung 1.5 mm, Gewindetiefe 60 mm, Spindeldrehzahl S 150 U/min, X ist Bohrachse:

```
;…
G01 F2000 G90 X100 Y0 Z0 ; Achsen positionieren
M19 S.POS=0 M3 S100      ; Spindel anhalten und positionieren
;…
G331 X40  I1.5 S150      ; Gewindebohren in X
G332 X110 I1.5 S150      ; Rückzug
:
```

Programmierbeispiel

Gewindebohren mit Relativdrehzahl

```
%Gewindebohren mit Relativdrehzahl

N010 G91 G19 G0 X100 M03 S2000
N020 S2[MC_GearIn Master=S1 \ ; Koppeln der Werkzeug-
RN=1 RD=1 Mode=256 \          ; Spindel S2 an die
PhaseShift=1800000 WAIT_SYN]  ; Hauptspindel S1

N030 #MAIN SPINDLE[S2]        ; Hauptspindel Werkzeugspindel S2
N040 G331 Z-100 K1.5 S200     ; Gewindebohren Rechtsgewinde
N050 G332 Z100 K1.5 S200      ; Herausfahren aus Gewindebohrung
N060 G01 X300 F1000
N070 S2[MC_GearOut WAIT_SYN]  ; Kopplung zur Hauptspindel lösen
N080 #MAIN SPINDLE[S1]
N090 M30
```

15.2.7 C-Achsbearbeitung

Diese Funktionalität ergänzt die bereits vorhandenen Drehfunktionen und ermöglicht die Stirn- und Mantelflächenbearbeitung von zylindrischen Werkstücken an Drehmaschinen und Fräsmaschinen mit Drehteller. Dabei wird das Werkstück von der rotatorischen Achse oder Spindel (C-Achse) und das angetriebene Werkzeug (z.B. ein Fräser) von den beiden translatorischen Achsen X (oder Y) und Z bewegt. Für die C-Achsbearbeitung sind dazu Einstellungen in den Parametern P-CHAN-00008 , P-AXIS-00015 und P-AXIS-00018 notwendig.

Beispiel für Konfiguration einer C-Achse in den Achsparametern:

```
kenngr.achs_typ      0x0002  #Bahnachse
oder
kenngr.achs_typ      0x0004  #Spindelachse
kenngr.achs_mode     0x0204  #MODULO und CAX
```

Die Stirnflächenbearbeitung und die Mantelflächenbearbeitung werden in kartesischen Koordinaten beschrieben.

Sämtliche Interpolationsarten wie Linear, Zirkular und Spline werden sowohl auf der Stirn- als auch auf der Mantelfläche unterstützt. Die Funktionalität ermöglicht auch die Bearbeitung von durch die Drehmitte verlaufenden Bahnkonturen. Bei Drehmaschinen wird dabei die C-Achse automatisch gerichtet.

Die 2,5D-Werkzeugradiuskorrektur kann über die bekannten G-Befehle genutzt werden.

Durch Verwendung der erweiterten Dynamiküberwachung kann bei der C-Achs-Funktionalität die Überschreitung von dynamischen Achskenngrößen speziell auch bei Konturen, die nahe der Drehmitte verlaufen, verhindert werden.

Die Hauptachsen in allen Bearbeitungsmodis sind X, Y (abh. vom Maschinentyp), Z und C.

15.2.7.1 Eintauschen der Spindel in den Bahnverbund (#CAX, #CAX OFF)

Dieser Basismodus ist für die C-Achsbearbeitung insbesondere bei Drehmaschinen notwendig, da hier die Hauptspindel in eine rotatorische Bahnachse (z.B. "C") überführt werden muss.

Hinweis

Auf Fräsmaschinen oder Bearbeitungszentren, die konstruktiv eine rotatorische Werkstückaufnahme besitzen (z.B. Drehteller), ist ebenfalls eine C-Achsbearbeitung möglich. Die Anwahl von #CAX ist in diesem Fall nicht erforderlich.

Die drei physikalischen Achsen X, Y, Z und der in den Bahnverbund eingetauschten Achse C können direkt programmiert werden. Die Linearachsen werden in kartesischen Koordinaten und die C-Achse in Winkeleinheiten programmiert.

Die Radius- bzw. Durchmesserprogrammierung hängt dabei von G52/G51 ab.

Jeweils zwei Linearachsen definieren die Hauptebene: ZX (G18) oder YZ (G19).

Syntax:

#CAX [**[** [<*Hauptspindelname*>,] <*C-Achsname*> **]**]

<*Hauptspindelname*>	Es kann ausschließlich die Hauptspindelbezeichnung entsprechend P-CHAN-00053 programmiert werden. Soll eine andere Spindel als C-Achse genutzt werden, so muss diese zuerst zur Hauptspindel deklariert werden (siehe Programmierbeispiel im. Kapitel [PROG// Wechsel der Hauptspindel [▶ 706]]). Ansonsten erfolgt die Ausgabe einer Fehlermeldung.
<*C-Achsname*>	Frei definierbare Bezeichnung der C-Achse im NC-Programm .Wenn kein C-Achsname programmiert ist, wird der Standardname gemäß P-CHAN-00010 benutzt.

Die Hauptebene (Kreisinterpolation, Werkzeugradiuskorrektur) bleibt die gleiche wie vor der Aktivierung der C-Achse.

Wird für die Spindel ein Befehl programmiert (M3, M4, M5 [▶ 642], etc.) obwohl die Achse noch als C-Achse im Bahnverbund ist, wird ein Fehler erzeugt.

Die Abwahl der C-Achse, d.h. das Zurückgeben der Achse an den Spindelinterpolator erfolgt durch:

Syntax:

#CAX OFF

Programmierbeispiel

C-Achsbearbeitung

Eintauschen der Spindel in den Bahnverbund

```
;…
#CAX                          ; Annahme: Default-C-Achse sei "C"
G01 G90 X50 Z10 C90 F200
#CAX OFF                      ; Abwahl C-Achsbetrieb
;...
#CAX[S, C] oder #CAX[C]       ; Annahme: Hauptspindel sei "S"
G01 G90 X50 Z10 C90 F200
#CAX OFF
;...
#MAIN SPINDLE [S2]            ; "S2" wird neue Hauptspindel "S"
#CAX[S, C]                    ; Anwahl C-Achsbetrieb
G01 G90 X50 Z10 C90 F200
;...
#CAX OFF                      ; Abwahl C-Achsbetrieb
;...
#CAX[S3, C]                   ; Fehler, "S3" ist keine Hauptspindel
```

15.2.7.2 Stirnflächenbearbeitung (#FACE, #FACE OFF)

Dieser Modus wird bei Drehmaschinen und Bearbeitungszentren angewählt. Die gewünschte Kontur auf der Stirnfläche wird in einem virtuellen kartesischen Koordinatensystem in Millimeter oder Inch programmiert.

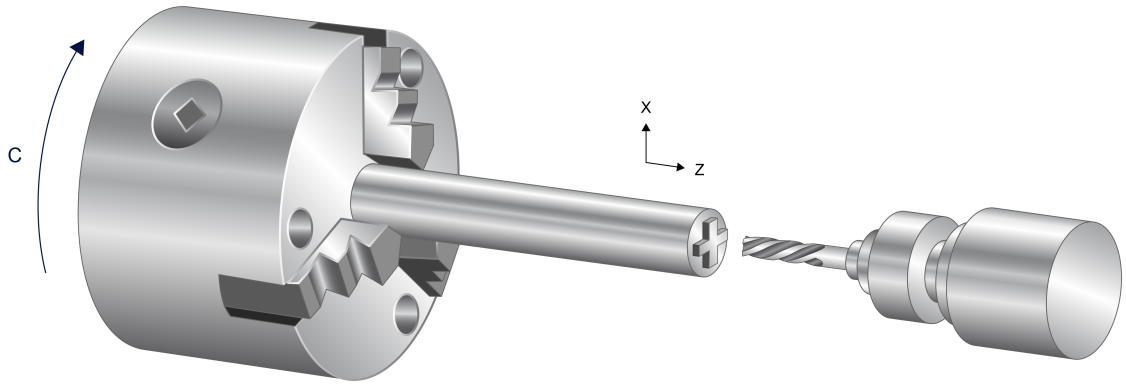


Abb. 166: Stirnflächenbearbeitung

Hinweis

Ab CNC-Version V3.00 muss für die Nutzung der Stirnflächenbearbeitung zwingend der Parameter P-CHAN-00262 mit der verwendeten Kinematik ID, abhängig von P-CHAN-00008, belegt werden.

a) Für Stirnflächentransformation 1 mit P-CHAN-00008=1 - **ID 13**

b) Für Stirnflächentransformation 2 mit P-CHAN-00008=2 - **ID 14**

Für die Programmierung der Kontur in kartesischen Koordinaten auf der Stirnfläche stehen die drei logischen Achsen X, Y (bzw. C) und Z zur Verfügung.

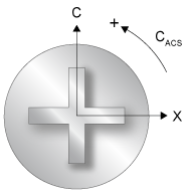


Abb. 167: Frontansicht Stirnflächenbearbeitung

Das folgende Bild zeigt die einzelnen Hauptebenen bei der Stirnflächenbearbeitung. Technische Bedeutung hat nur die dargestellte G17-Ebene.

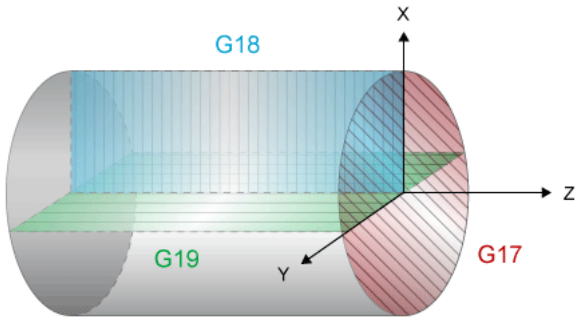


Abb. 168: Hauptebenen der Stirnflächenbearbeitung

Weitere Einstellmöglichkeiten für die Stirnflächenbearbeitung siehe [KITRA// KIN_TYP_13/14]

Syntax:

#FACE [*<Name 1. Hauptachse>*, *<Name 2. Hauptachse>* **]**

<Name 1. Hauptachse> Achsbezeichnung der ersten Hauptachse entsprechend der aktuellen Hauptebene.

<Name 2. Hauptachse> Achsbezeichnung der zweiten Hauptachse entsprechend der aktuellen Hauptebene (virtuelle kartesische Achse).

Bei Anwahl wird die Hauptebene (Kreisinterpolation, Werkzeugradiuskorrektur) immer durch die erste und zweite Hauptachse gebildet (G17). Ein Wechsel der Hauptebene während aktiver Stirnflächenbearbeitung mit G18, G19 ist nicht erlaubt.

Hinweis

Programmierte Mitschleppachsen sind von der Transformation nicht betroffen.

Der Modus wird abgewählt durch:

Syntax:

#FACE OFF

Durch #FACE OFF wird wieder in den zuvor aktiven Zustand zurückgekehrt. D.h., es erfolgt automatisch die Anwahl der zuletzt aktiven Hauptebene und die Wiederherstellung der zuletzt aktiven Achsversätze.

Programmierbeispiel

Programmierbeispiel für Drehmaschinen

Beispiel mit Achsbezeichner "C" für 2. Hauptachse

```
;…
#CAX[S, C]                  ;Annahme: Hauptspindel sei "S"
#FACE[X, C]                 ;Anwahl Stirnflächenbearbeitung
;…
G01 X40 C-30 Z50 F1000      ;Vorpositionieren
G01 Z30                     ;Zustellung
G01 X10 C40                 ;Kontur fahren
G01 Z50                     ;Rückzug
;…
#FACE OFF
#CAX OFF
;…
M30
```

Beispiel mit Achsbezeichner „Y“ für 2. Hauptachse.

Hinweis: Es darf im Kanal keine andere Achse mit dem gleichen Namen „Y“ geben.

```
;…
#CAX[S, Y]                  ;Annahme: Hauptspindel sei "S"
#FACE[X, Y]                 ;Anwahl Stirnflächenbearbeitung
;…
G01 X40 Y-30 Z50 F1000      ;Vorpositionieren
G01 Z30                     ;Zustellung
G01 X10 Y40                 ;Kontur fahren
G01 Z50                     ;Rückzug
;…
#FACE OFF
#CAX OFF
;…
M30
```

Programmierbeispiel

Programmierbeispiel für Bearbeitungszentren

Die rotatorische Achse (Werkstückachse) im Kanal sei "C2". Der Befehl #CAX ist nicht erforderlich.

```
;…
#FACE[X, C2]                ;Anwahl Stirnflächenbearbeitung
;…
G01 X40 C2=-30 Z50 F1000    ;Vorpositionieren
G01 Z30                     ;Zustellung
G01 X10 C2=40               ;Kontur fahren
G01 Z50                     ;Rückzug
;…
#FACE OFF
;…
M30
```

15.2.7.2.1 Stirnflächenbearbeitung mit 2 rotatorischen Achsen (#FACE 2ROT, #FACE OFF)

Für die Stirnflächenbearbeitung mit zwei rotatorischen Achsen ist die Kinematik ID 203 erforderlich. Dabei wird das Werkzeug von der rotatorischen Achse Y und der translatorischen Achse Z getragen. Das Werkstück wird über die rotatorische Achse C ausgerichtet.

Die Transformation kann mit dem Befehl #FACE 2ROT geschaltet werden. Der Anwender programmiert im kartesischen System X, Y, Z auf der Stirnfläche des Werkzeuges.

Die C-Achse muss einen Modulo-Bereich von 0° bis 360° haben, siehe P-AXIS-00126 und P-AXIS-00127.

Hinweis

Für die Nutzung dieser Stirnflächenbearbeitung muss zwingend P-CHAN-00262 mit dem Wert 203 für diese Transformation belegt werden.

Syntax:

#FACE 2ROT [AX1=<*Name*>** AX2=**<*Name*>** AX3=**<*Name*>**] | [AXNR1=.. AXNR2=.. AXNR3=..]**

AX1=<*Name*>	Achsbezeichnung der ersten Achse
AX2=<*Name*>	Achsbezeichnung der zweiten Achse
AX3=<*Name*>	Achsbezeichnung der dritten Achse
AXNR1=..	Logische Achsnummer P-AXIS-00016 der ersten Achse
AXNR2=..	Logische Achsnummer P-AXIS-00016 der zweiten Achse
AXNR3=..	Logische Achsnummer P-AXIS-00016 der dritten Achse

Der Modus wird abgewählt durch:

Syntax:

#FACE OFF

15.2.7.3 Mantelflächenbearbeitung (#CYL, #CYL OFF)

Dieser Modus kann bei Drehmaschinen und Bearbeitungszentren angewählt werden. Die gewünschte Kontur auf der zylindrischen Mantelfläche wird in einem virtuellen Koordinatensystem in Millimeter oder Inch programmiert.

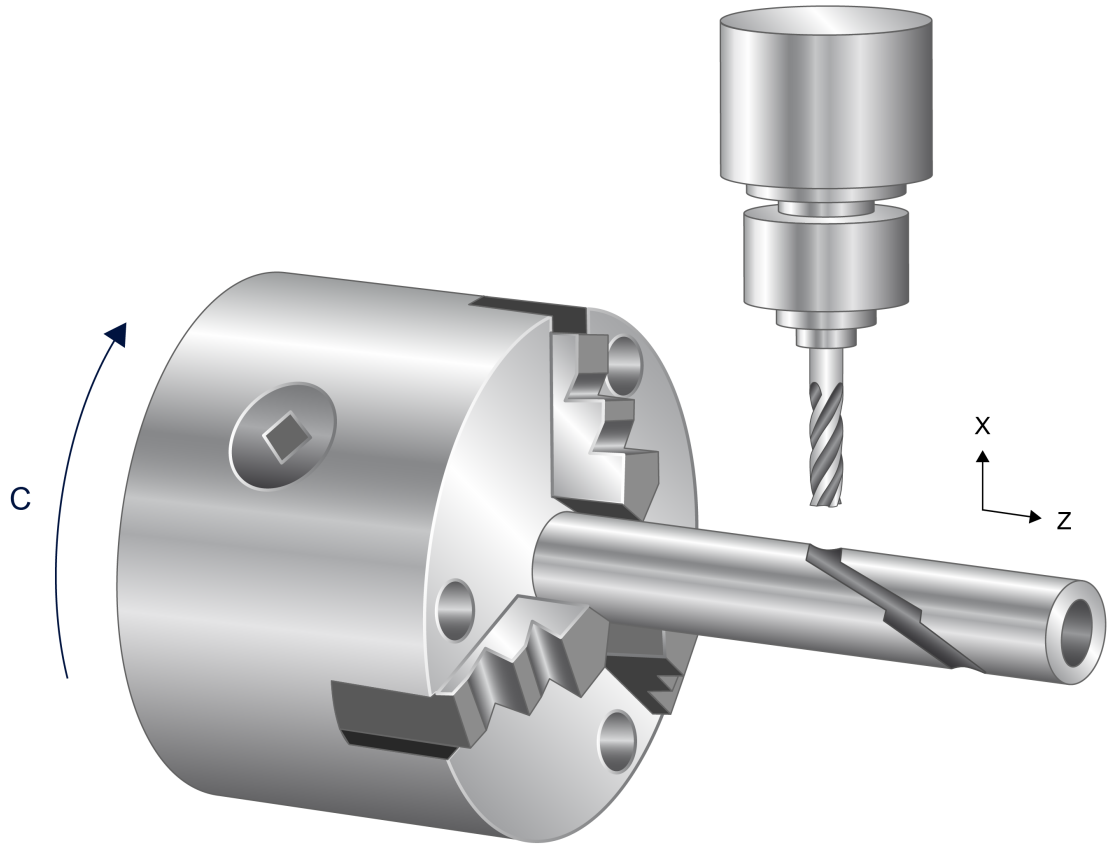
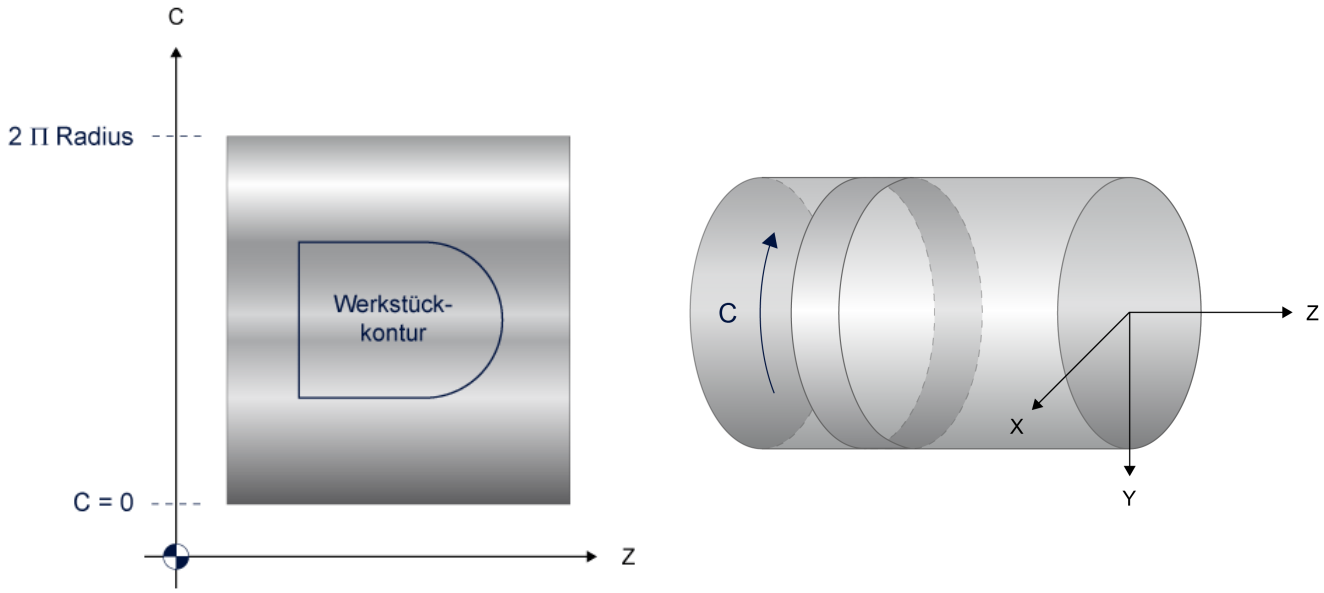


Abb. 169: Mantelflächenbearbeitung

Hinweis

Ab CNC-Version V3.00 muss für die Nutzung der Mantelflächenbearbeitung zwingend der Parameter P-CHAN-00262 mit dem Wert 15 für diese Transformation belegt werden.

Abhängig vom Maschinentyp stehen für die Programmierung der Kontur in kartesischen Koordinaten auf der Mantelfläche die drei logischen Achsen X, Y und Z zur Verfügung. Bei reinen Drehmaschinen ist die Y-Achse in der Regel nicht vorhanden. Zusätzlich muss der Radius des Werkstückes als Bezugsradius R mit programmiert werden.



Bei der Mantelflächenbearbeitung wird die Hauptebene durch Z-C gebildet.

Mantelflächenbearbeitung in G17

Durch die Vorgabe von erster und zweiter Hauptachse mit #CYL [..] wird eine Achskonfiguration Z-C gebildet, die eine Hauptebene implizit in G17 definiert. Hinzu kommt die Angabe des Bezugsradius.

Syntax:

#CYL [<Name 1.Hauptachse>**,** <Name 2.Hauptachse>**,** <Name 3.Hauptachse>..]

<Name 1.Hauptachse>	Achsbezeichnung der ersten Hauptachse entsprechend der aktuellen Hauptebene.
<Name 2.Hauptachse>	Achsbezeichnung der zweiten Hauptachse entsprechend der aktuellen Hauptebene (virtuelle lineare Achse, Abwicklung).
<Name 3.Hauptachse>..	Achsbezeichnung der dritten Hauptachse entsprechend der aktuellen Hauptebene mit Angabe des Bezugsradius in [mm, inch].

Hinweis

Programmierte Mitschleppachsen sind von der Transformation nicht betroffen. Ein Wechsel der Hauptebene während aktiver Mantelflächenbearbeitung mit G18, G19 ist nicht erlaubt

Der Modus wird abgewählt durch:

Syntax:

#CYL OFF

Durch #CYL OFF wird wieder in den zuvor aktiven Zustand zurückgekehrt. D.h., es erfolgt automatisch die Anwahl der zuletzt aktiven Hauptebene und die Wiederherstellung der zuletzt aktiven Achsversätze.

Programmierbeispiel

Programmierbeispiel für Drehmaschine, Programmierung in G17 mit Z-C

Beispiel mit Achsbezeichner "C" für 2. Hauptachse

```
…
#CAX [S, C]         ;Annahme „S" ist Hauptspindel
G01 X60 C45 F800    ;Zustell- und Positionierbewegung; X:60mm C:45°
#CYL [Z, C, X60]    ;Anwahl Mantelflächenbearbeitung
G00 G90 Z0 C0       ;Z: 0mm C:0mm!
G01 C100 F500
G02 Z100 R50
G01 C0
Z0
…
#CYL OFF
#CAX OFF
M30
```

Mantelflächenbearbeitung in G19

Mit #CYL LATERAL [..] wird eine Achskonfiguration hergestellt, die in Kombination mit G19 die Programmierung im virtuellen Koordinatensystem C-Z erlaubt. Hinzu kommt die Angabe des Bezugsradius.

Syntax:

#CYL LATERAL [RADIUS=..]

RADIUS=.. Angabe des Bezugsradius in [mm, inch].

Hinweis

Programmierte Mitschleppachsen sind von der Transformation nicht betroffen. Nach #CYL LATERAL ist auch die Verwendung von G17 bzw. G18 erlaubt, dies kann für Sonderfälle der Bearbeitung notwendig sein.

Der Modus wird abgewählt durch:

Syntax:

#CYL OFF

Mit #CYL OFF wird die Mantelflächenbearbeitung abgewählt und die zuvor aktive Achskonfiguration mit den zugehörigen Achsversätzen wiederhergestellt. Die aktuell gültige Hauptebene bleibt weiterhin aktiv.

Programmierbeispiel

Programmierbeispiele für Drehmaschine, Programmierung in G19 mit C-Z

```
%cyl_lat_A
N020 G00 X0 Y0 Z0
N020 #CAX [S,C]
N030 G00 X0 Y0 Z100 C0
N040 #CYL LATERAL [RADIUS=35]  ;Anwahl Mantelflächenbearbeitung
N060 G19                       ;G19 Ebene anwählen
N070 G01 G90 Z0 C0 F5000
N080 G01 G90 X10   F5000

N090 $FOR P2=1, 5, 1
N100 P3=P2*4
N110 P4=P3+2
N120 G01 G91 C-P3
N130 ZP3
N140 C[2*P3]
N150 Z-P3
N160 G90 C0
N170 G91 ZP4
N180 $ENDFOR

N190 $FOR P2=1, 5, 1
N200 P3=P2*4
N210 P4=P3*2+2
N220 G90 G02 KP3
N230 G91 G01 ZP4
N240 $ENDFOR

N270 #CYL OFF
N280 #CAX OFF
N290 M30
```

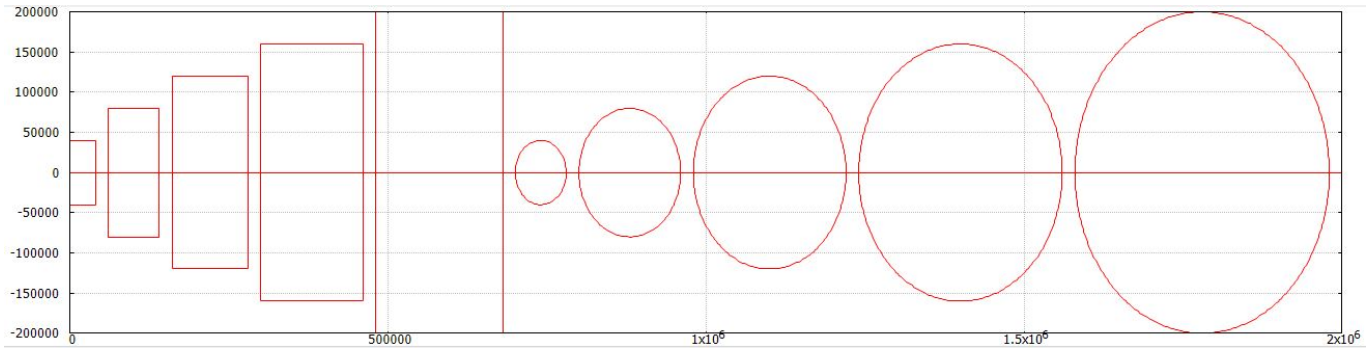


Abb. 170: Beispiel 1 für #CAX LATERAL mit G19

```
%cyl_lat_B
N020 G00 F2000 X0 Y0 Z0
N030 #CAX [S, C]
N040 #CYL LATERAL [RADIUS=20]
N060 G19
N070 G01 F1000 Z0 C0 X0  G161
N080 G02 J30  K0   C60 F2000
N090 G02 J30  K0   C0

N100 G02 J0   K30  Z60
N110 G02 J0   K30  Z0

N120 G03 J-30 K0   C-60
N130 G03 J-30 K0   C0

N140 G03 J0   K-30 Z-60
N150 G03 J0   K-30 Z0

N180 #CYL OFF
N190 #CAX OFF
N210 M30
```

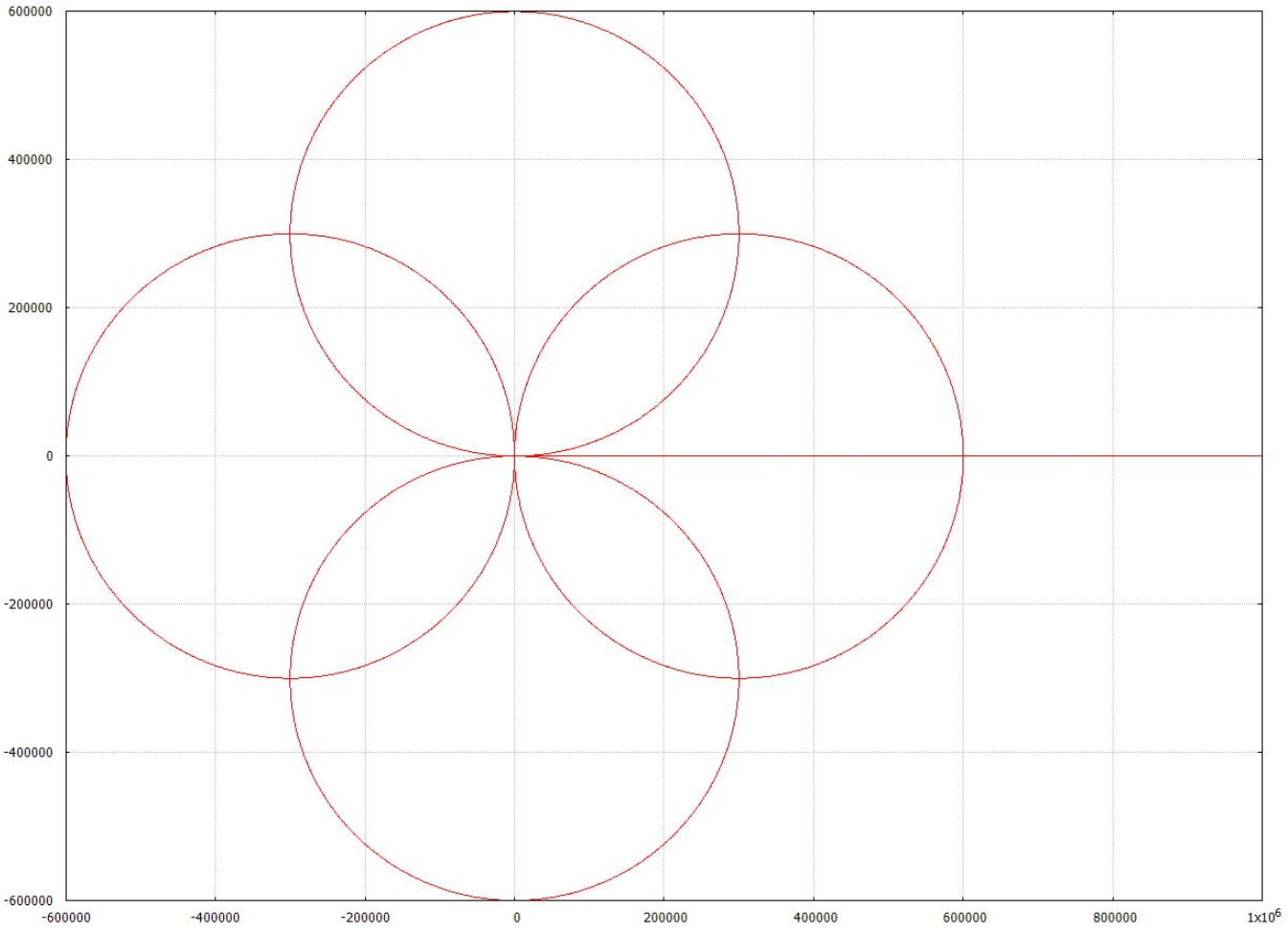


Abb. 171: Beispiel 2 für #CYL LATERAL mit G19

15.2.7.3.1 Mantelflächenbearbeitung mit 2 rotatorischen Achsen (#CYL 2ROT, #CYL OFF)

Drehbearbeitung in kartesischem System. Das Werkzeug wird von den translatorischen Achsen X, Z und der rotatorischen Achse Y getragen. Das Werkstück wird über die rotatorische Achse C ausgerichtet.

Die Transformation kann mit dem Befehl #CYL 2ROT geschaltet werden. Der Anwender programmiert im kartesischen System X, Y, Z. Zusätzlich wird über die Größe C der Winkel des Werkzeuges in der X-Y-Ebene programmiert. Bei C=0 steht das Werkzeug parallel zu X und senkrecht auf Y.

Hinweis

Für die Nutzung dieser Mantelflächenbearbeitung muss zwingend P-CHAN-00262 mit dem Wert 202 für diese Transformation belegt werden.

Die Kinematik ist in zwei Varianten verfügbar.

1. In der Variante LINKS (HD10 = 0) liegt die C-Achse links der Y-Achse. Die Y-Achse zeigt in Nullstellung nach links. Bei der Anwahl der Transformation muss die X-Achse rechts der C-Achse liegen.
2. In der Variante RECHTS (HD10 = 1) liegt die C-Achse rechts der Y-Achse. Die Y-Achse zeigt in Nullstellung nach rechts. Bei der Anwahl der Transformation muss die X-Achse links der C-Achse liegen.

Die C-Achse muss einen Modulo-Bereich von 0° bis 360° haben, siehe P-AXIS-00126 und P-AXIS-00127.

Syntax:

#CYL 2ROT [AX1=<*Name*>** AX2=**<*Name*>** AX3=**<*Name*>**] | [AXNR1=.. AXNR2=.. AXNR3=..]**

AX1=<*Name*>	Achsbezeichnung der ersten Achse
AX2=<*Name*>	Achsbezeichnung der zweiten Achse
AX3=<*Name*>	Achsbezeichnung der dritten Achse
AXNR1=..	Logische Achsnummer P-AXIS-00016 der ersten Achse
AXNR2=..	Logische Achsnummer P-AXIS-00016 der zweiten Achse
AXNR3=..	Logische Achsnummer P-AXIS-00016 der dritten Achse

Der Modus wird abgewählt durch:

Syntax:

#CYL OFF

15.2.7.4 Umschalten zwischen Stirnflächen- und Mantelflächenbearbeitung

Die Abwahl der C-Achsmodis erfolgt regulär über die beschriebenen Abwahlbefehle (z.B. mit dem Befehl #CYL OFF). Zwischen Stirnflächenbearbeitung und Mantelflächenbearbeitung ist es auch möglich, ohne vorherige Abwahl direkt in den anderen Bearbeitungsmodus zu wechseln. Nachfolgendes Programmierbeispiel zeigt dazu eine typische NC-Sequenz für den Wechsel zwischen C-Achsmodis:

Programmierbeispiel

Umschalten zwischen C-Achsmodis

```
N10 #CAX [..]      ;Aufnahme der LR-Spindel in den Bahnverbund
;..............
N120 #FACE [..]    ;Anwahl der Stirnflächenbearbeitung
;..............
N230 #CYL [..]     ;Direkter Übergang zur Mantelflächenbearbeitung
....... ..
N300 #CYL OFF      ;Abwahl der Mantelflächenbearbeitung und Übergang
                   ;zur konvent.Bearbeitung mit physikalischer C-Achse
N400 #CAX OFF      ;Abgabe der C-Achse an die lagegeregelte Spindel
N500 M30
```

15.2.7.5 Werkzeugversätze

Die Befehle #FACE und #CYL führen zur impliziten Kinematikanwahl. Im NC-Programm muss daher weder eine Kinematik-ID mit #KIN ID [▷ 745] [..] noch eine Transformationsanwahl mit #TRAFO ON [▷ 734] erfolgen.

Werkzeugversätze für die Stirnflächenbearbeitung

Die Stirnflächenbearbeitung unterstützt 2 Maschinentypen (Dreh-/Fräsmaschine). Die entsprechenden Werkzeugversätze müssen in den zugeordneten Versatzdaten der Kinematik-ID's 13 oder 14 in den Kanalparametern eingetragen werden. Alternativ, kann dies auch in den Werkzeugdaten erfolgen.

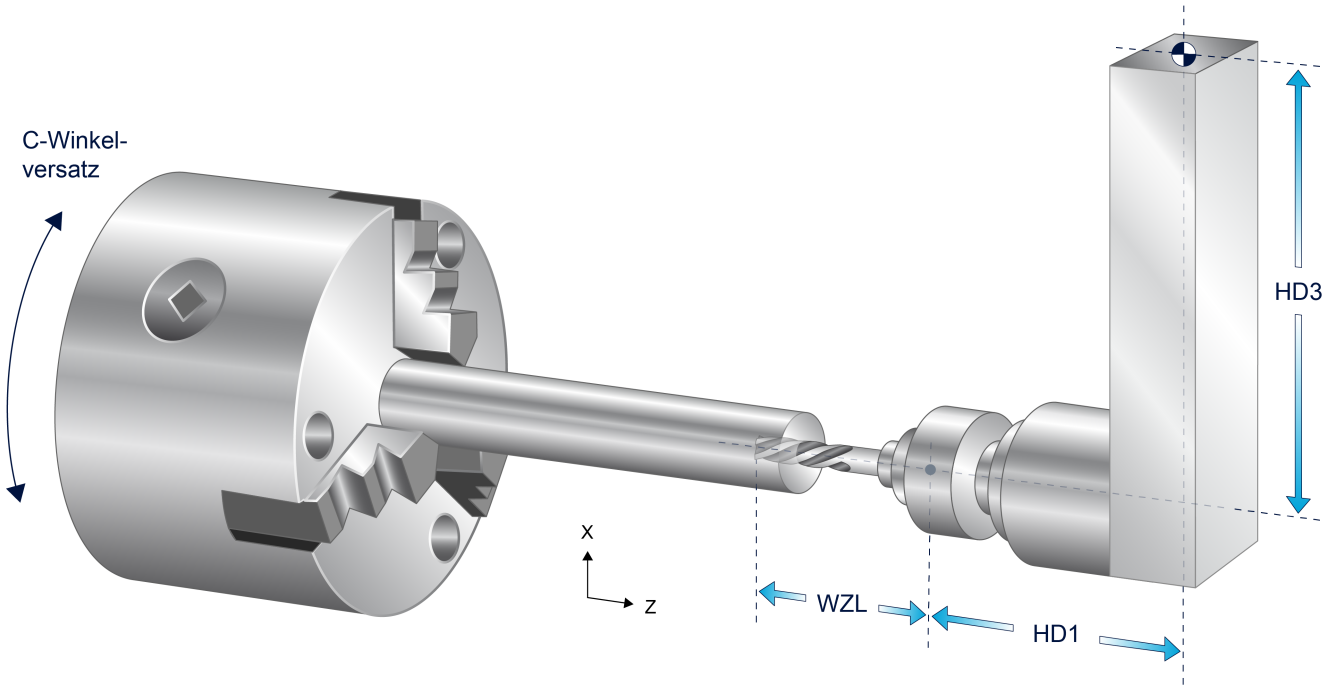


Abb. 172: Werkzeugversätze für die Stirnflächenbearbeitung

Beispiel

Beispiele für Einträge in den Kanalparametern

Für CNC-Versionen ab V3.00

#FACE[], Stirnflächenbearbeitung Drehmaschine (KIN-ID 13):

```
trafo[0].id                    13
trafo[0].param[0]              1080000   Z-Versatz [0.1µm]
trafo[0].param[1]              0         C-Winkelversatz [10^-4°]
trafo[0].param[2]              900000    X-Versatz[0.1µm]
```

#FACE[], Stirnflächenbearbeitung Fräsmaschine (KIN-ID 14):

```
trafo[1].id                    14
trafo[1].param[0]              1080000   Z-Versatz [0.1µm]
trafo[1].param[1]              0         C-Winkelversatz [10^-4°]
trafo[1].param[2]              900000    X-Versatz [0.1µm]
```

CNC-Versionen < V3.00

#FACE[], Stirnflächenbearbeitung Drehmaschine (KIN-ID 13):

```
kinematik[13].param[0]         1080000   Z-Versatz [0.1µm]
kinematik[13].param[1]         0         C-Winkelversatz [10^-4°]
kinematik[13].param[2]         900000    X-Versatz[0.1µm]
```

#FACE[], Stirnflächenbearbeitung Fräsmaschine (KIN-ID 14):

```
kinematik[14].param[0]         1080000   Z-Versatz [0.1µm]
kinematik[14].param[1]         0         C-Winkelversatz [10^-4°]
kinematik[14].param[2]         900000    X-Versatz [0.1µm]
```

Werkzeugversätze für die Mantelflächenbearbeitung

Die Mantelflächenbearbeitung führt implizit die Anwahl der Kinematik mit der ID 15 durch.

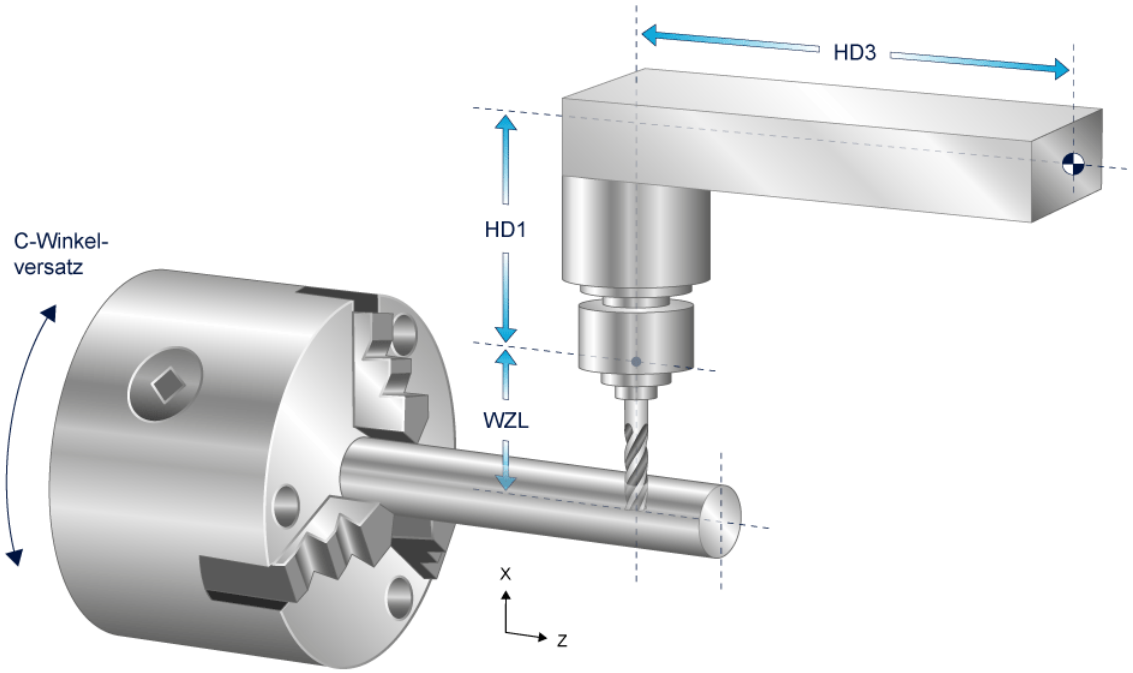


Abb. 173: Werkzeugversätze für die Mantelflächenbearbeitung

#CYCL[], Mantelflächenbearbeitung Drehmaschine (KIN-ID 15):

CNC-Versionen ab V3.00

```
trafo[0].id                    15
trafo[0].param[0]              700000   X-Versatz [0.1µm]
trafo[0].param[1]              0        C-Winkelversatz [10⁻⁴°]
trafo[0].param[2]              1200000  Z-Versatz [0.1µm]
```

CNC-Versionen < V3.00

```
kinematik[15].param[0]         700000   X-Versatz [0.1µm]
kinematik[15].param[1]         0        C-Winkelversatz [10⁻⁴°]
kinematik[15].param[2]         1200000  Z-Versatz [0.1µm]
```

15.2.8 Getriebeschalten (G112)

Syntax:

G112 <*Spindelname*>.. nicht modal

G112	Getriebe schalten
<*Spindelname*>..	Getriebedatensatz bestehend aus Spindelbezeichnung gemäß Kanalparameterliste und Nummer des Datensatzes

Im Gegensatz zum Getriebeschalten über M40-45, bei dem implizit auch der mechanische Schaltvorgang mit durchgeführt wird, wird durch das Programmieren von G112 zusammen mit dem S-Wort nur das Aktualisieren von Spindelgetriebedaten (Dynamikwerte) einer Stufe angestoßen.

Das mechanische Schalten der richtigen Getriebestufe muss der Anwender explizit z.B. durch selbst definierte M-Funktionen programmieren oder wie bereits erwähnt über M40-45.

Achtung

Vor dem Schalten von Spindelgetriebedaten muss die Spindel stehen. Dies kann durch einen Spindelstopp (M5) oder eine Spindelpositionierung (M19...) im vorhergehenden NC-Satz erreicht werden.

Programmierbeispiel

Getriebeschalten (G112)

```
%Test_G112

N010 G112 S2    (Dynamikdaten der Getriebestufe 2 für Spindel „S" laden)
N020 M3 S9000
N030 M5         (Spindel stoppen)
N040 G112 S1    (Dynamikdaten der Getriebestufe 1 für Spindel „S" laden)
N050 M4 S8000
N060 M30
```

15.2.9 Referenzpunktfahrt im DIN-Syntax (G74)

Syntax:

G74 <*Spindelname*>.. nicht modal

G74	Referenzpunktfahrt
<*Spindelname*>..	Spindelbezeichnung entsprechend P-CHAN-00053 mit Angabe eines Wertes.

Für lagegeregelte Spindeln kann eine Referenzpunktfahrt durchgeführt werden. Im Gegensatz zur Referenzierung von Linearachsen haben die mit dem Spindelnamen programmierten Werte keine Bedeutung für die Reihenfolge der Referenzierung, sondern werden nur zur Darstellung einer vollständigen Syntax benötigt.

Die Programmierung der Spindel-M-Funktionen ist im selben NC-Satz wie G74 nicht erlaubt.

Folgende Fälle sind bei der Referenzierung von Spindeln zu unterscheiden:

Programmierbeispiel

Referenzpunktfahrt (G74)

```
Fall 1:
Die Referenzierung der Spindel beginnt zeitgleich mit der
Y-Achsen-Referenzierung:
  Nxx G74 X2 Y1 S1

Fall 2:
Wie 1.! Ohne abzuwarten, bis die Spindel referenziert ist, wird mit dem
nächsten NC-Satz fortgefahren, so dass die X-Achse quasi-gleichzeitig
referenziert wird:
  Nxx G74 S1
  Nyy G74 X1 Y2

Fall 3:
Zuerst werden die Achsen X und Y referenziert. Danach beginnt die Spin-
del-Referenzierung:
  Nxx G74 X1 Y2
  Nyy G74 S1
```

15.2.10 Spindeloverride DIN-Syntax (G167)

Syntax:

G167 <*Spindelname*>.. nicht modal

G167	Spindeloverride auf 100 % festsetzen
<*Spindelname*>..	Spindeldrehzahl bestehend aus Spindelbezeichnung entsprechend P-CHAN-00053 und Angabe eines Wertes. Der Wert hat keine Bedeutung, sondern dient nur zur Darstellung einer vollständigen Syntax.

Mit der Funktion G167 wird die externe Beeinflussung des Spindeloverrides ausgeschaltet und die tatsächlich programmierte Drehzahl gefahren.

Programmierbeispiel

Override (G167)

```
:
N10 M3 S1000      (Bei Override 50% ist Drehzahl 500 U/min)
N20 G167 S1000    (Overridebeeinflussung aus, Spindel dreht mit 1000 U/
min)
N30 S3000         (Overridebeeinflussung wieder aktiv, Drehzahl 1500 U/
min)
Nxx
```

15.3 Programmierung in spindelspezifischer Syntax

Die spindelspezifische Syntax bietet den Vorteil, dass in einem NC-Satz mehrere Spindeln unabhängig voneinander programmiert werden können.

Diese erfolgt innerhalb einer dem Spindelnamen angefügten Klammerung. In dieser Klammerung sind nur bestimmte Befehle zulässig, die stets spindelspezifisch behandelt und ausgeführt werden. Die Hauptspindel kann nur mit ihrem Hauptspindelnamen programmiert werden (P-CHAN-00053).

Syntax:

<*Spindelname*> **[** [**M3** | **M4** | **M5** | **M19**] **REV**..] [**POS**..] { **M**.. } { **H**.. } [**G74**] [**G167**]
[**CALLAX** | **PUTAX**] [**GET_DYNAMIC_DATA**] [**G130**] [[**G135** | **G137**] [**G136**..]]
[**FEED_LINK..**] [**OVERRIDE**..] { \ } **]**
{ <*Spindelname*> **[** .. **]** }

<*Spindelname*>	Spindelbezeichnung gemäß [1] [▷ 894]-3 und P-CHAN-00053
M3, M4, M5, M19	Spindel-M-Funktionen
REV=..	Spindeldrehzahl
POS=..	Spindelposition
M=..	Anwenderspezifische M-Funktionen
H=..	Anwenderspezifische H-Funktionen
G74	Referenzpunktfahrt
G167	Spindeloverride 100%
CALLAX	Holen Spindelachse
PUTAX	Abgeben Spindelachse
GET_DYNAMIC_DATA	Übernahme neuer Werkzeugdynamikdaten
G130	Beschleunigungsgewichtung
G135, G136=.., G137	Vorsteuerung
FEED_LINK…	Spindelvorschubkopplung
OVERRIDE…	Spindeloverride
\	Trennzeichen ("Backslash") für übersichtliche Programmierung des Befehls über mehrere Zeilen

Programmierbeispiel

Programmierung in spindelspezifischer Syntax

```
:
N10 S[M3 REV500 M19 POS45 M18 M15 H20 …]     S2[M4 REV5000]
Nxx
:
```

15.3.1 Die Spindel-M-Funktionen

15.3.1.1 Spindel bewegen in spindelspezifischer Syntax (M3/M4/M5)

Syntax:

M03	Spindeldrehung im Uhrzeigersinn (cw)	modal
M04	Spindeldrehung im Gegenuhrzeigersinn (ccw)	modal
M05	Spindel stoppen	modal

Die Spindel-M-Funktionen M03, M04 legen die Spindeldrehrichtung fest und sind im Zusammenhang mit der Spindeldrehzahl (REV-Wort) zu verwenden. Mit M05 wird die Spindeldrehung gestoppt. Es ist zu beachten, dass diese Spindel-M-Funktion der Default-Spindelmodus nach Steuerungshochlauf und erstem Programmstart ist. Diese M-Funktionen sind haltend wirksam und dürfen innerhalb der Klammerung jeweils nur alleine programmiert werden.

Die Spindeldrehung wird aktiviert, wenn M03 oder M04 programmiert wurde und eine gültige Drehzahl (REV) gesetzt ist.

Wird am Programmende kein M05 gesetzt, so dreht die Spindel weiter.

Programmierbeispiel

Programmierung einer Spindel „S“:

```
N10 S[REV1000]      (Drehzahl 1000 U/min wird gespeichert,)
                    (keine Spindeldrehung da M05 Default)
N20 S[M03]          (Spindeldrehung cw mit 1000 U/min)
N30 S[M04]          (Spindeldrehung ccw mit 1000 U/min)
N40 S[REV500]       (Spindeldrehung ccw mit 500 U/min)
N50 S[M05 REV300]   (Spindelstopp, Drehzahl 300 U/min wird gespeichert)
N60 S[M04]          (Spindeldrehung ccw mit 300 U/min)
N70 S[M05]          (Spindelstopp)
N80 S[M03 REV1000]  (Spindeldrehung cw mit 1000 U/min)
N90 M30             (Programmende)
```

Programmierbeispiel

Programmierung von zwei Spindeln „S“ und „S2“:

```
N10 S[M03 REV1000]  S2[M04 REV2000]  (S cw 1000 U/min,S2 ccw 2000 U/min)
N20 S[M05]          S2[REV1500]      (S stop, S2 ccw mit 1500 U/min)
N30 S[M04]                           (S ccw 1000 U/min)
N40                 S2[M05]          (S2 stop)
N50 S[M05 REV300]                    (S stop, Drehzahl 300 U/min spei-
chern)
N60 S[M04]          S2[M04]          (S ccw 300 U/min, S2 ccw 1500 U/
min)
N70 S[M05]          S2[M05]          (S stop, S2 stop)
N80 M30                              (Programmende)
```

Kanalparametersatz [1] [▶ 894]:

Für M3, M4, M5 müssen die Synchronisationsarten spindelspezifisch festgelegt werden. Bei der Synchronisationsart „0“ (NO_SYNCH) wird die M-Funktion nicht ausgeführt.

```
:
spindel[0].bezeichnung                          S1
spindel[0].log_achs_nr                          6
spindel[0].s_synch                              0x00000001
spindel[0].m3_synch                             0x00000002
spindel[0].m4_synch                             0x00000002
spindel[0].m5_synch                             0x00000008
spindel[0].m19_synch                            0x00000001
```

15.3.1.2 Spindel positionieren in spindelspezifischer Syntax (M19, POS)

Syntax:

M19	Spindel positionieren	nicht modal
POS=..	Spindelposition in [°]	modal

Die Spindel-M-Funktion M19 bewirkt die Spindelpositionierung und ist im Zusammenhang mit dem POS-Wort zu verwenden. M03/M04 bzw. die Spindeldrehzahl (REV) im gleichen NC-Satz sind optional. Es muss jedoch eine gültige Spindeldrehzahl (> 0) gesetzt sein. M19 darf nicht zusammen mit M5 (Spindelstopp) verwendet werden.

Die Spindelposition POS in [°] ist haltend und muss bei einer erneuten Programmierung von M19 nicht nochmals angegeben werden. Wurde bisher noch keine Spindelposition programmiert, so wird per Standard auf Position „Null“ gefahren.

Rotiert die Spindel nicht, wird die Positionierung mit dem kürzesten Verfahrweg durchgeführt.

Spindelpositionierung mit M19 ist nur für lagegeregelte Spindeln erlaubt.

Programmierbeispiel

Programmierung einer Spindel „S“:

```
N10 S[REV100 POS45 M19 M3]          (Fahre cw mit 100 U/min auf Position
45)
N20 S[M03 REV1000]                  (Spindeldrehung cw mit 1000 U/min)
N30 S[M19 M4 REV150]                (Fahre ccw mit 150 U/min auf Position
45)
                                    (->POS ist haltend!)
N40 S[M05]                          (Spindelstopp)
N50 S[M19]                          (Fahre auf kürzestem Weg mit 150 U/min
auf)
                                    (Position 45)
N60 S[REV200 M19 POS90]             (Fahre auf kürzestem Weg mit 200 U/min
auf)
                                    (Position 90)
N70 S[M03 REV1000]                  (Spindeldrehung cw mit 1000 U/min)
N80 S[M05]                          (Spindelstopp)
N90 M30                             (Programmende)
```

Programmierbeispiel

Programmierung von zwei Spindeln „S“ und „S2“:

```
(Zu N10: Fahre beide Spindeln cw mit 1000 U/min auf Position 45)
N10 S[M03 REV200 M19 POS45] S2[M04 REV200 POS45 M19]
N20 S[REV1000] S2[REV1500]              (S E FAC 1000 U/min,)
                                        (S2 ccw mit 1500 U/min)
N30 S[M04]                              (S ccw 1000 U/min)
N40 S2[M19 POS0]                        (Fahre S2 ccw auf Posiion 0)
N50 S[M05 REV300]                       (S stop, 300 U/min speichern)

(zu N60: Fahre S cw mit 300 U/min auf Position 90,)
(zu N60: Fahre S2 cw mit 200 U/min Position 45)
N60 S[M03 M19 POS90] S2[M03 REV200 POS45 M19]
N80 M30                                 (Programmende)
```

Kanalparametersatz [1] [▶ 894]:

Für M19 muss die Synchronisationsart spindelspezifisch festgelegt werden. Bei der Synchronisationsart „0“ (NO_SYNCH) wird die M-Funktion nicht ausgeführt.

```
:
spindel[0].bezeichnung                  S1
spindel[0].log_achs_nr                  6
spindel[0].s_synch                      0x00000001
spindel[0].m3_synch                     0x00000002
spindel[0].m4_synch                     0x00000002
spindel[0].m5_synch                     0x00000008
spindel[0].m19_synch                    0x00000001
```

15.3.2 Spindeldrehzahl (REV)

Syntax:

REV=.. Spindeldrehzahl in [U/min] modal

Dem REV-Wort können Werte direkt oder per Parameter in [U/min] zugewiesen werden, wobei auch Dezimalzahlen (REAL-Format) zulässig sind.

In Verbindung mit den Spindel-M-Funktionen sind beim REV-Wort folgende Verwendungsarten zu unterscheiden:

1. REV-Wort in Verbindung mit M03, M04, M19:
   Wird das REV-Wort in Verbindung mit M03/M04 oder M19 programmiert, so wird der dem REV-Wort folgende Wert als Spindeldrehzahl interpretiert und an die Spindel ausge-geben.
2. REV-Wort in Verbindung mit M05:
   Zusammen mit M05 wird der dem REV-Wort folgende Wert als Spindeldrehzahl in die Arbeits-daten übernommen, aber nicht an die Spindel ausgegeben.

Das REV-Wort alleine erzeugt noch keine Bewegung im NC-Programm. Hierfür muss ein Spin-delmodus M03, M04, M19 bekannt sein. Entsprechend wird auch die Programmierung von M03, M04 und M19 erst eine Bewegung bewirken, wenn das REV-Wort gesetzt wird.

Hinweis

Bei einem negativen REV-Wert wird eine Fehlermeldung ausgegeben.

Programmierbeispiel

Programmierung mit Spindel S1:

```
N10 S1[REV300]          (Drehzahl 300 U/min wird gespeichert)
N20 S1[M04]             (Spindeldrehung ccw mit 300 U/min)
N30 S1[M03 REV1000]     (Spindeldrehung cw mit 1000 U/min)
N40 S1[REV500]          (M03 aktiv, somit Spindeldrehung cw mit 500 U/
min)
N50 S1[M05 REV100]      (Spindelstopp, Drehzahl 100 U/min wird gespei-
chert)
N60 S1[M04]             (Spindeldrehung ccw mit 100 U/min)
N70 S1[M05]             (Spindelstopp)
N80 M30                 (Programmende)
```

Kanalparametersatz [1] [▶ 894]:

Für das S-Wort muss die Synchronisationsart spindelspezifisch festgelegt werden. Bei der Syn-chronisationsart „0“ (NO_SYNCH) wird eine Fehlermeldung erzeugt, da ein S-Wort nicht unbe-rücksichtigt bleiben darf.

```
:
spindel[0].bezeichnung                  S1
spindel[0].log_achs_nr                  6
spindel[0].s_synch                      0x00000001
spindel[0].m3_synch                     0x00000002
spindel[0].m4_synch                     0x00000002
spindel[0].m5_synch                     0x00000008
spindel[0].m19_synch                    0x00000001
```

15.3.3 Anwenderspezifische M-/H-Funktionen in spindelspezifischer Syntax

Alle innerhalb der Klammerung programmierten anwenderspezifischen M-/H-Funktionen (Technologieinformationen) werden **immer** spindelspezifisch behandelt und ausgegeben.

Programmierbeispiel

Programmierung einer Spindel „S2“ mit zwei anwenderspezifischen M-Funktionen

```
N10 S2[M3 REV300 M10 M11]   (S2 dreht cw mit 300 U/min, gibt M10/ M11
aus)
N20 M30                     (Programmende)
```

Programmierbeispiel

Programmierung von zwei Spindeln „S“ und S2:

```
N10 S[M10 M11] S2[REV1000 M3 M11]   (S führt M10 und M11 aus, S2 dreht)
                                    (cw mit 1000 U/min, gibt M11 aus)
N20 M30                             (Programmende)
```

Kanalparametersatz [1] [▶ 894]:

Die anwenderspezifischen M-/H-Funktionen werden in P-CHAN-00027 und P-CHAN-00041 parametriert. Bei der Synchronisationsart „0“ (NO_SYNCH) wird die M-/H-Funktion nicht ausgeführt.

```
:
# Festlegung der M-Funktionen und Synchronisationsarten
:
# Festlegung der M-Funktionen und Synchronisationsarten
m_synch[1]                  0x00000001                  MOS
m_synch[2]                  0x00000002                  MVS_SVS
:
m_synch[10]                 0x00000002                  MVS_SVS
m_synch[11]                 0x00000008                  MVS_SVS
m_synch[12]                 0x00000004                  MVS_SVS

m_synch[48]                 0x00000008                  MNS_SNS
m_synch[49]                 0x00000002                  MVS_SVS
:                           :                           :
# Festlegung der H-Funktionen und Synchronisationsarten
h_synch[1] 0x00000001 MOS
h_synch[2] 0x00000001 MOS
:
```

15.3.4 Referenzpunktfahrt in spindelspezifischer Syntax (G74)

Die Referenzpunktfahrt kann spindelspezifisch durchgeführt werden. Die Programmierung der Spindel-M-Funktionen ist zusammen mit G74 nicht erlaubt.

Programmierbeispiel

Programmierung von zwei Spindeln „S“ und „S2“:

```
N10 S[G74] S2[G74]              ;Referenzpunktfahrt S und S2
N20 M30                         ;Programmende
```

15.3.5 Spindeloverride im spindelspezifischem Syntax (G167)

Mit der Funktion G167 wird die externe Beeinflussung des spindelspezifischen Overrides ausgeschaltet und die tatsächlich programmierte Drehzahl gefahren. Die Wirkung des programmierten Overridewertes für eine Spindel [▶ 704] bleibt erhalten.

Programmierbeispiel

Programmierung einer Spindel „S2“:

```
N10 S2[M3 REV1000]     ;Bei Override 50% ist Drehzahl 500 U/min
N20 S2[G167 REV1000]   ;Overridebeeinflussung aus,
                       ;Spindel dreht mit 1000 U/min
N30 S2[REV3000]        ;Overridebeeinflussung wieder aktiv,
                       ;Drehzahl 1500 U/min
N40 M30                ;Programmende
```

15.3.6 Abgeben/Anfordern von Spindelachsen (PUTAX/CALLAX)

Versionshinweis

Die Verfügbarkeit dieser Funktionalität ist von der Konfiguration und dem Versionsumfang abhängig.

Mit diesen Befehlen kann spindelspezifisch das Abgeben oder Holen der Spindelachse programmiert werden. Die Befehle dürfen nicht gleichzeitig zusammen mit anderen spindelspezifischen Befehlen verwendet werden.

Programmierbeispiel

```
%s-putcallax
(Move axis as spindle)
N10  S[CALLAX]
N20  M03 S1000
N30  G04 X2
N40  M05

(Exchange axis from spindle interpolator in channel)
N50  S[PUTAX]
N60  #CALL AX[ C, 4, 3]
(Drive axis in channel)
N70  X10 Y20 Z30 C40
N80  X-10 Y-20 Z-30 C--40
(Exchange axis from channel in spindle interpolator)
N90  #PUT AX[C]
N100 S[CALLAX]
(Move axis again as spindle)
N110 M04 S1000
N120 G04 X2
N130 M05
N140 M30
```

Programmierbeispiel

Programmierung mehrerer Spindeln:

```
:
N10 S[CALLAX] S2[CALLAX] S3[PUTAX] (S, S2 holen ihre Achse,
                                    (S3 gibt ihre ab)
N20 S[M3 REV200] S2[G74)            (S dreht cw mit 200 U/min),
                                    (S2 referenziert)
N30 S3[ M4 REV3000]        (Fehler, da S3 momentan ohne Achse)
N40 S[PUTAX REV400)        (Fehler, PUTAX muss allein programmiert sein)
:
```

15.3.7 Übernahme der Werkzeugdynamikdaten (GET_DYNAMIC_DATA/ DEFAULT_DYNAMIC_DATA)

Versionshinweis

Die Verfügbarkeit dieser Funktionalität ist von der Konfiguration und dem Versionsumfang abhängig.

Die Werkzeugdynamikdaten (Minimale-/Maximale Drehzahl, Max. Beschleunigung) werden nach der Programmierung eines neuen Werkzeuges (D-Wort, #TOOL DATA) automatisch beim Übergang der Spindel vom Stillstand in die Interpolation wirksam. Die Übernahme und Berücksichtigung geänderter Werkzeugdynamikdaten bei drehender Spindel erfolgt durch den spindelspezifischen Befehl "GET_DYNAMIC_DATA".

Durch den Befehl "DEFAULT_DYNAMIC_DATA" kann bei stehender Spindel wieder auf werkzeugunabhängige Standarddynamikdaten zurückgeschaltet werden. Die aktuell eingestellte Getriebestufe (P-TOOL-00016/P-TOOL-00017) wird dabei nicht geändert.

Diese Befehle dürfen nicht gleichzeitig und nicht in Kombination mit anderen spindelspezifischen Befehlen verwendet werden.

Programmierbeispiel

Übernahme der Werkzeugdynamikdaten

```
N10 T1 D1                    ;Bereitstellen von WZ-Dynamikdaten in den
                             ;WZ-Daten
N15 M6                       ;Einwechseln von Werkzeug 1
N20 S[M3 REV2002]            ;S dreht mit 2000 U/min und WZ-Dynamikdaten
                             ;aus D1
N25 X100 Y100                ;Verfahrsatz mit WZ-Dynamikdaten D1
N30 T99 D99                  ;Bereitstellen neuer WZ-Dynamikdaten in den
                             ;WZ-Daten bei drehender Spindel (N20)
N35 X200 Y150                ;Verfahrsatz mit WZ-Dynamikdaten D1
N40 S[GET_DYNAMIC_DATA]      ;Übernahme WZ-Dynamikdaten D99
N45 X150 Y200                ;Verfahrsatz mit WZ-Dynamikdaten D99
N50 X50 Y50                  ;Verfahrsatz mit WZ-Dynamikdaten D99
N60 S[M05] Z100              ;Spindel stop
N70 S[DEFAULT_DYNAMIC_DATA]  ;Auf Standard-Dynamikdaten setzen

N99 M30                      ;Programmende
```

15.3.8 Beauftragung der Spindelvorsteuerung (G135/G136/G137)

Versionshinweis

Die Verfügbarkeit dieser Funktionalität ist von der Konfiguration und dem Versionsumfang abhängig.

Mit diesen Befehlen kann spindelspezifisch die Vorsteuerung programmiert werden. Die Befehle dürfen nicht gleichzeitig zusammen mit anderen spindelspezifischen Befehlen verwendet werden.

Die Aktivierung wird mit G135 programmiert.

Eine spindelspezifische, prozentuale Gewichtung der berechneten Vorsteuergrößen erfolgt über G136. Sie ist auf 100% begrenzt.

Mit G137 wird die Vorsteuerung ausgeschaltet. Es ist auch möglich, die Anwahl und Gewichtung der Vorsteuerung im gleichen Satz anzugeben.

Wenn die Vorsteuerung während des NC-Programms ein- bzw. ausgeschaltet wird, bleiben die Gewichtungsfaktoren auf dem durch G136 eingestellten Werten bzw., wenn kein G136 programmiert wird, auf 100%.

Programmierbeispiel

Beauftragung der Spindelvorsteuerung

```
S[G135]                ;Aktivierung der Vorsteuerung für S
S[G136=80]             ;Angabe der Vorsteuergewichtung in Prozent
S[G137]                ;Deaktivierung der Vorsteuerung
S2[G135 G136=90]       ;Aktivierung mit Gewichtung 90% für S2
S2[G136=0]             ;Änderung der Gewichtung auf 0%
S1[G135]               ;Aktivierung mit Standardgewichtung 100% für S1
```

15.3.9 Spindelvorschubkopplung (FEED_LINK)

Normalerweise wird die Drehzahl einer Spindel nur durch das Programm gesteuert, eine Möglichkeit der Beeinflussung der Spindeldrehzahl in Abhängigkeit von anderen Einflussgrößen besteht im Allgemeinen nicht. Für bestimmte Technologien (z.B. Fräsen im HSC-Betrieb, Kantenleimen in der Holzverarbeitung) und zur Erreichung eines einheitlichen Fräsbildes auf der Werkstückoberfläche ist jedoch auch eine Beeinflussung der Spindeldrehzahl durch externe Einflussgrößen notwendig.

Mit dem nachfolgenden Befehl kann die spindelspezifische Drehzahl unter Vorgabe verschiedener Parameter dynamisch an den Bahnvorschub gekoppelt werden. Bei entsprechender Einstellung passt sich dann die Spindeldrehzahl unterschiedlichen Vorschubgeschwindigkeiten automatisch an. Dies ist insbesondere bei Werkstoffen notwendig, die durch nicht optimale Schnittwerte eventuell geschädigt werden können (z.B. Brandstellen an Werkstücken aus Holz oder Verlaufen durch Schmelzen bei Kunststoff etc.).

Achtung

Die Dynamikdaten der Spindel werden nicht berücksichtigt. Damit bei aktiver Vorschubkopplung keine zu großen Abweichungen zwischen Soll- und Ist- Kopplungsfaktoren auftreten, muss der Anwender sicherstellen, dass entweder die Spindeldynamik ausreichend hoch ist oder die Bahndynamik entsprechend den Anforderungen reduziert wird.

Syntax zur Programmierung einer Spindelvorschubkopplung:

<*Spindelname*> **[FEED_LINK** [**ON** | **OFF**] [[[**FACT=**..] [**CORR=**..]] | **AUTO**] [**MAIN**] **SRC=**.. **]**

<*Spindelname*>	Name der zu koppelnden Spindel
FEED_LINK	Kennung für die Spindelvorschubkopplung. Muss immer als **erstes** Schlüsselwort programmiert sein.
ON	Anwahl der Spindelvorschubkopplung
OFF	Abwahl der Spindelvorschubkopplung
FACT=..	**Synchronisierte** Kopplung: Vorgegebener Koppelfaktor in [0.1%] zwischen Bahnvorschub (F-Wort) und der Ausgangsdrehzahl der Spindel. Der Kopplungsfaktor kann während aktiver Kopplung jederzeit neu programmiert werden.
CORR=..	Korrekturfaktor in [0.1%] zur Modifikation des Kopplungsfaktors, der resultierende Kopplungsfaktor ergibt sich dann als Produkt von FACT und CORR (Priorität = FACT*CORR)
AUTO	**Fliegende** Kopplung: Koppelfaktor wird automatisch aus aktueller Solldrehzahl der Spindel und aktuellem Bahnvorschub bestimmt und bleibt bis zur Koppelabwahl konstant.
MAIN	Spindelvorschubkopplung wirkt nur, wenn mindestens eine Hauptachse an der Bewegung beteiligt ist. Werden nur Mitschleppachsen bewegt, bleibt die Spindeldrehzahl unbeeinflusst.
SRC=..	Kennung der Quelle der Vorschubankopplung: • FEED_VEL – Bahnvorschub (Standard) • EDGE_VEL – Kantengeschwindigkeit beim Kantenanleimen

- Das Schlüsselwort AUTO kann nur zusammen mit ON programmiert werden. FACT und CORR können auch bei bereits aktiver Vorschubkopplung erneut programmiert werden.
- Das Schlüsselwort MAIN kann nur zusammen mit ON programmiert werden.
- Bei Anwahl der Kopplung muss sich die Spindel im Endlosdrehen mit Drehzahl !=0 befinden. Der Bahnvorschub muss bei automatischer Bestimmung des Kopplungsfaktors ebenfalls !=0 sein.
- Die bei Abwahl der Vorschubkopplung aktive Spindeldrehzahl wird als Spindelsolldrehzahl beibehalten. Ggf. muss die Spindeldrehzahl also nach Abwahl der Kopplung über das S Wort neu beauftragt werden.

Achtung

Eine Spindelbeauftragung über das S-Wort ist bei aktiver Vorschubkopplung wirkungslos!

Hinweis

Es ist ein Werkzeug mit Dynamikparametern zu definieren und zu aktivieren. Über die Mindestdrehzahl (P-TOOL-00013) dieses Werkzeugs kann dann im Allgemeinen verhindert werden, dass die Spindel beim Fräsen mit aktiver Vorschubkopplung und Bahngeschwindigkeit 0 zum Stillstand kommt und ein „Festfressen“ des Werkzeugs auftritt.

Programmierbeispiel

Beispiel 1

Koppelfaktor wird manuell programmiert, Änderung des wirksamen Koppelfaktors durch Programmierung des Korrekturfaktors

```
%feed_link1

N10 G00 X0 Y0 Z0
N20 G01 G90 X20 F2
N30 M03 S15
N40 G01 G90 X40
N50 S[FEED_LINK ON FACT=500]
N60 G01 G91 X5 F2
N70 S[FEED_LINK CORR=1100]
N80 X10 F2
N90 S[FEED_LINK CORR=900]
N100 X20 F2
N110 S[FEED_LINK CORR=1100]
N120 X40 F2
N130 S[FEED_LINK CORR=900]
N140 X80 F2
N150 S[FEED_LINK CORR=1100]
N160 X40 F2
N170 S[FEED_LINK CORR=900]
N180 X20 F2
N190 S[FEED_LINK CORR=1100]
N200 X5 F2
N210 S[FEED_LINK OFF]
N220 M30
```

Programmierbeispiel

Beispiel 2

Koppelfaktor wird automatisch aus aktueller Solldrehzahl der Spindel und aktuellem Bahnvorschub

```
%feed_link2

N10 G00 X0 Y0 Z0
N20 G01 G90 X20 F1
N30 M03 S15
N40 G01 G90 X40
N50 S[FEED_LINK ON AUTO]
N60 G01 G91 X5 F1
N70 X10 F2
N80 X20 F4
N90 X40 F8
N100 X80 F16
N110 X40 F8
N120 X20 F4
N130 X10 F2
N140 X5 F1
N150 S[FEED_LINK OFF]
N160 M30
```

Programmierbeispiel

Beispiel 3

Koppelfaktor wird automatisch bestimmt, Vorschubkopplung wirkt nur bei Hauptachsbewegungen

```
%feed_link3

N10 M03 S200
N20 G01 X0 Y0 Z0 C0 F10
N30 S[FEED_LINK ON AUTO MAIN]
N40 G01 X50
N50 G01 X70 Y30
N60 G01 C45 ;Spindeldrehzahl bleibt konstant
N70 G01 Z40 F15
N80 G01 Z60
N90 G01 C90       ;Spindeldrehzahl bleibt konstant
N100 G01 C120     ;Spindeldrehzahl bleibt konstant
N110 G01 X50 Y50 Z50
N120 S[FEED_LINK OFF]
N130 M03 S200
N140 G01 X10
N150 M05 S0
N160 M30
```

15.3.10 Programmierbarer Spindeloverride (OVERRIDE)

Mit dieser Funktion kann die Spindeldrehzahl im NC-Programm beeinflusst werden. Der spindelspezifische programmierte Overridewert ist wirksam, wenn sich die entsprechende Spindel bewegt.

Ist zusätzlich auch noch ein externer Spindeloverride definiert, so ergibt sich der wirksame Override aus der Multiplikation der beiden Overridewerte.

Hinweis

Die Funktion G167 [▶ 697] unterdrückt nur die Wirkung des externen Overridewertes.

Syntax:

<*Spindelname*> **[OVERRIDE SPEED_FACT**=.. **]**

<*Spindelname*>	Name der Spindel
OVERRIDE	Kennung für die spindelspezifische Overrideprogrammierung. Muss immer als **erstes** Schlüsselwort programmiert sein.
SPEED_FACT=..	Overridefaktor für Drehzahlsätze [0.1%-200%]

Programmierbeispiel

Programmierbarer Spindeloverride (OVERRIDE)

```
%spdl_override
N10 G01 X100 Y100 Z100 F1000
N20 S[M3 REV1000]
N40 S[OVERRIDE SPEED_FACT=20]            ;Spindeloverride 20%
N50 X0                                   ;G01 Bewegung mit S200
N60 Y0                                   ;G01 Bewegung mit S200
N70 Z0                                   ;G01 Bewegung mit S200
N60 S[OVERRIDE SPEED_FACT=100]           ;Spindeloverride 100%
N90 Y100                                 ;G01 Bewegung mit S1000
N100 Z100                                ;G01 Bewegung mit S1000
N110 X200 Y200                           ;G01 Bewegung mit S1000
N120 X300 Y300 Z200                      ;G01 Bewegung mit S1000
N130 M5
M30
```

15.3.11 Beschleunigungsgewichtung (G130)

Mit den Funktionen G130 kann die Beschleunigung der Spindelachse verändert werden.

Eine Beeinflussung der Beschleunigung wird erreicht, indem man die entsprechenden Beschleunigungskennwerte prozentual verändert. Bei ruckbegrenztem Profil sind dies die Achsparameter P-AXIS-00001 und P-AXIS-00002.

Bei einer Programmierung mit G130 sind alle nicht bzw. noch nicht programmierten Achsen auf 100% eingestellt. Jede weitere Anwahl dieser Funktionen bezieht sich unabhängig von vorhergehenden Programmierungen auf 100%.

Zweimal hintereinander 50% programmiert bedeutet also, dass auf 50% und nicht auf 25% eingestellt wird. Eine Gewichtung über 100% hinaus ist bis zur Maximalbeschleunigung der Achse P-AXIS-00008 möglich.

Hinweis

Der Wert der Gewichtung bleibt über das Programmende erhalten und muss manuell zurückgesetzt werden.

Programmierbeispiel

Beschleunigungsgewichtung (G130)

```
N10 S[G130=70]  ;Beschleunigung der Spindel wird auf 70% begrenzt
N20 M03 S1000   ;Endlosdrehen CW
N30 S[G130=60]  ;Beschleunigung der Spindel wird auf 60% begrenzt
N40 M04 S1000   ;Endlosdrehen CCW
; …
N120 S[G130=100] ;Zurücksetzen des Wertes auf 100%
M30
```

15.4 Wechsel der Hauptspindel (#MAIN SPINDLE)

Syntax:
#MAIN SPINDLE [**[** <*Spindelname*> | <*Spindelnummer*> **]**]

<*Spindelname*>	Standardspindelbezeichnung gemäß [1] [▶ 894]-3.
<*Spindelnummer*>	log. Achsnummer der Spindel gemäß [1] [▶ 894]-3.

Über den Befehl #MAIN SPINDLE kann im NC-Programm die Festlegung der Hauptspindel geändert werden. Die neue Hauptspindel wird durch Angabe des Standardnamens (P-CHAN-00053) oder der zugehörigen logischen Achsnummer angewählt.

Ohne Angabe einer Spindel kann der Grundzustand (wie nach Hochlauf) wieder hergestellt werden, d.h. die im Kanalparameter P-CHAN-00051 vorgegebene Spindel wird wieder zur Hauptspindel.

Kanalparametersatz [1] [▶ 894]:

Beispielkonfiguration eines 1-kanaligen Systems mit 3 Spindeln. Spindel mit der Achsnummer 6 ist Hauptspindel:

```
# Spindeldaten
# ============
spdl_anzahl                3

main_spindle_ax_nr         6
main_spindle_name          S

spindel[0].bezeichnung     S1
spindel[0].log_achs_nr     6

spindel[1].bezeichnung     S2
spindel[1].log_achs_nr     11

spindel[2].bezeichnung     S3
spindel[2].log_achs_nr     30
```

Konfiguration nach Hochlauf:

S1 ist Hauptspindel mit dem Namen „S“.

„S2“ und „S3“ sind weitere Spindeln.

Programmierbeispiel

Wechsel der Hauptspindel

```
%
N10 S100 M3 S2[REV200 M3] S3[REV300 M4]
N20 #MAIN SPINDLE [S2]      (S2 wird neue Hauptspindel "S")
N30 S110 M3 S1[REV210 M3] S3[REV310 M4]
N40 #MAIN SPINDLE [S3]     (S3 wird neue Hauptspindel "S")
N50 S120 M3 S1[REV220 M3] S2[REV320 M4]
N60 #MAIN SPINDLE           (Zurück in Grundzustand S1 -> "S")
N70 S150 M3 S2[REV250 M3] S3[REV350 M4]
N80 M5 S2[M5] S3[M5]        (Alle Spindeln STOP)
N99 M30
```

Hinweis

Solange eine Spindel eine Hauptspindel ist, kann sie entweder mit dem festgelegten Hauptspindelnamen P-CHAN-00053 oder mit ihrem Standardnamen [1] [▷ 894]-3 programmiert werden. Erst nach Anwahl einer anderen Hauptspindel durch #MAIN SPINDLE[ ] ist sie wieder ausschließlich unter ihrem Standardnamen ansprechbar.

Für obiges Beispiel gilt:

Zulässige Name	Spindel 1	Spindel 2	Spindel 3
...nach Hochlauf	**S** oder **S1**	S2	S3
…nach #MAIN SPINDLE [S2]	S1	**S** oder **S2**	S3
…nach #MAIN SPINDLE [S3]	S1	S2	**S** oder **S3**
...nach #MAIN SPINDLE	**S** oder **S1**	S2	S3

Wie bereits erwähnt, kann die Hauptspindel in der herkömmlichen DIN-Syntax programmiert werden. In diesem Fall können alle Befehle gemäß der Tabelle im Kapitel Spindelprogrammierung [▷ 635] verwendet werden. Die Hauptspindel kann aber auch in der spindelspezifischen Syntax programmiert werden. In diesem Fall steht dann jedoch nur der eingeschränkte Befehlsumfang zur Verfügung (siehe ebenfalls in der Tabelle im Kapitel Spindelprogrammierung. [▷ 635]

Programmierbeispiel

Folgende NC-Zeilen sind für die Hauptspindel gleichermaßen zulässig:

```
:
N10 S1=1000 M3   oder
N20 S1000 M3   oder
N30 S1[REV1000 M3]   oder
N40 S[REV1000 M3]
:
```

15.5 Spindelsynchronbetrieb

Neben dem Synchronbetrieb von Bahnachsen (Definition, Aktivierung, Deaktivierung) können über den LINK-Befehl auch Master-/ Slavebeziehungen für Spindelachsen definiert werden.

Syntax:

#SET AX LINK [<*Kopplungsgruppe*>, <*Slave*> = <*Master*> {, <*Slave*> = <*Master*> }]

oder alternativ

#AX LINK [NBR] [<*Kopplungsgruppe*>, <*Slave*> = <*Master*> {, <*Slave*> = <*Master*> }]

<*Kopplungsgruppe*>	Nummer der Kopplungsgruppe (1)
<*Slave*>	Bezeichnung oder logische Achsnummer der Slavespindel des Kopplungspaares i (2)
<*Master*>	Bezeichnung oder logische Achsnummer der Masterspindel des Kopplungspaares i (2)
NBR	Mit dem Logikschalter NBR wird auf die Auswertung von logischen Achsnummern anstatt von Spindelnamen umgeschaltet. Die Achskopplungen müssen dann über die logischen Achsnummern definiert werden.

Hierbei gelten ergänzend zum Kapitel Synchronbetrieb [▶ 359] folgende Regeln:

- Die Kopplungspaare werden über die Namen bzw. logischen Achsnummern der Spindeln, die im Kanal bekannt sind, definiert. D.h. es können ausschließlich die im Kanal bekannten Spindeln gekoppelt werden.
- Im Kanalparameter [1] [▶ 894]-2 kann die Kopplungsgruppe 0 als Defaultgruppe vorbelegt werden. Diese ist nach dem Hochlauf direkt durch #ENABLE AX LINK bzw. #AX LINK ON ansprechbar. Sie kann nicht im NC-Programm neu definiert werden.

Programmierbeispiel

Spindelsynchronbetrieb, Programmierung und An-/Abwahl einer Spindelkopplung

Parametrierung in den Kanalparametern [1] [▶ 894]: S (Hauptspindelname für S1), S1, S2, S3. Die Kopplungspaare dürfen mit den Spindelnamen S, S2, S3 gebildet werden.

```
N10 #SET AX LINK[1, S2=S, S3=S] ;Hauptspindel ist Master für S2+S3
N20 #ENABLE AX LINK[1]          ;Anwahl der Spindelkopplungen
N30 S1000 M3            ;Hauptspindel S+S2+S3 drehen cw 1000 U/min
N40 #DISABLE AX LINK    ;Abwahl der Spindelkopplungen

oder alternativ

N10 #AX LINK[1, S2=S, S3=S]
N20 #AX LINK ON[1]
N30 S1000 M3
N40 #AX LINK OFF

oder alternativ

N10 #AX LINK NBR[1, 11=6, 17=6] ;Kopplung über log. Achsnummern
N20 #AX LINK ON[1]
N30 S1000 M3
N40 #AX LINK OFF
N50 M30                         ;Programmende
```

(1) 1 ... [Max. Anzahl Kopplungsgruppen–1], siehe [6] [▶ 894]-2.11

(2) Max. Anzahl Kopplungspaare , siehe [6] [▶ 894]-2.12

- In einem #SET AX LINK bzw. #AX LINK Befehl dürfen innerhalb eines Kopplungspaares keine unterschiedlichen Achstypen definiert werden. Die Kopplungspaare dürfen jedoch zusammen mit Kopplungspaaren anderen Achsentyps eine Kopplungsgruppe bilden.

Beispiele:

#SET AX LINK [1, B=S, S2=X] **FALSCH**

#SET AX LINK [1, B=X, S2=S] **ERLAUBT**

- Während einer aktiven Kopplungsgruppe darf eine Spindel, die in dieser Gruppe vorhanden ist, nicht mit #MAIN SPINDLE [..] zur Hauptachse deklariert werden, da sonst Inkonsistenzen zwischen Decoder und Interpolator entstehen.
- Des Weiteren muss dem Programmierer bewusst sein, dass durch Anwendung von #MAIN SPINDLE [..] eventuell bereits definierte Kopplungsgruppen nicht mehr aktiviert werden dürfen, weil die Spindelnamen nicht mehr konsistent sind.
- Die Technologie-Information für M03, M04, M05 werden vom NC-Kern sowohl für die Master- als auch die Slavespindel synchronisiert.
- Die externe Bewegungsbeeinflussung der Masterspindel durch FEEDHOLD und OVERRIDE wirkt nicht auf die Slavespindeln. Auch während einer aktiven Kopplung werten die Slavespindeln weiterhin ihre eigenen OVERRIDE- und FEEDHOLD-Schnittstellen aus.
- Master- und Slavespindeln fahren bei M19 auf die gleiche absolute Position. Die Position wird evtl. nicht zeitgleich erreicht, wenn die Startposition oder die dynamischen Daten der Spindeln unterschiedlich sind.
- Bei einem Programmabbruch wird die Kopplung aufgehoben.
- Wenn eine Spindel als Master oder als Slave aktiv ist, darf sie nur von dem Kanal, der den Link aktiviert hat, beauftragt werden.

Kanalparametersatz [1] [▶ 894]:

```
:
# Vorbelegung möglicher Achs-Links für Synchronbetrieb
# ================================================
#synchro_data.koppel_gruppe[0].paar[0].log_achs_nr_slave 4
#synchro_data.koppel_gruppe[0].paar[0].log_achs_nr_master 1
#synchro_data.koppel_gruppe[0].paar[0].mode 0 ->AX_LINK
#synchro_data.koppel_gruppe[0].paar[1].log_achs_nr_slave 11
#synchro_data.koppel_gruppe[0].paar[1].log_achs_nr_master 6
#synchro_data.koppel_gruppe[0].paar[1].mode 1 ->SPDL_LINK
:
```

15.6 Satzübergreifende Synchronisierung (Late Sync)

15.6.1 Implizite Synchronisierung

Beschleunigungs- und Verzögerungsvorgänge der Spindel können zu erheblichen Totzeiten bei der Programmbearbeitung führen, da häufig beim Stillstand der Bearbeitung die Spindel erst auf Drehzahl gebracht wird (z.B. M03 vom Typ MVS_SVS), bzw. während eines Positioniersatzes mit G00 (M3 vom Typ MVS_SNS).

Für M-Funktionen besteht über die implizite Synchronisation die Möglichkeit, die Quittierung erst zu prüfen, wenn auf eine Bearbeitung mit G01/G02/G03/G151 etc. umgeschaltet wird. Dieses Verhalten wird mit der Synchronisationsart MVS_SLM erreicht. Die Kennung kann nur exklusiv zu den anderen Synchronisationsarten verwendet werden (P-CHAN-00027).

Programmierbeispiel

Implizite Synchronisierung

```
:
N10 G00 M03 S1000 Z600                (M03: Synchronisationsmode MVS_SLM)
N20 X100 Y100
N30 Z400
N40 G01 Z200                          (Prüfen, ob M03 quittiert ist)
:
```

In N40 prüft der Interpolator auf Quittierung der M-Funktion zu Beginn des Bremszeitpunktes. Wenn die Quittierung erfolgt ist, wird am Satzende nicht angehalten. Ist keine Quittierung erfolgt, so wird verzögert und wenn bis zum Satzende keine Quittierung erfolgt ist, wird im Zielpunkt angehalten.

Bis zur Synchronisierung durch einen Bewegungssatz ist es möglich, weitere kanalspezifische M-Funktionen zu programmieren. Die Synchronisierung der kanalspezifischen M-Funktionen wird vollständig parallel zur achsspezifischen Synchronisierung behandelt.

15.6.2 Explizite Synchronisierung (#EXPL SYN)

Wird zum Positionieren G01 verwendet, so kann keine implizite Synchronisierung gemäß Kap. Implizite Synchronisierung [▶ 710] durchgeführt werden.

Zur satzübergreifenden Synchronisierung steht dazu der Befehl #EXPL SYN zur Verfügung, der eine explizite Synchronisierung von M-Funktion ermöglicht.

Syntax:

#EXPL SYN nicht modal

Eine M-Funktion. die über diesen zusätzlichen Befehl zu synchronisieren ist, wird in P-CHAN-00027 mit der Synchronisationsart MVS_SLP definiert. Die Kennung kann nur exklusiv zu den anderen Synchronisationsarten verwendet werden (P-CHAN-00027).

Programmierbeispiel

Explizite Synchronisierung

```
:
N10 G01 M03 S1000 Z200 F5000    ;M03: Synchronisationsmode MVS_SLP
N20 X100 Y100
N30 Z400
N40 #EXPL SYN                   ;Prüfen, ob M03 quittiert ist
:
```

Zum Bremszeitpunkt prüft die Bahn aufgrund der Anweisung "#EXPL SYN" ob die Quittierung eingetroffen ist, wenn dies nicht der Fall ist wird abgerampt.

Vor dem Synchronisierungsbefehl können weitere kanalspezifische und achsspezifische M-Funktionen bearbeitet werden.

15.7 Synchronisierung der Spindel-M-Funktionen

Die Synchronisation zwischen dem Interpolator und der jeweiligen Spindel erfolgt direkt, d.h. die Quittierung von M03, M04 (Drehzahl erreicht) und M05 (Drehzahl Null) erfolgt durch die Spindel selbst. Das Bit PLC_INFO, das zusätzlich zu den bestehenden Synchronisationsarten P-CHAN-00027 gesetzt werden kann, bestimmt ob auch mit der SPS quittiert werden soll. Dabei ist folgendes zu beachten:

Bei gesteuerten Spindeln wird generell bei jeder Spindel-M-Funktion automatisch auch mit der SPS quittiert. Es ist somit nicht erforderlich, das PLC_INFO-Bit zusätzlich zu setzen.

Sinnvoll ist die Verwendung des PLC_INFO-Bits bei geregelten Spindeln. Hier kann für jede Spindel-M-Funktion zusätzlich zur Synchronisationsart das Bit PLC_INFO gesetzt werden, wodurch dann eine Quittierung mit der SPS angestoßen wird.

Kanalparametersatz [1] [▶ 894]:

Spindel S1 soll geregelte Spindel sein.

```
:
spindel[0].bezeichnung                S1
spindel[0].log_achs_nr                6
spindel[0].s_synch                    0x00020001    PLC_INFO, MOS
spindel[0].m3_synch                   0x00020002    PLC_INFO, MVS_SVS
spindel[0].m4_synch                   0x00020004    PLC_INFO, MVS_SNS
spindel[0].m5_synch                   0x00020002    PLC_INFO, MVS_SVS
spindel[0].m19_synch                  0x00000008              MNS_SNS
spindel[0].s_prozess_zeit             0
spindel[0].m3_prozess_zeit            0
spindel[0].m4_prozess_zeit            0
spindel[0].m5_prozess_zeit            0
spindel[0].m19_prozess_zeit           0
:
```

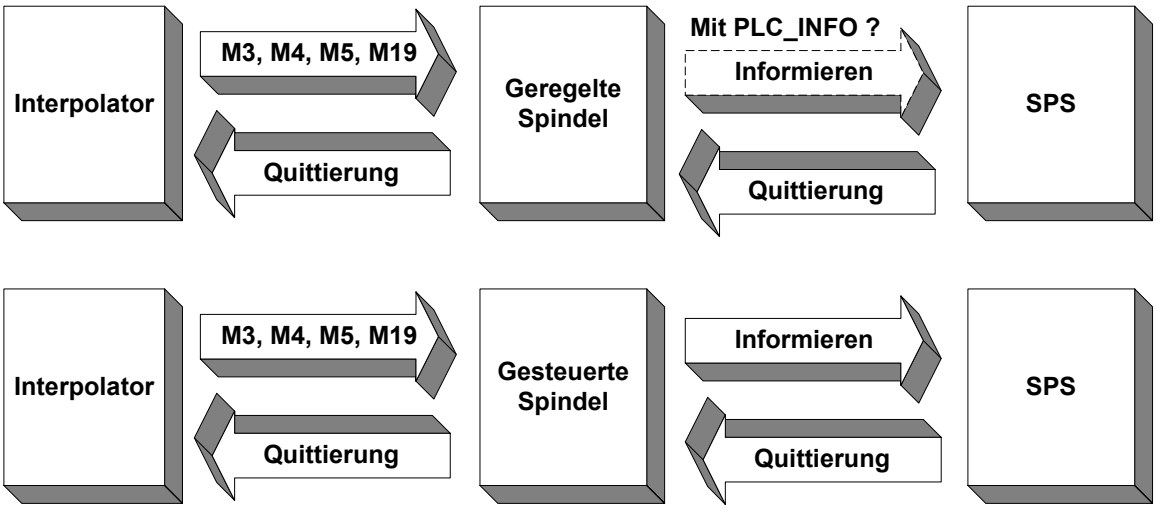


Abb. 174: Darstellung der Synchronisierung der Spindel-M-Funktion

15.8 PLCopen-Programmierung

Eine vollständige Liste der PLCopen-Funktionen findet sich in der Befehlsübersicht im Anhang unter PLCopen-Programmierung [▶ 713].

Innerhalb der Motion Control Platform (MCP) werden für die Bewegungsaufgaben verschiedene Funktionsbausteine (FB) zur Verfügung gestellt. Diese FB wirken auf eine einzelne Achse und werden über die SPS bedient. Jede Achse ist hierbei als s.g. Single Axis Interpolator (SAI) im System angelegt.

Solche Achsen können alternativ auch über das NC-Programm angesprochen werden, da ein solcher SAI wie eine herkömmliche Spindel im System angelegt wird. Für die nachfolgenden FBs stehen dazu spezielle NC-Befehle zur Verfügung, über die eine PLCopen konforme Programmierung in NC-Syntax möglich ist:

MC_Home	Referenzpunktfahrt
MC_MoveAbsolute	Achsbewegung auf absolute Position
MC_MoveAdditive	Relative Achsbewegung zur kommandierten Position
MC_MoveRelative	Relative Achsbewegung zur aktuellen Position
MC_MoveSuperImposed	Relative Achsbewegung zu einer bereits aktiven Bewegung
MC_MoveVelocity	Endlose Achsbewegung mit einer Geschwindigkeit
MC_Stop	Anhalten einer Achsbewegung
MC_GearIn	Getriebekopplung mit Übersetzung
MC_GearOut	Lösen einer Getriebekopplung
MC_Phasing	Phasenverschiebung von Kopplungen
MC_TouchProbe	Messen von Achspositionen

Folgendes Topologiebild zeigt die grundsätzliche Anordnung der SAI-(Spindel-)Achsen im Gesamtsystem:

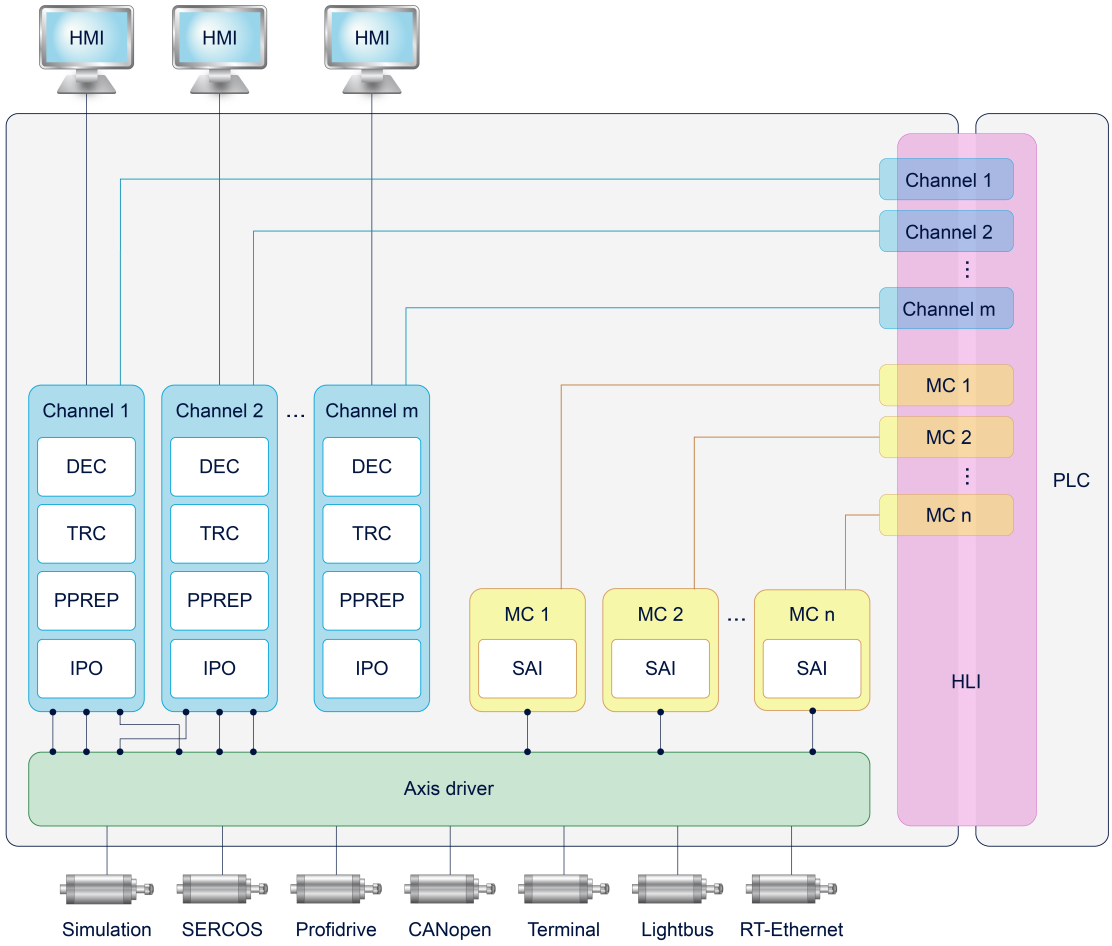


Abb. 175: SAI-Achsen in CNC-Topologie

Eine SAI-Achse wird im NC-Programm in spindelspezifischer Programmiersyntax angesprochen. Dazu muss sie analog zur Konfiguration einer Spindel in der Kanalparameterliste mit ihrem Adressbuchstaben und weiteren Daten im NC-Kanal bekannt gemacht werden. Wichtige Einträge in den Kanalparametern sind insbesondere:

- spdl_anzahl (P-CHAN-00082) – Gesamtzahl der (SAI-)Spindeln
- bezeichnung (P-CHAN-00007) – Name der (SAI-)Spindel
- log_achs_nr (P-CHAN-00036) – Logische Achsnummer der (SAI-)Spindel

Weitere Informationen können der Dokumentation [1] [▶ 894]-Kapitel: Konfiguration der Spindeln- und dem Kapitel Parametrierung von Spindeln [▶ 637] entnommen werden.

Die PLCopen-Funktionen

- MC_MoveSuperImposed
- MC_GearIn
- MC_GearOut
- MC_Phasing
- MC_TouchProbe

erfordern zusätzliche spezifische SAI-Eigenschaften der (Spindel-)Achse, die in den Achsparametern konfiguriert werden. Die notwendigen Einstellungen können der Dokumentation [2] [▷ 894]-Kapitel: Einstellungen für SAI- entnommen werden.

Im Folgenden wird zum jeweiligen NC-Befehl der zugehörige FB dargestellt. Die Syntax dieser NC-Befehle sowie die Einheiten der programmierten Werte lehnen sich dabei an die entsprechenden Eingangspins (VAR_INPUT) der zugeordneten FB's an.

`Allgemeine Syntax eines (SAI-)NC-Befehls:`

*<Spindelname>***[** *<FB-Name> <Eingangspin1> <Eingangspin2> <Eingangspin n...>* { \ } **]**

Der Achsname zu Beginn des NC-Befehls adressiert die (SAI)-Spindelachse, die über den NC-Kanal angesprochen wird.

Die Beschreibungen der Eingangspins sowie der Einheiten und Wertebereiche können auch der Dokumentation [9] [▷ 894] entnommen werden.

Hinweis

Alle Werte der Eingangspins werden in metrischen Einheiten programmiert. In der Grundeinstellung müssen die Werte in den spezifizierten internen Einheiten (z.B. 0.1 µm) angegeben werden. Mit P-CHAN-00182 kann auf die Angabe der Werte in Standardeinheiten (z.B. mm) umgeschaltet werden.

Der Eingangspin "Execute" wird immer implizit durch die Programmierung des NC-Befehls belegt. Es ist daher in den NC-Befehlen kein eigenes Schlüsselwort für diesen Pin vorhanden.

Für eine übersichtliche Programmierung des Befehls über mehrere Zeilen kann innerhalb der Klammerung [...] das Zeilentrennzeichen '\' verwendet werden.

Die Schlüsselworte "Id" und "WaitSyn" für die Job-Synchronisierung im NC-Programm haben in der PLC keine enstprechenden PINs. Die beiden Schlüsselworte stehen ab der CNC-Version V3.01.3100.01 zur Verfügung

Standardmäßig werden PLCopen-Funktionen unabhängig von der übrigen NC-Programmbearbeitung ausgeführt. Eine Synchronisierung zwischen PLCopen-Einzelachsaufträgen und der Bahnbewegung erfolgt nicht.

Für die Synchronisierung der PLCopen-Funktionen mit dem Programmablauf können jedoch Wartebedingungen definiert werden. Hierfür existieren zwei Möglichkeiten.

1) Synchronisierung am Satzende:

Die Angabe des Schlüsselworts "WaitSyn" veranlasst die CNC, vor dem Weiterschalten in den nächsten NC-Satz auf den Abschluss des PLCopen-Auftrags zu warten. Sind in der NC-Zeile mehrere PLCopen-Aufträge programmiert, erfolgt die Weiterschaltung erst, sobald alle Aufträge, für die das Schlüsselwort "WaitSyn" angegeben ist, fertig sind.

Syntax:

<*Spindelname*>[<*FB-Name*> [**WaitSyn**] <*Eingangspin2*> <*Eingangspin n...*> { \ }]

Programmierbeispiel

```
N10 G01 X100 F10000
N20 S[MC_MoveAbsolute WAIT_SYN POSITION=900000 …]
;Weiterschaltung in Satz N30 erfolgt sobald die Spindel S die ;Position
90° erreicht hat
N30 G01 X200 F100
```

2) Späte Synchronisierung:

Über das Schlüsselwort "Id" kann einem PLCopen Auftrag eine Job-ID zugeordnet werden. Mit dem Befehl #WAIT MC_STATUS SYN [ID<JobNr>] kann zu einem späteren Zeitpunkt auf das Ende des PLCopen-Auftrags gewartet werden.

Syntax:

<*Spindelname*>**[** <*FB-Name*> [**Id**=..>] <*Eingangspin2*> <*Eingangspin n...*> { \ } **]**

#WAIT MC_STATUS SYN [ID=.. { **ID**=.. } **]**

Falls mehrere PLCopen-Aufträge mit identischen Jobnummern gestartet werden, so wird die Jobnummer der zuletzt beauftragen PLCopen-Funktion zugeordnet. Eine späte Synchronisierung auf Auftragsende ist dann nur für den zuletzt gestarteten Auftrag möglich.

Programmierbeispiel

```
N10 G01 X100 F10000
N20 S[MC_MoveAbsolute Id100 POSITION=900000 …]
N30 G01 X200
N40 S2[MC_MoveVelocity Id200 Velocity=10000 …}
N50 G01 X300
N60 #WAIT MC_STATUS SYN [ID100 ID200]
; Weiterschaltung in Satz N70 erfolgt, falls Spindel S die
; Position 90° und Spindel S2 die Drehzahl 10°/s erreicht hat
N70 G0 X0
```

15.8.1 Befehl MC_Home

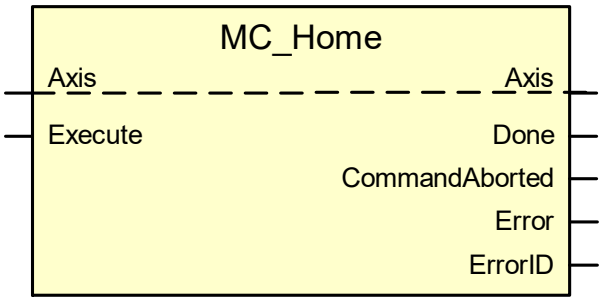
Mit MC_Home wird eine Referenzpunktfahrt (Justage) der Achse beauftragt. Wie eine Achse auf diese Beauftragung reagiert, hängt im Wesentlichen von der Art des Referenzierungsvorganges ab.

Optional kann auf den Abschluss des Auftrags gewartet (Schlüsselwort "WaitSyn") oder für eine spätere Synchronisierung eine Job-ID (Schlüsselwort "Id") vergeben werden.

Syntax NC-Befehl:

<*Achsname*>**[MC_Home** [**Id**=..] [**WaitSyn**] { \ } **]**

Blockdiagramm des Funktionsbausteins in PLCopen:



Programmierbeispiel

Befehl MC_Home

```
S[MC_Home]
```

15.8.2 Befehl MC_MoveAbsolute

Mit MC_MoveAbsolute wird eine Bewegung der Achse auf eine absolute Position beauftragt. Die Bewegung wird immer ruckbegrenzt mit dem in "Jerk" eingestellten konstanten Ruck ausgeführt. Der Wert gilt sowohl für das Beschleunigen mit "Acceleration", als auch für das Bremsen mit "Deceleration".

Werden die optionalen Parameter "Acceleration", "Deceleration" und "Jerk" nicht angegeben oder ≤ 0 gesetzt, so werden die Dynamikwerte aus der entsprechenden Achsliste übernommen.

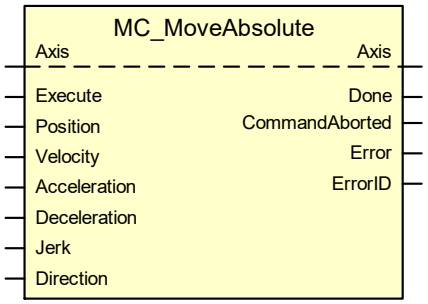
Optional kann auf den Abschluss des Auftrags gewartet (Schlüsselwort "WaitSyn") oder für eine spätere Synchronisierung eine Job-ID (Schlüsselwort "Id") vergeben werden.

Syntax NC-Befehl:

<*Achsname*>**[MC_MoveAbsolute Position**=.. **Velocity**=.. [**Acceleration**=..]
[**Deceleration**=..] [**Jerk**=..] **Direction**=.. [**Id**=..] [**WaitSyn**] { \ } **]**

Blockdiagramm des Funktionsbausteins in PLCopen:

Eingangspin	Einheit
Position	[0,1µm or 10^{-4} °]
Velocity	[1µm/s or 10^{-3} °/s]
Acceleration	[1mm/s² or 1°/s²]
Deceleration	[1mm/s² or 1°/s²]
Jerk	[1mm/s³ or 1°/s³]
Direction	1 Positive Richtung 2 Kürzester Weg 3 Negative Richtung 4 Derzeitige Richtung



Programmierbeispiel

MC_MoveAbsolute

```
S[MC_MoveAbsolute Position=133 Velocity=1000 Acceleration=500 \
Deceleration=600 Jerk=20000 Direction=2]
```

15.8.3 Befehl MC_MoveAdditive

üüüüüMit MC_MoveAdditive wird eine relative Bewegung beauftragt, zuzüglich zur kommandierten Position, wenn die Achse im Zustand 'Discrete Motion' ist. Die Bewegung wird immer ruckbegrenzt mit dem in "Jerk" eingestellten konstanten Ruck ausgeführt. Der Wert gilt sowohl für das Beschleunigen mit "Acceleration", als auch für das Bremsen mit "Deceleration".

Werden die optionalen Parameter "Acceleration", "Deceleration" und "Jerk" nicht angegeben oder ≤ 0 gesetzt, so werden die Dynamikwerte aus der entsprechenden Achsliste übernommen.

Befindet sich die Achse im Zustand 'Continuous Motion' und erhält eine Beauftragung durch diesen Befehl, wird die relative Strecke zur aktuellen Position zum Zeitpunkt der Beauftragung addiert.

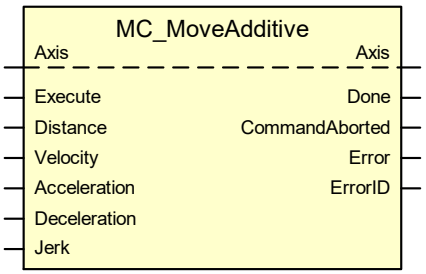
Optional kann auf den Abschluss des Auftrags gewartet (Schlüsselwort "WaitSyn") oder für eine spätere Synchronisierung eine Job-ID (Schlüsselwort "Id") vergeben werden.

Syntax NC-Befehl:

<*Achsname*>**[MC_MoveAdditive Distance**=.. **Velocity**=.. [**Acceleration**=..]
[**Deceleration**=..] [**Jerk**=..] [**Id**=..] [**WaitSyn**] { \ } **]**

Blockdiagramm des Funktionsbausteins in PLCopen:

Eingangspin	Einheit
Distance	[0,1µm or 10^{-4} °]
Velocity	[1µm/s or 10^{-3} °/s]
Acceleration	[1mm/s² or 1°/s²]
Deceleration	[1mm/s² or 1°/s²]
Jerk	[1mm/s³ or 1°/s³]



Programmierbeispiel

Befehl MC_MoveAdditive

```
S[MC_MoveAdditive Distance=277 Velocity=1100 Acceleration=550 \
Deceleration=660 Jerk=22000]
```

15.8.4 Befehl MC_MoveRelative

Mit MC_MoveRelative wird eine relative Bewegung zuzüglich zur aktuellen Position beauftragt. Unabhängig davon, ob sich die Achse im Zustand 'Discrete Motion' oder 'Continuous Motion' befindet. Die Bewegung wird immer ruckbegrenzt mit dem in "Jerk" eingestellten konstanten Ruck ausgeführt. Der Wert gilt sowohl für das Beschleunigen mit "Acceleration", als auch für das Bremsen mit "Deceleration".

Werden die optionalen Parameter "Acceleration", "Deceleration" und "Jerk" nicht angegeben oder ≤ 0 gesetzt, so werden die Dynamikwerte aus der entsprechenden Achsliste übernommen.

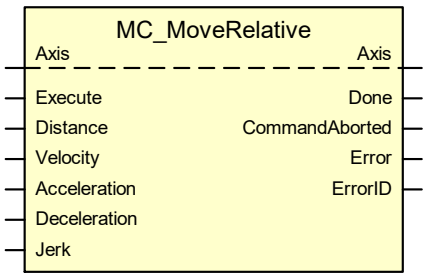
Optional kann auf den Abschluss des Auftrags gewartet (Schlüsselwort "WaitSyn") oder für eine spätere Synchronisierung eine Job-ID (Schlüsselwort "Id") vergeben werden.

Syntax NC-Befehl:

<*Achsname*>**[MC_MoveRelative Distance**=.. **Velocity**=.. [**Acceleration**=..]
[**Deceleration**=..] [**Jerk**=..] [**Id**=..] [**WaitSyn**] { \ } **]**

Blockdiagramm des Funktionsbausteins in PLCopen:

Eingangspin	Einheit
Distance	[0,1µm or 10^{-4} °]
Velocity	[1µm/s or 10^{-3} °/s]
Acceleration	[1mm/s² or 1°/s²]
Deceleration	[1mm/s² or 1°/s²]
Jerk	[1mm/s³ or 1°/s³]



Programmierbeispiel

Befehl MC_MoveRelative

```
S[MC_MoveRelative Distance=321 Velocity=1200 Acceleration=555 \
Deceleration=666 Jerk=22000]
```

15.8.5 Befehl MC_MoveSuperImposed

Mit MC_MoveSuperImposed wird eine relative Bewegung beauftragt, zusätzlich zu einer bereits aktiven Bewegung. Die aktive Bewegung wird nicht unterbrochen, sondern mit der beauftragten überlagert. Die Bewegung wird immer ruckbegrenzt mit dem in "Jerk" eingestellten konstanten Ruck ausgeführt. Der Wert gilt sowohl für das Beschleunigen mit "Acceleration", als auch für das Bremsen mit "Deceleration".

Werden die optionalen Parameter "Acceleration", "Deceleration" und "Jerk" nicht angegeben oder ≤ 0 gesetzt, so werden die Dynamikwerte aus der entsprechenden Achsliste übernommen.

Da sich bei der überlagerten Interpolation auch die Beschleunigungen überlagern, ist durch entsprechende Achsparametrierung sicherzustellen, dass die Achse nicht dynamisch überfordert wird.

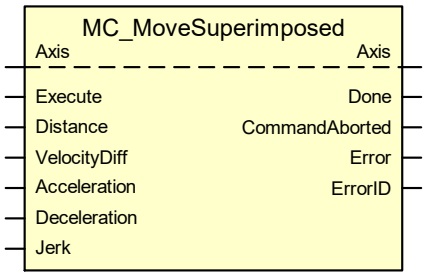
Optional kann auf den Abschluss des Auftrags gewartet (Schlüsselwort "WaitSyn") oder für eine spätere Synchronisierung eine Job-ID (Schlüsselwort "Id") vergeben werden.

Syntax NC-Befehl:

<*Achsname*>**[MC_MoveSuperImposed Distance**=.. **VelocityDiff**=..
[Acceleration=..] **[Deceleration**=..] **[Jerk**=..] **[Id**=..] **[WaitSyn]** { \ } **]**

Blockdiagramm des Funktionsbausteins in PLCopen:

Eingangspin	Einheit
Distance	[0,1µm or 10^{-4} °]
VelocityDiff	[1µm/s or 10^{-3} °/s]
Acceleration	[1mm/s² or 1°/s²]
Deceleration	[1mm/s² or 1°/s²]
Jerk	[1mm/s³ or 1°/s³]



Programmierbeispiel

Befehl MC_MoveSuperImposed

```
S[MC_MoveSuperImposed Distance=321 VelocityDiff=783 Acceleration=811 \
Deceleration=922 Jerk=45000]
```

15.8.6 Befehl MC_MoveVelocity

Mit MC_MoveVelocity beauftragt man eine endlose Bewegung mit der angegebenen Geschwindigkeit. Die Bewegung wird immer ruckbegrenzt mit dem in "Jerk" eingestellten konstanten Ruck ausgeführt. Der Wert gilt sowohl für das Beschleunigen mit "Acceleration", als auch für das Bremsen mit "Deceleration".

Werden die optionalen Parameter "Acceleration", "Deceleration" und "Jerk" nicht angegeben oder ≤ 0 gesetzt, so werden die Dynamikwerte aus der entsprechenden Achsliste übernommen.

Um die Bewegung zu stoppen muss der Befehl durch einen anderen Befehl unterbrochen werden, der eine neue Beauftragung an die Achse absetzt.

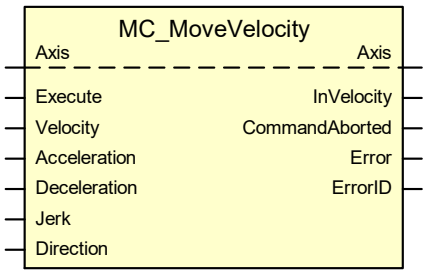
In Verbindung mit einem MC_MoveSuperImposed bleibt der Ausgang "InVelocity" TRUE.

Optional kann auf den Abschluss des Auftrags gewartet (Schlüsselwort "WaitSyn") oder für eine spätere Synchronisierung eine Job-ID (Schlüsselwort "Id") vergeben werden.

```
Syntax NC-Befehl:
```
*<Achsname>***[MC_MoveVelocity Velocity**=.. [**Acceleration**=..] [**Deceleration**=..]
[**Jerk**=..] **Direction**=.. [**Id**=..] [**WaitSyn**] { \ } **]**

Blockdiagramm des Funktionsbausteins in PLCopen:

Eingangspin	Einheit
Velocity	[1µm/s or 10^{-3} °/s]
Acceleration	[1mm/s² or 1°/s²]
Deceleration	[1mm/s² or 1°/s²]
Jerk	[1mm/s³ or 1°/s³]
Direction	1 Positive Richtung 3 Negative Richtung 4 Derzeitige Richtung



Programmierbeispiel

Befehl MC_MoveVelocity

```
S[MC_MoveVelocity Velocity=1333 Acceleration=770 Deceleration=880 \
Jerk=10000 Direction=1]
```

15.8.7 Befehl MC_Stop

Der MC_Stop führt zu einem gesteuerten Bewegungshalt und führt die Achse in den Zustand 'Stopping'. Der Bewegungshalt wird immer ruckbegrenzt mit dem in "Jerk" eingestellten konstanten Ruck für den Aufbau der Bremsverzögerung ausgeführt.

Werden die optionalen Parameter "Acceleration", "Deceleration" und "Jerk" nicht angegeben oder ≤ 0 gesetzt, so werden die Dynamikwerte aus der entsprechenden Achsliste übernommen.

Der Befehl bricht jede laufende Beauftragung durch andere (SAI-)Bewegungsbefehle ab.

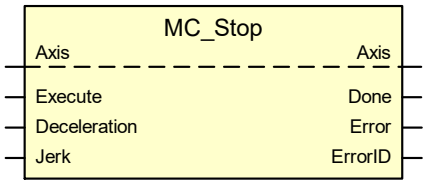
Optional kann auf den Abschluss des Auftrags gewartet (Schlüsselwort "WaitSyn") oder für eine spätere Synchronisierung eine Job-ID (Schlüsselwort "Id") vergeben werden.

```
Syntax NC-Befehl:
```

<*Achsname*>**[MC_Stop** [**Deceleration**=..] [**Jerk**=..] [**Id**=..] [**WaitSyn**] { \ } **]**

Blockdiagramm des Funktionsbausteins in PLCopen:

Eingangspin	Einheit
Deceleration	[1mm/s^2 or 1°/s^2]
Jerk	[1mm/s^3 or 1°/s^3]



Programmierbeispiel

Befehl MC_Stop

```
S[MC_Stop Deceleration=999 Jerk=25000]
```

15.8.8 Befehl MC_GearIn

Der MC_GearIn kommandiert eine Getriebekopplung mit einer Getriebeübersetzung. Diese definiert das Geschwindigkeitsverhältnis zwischen Master- und Slaveachse. Die Synchronisierung auf Geschwindigkeit erfolgt ruckbegrenzt. Der Ruckwert ist im Befehl anzugeben.

Werden die optionalen Parameter "Acceleration", "Deceleration" und "Jerk" nicht angegeben oder ≤ 0 gesetzt, so werden die Dynamikwerte aus der entsprechenden Achsliste übernommen.

Die Slaveachse kann entweder auf Mastersollwerte oder auf Masteristwerte gekoppelt werden. Die Auswahl wird im "Mode" getroffen.

Optional kann auf den Abschluss des Auftrags gewartet (Schlüsselwort "WaitSyn") oder für eine spätere Synchronisierung eine Job-ID (Schlüsselwort "Id") vergeben werden.

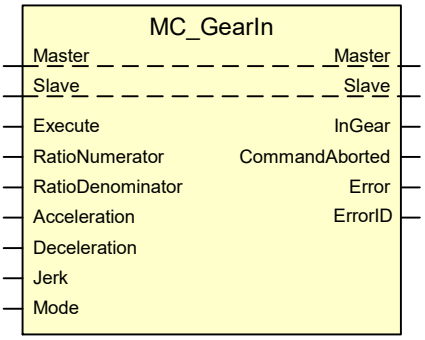
Mit dem Parameter "PhaseShift" kann die gewünschte Phasenlage von Master- und Slaveachse angegeben werden, die sich beim automatischen Phasenausgleich (Mode = 256) einstellt. Der Wert wird in metrischen Einheiten [0,1µm bzw.10^-4°] programmiert. Per Default ist dieser Parameter mit dem Wert 0 belegt.

```
Syntax NC-Befehl:
```

<*Achsname*>**[MC_GearIn Master**=.. **RatioNumerator**=.. **RatioDenominator**=..
[**Acceleration**=..] [**Deceleration**=..] [**Jerk**=..] **Mode**=.. [**Id**=..]
[**WaitSyn**] [**PhaseShift**=..] { \ } **]**

Blockdiagramm des Funktionsbausteins in PLCopen:

Eingangspin	Einheit	
Master *	Logische Achsnummer der Masterachse	
RatioNumerator *	Zähler des Verhältnisses der Getriebeübersetzung	
RatioDenominator *	Nenner des Verhältnisses der Getriebeübersetzung	
Acceleration	[1mm/s^2 or 1°/s^2]	
Deceleration	[1mm/s^2 or 1°/s^2]	
Jerk	[1mm/s^3 or 1°/s^3]	
Mode	0	Kopplungsart zwischen Master- und Slaveachse: Sollwertseitige Kopplung.
	128	Kopplungsart zwischen Master- und Slaveachse: Istwertseitige Kopplung.
	256	Automatischer Phasenausgleich: ein.



*Ergänzend zu PLCopen sind für diese Eingangspins folgende Optionen verfügbar:

Master	Alternativ zur logischen Achsnummer kann auch der Achsname der Masterspindel programmiert werden.
RatioNumerator	Alternative Kurzschreibweise RN
RatioDenominator	Alternative Kurzschreibweise RD

Programmierbeispiel

Befehl MC_GearIn

```
S[MC_GearIn Master=11 RatioNumerator=2 RatioDenominator=3 \
Acceleration=500 Deceleration=600 Jerk=20000 Mode=0]

Beauftragung mit Masterachsnamen, Defaultdynamikwerten und Kurzschreib-
weise der Getriebeübersetzung:
S[MC_GearIn Master=S2 RN=1 RD=3 PhaseShift=25 Mode=256 WaitSyn]
```

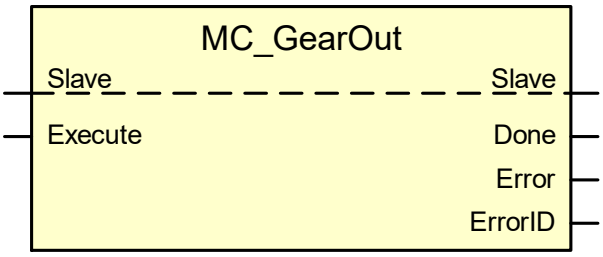
15.8.9 Befehl MC_GearOut

Der MC_GearOut löst die Koppelung der Slaveachse an die Masterachse, welche durch die Vorgabe eines Geschwindigkeitsverhältnisses hergestellt wurde. Die aktuelle Geschwindigkeit des Slave wird beibehalten (Endlosbewegung).

Optional kann auf den Abschluss des Auftrags gewartet (Schlüsselwort "WaitSyn") oder für eine spätere Synchronisierung eine Job-ID (Schlüsselwort "Id") vergeben werden.

```
Syntax NC-Befehl:
```
<*Achsname*>**[MC_GearOut** [**Id**=..] [**WaitSyn**] { \ } **]**

Blockdiagramm des Funktionsbausteins in PLCopen:



Programmierbeispiel

Befehl MC_GearOut

```
S[MC_GearOut]
```

15.8.10 Befehl MC_Phasing

Ein MC_Phasing wird dazu benutzt, um eine Verschiebung der Slaveachse bezüglich der Masterachse zu erreichen. Dazu wird aus Sicht der Slaveachse eine Phasenverschiebung der Masterachse vorgegeben und die Slaveachse versucht durch Beschleunigung oder Verzögerung diese Verschiebung zu beseitigen. Die Bewegung wird immer ruckbegrenzt mit dem in "Jerk" eingestellten konstanten Ruck ausgeführt. Der Wert gilt sowohl für das Beschleunigen mit "Acceleration", als auch für das Bremsen mit "Deceleration".

Werden die optionalen Parameter "Acceleration", "Deceleration" und "Jerk" nicht angegeben oder ≤ 0 gesetzt, so werden die Dynamikwerte aus der entsprechenden Achsliste übernommen.

Das mechanische Analogon ist die Lösung der Kopplung von Masterachse und Slaveachse für einen begrenzten Zeitraum.

Beim **Camming** bewirkt dieser Befehl eine Veränderung der 'scheinbaren' Masterposition aus Sicht des Slaves. Beim **Gearing** wird eine Phasenverschiebung zwischen Master und Slave durch Beauftragung einer überlagerten Bewegung im Slave veranlasst. Der MC_Phasing wirkt beim Gearing somit also wie ein MC_MoveSuperImposed (in den er steuerungsintern tatsächlich umgewandelt wird).

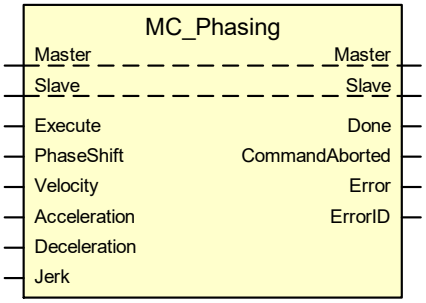
Die Dynamikwerte: "Velocity", "Acceleration" und "Deceleration" beziehen sich beim Camming auf das Verändern der 'scheinbaren' Masterposition aus Sicht des Slaves, während sie sich beim Gearing auf die überlagerte Bewegung der Slaveachse selbst beziehen.

Optional kann auf den Abschluss des Auftrags gewartet (Schlüsselwort "WaitSyn") oder für eine spätere Synchronisierung eine Job-ID (Schlüsselwort "Id") vergeben werden.

```
Syntax NC-Befehl:
```
<*Achsname*>**[MC_Phasing Master**=.. **PhaseShift**=.. **Velocity**=..
[Acceleration=..] [**Deceleration**=..] [**Jerk**=..] [**Id**=..] [**WaitSyn**] { \ } **]**

Blockdiagramm des Funktionsbausteins in PLCopen:

Eingangspin	Einheit
Master	Logische Achsnummer der Masterachse
PhaseShift	[0,1µm or 10^{-4} °]
Velocity	[1µm/s or 10^{-3} °/s]
Acceleration	[1mm/s² or 1°/s²]
Deceleration	[1mm/s² or 1°/s²]
Jerk	[1mm/s³ or 1°/s³]



Programmierbeispiel

Befehl MC_Phasing

```
S[MC_Phasing Master=11 PhaseShift=25 Velocity=1000 Acceleration=500 \
Deceleration=600 Jerk=20000]
```

15.8.11 Befehl MC_TouchProbe

Der MC_TouchProbe zeichnet eine Achsposition auf, wenn ein Triggerereignis auftritt. Der Messkanal des Antriebes und das Messverfahren (steigende, fallende Flanke des Triggersignals) wird über die Referenz für die Trigger-Signalquelle festgelegt.

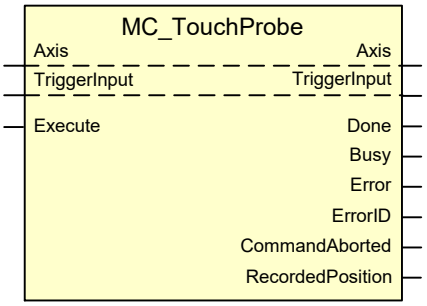
Optional kann auf den Abschluss des Auftrags gewartet (Schlüsselwort "WaitSyn") oder für eine spätere Synchronisierung eine Job-ID (Schlüsselwort "Id") vergeben werden.

Syntax NC-Befehl:

<*Achsname*>**[MC_TouchProbe Channel**=.. [**Mode**=..] [**Id**=..] [**WaitSyn**] { \ } **]**

Blockdiagramm des Funktionsbausteins in PLCopen:

Eingangspin		Einheit	
TriggerInput	Channel	----	
	Mode	---	Ohne Modeangabe Messsignalauswertung gemäß P-AXIS-00518
		1	Messsignal bei fallender Flanke
		2	Messsignal bei steigender Flanke



Programmierbeispiel

MC_TouchProbe

```
S[MC_TouchProbe Channel=1 Mode=2]
```

16 Makroprogrammierung (#INIT MACRO TAB)

Makros ermöglichen die Zuordnung eines Aliasnamens (Makronamen) zu einem ausführbaren NC-Code (Makroinhalt). Der Makroinhalt kann aus mathematischen Ausdrücken und NC-Befehlen bestehen. Bei Angabe des Makronamens (Makroaufruf) wird der entsprechende NC-Code ausgeführt.

Makros unterstützen die Erstellung wartungsarmer und übersichtlicher NC-Programme, da Änderungen nur einmal im Makroinhalt vorgenommen werden müssen.

Makros können in der Kanalparameterliste [1] [▶ 894]-1 konfiguriert und zusätzlich auch im NC-Programm definiert werden.

`Syntax einer Makrodefinition:`

"<*Makroname*>" = "<*Makroinhalt*>"

<*Makroname*> Name des Makros (Alias)

<*Makroinhalt*> Ausführbarer NC-Code

- Makroname und Makroinhalt müssen in Anführungszeichen "..." eingeschlossen sein.
- Beim Makroname wird zwischen Groß- und Kleinschreibung unterschieden.
- Das Überschreiben bereits angelegter Makros ist konfigurierbar [6] [▶ 894]-6.39.
- Makrodefinitionen sind nach Programmende (M30) oder CNC Reset weiterhin gültig. Sie bleiben bis zum nächsten Steuerungshochlauf oder #INIT MACRO TAB erhalten.
- Makrodefinitionen, die in der Kanalparameterliste konfiguriert wurden, werden durch #INIT MACRO TAB nicht gelöscht.
- Die maximale Anzahl der Makrodefinitionen [6] [▶ 894]-6.25 sowie deren Länge [6] [▶ 894]-6.37/-6.38 sind fest vorgegeben. Ab den Versionen V3.1.3079.17 bzw. V3.1.3107.10 sind die Werte einstellbar:
  - Maximale Anzahl Makros P-CHAN-00509
  - Maximale Anzahl vordefinierter Makros P-CHAN-00510
  - Maximale Zeichenanzahl Makroname P-CHAN-00511
  - Maximale Zeichenanzahl Makroinhalt P-CHAN-00512
- Makrodefinitionen können mit anderen NC-Befehlen im gleichen Satz stehen.

Der Aufruf des Makros erfolgt über den Makronamen. Er kann auch mit anderen NC-Befehlen im gleichen Satz stehen.

`Syntax eines Makroaufrufs:`

"<*Makroname*>"

Programmierbeispiel

Makrodefinitionen und Verwendung

```
N10 "POSITION_1" = "X200 Y200 Z300"    ;Makrodefinition
N20 "POSITION_2" = "X300 Y100 Z50"     ;Makrodefinition
:
N200 "POSITION_1"    (Makroaufruf, ausgeführt wird X200 Y200 Z300)
:
N500 "POSITION_2"    (Makroaufruf, ausgeführt wird X300 Y100 Z50)
```

Versionshinweis

Ab Version **V2.11.2010.02** ersetzt der Befehl **#INIT MACRO TAB** den Befehl **#INIT MAKRO TAB**. Dieser ist aus Kompatibilitätsgründen weiterhin verfügbar, es wird aber empfohlen, diesen in neuen NC-Programmen nicht mehr zu verwenden.

Syntax zur Initialisierung der Makrotabelle:

#INIT MACRO TAB

Durch diesen Befehl werden alle zuvor in NC-Programmen definierten Makros in der Tabelle gelöscht. Die über die Kanalparameter [1] [▷ 894]-1 vorbelegten Makros bleiben erhalten.

16.1 Schachtelung von Makros

Die Verwendung von Makros auf der rechten Seite der Zuweisung (s.g. Schachtelung) in Kombination mit NC-Code ist erlaubt. Die maximale Schachtelungstiefe ist fest vorgegeben [6] [▶ 894]-6.40.

Die Schachtelung wird durch ein vorangestelltes '\'-Zeichen vor den begrenzenden Anführungszeichen angezeigt. Es ist darauf zu achten, dass immer komplette Ausdrücke eines NC-Satzes (NC-Befehl, mathematischer Ausdruck, Term) in einem Makro zusammengefasst sind. Damit ist ausgeschlossen, dass ein Makro nur den Adressbuchstaben eines NC-Befehls ohne den dazugehörigen mathematischen Ausdruck repräsentiert. Auf diesen Zusammenhang wird in den folgenden Kapiteln eingegangen.

Syntax Schachtelung von Makros:

"<*Makroname*>" = "<*NC_Code*> \"< *Makroname_i*> \" <*NC_Code*>"

Achtung

Ein Makro darf nicht den eigenen Makronamen als geschachtelten Aufruf enthalten. Zulässig ist nur der geschachtelte Aufruf anderer Makros.

Programmierbeispiel

Schachtelung von Makros

```
;Beispiel 1:
N10 "POS_1" = "X500 Y200"              (Makrodefinition)
N20 "MOVE1" = "G01 \"POS_1\" F1000"    (Makrodefinition mit Schachtelung)
N30 "MOVE1"                            (Makroaufruf)
M30


;Beispiel 2:
N10 " STRING_1 " = " 5*12 "            (Makrodefinitionen)
N20 " STRING_2 " = " G \"STRING_1\" + 5 "
N30 " STRING_3 " = " M \" STRING_1\ " \" STRING_2 \" "
:
N200 " STRING_3 "                      (Aufruf des geschachtelten Makros)
:                                      (entspricht: N200 M60 G65        )
```

16.2 Verwendung in mathematischen Ausdrücken

Makronamen können arithmetischen Ausdrücken und Teilen davon zugewiesen werden. Eine rekursive Behandlung (Schachtelung) ist auch innerhalb von mathematischen Ausdrücken möglich.

Es ist darauf zu achten, dass immer geschlossene Ausdrücke in einem Makroinhalt zusammengefasst sind, d.h. Terme, deren Ergebnis nicht durch das Einfügen von ´[´ am Anfang und ´]´ am Ende beeinflusst wird.

Programmierbeispiel

Verwendung in mathematischen Ausdrücken

```
Richtig:
N10 "STRING1" = "0.5"
N20 "STRING2" = "5 * 12"
N30 "STRING3" = "SIN[89.5 + \"STRING1\"]"
N40 X[-2 * "STRING1" + "STRING2" + "STRING3"]          (Fahren nach X60)
M30

Falsch: Die Makros enthalten nur unvollständige mathematische Ausdrücke
N10 "STRING1" = "COS["
N20 "STRING2" = "90]"
N30 "STRING3" = " \"STRING1\" \"STRING2\" "           Fehler
```

16.3 Trennung von Adressbuchstabe und mathematischem Ausdruck

Es besteht die Möglichkeit, dass dem Makronamen nur der Adressbuchstabe zugewiesen und der mathematische Ausdruck ausprogrammiert oder in einem zweiten Makro definiert wird.

Dadurch können Makros, die die Hauptachsen referenzieren, in einem übergeordneten NC-Programm definiert werden. Im Unterprogramm oder Zyklus können dann die Makronamen verwendet werden, wodurch eine Unabhängigkeit von der angewählten Bearbeitungsebene resultiert.

Programmierbeispiel

Trennung von Adressbuchstabe und mathematischem Ausdruck

```
Im Makroinhalt ist nur der Adressbuchstabe enthalten. Der mathematische
Ausdruck wird ausprogrammiert oder in einem separaten Makro definiert:
"1.HA" = "X"    "2.HA" = "Y"          (HA: Hauptachse)
"Ziel_1.HA" = "V.E.POS1 + P12"
"1.HA" "Ziel_1.HA" "2.HA" 100
```

16.4 Einschränkungen

Im Makroinhalt darf kein Zeilenende- und Stringendezeichen (´\0´) eingelesen weden. Die Makrodefinition darf sich also nicht über mehrere Zeilen hinweg erstrecken.

```
"Macro_Move" = "X100 G01 \0"
"Macro_Move2" = "X100
                 G01"
...
M30
```

Im Makroinhalt dürfen keine Steuersatzanweisungen ($) enthalten sein.

```
"IF"     = "$IF"
"END_IF" = "$ENDIF"

P1 = 0

"IF" P1 == 0
P2 = 2
"ENDIF"
...
M30
```

Im Makroinhalt dürfen keine Stringkonstanten enthalten sein. Stringfunktionen oder V.E-Variablen vom Typ String sind jedoch erlaubt.

Rekursive Aufrufe führen bei Ausführung des Makros zu einem Fehler.

```
"Macro_Recursive" = "G01 X100 \"Macro_Recursive\""
...
M30
```

17 5-Achs-Funktionalität

17.1 Rotation Tool Center Point (RTCP) (#TRAFO ON/OFF)

Hinweis

Die Nutzung dieser Funktionalität erfordert die Lizenzierung des Erweiterungspaketes "Transformationen". Sie ist nicht im Umfang der Standardlizenz enthalten.

Syntax:

#TRAFO ON	Anwahl
#TRAFO OFF	Abwahl

Die Funktion RTCP stellt eine Werkzeugkorrektur im Raum dar.

Nach Anwahl von RTCP bleibt der Eingriffspunkt des aktuellen Werkzeugs bei Änderung der Werkzeugorientierung ortsfest bezüglich des Werkstücks (es ist die Belegung der Werkzeugkopfversatzparameter P-CHAN-00094 und P-TOOL-00009 zur Parametrierung der kinematischen Transformation zu beachten).

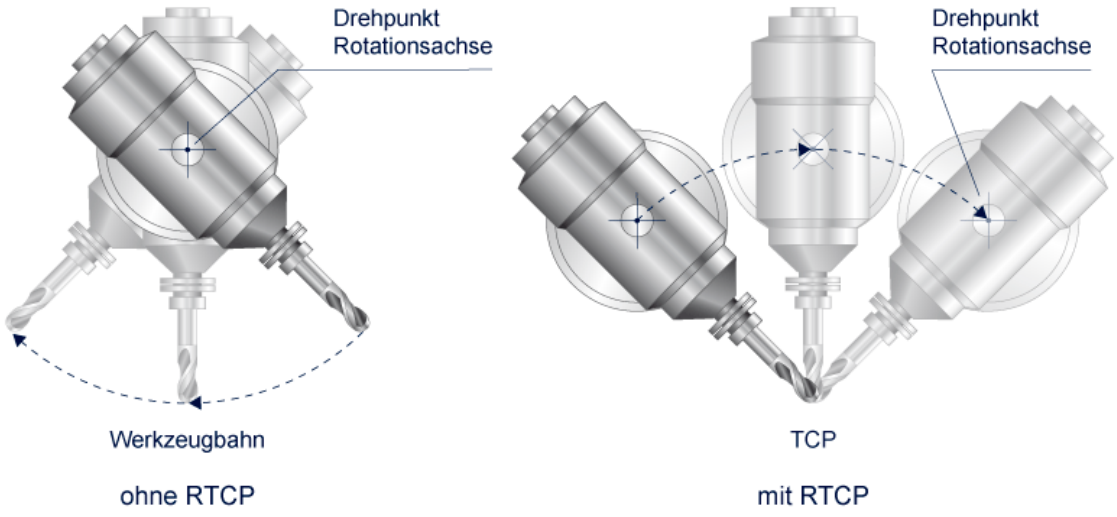


Abb. 176: Bewegungsführung ohne/mit RTCP

Wird die rotatorische Maschinenachse bewegt, so ergibt sich die links dargestellte Bewegung.

RTCP verschiebt den Drehmittelpunkt in die Fräserspitze (center point of tool rotation), indem die durch die Werkzeugbewegung resultierenden Versätze in den Achsen X, Y, Z taktweise durch entsprechend entgegengesetzte Bewegungen kompensiert werden.

Nur die Achsen X, Y, Z sind Output der kinematischen Transformation, die rotatorischen Maschinenachsen werden wie gewöhnlich programmiert.

Die Funktion RTCP darf bei aktivem TLC (siehe Kapitel Tool Length Compensation (#TLC ON/OFF) [▶ 740]) nicht angewählt werden.

Achtung

Bei aktiver kinematischer Transformation werden achsspezifische Werkzeugversätze aus *ax_versatz[<ax_index>]* (P-TOOL-00006) nur in den Achsen berücksichtigt, die nicht von der Transformationsfunktion beeinflusst werden. Abhängig vom Transformationstyp sind dies z.B. bei RTCP typischerweise alle Achsen mit Index > 2.

Die achsspezifischen Werkzeugversätze der ersten 3 Achsen (Index 0, 1, 2) werden bei aktiver Trafo **nicht** berücksichtigt. Sollen für diese Achsen Werkzeugversätze auch bei aktiver Trafo wirken, sind diese in den oben genannten Kinematikversätzen des Werkzeuges einzutragen (P-TOOL-00009).

Programmierbeispiel

RTCP Beispiel 1

```
N10 T1 D1                  (Anwahl der 2,5-D-Werkzeugkorrektur)
N20 #TRAFO ON              (Anwahl von RTCP)

N30 G01 F100 B45 C30       (Programmierung der Drehachsen ändert WZ-)
                           (Orientierung. WZ-Eingriffspunkt bleibt)
                           (ortsfest)
N40...
.
.
N100 #TRAFO OFF            (Abwahl von RTCP)
N200 M30
```

Programmierbeispiel

RTCP Beispiel 2

```
N10 #KIN ID [1]                    (Anwahl der Maschinenkinematik)
N20 #TRAFO ON                      (Anwahl von RTCP)
N30 G01 G18 X20 Y0 Z25 B90 F500    (Werkstück von rechts anfahren)
N40 G91 X-8                        (Startposition anfahren)
N50 G90 G02 X-12 I-12 B-90 F2000   (Bearbeiten)
N60...
```

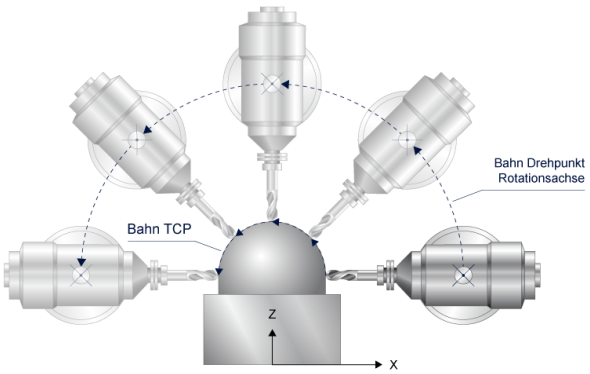


Abb. 177: Bewegungsführung mit RTCP

17.2 Transformation von PCS-Positionen (#TRAFO PCS ON/OFF)

Funktionalität verfügbar ab V3.1.3110

Hinweis

Die Nutzung dieser Funktionalität erfordert die Lizenzierung des Erweiterungspaketes "Transformationen". Sie ist nicht im Umfang der Standardlizenz enthalten.

Syntax:

#TRAFO PCS ON	Anwahl
#TRAFO PCS OFF	Abwahl

Der Befehl #TRAFO PCS ON/OFF aktiviert bzw. deaktiviert eine Transformation der Programmierkoordinaten. Die Transformationstyp ist über den konfigurierten Parameter trafo_pcs.type (P-CHAN-00829) und die angegebenen Parameter trafo_pcs.param[i] (P-CHAN-00263) bestimmt.

Neben den kartesischen [▶ 752] und kinematischen Transformationen gibt es auch die Möglichkeit diese Transformation zwischen dem Programmierkoordinatensystem und den kartesischen Transformationen einzufügen. Diese Transformationen bildet gemäß ihrer Transformationstyp die programmierten Koordinaten im NC-Programm ab und erst anschließend werden die abgebildeten Positionen mit den kartesischen und/oder der kinematischen Transformation abgebildet.

Der Unterschied zum #TRAFO ON/OFF Befehl sind nur die Ein- und Ausgangspositionen zu den Transformationen. Klassische Maschinenkinematiken, die mit #TRAFO ON/OFF [▶ 734] programmiert werden, wirken nach den kartesischen Transformationen [▶ 752] und haben physikalischen Bezug. Die hier aktivierte Transformation wirkt direkt auf die im NC-Programm programmierten Positionen und stellt eher eine mathematische Abbildung dar.

Im nachfolgenden Schaubild sind die einzelnen Ebenen der Positionen in der Transformationskette der CNC abgebildet.

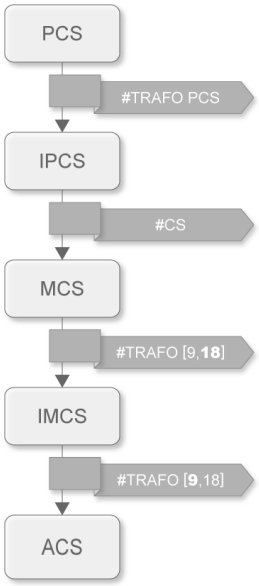


Abb. 178: PCS.Transformation im System

Programmierbeispiel

#TRAFO PCS ON/OFF

```
( Voraussetzung:
( trafo_pcs.type 212 Achskopplungstransformation)
( Masterachse ist Z, Slaveachse C, Kopplungsfaktor 0.5)

N10 #CS ON [0, 0, 20, 0, 0, 0]
N20 G01 F1000 Z100 C0  (Z-ACS-Pos=120 C-ACS-Pos=0)
N30 #TRAFO PCS ON      (Anwahl von PCS Tafo)

N40 G01 Z200           (Z-ACS-Pos=220 C-ACS-Pos=50)

N50 #TRAFO PCS OFF     (Abwahl von PCS Trafo)
N60 G01 Z100 C0        (Z-ACS-Pos=120 C-ACS-Pos=0)
N70 M30
```

17.3 Transformationsstack (#TRAFO STACK)

Der Befehl #TRAFO STACK ermöglicht die Verwendung eines festgelegten Transformationsstacks, der in den Kanalparametern vorkonfiguriert wurde. [CHAN// Parameter der Transformationsstacks]. Dadurch können unabhängig vom NC-Programm maschinenspezifische Gruppierungen von kartesischen und kinematischen Transformationen angelegt werden. Der Befehl #TRAFO STACK kombiniert also #TRAFO und mehrere #(A|B)CS Befehle in einem kompakten Befehl.

Alternativ kann das Gruppieren von Koordinatensystemen (#CS/ #ACS/ #BCS) und kinematischer Transformation auch im NC-Programm erfolgen.

Die Vergabe eines Gruppennamens ist ebenso möglich wie das gleichzeitige Aktivieren und Deaktivieren aller Gruppenelemente.

Hinweis

Transformationen sind eine lizenzpflichtige Zusatzoption.

Syntax Anwahl eines gespeicherten bzw. konfigurierten Stacks:

#TRAFO STACK ON [NAME=<*StackName*>**]**

Syntax Speichern / Überschreiben eines Stacks:

#TRAFO STACK DEF [NAME=<*StackName*> [**KINSTEP1=**<*KinId*>] [**KINSTEP2=**<*KinId*>]

{**ID=**<*Ident*> **GRP=**<*GrpId*> [**IDX=**..] } **]**

NAME=<*StackName*>	Konfigurierter oder neuer Name des Stacks
KINSTEP1=<*KinId*>	Konfigurierte oder neue Kinematik-ID der ersten Stufe
KINSTEP2=<*KinId*>	Konfigurierte oder neue Kinematik-ID der zweiten Stufe

Triplets zur Definition kartesischer Transformationsgruppen, bestehend aus:

ID=<*Ident*>	Name des Koordinatensystems. Die ID muss in P-CHAN-00490 definiert sein.
GRP=<*GrpId*>	Gruppeneinordnung des Koordinatensystems. Zulässige Angaben:
	CS: #CS-Bearbeitungskoordinatensystem
	ACS: #ACS-Koordinatensystem
	BCS: #BCS-Basiskoordinatensystem
IDX=..	Optional, Index bzw. Platz innerhalb der angegebenen Gruppe. Ohne Indexangabe wird der nächste freie Index innerhalb der Gruppe belegt.
	Es können 10 Triplets programmiert werden. Jedoch nur maximal 5 je Gruppe.

`Syntax Abwählen aller aktiven kartesischen und kinematischen Transformationen:`

#TRAFO STACK OFF

Der Befehl #TRAFO STACK ON [..] **deaktiviert** alle aktuell aktiven kartesischen und kinematischen Transformationen und aktiviert alle, diesem Stacknamen zugeteilten, kartesischen und kinematischen Transformationen.

Enthält ein Transformationsstack keine kinematische Transformation, so werden vorher aktive kinematische Transformationen abgeschaltet.

Bei Programmstart werden alle Stacks, gemäß der Parametrierung im Kanal (P-CHAN-00752 bis P-CHAN-00756) , zurückgesetzt.

Nach aktivieren eines Transformationsstack können die aktivierten Transformationen mit #TRAFO und #(A|B)CS weiterhin verändert werden.

Mit P-CHAN-00757 kann ein Transformationsstack bereits bei Programmstart automatisch aktiviert werden.

Programmierbeispiel

Definieren und verwenden eines Stacks im NC-Programm 1

Die CS-IDs müssen alle bereits in der Kanalparameterliste definiert sein.

```
%100
;Kanal test1 [0,0,0]

#TRAFO STACK DEF [NAME=STACK1 \
                  GRP=CS  ID=test1 \
                  GRP=ACS ID=test1 \
                  GRP=BCS ID=test1 ]
#TRAFO STACK ON [NAME=STACK1]
;Transformationsstack (von unten nach oben):
;[0,0,0] = CS-test1
;[0,0,0] = ACS-test1
;[0,0,0] = BCS-test1
:
```

Definieren und verwenden eines Stacks im NC-Programm 2

Sofern CS-IDs noch nicht im Programm verwendet wurden, ist es möglich, diese neu zu definieren.

```
%101
;Kanalparameter: test1 [0,0,0]

#TRAFO STACK DEF [NAME=STACK1 KINSTEP1=KIN1 \
                  GRP=CS ID=test1 \
                  GRP=ACS ID=test1 \
                  GRP=BCS ID=test1 ]
#CS DEF [test1][1,10,3] ;Neudefinition test1 gilt nur für #CS

#TRAFO STACK ON [NAME=STACK1]
;Transformationsstack (von unten nach oben):
;[1,10,3] = CS-test1
;[0, 0,0] = ACS-test1
;[0, 0,0] = BCS-test1
;KIN1
:
```

Definieren und verwenden eines Stacks im NC-Programm 3

Bis zu 5 CS können einer Gruppe zugewiesen werden, zu jeder ID muss die Gruppe aber explizit angegeben werden.

```
%102
;Kanalparameter: test1 [0,0,0]
;Kanalparameter: test2 [10,10,10]

#TRAFO STACK DEF [NAME=STACK1 \
                  GRP=CS ID=test1 \
                  GRP=CS ID=test2 ]
#TRAFO STACK ON [NAME=STACK1]
;Transformationsstack (von unten nach oben):
;[10,10,10] = CS-test2
;[ 0, 0, 0] = CS-test1
:
```

Verwenden mehrerer Stacks im NC-Programm

Bis zu 5 Stacks können im Kanal definiert sein.

```
%103
;Kanalparameter CS:
;basis [100,10,90]
;boffs [  0, 0, 0]
;aufsp [ 10,10,10]
;aoffs [  0, 0, 0]
;werks [  0, 0, 0]
;trafo_stack[0].name        acs
;trafo_stack[2].name        worko
;trafo_stack[2].kin[0]      KIN1
;trafo_stack[2].bcs.id[0]   basis
;trafo_stack[2].bcs.id[1]   boffs
;trafo_stack[2].acs.id[0]   aufsp
;trafo_stack[2].acs.id[1]   aoffs
;trafo_stack[2].cs.id[0]    werks
;trafo_stack[3].name        work
;trafo_stack[3].kin[0]      KIN1
;trafo_stack[3].bcs.id[0]   basis
```

```
;trafo_stack[3].acs.id[0]   aufsp

#TRAFO STACK ON [NAME=work]
;Transformationsstack (von unten nach oben):
;aufsp [ 10,10,10]
;basis [100,10,90]
;KIN1
:
#TRAFO STACK OFF

#BCS DEF [boffs] [0.01, -0.05, 0.1]
#ACS DEF [aoffs] [-0.03, -0.02, 0.0]

#TRAFO STACK ON [NAME=worko]
;Transformationsstack (von unten nach oben):
;aoffs [-0.03, -0.02, 0.0]
;aufsp [ 10,10,10]
;boffs [0.01, -0.05, 0.1]
;basis [100,10,90]
;KIN1
:
#TRAFO STACK OFF

#TRAFO STACK ON [NAME=acs]
:
```

17.4 Tool Length Compensation (#TLC ON/OFF)

Hinweis

Nicht im Umfang der Standardlizenz! Die Nutzung dieser Funktionalität erfordert die Lizenzierung des Erweiterungspaketes "Transformationen".

Syntax:

#TLC ON [*<delta_tool_length>* **]**	Anwahl TLC
#TLC OFF	Abwahl TLC

<delta_tool_length> Werkzeuglängendifferenz in [mm, inch].

Mit TLC lassen sich NC-Programme, die von einem Programmiersystem erzeugt wurden und eine bestimmte Werkzeuglänge berücksichtigen, auch bei geänderter Werkzeuglänge an der Maschine weiterverwenden. Es ist zu beachten, dass keine neuen Versätze oder Radien des Werkzeugs korrigiert werden.

Hinweis

Die Funktion TLC darf bei aktivem RTCP nicht angewählt werden.

Der Fehler mit ID 20669 wird ausgegeben. Die beiden Funktionen schließen sich gegenseitig aus.

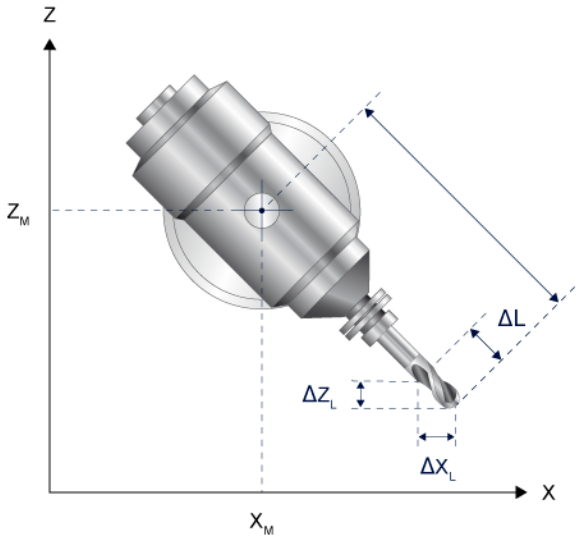


Abb. 179: Bei Änderung der Werkzeuglänge sorgt TLC taktweise für die Transformation von ΔL.

Programmierbeispiel

Tool Length Compensation

```
N10 #TLC ON [15] (Anwahl TLC mit verlängertem Werkzeug,ΔWZL =15 mm)
N20 .
.
N100 #TLC OFF         (Abwahl von TLC)
N200 #TLC ON [-20]    (Anwahl TLC mit verkürztem Werkzeug,ΔWZL = -20 mm)
N210 .
.
N300 #TLC OFF         (Abwahl von TLC)
N200 M30
```

17.5 Werkzeug ausrichten (#TOOL ORI CS)

Syntax:

#TOOL ORI CS Anwahl der Werkzeugausrichtung

Im ersten Bewegungssatz nach #TOOL ORI CS wird das Werkzeug parallel zur 3. Hauptachse des aktuellen BKS (auch W_0- oder MKS möglich) ausgerichtet. Eine Programmierung der Drehachsen in diesem Satz überschreibt die für die Ausrichtung gültigen Positionen.

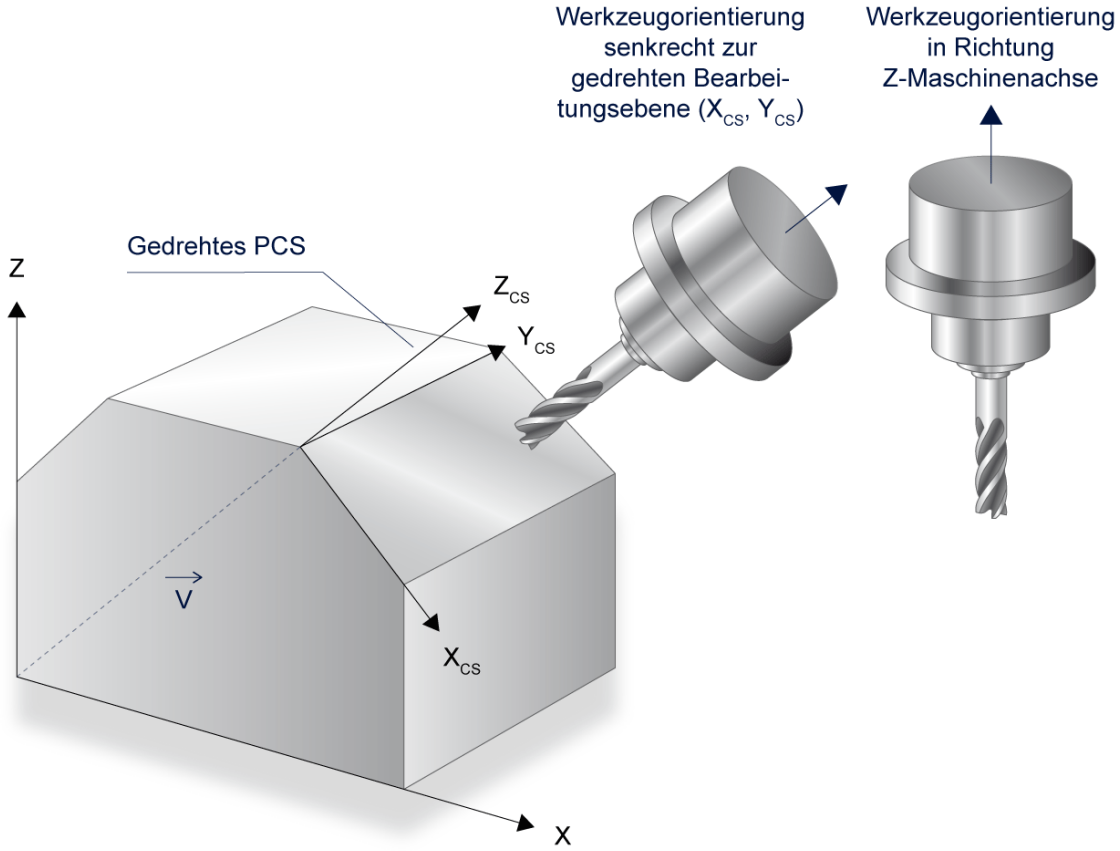


Abb. 180: Werkzeugausrichtung senkrecht zur X-Y-Bearbeitungsebene

Programmierbeispiel

Werkzeug ausrichten

```
N10 B10 C20           (WZ schräg stellen)
N20 #TOOL ORI CS      (Im nächsten Bewegungssatz WZ parallel zur Z-Achse)
                      (des aktuellen BKS, hier MKS, ausrichten)
N30 X0 Y0             (Verfahrsatz im MKS, WZ wird ausgerichtet B=0,C=0)

N40 B25 C-80          (WZ schräg stellen)
N50 #TOOL ORI CS      (WZ ausrichten im nächsten Bewegungssatz)
N60 #TRAFO ON         (RTCP-Anwahl)

N70 #CS ON[0,0,0,-80,-30,45]  (Übergang in ein gedrehtes BKS)
N80 X100              (Verfahrsatz im BKS. WZ ausrichten aus N50 bezieht)
                      (sich jedoch auf das MKS, also B=0,C=0)
N90 #TOOL ORI CS      (WZ ausrichten im gedrehten BKS)
N100 Y150             (WZ parallel zur Z-Achse des BKS ausgerichtet)

N110 #TOOL ORI CS
N120 Z100 B45 C10     (Bei Programmierung der Drehachsen ist #TOOL ORI
CS)
                      (wirkungslos)
N130 G18              (Wechsel in die X-Z-Interpolationsebene)
N140 #TOOL ORI CS     (WZ parallel zur Y-Achse ausrichten)
N150 X0               (WZ senkrecht zur X-Z-Bearbeitungsebene)
M30
```

17.6 Maschinenkinematik (#KIN ID)

Hinweis

Nicht im Umfang der Standardlizenz! Die Nutzung dieser Funktionalität erfordert die Lizenzierung des Erweiterungspaketes "Transformationen".

Syntax zur Festlegung der Maschinen-/Werkzeugkopfkinematik:

#KIN ID [**[**<*kin_id*>**]**] Festlegung Maschinen-/Werkzeugkopfkinematik

<*kin_id*> Kinematik-ID. Sie dient zur Identifizierung der in der Steuerung implementierten, maschinen- bzw. werkzeugkopfspezifischen Kinematiktypen. Ihre Default-Einstellung nach dem Steuerungshochlauf wird in P-CHAN-00032 parametriert.

Über die Programmierung von #KIN ID ohne Parameter wird die Standard-Kinematik-ID eingestellt.

Weiterhin kann über die Belegung des Elements „kin_id“ in der Werkzeugdatenliste der Kinematikwechsel automatisch mit der Werkzeuganwahl durchgeführt werden.

Eine unbekannte Kinematik-ID führt bei Anwahl von RTCP, TLC oder TOOL ORI CS zur Ausgabe einer Fehlermeldung und zum Anhalten der Dekodierung.

Bei Anwahl von Kinematik-ID 0 wird ohne Warnung oder Fehlermeldung keine Kinematik aktiviert.

Achtung

Ein Kinematikwechsel mit #KIN ID... bei aktivem RTCP oder TLC ist nicht erlaubt.

Programmierbeispiel

Maschinenkinematik

```
N10 #TOOL ORI CS      (Werkzeug ausrichten, Default-Kinematik)
                      (aus P-CHAN-00032 gültig.)
N20 T1 D1             (Werkzeuganwahl)
N30 #TRAFO ON         (RTCP-Anwahl, Default-Kinematik)
                      (aus P-CHAN-00032 gültig)
.
.
N40 #TRAFO OFF        (RTCP-Abwahl)
N50 #KIN ID[2]        (Anwahl der Kinematik, die in der)
                      (internen Bibliothek unter der ID '2' )
                      (abgelegt ist)
N60 #TRAFO ON         (RTCP-Anwahl, Kinematik 2)
.
.
N70 #TRAFO OFF        (RTCP-Abwahl)
N80 M30
```

17.7 Modifizieren von Kinematikeigenschaften (#KIN DATA)

Dieser Befehl ermöglicht es bei aktiver Transformation (#TRAFO ON [▶ 734]) Eigenschaften der aktiven Kinematik zu modifizieren. Dazu zählt z.B. die Möglichkeit bei redundanten Freiheitsgraden einer Kinematik festzulegen, welche Achsen sich nicht bewegen sollen.

Versionshinweis

Dieser Befehl ist verfügbar ab der CNC-Version V3.1.3080.

Hinweis

Es wird nur die Kinematik ID 210 unterstützt.

Wird der Befehl bei anderen Kinematiken angewendet, so bleibt er wirkungslos.

Syntax:

#KIN DATA [LOCKDOF { **AX=**<*Achsname*>} | { **AXNR**=..} **]**	Anwahl zu sperrender Achsen
#KIN DATA [UNLOCKDOF { **AX**=<*Achsname*>} | { **AXNR**=..} **]**	Anwahl freizugebender Achsen

LOCKDOF	Für die angegebenen Achsen wird der Eintrag im Parameter P-CHAN-00458 gesetzt. Im Gegensatz zur Variable V.G.KIN[i].LOCK_DOF[<AXIDX>] kann der Befehl auch bei aktiver Transformation angewendet werden.
UNLOCKDOF	Für die angegebenen Achsen wird der Eintrag im Parameter P-CHAN-00458 zurückgesetzt. Im Gegensatz zur Variable V.G.KIN[i].LOCK_DOF[<AXIDX>] kann der Befehl auch bei aktiver Transformation angewendet werden.
AX=<*Achsname*>	Namen der zu sperrenden/freizugebenden Achsen
AXNR=..	Logische Achsnummern der zu sperrenden/freizugebenden Achsen, positive Ganzzahl

Beispiel

Anwenden des #KIN DATA-Befehls

Ausgangslage für das nachfolgende Programmierbeispiel ist eine vereinfachte Konfiguration einer Koppelkinematik mit einem stehenden Roboter auf einer X-Linearachse. Bei programmiertem TCP (Achsbezeichner X) bewegt sich aufgrund der Bewegungspriorität in P-CHAN-00450 zuerst die X-Linearachse.

Auszug aus der Parametrierung der Kanalparameter:

```
gruppe[0].achse[00].log_achs_nr 1
gruppe[0].achse[00].bezeichnung X
gruppe[0].achse[00].default_feed_axis 0
...
gruppe[0].achse[06].log_achs_nr 7
gruppe[0].achse[06].bezeichnung X_LIN
gruppe[0].achse[06].default_feed_axis 0
gruppe[0].achse[07].log_achs_nr 8
gruppe[0].achse[07].bezeichnung X_ROB
gruppe[0].achse[07].default_feed_axis 0
...

trafo[0].id 210
trafo[0].group[0].name LIN_ROB
trafo[0].group[0].chain[0] LIN
trafo[0].group[0].chain[1] ROB
trafo[0].group[0].move_prio[0] LIN
trafo[0].group[0].move_prio[1] ROB

trafo[1].id 91
trafo[1].name LIN
...
trafo[2].id 45
trafo[2].name ROBOT
...
```

Im folgenden Beispiel wird im Satz N01 die X-Linearachse gesperrt. Es kann somit nur noch der Roboter die programmierte Bewegung fahren.

Im Satz N03 wird die X-Linearachse wieder entsperrt. Somit fährt sie die programmierte X-Bewegung, da sie in der Bewegungspriorität vor dem Roboter steht.

```
N01 #KIN DATA [LOCKDOF AXNR=7]
N02 G00 G90 X1500
( Kartesische Achspositionen: X=1500, X_LIN=0, X_ROB=0
N03 #KIN DATA [UNLOCKDOF AX=X_LIN]
N02 G00 G90 X1000
( Kartesische Achspositionen: X=1000, X_LIN=-500, X_ROB=1500
```

17.8 Positionierbewegungen ohne Ausgleichsbewegung (#PTP ON/ OFF, #AX LOCK ALL, #AX UNLOCK ALL)

Hinweis

Nicht im Umfang der Standardlizenz! Die Nutzung dieser Funktionalität erfordert die Lizenzierung des Erweiterungspaketes "Transformationen".

Syntax:

#PTP ON	Anwahl Trafo PTP Bewegungsführung
#PTP OFF	Abwahl Trafo PTP Bewegungsführung

Wenn nach Anwahl der kinematischen Transformation das Werkzeug positioniert und orientiert wird, führt dies zu Ausgleichsbewegungen in den Maschinenachsen, da die Werkzeugspitze (TCP) auf der Bahn geführt wird. Sind diese Ausgleichsbewegungen unerwünscht, so kann über die oben aufgeführten Befehle eine zeitoptimalere Bewegungsführung erreicht werden.

Bei 5-Achsmaschinen wird die Bewegung auf Basis des Werkzeugkopfbezugspunktes durchgeführt; - der Bezugspunkt bewegt sich mit dem programmierten Vorschub (F-Wort) oder Eilgang auf einer Geraden, der TCP nicht (siehe folgende Abb.).

Bei nicht kartesischen Maschinenstrukturen (z.B. Roboter, Tripod) bewegen sich weder der Bezugspunkt noch der TCP auf einer Geraden. Der programmierte Vorschub (F-Wort) oder Eilgang wirkt auf die Maschinenachsen. Es ist jedoch sichergestellt, dass am Ende der Bewegung der TCP auf dem programmierten PCS-Zielpunkt steht.

Die Bewegungsprogrammierung ist identisch zur PCS-Programmierung, die Umrechnung der PCS-Koordinaten in MCS-Koordinaten wird von der Steuerung ausgeführt. Im Unterschied zur Verwendung des Befehls #WCS TO MCS wird die Versatz- und Werkzeugverrechnung identisch wie bei aktiver kinematischer Transformation durchgeführt.

Nach #PTP ON aktives Konturüberschleifen (G261) [▶ 264] arbeitet implizit mit dem DIST_SOFT-Verfahren [▶ 280]. Somit ist sichergestellt, dass sich unabhängig von der aktiven Vorschubgruppe ein optimaleres Vorschubprofil ergibt. Nach #PTP OFF wird wieder das zuletzt parametrierte Überschleifverfahren aktiviert.

Achtung

Während aktiver Trafo PTP Bewegungsführung werden im Echtzeitteil der CNC in den PCS-Koordinaten ACS-Werte angezeigt!

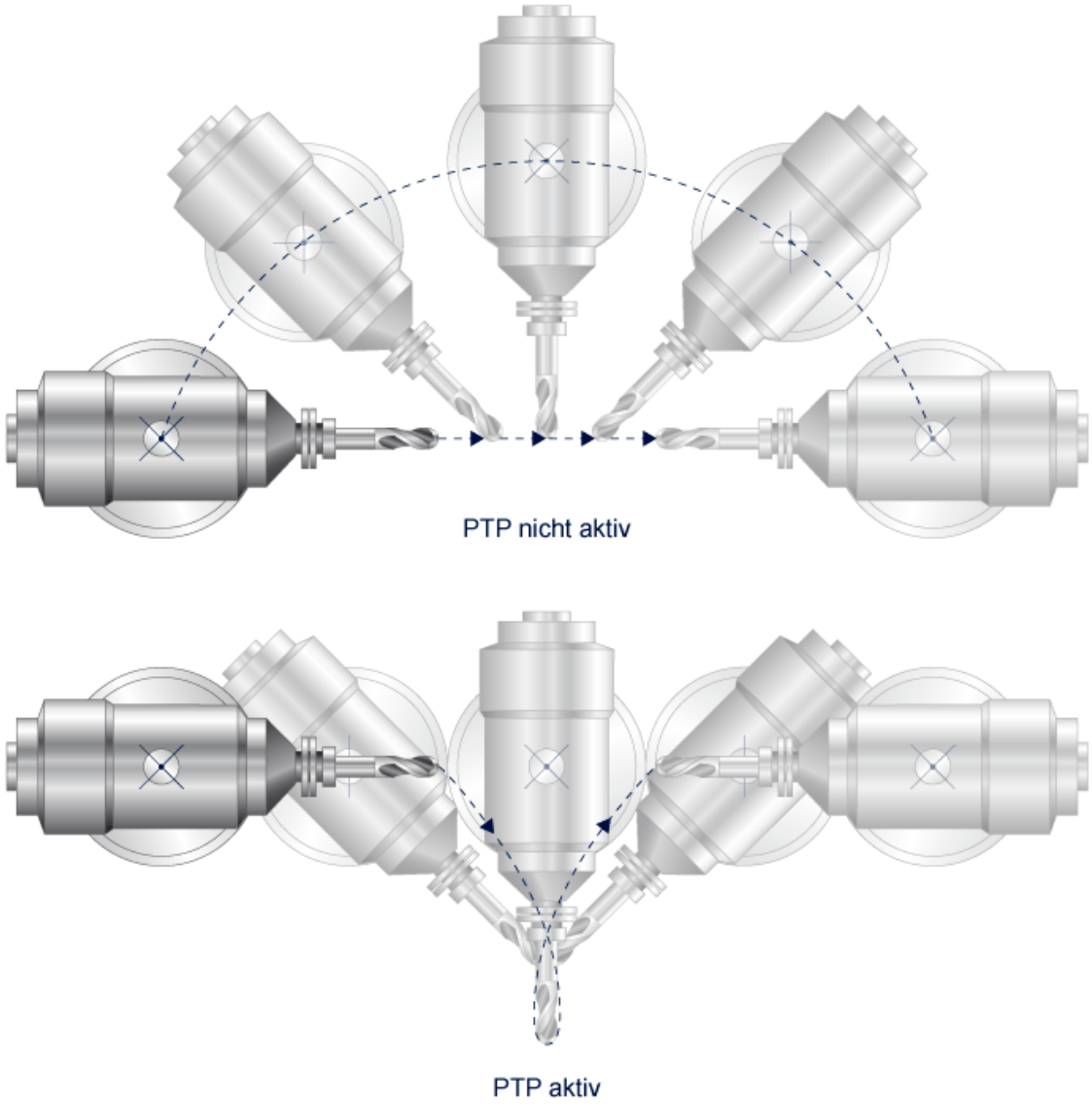


Abb. 181: Bewegungsführung ohne / mit #PTP

Achtung

Während aktiver Trafo PTP Bewegungsführung ist die An-/Abwahl zusätzlicher Koordinatentransformationen (#(A)CS ON/OFF, #MCS ON/OFF usw.) nicht erlaubt!

Programmierbeispiel

Positionierbewegungen ohne Ausgleichsbewegung (PTP)

Beispiel zur vorangegangenen Abbildung:

```
N10 T1 D1                  ;Werkzeuganwahl
N20 G00 X-200 Y0 Z0 A90    ;MCS Anfahrposition
N30 #TRAFO ON              ;RTCP-Anwahl, Default-Kinematik
                           ;aus P-CHAN-00032 gültig
N40 #PTP ON                ;Trafo PTP Bewegung ein
N50 G00 X200 Y0 Z0 A-90    ;PCS Zielposition
...
N180 #PTP OFF              ;Trafo PTP Bewegung aus
N185 G01 X100 Y150 F5000
…
N500 #TRAFO OFF            ;RTCP-Abwahl
N999 M30
```

Beispiel mit automatischer Werkzeugausrichtung:

```
N10 T1 D1                  ;Werkzeuganwahl
N20 #TRAFO ON              ;RTCP-Anwahl, Default-Kinematik)
                           ;aus P-CHAN-00032 gültig
N30 #TOOL ORI CS           ;Werkzeug ausrichten, Default-Kinematik
                           ;aus P-CHAN-00032 gültig.
N40 #PTP ON                ;Trafo PTP Bewegung ein
N50 G00 X0 Y0 Z100 G90     ;PCS Anfahrposition
...
N180 #PTP OFF              ;Trafo PTP Bewegung aus
N185 G01 X100 Y150 F5000
…
N500 #TRAFO OFF            ;RTCP-Abwahl
N999 M30
```

Hinweis

Im Zusammenhang mit der Trafo PTP Bewegungsführung kann es erforderlich sein, für bestimmte Achsen die sich ergebenden Verfahrbewegungen aus technologischen Gründen zu unterdrücken (z.B. um innerhalb von Softwareendschaltergrenzen zu bleiben).

Zu diesem Zweck können während aktiver PTP-Programmierung Bewegungen einzelner Achsen mit dem Befehl #AX LOCK/UNLOCK ALL temporär gesperrt werden.

Syntax:

#AX LOCK [{ **AX=**<*Achsname*> } | { **AXNR=..** } **]** Anwahl zu sperrender Achsen

AX=<*Achsname*> Namen der zu sperrenden Achsen

AXNR=.. Logische Nummern der zu sperrenden Achsen, Positive Ganzzahlen

Syntax:

#AX UNLOCK ALL Freigabe gesperrter Achsen. Sind mehrere Achsen gesperrt, so können diese nur gemeinsam wieder freigegeben werden. Die Freigabe nur bestimmter Achsen ist nicht möglich.

Auch wenn nur eine einzelne Achse gesperrt ist, muss diese mit ALL freigegeben werden.

Die Programmierung von #AX LOCK, #AX UNLOCK ALL ist nur während aktivem #PTP erlaubt.

Gesperrte ACS Ausgangsachsen der kinematischen Transformation bewegen sich nicht bei G00 und G01.

Programmierbeispiel

Positionierbewegungen ohne Ausgleichsbewegung (PTP)

Anwahl der zu sperrenden Achse bzw. Freigabe aller gesperrten Achsen.

```
Korrekt:
N10 #CYL[EDGES=4 ROUNDING=5 LENGTH1=40 LENGTH2=40]
N20 #PTP ON
N30 #AX LOCK[AX=Z AX=B] ;alternativ: #AX LOCK[AXNR=3 AXNR=5]
N40 G00 G90 U30
.....
N60 #AX UNLOCK ALL  ;Mit impliziter Positionsanforderung
N70 #PTP OFF
N80 G01 G90 Z0 F3000
N90 G01 U40 F2000
.....

Falsch:
N10 #AX LOCK[AX=Z] ;Programmierung vor PTP
N20 #CYL[EDGES=4 ROUNDING=5 LENGTH1=40 LENGTH2=40]
N30 #PTP ON
N40 G00 G90 U30
.....
N60 #AX UNLOCK ALL ;Mit impliziter Positionsanforderung
N70 #PTP OFF
N80 G01
.....
N200 #CYL OFF
```

17.9 Koordinatensysteme

Die Anzahl der möglichen Koordinatensysteme ist auf [SYSP// Nummer 6.17] beschränkt.

17.9.1 Standardprogrammierung

17.9.1.1 Definition eines Bearbeitungskoordinatensystems (#CS DEF, #CS ON/OFF, #CS MODE ON/OFF)

`Syntax der CS-Programmierung:`

Definition und Speicherung eines CS:

#CS DEF [**[**<*CS-ID*>**]**] **[** <*v1*>,<*v2*>,<*v3*>,<*φ1*>,<*φ2*>,<*φ3*> **]**

Definition und Speicherung mit gleichzeitiger Aktivierung:

#CS ON [**[**<*CS-ID*>**]**] **[** <*v1*>,<*v2*>,<*v3*>,<*φ1*>,<*φ2*>,<*φ3*> **]**

#CS ON [<*CS-ID*>**]** Anwahl eines gespeicherten CS

#CS ON Anwahl des zuletzt definierten CS

#CS OFF Abwahl des zuletzt aktivierten CS

Der Parameter CS-ID darf hier nicht programmiert werden, da nur die Abwahl des zuletzt aktivierten CS erlaubt ist.

<*CS-ID*> Koordinatensystem-ID. Bei Programmstart wird die CS-ID mit dem Standardwert 1 belegt. Wird bei #CS DEF bzw. #CS ON die CS-ID **nicht** programmiert, so wird dafür die nächste freie CS-ID automatisch ermittelt. CS dieser Art sind jedoch nach ihrer Abwahl mit #CS OFF nicht mehr verfügbar!

Maximal können 5 CS-Definitionen gleichzeitig gespeichert werden.

<*vi*> Komponenten des translatorischen Verschiebungsvektors in [mm, inch]. (Diese beziehen sich auf die Hauptachsen in der Reihenfolge bei G17).

<*φi*> Drehwinkel in [°].

Hinweis: Die Komponenten von Verschiebung und Drehung sind durch Kommas getrennt. Lücken in der Reihenfolge können durch aufeinanderfolgende Kommas ", ," markiert werden. Zur besseren Lesbarkeit wird jedoch empfohlen, diese Komponenten mit 0 zu programmieren.

Beispiel: [,,10,,,45] ↔ [0,0,10,0,0,45]

Eine verkürzte Programmierung der Komponenten ist ebenfalls zulässig.

Beispiel: [0,0,10] nur Verschiebung

[0,0,10,0,30] Verschiebung und 2 Drehwinkel

#CS OFF ALL Abwahl aller Bearbeitungskoordinatensysteme CS

Ein CS (Bearbeitungskoordinatensystem BKS) wird durch die relative Verschiebung (v2 in folgender Abb.) und Drehung zum aktuellen Werkstück-KS (WKS) charakterisiert. Aktuelle Nullpunktverschiebung, Platzversatz und Bezugspunktverschiebung (v1 in folgender Abb.) bestimmen die Lage des CS bezüglich des Maschinen-KS (MKS).

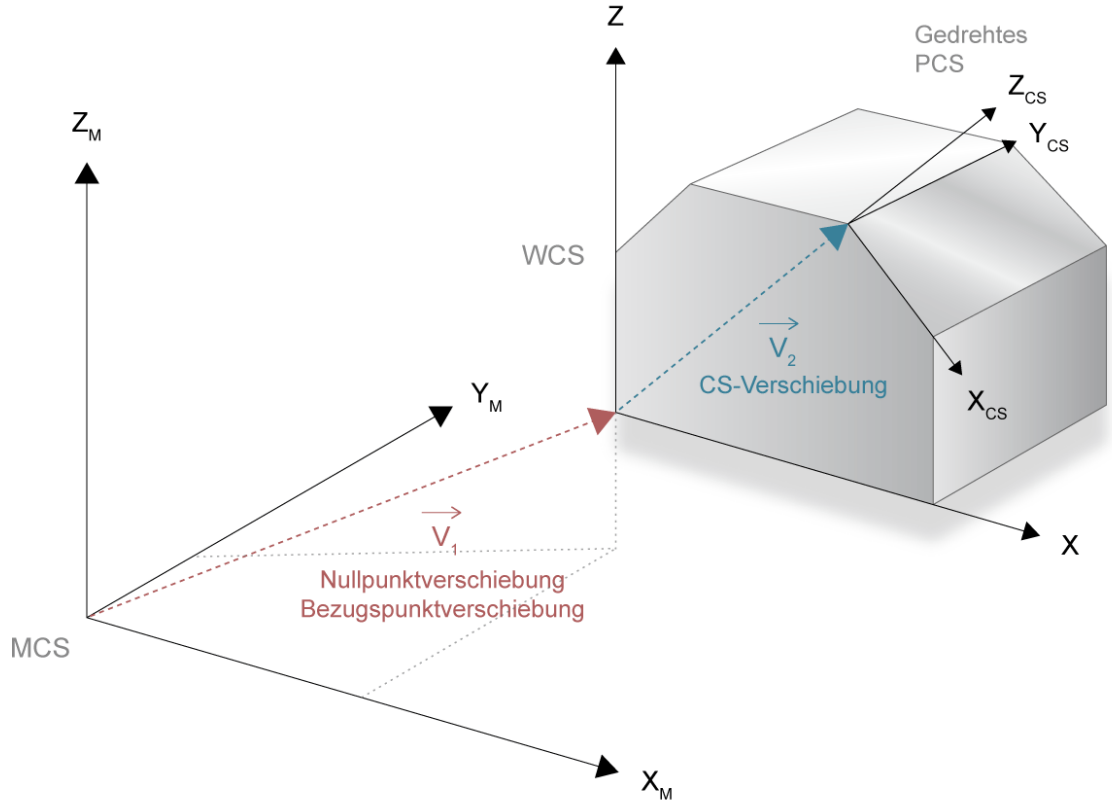


Abb. 182: Bearbeitung an einer schiefen Ebene

Grundeinstellung Drehreihenfolge und Drehmodus:

Für die Festlegung der Drehungen φ1, φ2 und φ3 gilt, dass diese in der Grundeinstellung in der nachfolgend aufgeführten Reihenfolge jeweils in mathematisch positiver Drehrichtung durchgeführt werden (folgende Abb.):

1. Drehung mit Winkel φ3 um die 3. Achse (z.B. z)
2. Drehung mit Winkel φ2 um die neue 2. Achse (z.B. y´)
3. Drehung mit Winkel φ1 um die neue 1. Achse (z.B. x´´)

Diese Drehreihenfolge wird auch als YAW(gieren) - PITCH(neigen) - ROLL(rollen) bezeichnet. Die Drehungen beziehen sich hier immer auf die neuen Achsen des aktuell gedrehten CS (Drehmodus).

(Die angegebene Reihenfolge der Achsen entspricht hierbei, unabhängig von G17/G18/G19, immer der Reihenfolge der Hauptachsen bei G17).

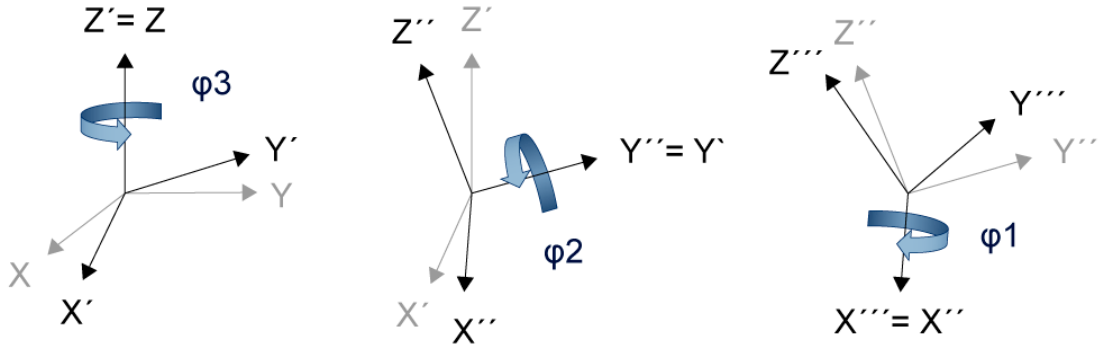


Abb. 183: Definition eines CS durch 3 Drehungen bezogen auf die neuen Achsen

Freie Angabe der Drehreihenfolge und Drehmodus:

Jede Orientierung im Raum kann durch die Verkettung von drei Basisrotationen erreicht werden. Es existieren 6 mögliche Drehreihenfolgen um zwei Achsen (klassische Eulerwinkel genannt) und 6 Drehreihenfolgen um 3 Achsen (Tait-Bryan-Winkel genannt).

Drehungen beziehen sich entweder immer auf feste Raumachsen (extrinsische Drehung) oder auf die neuen Achsen des aktuell gedrehten CS (YAW, intrinsische Drehung). Folgende Abbildung unten stellt diesen Unterschied, im Vergleich zur obigen Abbildung, dar.

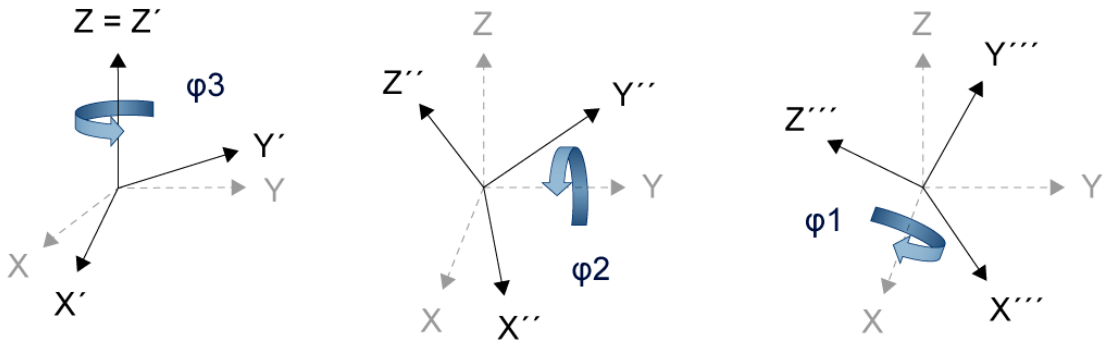


Abb. 184: Definition eines CS durch 3 Drehungen um feste Raumachsen

Die Drehreihenfolge wird durch P-CHAN-00394 konfiguriert. Im NC-Programm kann diese konfigurierte Drehreihenfolge durch folgenden Befehl geändert werden.

Syntax Drehreihenfolge:

#CS MODE ON [ROTATION_SEQUENCE=<*rot_sequence*>**]**

<*rot_sequence*>	Drehreihenfolge als String gemäß:	
	Euler-Winkel	Tait-Bryan-Winkel
	XYX, XZX, YXY, YZY, ZXZ, ZYZ	XYZ, XZY, YXZ, YZX, ZXY, **ZYX** (Default)

Syntax Abwahl und Wiederherstellen der Grundeinstellung:

#CS MODE OFF [ROTATION_SEQUENCE]

Auch der Drehmodus wird durch P-CHAN-00393 konfiguriert. Im NC-Programm kann dieser konfigurierte Drehmodus durch folgenden Befehl geändert werden.

Syntax Drehmodus:

#CS MODE ON [ROTATION_MODE_FIXED]

ROTATION_MODE_FIXED	Drehung um die (festen) Raumachsen des Koordinatensystems zu Beginn der Drehungen

Syntax Abwahl und Wiederherstellen der Grundeinstellung:

#CS MODE OFF [ROTATION_MODE_FIXED]

Ohne #CS MODE OFF [...] bleiben die Einstellungen bis Hauptprogrammende (M30) bzw. RESET aktiv. Beim nächsten Programmstart sind wieder die Grundeinstellungen wirksam.

Ein über #CS DEF [<*CS-ID*>] [...] bzw. CS ON [<*CS-ID*>] [...] definiertes CS wird bezüglich seiner Lage zum aktuellen WKS gespeichert und kann über #CS ON [<CS-ID>] ohne Angabe von Parametern wieder angewählt werden. Zwischenzeitliche Änderungen der Gesamtverschiebung im MKS ergeben jedoch eine neue Lage des CS bezüglich des MKS.

Während der Bearbeitung im CS können Nullpunkt- Bezugspunkt- und Istwertverschiebungen programmiert werden. Diese sind allerdings nur bis zur Abwahl des CS gültig und werden nicht gespeichert. Die Achsbezeichnungen bleiben im CS erhalten. Näheres hierzu siehe im Kapitel „Verschiebungen im Koordinatensystem [▶ 769]“.

Programmierbeispiel

Beispiel 1

```
N005 P1 = 2
N010 #CS DEF [1][P1,15,5,20,30,45]  (Definition und Speicherung eines CS)
                                    (unter ID 1)
                                    (Verschiebungen relativ: X2, Y15, Z5)
                                    (Drehung Z:45° Y':30° X'':20°)
N020 #CS ON[1]                      (Aktivierung des CS mit ID 1)
:
N100 #CS OFF                        (Abwahl des CS mit ID 1)
:
N200 P1=10
N210 #CS ON [P1,15,5,2,3,60]        (Definition u. Aktivierung eines CS)
                                    (unter der automatisch bestimmten ID
2)
:
N300 #CS OFF     (Abwahl des zuletzt aktivierten CS (ID 2))
                 (Danach ist das CS mit ID 2 gelöscht!)
:
N400 M30
```

Programmierbeispiel

Beispiel 2

```
N05 P1 = 2
N10 #CS DEF [3][P1,15,5,2,3,4.5]      (Definition und Speicherung)
                                      (eines CS unter ID 3)
N20 #CS DEF [2][P1,15,5,2,3,4.5]      (Definition und Speicherung)
                                      (eines CS unter ID 2)
N30 #CS DEF [5][0,1,2,0,30,30]        (Definition und Speicherung)
                                      (eines CS unter ID 5)
N30 #CS ON                            (Aktivierung des CS mit der)
                                      (zuletzt programmierten ID 5)
:
N50 #CS OFF
N60 #CS ON [3]                        (Aktivierung des CS mit ID 3)
:
N80 #CS OFF
N90 #CS DEF [3][1,1.2,1.3,0,0,33]     (Neudefinition des CS mit ID 3)
:
M30
```

Programmierbeispiel

Beispiel 3

Werden mehrere Koordinatensysteme nacheinander z.B.mit CS ON [...] (ohne *CS_ID*) angewählt, so bilden diese ein verkettetes neues Gesamt-CS. Dieses muss durch entsprechende #CS OFF dann wieder schrittweise abgewählt werden.

Die kombinierte CS-Anwahl mit und ohne CS-Ids ist zulässig, wird jedoch aus Gründen der Programmübersichtlichkeit nicht empfohlen.

Beispiel einer mehrfachen Programmierung von CS (ohne *CS_ID*):

```
N010 #CS ON [0,0,0,0,0,20]   (Definition u. Aktivierung eines CS unter)
                             (der automatisch bestimmten ID 1)
                             (Keine Verschiebungen, nur Drehung 20°um Z)
:
N050 #CS ON [0,0,0,0,0,30]   (Definition u. Aktivierung eines CS unter)
                             (der automatisch bestimmten ID 2)
                             (Keine Verschiebungen, nur Drehung 30°um Z)

->(Es ergibt sich ein Gesamt-CS mit einer Drehung 50° um Z)
:
N100 #CS OFF     (Abwahl des CS mit ID 2, danach ist das CS)
                 (mit ID 2 gelöscht!)

->(Wirksam bleibt das CS mit ID 1 mit der Drehung 20°um Z)
:
N200 #CS OFF     (Abwahl des CS mit ID 1, danach ist das CS)
                 (mit ID 1 gelöscht und alle CS wieder abgewählt!)
:
N400 M30
```

Programmierbeispiel

Ändern von Drehmode und Drehreihenfolge

```
Umschalten auf Drehungen um feste Achsen (extrinsische Drehungen)
:
#CS MODE ON [ROTAION_MODE_FIXED]     ;Drehung um die festen Achsen
#CS MODE ON [ROTATION_SEQUENCE=XYZ] ;Drehung um die Achsen X->Y->Z
#CS ON [0,0,0,90,0,90]               ;Drehung X:90° Y:0° Z:90°
:
#CS OFF
#CS MODE OFF [ROTATION_SEQUENCE]  ;Zurück auf Drehsequenz ZYX
#CS MODE OFF [ROTAION_MODE_FIXED] ;Abwahl Drehmode um feste Achsen
:
Umschalten auf Drehungen um die jeweils neuen Achsen (intrinsische Dre-
hungen)
:
#CS MODE ON [ROTATION_SEQUENCE=XZX] ;Drehung um Achsen X->Z'->X''
#CS ON [0,0,0,90,0,90]               ;Drehung X:90° Y:0° Z:90°
:
#CS OFF
#CS MODE OFF [ROTATION_SEQUENCE]     ;Zurück auf Drehsequenz ZYX
```

17.9.1.2 Definition/Aktivierung eines Koordinatensystems zur Aufspannlagenkorrektur (#ACS)

Das Koordinatensystem zur Aufspannlagenkorrektur (ACS) dient zur Kompensation einer Schieflage des Werkstücks oder der Werkstückpalette. Seine Definition sowie An- und Abwahl erfolgen analog zu der des Bearbeitungskoordinatensystems (CS).

`Syntax der ACS-Programmierung:`

Definition und Speicherung eines ACS:

#ACS DEF [**[***<ACS-ID>***]**] **[** *<v1>,<v2>,<v3>,<φ1>,<φ2>,<φ3>* **]**

Definition und Speicherung mit gleichzeitiger Aktivierung:

#ACS ON [**[***<ACS-ID>***]**] **[** *<v1>,<v2>,<v3>,<φ1>,<φ2>,<φ3>*

#ACS ON [*<ACS-ID>***]** Anwahl eines gespeicherten ACS

#ACS ON Anwahl des zuletzt definierten ACS

#ACS OFF Abwahl des zuletzt aktivierten ACS

Der Parameter ACS-ID darf hier nicht programmiert werden, da nur die Abwahl des zuletzt aktivierten ACS erlaubt ist.

<ACS-ID> Koordinatensystem-ID. Bei Programmstart wird die ACS-ID mit dem Standardwert 1 belegt. Wird bei #ACS DEF bzw. #ACS ON die ACS-ID nicht programmiert, so wird dafür die nächste freie ACS-ID automatisch ermittelt. ACS dieser Art sind jedoch nach ihrer Abwahl mit #ACS OFF nicht mehr verfügbar!

<vi> Komponenten des translatorischen Verschiebungsvektors in [mm, inch]. (Diese beziehen sich auf die Hauptachsen in der Reihenfolge bei G17).

<φi> Drehwinkel in [°].

Hinweis: Die Komponenten von Verschiebung und Drehung sind durch Kommas getrennt. Lücken in der Reihenfolge können durch aufeinanderfolgende Kommas ", ," markiert werden. Zur besseren Lesbarkeit wird jedoch empfohlen, diese Komponenten mit 0 zu programmieren.

Beispiel: [,,10,,,45] ↔ [0,0,10,0,0,45]

Eine verkürzte Programmierung der Komponenten ist ebenfalls zulässig.

Beispiel: [0,0,10] nur Verschiebung

[0,0,10,0,30] Verschiebung und 2 Drehwinkel

Das ACS ist haltend wirksam und kann unabhängig von einem CS an- bzw. abgewählt werden.

Im ACS können Nullpunkt- und Bezugspunktverschiebungen programmiert werden. Diese sind allerdings nur bis zur Abwahl des ACS gültig und werden nicht gespeichert.

#ACS OFF ALL Abwahl aller ACS

Programmierbeispiel

ACS Beispiel 1

```
N005 P1 = 2
N010 #ACS DEF [1][P1,15,5,20,30,45](Definition und Speicherung)
                                   (eines ACS unter ID 1)
                                   (Verschiebungen relativ: X2, Y15, Z5)
                                   (Drehung Z:45°Y': 30° X'':20°)
N020 #ACS ON[1]                    (Aktivierung des ACS mit ID 1)
:
N100 #ACS OFF                      (Abwahl des ACS mit ID 1)
:
N200 P1=10
N210 #ACS ON [P1,15,5,2,3,60]      (Definition u. Aktivierung eines ACS)
                                   (unter der automatisch bestimmten ID
2)
:
N300 #ACS OFF     (Abwahl des zuletzt aktivierten ACS (ID2))
                  (Danach ist das ACS mit ID 2 gelöscht!)
:
N400 M30
```

Programmierbeispiel

ACS Beispiel 2

```
N5 P1 = 2
N10 #ACS DEF [1][10,15,5,2,3,4.5]      (Definition und Speicherung)
                                       (eines ACS unter ID 1)
N20 #ACS DEF [3][0,15,5,2,3,4.5]       (Definition und Speicherung)
                                       (eines ACS unter ID 3)
N30 #ACS DEF [P1+3][2*P1,1,2,0,30,30] (Definition und Speicherung eines)
                                       (ACS unter ID 5)
N30 #ACS ON                            (Aktivierung des ACS mit der)
                                       (zuletzt programmierten ID 5)
:
N50 #ACS OFF
N60 #ACS ON [3]                        (Aktivierung des ACS mit ID 3)
:
N80 #ACS OFF
N90 #ACS DEF [3][0,1.2,1.3,0,0,3]      (Neudefinition des ACS mit ID 3)
:
M30
```

Programmierbeispiel

ACS Beispiel 3

Werden mehrere Koordinatensysteme nacheinander z.B.mit ACS ON [...] (ohne A*CS_ID*) angewählt, so bilden diese ein verkettetes neues Gesamt-ACS. Dieses muss durch entsprechende #ACS OFF dann wieder schrittweise abgewählt werden.

Die kombinierte ACS-Anwahl mit und ohne ACS-Ids ist zulässig, wird jedoch aus Gründen der Programmübersichtlichkeit nicht empfohlen.

Beispiel einer mehrfachen Programmierung von ACS (ohne A*CS_ID*):

```
N010 #ACS ON [0,0,0,0,0,20] (Definition u. Aktivierung eines ACS unter)
                            (der automatisch bestimmten ID 1)
                            (Keine Verschiebungen, nur Drehung 20°um Z)
N020 #ACS ON [0,0,0,0,0,30] (Definition u. Aktivierung eines ACS unter)
                            (der automatisch bestimmten ID 2)
                            (Keine Verschiebungen, nur Drehung 30°um Z)

->(Es ergibt sich ein Gesamt-ACS mit einer Drehung 50°um Z)
:
N100 #ACS OFF   (Abwahl des ACS mit ID 2, danach ist das ACS)
                (mit ID 2 gelöscht!)

->(Wirksam bleibt das ACS mit ID 1 mit der Drehung 20°um Z)
:
N200 #ACS OFF   (Abwahl des ACS mit ID 1, danach ist das ACS)
                (mit ID 1 gelöscht und alle ACS wieder abgewählt!)
:
N400 M30
```

17.9.1.3 Verkettung von Koordinatensystemen

Durch Kombination von ACS und CS lassen sich neue Koordinatentransformationen bilden.

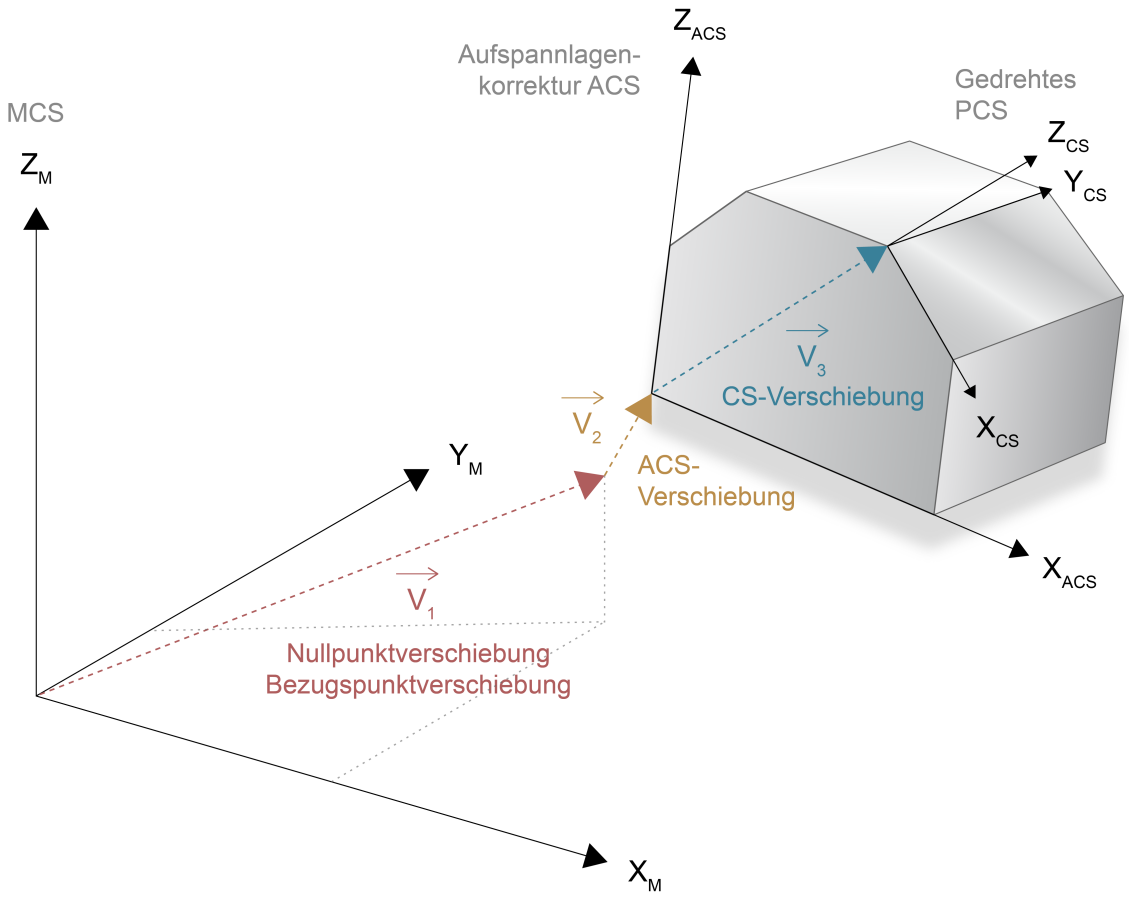


Abb. 185: Durch die Kombination von ACS und CS wird die Bearbeitung an einer schiefen Ebene bei schief liegendem Werkstück ermöglicht

Mehrere ACS und CS werden jeweils getrennt in der Reihenfolge ihrer Anwahl relativ verknüpft. Das resultierende ACS wird dann mit dem resultierenden CS zur Gesamttransformation verknüpft. Die Verknüpfung erfolgt unabhängig von der Programmierung immer mit dem ACS zuerst.

Maximal können 10 ACS/CS Kombinationen zu einer Gesamttransformation verknüpft werden.

Die Abwahl der einzelnen ACS erfolgt in umgekehrter Reihenfolge, wie die Anwahl, gleiches gilt für die CS. Um dies zu vereinfachen wird mit #(A)CS OFF kein ID-Parameter programmiert (siehe beide Abb. unter Definition eines Bearbeitungskoordinatensystems (#CS DEF, #CS ON/OFF, #CS MODE ON/OFF) [▶ 752]).

Programmierbeispiel

Verknüpfungen von Koordinatensystemen

```
N100 #CS ON [1]                (                    CS[1])
N110 #ACS ON [2]               (ACS[2] o            CS[1])
N120 #ACS ON [1]               (ACS[2] o ACS[1] o   CS[1])
N130 #CS ON [2]                (ACS[2] o ACS[1] o   CS[1] o CS[2])
N140 #ACS OFF                  (ACS[2] o            CS[1] o CS[2])
N140 #CS OFF                   (ACS[2] o            CS[1])
N150 #ACS OFF                  (                    CS[1])
N160 #CS OFF
M30
```

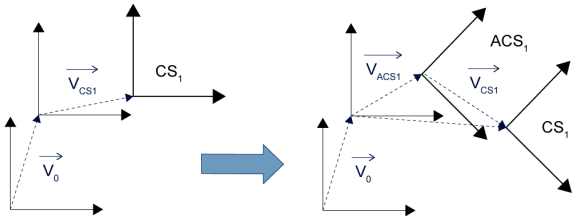


Abb. 186: Aktivierung oder Änderung der Aufspannlagenkorrektur ohne dass bereits aktive Koordinatensysteme abgewählt werden müssen

Bei der relativen Verknüpfung von ACS oder CS ist zu beachten, dass eine geänderte Reihenfolge der Anwahl im Allgemeinen unterschiedliche Ergebnisse zur Folge hat (siehe nachfolgende Abbildung).

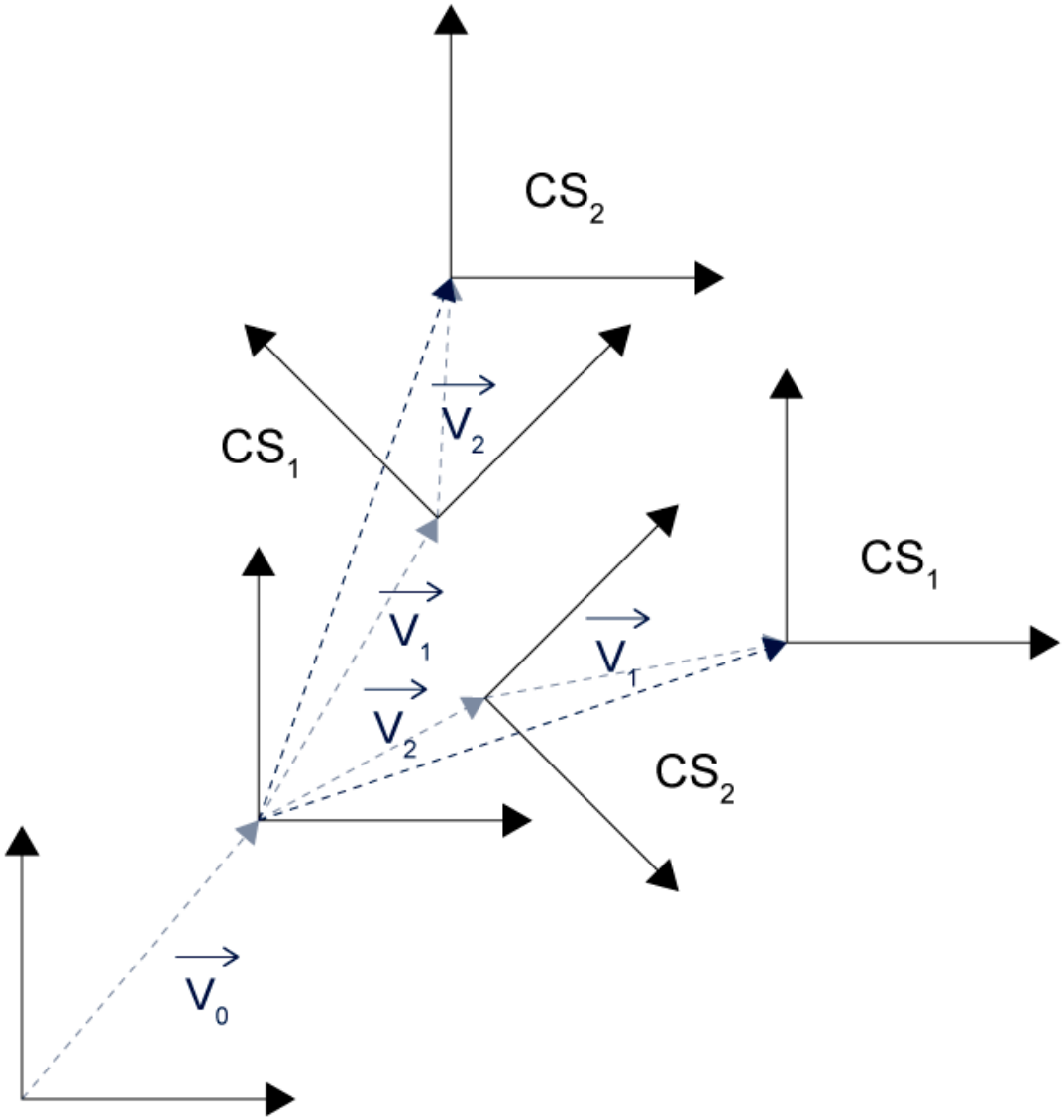


Abb. 187: Ergebnis einer CS-Verknüpfung in Abhängigkeit von der Anwahlreihenfolge (CS[1] - CS[2] bzw CS[2] - CS[1]).

Das CS (oder ACS) mit der gleichen ID kann auch mehrfach angewählt und mit sich selbst verkettet werden.

Programmierbeispiel

Verkettung von Koordinatensystemen

```
N10 #CS DEF[1][0,0,0,0,0,20]
N20 LL TEILEPRG                    (Kontur im System X-Y)
N30 #CS ON[1]
N40 LL TEILEPRG                    (X'-Y')
N50 #CS ON[1]
N60 LL TEILEPRG                    (X''-Y'')
N70 #CS OFF
N80 #CS OFF
M30
```

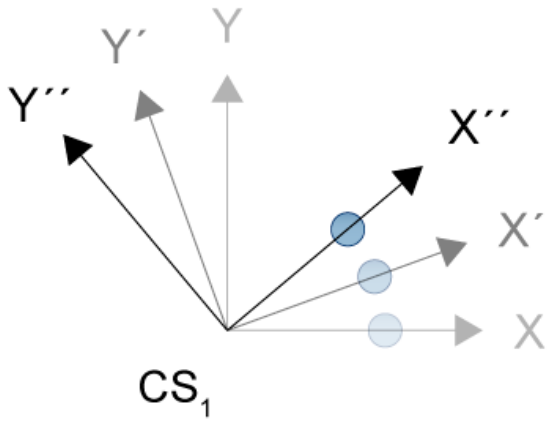


Abb. 188: Verkettung von Koordinatensystemen

Über folgende NC-Befehle kann die aktuell aktive Gesamttransformation gespeichert werden:

Syntax:

#CS DEF ACT [<*CS_ID*>]

#ACS DEF ACT [<*ACS_ID*>]

Mit folgenden NC-Befehlen lassen sich, im Gegensatz zur sequentiellen Abwahl der (A)CS über (A)CS OFF, die aus der Verknüpfung von CS bzw. ACS gebildeten Teiltransformationen direkt abwählen.

Syntax:

#CS OFF ALL	Abwahl aller CS
#ACS OFF ALL	Abwahl aller ACS

Programmierbeispiel

Verkettung von Koordinatensystemen

```
N10 #CS ON[3]
N20 #CS ON[4]
N30 #CS DEF ACT[5]          (Speicherung CS[3] o CS[4] unter CS[5])
N31 #CS OFF ALL             (Abwahl aller CS)
N32 #ACS ON[3]
N33 #ACS ON[4]
N34 #ACS DEF ACT[5]         (Speicherung ACS[3] o ACS[4] unter CS[5])
N35 #ACS OFF ALL            (Abwahl aller ACS)
N36 X0 Y0 Z0
N360 #CS ON [5]
N370 #ACS ON[5]
N380 #CS DEF ACT[1]         (Speicherung ACS[5] o CS[5] unter CS[1])
N390 #ACS OFF ALL
N400 #CS OFF ALL
N500 #CS ON                 (Anwahl CS[1])
N510 #CS OFF
M30
```

17.9.1.4 Definition/Aktivierung eines Basiskoordinatensystems (#BCS)

Versionshinweis

Diese Funktionalität ist verfügbar ab V3.1.3079.36

Das Basiskoordinatensystem (BCS) dient zur Kompensation von Verschiebungen zum Basissystem.

`Syntax der BCS-Programmierung:`

Definition und Speicherung eines BCS:

#BCS DEF [**[***<BCS-ID>***]**] **[** *<v1>,<v2>,<v3>,<φ1>,<φ2>,<φ3>* **]**

Definition und Speicherung mit gleichzeitiger Aktivierung:

#BCS ON [**[***<BCS-ID>***]**] **[** *<v1>*,*<v2>*,*<v3>*,*<φ1>*,*<φ2>*,*<φ3>* **]**

#BCS ON [*<BCS-ID>***]** Anwahl eines gespeicherten BCS

#BCS ON Anwahl des zuletzt definierten BCS

#BCS OFF Abwahl des zuletzt aktivierten BCS

Der Parameter BCS-ID darf hier nicht programmiert werden, da nur die Abwahl des zuletzt aktivierten BCS erlaubt ist.

<BCS-ID> Koordinatensystem-ID. Bei Programmstart wird die BCS-ID mit dem Standardwert 1 belegt. Wird bei #BCS DEF bzw. #BCS ON die BCS-ID nicht programmiert, so wird dafür die nächste freie BCS-ID automatisch ermittelt. BCS dieser Art sind jedoch nach ihrer Abwahl mit #BCS OFF nicht mehr verfügbar!

<vi> Komponenten des translatorischen Verschiebungsvektors in [mm, inch]. (Diese beziehen sich auf die Hauptachsen in der Reihenfolge bei G17).

<φi> Drehwinkel in [°].

Hinweis: Die Komponenten von Verschiebung und Drehung sind durch Kommas getrennt. Lücken in der Reihenfolge können durch aufeinanderfolgende Kommas ", ," markiert werden. Zur besseren Lesbarkeit wird jedoch empfohlen, diese Komponenten mit 0 zu programmieren.

Beispiel: [,,10,,,45] ↔ [0,0,10,0,0,45]

Eine verkürzte Programmierung der Komponenten ist ebenfalls zulässig.

Beispiel: [0,0,10] nur Verschiebung

[0,0,10,0,30] Verschiebung und 2 Drehwinkel

Das BCS ist haltend wirksam und kann unabhängig von einem CS/ ACS an- bzw. abgewählt werden.

Im BCS können Nullpunkt- und Bezugspunktverschiebungen programmiert werden. Diese sind allerdings nur bis zur Abwahl des BCS gültig und werden nicht gespeichert.

#BCS OFF ALL Abwahl aller BCS

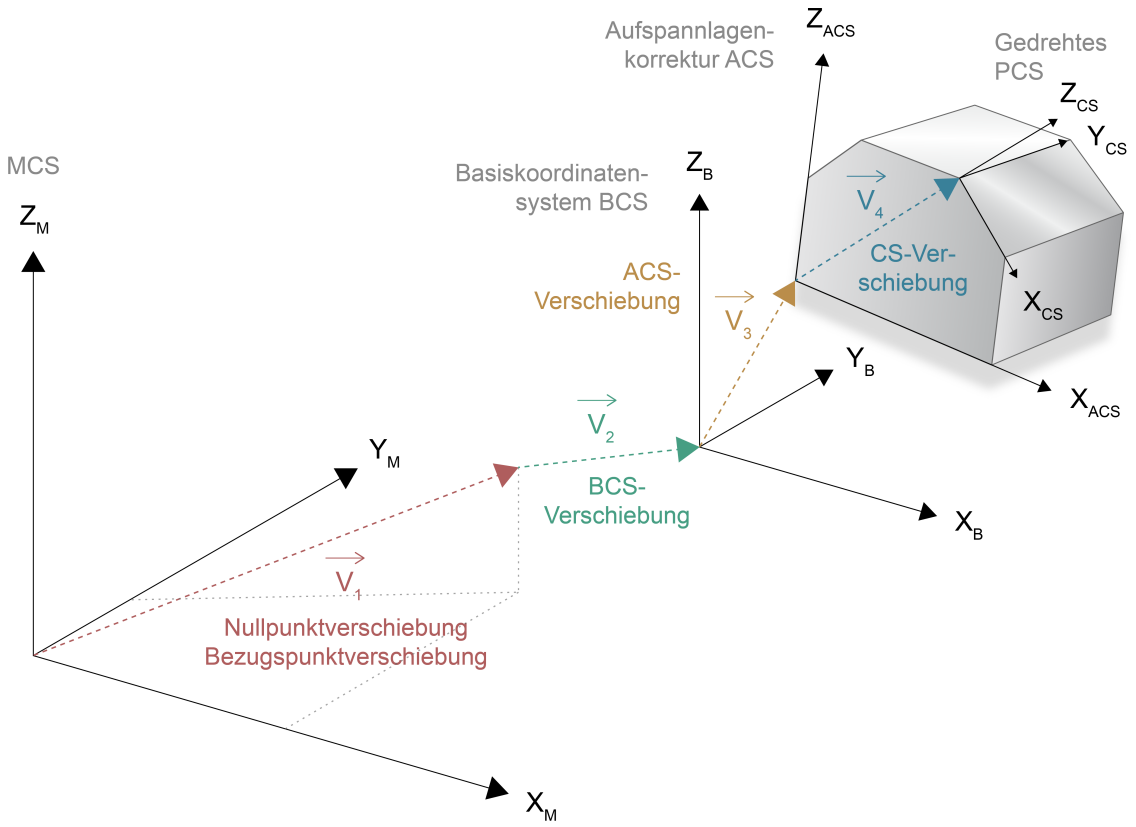


Abb. 189: Verkettung mit Basiskoordinatensystem #BCS

17.9.1.5 Verschiebungen im Koordinatensystem

Wird ein neues Koordinatensystem definiert und aktiviert, so sind die Verschiebungen in diesem Koordinatensystem zunächst NULL. Programmierte Verschiebungen in unterlagerten Koordinatensystemen bleiben wirksam.

Im neuen, aktuellen CS können zusätzliche Verschiebungen programmiert werden. Diese wirken additiv zu allen bisher wirksamen unterlagerten Verschiebungen, sind aber nur bis zur Abwahl des aktuellen CS gültig (lokal) und werden nicht gespeichert.

Folgende Verschiebungen sind lokal pro Koordinatensystem möglich:

- Nullpunktverschiebungen (G53, G55 ..)
- Bezugspunktverschiebungen (G92)
- Istwertverschiebungen (#PSET)

Hinweis

Die werkzeugspezifischen Achsverschiebungen sind nicht pro Koordinatensystem, sondern immer global gültig.

Die relative Addition von Achsverschiebungen pro Koordinatensystem gilt für Hauptachsen (ersten drei Kanalachsen des Koordinatensystems) **und** für alle weiteren im Kanal vorhandenen Achsen, den s.g. Mitschleppachsen (Kanalachsen nach den 3 Hauptachsen, rotatorische Achsen, etc.).

Die Wirksamkeit der Verschiebungen der Mitschleppachsen kann mit folgender #CS MODE Option entweder für das aktuelle CS oder global geschaltet werden. Entsprechend ändern sich dadurch auch die Werte der Verschiebungen in der Anzeige.

Syntax für Verwaltung der Verschiebungen der Mitschleppachsen:

#CS MODE ON [AXES_OFFSETS_LAYER_SPECIFIC]	Verschiebungen wirksam im aktuellen CS.
#CS MODE OFF [AXES_OFFSETS_LAYER_SPECIFIC]	Verschiebungen global wirksam. Keine relative Addition.

Die Grundeinstellung kann mit dem Kanalparameter P-CHAN-00397 festgelegt werden.

Beispiel

Entwicklung der Verschiebungswerte im CS, wenn:

A: Verschiebungen lokal im CS wirksam sind

B:Verschiebungen global wirksam sind

```
N10 X0 Y0 Z0 A0 B0
N20 G92 B100          ;A: X=0, B=100
                      ;B: X=0, B=100
:
N100 #CS ON [ICS][50,50,0]
N120 G92 X100 B200
N110 X1 B1            ;A:X=50+100+1=151, B=100+200+1=301
                      ;B:X=50+100+1=151, B=200+1=201
```

```
:
N200 #CS ON [PCS][20,20,0]
N220 G92 X200 B400
N210 X2 B2           ;A: X=50+100+20+200+2=372, B=100+200+400+2=702
                     ;B: X=50+100+20+200+2=372, B=400+2=402
:
N300 #CS DEL
N310 X3 B3           ;A: X=50+100+3=153, B=100+200+3=303
                     ;B: X=50+100+3=153, B=200+3=203
:
M30
```

17.9.1.6 Effektor-Koordinatensystem (#ECS ON/OFF)

Hinweis

Diese Funktionalität ist nicht im Umfang der Standardlizenz enthalten!
Die Nutzung dieser Funktionalität erfordert die Lizenzierung des Erweiterungspaketes "Transformationen".

Das Effektor-Koordinatensystem wird hauptsächlich zur Durchführung einer Rückzugsstrategie nach Werkzeugbruch, NC-Reset oder Programmabbruch bei Bearbeitung mit beliebig orientiertem Werkzeug verwendet. Bei der Bestimmung des ECS handelt es sich um die Umkehrung des Befehls TOOL ORI CS (Kapitel Werkzeug ausrichten (#TOOL ORI CS) [▶ 743]). Hier wird statt der Ausrichtung des Werkzeugs auf die Bearbeitungsebene eine zur Werkzeugachse senkrechte Bearbeitungsebene ermittelt.

Syntax:

#ECS ON	Anwahl ECS
#ECS OFF	Abwahl ECS

Bei Aktivierung des ECS darf kein anderes Koordinatensystem (KS) aktiv sein.

Das ECS wird aus den Positionen der Orientierungsachsen so bestimmt, dass dessen Z-Achse parallel zur aktuellen Werkzeugachse ist. Die Lage der X- und Y-Achse ist hierbei unbestimmt (beliebig) und muss deshalb intern festgelegt werden. Der Nullpunkt des ECS liegt im allgemeinen Fall außerhalb der Werkzeugspitze oder der Werkzeugachse, so dass ein kollisionsfreier Rückzug des Werkzeugs nur durch relative Verfahrbewegungen entlang der Effektor Z-Achse sichergestellt werden kann.

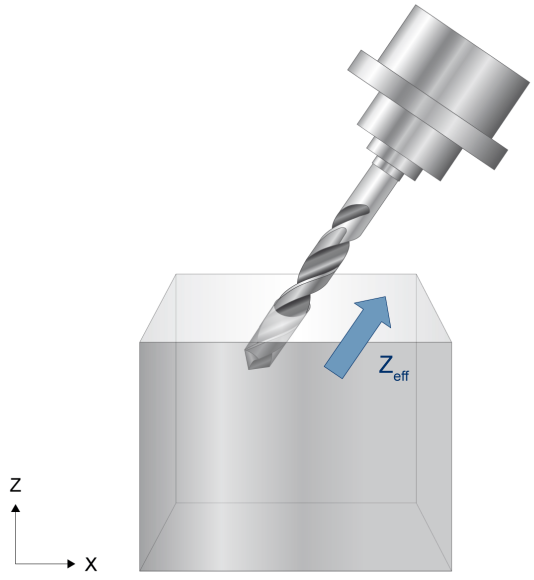


Abb. 190: Bearbeitung in schräger Bohrung

Programmierbeispiel

Effektor Koordinatensystem (ECS)

```
N01 #TRAFO ON                    (Anwahl Kinematik)
N05 #CS ON[1.5,0,32,14.5,0,45] (Anwahl eines BKS)
N10 #TOOL ORI CS
N15 X0 Y0 Z0
N20 LL TEILEPRG    (Unterprogrammaufruf zur Konturbearbeitung)
...
(Werkzeugbruch, NC-Reset)
(Rückzugsstrategie)
N01 #TRAFO ON      (Anwahl Kinematik)
N05 #ECS ON        (Berechnung des ECS)
                   (entsprechend der Position der Orientierungs-)
                   (achsen)
N10 G91 G01 F200
N20 Z62            (Rückzugsbewegung entlang der WZ-, ECS-Z-Achse)
:
N400 M30
```

17.9.1.7 Temporärer Übergang in das Maschinenkoordinatensystem (#MCS ON/OFF)

Hinweis

Nicht im Umfang der Standardlizenz! Die Nutzung dieser Funktionalität erfordert die Lizenzierung des Erweiterungspaketes "Transformationen".

Mit der Funktionalität MCS lassen sich die aktive kinematische und / oder kartesische Transformation sowie sämtliche, in die Achsen eingerechnete Versätze temporär deaktivieren, so dass die Maschinenachsen direkt positioniert werden können. Nach Verlassen des MCS wird der bei Anwahl vorhandene Zustand wiederhergestellt.

Beispielsweise erfordert das Wechseln des Werkzeuges meist das Anfahren einer definierten Werkzeugwechselposition mit, bezüglich des Maschinennullpunkts, bekannten Koordinaten. Im KS kann das Anfahren dieser Maschinenposition problematisch werden, da über das NC-Programm die KS-Achsen positioniert werden.

Syntax:

#MCS ON [**EX TOOL**]	Aktivierung temporärer Übergang in MCS
#MCS OFF	Deaktivierung temporärer Übergang in MCS

EX TOOL	Bei Werkzeugwechsel im MCS werden keine Werkzeugversätze eingerechnet, so dass auch weiterhin die Maschinenachsen direkt positioniert werden können. Dies erfolgt erst bei #MCS OFF.

Im MCS bestehen keine Beschränkungen bezüglich verwendbarer NC-Funktionalität, die An-/Abwahl von RTCP / TLC , CS, ACS, BCS und ECS ist jedoch nicht möglich.

Programmierte Verschiebungen sind auch im MCS nur bis zu dessen Abwahl gültig und werden nicht gespeichert.

Bei Änderung der Achskonfiguration durch externen Achstausch (z.B. #CALL AX ...) im MCS ist zu beachten, dass zur Reaktivierung der kinematischen und / oder der kartesischen Transformation eine definierte Achsanordnung notwendig ist.

Programmierbeispiel

Temporärer Übergang in das Maschinenkoordinatensystem (MCS)

```
N10 #TRAFO ON
N20 #CS ON[1.5,0,32,14.5,0,45]    (Anwahl eines BKS)
N30 G01 G90 F5000
N40 X0 Y0 Z0
N50 #MCS ON EX TOOL    (Übergang in das Maschinen-KS mit der Option)
                       ('EX TOOL' - WZ wird erst bei MCS OFF)
                       (eingerechnet)
N60 T1 D1              ( Werkzeuglänge wird NICHT hier verrechnet,
                       ( sondern erst in Zeile N70)
;...
N70 #MCS OFF           (Abwahl des MCS, RTCP und CS werden wieder)
                       (aktiviert)
N100 #TRAFO OFF
N110 #CS OFF
N400 M30
```

17.9.1.8 Hilfsfunktionen zur Koordinatentransformation (#WCS TO MCS, #MCS TO WCS)

Hinweis

Nicht im Umfang der Standardlizenz! Die Nutzung dieser Funktionalität erfordert die Lizenzierung des Erweiterungspaketes "Transformationen".

Bei 5-Achsmaschinen und Maschinen mit nicht kartesischem Achsaufbau (z.B. Hexapoden) finden zwei typische Varianten der Festlegung der Verfahrbewegungen Anwendung.

1. Fall: Anwender programmiert die Raumkontur über Kreis oder Gerade bzw. Polynomform und die Werkzeugspitze (TCP) wird entsprechend dem programmierten Konturverlauf auf der Bahn geführt.

2. Fall: Anwender programmiert den Zielpunkt in Raum/Werkstückkoordinaten (WCS) die in Maschinenkoordinaten (MCS) abgebildet werden. Der TCP bewegt sich entsprechend den maximal möglichen Geschwindigkeiten in den Achsen auf einer nicht einfach vorhersehbaren Kurve (PTP). Bedingt durch den Wegfall der Bewegung des TCP entlang einer Raumkurve ist die PTP-Bewegung im Allgemeinen schneller als die TCP-Bahnbewegung.

Die oben genannte Abbildung von programmierten WCS-Zielpunkten in MCS-Zielpunkte (Rückwärtstransformation) kann im NC-Programm durch nachfolgende NC-Befehle beauftragt werden. Das Anfahren der berechneten MCS-Zielpunkte muss dann vom Anwender über Absolutmaßeingabe explizit programmiert werden.

Diese Methode kann z.B. dazu verwendet werden, sequentiell jede Einzelachse aus einem Kollisionsbereich herauszufahren.

Achtung

Die Funktionen können nur bei inaktiver kinematischer Transformation (#TRAFO OFF) und inaktivem Koordinatensystem (#CS OFF (ALL)) verwendet werden.

`Syntax Berechnung von Maschinenkoordinaten (MCS) aus Werkstückkoordinaten (WCS):`
#WCS TO MCS [[**CS..**] [**KIN** [**IS[**<*kin_id*>**]=..**]] **]**

CS..	Berechnung der Zielpunkte unter Berücksichtigung einer inaktiven kartesischen Transformation mit einer bestimmtem gültigen ID.
KIN	Berechnung der Zielpunkte unter Berücksichtigung der aktuell gültigen, aber inaktiven kinematischen Transformation.
IS[<*kin_id*>]=..	Bei kinematischen Transformationen, die aus Teilkinematiken mit mehreren Lösungen bestehen, kann die gewünschte Lösung mittels zusätzlichem IS-Wert angegeben werden, vgl. Status & Turn [▶ 799]. Die Kinematik-ID von IS adressiert die Teilkinematik, die mehrere Lösungen hat. Der Statuswert beschreibt die gewünschte Pose. Verfügbar ab V3.1.3080.13

Für die Angabe der WCS-Zielpunkte und die Ablage der berechneten MCS-Zielpunkte stehen die folgenden achsspezifischen Variablen zur Verfügung, auf die lesend und schreibend zugegriffen werden kann:

V.A.WCS.* Zugriff auf achsspezifische Hilfsgröße Werkstückkoordinate (WCS). Der Wert entspricht bei aktiver Transformation i. a. nicht den programmierten Werkstückkoordinaten.

V.A.MCS.* Zugriff auf achsspezifische Hilfsgröße Maschinenkoordinate (MCS). Der Wert entspricht bei inaktiver Transformation nicht den programmierten Maschinenkoordinaten.

Programmierbeispiel

Standard-Rückwärtstransformationen mit #WCS TO MCS

Die NC berechnet über kartesische und kinematische Rückwärtstransformation von bestimmten WCS-Punkten die MCS-Zielpunkte. Auf die berechneten MCS-Positionen kann der Anwender über NC-Programmierung zugreifen und im NC Programm weiterverwenden. Der Anwender bestimmt über die Angabe der Schlüsselwörter und der dazugehörigen CS-ID die Transformationen, die durchgeführt werden sollen.

```
N02 #CS DEF[1][0,0,0,0,0,45]
N00 #KIN ID [1]
N10 V.A.WCS.X=10
N20 V.A.WCS.Y=10
N30 V.A.WCS.Z=100
N40 #WCS TO MCS[CS1, KIN] ;Transformation der WCS in MCS
N50 G00 Z=V.A.MCS.Z ;Fahren der Einzelachsen auf MCS-Zielpunkte
N60 G00 X=V.A.MCS.X
N70 G00 Y=V.A.MCS.Y
N…
:
Fehlerhafte Verwendungen:
N05 #CS DEF[1][0,0,0,0,0,45]
N10 #CS ON[1]
N20 V.A.WCS.X=100
N30 V.A.WCS.Y = 0
N40 V.A.WCS.Z = 0
N50 #WCS TO MCS [CS1] <- nicht erlaubt, da ein CS aktiv ist


N05 #KIN ID[12]
N10 #TRAFO ON
N20 V.A.WCS.X=100
N30 V.A.WCS.Y = 0
N40 V.A.WCS.Z = 0
N50 #WCS TO MCS [KIN] <- nicht erlaubt, da eine Trafo aktiv ist
```

Programmierbeispiel

Rückwärtstransformationen mit #WCS TO MCS bei kinematischen Transformationen mit mehreren Lösungen

Bei kinematischen Transformationen mit mehreren WCS-Lösungen zu einem MCS-Wert kann die gewünschte Lösung mittels IS-Wert („Status“) angegeben werden.

```
N10 #KIN ID[45]
N20 #WCS TO MCS[KIN IS[45]='0b010']
:
;Verwendungen mit Koppelkinematik (Kinematiktyp 210), bestehend
;aus Teilkinematik mit ID 45 (mehrere Lösungen) und Teilkinematik
;mit ID 91 (eine Lösung):
N10 #KIN ID[210]
N20 #WCS TO MCS[KIN IS[45]='0b010']
:

N10 #KIN ID[45]
N20 #WCS TO MCS[KIN IS='0b010'] <- nicht erlaubt, da für IS keine Teil-
kinematik-ID angegeben wurde

N10 #KIN ID[210]
N20 #WCS TO MCS[KIN IS[91]='0b010'] <- nicht erlaubt, da Teilkinematik
91 nicht mehrere Lösungen hat
```

Die umgekehrte Abbildung von MCS-Zielpunkten in WCS-Zielpunkte (Vorwärtstransformation) wird mit dem folgenden Befehl durchgeführt. Diese Methode kann z.B. zur Abbildung von Messwerten (i.d.R. im MCS ermittelt) in das WCS verwendet werden.

Syntax Berechnung von Werkstückkoordinaten (WCS) aus Maschinenkoordinaten (MCS):

#MCS TO WCS [[**CS..**] [**KIN**] **]**

CS..	Berechnung der Zielpunkte unter Berücksichtigung einer inaktiven kartesischen Transformation mit einer bestimmtem gültigen ID.
KIN	Berechnung der Zielpunkte unter Berücksichtigung der aktuell gültigen, aber inaktiven kinematischen Transformation.

17.9.1.9 Hilfsfunktion zur Berechnung der Bewegungsgrenzen im Werkstückkoordinatensystem (#GET WCS POSLIMIT)

Über den nachfolgenden Befehl werden die Grenzen der Bewegung im aktuellen Werkstückkoordinatensystem (WCS) in Richtung eines programmierten Bewegungsrichtungsvektors mit den Komponenten (VC1, VC2, VC3) berechnet. Die Vektorkomponenten müssen nicht normiert angegeben werden. Grundlage für die Wegbegrenzung sind die Softwareendschalter auf Achsebene. Die Steuerung berechnet auf Basis dieser Werte die Bewegungsgrenzen im aktuellen Koordinatensystem.

Syntax:

#GET WCS POSLIMIT [VC1=.. VC2=.. VC3=..]

VC1=..,
VC2=.., Komponenten des Richtungsvektors, Realzahl
VC3=..

Das Ergebnis der Berechnung kann mit den folgenden globalen Variablen für die ersten drei Achsen im Koordinatensystem ausgelesen werden.

V.G.WCS_POSLIMIT_1	Bewegungsgrenze in der ersten Hauptachse im WCS
V.G.WCS_POSLIMIT_2	Bewegungsgrenze in der zweiten Hauptachse im WCS
V.G.WCS_POSLIMIT_3	Bewegungsgrenze in der dritten Hauptachse im WCS

Achtung

Die programmierte Bewegungsrichtung muss bei der tatsächlich ausgeführten Bewegung zwingend eingehalten werden, ansonsten stimmen die berechneten Bewegungsgrenzen nicht!

Ebenso dürfen bei aktiver kinematischer Transformation (#RTCP) keine rotatorischen Achsen programmiert werden!

Programmierbeispiel

Hilfsfunktion zur Berechnung der Bewegungsgrenzen im Werkstückkoordinatensystem

```
N05...
N10 G98 X-100 Y-100 Z-100          (Verschieben der negativen SWE)
N20 G99 X200 Y200 Z300           (Verschieben der positiven SWE)
N30 #ECS ON                        (Anwahl Effektorkoordinatensystem)
N40 #GET WCS POSLIMIT [VC1=0 VC2=0 VC3=1] (Berechnen WCS Bewegungsgren-
ze)
N50 G01 G90 Z=V.G.WCS_POSLIMIT_3 F2000 (Anfahren WCS Bewegungsgrenze in
Z)
N60 #ECS OFF                            (Abwahl Effektorkoordinatensystem)
N70...
```

Beispiel für richtige Verwendung des Befehls

```
N05…
N10 #CS ON[0,0,0,0,0,45]
N20 #GET WCS POSLIMIT [VC1=1 VC2=1 VC3=0]
N25 G01 G90 F2000
N30 X=V.G.WCS_POSLIMIT_1 Y=V.G.WCS_POSLIMIT_2 Z=V.G.WCS_POSLIMIT_3
N40 #CS OFF
N50...
```

Falsch, tatsächliche Bewegungsrichtung nicht nach Vorgabe:

```
N05…
N10 #CS ON[0,0,0,0,0,45]
N20 #GET WCS POSLIMIT [VC1=1 VC2=1 VC3=0]
N25 G01 G90 F2000
N30 X=V.G.WCS_POSLIMIT_1 Y=V.G.WCS_POSLIMIT_2
N40 Z=V.G.WCS_POSLIMIT_3
N50...
```

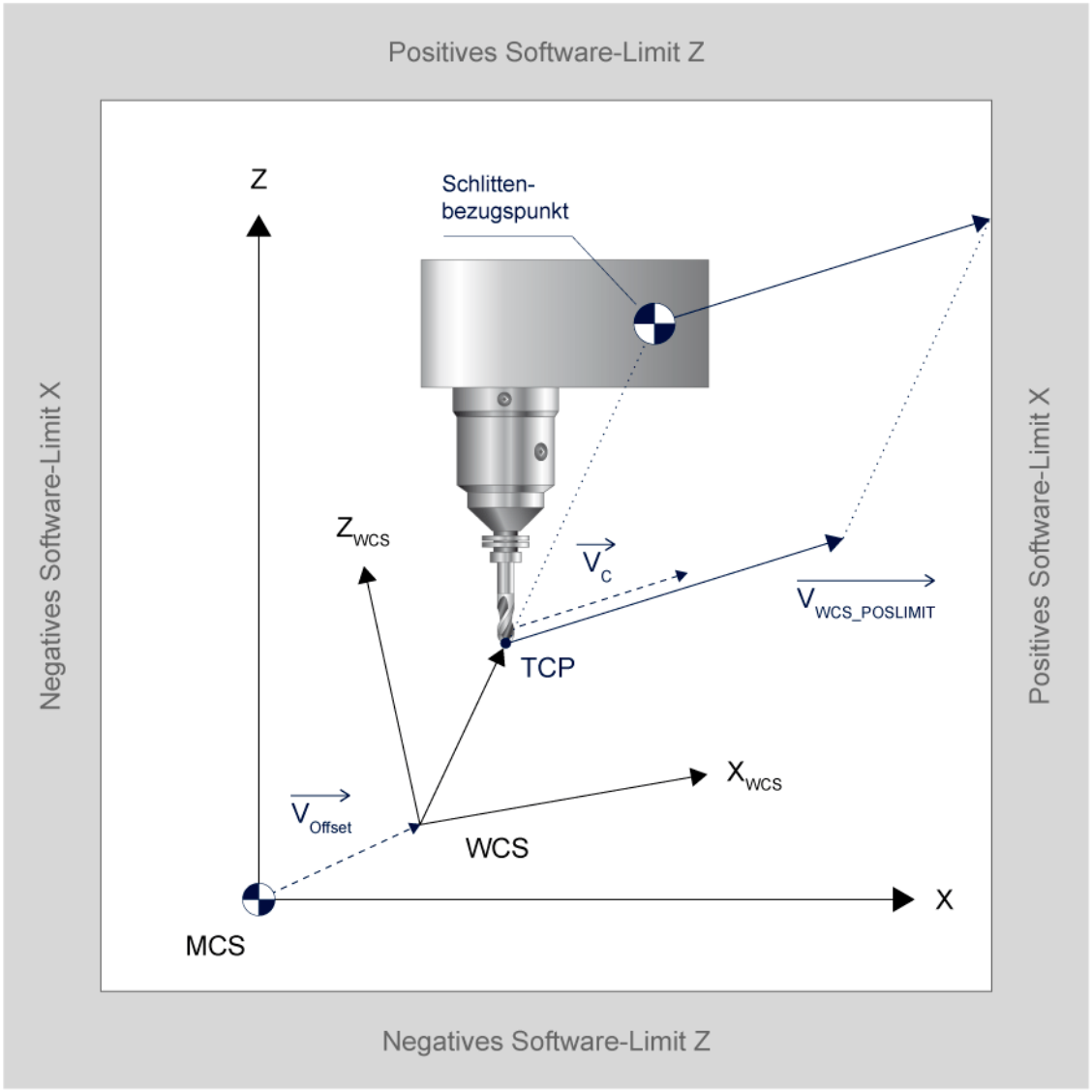


Abb. 191: Bewegungsgrenzen in der ZX-Ebene im aktuellen WCS

17.9.2 Erweiterte Programmierung

Bei dieser Programmierung wird ein Koordinatensystem (CS) nicht über eine ID-Nummer sondern über einen Namen adressiert. Die Verkettung sowie der Aufbau eines CS-Stapels wird über die Reihenfolge der programmierten Definitionen bestimmt. Für die Aktivierung und Anpassung des CS-Stapels im weiteren Programmablauf stehen zusätzliche NC-Befehle zur Verfügung.

17.9.2.1 Definition und Verkettung von Koordinatensystemen (#CS ADD)

Syntax Definition und Verkettung von CS:

#CS ADD [<*Name*>**] [** <*v1*>,<*v2*>,<*v3*>,<*φ1*>,<*φ2*>,<*φ3*> **]**

<*Name*>	Name des CS mit maximal 8 Zeichen. Mit diesem Name ist das CS mit allen weiteren entsprechenden NC-Befehlen adressierbar. Die Namen ACS, IMCS und MCS haben eine vorbelegte feste Bedeutung und dürfen nicht frei vergeben werden. (siehe dazu #TRANSFORM-Befehl). Maximal können 5 CS-Definitionen definiert und verkettet werden.
<*vi*>	3 Komponenten des translatorischen Verschiebungsvektors in [mm, inch]. (Diese beziehen sich auf die Hauptachsen in der Reihenfolge bei G17).
<*φi*>	3 Drehwinkel in [°].

Die Verkettung zu einem CS-Stapel wird durch Programmierung einer #CS ADD Sequenz aufgebaut. Hierbei bildet immer das neue zuletzt programmierte CS das oberste CS des Stapels. Die Verkettung der einzelnen CS erfolgt gemäß dem Stapelaufbau in der Reihenfolge der Programmierung von unten nach oben. Die translatorischen und rotatorischen Verschiebungen beziehen sich auf den Ursprung des vorhergehenden CS. Für die Drehreihenfolge und den Drehmodus gelten die Regeln wie bei der CS-Standardprogrammierung.

Der CS-Stapel kann auf Interpolatorebene zur Anzeige der jeweiligen transformierten aktuellen Koordinaten genutzt werden.

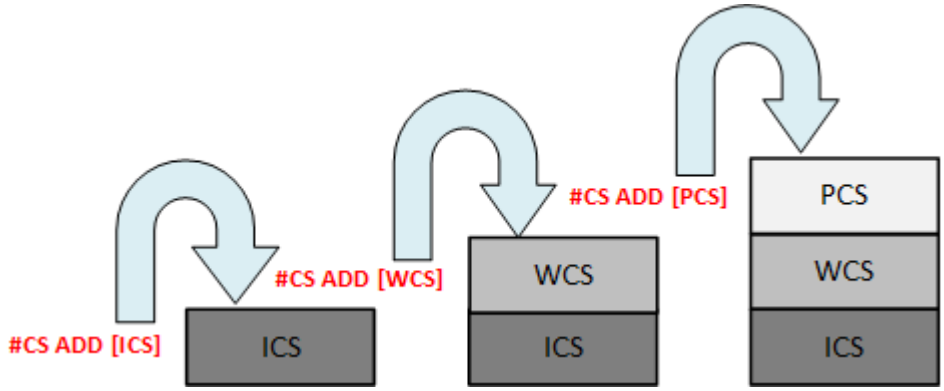


Abb. 192: Aufbau eines CS-Stapels mit #CS ADD

Programmierbeispiel

3 CS mit #CS ADD zu einem Stapel verketten

```
:
N10 #CS ADD [ICS] [10,10,0,0,0,45]
N20 #CS ADD [WCS] [15,20,0,0,0,0]
N30 #CS ADD [PCS] [10,15,0,0,0,5]
:
Nxx M30
```

17.9.2.2 Anwahl/Aktivierung eines Koordinatensystems (#CS SELECT)

Damit neu programmierte Positionen im richtigen CS des CS-Stapels wirken, muss die entsprechende Verkettung aktiviert werden. Dazu steht folgender NC-Befehl zur Verfügung:

```
Syntax Anwahl/Aktivierung CS:
```
#CS SELECT [*<Name>***]**

<Name> Name des zu aktivierenden CS bzw. verketteten Bereichs eines CS-Stapels mit maximal 8 Zeichen.

Nach Anwahl eines CS gelten alle programmierten Positionen genau in diesem CS. Alle darüberliegenden CS im CS-Stapel werden nur für Anzeigezwecke genutzt. Ohne mindestens ein zuvor definiertes CS bzw. einen zuvor definierten CS-Stapel kann #CS SELECT nicht benutzt werden.

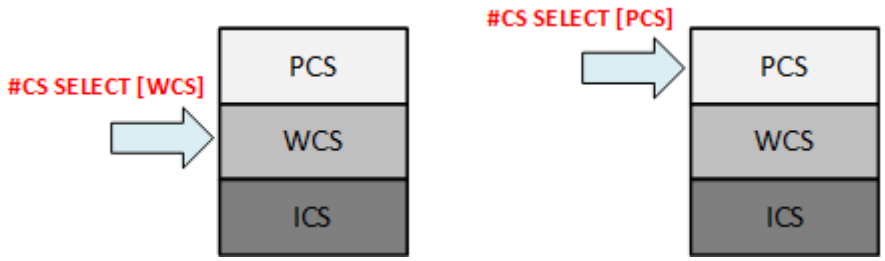


Abb. 193: Aktivierung CS Stapel mit #CS SELECT

Programmierbeispiel

Basierend auf einem CS-Stapel eine CS-Verkettung aktivieren

```
:
N10 #CS ADD [ICS] [10,10,0,0,0,45]
N20 #CS ADD [WCS] [15,20,0,0,0,0]
N30 #CS ADD [PCS] [10,15,0,0,0,5]
:
N60 #CS SELECT [WCS]   ;Aktive CS-Verkettung ICS-WCS
:
N80 #CS SELECT [PCS]   ;Aktive CS-Verkettung ICS-WCS-PCS
:
Nxx M30
```

17.9.2.3 Anwahl/Aktivierung des Maschinenkoordinatensystems (#CS SELECT[MCS/IMCS])

Die Aktivierung einer kinematischen Transformation erzeugt ein vordefiniertes Maschinen-CS, dessen Ursprung im Maschinennullpunkt liegt.

Besteht die Transformation aus einer Stufe, so wird diesem CS der vorbelegte Namen MCS zugeordnet.

Kinematische Transformation, die aus 2 Stufen bestehen, legen neben dem MCS noch ein weiteres unterlagertes Zwischen-CS mit dem vorbelegten Namen IMCS an. Bei dieser Anordung wird das MCS durch die Kombination der beider Transformationsstufen und das IMCS durch die erste Transformationsstufe gebildet.

Diese Namen dürfen nur im Zusammenhang mit dem Anwahlbefehl #CS SELECT verwendet werden. Wird ein anwenderspezifisches CS über #CS ADD mit diesen Namen programmiert, erfolgt die Ausgabe einer Fehlermeldung.

Syntax Anwahl/Aktivierung vordefiniertes MCS/IMCS:

#CS SELECT [MCS] Anwahl Maschinen-CS , gebildet aus ein- oder zweistufiger Transformation

#CS SELECT [IMCS] Anwahl Zwischen-CS, gebildet aus erster Stufe bei zweistufiger Transformation

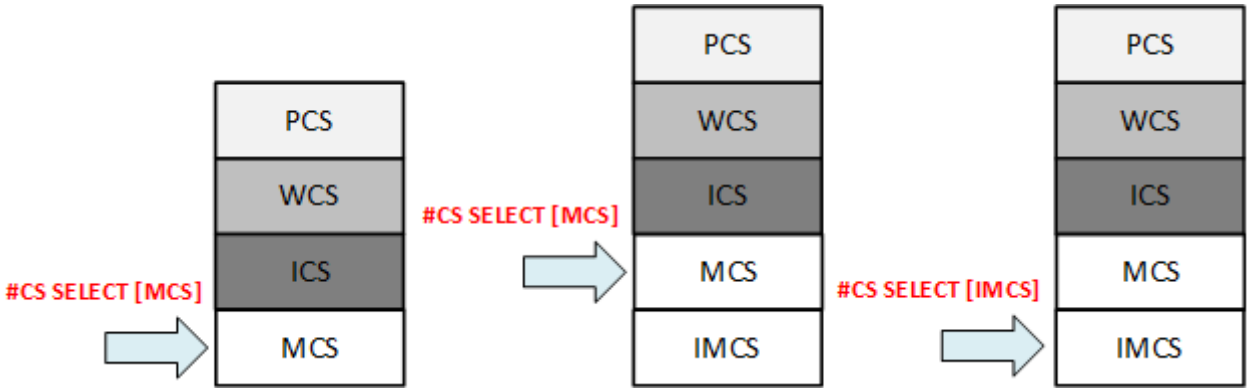


Abb. 194: Aktivierung MCS bei einstufiger… ….und zweistufiger Trafo

Hinweis

Abgrenzung #CS SELECT[MCS/IMCS] und #MCS ON

Der Befehl #CS SELECT [MCS/IMCS] aktiviert das erste kartesische CS im Maschinennullpunkt, wenn eine ein- oder zweistufige kinematische Transformation aktiv ist.

Der Befehl #MCS ON hingegen deaktiviert alle aktiven kartesischen und kinematischen Transformationen und Versätze, sodass die Maschinenachsen direkt positioniert werden können.

17.9.2.4 Ändern der Definition eines Koordinatensystems (#CS SET)

Syntax Ändern der Definition eines CS:

#CS SET [<*Name*>**] [** <*v1*>,<*v2*>,<*v3*>,<*φ1*>,<*φ2*>,<*φ3*> **]**

<*Name*>	Name des zu ändernden CS mit maximal 8 Zeichen.
<*vi*>	3 Komponenten des translatorischen Verschiebungsvektors in [mm, inch]. (Diese beziehen sich auf die Hauptachsen in der Reihenfolge bei G17).
<*φi*>	3 Drehwinkel in [°].

Mit dem Befehl #CS SET kann die Definition eines bereits mit #CS ADD definierten CS geändert werden. Es spielt hierbei keine Rolle, ob das zu ändernde CS ein aktives CS (#CS SELECT) oder ein CS in einer Verkettung eines CS-Stapels ist. Die translatorischen neuen Komponenten beziehen sich immer auf das darunterliegende CS.

Nach der CS-Änderung werden alle Positionen neu berechnet, so dass auch in der Anzeige die geänderten CS-Daten mit berücksichtigt werden.

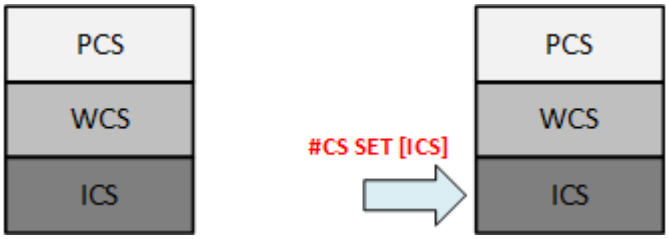


Abb. 195: Ändern einer CS-Definition mit #CS SET

Programmierbeispiel

Basierend auf einem CS-Stapel eine CS-Definition ändern

```
:
N10 #CS ADD [ICS] [10,10,0,0,0,45]
N20 #CS ADD [WCS] [15,20,0,0,0,0]
N30 #CS ADD [PCS] [10,15,0,0,0,5]
:
N80 #CS SET [ICS] [20,10,0,0,0,10]   ; Definition von ICS ändern
:
Nxx M30
```

17.9.2.5 Löschen und Abbau der Verkettung von Koordinatensystemen (#CS DEL)

```
Syntax Löschen und Abbau eines CS-Stapels:
```

#CS DEL

Durch Programmierung von #CS DEL wird immer das oberste CS eines CS-Stapels gelöscht. Wird ein aktives CS (#CS SELECT) gelöscht, dann wird das neue oberste CS automatisch zum aktiven CS.

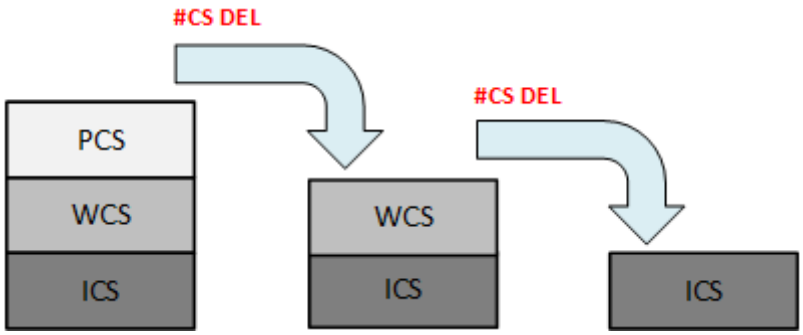


Abb. 196: Löschen eines CS mit #CS DEL

Das Löschen eines CS-Stapels kann entweder schrittweise mit #CS DEL erfolgen oder in einem Schritt mit:

#CS DEL ALL

Nach diesem Befehl sind alle CS gelöscht. Ein evtl. vorhandenes MCS wird zum aktiven CS.

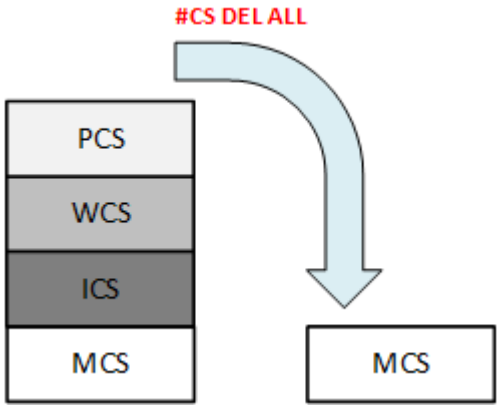


Abb. 197: Löschen aller CS mit #CS DEL ALL

Programmierbeispiel

Basierend auf einem CS-Stapel mit unterlagertem MCS löschen mit #CS DEL und #CS DEL ALL

```
:
N10 #CS ADD [ICS] [10,10,0,0,0,45]
N20 #CS ADD [WCS] [15,20,0,0,0,0]
N30 #CS ADD [PCS] [10,15,0,0,0,5]
:
N60 #CS SELECT [WCS]   ;Aktive CS-Verkettung MCS-ICS-WCS
:
N80 #CS DEL       ;Löschen von PCS, WCS wird oberstes CS
                  ;Aktive CS-Verkettung MCS-ICS-WCS
N90 #CS DEL       ;Löschen von WCS, aktive CS-Verkettung MCS-ICS
:
oder alternativ
N80 #CS DEL ALL   ;Löschen aller CS, MCS wird oberstes aktives CS
:
Nxx M30
```

17.9.2.6 Nachführen eines Koordinatensystems (#CS TRACK)

Ein bereits definiertes CS wird so verschoben, dass die aktuelle Istposition P einer vorgegebenen Nachführposition entspricht. Durch den Befehl werden die physikalischen Achsen nicht bewegt. Das nachgeführte CS muss hierbei nicht das aktive CS (#CS SELECT) sein.

Alle CS eines CS Stapels oberhalb des nachgeführten CS beziehen sich dann relativ auf die neue Lage des nachgeführten CS.

Syntax zur Definition eines nachgeführten CS:

#CS TRACK [*<Name>***]** [*<POS_X>*, *<POS_Y>*, *<POS_Z>* **]**

<Name>	Name des nachzuführenden CS mit maximal 8 Zeichen.
<POS_X, Y, Z>	3 Komponenten der neuen Istposition P in [mm, inch] im nachgeführten CS.

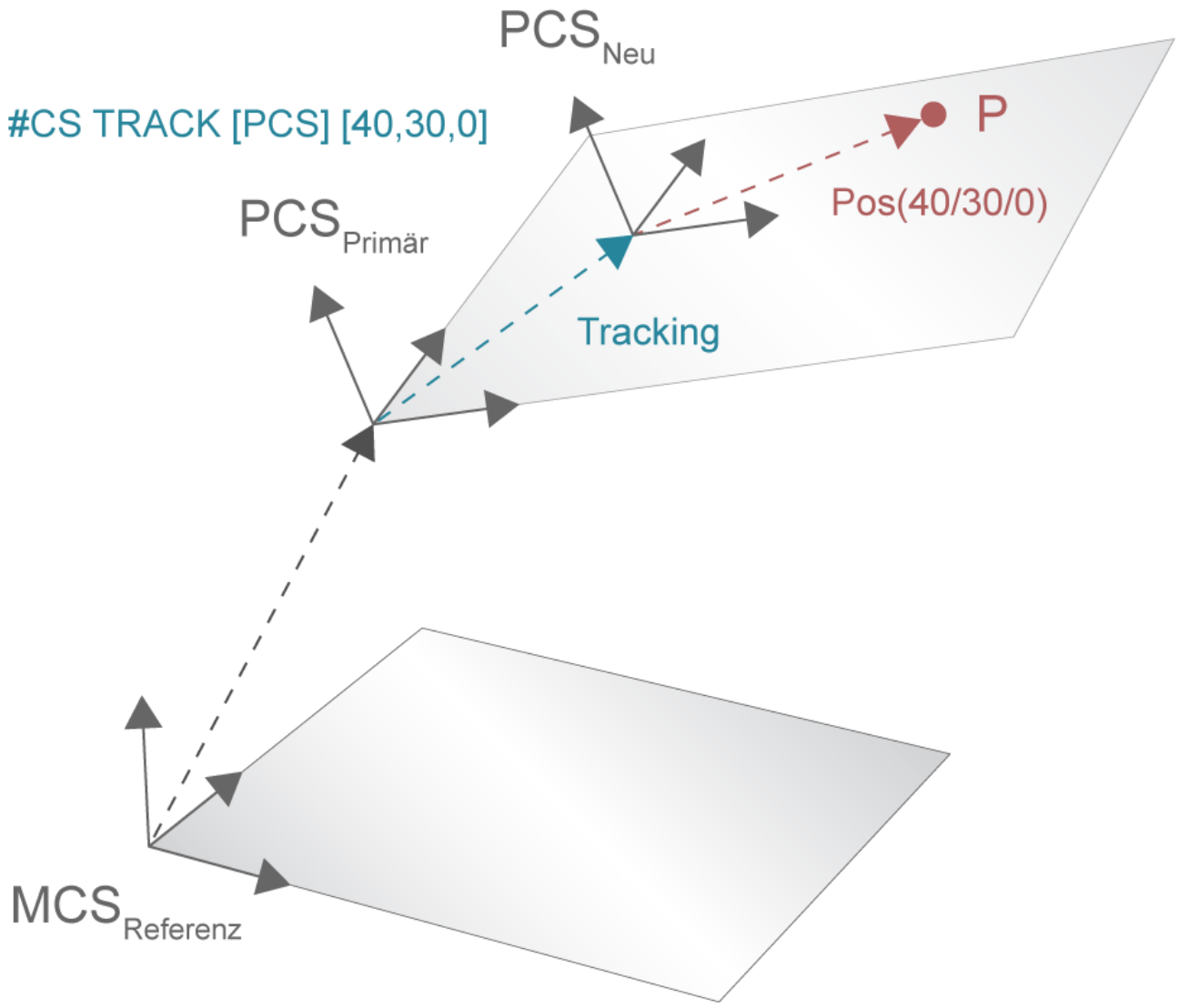


Abb. 198: Nachführen eines PCS in XY mit #CS TRACK

Programmierbeispiel

Nachführen eines CS mit #CS TRACK

Die Aufgabestellung ist wie folgt:
Ausgehend vom Punkt P(150,100,0) in einem CS soll ein nachgeführtes CS aufgespannt werden, indem dieser identische Punkt die folgende Position P_{Track}(40,30,0) hat.

```
N10 #CS ADD [PCS][...] [...]
N20 #CS SELECT [PCS]
N30 G0 X150 Y100 Z0 ; Fahre auf PCS-Position P(150,100,0)
:
N50 #CS TRACK [PCS] [40,30,0] ;P(150,100,0) im nachgeführten PCS:
P_Track(40,30,0)
:
M30
```

17.9.2.7 Lesen der Gesamtverschiebung im aktiven Koordinatensystem

Im mit #CS SELECT aktiven CS kann die durch Verkettung gebildete Gesamtverschiebung zum Referenzmaschinen-CS (MCS) durch folgende Variablen gelesen werden:

Syntax V.G.-Variablen zum Lesen der Gesamtverschiebung:

V.G.SELECTED_CS.TRANS.X	Gesamtverschiebung in 1. Hauptachse in [mm, inch]
V.G.SELECTED_CS.TRANS.Y	Gesamtverschiebung in 2. Hauptachse in [mm, inch]
V.G.SELECTED_CS.TRANS.Z	Gesamtverschiebung in 3. Hauptachse in [mm, inch]

17.9.2.8 Koordinatentransformation zwischen Koordinatensystemen (#TRANSFORM)

Hinweis

Diese Funktionalität ist eine lizenzpflichtige Zusatzoption.

Mit dem #TRANSFORM Befehl können im NC-Programm basierend auf dem aktuellen CS-Stapel beliebig Koordinaten eines 3D-Punktes von einer Stapelebene in eine andere Stapelebene umgerechnet werden. Die dazu notwendige Transformationsrechnung besteht je nach Aufbau des CS-Stapels aus einer Kombination aus kinematischer und kartesischer Vorwärts- oder Rückwärtstransformation.

Hilfs- und Zusatzachsen werden bei der Transformationsrechnung nur bzgl. ihrer Nullpunktverschiebungen zwischen den Stapelebenen berücksichtigt.

Der #TRANSFORM-Befehl kann auch bei bereits aktiven Transformationen (#CS SELECT) im Kanal verwendet werden.

Syntax des Transformationsbefehls:

#TRANSFORM [<*CS_Source*>**] [**<*CS_Dest*>**] [***POS_X*>, <*POS_Y*>, <*POS_Z*>**]** [**[** {<*POS_Aux*>=..} **]**]

<*CS_Source*>	Name des Eingangs-CS mit maximal 8 Zeichen.
<*CS_Dest*>	Name des Ziel-CS mit maximal 8 Zeichen.
<*POS_X, Y, Z*>	3 Komponenten des zu transformierenden Punktes in [mm, inch] im Eingangs-CS.
<*POS_Aux*>=..	Eingangskoordinaten der Hilfs- und Zusatzachsen

Hinweis

Eine Fehlermeldung wird generiert, wenn:

a) der Name eines Koordinatensystems nicht definiert ist

b) keine 3 Eingangskoordinaten programmiert sind

Das Ergebnis der Transformationsrechnung wird in speziellen achsspezifischen Variablen (V.A.) bereitgestellt. Die Achse kann entweder per Name oder Achsindex programmiert werden:

Syntax Ergebnisvariablen:

V.A.TRANSFORM.<*Achsname*>	Achsspezifische Koordinate nach Berechnung in [mm, inch]
oder	
V.A.TRANSFORM[<*Achsindex*>**]**	Achsspezifische Koordinate nach Berechnung in [mm, inch]

Programmierbeispiel

Folgende Beispiele zeigen die schematische Verwendung des #TRANSFORM-Befehls in einem CS-Stapel, der aus kinematischen und kartesischen Transformationen aufgebaut ist. Die Angabe von Eingangs- und Zielkoordinatensystem bestimmt hierbei, ob die Berechnung eine Vorwärts- oder Rückwärtstransformation erfordert.

Beispiel 1, Basiskonfiguration ist eine 5-Achskinematik mit CS-Stapel.

Transformation für den Punkt (50,0,10) mit den Mitschleppachspositionen A10 und B20 aus dem ACS System in das WCS-System. Es wird eine kinematisch-kartesische Vorwärtstransformation durchgeführt.

#TRANSFORM [ACS] [WCS] [50, 0, 10] [A10 B20]

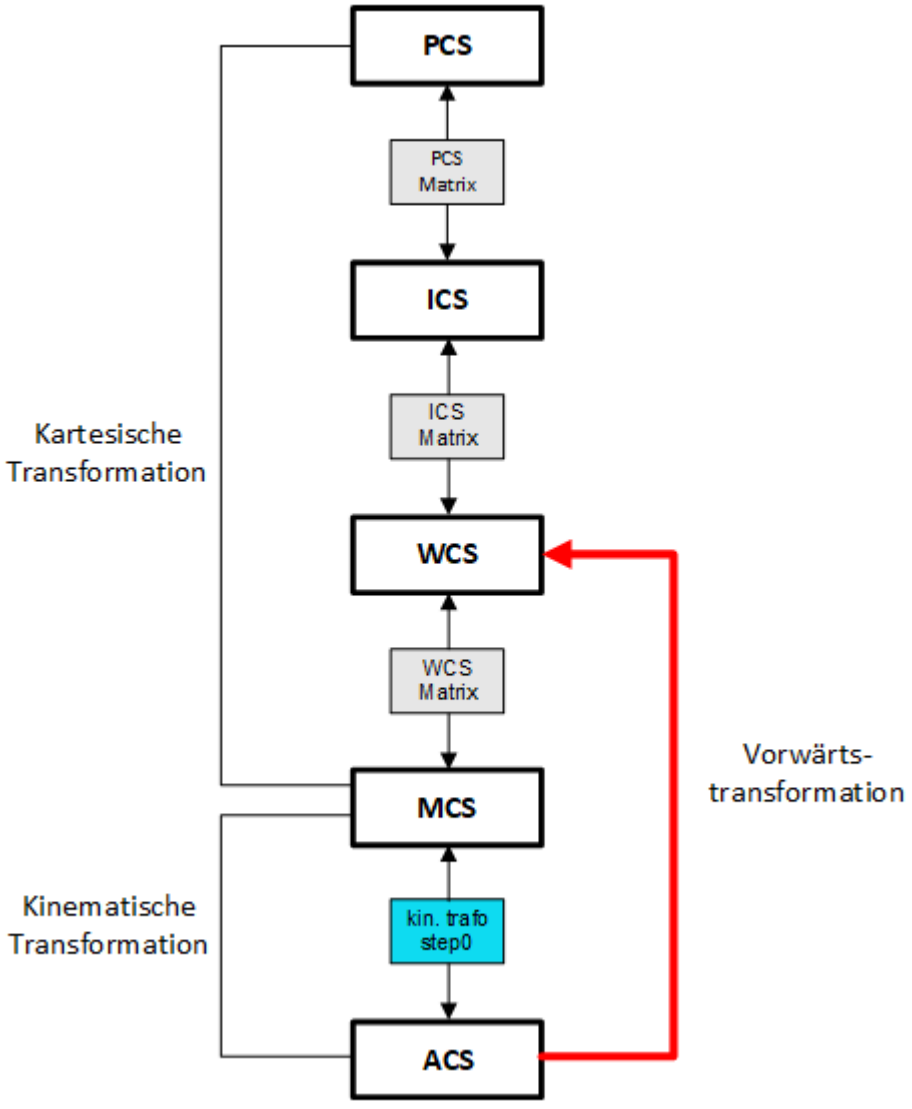


Abb. 199: Beispiel Vorwärtstransformation mit #TRANSFORM

Beispiel 2, Basiskonfiguration ist eine 5-Achskinematik mit CS-Stapel.

Transformation für den Punkt (10,-10,15) mit den Mitschleppachspositionen A45 und B90 aus dem MCS-System in das ACS-System. Es wird eine rein kinematische Rückwärtstransformation durchgeführt.

#TRANSFORM [MCS] [ACS] [10, -10, 15] [A45 B90]

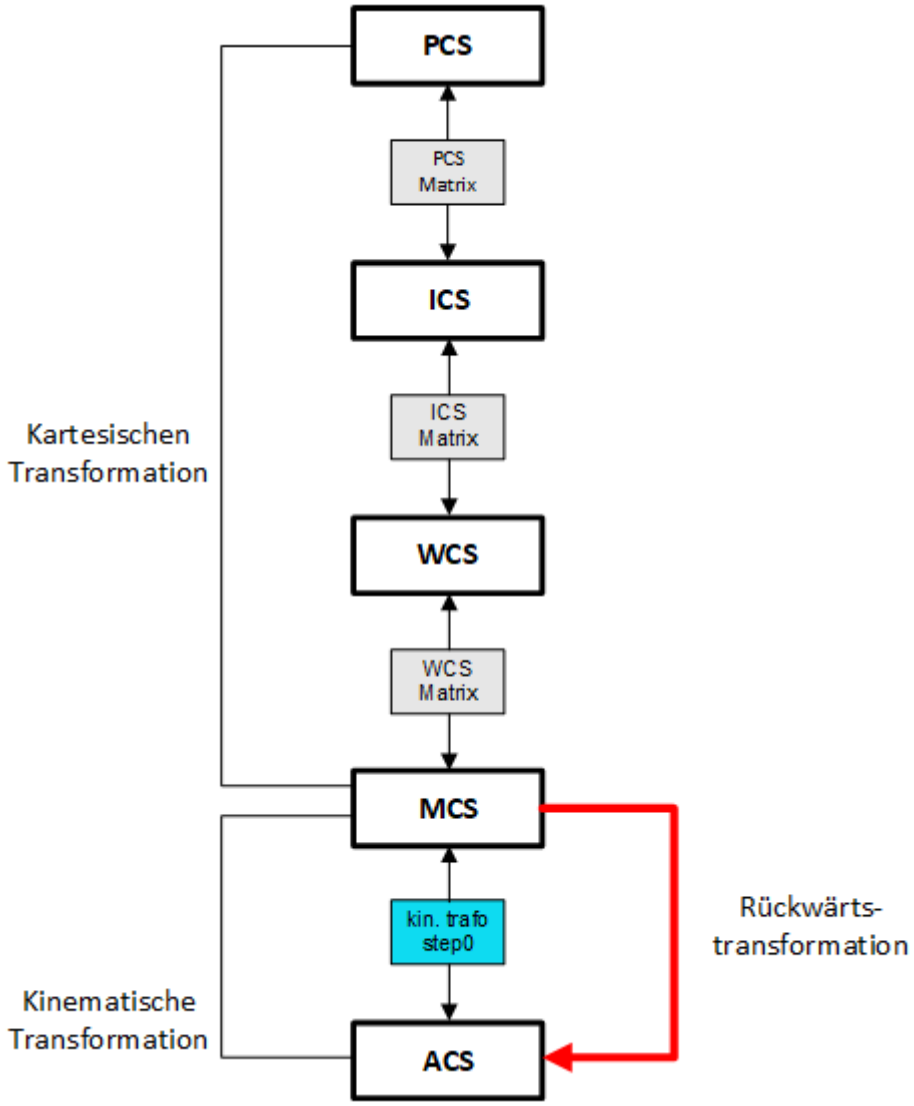


Abb. 200: Beispiel Rückwärtstransformation mit #TRANSFORM

17.10 Orientierungsprogrammierung

Die Programmierung der Werkzeugorientierung kann auf unterschiedliche Weise erfolgen. Im Wesentlichen hängt die Darstellung der Orientierung von der zugrundeliegenden Kinematik (5-Achs- bzw. Roboterkinematik) sowie den Einstellungen der eingesetzten CAD/CAM-Systeme ab.

Im klassischen Fall werden bei der 5-Achsbearbeitung die kartesischen Koordinaten in Punkt/Euler-Winkel-Darstellung programmiert. Bei einer Fünfachsmaschine mit CA Werkzeugkopf also z.B. die Positionen über X, Y und Z und die Maschinenwinkel für die Orientierung über die konfigurierten Achsbezeichner C1 und A1.

Beispiel:

X50 Y50 Z100 C1=45 A1=45

Da in CAD/CAM-Systemen die Konturen eines Werkstücks in der Regel in Vektordarstellung vorliegen, ist es üblich, die NC-Programme ebenfalls in Vektordarstellung zu generieren. Dadurch ist die Konturbeschreibung unabhängig von der Maschinen- bzw. Kinematikstruktur.

Die Werkzeugorientierung wird durch einen Vektor definiert, der von der Werkzeugspitze (TCP) in Richtung der Werkzeugeinspannung gerichtet ist. Die Komponenten des Richtungsvektors werden immer mit A, B, C (bzw. I, J, K) programmiert. Bei aktiver Vektorprogrammierung können deshalb gleichnamig konfigurierte Mitschleppachsen aus Gründen der Eindeutigkeit nicht programmiert werden.

Bei 5-Achskinematiken erfolgt die Definition der Vektorkomponenten über die "virtuellen" Achsbezeichner A, B und C.

Beispiel:

X50 Y50 Z100 A-0.5 B0.5 C0.7071

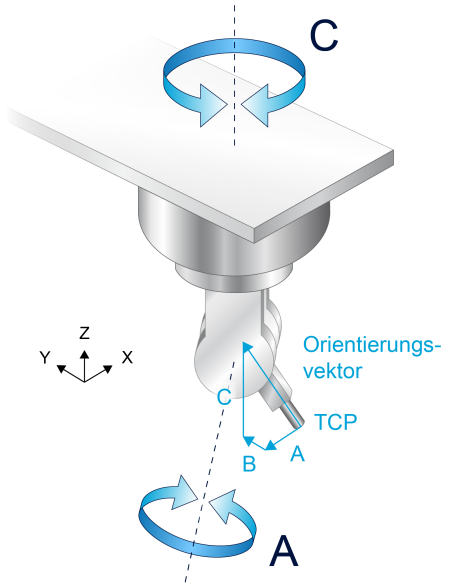


Abb. 201: Orientierungsvektor am 5-Achskopf

Bei 6-Achskinematiken (z.B. Roboter) erfolgt die Definition der Vektorkomponenten ebenfalls über die "virtuellen" Achsbezeichner A, B und C bzw. die im Robotikbereich auch verwendeten Achsbezeichner I, J und K.

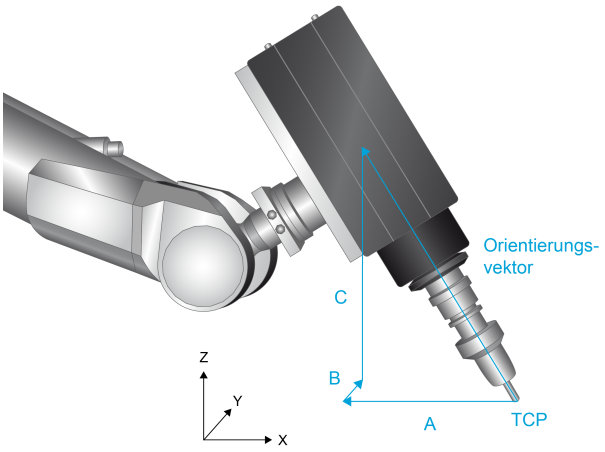


Abb. 202: Orientierungsvektor am Roboter

Beispiel:

X-17.083 Y29.630 Z10 A-0.17083 B0.29630 C0.93969 oder

X-17.083 Y29.630 Z10 I-0.17083 J0.29630 K0.93969

Für die korrekte Auswertung der Werkzeugorientierung muss der entsprechende Orientierungsmode gesetzt und die entsprechende Transformation aktiviert worden sein.

17.10.1 Programmierung und Konfiguration für 5-Achskinematiken (#ORI MODE)

Die Auswertung der Punkt-Vektorprogrammierung nach Trafoanwahl wird durch den NC-Befehl #ORI MODE[..] aktiviert. Für die klassische Punkt-Vektor-Darstellung wird der Mode VECTOR_2DOF (2 **D**egrees **O**f **F**reedom) gesetzt. Er bleibt bis zum Programmende (M30) oder einer erneuten programmierten Änderung gültig.

`Syntax für Punkt-Vektor-Darstellung:`

#ORI MODE [VECTOR_2DOF]

VECTOR_2DOF — A, B, C sind Komponenten des Richtungsvektors. Die Adressbuchstaben A, B, C sind zwingend zu verwenden; sie haben keinen Bezug zu den konfigurierten Achsnamen in der Kanalliste. Die Vektorkomponenten müssen nicht normiert sein

#ORI MODE [VECTOR_2DOF] bewirkt die Voranwahl der Orientierungsprogrammierung. Erst bei aktiver Transformation (#TRAFO ON) werden dann Punkt-Vektordarstellungen erkannt und ausgewertet.

`Syntax umschalten auf klassische Orientierungsprogrammierung:`

#ORI MODE [ANGLE]

ANGLE — Winkelwerte über konfigurierte Achsnamen (Default)

Besonderheiten im Zusammenhang mit aktiven Koordinatensystemen (CS):

- Bei 5-Achs-RTCP-Transformationen (unvollständig) und aktivem CS erfolgt die Abbildung der Orientierung immer unabhängig von P-CHAN-00247.
- Bei vollständigen 5-Achstransformationen und aktivem CS erfolgt die Abbildung der Orientierung abhängig von P-CHAN-00247.
- Auch bei der Rohrbearbeitung ist die Vektorprogrammierung erlaubt. Die virtuelle Achsprogrammierung darf nicht aktiv sein (siehe [FCT-M5, Kin-ID 90 – HD14 = 0])

Alternativ kann die Punkt-Vektorprogrammierung mit P-CHAN-00177 vorkonfiguriert werden. Mit *ori.mode* bestimmt der Anwender, ob im NC-Programm die mit A,B,C programmierten Werte bei aktiver kinematischer Transformation in Ihrer Bedeutung als normale Koordinaten bzw. Winkelwerte eingelesen werden, oder als entsprechende Vektorkomponenten zu interpretieren sind.

Folgende Kennungen sind alternativ zu konfigurieren:

ori.mode ANGLE Winkelwerte über konfigurierte Achsnamen (Default)

ori.mode VECTOR_2DOF Vektorkomponenten über A, B, C

Wenn *ori.mode* nicht belegt wurde, dann ist die Standardeinstellung für die Orientierungsprogrammierung aktiv (Orientierungsvorgabe über Drehwinkel).

Programmierbeispiel

Umschalten der Orientierungsprogrammierung auf Punkt-Vektordarstellung

```
%example_1
:
#KIN ID [9]
:
#ORI MODE [VECTOR_2DOF]
#TRAFO ON
G01 F1000
X79.993 Y57.197 Z-39.993 A0.67520 B0.29702 C-0.67520
X79.973 Y57.392 Z-39.973 A0.66945 B0.32198 C-0.66945
X79.941 Y57.586 Z-39.941 A0.66316 B0.34705 C-0.66316
:
X79.255 Y58.978 Z-39.255 A0.58988 B0.55144 C-0.58988
X79.121 Y59.121 Z-39.121 A0.57735 B0.57735 C-0.57735
X78.903 Y59.319 Z-38.903 A0.55691 B0.61620 C-0.55691
X78.666 Y59.493 Z-38.666 A0.53439 B0.65487 C-0.53439
X78.414 Y59.643 Z-38.414 A0.50964 B0.69321 C-0.50964
X75.000 Z-35.000 A0.00000 B1.00000 C0.00000
:
#TRAFO OFF
M30
```

Programmierbeispiel

Umschalten zwischen Punkt-Vektor und Punkt-Winkeldarstellung

```
%example_2
:
#KIN ID [9]
:
#ORI MODE [VECTOR_2DOF]
#TRAFO ON
G01 F1000
X79.993 Y57.197 Z-39.993 A0.67520 B0.29702 C-0.67520
X79.973 Y57.392 Z-39.973 A0.66945 B0.32198 C-0.66945
X79.941 Y57.586 Z-39.941 A0.66316 B0.34705 C-0.66316
...
X79.255 Y58.978 Z-39.255 A0.58988 B0.55144 C-0.58988
X79.121 Y59.121 Z-39.121 A0.57735 B0.57735 C-0.57735
X78.903 Y59.319 Z-38.903 A0.55691 B0.61620 C-0.55691
X78.666 Y59.493 Z-38.666 A0.53439 B0.65487 C-0.53439
X78.414 Y59.643 Z-38.414 A0.50964 B0.69321 C-0.50964
X75.000 Z-35.000 A0.00000 B1.00000 C0.00000
:
#TRAFO OFF
:
#ORI MODE [ANGLE]
#TRAFO ON
G01 F1000
X10 Y10 Z10 C90 A15
X20 Y10 Z10 C90 A30
:
#TRAFO OFF
M30
```

17.10.2 Programmierung und Konfiguration für 6-Achskinematiken (Roboter) (#ORI MODE)

Die Auswertung der Punkt-Vektorprogrammierung nach Trafoanwahl wird durch den NC-Befehl #ORI MODE[..]aktiviert. Für die klassische Punkt-Vektor-Darstellung wird der Mode VECTOR_ABC oder VECTOR_IJK gesetzt. Er bleibt bis zum Programmende (M30) oder einer erneuten programmierten Änderung gültig. Das Verhalten der feststehenden Drehachse wird über 2 zusätzliche Schlüsselworte festgelegt.

Syntax für Punkt-Vektor-Darstellung:

#ORI MODE [VECTOR_ABC | VECTOR_IJK FIXED_AX_IDX=..]

oder

#ORI MODE [VECTOR_ABC | VECTOR_IJK TOOL_AX_IN_PLANE=..]

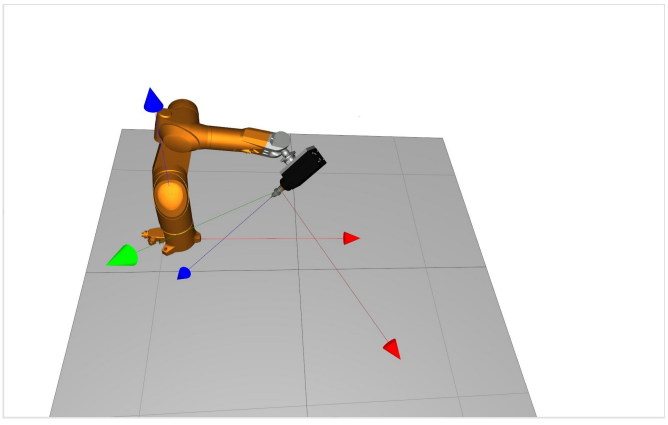
VECTOR_ABC	A, B, C sind Komponenten des Richtungsvektors. Die Adressbuchstaben A, B, C sind zwingend zu verwenden; - sie haben keinen Bezug zu den konfigurierten Achsnamen in der Kanalliste. Die Vektorkomponenten müssen nicht normiert sein.
VECTOR_IJK	I, J, K sind Komponenten des Richtungsvektors. Die Adressbuchstaben I, J, K sind zwingend zu verwenden. Die Vektorkomponenten müssen nicht normiert sein. Die gleichzeitige Kreisprogrammierung nach DIN 66025 über I, J, K ist nicht zulässig.

Aus den 3 Vektorkomponenten ergeben sich steuerungsintern für 2 Achsen Drehwinkel zur Werkzeugorientierung. Die Winkelstellung der dritten Drehachse ergibt sich zum Anwahlzeitpunkt der kinematischen Transformation aus den Gelenkwinkelstellungen und bleibt während der Vektorprogrammierung unverändert.

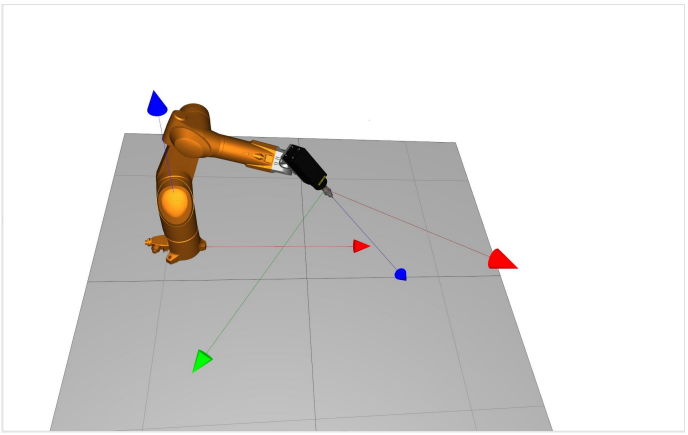
Der Achsindex der Drehachse, die gemäß Eulerkonvention nicht an der Orientierung beteiligt ist, ergibt sich aus der Betrachtung der Reihenfolge der Achsen, über die die Position und Handorientierung des Roboters definiert wird (siehe hierzu Beschreibung P-CHAN-00178).

FIXED_AX_IDX=..	Achsindex der feststehenden Drehachse.

Beispiel: Drehachse C Winkelstellung 45° bei Anwahl, FIXED_AX_IDX = 5



Startorientierung des Roboters

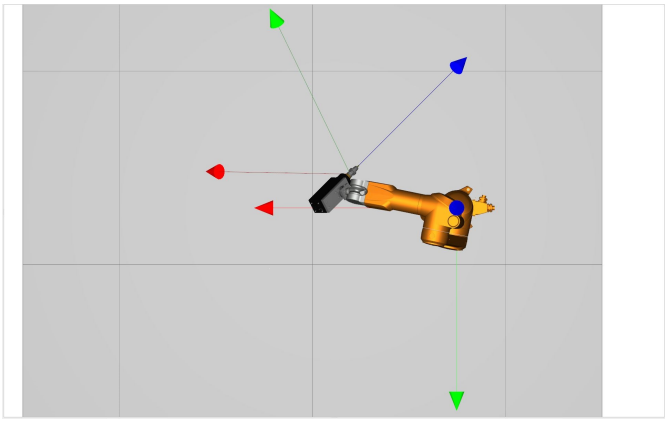


Orientierung im Zielpunkt

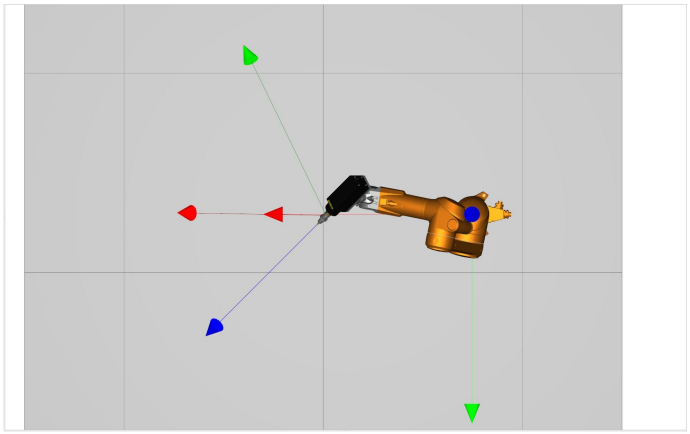
Alternativ zur fixen Drehachse kann auch die Ebene (YZ, ZX) festgelegt werden, in der entweder die Z oder Y-Werkzeugachse liegt. Der dritte Winkel wird dann so bestimmt, dass die gewählte Werkzeugachse im Zielpunkt parallel zur festgelegten Ebene liegt (siehe hierzu Beschreibung P-CHAN-00436).

TOOL_AX_IN_PLANE=.. Ebene, zu der eine Werkzeugachse parallel liegt.

Beispiel 1: Werkzeugachse Z (rot) parallel zur Basisebene ZX, TOOL_AX_IN_PLANE = 1

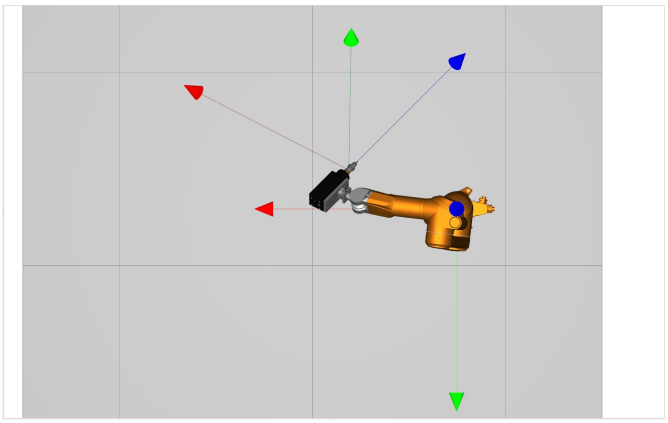


Basisebene ZX: Startorientierung

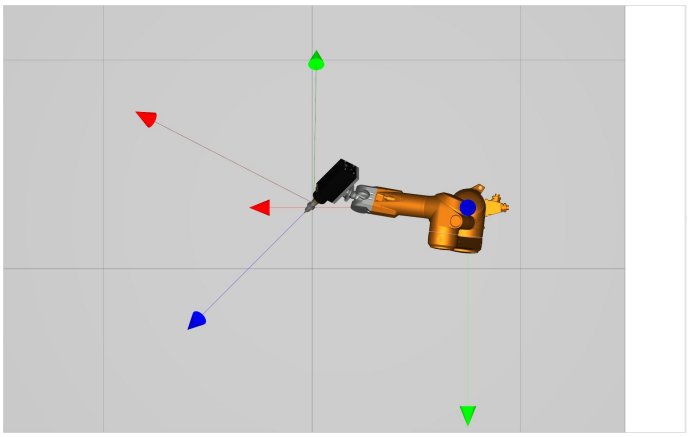


Orientierung im Zielpunkt

Beispiel 2: Werkzeugachse Y (grün) parallel zur Basisebene YZ, TOOL_AX_IN_PLANE = 2



Basisebene YZ: Startorientierung



Orientierung im Zielpunkt

#ORI MODE [VECTOR_...] bewirkt die Voranwahl der Orientierungsprogrammierung. Erst bei aktiver Transformation (#TRAFO ON) werden dann Punkt-Vektordarstellungen erkannt und ausgewertet

`Syntax umschalten auf klassische Orientierungsprogrammierung:`

#ORI MODE [ANGLE]

ANGLE Winkelwerte über konfigurierte Achsnamen (Standard).

Besonderheiten im Zusammenhang mit aktiven Koordinatensystemen (CS):

- Bei vollständigen 6-Achstransformationen und aktivem CS erfolgt die Abbildung der Orientierung immer unabhängig von P-CHAN-00247.

Alternativ kann die Punkt-Vektorprogrammierung mit P-CHAN-00177 vorkonfiguriert werden. Mit *ori.mode* bestimmt der Anwender, ob im NC-Programm die mit A, B, C oder I, J, K programmierten Werte bei aktiver kinematischer Transformation in ihrer Bedeutung als normale Koordinaten bzw. Winkelwerte eingelesen werden, oder als entsprechende Vektorkomponenten zu interpretieren sind.

Folgende Kennungen sind alternativ zu konfigurieren:

ori.mode	ANGLE	Winkelwerte über konfigurierte Achsnamen (Standard)
ori.mode	VECTOR_ABC	Vektorkomponenten über A, B, C
ori.mode	VECTOR_IJK	Vektorkomponenten über I, J, K

Wenn *ori.mode* nicht belegt wurde, dann ist die Standardeinstellung für die Orientierungs-programmierung aktiv (Orientierungsvorgabe über Drehwinkel).

Der Achsindex der feststehenden Drehachse wird in P-CHAN-00178 angegeben:

ori.fixed_axis_index *<idx>* Achsindex der feststehenden Drehachse

Die Ebene, zu der eine Werkzeugachse parallel liegt, wird im Kanalparameter P-CHAN-00436 angegeben:

ori.tool_ax_in_plane *<id>* Ebene, zu der die Werkzeugachse parallel liegt

Die Angaben von feststehender Drehachse P-CHAN-00178 und der Ebene der Werkzeugachse P-CHAN-00436 sind zueinander exklusiv. Wenn beide Parameter belegt sind, erfolgt im Hochlauf der Steuerung die Ausgabe der Meldung ID 22027 und beide Werte werden zu Null korrigiert.

Programmierbeispiel

Umschalten der Orientierungsprogrammierung auf Punkt-Vektordarstellung (ABC) und Angabe der feststehenden Drehachse

```
%example_1
;…
#KIN ID [45]
;…
#ORI MODE [VECTOR_ABC FIXED_AX_IDX=5]
#TRAFO ON
X50       Y50                A50      B0 C0
X75       Y150      Z180     A0       B0 C1
X149.316  Y150      Z180     A-0.0457 B0 C0.999
X149.316  Y150      Z165.012 A-0.0457 B0 C0.999
X150.0018 Y150      Z150.0279 A-0.0457 B0 C0.999
X162.1716 Y150      Z150.0621 A0.0349  B0 C0.9994
X172.1268 Y149.9997 Z149.3631 A0.1013  B0 C0.9949
X178.7241 Y149.9997 Z148.5459 A0.1454  B0 C0.9894
X188.532  Y149.9997 Z146.7645 A0.2111  B0 C0.9775
X198.2064 Y149.9997 Z144.3474 A0.2758  B0 C0.9612
X207.7002 Y149.9994 Z141.2733 A0.3393  B0 C0.9407
X216.978  Y149.9994 Z137.5713 A0.4012  B0 C0.916
;…
X150      Y150      Z180     A0.6111  B0.0014 C0.7916
X150      Y150      Z180     A0.0631  B0.0001 C0.998
X150      Y150      Z180     A0       B0      C1
;…
#TRAFO OFF
M30
```

Programmierbeispiel

Umschalten der Orientierungsprogrammierung auf Punkt-Vektordarstellung (IJK) und Angabe der Ebene, zu der die Werkzeugachse parallel liegt

```
%example_2
;…
#KIN ID [45]
;…
#ORI MODE [VECTOR_IJK TOOL_AX_IN_PLANE=1]
#TRAFO ON
X75      Y150    Z180      I0      J0      K1
X10.874  Y0      Z-29.875  I-.099  J0      K.995
X10.846  Y.666   Z-29.872  I-.099  J-.006  K.995
X10.667  Y1.976  Z-29.854  I-.097  J-.018  K.995
X10.464  Y2.748  Z-29.792  I-.095  J-.025  K.995
X10.208  Y3.429  Z-29.668  I-.093  J-.031  K.995
X9.879   Y4.075  Z-29.46   I-.091  J-.037  K.995
X9.517   Y4.713  Z-29.296  I-.088  J-.043  K.995
X9.126   Y5.328  Z-29.166  I-.085  J-.049  K.995
X8.285   Y6.492  Z-29.086  I-.077  J-.06   K.995
X7.387   Y7.597  Z-29.317  I-.068  J-.07   K.995
X6.9     Y8.108  Z-29.472  I-.063  J-.075  K.995
X6.385   Y8.598  Z-29.664  I-.058  J-.079  K.995
X5.825   Y9.038  Z-29.8    I-.053  J-.082  K.995
X5.218   Y9.412  Z-29.841  I-.047  J-.086  K.995
X3.924   Y10.011 Z-29.852  I-.035  J-.091  K.995
X2.56    Y10.43  Z-29.849  I-.023  J-.095  K.995
X1.182   Y10.657 Z-29.835  I-.01   J-.097  K.995
X.461    Y10.682 Z-29.769  I-.004  J-.098  K.995
X-.257   Y10.636 Z-29.639  I.002   J-.098  K.995
X-1.027  Y10.509 Z-29.435  I.009   J-.097  K.995
X-1.696  Y10.366 Z-29.297  I.015   J-.096  K.995
X-3.083  Y9.956  Z-29.106  I.028   J-.093  K.995
X-4.428  Y9.482  Z-29.258  I.041   J-.088  K.995
X-5.462  Y9.007  Z-29.478  I.05    J-.083  K.995
X-6.068  Y8.681  Z-29.657  I.055   J-.08   K.995
X-6.642  Y8.299  Z-29.782  I.061   J-.076  K.995
X-7.696  Y7.337  Z-29.826  I.07    J-.067  K.995
X-8.601  Y6.233  Z-29.831  I.078   J-.057  K.995
;…
#TRAFO OFF
M30
```

17.11 Status & Turn (IS, IT)

Als Alternative zur achsspezifischen Positionierung und zur genaueren Positionsvorgabe einer #PTP Bewegung bei Industrierobotern besteht die Option, die Roboterpose zu der entsprechenden kartesischen Position mit anzugeben.

Die Roboterpose wird hierfür mit Hilfe des Status (IS) beschrieben.

Zusätzlich können die Vorzeichen der Achspositionen mit Hilfe des Turn (IT) beschrieben werden.

Die Programmierung von Turn ohne Status ist nicht zulässig.

Hinweis

Die Roboterpositionierung mit Status & Turn steht nur für den Kinematiktyp 45 zur Verfügung.

Status-Bit

Eine Übersicht der Roboterposen ist unter Posen der Kinematik des Sechsachs-Gelenkarmroboters zu finden.

Die Pose des Roboters wird in 3 Kriterien unterteilt. Trifft ein Kriterium zu, wird ein entsprechender Zahlenwert zum Status hinzugefügt.

1. Kriterium: Befindet sich die Handwurzel hinter Achse A1, wird dezimal 1 bzw. binär 1 addiert (im Bild links gelber Bereich)

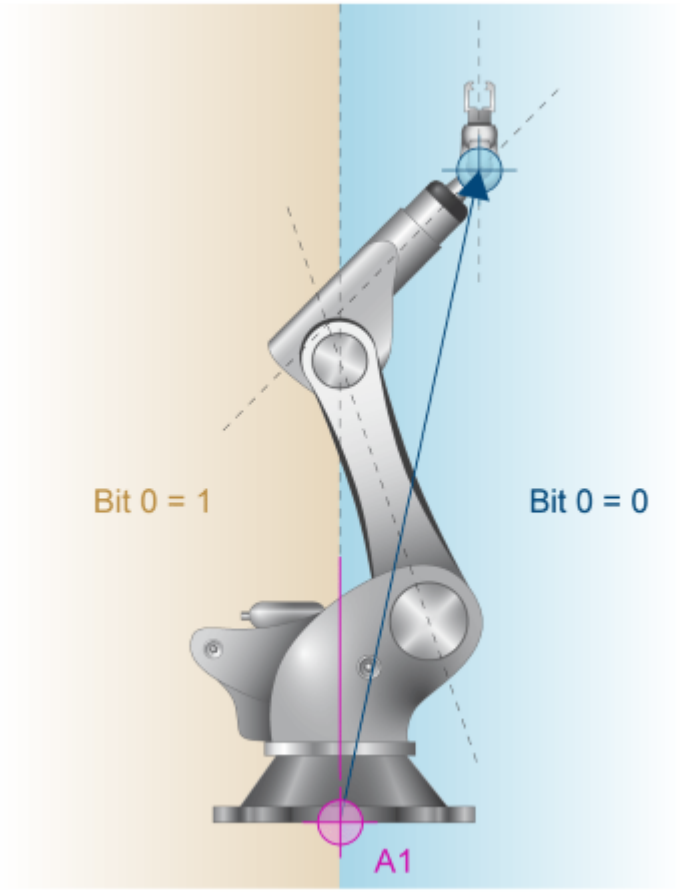


Abb. 203: Der Schnittpunkt der Handachsen (Pfeilspitze) liegt im (blauen) Grundbereich

2. Kriterium: Befindet sich die Handwurzel vor der Geraden durch die Achsen A2 und A3, wird dezimal 2 bzw. binär 10 addiert (Bild Mitte u. Rechts).

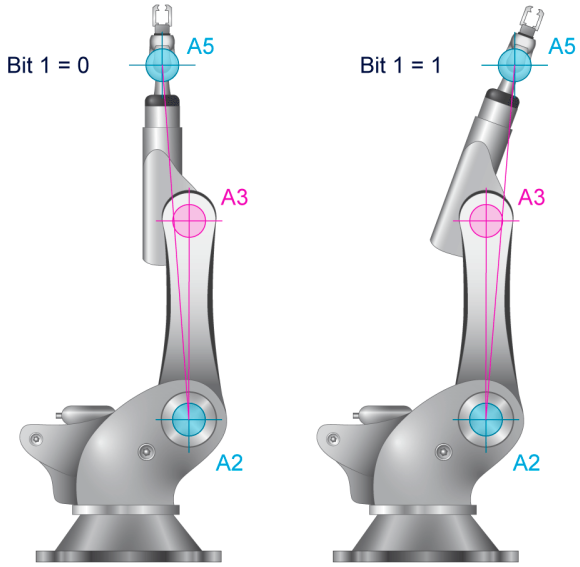


Abb. 204: Status-Bit 1 für Roboter mit einem Offset zwischen Achse A3 und Achse A5

3. Kriterium: Gibt Position von Achse A5 an. Ist A5 > 0, wird dezimal 4 bzw. binär 100 addiert.

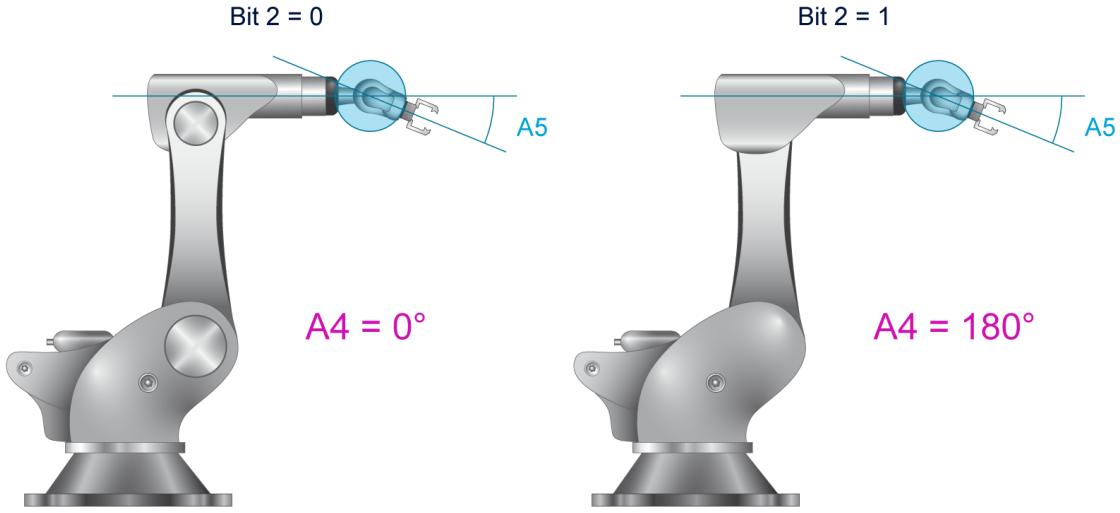


Abb. 205: Status-Bit 2 bei Achswinkelstellung A4=0° und A4=180°

Turn-Bit (optional)

Der optionale Turn-Wert listet die negativen Vorzeichen der Achswinkel auf.

Betrachtet man den Turn-Wert in einer binären Darstellung, wird das Vorzeichen jedes Achswinkels einem Bit zugeordnet. Diese werden dann zu einer Zahl, dem Turn, addiert.

Ist der Achswinkel einer Achse < 0° ist der Wert 1.

	A6 < 0°	A5 < 0°	A4 < 0°	A3 < 0°	A2 < 0°	A1 < 0°
Binär	100000	010000	001000	000100	000010	000001
Dezimal	32	16	8	4	2	1

Sind so alle 6 Achswinkel im negativen Bereich, ergibt sich ein Turn-Wert von dezimal 63 bzw. binär 111111, entsprechend bei 6 positiven Achswinkeln dezimal 0 und binär 000000.

Hinweis

Der Turn-Wert muss mit dem Status-Wert konsistent sein.

Beispielsweise beschreibt eine 1 im Status-Bit 2 einen negativen A5-Winkel. Wird zusätzlich das A5-Turn-Bit auf 0 (positiver A5-Winkel) gesetzt, widersprechen sich der Status- und der Turn-Wert. In solch einem Fall wird eine Fehlermeldung ausgegeben.

Beschreibung

Zur eindeutigen Programmierung der Roboterposen (Kin_Typ_45) mit kartesischen Zielkoordinaten einer PTP-Bewegung stehen die zusätzlichen Parameter Status & Turn zur Verfügung.

```
Syntax Programmierung Status & Turn mit den Prefixen „IS“ und „IT“ (optional):
#PTP ON
G.. X.. Y.. … IS.. IT..
#PTP OFF
```

Binärzahlen können mit der folgenden Syntax programmiert werden:

'B<0…1>', oder '2#<0…1>', bzw. '02#<0…1>'.

Bei Verwendung von Binärzahlen ergibt sich somit die folgende Syntax:

Status: **IS**'Bxxx'

Turn: **IT**'Bxxxxxx'

Bei Verwendung von Dezimalzahlen ergibt sich die folgende Syntax:

Status: **IS**<*expr*>

Turn: **IT**<*expr*>

Anzeigewerte

Folgende CNC-Objekte stehen für diese Funktionalität zur Verfügung:

- mc_st_valid_r: Gültigkeit des Status & Turn-Werts
  (Task COM- Indexgruppe 0x12010<C_{ID}> Indexoffset 0xB1)
- mc_st_status_r: Status-Wert der Kinematik 45
  (Task COM- Indexgruppe 0x12010<C_{ID}> Indexoffset 0xB2)
- mc_st_turn_r: Turn-Wert der Kinematik 45
  (Task COM- Indexgruppe 0x12010<C_{ID}> Indexoffset 0xB3)

Hinweis

Sind Status & Turn nicht programmiert, wird der Zielpunkt auf Achswinkelebene per "Shortest Way-Strategie" ermittelt.

Programmierbeispiel

Status & Turn mit Binärzahlen programmiert

```
N010 #PTP ON
N020 G01 X1100 Y0 Z1400 A0 B90 C0 IS'B010' IT'B000010' F5000
N030 G01 X1200 ;Zielpunkt wird per Shortest Way ermittelt
N040 #PTP OFF
```

Programmierbeispiel

Status & Turn mit Dezimalzahlen programmiert

```
N010 #PTP ON
N020 G01 X1000 Y0 Z1400 A0 B90 C0 IS2 IT2 F5000
N030 #PTP OFF
N040 G01 X1500
N050 G01 Y1000
N060 G01 X-1000
N070 #PTP ON
N080 G01 X-1000 Y-1000 ;Zielpunkt wird per Shortest Way ermittelt
N090 #PTP OFF
```

17.12 Mehrstufige Transformationen

17.12.1 Parametrierung

Für die Definition einer Transformation müssen in den Kanalparametern spezifische Kinematikdaten gesetzt werden. Folgendes Beispiel zeigt für eine **einfache** Transformation mit der ID 87 die Belegung der Kinematikstruktur mit Daten:

```
trafo[0].id                                                    87
trafo[0].param[0]                                          300000
trafo[0].param[1]                                              0
trafo[0].param[2]                                              0
```

Kinematiken, die aus verschiedenen Transformationsstufen (s.g. **mehrstufigen** Transformationen) bestehen, werden in spezifischen Datenstrukturen (*kin_step[..]*) definiert:

Hinweis

Momentan dürfen mehrstufige Transformationen maximal aus 2 Stufen (*kin_step[0]* und *kin_step[1]*) bestehen. Für jede Transformationsstufe können maximal 10 verschiedene Transformationen (*trafo[0]* bis *trafo[9]* konfiguriert werden.

Für die erste Transformationsstufe müssen die Parameter in der Kinematikstruktur kin_step[0] gesetzt werden. Das folgende Beispiel zeigt die Definition der ersten Kinematik mit der ID 65.

```
kin_step[0].trafo[0].id                                        65
kin_step[0].trafo[0].param[0]                              500000
kin_step[0].trafo[0].param[1]                                  0
kin_step[0].trafo[0].param[2]                                  0
...
```

Die Definition der ersten Stufe (*kin_step[0].trafo[0]*) ist vollständig äquivalent zur Definition einer einfachen Transformation (trafo[0]). Die verschiedenen Stufen werden gemäß der Richtung der Vorwärtstransformation angelegt.

Für die zweite Transformationsstufe müssen die Parameter in der Kinematikstruktur kin_step[1] gesetzt werden. Das folgende Beispiel zeigt die Definition einer zweiten Kinematik mit der ID 51.

```
kin_step[1].trafo[0].id                                        51
kin_step[1].trafo[0].param[0]                              200000
kin_step[1].trafo[0].param[1]                               50000
kin_step[1].trafo[0].param[2]                                  0
...
```

Zusätzlich kann in der Kanalliste eine Standard-Kinematik-ID festgelegt werden. Das folgende Beispiel zeigt die Definition einer Standard-ID 87 für eine **einfache** Transformation:

```
kinematic_id                                                   87
```

Die Standard-Kinematik- IDs für **mehrstufige** Transformationen müssen in spezifischen Parametern definiert werden. Auch hier ist die Definition der ersten Standard-ID (*default_id_of_kin_step[0]*) vollständig äquivalent zur Definition einer einfachen Standard-ID (*kinematic_id*).

Das folgende Beispiel zeigt die Definition einer Standard-ID 65 für die erste Transformationsstufe (step 0) und die Standard-ID 51 für die zweite Transformationsstufe (step 1).

```
default_id_of_kin_step[0]                                      65
default_id_of_kin_step[1]                                      51
```

17.12.2 Getrennte Voranwahl und Aktivierung

Bei mehrstufigen Transformationen erfolgt die Voranwahl der Kinematik-ID's mit einem erweiterten #KIN ID-Befehl:

Syntax:

#KIN ID [*<kin_id_first_step>* **,** *<kin_id_second_step>* **]**

<kin_id_first_step>	Setzen der Nummer der Kinematik-ID der ersten Transformationsstufe
<kin_id_second_step>	Setzen der Nummer der Kinematik-ID der zweiten Transformationsstufe

Anstelle der Kinematik-ID's sind auch folgende Schlüsselworte möglich:

NONE	Abwahl der Kinematik-ID
DEFAULT	Setzen der Standard-Kinematik-ID aus der Kanalliste

Hinweis

Die An/Abwahl der Transformation wird durch den Standardbefehl #TRAFO ON/OFF geschaltet.

Beispiel

```
Nxx #KIN ID [86,1]      ;Voranwahl Kinematik 86 und 1
Nxx #KIN ID [86,NONE]   ;Voranwahl Kinematik 86 und Abwahl 1
;..oder nur erste Stufe adressieren wählt immer zweite Stufe ab
Nxx #KIN ID [86]        ;Voranwahl Kinematik 86 und Abwahl 1

Nxx #KIN ID [NONE,3]      ;Abwahl Kinematik 86 und Anwahl 3
Nxx #KIN ID [NONE,NONE]   ;Abwahl beider Kinematiken
;..oder nur erste Stufe adressieren wählt immer zweite Stufe ab
Nxx #KIN ID [NONE]        ;Abwahl beider Kinematiken

Nxx #KIN ID [DEFAULT,1]          ;Voranwahl Standardkinematik und 1
Nxx #KIN ID [DEFAULT, DEFAULT]   ;Voranwahl Standardkinematiken für
                                 ;beide Stufen
Nxx #KIN ID                      ;Voranwahl Standardkinematiken für
                                 ;beide Stufen

:
Nxx #TRAFO ON  ;Kinematische Transformationen gemäß Voranwahl aktivieren
               ;verursacht Fehler, wenn beide Kinematiken NONE
:
Nxx #TRAFO OFF ;Kinematische Transformationen deaktivieren
```

17.12.3 Kombinierte Voranwahl und Aktivierung

Zur Vereinfachung der Programmierung kann die Voranwahl sowie die Aktivierung/ Deaktivierung der mehrstufigen Transformationen mit einem erweiterten #TRAFO-Befehl erfolgen:

Syntax:

#TRAFO [*<kin_id_first_step>* **,** *<kin_id_second_step>* **]**

<kin_id_first_step>	Setzen der Nummer der Kinematik-ID der ersten Transformationsstufe, Aktivierung der zugehörigen Transformation
<kin_id_second_step>	Setzen der Nummer der Kinematik-ID der zweiten Transformationsstufe, Aktivierung der zugehörigen Transformation

Anstelle der Kinematik-ID's sind auch folgende Schlüsselworte möglich:

OFF	Transformationsstufe deaktivieren, zugehörige Kinematik-ID bleibt gesetzt.
ON	Transformationsstufe wird basierend auf der zugehörigen Kinematik-ID aktiviert.
NONE	Transformationsstufe deaktivieren, zugehörige Kinematik-ID abwählen.
DEFAULT	Setzen der Standard-Kinematik-ID aus der Kanalliste, Aktivierung der zugehörigen Transformation

Hinweis

Mit der Anwahl einer Kinematik-ID werden mit diesem Befehl implizit immer die zugehörigen Transformationsstufen aktiviert.

Hinweis

Wird die Kinematik-ID mit NONE abgewählt, so wird auch die Anzeige für diese Transformationsstufe abgeschaltet.

Wird die Kinematik-ID mit OFF abgewählt, so bleibt die Anzeige für diese Transformationsstufe eingeschaltet.

Die erweiterte Syntax kann auch auf **einfache** Transformationen angewendet werden. Voranwahl und Aktivierung bzw. Deaktivierung der Transformation erfolgen im gleichen NC-Befehl:

```
Nxx #TRAFO [87]   ;Voranwahl Kinematik 87 und Aktivierung Trafo
Nxx #TRAFO [OFF]  ;Abwahl und Deaktivierung Trafo 87
Nxx #TRAFO [ON]   ;Erneute Aktivierung Trafo 87
```

Wenn bei mehrstufigen Transformationen die zweite ID nicht programmiert ist, dann wird diese komplett deaktiviert bzw. implizit wie ein NONE behandelt.

```
Nxx #TRAFO [65,51] ;Voranwahl Kinematik 65+51 und Aktivierung Trafos
Nxx #TRAFO [65]    ;Voranwahl Kinematik 65, Abwahl 2. Kinematik+Trafo
```

Beispiel

```
Nxx #TRAFO [65,51]  ;Aktivierung Trafo 1(65), Trafo 2(51)
Nxx #TRAFO [86,OFF] ;Aktivierung Trafo 1(65), Deaktivierung Trafo 2
;..oder nur erste Trafo adressieren
Nxx #TRAFO [65]     ;Aktivierung Trafo 1(65), Deaktivierung Trafo 2

Nxx #TRAFO [OFF,OFF]  ;Deaktivierung Trafo 1 und Trafo 2
Nxx #TRAFO [ON,ON]    ;Aktivierung Trafo 1 und Trafo 2
Nxx #TRAFO [ON,OFF]   ;Aktivierung Trafo 1, Deaktivierung Trafo 2

Nxx #TRAFO [NONE,NONE] ;Beide Trafos deaktivieren und beide
                       ;Kinematik-ID's abwählen
;..oder nur erste Trafo adressieren
Nxx #TRAFO [NONE]      ;Beide Trafos deaktivieren und beide
                       ;Kinematik-ID's abwählen

Nxx #TRAFO [DEFAULT,51]       ;Standard Trafo 1 und Trafo 2 aktivieren
Nxx #TRAFO [DEFAULT, DEFAULT] ;Beide Standardtrafos aktivieren

Nxx #TRAFO [65, DEFAULT]   ;Aktivierung Trafo 1 und Standardtrafo 2
Nxx #TRAFO [DEFAULT, NONE] ;Aktivierung Standardtrafo 1,
                           ;Deaktivierung Trafo 2
                           ;und Abwahl Kinematik-ID Stufe 2
```

Die Programmierung in Kombination mit der Standardsyntax #TRAFO ON/OFF ist möglich, folgendes muss jedoch beachtet werden:

Der Befehl

```
Nxx #TRAFO OFF oder #TRAFO[OFF,OFF]
```

deaktiviert beide Transformationen, aber die Voranwahl der Kinematik-ID's bleibt erhalten. Ein erneutes #TRAFO ON oder #TRAFO[ON, ON] aktiviert erneut beide Transformationen

Der Befehl

```
Nxx #TRAFO[NONE,NONE]
```

Ist dazu nicht equivalent, weil durch diesen Befehl die Voranwahl der Kinematik-ID's abgewählt wurde. Ein nachfolgender #TRAFO ON hat keine Wirkung.

17.12.4 Zusammenfassung

- #TRAFO[..] ändert gleichzeitig die Voranwahl von Kinematiken gesetzt durch #KIN ID[..]
- #TRAFO[..] aktiviert analog zu #TRAFO ON kinematische Transformationen
- Wenn die zweite ID nicht programmiert ist (z.B. #TRAFO[65] oder #KIN ID[65]), dann wird diese zweite ID implizit mit NONE belegt
- Kinematik-ID’s können nicht geändert werden, wenn mindestens eine kinematische Transformation noch aktiv ist
- #KIN ID, #TRAFO ON/OFF und #TRAFO[..] können unter Beachtung der Regeln kombiniert programmiert werden
- Sind beide ID's NONE, verursacht #TRAFO ON einen Fehler

18 Programmieren von Moduloachsen

Der Standardmodus der Moduloprogrammierung unterstützt durch die Programmierung von 2 Vorzeichen die spezifische Festlegung von Drehrichtung und Position sowie die Begrenzung auf maximal eine Umdrehung bei Absolutmaßangabe.

Syntax:

<Achsname> [**+** | **-**] *<pos>*

<Achsname>	Bezeichnung der Moduloachse. Lange Achsbezeichnungen werden nicht unterstützt (z.B. "C_MODULO").
+ | -	Das 1. Vorzeichen nach dem Achsnamen bestimmt immer die Drehrichtung: - bedeutet Drehung im Uhrzeigersinn + bedeutet Drehung gegen den Uhrzeigersinn kein Vorzeichen bedeutet Drehung auf kürzestem Weg (Optimiertes Richten)
<pos>	Achsposition in [°]. Bei Absolutmaßangabe kann die Positionsangabe mit einem weiteren Vorzeichen behaftet sein. Die Zugehörigkeit zur Positionsangabe kann für optimiertes Richten durch Klammerung [..] erzwungen werden.

Achtung

Die Programmierung von 2 Vorzeichen (Richtung und Position) ist nur zulässig, wenn die Achse vom Achsmode "Modulo" ist (P-AXIS-00015). Die Positionierung auf kürzestem Weg erfolgt immer dann, wenn kein Vorzeichen direkt nach dem Achsnamen programmiert wurde.

Zusätzlich besteht die Möglichkeit auf einen Modus zu wechseln, bei dem generell immer auf kürzestem Weg positioniert wird (Kapitel Positionierung auf kürzestem Weg [▶ 814]). Die Programmierung von 2 Vorzeichen ist in diesem Modus ebenfalls zulässig. Die Auswertung erfolgt jedoch gemäß folgender Regel:

-- => + (Minus Minus ist Plus)

+- => - (Plus Minus ist Minus)

-+ => - (Minus Plus ist Minus)

++ => + (Plus Plus ist Plus)

Programmierung in Absolutmaßangabe (G90)

- Der mit der Achse programmierte Wert (Zielposition) wird in den Modulobereich verschoben. Es kann also maximal eine Umdrehung gefahren werden.
- Der Wert kann ein numerischer Ausdruck sein wie z.B. [3*2+5] , P1 , [P1+P2-3] , [-30].
- Das erste Vorzeichen nach dem Achsnamen bestimmt immer die Drehrichtung, und jedes weitere Vorzeichen wird als zur (absoluten) Positionsangabe gehörig interpretiert.

Beispiel (Annahme: 360° Modulo)

```
G90 G1 C+560 <=> G90 G1 C+200 (Bewegung zu Position 200 in + Richtung)
G90 G1 C-P1 (Gehe zu Position P1 (Modulo implizit) in - Richtung)
```

- Wenn die programmierte Position = aktuelle Position, erfolgt keine Bewegung.
- Der Verfahrweg einer Moduloachse wird nicht durch Softwareendschalter begrenzt.

Programmierung in Relativmaßangabe(G91)

- Der mit der Achse programmierte Wert beschreibt den zu fahrenden Weg im Bezug zur vorhergehenden Position. Das erste Vorzeichen nach dem Achsname bestimmt immer die Drehrichtung. Weitere Vorzeichen sind bei relativer Wegangabe nicht zulässig.

Beispiel (Annahme: 360° Modulo)

```
G91 G1 C+560 (Bewegung zu "aktuelle Position plus 560" in + Richtung)
```

- Wenn der Wert größer als der Modulowert ist, wird die Anzahl der Umdrehungen berücksichtigt. Es können also mehrere Umdrehungen gefahren werden.

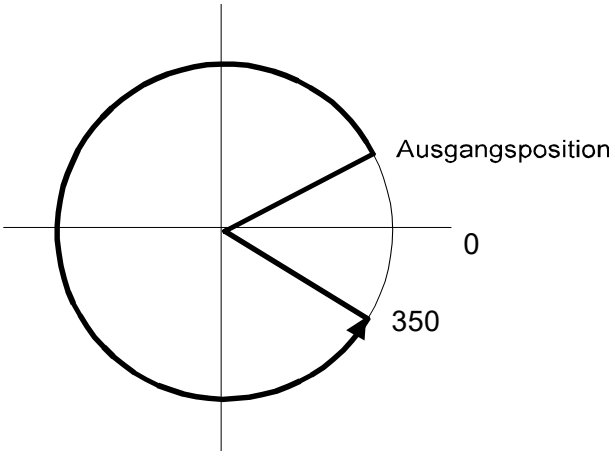
Folgende V.A.-Variablen ermöglichen den Lesezugriff auf die aktuellen achsspezifischen Moduloeinstellungen:

V.A.MODE[i]	liefert den Achsmode gemäß der Achstabelle. z.B. 4 wenn die Achse vom Typ modulo ist. -> wird gebraucht zum Lesen von Moduloachsen
V.A.MODULO_VALUE[i]	wird genutzt zum Lesen des Modulobereiches z.B. 360 bei einem Modulobereich von 0-360°. (bei "Nichtmoduloachsen" ist dieser Wert nicht relevant).

Programmierbeispiel

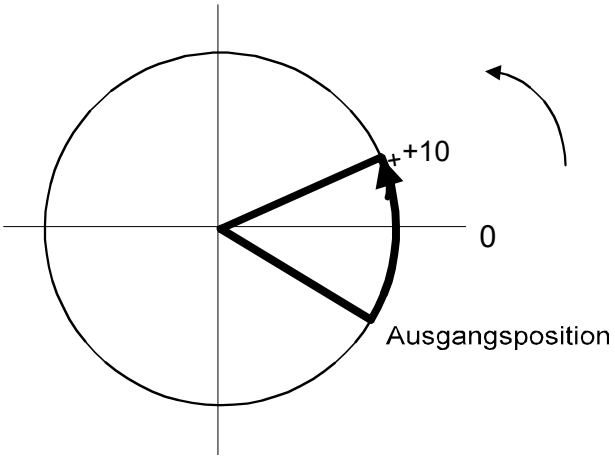
Programmierbeispiele für Moduloprogrammierung mit Absolutmaßangabe

G90 G1 C+350 <=> Gehe zu Position 350 in + Richtung



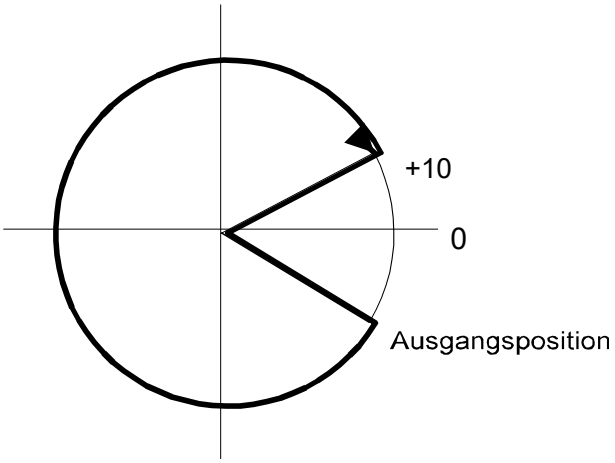
P1 =+10

G90 G1 C+P1 <=> G1 C+10 <=> Gehe zu Position 10 in + Richtung

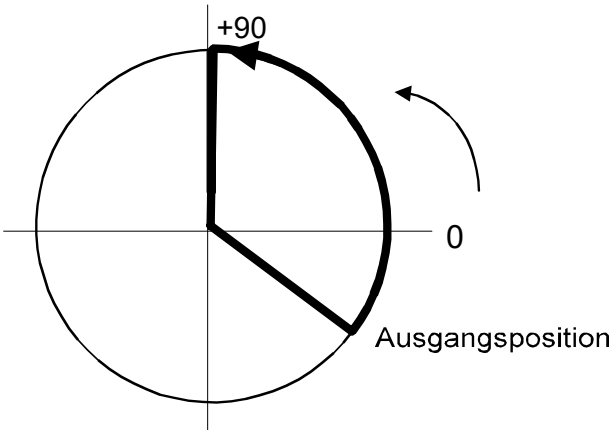


P1= -350

G90 G1 C-P1 <=> G1 C-[-350] <=> G1 C-[10] <=> Gehe zu Position 10 in - Richtung



G90 G1 C+450 <=> G1 C+[450 mod 360] <=> Gehe zu Position 90 in + Richtung



Programmierbeispiel

Beispiele für korrekte Programmierung:

```
C+200      Drehe in positiver Richtung auf Position 200

C-200      Drehe in negativer Richtung auf Position 200

C+-200     Drehe in positiver Richtung auf Position -200 (= +160)

C+[-200]   Drehe in positiver Richtung auf Position -200 (= +160)

C—200      Drehe in negativer Richtung auf Position -200 (= +160)

C-[-200]   Drehe in negativer Richtung auf Position -200 (= +160)

C200       Drehe auf kürzestem Weg auf Position 200

C[+200]    Drehe auf kürzestem Weg auf Position 200

C[-200]    Drehe auf kürzestem Weg auf Position -200
```

Programmierbeispiel

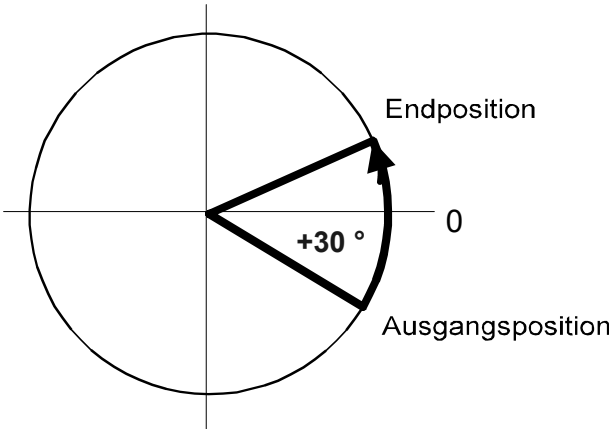
Beispiele für falsche Programmierung

Keine, da das erste Vorzeichen nach dem Achsnamen immer die Drehrichtung bestimmt und jedes weitere Vorzeichen als zur Positionsangabe gehörig interpretiert wird.

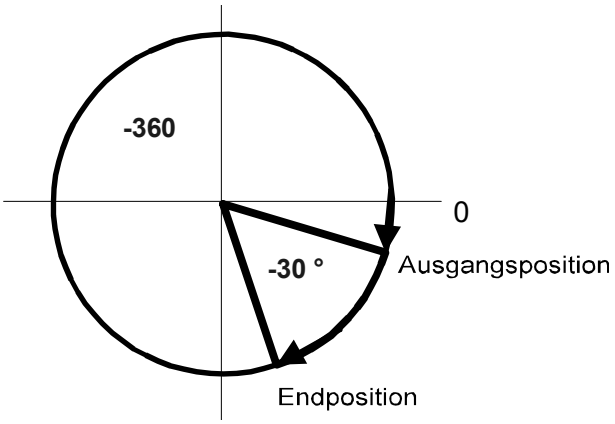
Programmierbeispiel

Programmierbeispiele für Moduloprogrammierung mit Relativmaßangabe

G91 G1 C+30



G91 G1 C-30



Programmierbeispiel

Beispiele für korrekte Programmierung

```
C+200   Drehe in positiver Richtung auf "aktuelle Position plus 200"

C-200   Drehe in negativer Richtung auf "aktuelle Position minus 200"

C200    Drehe in positiver Richtung auf "aktuelle Position plus 200"
```

Programmierbeispiel

Beispiele für falsche Programmierung

```
C+-200    Fehler: Negative Wegangabe bei rel. Programmierung unzulässig.

C—200     Fehler: Negative Wegangabe bei rel. Programmierung unzulässig.

P1=-1
C-[P1]    Fehler: Negative Wegangabe bei rel. Programmierung unzulässig.
```

18.1 Positionierung auf kürzestem Weg

Dieser Modus ermöglicht die Positionierung der Modulorundachsen immer auf dem kürzesten Weg. Auf eine Vorzeichenprogrammierung kann somit verzichtet werden.

Der Modus wird durch den Parameter P-CHAN-00346 aktiviert. Wird dennoch eine Vorgabe der Drehrichtung gewünscht, muss in diesem Modus zwingend mit Relativprogrammierung gearbeitet werden.

Programmierbeispiel

Programmierbeispiele für Moduloprogrammierung mit Absolutmaßangabe

Annahme: Modulobereich 0-360°, G90, momentane Achsposition 0°

```
C200    Drehe auf kürzestem Weg (-160) auf Position 200

C+200   Drehe auf kürzestem Weg (-160) auf Position 200

C-200   Drehe auf kürzestem Weg (+160) auf Position -200 (= +160)
```

Programmierbeispiel

Programmierbeispiele für Moduloprogrammierung mit Relativmaßangabe

Annahme: Modulobereich 0-360°, G91, momentane Achsposition +60°

```
C+200   Drehe in positiver Richtung auf Position 260 (60+200)

C-200   Drehe in negativer Richtung auf Position 220 (60-200)
```

19 Erweiterte Werkzeugprogrammierung

19.1 Beschreibung der Funktion

19.1.1 Werkzeug-ID

Im Rahmen der erweiterten Werkzeugprogrammierung werden dem Aufgabenbereich Werkzeugmanagement (WZM) neue, werkzeugspezifische Kommunikationsobjekte durch die CNC zur Verfügung gestellt. Es handelt sich zum einen um die erweiterte Bezeichnung für Werkzeuge (WZ) und zum anderen um Standgrößen von Werkzeugen.

In der Standardprogrammierung werden WZ im NC-Programm durch einteilige Nummern identifiziert. Entsprechend der DIN 66025 wird diese numerische ID mit dem D-Wort zum Einrechnen neuer Daten (rechnerischer WZ-Wechsel) programmiert. In Verbindung mit dem T-Sprachbefehl legt die ID das nächste WZ fest, das physikalisch eingewechselt werden soll.

Zum Einrechnen neuer Daten sind diese von der heutigen externen WZ-Verwaltung anzufordern. Die WZ-Verwaltung besitzt spezielle, herstellerspezifische Algorithmen, um aus der WZ-ID das einzuwechselnde WZ zu bestimmen. Dabei ist zu berücksichtigen, dass mit der übermittelten WZ-ID nur der WZ-Typ festgelegt wird und im WZ-Magazin evtl. mehrere WZ gleichen Typs (Schwester-WZ) einsatzbereit sind. Es lassen sich also im NC-Programm die WZ-Exemplare nicht eindeutig identifizieren.

Die T-Nummer dient in erster Linie als Techno-Information, d.h. sie gelangt über den NC-Kanal auf die Schnittstelle zur SPS. Nach DIN 66025 wird das physikalische Einsetzen des neuen WZ in die Arbeitsspindel mit M06 getriggert. Die getrennte Angabe der beiden Befehle „T mit WZ-Nummer“ und M06 kann dazu genutzt werden, nach dem T-Befehl vorbereitende Maßnahmen zu ergreifen (z.B. im WZ-Magazin), bevor das WZ mit M06 tatsächlich in die Arbeitsspindel eingesetzt wird.

Im erweiterten WZM identifizieren sich WZ-Daten über eine dreiteilige sog. Tool-ID Nummer:

Tool-ID = Grund-WZ-Nummer + Schwester-WZ-Nummer + Abwandlungsnummer

Die Grund-WZ-Nummer beschreibt den WZ-Typ und die Schwester-WZ-Nummer definiert ein WZ-Exemplar diesen Typs. Die Abwandlungsnummer hat rein datentechnische Bedeutung. Für ein WZ können damit unterschiedliche Datensätze verwendet werden.

19.1.2 Standgrößenerfassung

Bei der Standgrößenerfassung werden die Eingriffszeit (Standzeit) und der im Eingriff zurückgelegte Weg (Standweg) eines WZ berechnet und die ermittelten Werte an den WZM gesendet. Für die Konfiguration der Standgrössenerfassung sind diverse Kanalparameter erforderlich.

In der Grundeinstellung wird nur das Verfahren von Bewegungssätzen berücksichtigt. Das Positionieren im Eilgang wirkt sich auf die Standgrößen nicht aus.

Die Daten WZ-ID, Eingriffszeit und Eingriffsweg werden nach dem Auswechseln des WZ vom Interpolator angezeigt. Der Triggerpunkt für die Aktivierung der Standgrössenerfassung kann durch P-CHAN-00482 entweder mit dem T- oder dem D-Wort verknüpft werden.

Die Standzeit wird in ms, der Standweg in mm erfasst.

Mit Hilfe von Gewichtungsfaktoren [▶ 820] kann die Standgrößenerfassung an den WZ-Einsatz angepasst werden.

19.2 Verwenden der Werkzeug-ID (V.TOOL.) (#TOOL DATA, #TOOL PREP)

Die Tool-ID wird über Klartextbefehle programmiert. Die Anforderung neuer Werkzeudaten mit #TOOL DATA entspricht dem D-Befehl, der vorbereitende Technologiebefehl für den physikalischen Werkzeugwechsel mit #TOOL PREP dem T-Befehl.

Syntax:

#TOOL DATA [<*basic*> [**,** <*sister*> [**,** <*variant*>]] **]**	Anfordern neuer WZ-Daten
#TOOL PREP [<*basic*> [**,** <*sister*> [**,** <*variant*>]] **]**	Ankündigung des WZ-Wechsels

Die Anzahl der Parameter ist fest vorgegeben [6] [▶ 894]-9.18. Sinnvolle Werte liegen zwischen 1 und 3. Falls zwei Parameter erwartet werden, so sind dies die Grund- und die Schwesterwerkzeug-Nummer.

Zwingend ist die Angabe der Grundwerkzeug-Nummer (**basic**). Ist darüber hinaus die Anzahl Parameter mit 3 belegt, so können **sister** für die Schwesterwerkzeug-Nummer und **variant** für die Abwandlung optional programmiert werden. Falls sister oder variant nicht programmiert sind (Komma folgt auf Komma oder Klammer zu ')' folgt auf Komma), so wird dafür 0 eingesetzt.

Programmierbeispiel

Programmierbefehle und Variablen mit 'sister' und 'variant'

```
#TOOL DATA [ P10, "SISTER", 0 ] <=> #TOOL DATA [ P10, "SISTER", ]
und entsprechend
#TOOL PREP [P10, 0, "VARIANT"]  < = > #TOOL PREP [P10, , "VARIANT"]
```

Die mathematischen Ausdrücke bei Angabe des D- und T-Befehls sind als Grund-WZ-Nummer zu interpretieren. Damit bleibt dem WZM der gleiche Freiheitsgrad wie bisher für die Auswahl des WZ-Datensatzes erhalten.

Syntax:

T<*basic*> bzw. **D**<*basic*>

Der Zugriff auf die Elemente der Tool-ID wird über Decodervariablen realisiert. In der aktuellen SW-Version besteht bereits die Möglichkeit, die Nummer des zuletzt von der externen Werkzeugverwaltung angeforderten Werkzeuges über V.G.T_AKT abzufragen. Aus Kompatibilitätsgründen bleibt diese Variable weiterhin erhalten. Sie repräsentiert parallel zur unten eingeführten, neuen Syntax immer die Grundwerkzeug-Nummer.

Syntax:

V.TOOL.BASIC	Lesender Zugriff auf zuletzt programmierte Grundwerkzeug-Nummer
V.TOOL.SISTER	Lesender Zugriff auf zuletzt programmierte Schwesterwerkzeug-Nummer
V.TOOL.VARIANT	Lesender Zugriff auf zuletzt programmierte Abwandlungsnummer

Programmierbeispiel

Programmierbefehle und Variablen mit 'sister'

```
#TOOL DATA [ P10, "SISTER", 3 ]
.....
#TOOL PREP [V.TOOL.BASIC, V.TOOL.SISTER, V.TOOL.VARIANT]
```

19.3 Werkzeugdaten auffrischen (#TOOL REFRESH)

Für das erneute Einrechnen von Werkzeugdaten des aktiven Werkzeuges z.B. in zusätzlichen Achsen nach einem Achstausch im 2.5D-Betrieb oder Aktualisieren kinematikrelevanter Versatzdaten in Kanal- oder Werkzeugparametern bei aktiver Kinematik steht folgender Befehl zur Verfügung:

Syntax:

#TOOL REFRESH Daten des aktiven Werkzeuges auffrischen

Der Befehl bewirkt ein sofortiges internes Einrechnen der bereits vorhandenen Daten des aktiven Werkzeuges. Es werden keine Werkzeugdaten mehr aus einer internen Werkzeugliste [5] [▶ 894] übernommen oder von einer externen Werkzeugverwaltung angefordert. Das erneute Programmieren des D-Wortes ist nicht erforderlich.

19.4 Lesen/Löschen von Standgrößen (#TOOL LIFE READ/REMOVE)

Für den Zugriff auf die Standgrößen vor dem nächsten Werkzeugwechsel stehen nachfolgende Befehle zur Verfügung. Diese sind z.B. während dem Synchronbetrieb bei der Ermittlung der Standgrößen von Werkzeugen der Slaveachsen erforderlich, da diese wie im Kapitel Gewichtungsfaktoren für Standzeit und Standweg (V.TLM) [▶ 820] beschrieben nicht mit erfasst werden. Dazu wird über den nachfolgenden Befehl die Standgröße des momentan im Kanal aktiven Werkzeuges gelesen und an die Werkzeugverwaltung für ein beliebiges Werkzeug (z.B. Tool-ID eines Werkzeuges einer Slaveachse) ausgegeben.

Syntax:

#TOOL LIFE READ [<*basic*> [, <*sister*> [, <*variant*>]] **]** — Aktuelle Standgrößen lesen und einer Tool-ID in der Werkzeugverwaltung zuordnen

Für die Programmierung der Tool-ID gelten die gleichen Regeln wie z.B. für #TOOL PREP (Kapitel Programmierbefehle und Variablen [▶ 816]). Die Tool-ID kann gemäß HÜEMNOS-Konvention 1,2 oder 3 WZ-Angaben enthalten. Es ist immer mindestens eine Werkzeugangabe erforderlich, die dann stets als Grundwerkzeug-Nummer (basic) interpretiert wird. Die Werkzeugangaben können beliebige mathematische Ausdrücke sein; z.B. V.G.T_AKT, V.TOOL.BASIC, Pxx, V.L.xxx etc.

Die interne Standgrößenerfassung wird nach Übergabe an die Werkzeugverwaltung nicht auf 0 zurückgesetzt, sondern weiter aufsummiert.

Durch einen weiteren Befehl können die bislang erfassten Standgrößen des momentan im Kanal aktiven Werkzeuges ohne Ausgabe an die Werkzeugverwaltung auf 0 zurückgesetzt werden.

Syntax:

#TOOL LIFE REMOVE — Aktuelle Standgrößen löschen

Programmierbeispiel

Lesen und Löschen von Standgrößen

```
....
:
....   Normalbetrieb
:
#TOOL LIFE READ [V.G.T_AKT]    (Lese Standgrößen des akt. Werkzeuges)
#TOOL LIFE REMOVE              (Standgrößen zurücksetzen zur getrennten)
                               (Erfassung der Standgrößen während)
                               (Synchronbetrieb)

Anwahl Synchronbetrieb
:
....   Synchronbetrieb
:
Abwahl Synchronbetrieb
#TOOL LIFE READ [10]          (Annahme: T10 wird im Synchronbetrieb)
                              (mit bewegt: Ausgabe Standgrößen)
                              (an Werkzeugverwaltung für T10)
:
....   Normalbetrieb
:
```

19.5 Gewichtungsfaktoren für Standzeit und Standweg (V.TLM)

Die Gewichtung der Standgrößen kann über das NC-Programm verändert werden. Die veränderbaren Faktoren dienen zur Anpassung der Standgrößenerfassung an den Werkzeugeinsatz.

Bei jedem mit dem T- bzw. D-Befehl (siehe P-CHAN-00482) eingeleiteten Werkzeugwechsel werden automatisch die komplette Werkzeug-ID, die Standzeit und der Standweg an die Werkzeugverwaltung geschickt. Anschließend werden alle Größen genullt und die Standgrößenerfassung für das neu eingewechselte Werkzeug startet.

Für die Programmierung der Gewichtungsfaktoren der Standzeit und des Standweges dienen die folgenden zwei Decodervariablen (Zugriff nicht synchron zur Echtzeit):

Syntax:

V.TLM.TIME_FACT Gewichtungsfaktor der Standzeit ≥ 0.0
V.TLM.DIST_FACT Gewichtungsfaktor des Standweges ≥ 0.0

Die Variablen sind schreib- und lesbar. Beim ersten Programmstart nach Steuerungshochlauf sind beide Faktoren 1.0 (100%). Im NC-Programm geänderte Faktoren sind über Programmende und Reset haltend wirksam. Sie müssen bei Bedarf wieder explizit im NC-Programm auf 1.0 zurückgesetzt werden, können aber auch mit Werten > 1.0 programmiert werden, falls dies erforderlich ist. Beide Variablen dürfen in einem NC-Satz beschrieben werden.

Beispiel

Ist ein Werkzeug immer im Eingriff, so sollte mit Faktor 1.0 (100%) gewichtet werden. Findet aber nur auf der Hälfte des Verfahrweges ein Materialabtrag statt, so kann mit einem Gewichtungsfaktor von 0.5 gerechnet werden. Der Standardwertwert der Gewichtungsfaktoren für Standzeit und Standweg beträgt 1.0.

Bedingungen für die Erfassung:

- Eilgangsätze werden bei der Standgrößenerfassung nicht berücksichtigt.
- Beträgt der Vorschub Null, so ruht auch die Standgrößenerfassung.
- Es werden alle Bewegungsarten mit Ausnahme der Eilganginterpolation für die Standgrößenerfassung erfasst. Z.B. werden G01, G02, G03, Splineinterpolation und G63 einbezogen.
- Die Gewichtungsfaktoren gehen in die Berechnung ein.
- Bei an der Bewegung beteiligten Achsen wird nicht zwischen Haupt- und Mitschleppachsen unterschieden. Es wird immer der Bahnvorschub für die Wegaddition genommen. Im Fall von allein im Satz programmierten Mitschleppachsen wird der zurückgelegte Weg der Mitschleppachse zum Standweg addiert. Falls dies nicht erwünscht ist, kann der Programmierer durch die Vorgabe der Gewichtungsfaktoren V.TLM.TIME_FACT/DIST_FACT = 0.0 korrigierend eingreifen.
- Nicht berücksichtigt werden aktive Master-Slave-Anordnungen.
- Bei einem Reset oder Programmabbruch werden ebenfalls die zuletzt aktuellen Werte in der Datenbasis der Werkzeugverwaltung aufgenommen.
- Wird nur eingewechselt, d.h. bisher war kein Werkzeug in der Arbeitsspindel und die aktuelle T-Nummer gleich Null, so werden keine Daten verschickt.
- Werkzeugdaten werden nur verschickt, wenn tatsächlich eine Werkzeugverwaltung vorhanden ist (P-CHAN-00016).

19.6 Setzen von Standgrößen (#TOOL LIFE DEF)

Die Aufzeichnung der (relativen) Standgrößen von Weg und Zeit des aktuellen Werkzeuges erfolgt, sobald dieses Werkzeug für die Standgrößenberechnug relevante Bewegungen (z.B. G01, G02, G03) im NC-Kanal ausführt. Hierbei beginnt die Berechnung in der Regel bei Null. Die erfassten inkrementellen Werte werden dann mit dem nächsten Werkzeugwechsel an eine externe Werkzeugverwaltung übermittelt. Diese bildet durch Aufsummierung und Gewichtung die absoluten Werte für Standweg und Standzeit.

In bestimmten Situationen kann es erforderlich sein, im NC-Kanal die inkrementelle Berechnung der Standgrößen für ein Werkzeug nicht von Null sondern von bestimmten Werten zu starten. Für die Initialisierung der Standgrößen von Weg und Zeit für das aktuell im NC-Kanal befindliche Werkzeug steht dazu folgender NC-Befehl zur Verfügung:

Syntax:

#TOOL LIFE DEF [DIST=.. **TIME**=..**]**	Standgrößen des aktiven Werkzeuges setzen
DIST=..	Standweg in [mm]
TIME=..	Standzeit in [s]

20 Positionierachsen

Eine vollständige Liste der Achsspezifische Zusatzfunktionen findet sich in der Befehlsübersicht im Anhang unter Achsspezifische Zusatzfunktionen (<X>[..]) [▶ 891].

Positionierachsen sind translatorische oder rotatorische Achsen, die im gleichen NC-Kanal unabhängig vom Bahnachsverbund interpoliert werden können. Jede Positionierachse besitzt ihren eigenen Achsinterpolator und kann mit einer individuellen Geschwindigkeit (Vorschub) beauftragt werden. Rotatorische Positionierachsen im Modulobetrieb verfahren immer auf kürzestem Weg (shortest way).

Einschränkungen:

Eine Achse kann nicht als Positionierachse programmiert werden, wenn:

- Diese Achse momentan im Synchronbetrieb fährt
- Eine kinematische oder kartesische Transformation aktiv ist und bestimmte Voraussetzungen nicht gegeben sind (siehe Kapitel Kartesische / kinematische Transformationen und Positionierachsen [▶ 833]).
- Satzvorlauf aktiv ist
- Simulationsmodis (Online-Simulation, Konturvisualisierung, Fertigungszeitberechnung) aktiv sind
- Splineinterpolation aktiv ist
- Polynomüberschleifen aktiv ist
- Drehfunktionen aktiv sind

20.1 Unabhängige Achsen (INDP_SYN, INDP_ASYN) (#WAIT INDP, #WAIT INDP ALL)

Bei der Programmierung s.g. unabhängiger Achsen werden zwei Betriebsmodi unterschieden:

- Sollwertseitige Synchronisierung von Bahnachsen und unabhängigen Achsen jeweils am Satzende.
- Sollwertseitige Synchronisierung von Bahnachsen und unabhängigen Achsen über mehrere Satzgrenzen hinweg.

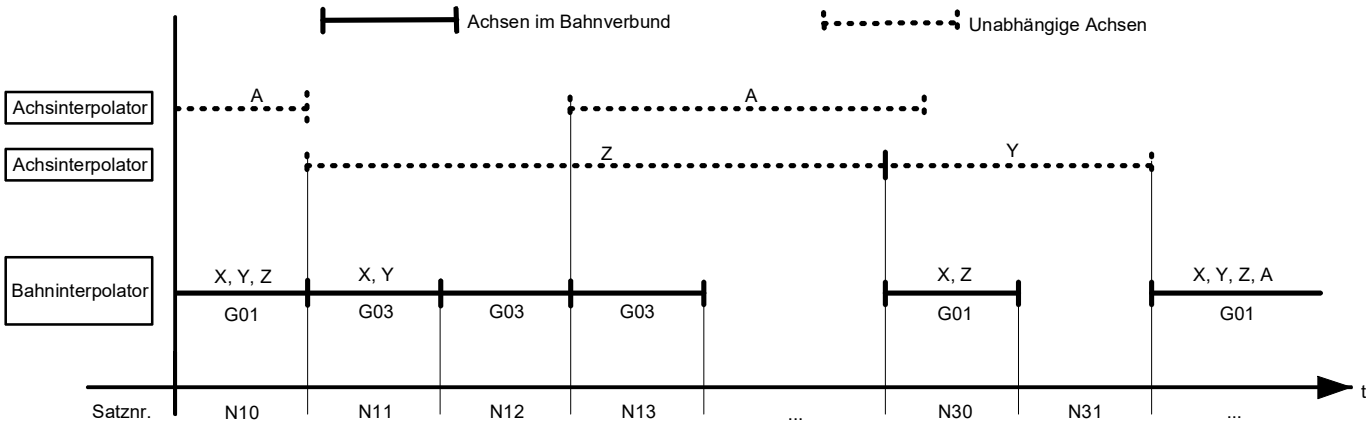


Abb. 206: Prinzipielles Bewegungsschema Bahnachsverbund/unabhängige Achsen

Hinweis

Bei unabhängigen Achsen werden in der Grundeinstellung keine Verschiebungen berücksichtigt. Ab der Version V3.1.3081.05 kann im NC-Befehl durch Angabe von INCL_OFFSETS die Einrechnung von Verschiebungen in die programmierte Achsposition aktiviert werden.

Der additive Handsatzbetrieb (G201/G202) mit einer unabhängigen Achse ist möglich.

Syntax zur Programmierung unabhängiger Achsen:

<Achsname> **[INDP_SYN** | **INDP_ASYN G90** | **G91 G00** | [**G01** | **G100 FEED**=.. |
TIME=.. | **FEED_MAX_WEIGHT**=..] **POS**=.. [**SLOPE_TYPE**=*<ident>*]
{**M..**} {**H..**} [**DRY_RUN**] [**ACC_WEIGHT**=..] [**RAPID_ACC_WEIGHT**=..]
[**INCL_OFFSETS**] { \ } **]**

<Achsname>	Name der unabhängigen Achse
INDP_SYN	Kennung für die synchrone (satzweise) unabhängige Achsbewegung. Übergang zum nächsten Satz erfolgt erst dann, wenn alle Achsen ihre Endpositionen erreicht haben. Muss immer als erstes Schlüsselwort programmiert sein.
INDP_ASYN	Kennung für die asynchrone (satzübergreifende) unabhängige Achs-bewegung. Keine Synchronisierung auf Endpositionen. Die sollwertseitige Synchronisierung erfolgt durch einen speziellen Befehl (#WAIT INDP) oder durch Programmierung der unabhängigen Achse als bewegte Bahnachse. Muss immer als erstes Schlüsselwort programmiert sein.
G90 / G91	Absolut- / Relativmaß
G00 / G01	Eilgang- / Linearinterpolation
FEED=..	Achsspezifischer Vorschub in [mm/min, m/min, inch/min]
TIME=..	Achsspezifische Verfahrzeit in [s]

FEED_MAX_WEIGHT=..	Gewichtungsfaktor in [%] bezogen auf den achsspezifischen maximalen Vorschub P-AXIS-00212. Es sind nur Gewichtungswerte kleiner 100% erlaubt. (gemäß G194, Kapitel Bearbeitungszeit/Vorschubgeschwindigkeit (G93/G94/G95/G194) [▶ 155])
POS=..	Achsposition in [mm, inch]
SLOPE_TYPE=<*ident*>	Slope-Profiltypen gemäß #SLOPE [TYPE=...], Kapitel Unabhängige Achsen [▶ 377]) Wenn kein Slopetyp programmiert ist, wird per Default der Slopetyp aus dem Kanalparametersatz P-CHAN-00071 gesetzt.
Alte Syntax: *SLOPE_PROFIL=..*	*Slope-Profiltypen (0, 1, 2, 3) Wenn kein Slopetyp programmiert ist, wird per Default der Slopetyp aus dem Kanalparameter* P-CHAN-00071 *gesetzt.*
G100 **[ab V2.11.2801.05]**	Bei Verwenden der Messtypen 1, 2 oder 7 (Kapitel Messfunktionen [▶ 96]) kann auch eine Messfahrt mit unabhängigen Achsen durchgeführt werden. Das Latchen der Messposition erfolgt dabei für jede beteiligte Achse individuell. Eine unabhängige Messfahrt ist auch parallel zu einer Bahnbewegung oder G100 Messfahrt möglich. Weitere Informationen siehe [FCT-C4//Messen].
DRY_RUN	Trockenlauf der Achsbewegung. Die Bewegung wird nur im NC-Kanal ausgeführt ohne dass die Achse real verfahren wird. Hierdurch kann die Achskoordinate innerhalb des Kanals gegenüber der physikalischen Achse verschoben werden. Diese Verschiebung wird automatisch bei jedem Programmstart oder durch ein explizit programmiertes #CHANNEL INIT [CMDPOS] wieder aufgehoben (siehe Programmierbeispiel 3).
M..	Achsspezifische M-Funktionen (*)
H..	Achsspezifische H-Funktionen (*)
ACC_WEIGHT=.. **[ab V3.1.3079.06]**	Achsspezifischer Gewichtungsfaktor in [%] bezogen auf die Beschleunigung bei G01, G02, G03 (gemäß Kapitel Beschleunigungsgewichtung (G130/G131/G230/G231/G333/G334) [▶ 150])
RAPID_ACC_WEIGHT=.. **[ab V3.1.3079.06]**	Achsspezifischer Gewichtungsfaktor in [%] bezogen auf die Beschleunigung bei G00 (gemäß Kapitel Beschleunigungsgewichtung (G130/G131/G230/G231/G333/G334) [▶ 150])
INCL_OFFSETS **[ab V3.1.3081.05]**	Einrechnen der aktuell aktiven achsspezifischen Verschiebungen (z.B. G55, G92 etc.) in die programmierte Achsposition POS
\	Trennzeichen ("Backslash") für übersichtliche Programmierung des Befehls über mehrere Zeilen

(*) nur möglich mit den Synchronisationsarten MOS, MVS_SVS, MVS_SNS, MNS_SNS.

Achsspezifische M/H-Funktionen können auch ohne die Programmierung einer Bewegung an eine unabhängige Achse ausgegeben werden. Dazu ist zusätzlich nur die Kennung INDP_SYN bzw. INDP_ASYN erforderlich:

Syntax:

<*Achsname*> **[INDP_SYN** | **INDP_ASYN M**.. {**M**..} **H**.. {**H**..} { \ } **]**

<*Achsname*>	Name der unabhängigen Achse
INDP_SYN/INDP_ASYN	Kennung für eine unabhängige Achse
M..	Achsspezifische M-Funktion
H..	Achsspezifische H-Funktion
\	Trennzeichen ("Backslash") für übersichtliche Programmierung des Befehls über mehrere Zeilen

Die sollwertseitige Synchronisation bestimmter asynchroner Achsbewegungen (INDP_ASYN) wird erzwungen durch:

Syntax:

#WAIT INDP [*<Achsname>* { **,***<Achsname>* } **]**

<Achsname> Name der asynchronen Achse

Hinweis

Wird eine asynchrone Achse vor oder ohne dem entsprechenden #WAIT INDP [ ] erneut mit einer Bewegung programmiert, so wird die sollwertseitige Synchronisation implizit im Interpolator durchgeführt.

Die sollwertseitige Synchronisation von allen momentan aktiven asynchronen Achsbewegungen (INDP_ASYN) wird erzwungen durch:

Syntax:

#WAIT INDP ALL

Hinweis

Wird die Achse einer vorbelegten achsspezifischen M/H-Funktion (P-CHAN-00039, P-CHAN-00025) im gleichen NC-Satz als unabhängige Achse programmiert, so wird eine Fehlermeldung ausgegeben.

Beispiel: M10 ist X-achsspezifisch vorbelegt (m_default_outp_ax_name[10] x):

N10 **M10** **X** [INDP_SYN G01 G90 POS10 FEED1000 M7]

|_____| **<- Fehler!**

Programmierbeispiel

Unabhängige Achsen

```
;Beispiel 1:
;N10 ist beendet, wenn X,Y und die unabh. synchr. Z-Achse ihre
;Bewegung beendet haben
N10 X10 Y11 Z[INDP_SYN POS50 G01 FEED100 G90]

;N20 wird ausgeführt, nachdem alle Bewegungen in N10 beendet sind
N20 X20

;N30 ist beendet, wenn X u. Y ihre Bewegung beendet haben; die
;unabhängige asynchrone Achse fährt weiter
N30 X5 Y10 Z[INDP_ASYN POS500 G01 FEED200 G90]

;N40 wird interpoliert, die asynchrone unabhängige Z-Achse fährt weiter
N40 X20 Y30

;Erzwungene Synchronisierung der Z-Achse: warten, bis Zielposition
;Z500 aus N30 erreicht ist
N50 #WAIT INDP[Z]

;Interpolation in N60 mit X, Y, Z im Bahnverbund wird gestartet,
;nachdem Synchronisation in N50 erfolgte
N60 X30 Y40 Z60
N70 Z[INDP_SYN M50]   ;Ausgabe von M50 über unabhängige Z-Achse
N80 …

;Beispiel 2:
;N10 wird interpoliert, die unabhängige asynchrone Z-Achse fährt
;weiter
N10 X10 Y11 Z[INDP_ASYN POS500 G01 FEED200 G90]

;N20 wird interpoliert, die unabhängige asynchrone Z-Achse fährt
;weiter
N20 X20 Y22

;Implizite Synchronisation der Z-Bewegung von N10 bevor Bewegung
;Z550 begonnen wird
N30 Z550

N40 X20 Y30 Z60  ;N40 wird interpoliert
N50 …
```

```
;Beispiel 3:
%dry_run
N100 X1 Y2 Z3                                ;IPO=3, LR=3, offset=0

N200 G01 X10 F100 Z[INDP_SYN POS=4 G01 G90 \
                    FEED=120 DRY_RUN]        ;IPO=4, LR=3, offset=1
N300 Y20 F1000
N350 Z[INDP_SYN POS=7 G00 G90]               ;IPO=7, LR=6, offset=1
N360 Z[INDP_SYN POS=4 G01 G91 \
       FEED=100 DRY_RUN]                     ;IPO=11, LR=6, offset=5

;Abloeschen des DRY_RUN-offsets
N001 #TIME 2
N111 #CHANNEL INIT[CMDPOS]                   ;IPO=6, LR=6, offset=0
N222 #TIME 2
N400 Y10 Z5
M30

;Beispiel 4:
;Die unabhängige synchrone X-Achse fährt ihre Position unter
;Einrechnung des aktiven G92-Versatzes an
N10 G0 X0
N20 G92 X200
N10 X[INDP_SYN POS50 G01 FEED100 G90 INCL_OFFSETS] ;X fährt auf 250
```

20.2 Pendelachsen (OSC)

Versionshinweis

V200
V260
V300

Die Verfügbarkeit dieser Funktionalität ist von der Konfiguration und dem Versionsumfang abhängig.

Für bestimmte Bearbeitungstechnologien wie z.B. Schleifen ist eine oszillierende Achsbewegung erforderlich, die weitgehend unabhängig von einer Bahnbewegung ausgeführt wird.

Diese im Folgenden als „Pendelbewegung“ bezeichnete Bewegung führt das Werkzeug periodisch reversierend über dem Werkstück aus.

Beispielhaft ist im Folgenden eine Pendelachse beim Schleifen dargestellt. Die Werkstückbearbeitung erfolgt dabei durch Überlagerung der X - Pendelbewegung mit Positionierbewegungen in Y und Z- Achse.

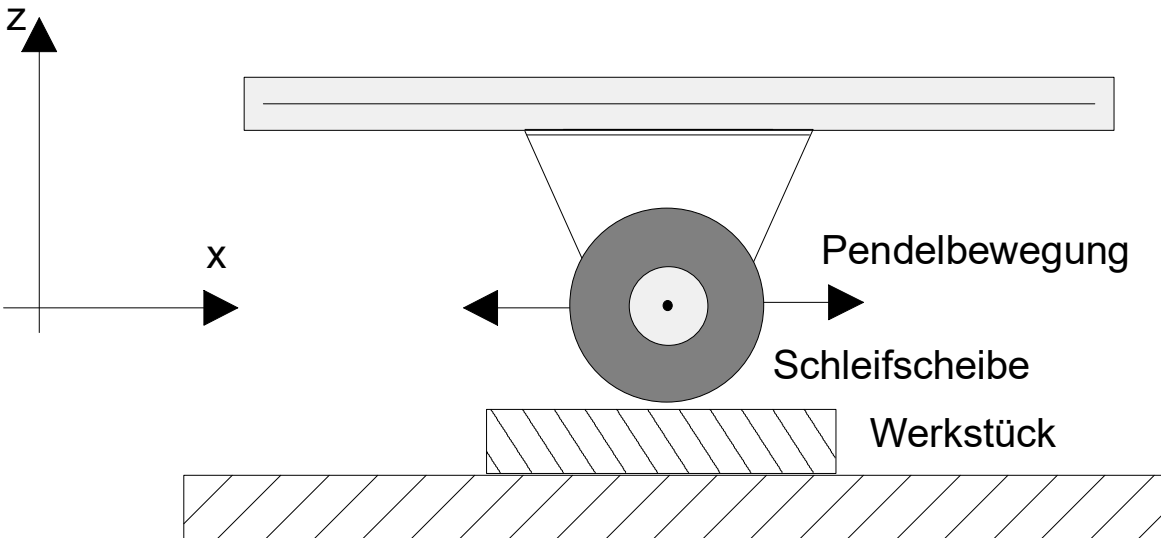


Abb. 207: Schleifen mit Pendelachse

Die wesentlichen Eigenschaften der Pendelbewegung ergeben sich über zwei Absolutpositionen zwischen denen die Pendelbewegung ausgeführt wird, sowie der Vorschubgeschwindigkeit.

Start, Stopp und Parametrierung der Pendelbewegung erfolgen über das NC-Programm.

Innerhalb der konfigurierten Achsen kann jede beliebige Achse als Pendelachse festgelegt werden. Die Pendelbewegung erfolgt asynchron zur Bahnbewegung.

Die Deaktivierung der Pendelbewegung erfolgt entweder:

- direkt durch einen NC-Befehl
- oder implizit durch Programmierung einer Bahnbewegung für d. Pendelachse
- oder implizit bei der Anforderung von Achspositionen zur Synchronisation von Decodierung und Interpolation
- oder implizit am Programmende.

Durch Angabe des Profiltyps in den Kanalparametern P-CHAN-00071 (linearer/nichtlinearer Slope) kann für die Pendelbewegung die Art des Geschwindigkeitsverlaufs in der dynamischen Phase festgelegt werden.

Bei einer Moduloachse müssen die Pendelpositionen innerhalb des Modulobereichs angegeben werden. Ist dies nicht der Fall, wird der Fehler ID 22277 ausgegeben.

Die Programmiersyntax lehnt sich an die achsspezifische Programmierung von unabhängigen Achsen an. Nach dem Achsbezeichner erfolgt die Parametrierung der Pendelbewegung über Schlüsselworte und ggf. zugehörige Werte:

Syntax zur Programmierung einer Pendelbewegung:

<Achsname> **[OSC ON** | [**OFF** | **OFF FEED=..** | **OFF INSTANT**]
FEED=.. | **FREQ**=.. | **TIME**=.. [**1ST_POS**=.. **2ND_POS**=..]
| [**ZERO_POS**=.. **EXCUR**=..] [**1ST_DELT**=.. **2ND_DELT**=..] [**NBR_OSC**=..]
[**INCL_OFFSETS**] [**SHORT**] { \ } **]**

<Achsname>	Name der Pendelachse
OSC	Kennung für die Funktionalität "Pendeln". Muss immer als erstes Schlüsselwort programmiert sein.
ON	Pendeln einschalten. Bei aktiver Bahnbewegung wird am Satzende angehalten und dann die Pendelbewegung beauftragt.
OFF	Pendeln ausschalten. Aktueller Pendelzyklus wird zu Ende gefahren. Danach kann die Pendelachse wieder im Bahnverbund bewegt werden. Ohne vorherige Abwahl wird die Pendelbewegung bei Programmierung einer neuen Achsbewegung implizit abgebrochen.
OFF FEED=..	Schneller Pendelstopp. Der aktuelle Pendelzyklus wird abgebrochen und die Achse fährt mit vorgegebenem Vorschub auf Zielposition (2ND POS). Danach kann die Pendelachse wieder im Bahnverbund bewegt werden.
OFF INSTANT	Sofortiger Pendelstopp. Achse hält unmittelbar an und kann sofort wieder im Bahnverbund bewegt werden. Verfügbar ab V3.1.3107.38
FEED=..	Vorschub der Pendelbewegung in [mm/min, m/min, inch/min]
FREQ=..	Frequenz der Pendelbewegung in [Hz]
TIME=..	Periodendauer der Pendelbewegung in [s]
1ST_POS=..	Erste Umkehrposition in [mm, inch]
2ND_POS=..	Zweite Umkehrposition in [mm, inch]
ZERO_POS=..	Nullpunkt bzw. Nulldurchgang der Pendelbewegung in [mm, inch]
EXCUR=..	Auslenkung in [mm, inch]
1ST_DELT=..	Wartezeit an erster Umkehrposition in [s]
2ND_DELT**=..**	Wartezeit an zweiter Umkehrposition in [s]
NBR_OSC=..	Anzahl Schwingungen
INCL_OFFSETS **[ab V3.1.3081.05]**	Einrechnen der aktuell aktiven achsspezifischen Verschiebungen (z.B. G55, G92 etc.) in die programmierten Umkehrpositionen 1ST_POS/2ND_POS bzw. Nulldurchgang ZERO_POS
SHORT **[ab V3.1.3113.0]**	Pendeldistanz wird bei einer Moduloachse auf dem kürzesten Weg zurückgelegt.
\	Trennzeichen ("Backslash") für übersichtliche Programmierung des Befehls über mehrere Zeilen

Die Eigenschaft der Pendelbewegung wird durch die Lage der Umkehrpositionen und dem Achsvorschub bestimmt. Die Umkehrpositionen können entweder direkt angegeben werden oder sie werden alternativ über Nullposition und Auslenkung automatisch bestimmt.

Bei den Pendelpositionen handelt es sich immer um **Absolutpositionen**.

Nach Abwahl der Pendelbewegung wird immer auf Pendelposition 2 gestoppt!

Die Pendelgeschwindigkeit kann alternativ über den Vorschub, die Frequenz oder die Periodendauer festgelegt werden.

Sofern keine Beschränkung aufgrund der dynamischen Achskenngrößen auftritt, werden bei Verwendung des linearen Slopes die Frequenz und die Periodendauer exakt eingehalten, beim nichtlinearen Slope näherungsweise.

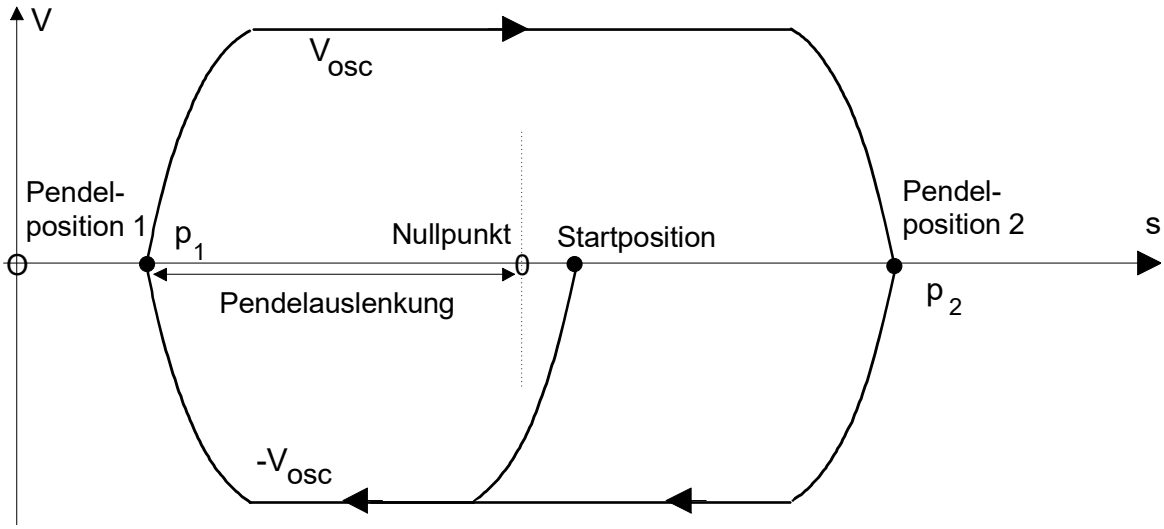


Abb. 208: Positioniervorgang bei der Pendelbewegung

Programmierbeispiel

Angabe des Pendelfahrweges über Umkehrpositionen, in die Versätze eingerechnet werden
```
N10 X[OSC ON 1ST_POS=-100 2ND_POS=100 FEED=1000 INCL_OFFSETS]
```

Angabe des Pendelfahrweges über Nullposition und Auslenkung,
10 Schwingungen
```
N20 X[OSC ON ZERO_POS=0 EXCUR=100 FEED=1000 NBR_OSC=10]
```

Angabe der Pendelfrequenz 1 Hertz
```
N30 X[OSC ON ZERO_POS=0 EXCUR=100 FREQ=1]
```

Angabe der Pendelperiodendauer 4s
```
N40 X[OSC ON ZERO_POS=0 EXCUR=100 TIME=4]
```

Pendelbewegung mit Zustellbewegung einer Bahnachse
```
N50 X[OSC ON 1ST_POS=111 2ND_POS=222 FREQ=1]
N60 G01 G90 Y500 F200
```

Pendeln mit Wartezeiten an den Umkehrpositionen von jeweils 0.5 s:
```
N70 X[OSC ON 1ST_POS-100 1ST_DELT0.5 2ND_POS200 2ND_DELT0.5 FEED1000]
```

Pendeln einer Moduloachse (0-360°) auf dem kürzesten Weg über die Modulogrenze
```
N80 X[OSC ON 1ST_POS=10 2ND_POS=350 FEED=1000 SHORT]
```

Abwahl Pendeln
Pendelbewegung wird mit Erreichen der Umkehrposition 2 beendet.
```
N80 X[OSC OFF]
```

Schneller Pendelstopp
Wird in Verbindung mit OFF ein Vorschub FEED programmiert, so wird die Pendelbewegung sofort abgebrochen (Feedhold der Pendelachse) und es wird mit dem neuen Vorschub direkt auf Umkehrposition 2 gefahren.
```
N90 X[OSC OFF FEED=5000]
```

Sofortiger Pendelstopp

Die aktuelle Pendelbewegung wird abgebrochen (Feedhold der Pendelachse). Nach dem Stillstand wird die aktuelle Achsposition mit dem Decoder synchronisiert. Die Achse steht dem Bahnverbund wieder zur Verfügung.

```
N90 X[OSC OFF INSTANT]
```

20.3 Kartesische / kinematische Transformationen und Positionierachsen

20.3.1 Positionierung und Versätze

Bei konventionellem Betrieb und CS-Betrieb (#CS, #ACS) mit aktiver unabhängiger Achse oder Pendelachse muss der Anwender Absolutpositionen für diese Achse programmieren. D.h. wenn ein Werkzeug eingewechselt wurde, muss eventuell die Länge des Werkzeugs bei der Programmierung der Achse berücksichtigt werden. Zuvor programmierte Nullpunktversätze (G54...G59) oder Bezugspunktverschiebungen (G92) sind nicht wirksam.

Bei Betrieb mit aktiver kinematischer Transformation (#TRAFO) werden die Werkzeugversätze direkt in der Transformation berücksichtigt. D.h. diese werden auch bei aktiven unabhängigen Achsen oder Pendelachsen berücksichtigt. Auch hier sind zuvor programmierte Nullpunktversätze (G54...G59) oder Bezugspunktverschiebungen (G92) nicht wirksam.

20.3.2 Einschränkungen

Vor der Neuanwahl einer kartesischen und/ oder kinematischen Transformation muss eine Pendelbewegung oder unabhängige Achsbewegung deaktiviert werden.

Eine Positionierachse darf nur programmiert werden:

- bei kartesischen Kinematiken, und
- nur in der 3. Achse (i. A. Z-Achse) bei senkrecht auf dem XY-Maschinenbett stehendem Werkzeug (z.B. A-Achse auf 0° bei CA-Kopfkinematik).

Programmierbeispiel

Programmierung unabhängiger Achsen:

```
N10 #KIN ID[9]
N20 #TRAFO ON
N30 Z[INDP_ASYN G01 G90 POS20 F0.01 SLOPE_TYPE=STEP]
N40 Z[INDP_ASYN G01 G90 POS-20 F0.01 SLOPE_TYPE=STEP]
N50 G01 G90 X100 F0.1
N60 #TRAFO OFF
N100 M30
```

```
N10 G00 X0 Y0 Z0 C0
N20 #CS ON[0,0,0,0,0,45]
N30 Z[INDP_ASYN G01 G90 POS20 F0.01 SLOPE_TYPE=STEP]
N40 Z[INDP_ASYN G01 G90 POS-20 F0.01 SLOPE_TYPE=STEP]
N50 G01 G90 X100 F0.1
N60 #CS OFF
N100 M30
```

```
N10 #KIN ID[9]
N20 #TRAFO ON
N30 #CS ON[0,0,0,0,0,45]
N40 Z[INDP_ASYN G01 G90 POS20 F0.01 SLOPE_TYPE=STEP]
N50 Z[INDP_ASYN G01 G90 POS-20 F0.01 SLOPE_TYPE=STEP]
N60 G01 G90 X100 F0.1
N70 #CS OFF
N80 #TRAFO OFF
N100 M30
```

Programmierbeispiel

Programmierung von Pendelachsen:

```
N10 G00 X0 Y0 Z0 C0
N20 #KIN ID[9]
N30 #TRAFO ON
N40 Z[OSC ON 1ST_POS=10 2ND_POS=20 FEED=1.00]
N50 G01 G90 X100 Y100 F0.1
N60 Z[OSC OFF FEED=2.00]
N70 #TRAFO OFF
N100 M30
```

```
N10 G00 X0 Y0 Z0 C0
N20 #CS ON[0,0,0,0,0,45]
N30 Z[OSC ON 1ST_POS=10 2ND_POS=20 FEED=1.00]
N40 G01 G90 X100 Y100 F0.1
N50 Z[OSC OFF FEED=2.00]
N60 #CS OFF
N100 M30
```

```
N10 G00 X0 Y0 Z0 C0
N20 #KIN ID[9]
N30 #TRAFO ON
N40 #CS ON[0,0,60,0,0,45]
N50 Z[OSC ON 1ST_POS=10 2ND_POS=20 FEED=1.00]
N60 G01 G90 X100 Y100 F0.1
N70 Z[OSC OFF FEED=2.00]
N80 #CS OFF
N90 #TRAFO OFF
N100 M30
```

Programmierbeispiel

Unzulässig verschachtelte Programmierung

Der folgende Programmauschnitt zeigt eine unzulässige verschachtelte Programmierung von CS mit kinematischer Transformation und Pendeln.

```
N10 #KIN ID[9]
N20 #TRAFO ON
N30 Z[OSC ON 1ST_POS=10 2ND_POS=20 FEED=1.00]
N40 G01 G90 X100 Y100 F0.1
N50 #CS ON[0,0,0,0,0,45]
N60 G01 G90 X100 F0.1
N70 #CS OFF
N80 #TRAFO OFF
N90 Z[OSC OFF FEED=2.00]
N100 M30
```

21 Achsspezifische Programmierung

Eine vollständige Liste der Achsspezifische Zusatzfunktionen findet sich in der Befehlsübersicht im Anhang unter Achsspezifische Zusatzfunktionen (<X>[..]) [▶ 891].

Die Programmiersyntax der folgenden NC-Befehle lehnt sich an die achsspezifische Programmierung von Positionierachsen an. Nach dem Achsbezeichner erfolgt die Parametrierung über Schlüsselworte und ggf. zugehörige Werte.

21.1 Ein-/Ausschalten von Achskompensationen im NC-Programm (COMP)

Versionshinweis

Diese Funktionalität ist verfügbar ab CNC-Version V2.10.1501.00

Neben der Möglichkeit über entsprechende Achsparameter können die verschiedenen Achskompensationen [FCT-C5] auch direkt im NC-Programm an- und abgewählt werden. Hierbei können achsspezifisch in einem NC-Satz für mehrere Achsen verschiedene Achskompensationen gleichzeitig aktiviert bzw deaktiviert werden.

Hinweis

Das Ausschalten der Achskompensationen über den COMP-Befehl wirkt NC-Programmübergreifend, d.h. bei Programmende werden die Kompensationen nicht automatisch reaktiviert. Sie müssen im folgenden NC-Programm explizit über den COMP-Befehl wieder eingeschaltet werden.

Syntax:

<*Achsname*> **[COMP** [[**ON** | **OFF** [**BACKLASH CROSS PLANE LEAD TEMP FRICT CROSSTALK**]] | **OFF_ALL**]

[**NO_MOVE**] { \ } **]**

<Achsname>	Name der Achse
COMP	Kennung für die An-/Abwahl von achsspezifischen Kompensationen. Muss immer als erstes Schlüsselwort programmiert sein.
ON	Programmierte Kompensation(en) einschalten
OFF	Programmierte Kompensation(en) ausschalten
BACKLASH	Schlüsselwort für Losekompensation [ab V3.1.3081.05
CROSS	Schlüsselwort für Kreuzkompensation
PLANE	Schlüsselwort für Flächenkompensation
LEAD	Schlüsselwort für Spindelsteigungsfehlerkompensation
TEMP	Schlüsselwort für Temperaturkompensation
FRICT	Schlüsselwort für Reibungskompensation [ab V2.11.2022.05]
CROSSTALK	Schlüsselwort für Nickkompensation [ab V3.1.3079.32]
OFF_ALL	Alle Kompensationen ausschalten. Dem Schlüsselwort dürfen keine Kompensationsbezeichner folgen.
NO_MOVE	Standardmäßig wird der beim Ein-/Ausschalten von Achskompensationen entstehende Positionsoffset ausgefahren, bevor die Programmbearbeitung fortgesetzt wird. Durch Angabe des Schlüsselworts NO_MOVE kann diese Bewegung unterdrückt werden. Der Kanal wird mit den geänderten Achspositionen initialisiert. Das Ausfahren des Positionsoffsets erfolgt erst mit der nächsten, im NC-Programm programmierten Achsbewegung.
\	Trennzeichen ("Backslash") für übersichtliche Programmierung des Befehls über mehrere Zeilen

Programmierbeispiel

Achsspezifische Programmierung

```
;Ausschalten von Kreuz- und Flächenkompensation in der X-Achse
N10 X[COMP OFF CROSS PLANE
;Kompensationsprogrammierung mehrerer Achsen in einem NC-Satz
N50 X[COMP OFF CROSS] Y[COMP ON LEAD TEMP]
;Ausschalten aller Kompensationen in der Z-Achse
N100 Z[COMP OFF_ALL]
;Ausschalten aller Kompensationen der Y-Achse ohne Achsbewegung
N200 Y[COMP OFF_ALL NO_MOVE]
```

21.2 Abstandsregelung (Getastete Spindeln) (DIST_CTRL)

Versionshinweis

Die Verfügbarkeit dieser Funktionalität ist von der Konfiguration und dem Versionsumfang abhängig.

Mithilfe dieser Funktionalität kann bei entsprechender Ausrüstung der werkzeugtragenden Achse (Spindelachse) der Abstand des Werkzeugs zu einer unregelmäßigen Werkstückoberfläche vorgegeben werden. Dieser Abstand wird über ein Messsystem erfasst und von der NC kontinuierlich der unregelmäßigen Oberfläche nachgeführt.

Die Freischaltung der Abstandsregelung für eine getastete Spindel erfolgt durch den Parameter P-AXIS-00328. Die Aktivierung erfolgt über den nachfolgenden NC-Befehl. Weitergehende Informationen können der Funktionsbeschreibung "Abstandsregelung" [FCT-M3] entnommen werden.

Syntax Anwahl mit Angabe der Position der Werkstückoberfläche:

<*Achsname*> **[DIST_CTRL ON** | **DRYRUN** [**SET_POS**=..] **]**

Syntax Anwahl mit Angabe konstanter Abstand zur Werkstückoberfläche:

<*Achsname*> **[DIST_CTRL ON** | **DRYRUN CONST_DIST** [**SET_DIST**=..] **]**

Syntax Abwahl oder Offset einfrieren:

<*Achsname*> **[DIST_CTRL** [**OFF** [**NO_MOVE**]] | **FREEZE]**

Syntax Sensor prüfen oder referenzieren:

<*Achsname*> **[DIST_CTRL CHECK_POS** | **REF]**

Die folgenden Parameter können optional auch in Kombination mit der An-/Abwahl programmiert werden:

Syntax zusätzliche Parametrierung Sensor:

<*Achsname*> **[DIST_CTRL** [**SENSOR_SOURCE**=<*ident*> **SENSOR_VAR**=..] [**VAL1**=.. - **VAL5**=..] { \ } **]**

Syntax zusätzliche Parametrierung für die Regelung:

<*Achsname*> **[DIST_CTRL** [**KP**=..] [**I_TN**=..] [**D_TV**=..] { \ } **]**

Syntax zusätzliche Parametrierung für die Signalglättung des Sensors:

<*Achsname*> **[DIST_CTRL** [**FILTER_TYPE**=..] [**N_CYCLES**=..] [**FG_F0**=..] [**ORDER**=..]
[**SMOOTH_FACT**=..] [**KALMAN_SIGMA**=..] { \ } **]**

<Achsname>	Name der werkzeugtragenden Achse.
DIST_CTRL	Kennung für die Funktionalität "Getastete Spindeln". Muss immer als erstes Schlüsselwort programmiert sein.
ON	Abstandsregelung einschalten bei Vorgabe der Position der Werkstückoberfläche. Beim Einschalten muss eine Sollposition mit SET_POS gesetzt sein.
SET_POS=..	Vorgabe der Position der Werkstückoberfläche in [mm, inch] (Absolutposition). Bei Reset oder Programmende wird die Vorgabe zurückgesetzt, d.h. vor dem Wiedereinschalten der Abstandsregelung muss eine neue Vorgabe vorgegeben werden.
CONST_DIST	In Verbindung mit ON Abstandsregelung einschalten bei Vorgabe eines konstanten Abstandes zur Werkstückoberfläche. Beim Einschalten muss ein Abstand mit SET_DIST gesetzt sein. **[ab V2.11.2804.03]**
SET_DIST=..	Sollvorgabe des konstanten Abstandes zur Werkstückoberfläche in [mm, inch]. Bei Reset oder Programmende wird der Abstand zurückgesetzt, d.h. vor dem Wiedereinschalten der Abstandsregelung muss ein neuer Abstand vorgegeben werden.
DRYRUN	In Verbindung mit ON wird im Modus DRYRUN die Achse bei Änderungen der Werkstückoberfläche nicht nachgeführt! Dies ermöglicht die Auswertung von Daten (Bsp. Filterwirkung) ohne Rückkopplung der Regelung. **[ab V3.1.3079.23]** Beim Einschalten der Abstandsregelung mit Angabe der Position der Werkstückoberfläche muss eine Sollposition mit SET_POS gesetzt sein. Beim Einschalten der Abstandsregelung mit Vorgabe eines konstanten Abstandes zur Werkstückoberfläche muss ein Sollabstand mit SET_DIST gesetzt sein.

OFF	Abstandsregelung ausschalten.
NO_MOVE	Standardmäßig wird beim Ausschalten der Abstandsregelung der entstandene Korrekturoffset ausgefahren. Durch Angabe von NO_MOVE in Kombination mit OFF kann diese Bewegung unterdrückt werden. Der Kanal wird mit den geänderten Achspositionen initialisiert. Das Ausfahren des Positionsoffsets erfolgt erst mit der nächsten, im NC-Programm programmierten Achsbewegung.
FREEZE	Einfrieren des ausgeregelten Abstandes über Werkstück. Die Achsposition bzw. der ausgegebene Korrekturwert wird gehalten. Die Nachführung der Achse wird unterbrochen.
CHECK_POS	Prüfen, ob Position im Toleranzfenster ist.
REF	Messsystem (Sensor) referenzieren (nur wenn kein Absolutmesssystem vorhanden ist).
SENSOR_SOURCE=*<ident>*	Angabe der Quelle für das Sensorsignal **[ab V3.1.3080.12 bzw. V3.1.3107.45]**. Folgende Quellen können für die achsspezifische Abstandsregelung eingestellt werden. Gültige Kennungen: DEFAULT: Ist als Sensorquelle „DEFAULT“ ausgewählt, stellt die CNC intern automatisch auf Sensorquelle „SECOND_ENCODER“ VARIABLE: Die Übergabe des Sensorsignals an die CNC erfolgt über eine V.E.-Variable. Zusätzlich muss hierfür der Name der V.E.-Variablen über den Parameter „SENSOR_VAR“ angegeben werden. SECOND_ENCODER: Es ist zu beachten, dass der 1. konfigurierte Geber (P-AXIS-00823) für die Lageregelung der Achse verwendet wird, der 2. Geber (P-AXIS-00824) für die Abstandsregelung.
SENSOR_VAR=..	Name der V.E.-Variablen über die das Sensorsignal an die CNC übermittelt wird. **[ab V3.1.3080.12 bzw. V3.1.3107.45]**
VAL1=..-VAL5=..	Fünf frei belegbare Werte im Realformat.
KP=..	Gewichten des Ausgabewertes der Abstandsregelung. Die Parametrierung kann analog zu P-AXIS-00759 durchgeführt werden. Der Wertebereich ist auf 0.0 < KP <= 2.0 beschränkt. Bei KP-Werten kleiner 1.0 wird die Dynamik der Abstandsregelung reduziert, bei KP-Werten größer als 1.0 wird die Dynamik erhöht. Durch einen KP-Faktor kleiner 1 kann ein mögliches Überschwingen der Abstandsregelung reduziert und bei kleinen Abstandsfehlern die Regelung beruhigt werden. **[ab V2.11.2809.06 bzw. V3.1.3079.06]**
I_TN=..	Nachstellzeit des Integral-Anteils des PID-Reglers in [s]. Die Nachstellzeit gibt an, nach welcher Zeit der P- und I-Anteil der Stellgröße gleich groß sind. Die Parametrierung kann nach Vorbild von P-AXIS-00764 durchgeführt werden. Der Wertebereich ist auf 0.0 <= I_TN <= 50.0 beschränkt. Eine große Nachstellzeit führt zu einer robusteren Regelung. Je kleiner die Nachstellzeit, desto stärker der I-Anteil und desto schneller die Regelung. Eine kleine Nachstellzeit regt Überschwingen stärker an. **[ab V2.11.2809.06 bzw. V3.1.3079.06]**
D_TV=..	Vorhaltezeit des Differential-Anteils des PID-Reglers in [s]. Die Vorhaltezeit gibt an, nach welcher Zeit der P- und D-Anteil der Stellgröße gleich groß sind. Die Parametrierung kann nach Vorbild von P-AXIS-00765 durchgeführt werden. Der Wertebereich ist auf 0.0 <= D_TV <= 2.0 beschränkt. Je größer die Vorhaltezeit, desto stärker der D-Anteil. **[ab V2.11.2809.06 bzw. V3.1.3079.06]**
FILTER_TYPE=..	Filtertyp für die Filterung der Geberwerte gemäß P-AXIS-00782. **[ab V3.1.3079.23]**
N_CYCLES=..	Anzahl der Messwerte, die für die Filterung verwendet werden gemäß P-AXIS-00413. **[ab V3.1.3079.23]**
FG_F0=..	Grenzfrequenz für den Tiefpassfilter in [Hz] gemäß P-AXIS-00508. **[ab V3.1.3079.23]**
ORDER=..	Ordnung des Tiefpassfilters gemäß P-AXIS-00507. **[ab V3.1.3079.23]**
SMOOTH_FACT=..	Glättungsfaktor des exponentiellen Mittelwertfilters gemäß P-AXIS-00784. Gibt die Gewichtung des aktuellen Messwertes an.
KALMAN_SIGMA=..	Unsicherheit der aufgenommenen Messwerte gemäß P-AXIS-00783. **[ab V3.1.3079.23]**
\	Trennzeichen ("Backslash") für übersichtliche Programmierung des Befehls über mehrere Zeilen.

Hinweis

Eine bei Programmende noch aktive Abstandsregelung wird nicht automatisch abgewählt.

Bei Reset oder Achsfehler wird eine aktive Abstandsregelung immer automatisch ausgeschaltet.

Hinweis

Die Parameter des PID-Reglers werden nach Programmende nicht zurückgesetzt.

Programmierbeispiel

Programmierbeispiele zur Abstandsregelung

```
%DIST_1
;Erwartete Position der Werkstückoberfläche setzen
N10 Z[DIST_CTRL SET_POS=30]
N20 Z[DIST_CTRL ON]          ;Anwahl
; …
Nxx Z[DIST_CTRL OFF]         ;Abwahl
N999 M30


%DIST_2
;Anwahl + erwartete Position der Werkstückoberfläche setzen
N10 Z[DIST_CTRL ON SET_POS=30]
; …
Nxx Z[DIST_CTRL FREEZE]      ;Position halten
; …
Nxx Z[DIST_CTRL OFF]         ;Abwahl
N999 M30


%DIST_3
;Anwahl + erwartete Position der Werkstückoberfläche setzen
N10 Z[DIST_CTRL ON SET_POS=50]

;Abstandsregelung ausschalten, die Z-Achse bewegt sich dabei nicht
Nxx Z[DIST_CTRL OFF NO_MOVE]
;Der entstandene Korrekturoffset wird beim Fahren auf die Zielposition
;100 mit berücksichtigt
Nxx G0 Z100
N999 M30


%DIST_4
;Setzen der Abstandsparameter
N10 Z[DIST_CTRL SET_POS=30]
;Anwahl bei Vorgabe der Position der Werkstückoberfläche (SET_POS)
N20 Z[DIST_CTRL ON]
; …
Nxx Z[DIST_CTRL OFF]         ;Abwahl
```

```
; …
;Anwahl bei Vorgabe der Werkstückoberfläche (SET_DIST)
Nxx Z[DIST_CTRL SET_DIST=10]
Nxx Z[DIST_CTRL ON CONST_DIST]
; …
Nxx Z[DIST_CTRL OFF]          ;Abwahl
N999 M30


%DIST_5
;Auswahl des Filtertyps
N10Z[DIST_CTRL FILTER_TYPE=KALMAN_MA]
;Parametrierung des Filters
N20 Z[DIST_CTRL N_CYCLES=30 KALMAN_SIGMA=1000]
;Überprüfen der Filterwirkung auf das Sensorsignal
N30 Z[DIST_CTRL DRYRUN]
;…
;Parametrieren des PID-Reglers
Nxx Z[DIST_CTRL KP=0.3 I_TN=0 D_TV=0.01]
;Aktivieren der Abstandsregelung
Nxx Z[DIST_CTRL ON CONST_DIST SET_DIST=1]
; …
;Wechsel des Filters
Nxx Z[DIST_CTRL FILTER_TYPE=KALMAN_EXPO SMOOTH_FACT=0.3]
; …
Nxx Z[DIST_CTRL OFF]         ;Abwahl
N999 M30
```

21.3 Programmierbarer Achsoverride (OVERRIDE)

Mit dieser Funktion kann der Achsvorschub im NC-Programm, wenn erforderlich, für Vorschub und Eilgangsätze unterschiedlich beeinflusst werden. Der achsspezifische programmierte Overridewert ist bei Bahnbewegungen wirksam, wenn sich die entsprechende Achse bewegt. Die Wirkungsweise der Echtzeitbeeinflussungen des Vorschubs über die PLC bleibt davon unberührt.

Als weitere Funktion steht auch ein programmierbarer Bahnoverride [▶ 464] zur Verfügung.

Bei mehreren in einem NC-Satz bewegten Achsen mit unterschiedlichen achsspezifischen Overridewerten wirkt immer der kleinste Override. Ist zusätzlich auch noch ein Bahnoverride definiert, so ergibt sich der wirksame Override aus der Multiplikation der beiden Overridewerte.

Hinweis

Die Funktion G166 [▶ 184] unterdrückt die Wirkung der programmierten Overridewerte.

Syntax:

<Achsname> **[OVERRIDE FEED_FACT=.. RAPID_FACT=.. { \ }]**

<Achsname>	Name der Achse
OVERRIDE	Kennung für die achsspezifische Overrideprogrammierung. Muss immer als erstes Schlüsselwort programmiert sein.
FEED_FACT=..	Overridefaktor für Vorschubsätze in [0.1%-200%]
RAPID_FACT=..	Overridefaktor für Eilgangsätze in [0.1%-200%]
\	Trennzeichen ("Backslash") für übersichtliche Programmierung des Befehls über mehrere Zeilen

Programmierbeispiel

Programmierbarer Achsoverride

```
%ax_override

N10  G01 X100 Y100 Z100 F1000
N40  X[OVERRIDE FEED_FACT=20 RAPID_FACT=60] ;Override X G01 20%, G00 60%
N50  Y[OVERRIDE FEED_FACT=30 RAPID_FACT=70] ;Override Y G01 30%, G00 70%
N60  Z[OVERRIDE FEED_FACT=40 RAPID_FACT=80] ;Override Z G01 40%, G00 80%
N50  G00 X0                    ;G00 Bewegung mit 60% Override
N60  Y0                        ;G00 Bewegung mit 70% Override
N70  Z0                        ;G00 Bewegung mit 80% Override
N80  G01 X100 F2000            ;G01 Bewegung mit 20% Override
N90  Y100                      ;G01 Bewegung mit 30% Override
N100 Z100                      ;G01 Bewegung mit 40% Override
N110 X200 Y200                 ;G01 Bewegung mit 20% Override
N120 X300 Y300 Z200            ;G01 Bewegung mit 20% Override
M30
```

21.4 Programmierbare Beschleunigungsüberlast (DYNAMIC)

Aus technologischen Gründen kann es in Verbindung mit konturbeeinflussenden Verfahren erforderlich sein, den Antrieb über die vorgegebenen Dynamikgrenzwerte hin zu belasten, um z.B. eine konstante Bahngeschwindigkeit in Polynomkonturen zu gewährleisten.

Über den nachfolgenden achsspezifischen Befehl kann in Verbindung mit dem zugeordneten Parameter P-AXIS-00394 die Dynamik der Achse im Programmkontext über die zulässige Maximalbeschleunigung P-AXIS-00008 hinaus in Prozent % gewichtet werden. P-AXIS-00394 stellt dabei die zulässige Obergrenze für den Beschleunigungsgewichtungsfaktor der Achse in Promille ‰ dar. Der Gewichtungsfaktor bezieht sich auf die Vorschubdynamikgrenzwerte des entsprechenden aktiven Slope-Profiles.

Aktuell kann die Funktion zur Gewichtung der Beschleunigung in Verbindung mit dem Überschleifverfahren 6 verwendet werden.

Syntax:

<Achsname> **[DYNAMIC DIST_SOFT | ACC_FACT**=.. { \ } **]**

<Achsname>	Name der Achse
DYNAMIC	Kennung für die Dynamikgewichtung der Achse. Muss immer als erstes Schlüsselwort programmiert sein.
DIST_SOFT	Kennung für das Polynomüberschleifverfahren 6
ACC_FACT=..	Achsspezifischer Gewichtungsfaktor in [%]
\	Trennzeichen ("Backslash") für übersichtliche Programmierung des Befehls über mehrere Zeilen

Hinweis

Der minimale Gewichtungswert ist 100%!
Der maximale Gewichtungswert wird auf P-AXIS-00394 begrenzt.

Programmierbeispiel

Programmierbare Beschleunigungsüberlast

```
%dynamic
N10 #SLOPE[TYPE=STEP]
N20 #CONTOUR MODE[DIST_SOFT PATH_DIST=35 ACC_MAX=100 ]
N30 C[DYNAMIC DIST_SOFT ACC_FACT=200]
;Beschleunigungsüberschreitungsfaktor für C-Achse 200%
N30 G1 G91 G261
N40 X59.485 F10000
N50 X105.172 C26.992
N60 X113.189 C46.171
N70 X100.348 C-46.171
N80 X99.179 C-26.992
N90 G260 X138.799
N100 G261
M30
```

21.5 Synchronisieren einer Achse auf Bahnverbund (SYNC IN / OUT)

Versionshinweis

Die Funktionalität ist verfügbar ab CNC-Version V2.11.2013.22

Bestimmte Prozesse erfordern eine synchrone Bewegung einer einzelnen Achse (Slaveachse) mit einem Bahnverbund. Dabei wird an bestimmten programmierten Positionen gefordert, dass sich die Slaveachse an einer bestimmten Position mit einer bestimmten Geschwindigkeit befindet. Die Slaveachse fährt dann mit der Synchrongeschwindigkeit, bis die Synchronisation wieder aufgehoben wird.

Typische Applikationsbeispiele sind bei Maschinen mit durchlaufendem Endlosmaterial zu finden. Hier muss das Material an einer bestimmten Stelle abgeschnitten werden, während der Bahnverbund weiterfährt. Das rotierende Messer muss zu einer bestimmten Masterposition (Werkstücklänge) in Schnittposition sein und sich dann mit gleicher Geschwindigkeit mitbewegen, bis der Schnitt erfolgt ist.

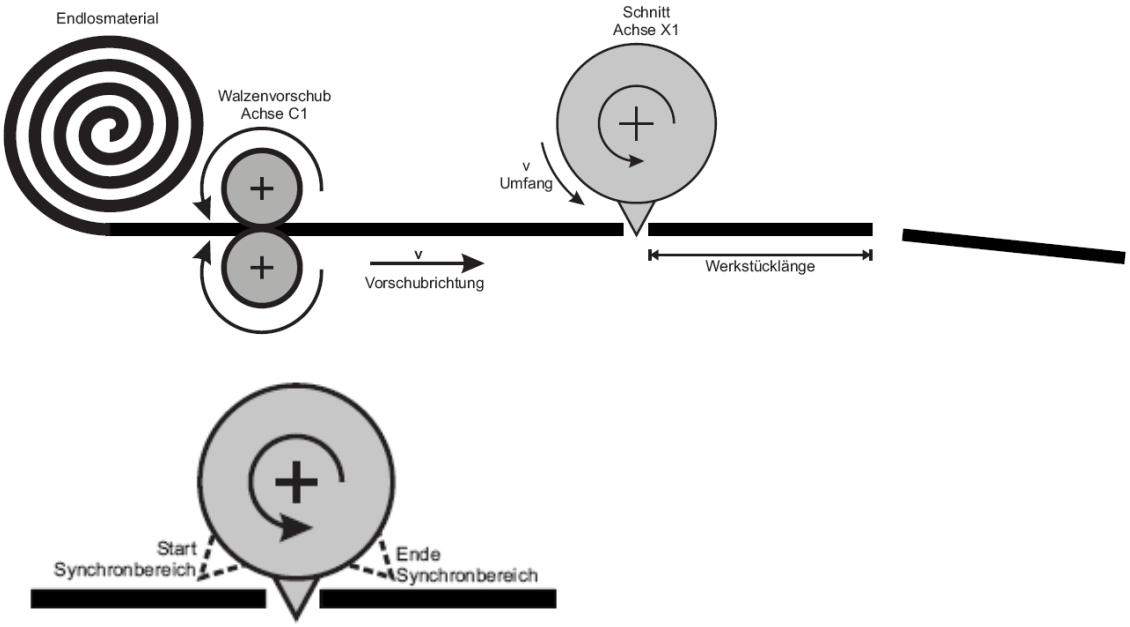


Abb. 209: Synchronisiertes Schneiden

Einschränkungen:

Eine Achse kann nicht synchronisiert werden, wenn:

- diese Achse momentan im Bahnverbund fährt

Konfiguration:

Für die Benutzung der Funktionalität muss in der Hochlaufliste ([STUP]) folgende Einstellung vorgenommen werden:

configuration.channel[0].path_preparation.function FCT_DEFAULT | **FCT_SYNC**

configuration.channel[0].interpolator.function FCT_IPO_DEFAULT | **FCT_SYNC**

Syntax zur Programmierung von Synchronbewegungen:

<Achsname> **[SYNC IN | OUT G90 | G91 G00 | G01 FEED**=.. **FEED_MAX_WEIGHT=.. POS=.. DIST=..** { \ } **]**

<Achsname>	Name der zu synchronisierenden Achse
SYNC	Kennung für eine Synchronbewegung der Achse. Muss immer als erstes Schlüsselwort programmiert sein.
IN	Kennung um Beginn der Synchronbewegung zu markieren.
OUT	Kennung um Ende der Synchronbewegung zu markieren.
G90 / G91	Absolut- / Relativmaß
G00 / G01	Eilgang- / Linearinterpolation
FEED=..	Achsspezifischer Vorschub in [mm/min, m/min, inch/min]
FEED_MAX_WEIGHT=..	Gewichtungsfaktor in [%], bezogen auf den achsspezifischen maximalen Vorschub P-AXIS-00212. Es sind nur Gewichtungswerte kleiner 100% erlaubt (gemäß G194 [▶ 155]).
POS=..	Achsposition in [mm, inch], an der die Synchrongeschwindigkeit erreicht wird.
DIST=..	Distanz in [mm, inch], auf der die Synchrongeschwindigkeit gefahren wird.
\	Trennzeichen ("Backslash") für übersichtliche Programmierung des Befehls über mehrere Zeilen.

Programmierbeispiel

Synchronisieren einer Achse auf Bahnverbund

```
%sync
N010 G90 X0 Y0 Z0 A0
N020 G91 F5000
N030 X=67.913 A[SYNC IN G01 FEED_MAX_WEIGHT=100 G91 POS=130 DIST=70]
              ;A-Achse erreicht max. Geschwindigkeit an Position 130,
              ;während X genau an dieser Stelle Position 67.913 erreicht

N040 X=1.5    ;A und X-Achse fahren synchron, dabei wird Geschwindigkeit
              ;der X-Achse so bestimmt, dass die X-Achse 3mm zurücklegt
N050 X=1.5    ;während die A-Achse 70° mit Max.geschwindigkeit fährt

N060 X=14.541 A[SYNC OUT G91 G0 POS160] G261
              ;Am Anfang dieses Satzes wird Synchronbewegung aufgelöst
              ;Die Bahnachsen fahren wieder mit programmiertem Vorschub,
              ;die A-Achse fährt unabhängig auf die angegebene Position

N070 X=15.862 Z=1.248 Y=0.185
N080 X=15.992 Z=1.889 Y=0.213
N090 X=32.243 Z=3.306 Y=0.482
N100 X=22.186
N110 X=31.696 Z=-2.597 Y=-0.389
N120 X=25.297 Z=-3.846 Y=-0.491
N130 X=39.819 A[SYNC IN G01 FEED_MAX_WEIGHT=100 G91 POS=130 DIST=70]
N140 X=1.257
N150 X=1.257
N160 X=200    A[SYNC OUT G91 G0 POS160]
N180 M30
```

21.6 Programmierung eines Achspolynoms (POLY)

Versionshinweis

Die Funktionalität ist verfügbar ab CNC-Version V2.11.2016.08

Eigenschaften der achsspezifischen Polynome

Die Bewegungsvorschrift einer Achse kann durch achsweise definierte Polynome programmiert werden.

Diese achsspezifische Polynombewegung ist für Linearbewegungen (G00, G01) programmierbar. Dabei werden die Dynamikparameter der jeweils aktiven G-Funktion (G00 bzw. G01) verwendet.

Pro Achspolynom wird eine obere Grenze des Polynomparameters angegeben, bis zu der der Polynomparameter interpoliert wird. Falls die obere Grenze nicht programmiert wurde, wird ihr der Wert 1.0 zugewiesen.

Die Polynomkoeffizienten eines Achspolynoms werden in eckigen Klammern nach dem Achsbezeichner in aufsteigender Reihenfolge angegeben. Zur Kennzeichnung ist das Schlüsselwort **POLY** immer als Erstes anzugeben. Nicht erforderliche höherwertige Polynomkoeffizienten können weggelassen werden. Den nicht programmierten Koeffizienten wird der Wert 0 zugewiesen. Zumindest der erste Koeffizient 'A0' muss angegeben werden.

Der maximal mögliche Grad des Polynoms ist 5.

Auswertung

Die Polynomkoeffizienten beziehen sich auf die Angabe absoluter Achspositionen und werden in ein Polynom 5. Grades eingesetzt.

$p(s) = A0 + A1 * s + A2 * s^2 + A3 * s^3 + A4 * s^4 + A5\ s^5$

Der Polynomparameter wird für das Polynom von Null bis zur programmierten oberen Grenze simultan zum zurückgelegten Bahnfahrweg durchinterpoliert.

Damit gilt für die absolute Achsposition der Polynomachse in mm bzw. Grad:

Bei Bewegungsanfang ($s = 0$):

$$p(0) = A0$$

Bei Bewegungsende ($s = L$):

$$p(L) = A0 + A1 * L + A2 * L^2 + A3 * L^3 + A4 * L^4 + A5\ L^5$$

Die achsspezifische Polynomprogrammierung ist satzweise wirksam, sie muss also, falls erforderlich, im nächsten Bewegungssatz erneut für die jeweilige Achse verwendet werden.

Hinweis

Bei der Bestimmung der Polynomkoeffizienten ist darauf zu achten, dass die Achsposition stetig ist. D.h. der Polynomwert an der Stelle 0 muss der Position der Achse aus dem vorhergehenden Bewegungssatz entsprechen.

Bei der wiederholten Programmierung von Achspolynomen in aufeinanderfolgenden NC-Zeilen muss die Endposition eines Polynoms der Startposition des Folgepolynoms entsprechen.

Da bei einem Polynom der Wert an der Stelle 0 nur von dem Koeffizienten A0 bestimmt wird, gilt: A0 = Position der Achse aus dem vorhergehenden Bewegungssatz.

Programmierung

Schema:

Achse [POLY L<*Maximalwert Polynomparameter*> A0 A1 A2 A3 A4 A5]

Beispiel:

X [POLY L=1.0 A0=0.1 A1=0.2 A2=0.3 A3=0.4 A4=0.5 A5=0.6]

Im gleichen NC-Satz ist eine gemischte Programmierung von Linearbewegungen und einem* achsspezifischen Polynom möglich. In die programmierten Polynompositionen werden evtl. aktive Verschiebungen (G54, G92, #PSET...) mit eingerechnet.

Für die programmierte Polynomachse wird keine Dynamikplanung/-überwachung durchgeführt. Eine sollwertseitige Überwachung der Softwareendlagen findet ebenfalls nicht statt. (Nur istwertseitige Endlagenüberwachung).

Syntax:

<*Achsname*> **[POLY** [**L**=..] **A0**=.. [**A1**=.. **A2**=.. **A3**=.. **A4**=.. **A5**=..] **]**

<*Achsname*>	Name der Polynomachse
POLY	Kennung für die Polynomprogrammierung der Achse. Muss immer als erstes Schlüsselwort programmiert sein.
L=..	Obere Grenze des Polynomparameters der zu bewegenden Achse ohne Einheit (optional, wenn nicht programmiert erhält L den Wert 1.0, programmierter Wert muss > 0 sein)
A0=..	Erster Polynomkoeffizient, muss programmiert werden (Startwert des Polynoms)
A1=.. - A5=..	Zweiter bis sechster Polynomkoeffizient, x, x^2, x^3, x^4, x^5 (optional, nicht programmierte Koeffizienten sind per Default mit 0 belegt)

Programmierbeispiel

Programmierung eines Achspolynoms

```
;C-Achse, Polynomparameter L = 0.7 A0 = 0.1, A1 = 0.3, A2 = 0.5
Nxx C0.1
Nxx C[POLY L=0.7 A0=0.1 A1=0.3 A2=0.5]

;X-Achse, Polynomparameter L = 0.3, A0 = 0.2, A1= 0.5
Nxx X0.2
Nxx X[POLY L=0.3 A0=0.2 A1=0.5]

;Minimalprogrammierung ohne Polynomparameter (Default L1),
;nur A0-Koeffizient
Nxx X0.2
Nxx X[POLY A0=0.2]

;Gemischte Programmierung von Linearbewegung und Achspolynom
Nxx C0.1
Nxx G01 F1000 X100 Y150 C[POLY L=0.7 A0=0.1 A1=0.3 A2=0.5]

;Hinweis: Die Gleichheitszeichen zw. Schlüsselwort und Wert
;sind optional
```

21.7 Setzen einer Achsposition im Kanal (SET_POSITION)

Versionshinweis

Diese Funktionalität ist verfügbar ab der CNC-Version V2.11.2808

Mit diesem Befehl wird die aktuelle Position einer Achse im NC-Kanal auf einen definierten Wert gesetzt. Dieser Wert wirkt auf ACS-Ebene (im Lageregler). Hierbei wird keine Bewegung ausgeführt, sondern im Lageregler wird nach Umsetzung der Achsposition die Achse als referenziert markiert. Danach erfolgt die Initialisierung des NC-Kanals mit den neuen Achspositionen, wobei aktive Verschiebungen berücksichtigt werden.

Die Vorgabe der neuen Achsposition erfolgt entweder als absoluter Wert (POS) oder als relativer Wert zur aktuellen Position (OFFSET).

Syntax:

<*Achsname*> **[SET_POSITION POS**=.. | **OFFSET**=.. { \ } **]**

<*Achsname*>	Name der Achse
SET_POSITION	Kennung für die Funktionalität des Setzens einer Achsposition. Muss immer als erstes Schlüsselwort programmiert sein.
POS=..	Neu definierte absolute Achsposition in [mm, inch]
OFFSET=..	Relativer Versatz zur aktuellen Achsposition in [mm, inch]
\	Trennzeichen (Backslash") für übersichtliche Programmierung des Befehls über mehrere Zeilen

Programmierbeispiel

Setzen einer Achsposition

```
%set_pos.nc
N010 G01 F2000 X0 Y0 Z0 A0 B0 C0
N020 $FOR P1=0,100,1
N030 G91 X100 ;Achse X fährt auf 10000mm
N040 $ENDFOR
N050 X[SET_POSITION POS=100] ;Setzen der X-Achsposition auf 100
:
:
N999 M30
```

21.8 Abheben / Senken einer Achse (LIFT)

Weitergehende Informationen können der Funktionsbeschreibung "Kollisionsvermeidung durch LIFT-Funktion" [FCT-A11] entnommen werden.

Satzübergreifendes Abheben/Senken

Die Programmierung orientiert sich an der Syntax für unabhängige Achsen. Beim Start des Abhebens/Senkens können die entsprechenden Parameter programmiert werden. Diese sind nicht haltend, d.h. sie werden, falls erforderlich, bei jedem Start neu gesetzt.

Syntax:

<*Achsname*> **[LIFT_START** [**DOWN**] [**G90** | **G91**] [**POS**=..] **POS_LIMIT**=.. **]**

<*Achsname*>	Name der Liftachse
LIFT_START	Kennung für den Start der (satzübergreifenden) unabhängigen Abhebebewegung der Achse. Muss immer als erstes Schlüsselwort programmiert sein.
DOWN	Über DOWN kann die Richtung der Achsbewegung invertiert werden, d.h. die Bewegung geht in Richtung des negativen Softwareendschalters. Die Standardrichtung ohne Angabe ist in Richtung des positiven Softwareendschalters. Nur bei Advanced-Lifting verfügbar
G90 / G91	Absolut- / Relativmaß, das Standardmaß ist G90. G91 ist nicht haltend, sondern nur für die Abhebe/Senkbewegung wirksam.
POS=..	Zielposition der Liftachse nach der Abhebebewegung in [mm, inch]. Standard ist die aktuelle Sollposition der Achse (siehe V.A.ABS.<*Achsname*>).
POS_LIMIT=..	Maximale Anhebehöhe bzw. Absenktiefe in [mm, inch]

Syntax:

<*Achsname*> **[LIFT_END]**

<*Achsname*>	Name der Liftachse
LIFT_END	Kennung für das Ende der (satzübergreifenden) unabhängigen Abhebebewegung der Achse.

Programmierbeispiel

Satzübergreifendes Abheben/Senken

```
N10 X10 Y20 Z30     ;Schneiden mit Laser
N20 M5              ;Laser aus
N30 Z[LIFT_START POS=12 POS_LIMIT=100]    ;Z-Achse abheben
N30 G01 X.. Y..
N40 G02 X.. Y..
N50 G03 X.. Y..
N60 G01 X.. Y..
N70 Z[LIFT_END]     ;Z-Achse absolut absenken auf Ziel 12 mm
N80 M4              ;Laser ein
N90 X20 Y20 ...

N10 X10 Y20 Z30
N30 Z[LIFT_START POS=12 POS_LIMIT=100] ;Z-Achse abheben
N40 G01 X.. Y..
N50 G01 X.. Y..
N60 Z[LIFT_END]     ;Z-Achse absolut absenken auf Ziel 12 mm
N70 X100

;alternative Programmierung
N110 X10 Y20 Z30
N140 G01 X.. Y.. Z[LIFT_START POS=12 POS_LIMIT=100]
N150 G01 X.. Y.. Z[LIFT_END]
N170 X100
```

Abheben/Senken in einem NC-Satz

Die Programmierung orientiert sich an der Syntax für unabhängige Achsen. Beim Start des Abhebens/Senkens können die entsprechenden Parameter programmiert werden. Diese sind nicht haltend, d.h. sie werden, falls erforderlich, bei jedem Start neu gesetzt.

Syntax:

<*Achsname*> **[LIFT** [**DOWN**] [**G90** | **G91**] [**POS**=..] **POS_LIMIT**=.. **]**

<*Achsname*>	Name der Liftachse
LIFT	Kennung für den Start und das Ende der unabhängigen Abhebebewegung der Achse im aktuellen NC-Satz. Muss immer als erstes Schlüsselwort programmiert sein.
DOWN	Über DOWN kann die Richtung der Achsbewegung invertiert werden, d.h. die Bewegung geht in Richtung des negativen Softwareendschalters. Die Standardrichtung ohne Angabe ist in Richtung des positiven Softwareendschalters. Nur bei Advanced-Lifting verfügbar
G90 / G91	Absolut-/Relativmaß. Das Standardmaß ist G90. G91 ist nicht haltend, sondern nur für die Abhebe/Senkbewegung wirksam.
POS=..	Zielposition der Liftachse nach der Abhebebewegung in [mm, inch]. Standard ist die aktuelle Sollposition des Achse (siehe V.A.ABS.<*Achsname*>).
POS_LIMIT=..	Maximale Abhebehöhe bzw. Absenktiefe in [mm, inch]

Programmierbeispiel

Abheben/Senken in einem NC-Satz

```
; einzeilige Programmierung
N200 Z40
N240 X10 Y.. Z[LIFT POS=30 POS_LIMIT=300]
N250 X20 Y.. Z[LIFT POS=20 POS_LIMIT=300]
N260 X30 Y.. Z[LIFT POS=25 POS_LIMIT=300]
N270 X.. Y.. Z[LIFT POS=30 POS_LIMIT=300]
N280 X.. Y.. Z[LIFT POS=30 POS_LIMIT=300]
```

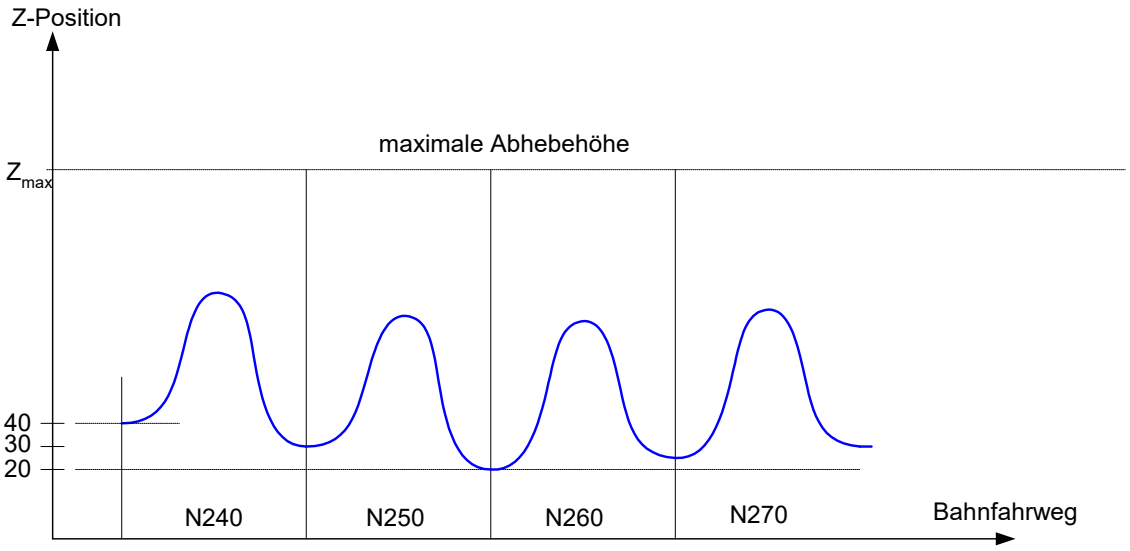


Abb. 210: Einzeiliges Abheben

21.9 Fahren auf Festanschlag (FIXED_STOP)

Weitergehende Informationen können der Funktionsbeschreibung"Fahren auf Festanschlag " [FCT-M8] entnommen werden.

Syntax:

<Achsname> **[FIXED_STOP** [**ON** | **OFF**] [**TORQUE_LIMIT**=..] [**POS_LAG_LIMIT**=..] [**CYCLES**=..] [**WINDOW**=..] [**START**=..] [**END**=..] [**ERR_NOT_DETECTED**=..] { \ } **]**

<Achsname>	Name der Achse, auf die die Funktion „Fahren auf Festanschlag“ angewendet werden soll.
FIXED_STOP	Kennung für die Funktionalität „Fahren auf Festanschlag“. Muss immer als erstes Schlüsselwort programmiert sein.
ON	Aktivieren der Funktion „Fahren auf Festanschlag“ für diese Achse. Für die Achse muss zusätzlich Verfahrinformation vorgegeben werden.
OFF	Deaktivieren der Funktion „Fahren auf Festanschlag“. Zusätzlich sollte für die Achse eine Bewegung programmiert sein, die vom Festanschlag wegführt.
TORQUE_LIMIT=..	Vorgabe des Drehmomentgrenzwerts beim Fahren auf den Festanschlag. Die Skalierung wird durch die Parametrierung der Funktion „Fahren auf Festanschlag“ aus den Achsparametern der Achse bestimmt (s. P-AXIS-00724). Normalerweise ist dies in Prozent [%] vom Nennmoment des Antriebs. Falls im NC-Programm für die Achse kein Drehmomentwert vorgegeben ist, wird der Standardwert aus dem Achsparameter P-AXIS-00729 des Drehmomentgrenzwerts verwendet.
POS_LAG_LIMIT=..	Grenzwert für den Schleppabstand in [mm, inch, °]. Wird dieser Grenzwert überschritten, wechselt die CNC nach den in CYCLES angegebenen LR-Zyklen in den Zustand „Festanschlag erreicht“. Falls im NC-Programm für die Achse kein Schleppabstandslimit vorgegeben wurde, wird der Standardwert aus dem Achsparameter P-AXIS-00712 verwendet.
CYCLES=..	Anzahl von Lagereglerzyklen, in denen der Schleppabstand über dem vorgegebenen Limit POS_LAG_LIMIT liegen muss, bevor die Steuerung in den Zustand „Festanschlag erreicht“ wechselt. Falls für die Achse im NC-Programm die Anzahl der Lagereglerzyklen nicht angegeben sind, wird der Standardwert aus dem Achsparameter P-AXIS-00714 verwendet.
WINDOW=..	Toleranzfenster für Position des Festanschlags in [mm, inch, °]. Nachdem der Festanschlag erreicht ist, prüft die Steuerung, ob die Istposition des Antriebs das vorgegebene Toleranzfenster verlässt, um ein Losbrechen des Anschlags zu detektieren. Falls im NC-Programm für die Achse kein Toleranzfenster angegeben ist, wird der Standardwert aus dem Achsparameter P-AXIS-00713 verwendet. Bei einem Wert von 0 ist die Überwachung abgeschaltet.
START=..	Die Überwachung auf Erreichen des Festanschlags kann über diesen Parameter um einen Prozentwert [%] bezogen auf den Bahnfahrweg verzögert werden, um beim Losfahren der Achse eine falsche Detektion eines Anschlags aufgrund von Reibung etc. zu verhindern. Falls im NC-Programm dieser Parameter nicht angegeben ist, erfolgt die Überwachung immer von Beginn der Bewegung (START = 0%)
END=..	Die Überwachung auf Erreichen des Festanschlags kann über diesen Parameter um einen Prozentwert [%] bezogen auf den Bahnfahrweg vorzeitig beendet werden, um beim Bremsen der Achse auf den Zielpunkt eine falsche Detektion eines Anschlags zu verhindern. Falls dieser Parameter im NC-Programm nicht angegeben ist, erfolgt die Überwachung immer bis zum Zielpunkt des Bewegungssatzes (END = 100%)

ERR_NOT_DETEC-TED=..	Mit diesem Parameter kann die Ausgabe der Fehlermeldung P-ERR-50886 bei nicht erfasstem Anschlag in der Anfahrbewegung unterdrückt werden. Dadurch können z.B. mit dem „Fahren auf Festanschlag“ einfache Messvorgänge ausgeführt werden. 0: Fehlermeldung P-ERR-50886 ausgeben. 1: Fehlermeldung P-ERR-50886 nicht ausgeben.
\	Trennzeichen (Backslash") für übersichtliche Programmierung des Befehls über mehrere Zeilen

21.10 Rotatorische Achsen

Versionshinweis

Diese Funktionalitäten sind verfügbar ab der CNC-Version V3.1.3079.40

Die Eigenschaften rotatorischer Achsen bzgl. der Überwachung und Festlegung von Softwareendschaltern und Modulogrenzen werden in den achsspezifischen Konfigurationslisten festgelegt. Nach Steuerungshochlauf bzw. Programmstart können diese Eigenschaften nicht mehr geändert werden. Bestimmte Applikationen erfordern jedoch aus technologischen Gründen die Änderung dieser Einstellungen im NC-Programm.

Zum Beispiel sind beim Drahterodieren rotatorische Achsen erforderlich, die sowohl als Moduloachsen als auch als Endlosdrehachsen betrieben werden können. Zusätzlich erfordern die Überwachung und Parametrierung der Softwareendschalter ebenfalls eine Änderung im NC-Programm. Hierzu stehen die folgenden Programmierbefehle zur Verfügung.

21.10.1 Programmierung der Softwareendschalterüberwachung (POS_LIMIT)

Die Grundeinstellung der Softwareendschalter (SWE) wird in den achsspezifischen Listen über die Parameter P-AXIS-00177 und P-AXIS-00178 konfiguriert. Nach Hochlauf der Steuerung werden diese Grenzwerte für translatorische und für jede rotatorische **nicht-modulo** Achse überwacht. Durch Setzen von P-AXIS-00705 kann die Wirksamkeit der Überwachung für jede Achse, unabhängig von Achstyp und Achsmode, aus- und eingeschaltet werden. Bei Moduloachsen sind die Grenzwerte nur wirksam, wenn sie innerhalb des Modulobereiches liegen.

Diese Grundeinstellungen können durch einen NC-Befehl geändert werden:

Syntax:

<*Achsname*> **[POS_LIMIT ON** | **OFF** | **DEFAULT** [**MIN**=.. **MAX**=..] [**BEHAVIOUR**=<*error_mode*>] { \ } **]**

<*Achsname*>	Name der Achse
POS_LIMIT	Kennung für die achsspezifische Programmierung der Softwareendschalterüberwachung. Muss immer als erstes Schlüsselwort programmiert sein.
ON	Anwahl der Softwareendschalterüberwachung mit neuen Grenzwerten. MIN, MAX sind optional, wenn nicht programmiert, sind die bisherigen Grenzwerte weiterhin gültig.
OFF	Abwahl der Softwareendschalterüberwachung
DEFAULT	Zurücksetzen auf die konfigurierten Standardwerte gemäß P-AXIS-00177, P-AXIS-00178 und P-AXIS-00705
MIN=..	Unterer Endschaltergrenzwert in [mm, inch, °]
MAX=..	Oberer Endschaltergrenzwert in [mm, inch, °]
BEHAVIOUR =<*error_mode*>	Festlegen des Fehlverhaltens beim Überfahren der Softwareendschalter: ERROR: Überfahren führt ab der Bahnplanung (Überwachung des Sollwertes) zu einem Fehler ERROR_LR: Ein Überfahren führt während der Bahnplanung zu einer Warnung. Im Lageregler (Überwachung des Istwertes) wird beim Überfahren ein Fehler ausgegeben. WARNING: Beim Überfahren werden in der Bahnplanung sowie im Lageregler nur Warnungen ausgegeben.
\	Trennzeichen ("Backslash") für übersichtliche Programmierung des Befehls über mehrere Zeilen

Hinweis

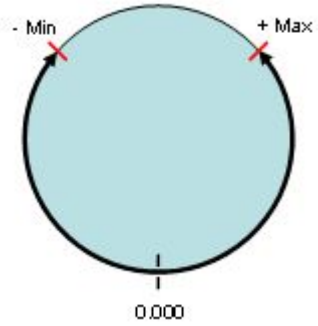
Die Überprüfung der Softwareendschalter erfolgt im Achskoordinatensystem. Der CNC-Kanal überprüft hierbei die Sollpositionen, während der Lageregler die Istpositionen prüft.

Hinweis

Die konfigurierten Grenzwerte P-AXIS-00177 und P-AXIS-00178 werden durch diesen Befehl nicht geändert. Bei CNC-Reset, Programmstart und Achstausch (z.B. #CALL AX) werden die Endschaltergrenzwerte auf die konfigurierten Standardwerte zurückgesetzt. Die Änderung der Endschaltergrenzwerte durch erneutes Einlesen von Achslisten wird beim nächsten Programmstart berücksichtigt.

Programmierbeispiel

Programmierung Softwareendschalterüberwachung




```
%pos_limit
:
;Anwahl SWE-Überwachung:
N100 A[POS_LIMIT ON MIN=-135 MAX=135 BEHAVIOUR=WARNING]
:
;Anwahl konfigurierte Standardwerte:
N200 A[POS_LIMIT DEFAULT]
:
;Abwahl SWE-Überwachung:
N300 A[POS_LIMIT OFF]
:
M30
```

21.10.2 Programmierung des Modulobereichs (MODULO)

Die Grundeinstellung der Modulogrenzen wird in den achsspezifischen Listen über die Parameter P-AXIS-00126 und P-AXIS-00127 konfiguriert. Nach Hochlauf der Steuerung sind diese Grenzwerte für rotatorische Achsen mit Achsmode MODULO und Spindeln aktiv. Bei Linearachsen ist die Modulorechnung in der Grundeinstellung ausgeschaltet. Durch Setzen von P-AXIS-00557 kann die Wirksamkeit der Modulorechnung für jede Achse, unabhängig von Achstyp und Achsmode, aus- und eingeschaltet werden.

Diese Grundeinstellungen können durch einen NC-Befehl geändert werden:

Syntax:

<*Achsname*> **[MODULO ON** | **OFF** | **OFF_POS_INIT** | **DEFAULT** | **SHIFT** [**MIN**=.. **MAX**=..] { \ } **]**

<*Achsname*>	Name der Achse
MODULO	Kennung für die achsspezifische Programmierung der Modulofunktion. Muss immer als erstes Schlüsselwort programmiert sein.
ON	Anwahl der Modulofunktion mit neuen Grenzwerten. MIN, MAX sind optional, wenn nicht programmiert, sind die bisherigen Grenzwerte weiterhin gültig.
OFF	Abwahl der Modulofunktion.
OFF_POS_INIT	Abwahl der Modulofunktion mit einer impliziten Positionsinitialisierung des NC-Kanals (siehe #CHANNEL INIT [CMDPOS] [▶ 179]).
DEFAULT	Zurücksetzen auf die konfigurierten Standardwerte gemäß P-AXIS-00126, P-AXIS-00127 und P-AXIS-00557.
SHIFT	Umrechnen der aktuellen Sollposition in den Modulobereich. Nach dem Umrechnen sind wieder die bisherigen Moduloeinstellungen gültig.
MIN=..	Untere Modulogrenze in [°]
MAX=..	Obere Modulogrenze in [°]
\	Trennzeichen ("Backslash") für übersichtliche Programmierung des Befehls über mehrere Zeilen

Wenn kinematische und/oder kartesische Transformationen im Kanal aktiv sind, beziehen sich die programmierten Modulogrenzen auf das Programmierkoordinatensystem (PCS) oder das Maschinen koordinatensystem (MCS). Ohne aktive Transformationen ist der Modulobereich im Achskoordinatensystem (ACS) definiert.

Im Lageregler ist die Moduloeigenschaft der Achse nach dem Hochlauf fix und kann durch die Programmierung nicht verändert werden.

Wenn der Modulobereich geändert oder aktiviert wird, erfolgt die Umrechnung der aktuellen Achsposition in den neuen Modulobereich.

Programmierbeispiel

Programmierung Modulofunktion

```
%modulo_calc_1
:
N100 A[MODULO ON MIN=0 MAX=360] ;Anwahl Moduloüberwachung
N110 A-+700                     ;in negativer Richtung auf Position 340°
N120 A[MODULO DEFAULT]          ;Anwahl konfigurierte Standardwerte
:
N200 A[MODULO OFF]
N210 X100 A2000                 ;Bewegung ohne Moduloberechnung
N220 A[MODULO ON MIN=0 MAX=360] ;Position 2000 auf mod(360)=200
N230 A-+700                     ;in negativer Richtung auf Position 340°
:
M30

;Programmierbeispiel für Moduloumrechnung (SHIFT):
%modulo_calc_2
:
N10 A[MODULO OFF]                  ;Freischalten Mehrfachdrehen
N20 X100 A2000                     ;A-Achse mehrfachdrehen ohne Modulo
N30 A[MODULO SHIFT MIN=0 MAX=360] ;Position 2000 einmalig auf
                                   ;mod(360)=200
N40 X500 A2000                     ;A-Achse mehrfachdrehen ohne Modulo
:
M30
```

22 2-Pfadprogrammierung

Die 2-Pfadprogrammierung ermöglicht die Beschreibung von 2 unabhängigen Konturen in einem NC-Kanal. Verwendet wird sie insbesondere beim Drahterodieren, sie kann aber auch bei anderen Fertigungsverfahren und Anwendungen eingesetzt werden.

Beim Erodieren schneidet ein von einer Rolle kontinuierlich ablaufender Draht über zwei 2 Führungselemente unterhalb und oberhalb des Werkstückes basierend auf dem Prinzip der Funkenerosion durch das Material.

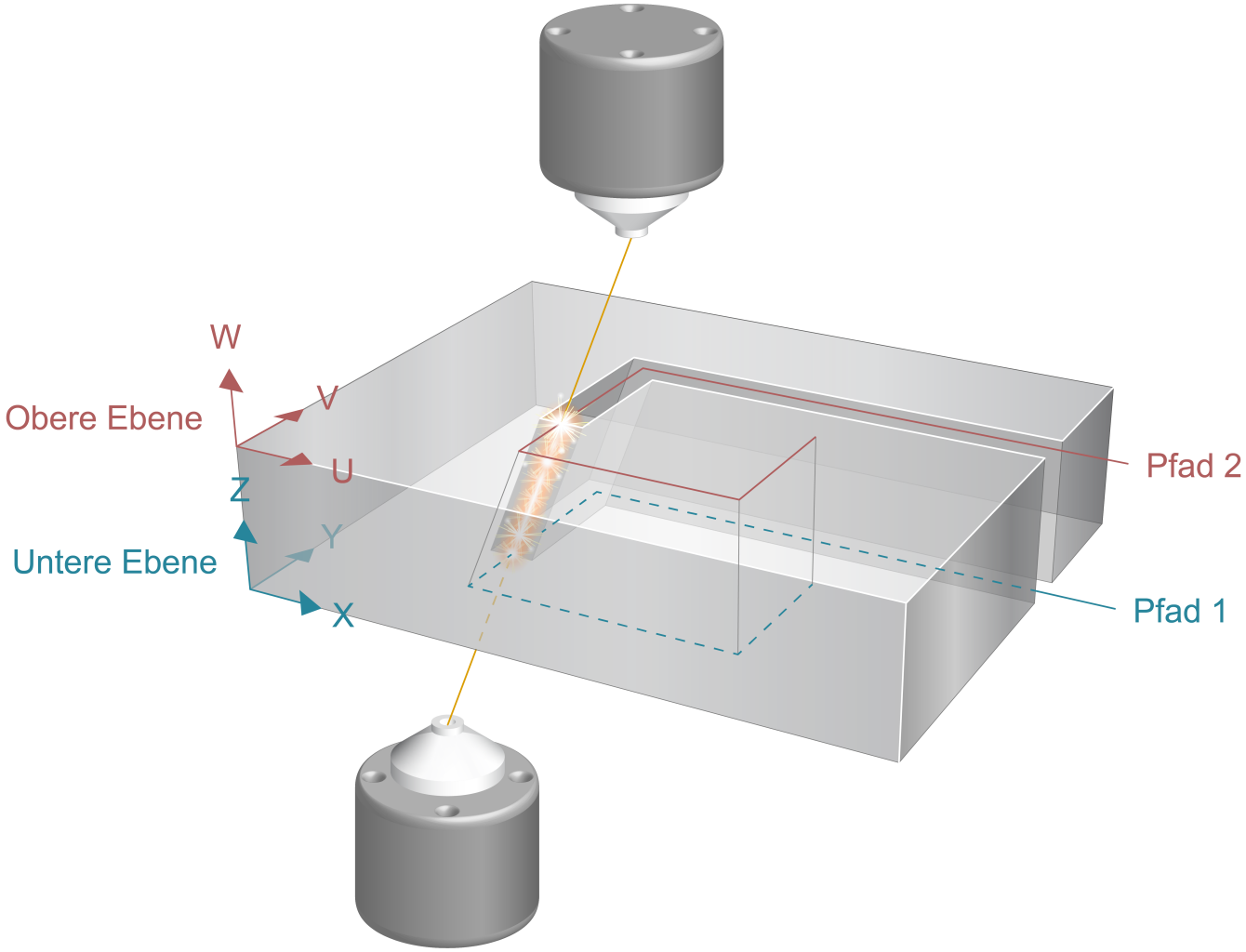


Abb. 211: Prinzipbild Drahterodieren mit 2-Pfadprogrammierung

Die Bewegung der unteren und oberen Führungselemente erfolgt jeweils auf eigenen Bahnen in Ebenen, die durch individuelle Koordinatensysteme definiert sind. Die Ebenen liegen in der Regel parallel aber in unterschiedlichen Höhen. Die Festlegung und Ausrichtung dieser Koordinatensysteme hängt hierbei von der zu schneidenden Materialdicke ab.

Sind die Bahnen bzw. Konturelemente der Führungspunkte identisch programmiert, so ergeben sich senkrechte Schnitte. Die Bahnen können aber auch unterschiedliche Konturen beschreiben, wodurch sich dann schräge Schnitte ergeben, die von einer Kontur in die andere Kontur übergehen. Die einander zugeordneten Konturelemente der unteren und oberen Pfade werden dabei immer im gleichen NC-Satz programmiert.

Die beiden Pfade werden in 2 Achsgruppen programmiert. Im Allgemeinen werden die Achsnamen X, Y, Z für den unteren Pfad verwendet und für den oberen die Namen U, V, W. Die Zuordnung zwischen Achsgruppe und Pfad ist in der Kanalliste konfigurierbar und nach der Initialisierung der CNC (Hochlauf) festgelegt.

Hinweis

Diese Funktionalität ist eine lizenzpflichtige Zusatzoption.

22.1 Konfiguration

Die Konfiguration der CNC für die 2-Pfadprogrammierung erfolgt in der Kanalparameterliste. Folgende Einstellungen sind erforderlich:

- Der Kanalparameter P-CHAN-00261 (multi_path_configuration) ist auf 1 gesetzt.
- Mindestens 6 Achsen (2x3 Bahnachsen) sind ohne Lücke konfiguriert.
- Optional: Technologiebedingt kann eine Zusatzachse als 7. Achse erforderlich sein, um notwendige Ausgleichsbewegungen berechnen zu können. So z.B. beim Drahterodieren, bei dem die Durchstoßpunkte des Drahtes mit der unteren und oberen Ebene auf die Bewegungen der Drahtführungen abgebildet werden. Die 7. Achse wird direkt ohne Lücke nach den 6 Bahnachsen konfiguriert.

Beispiel

2-Pfadkonfiguration mit 7 Achsen als Auszug aus der Kanalparameterliste:

```
multi_path_configuration 1

gruppe[0].bezeichnung IPO_1

gruppe[0].achs_anzahl 7

gruppe[0].achse[0].log_achs_nr 1
gruppe[0].achse[0].bezeichnung X

gruppe[0].achse[1].log_achs_nr 2
gruppe[0].achse[1].bezeichnung Y

gruppe[0].achse[2].log_achs_nr 3
gruppe[0].achse[2].bezeichnung Z

gruppe[0].achse[3].log_achs_nr 4
gruppe[0].achse[3].bezeichnung U

gruppe[0].achse[4].log_achs_nr 5
gruppe[0].achse[4].bezeichnung V

gruppe[0].achse[5].log_achs_nr 6
gruppe[0].achse[5].bezeichnung W

gruppe[0].achse[6].log_achs_nr 7
gruppe[0].achse[6].bezeichnung Z1
```

Die Achsen X, Y, Z mit Index [0..2] werden für die untere Bahninterpolation und die Achsen U, V, W mit Index [3..5] für die obere Bahninterpolation verwendet. Die Achse Z1 mit Index [7] wird für die notwendige Berechnung der Bewegungen der Drahtführungen benötigt.

22.2 Allgemeine 2-Pfadsyntax

Zur Sicherstellung der Zuordnung müssen zusammengehörende Konturelement der beide Pfade immer in einem NC-Satz programmiert werden. Als Kennzeichnung bzw. Abgrenzung der jeweiligen Pfadabschnitte wird das Separatorzeichen ':' verwendet.

Hinweis

Bei aktiver 2-Pfadprogrammierung (P-CHAN-00261) steht das ':'-Zeichen als Sprungzielmarkierung von Satznummern für $GOTO nicht mehr zur Verfügung. Sprungziele können aber weiterhin per Sprunglabel gesetzt werden.

Alle NC-Befehle vor dem ersten ':' - Separator im NC-Satz gelten für beide Pfade (Globaler Abschnitt).

Die Befehle nach dem ersten ':' – Separator sind dem ersten Pfad zugeordnet. Sie wirken nur auf der unteren Bahn und es sind auch nur die Achsnamen der unteren Ebene, also z.B. X, Y, Z zulässig.

Die Befehle nach dem zweiten ':' – Separator sind dem zweiten Pfad zugeordnet. Sie wirken nur auf der oberen Bahn und es sind auch nur die Achsnamen der oberen Ebene, also z.B. U, V, W zulässig.

Eine Achse der unteren Bahn darf nicht im Abschnitt der oberen Bahn programmiert werden und umgekehrt. Diese Fehlprogrammierung wird durch eine Fehlermeldung abgefangen (P-ERR-21591)

Soll die Zusatzachse in Kombination mit der 2-Pfadsyntax programmiert werden, so muss dies im globalen Abschnitt erfolgen.

Allgemein besteht die 2-Pfadsyntax aus drei Abschnitten:

Globale Befehle - unterer Pfad (Pfad 1) - oberer Pfad (Pfad 2)

Ist ein Abschnitt nicht erforderlich, so bleibt dieser Abschnitt leer. Die Abgrenzung der Abschnitte ist durch den ':' – Separator markiert.

Schema der 2-Pfadsyntax:

<Globaler Abschnitt> **:** *<Abschnitt Pfad1>* **:** *<Abschnitt Pfad2>*

Beispiel:

```
N.. G01 G90 F100 : X100 Y100 Z0 : U100 V100 W0
```

Sonderregel:

Wenn keine pfadspezifischen Angaben erforderlich sind, ist es auch möglich, die Achsbewegungen ohne Separatoren zu programmieren. Die Zuordnung der Achsen zum korrekten Pfad erfolgt dann über die Achsnamen.

Beispiel:

```
N.. G01 G90 X0 Y0 Z0 U0 V0 W0 F1000 ;Pfad 1:X,Y,Z Pfad 2:U,V,W
```

22.3 Globale und pfadspezifische Befehle

In der 2-Pfadprogrammierung können bestimmte CNC-Funktionen nicht verwendet werden. In den folgenden Kapiteln sind die Einschränkungen bzgl. der zulässigen Programmierkombinationen beschrieben.

22.3.1 G-Funktionen

Die folgenden G-Funktionen dürfen uneingeschränkt sowohl global als auch pfadspezifisch programmiert werden:

- G0, G1, G2, G3
- G90, G91
- G163

Generell nicht erlaubt sind bei konfigurierter 2-Pfadprogrammierung die nachfolgend aufgeführten G-Funktionen. Bei Programmierung erfolgt die Ausgabe einer Fehlermeldung (ID 21592).

- G20, G21, G22, G23
- G33
- G51, G52
- G61
- G95, G96, G97
- G150, G151
- G196
- G260, G261
- G301, G302
- G351

Alle weiteren G-Funktionen dürfen nur im globalen Abschnitt der 2-Pfadprogrammierung oder in einem eigenen NC-Satz programmiert werden. Bei pfadspezifischer Verwendung erfolgt die Ausgabe einer Fehlermeldung (ID 21593).

22.3.2 Sonstige Funktionen

Die folgenden Funktionen dürfen uneingeschränkt sowohl global als auch pfadspezifisch programmiert werden:

- Kreismittelpunkt I, J, K
- Kreisradius R

Der Vorschub (F-Wort) darf nur im globalen Abschnitt der 2-Pfadprogrammierung oder in einem eigenen NC-Satz programmiert werden. Die Interpolation bezieht hierbei den Vorschub immer auf den längeren Anteil der beiden programmierten Pfadabschnitte.

- Vorschub F

22.3.3 Zusatzfunktionen

Die nachfolgend aufgeführten #-Befehle dürfen bei konfigurierter 2-Pfadprogrammierung nicht programmiert werden. Bei Programmierung erfolgt die Ausgabe einer Fehlermeldung (ID 21595).

- #AKIMA ..
- #CAX ..
- #CYL
- #CYL OFF
- #CONTROL AREA ..
- #ECS ..
- #MCS ..
- #MCS TO WCS
- #WCS TO MCS
- #FACE ..
- #HSC ..
- #ROTATION ..
- #SPLINE ..
- #TOOL ORI CS

Alle weiteren #-Befehle dürfen bei konfigurierter 2-Pfadprogrammierung in einem eigenen NC-Satz programmiert werden. In Kombination mit einer pfadspezifischen Programmierung im gleichen NC-Satz erfolgt die Ausgabe einer Fehlermeldung (ID 21594).

22.3.4 M/H-Funktionen

Technologiefunktionen wie z.B. M-, H-Funktionen werden in beiden Pfaden als gemeinsame Funktionen behandelt. Die Schnittstelle zur SPS ist zwar kanal- oder achsspezifisch ausgelegt, aber beide Pfade werden im gleichen NC-Kanal interpoliert. Deshalb unterscheidet die Programmierung und Synchronisation mit der SPS nicht zwischen unterem und oberem Pfad.

Beispiel:

```
N.. F1000 G90 : G0 X400 M501 : G1 U500 M502
N.. M601
N.. : X49 : U300 M700]
```

22.3.5 Parameter und Variablen

Parameter (P) und Variablen (V.x) werden kanalspezifisch programmiert und in der Art und Weise globaler Befehle behandelt. Deshalb unterscheidet die Programmierung nicht zwischen unterem und oberem Pfad.

Beispiel:

```
N.. P1 = 200 P2 = 300
N.. F1000 G90 G01 : X100 V.E.Test=1 : U200 V.E.Test2=2
N.. : XP1 : UP2
N.. P2=400 : X300
N.. : G0 X400 : G1 U500 P2=300
N.. : P2=200 : U600
```

Programmierung des Werkzeugradius

Gleicher Werkzeugradius in beiden Pfaden

```
N.. V.G.WZR=0.15
```

alternativ

```
N.. V.G.WZR=0.15 : X100 Y20: U100 V20
```

Unterschiedliche Werkzeugradien in beiden Pfaden

```
N.. : V.G.WZR=0.139 : V.G.WZR=0.15
alternativ
N.. : V.G.WZR=0.139 X100 Y20 : V.G.WZR=0.15 U100 V20
```

Festlegen des Werkzeugradius im Referenzpfad

```
N.. : V.G.WZR=0.134 :
```

Festlegen des Werkzeugradius im zweiten Pfad

```
N.. : : V.G.WZR=0.151
```

22.3.6 Programmierbeispiele zur Syntax

Linearbewegungen

Beispiel 1:

Global: Linearbewegungen, Vorschub, Absolutpositionen

Pfad 1: Achspositionen

Pfad 2: Achspositionen

```
N.. G01 F10 G90 : X10 Y10 Z0 : U10 V10 W0
```

Beispiel 2:

Global: Vorschub

Pfad 1: Linearbewegung, Absolutpositionen, Achspositionen

Pfad 2: Linearbewegung, Absolutpositionen, Achspositionen

```
N.. F10 : G01 G90 X10 Y10 Z0 : G01 G90 U10 V10 W0
```

Beispiel 3:

Global: Vorschub

Pfad 1: Linearbewegung, **Absolutpositionen**, Achspositionen

Pfad 2: Linearbewegung, **Relativpositionen**, Achspositionen

```
N.. F10 : G01 G90 X10 Y10 Z0 : G01 G91 U10 V10 W0
```

Beispiel 4:

Global: Linearbewegungen, Vorschub, Absolutpositionen

Pfad 1: **Eilgang**, **Absolutpositionen**, Achspositionen

Pfad 2: Linearbewegungen, **Relativpositionen**, Achspositionen

```
N.. G01 F10 G90: G00 G90 X10 Y10 Z0 : G91 U10 V10 W0
```

Beispiel 5:

Global: Linearbewegungen, Vorschub, Absolutpositionen

Pfad 1: ---

Pfad 2: Achspositionen

```
N.. G01 F10 G90 : : U10 V10 W0
```

Beispiel 6:

Global: Linearbewegungen, Vorschub, Absolutpositionen

Pfad 1: Achspositionen

Pfad 2: ---

```
N.. G01 F10 G90 : X10 Y10 Z0
oder
N.. G01 F10 G90 : X10 Y10 Z0 :
```

Beispiel 7:

Global: Linearbewegungen, Vorschub, Absolutpositionen, **Achsposition Zusatzachse**

Pfad 1: Achspositionen

Pfad 2: Achspositionen

```
N.. G01 F10 G90 Z1=100 : X10 Y10 Z0 : U10 V10 W0
```

Zirkularbewegungen

Beispiel 1:

Global: Zirkularbewegungen, Vorschub, Absolutpositionen, Kreismittelpunkte

Pfad 1: Achspositionen Kreisendpunkt

Pfad 2: Achspositionen Kreisendpunkt

```
N.. G02 F10 G90 I10 J0 : X20 Y0 Z0 : U20 V0 W0
```

Beispiel 2:

Global: Zirkularbewegungen CW, Vorschub, Absolutpositionen, Kreisradius

Pfad 1: Achspositionen Kreisendpunkt

Pfad 2: Achspositionen Kreisendpunkt

```
N.. G02 F10 G90 R10 : X20 Y0 Z0 : U20 V0 W0
```

Beispiel 3:

Global: Zirkularbewegungen CW, Vorschub, Absolutpositionen

Pfad 1: Kreismittelpunkte, Achspositionen Kreisendpunkt

Pfad 2: Kreismittelpunkte, Achspositionen Kreisendpunkt

```
N.. G02 F10 G90 : I10 J0 X20 Y0 Z0 : I15 J0 U30 V0 W0
```

Beispiel 4:

Global: Vorschub, Absolutpositionen

Pfad 1: **Zirkularbewegung CW**, Kreismittelpunkte, Achspositionen Kreisendpunkt

Pfad 2: **Zirkularbewegung CCW**, Kreismittelpunkte, Achspositionen Kreisendpunkt

```
N.. F10 G90 : G02 I10 J0 X20 Y0 Z0 : G03 I15 J0 U30 V0 W0
```

Beispiel 5:

Global: Vorschub, Absolutpositionen

Pfad 1: **Zirkularbewegung CW**, **Kreismittelpunkte**, Achspositionen Kreisendpunkt

Pfad 2: **Zirkularbewegung CCW, Kreisradius**, Achspositionen Kreisendpunkt

```
N.. F10 G90 : G02 I10 J0 X20 Y0 Z0 : G03 R15 U30 V0 W0
```

Beispiel 6:

Global: Zirkularbewegungen CW, Vorschub, Absolutpositionen, Kreiradius (**G163**)

Pfad 1: Achspositionen Kreisendpunkt

Pfad 2: Achspositionen Kreisendpunkt

```
N.. G02 F10 G90 G163=10 : X20 Y0 Z0 : U20 V0 W0
```

Beispiel 7:

Global: Vorschub, Absolutpositionen

Pfad 1: **Zirkularbewegungen CW, Kreisradius (G163)**, Achspositionen Kreisendpunkt

Pfad 2: **Zirkularbewegungen CCW, Kreisradius**, Achspositionen Kreisendpunkt

```
N.. F10 G90 : G02 G163=10 X20 Y0 Z0 : G03 R15 U30 V0 W0
```

Kombination Linear/Zirkularbewegungen:

Beispiel 1:

Global: Vorschub, Absolutpositionen

Pfad 1: **Linearbewegung**, Achspositionen

Pfad 2: **Zirkularbewegung CCW, Kreisradius**, Achspositionen Kreisendpunkt

```
N.. F10 G90 : G01 X20 Y0 Z0 : G03 R15 U30 V0 W0
```

Beispiel 2:

Global: Vorschub, Absolutpositionen, Zirkularbewegung CW, Kreisradius

Pfad 1: **Linearbewegung**, Achspositionen

Pfad 2: Achspositionen Kreisendpunkt

```
N.. F10 G90 G02 R15 : G01 X20 Y0 Z0 : U30 V0 W0
```

Beispiel 3:

Global: Vorschub, Absolutpositionen, Linearbewegung, **Achsposition Zusatzachse**

Pfad 1: **Zirkularbewegung CW, Kreismittelpunkte**, Achspositionen Kreisendpunkt

Pfad 2: Linearbewegung, Achspositionen

```
N.. F10 G90 G01 Z1=100 : G02 I10 J0 X20 Y0 Z0 : U30 V0 W0
```

Beispiel 4:

Global: Vorschub, Absolutpositionen

Pfad 1: **Zirkularbewegung CW, Kreismittelpunkte**, Achspositionen Kreisendpunkt

Pfad 2: Linearbewegung, Achspositionen

```
N.. F10 G90 : G02 I10 J0 X20 Y0 Z0 : G01 U30 V0 W0
```

22.4 NC Programmbeispiel

```
%2path_cone

;Pfad 1: Quadrat mit runden Ecken
;Pfad 2: Halbkreise + Linien
;Pfad 2 ohne Bewegung an den Linienenden, wenn in Pfad 1 Achsen fahren

N0050 G00 G90 G60 X0 Y0 Z0 U0 V0 W0 Z1=200

N0090 X25 Y0 Z0 U5 V0 W0

N0110 G90 F20000

N0120 Z1=200 : G01 X25 Y-20 : G01 U5 V-5
N0130 Z1=200 : G02 X20 Y-25 R5 :
N0140 Z1=200 : G01 X-20 Y-25 : G02 U-5 V-5 R5
N0150 Z1=200 : G02 X-25 Y-20 R5 :
N0160 Z1=200 : G01 X-25 Y20 : G01 U-5 V5
N0170 Z1=200 : G02 X-20 Y25 R5 :
N0180 Z1=200 : G01 X20 Y25 : G02 U5 V5 R5
N0190 Z1=200 : G02 X25 Y20 R5 :
N0200 Z1=200 : G01 X25 Y0 : G01 U5 V0
N0210 X25 Y0 Z0 U5 V0 W0
N0250 G01 X0 Y0 U0 V0
N0260 M30
```

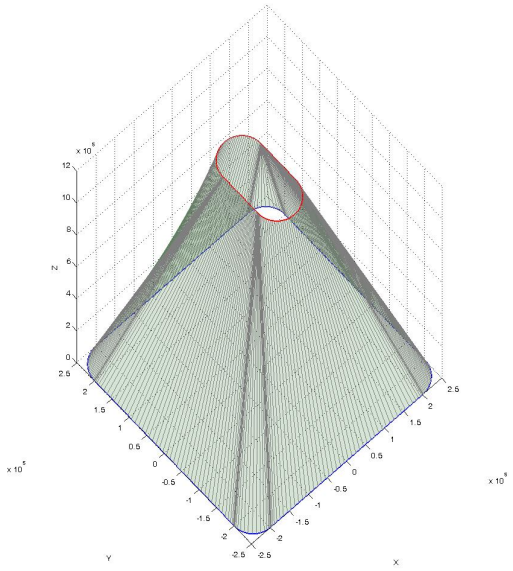


Abb. 212: Programmbeispiel 2Path_Cone

22.5 Definition von unterer und oberer Ebene

Die Festlegung von unterer und oberer Ebene für die beiden Pfade erfolgt basierend auf der Gruppe der Befehle der erweiterten Programmierung von Koordinatensystemen mit: #CS ADD bzw. #CS SET.

Durch die zusätzliche Angabe von translatorischen und rotatorischen Verschiebungen in den entsprechenden NC-Befehlen kann so ein neues CS-Paar, bestehend aus einem unteren CS (Referenz-CS) und einem oberen CS (Sekundär-CS), definiert werden. In Allgemeiner Form kann die Syntax für ein neues CS-Paar wie folgt dargestellt werden:

Schema 2-Pfadspezifischen CS-Programmierung für untere und obere Ebene:

#CS-Befehl [<*Name*>] [<*Untere (Referenz-)Ebene XYZ*>] [<*Obere (Sekundär-)Ebene UVW*>]

Syntax Definition und Verkettung von CS:

#CS ADD [<*Name*>**] [** <$vi_{ref.}$>,<$\varphi i_{ref.}$> **] [** <$vi_{sec.}$>,<$\varphi i_{sec.}$> **]**

Syntax Ändern der Definition eines CS:

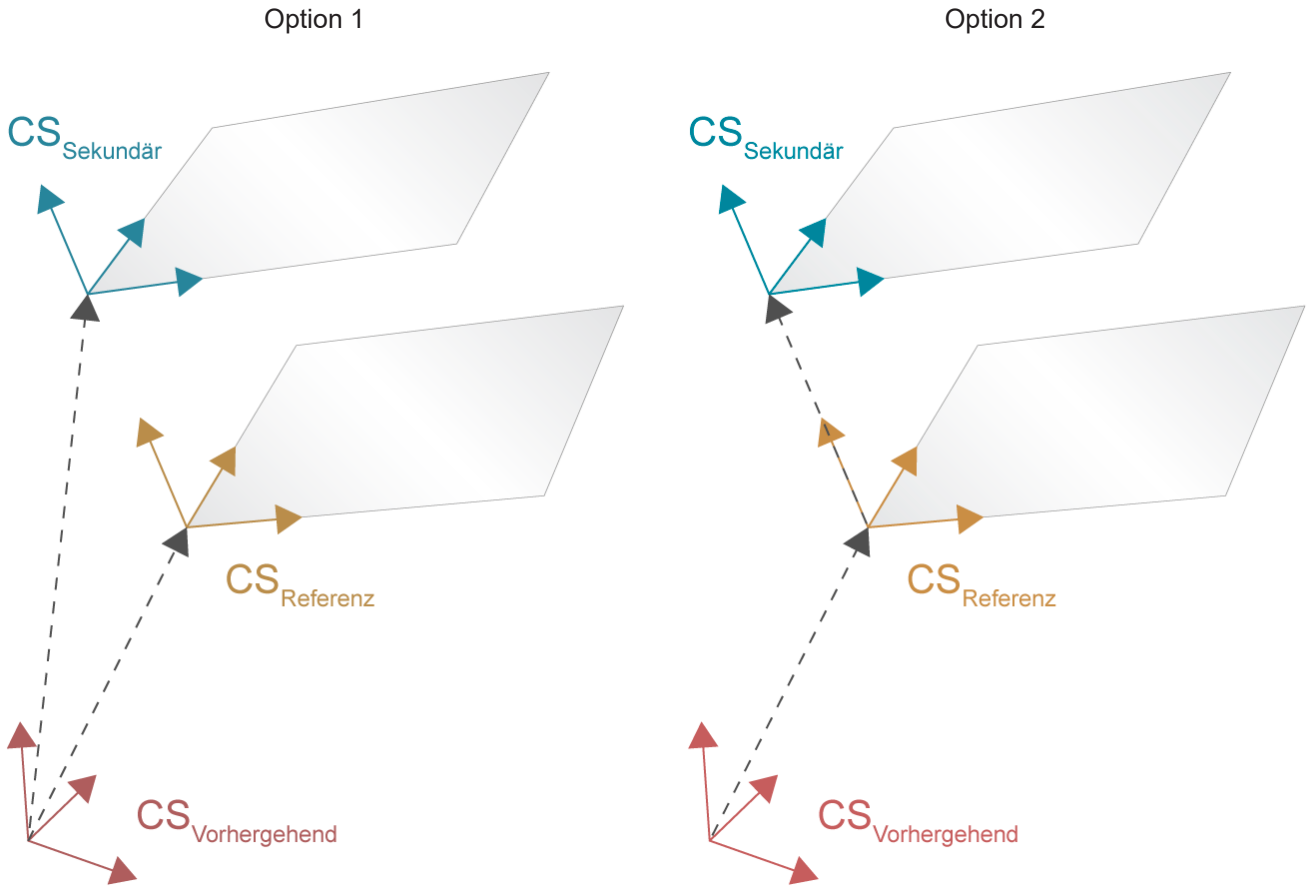
#CS SET [<*Name*>**] [** <$vi_{ref.}$>,<$\varphi i_{ref.}$> **] [** <$vi_{sec.}$>,<$\varphi i_{sec.}$> **]**

Beispiel:

```
#CS ADD [ICS] [10, 10, 0, 0, 0, 0] [10, 10, 25, 0, 0, 0]
#CS SET [WCS] [15, 20, 0, 0, 0, 0] [25, 10, 30, 0, 0, 0]
```

Für die Definition der translatorischen und rotatorischen Verschiebungen stehen 2 Optionen zur Verfügung:

- OPTION 1: Referenz- und Sekundär-CS beziehen sich beide relativ auf den Ursprung eines vorhergehenden CS-Paares
- OPTION 2: Sekundär-CS bezieht sich relativ auf den Ursprung des eigenen Referenz-CS



Optionen zur Definition des Ursprungbezugs von Referenz- und Sekundär-CS

Die Grundeinstellung kann mit dem Kanalparameter P-CHAN-00396 festgelegt werden. Im NC-Programm wird die gültige Option mit dem NC-Befehl #CS MODE gesetzt:

Syntax Relativmodus des Sekundär-CS:

#CS MODE ON [2ND_ON_ACTUAL_1ST_PATH]	Option 2
#CS MODE OFF [2ND_ON_ACTUAL_1ST_PATH]	Option 1, Grundzustand

Programmierbeispiel

Definition von Referenz- und Sekundär-CS

Sind bei der Definition auch Rotationen beteiligt, so wird bei Option 1 die Beschreibung des Sekundär-CS komplexer. Zum Beispiel muss eine Verschiebung des Sekundär-CS von 10 mm in Z-Richtung bei der Programmierung wie folgt umgesetzt werden:

```
#CS ADD [PCS][50,50,0,10,5,0] [50.85831, 48.26351, 9.81060,10,5,0]
..
```

Wird alternativ das Sekundär-CS gemäß Option 2 relativ zum eigenen Referenz-CS definiert, so vereinfacht sich die Programmierung:

```
#CS MODE ON[2ND_ON_ACTUAL_1ST_PATH]
#CS ADD [PCS][50,50,0,10,5,0][0,0,10,0,0,0]
..
```

Hinweis

Bei der 2-Pfadprogrammierung kann das Tracking mit #CS TRACK nur für das Referenz-CS kommandiert werden. Das Sekundär-CS wird so nachgeführt, dass die relativen Verschiebungen und Rotationen zum neuen Referenz-CS konstant bleiben.

Weitere drahtspezifische Optionen, die mit #CS MODE gesetzt werden können:

Die Berechnung der Schnittpunkte des Erodierdrahtes mit den definierten Koordinatensystemebenen (Durchstoßpunkte) erfolgt gemäß der Einstellung des Kanalparameters P-CHAN-00398 (Grundeinstellung).

Mit folgenden #CS MODE Einstellungen kann die Schnittpunktberechnung für die kartesische Vorwärts- und Rückwärtstransformation auch im NC-Programm parametriert werden.

Syntax für Einstellungen zur Schnittpunktberechnung:

#CS MODE ON [INTERSECTION]	Anwahl Schnittpunktberechnung mit X,Y- und U,V-Ebene bei Vorwärts- und Rückwärtstransformation
#CS MODE OFF [INTERSECTION]	Abwahl Schnittpunktberechnung

In der Grundeinstellung bei Programmstart werden die Koordinaten der Durchstoßpunkte, welche von der CNC bereitgestellt werden, im jeweiligen lokalen XY-Referenz- bzw. UV-Sekundärsystem dargestellt. Die lokalen Koordinaten für Z und W sind hierbei 0.

Zur Diagnose kann es notwendig sein, die Durchstoßpunkte global im MCS-Referenzsystem darzustellen. Hier sind dann Z und W globale Koordinaten und ungleich 0.

Zu diesem Zweck kann die Anzeige der Koordinaten der Durchstoßpunkte mit #CS MODE zwischen lokalem CS und globalem MCS umgeschaltet werden:

Syntax zur Anzeige der Schnittpunktkoordinaten:

#CS MODE ON [DISP_GLOBAL]	Anwahl/Anzeige globaler Koordinaten
#CS MODE OFF [DISP_GLOBAL]	Abwahl/Anzeige lokaler Koordinaten, Grundeinstellung

22.6 Verschieben eines Koordinatensystems (#CS SHIFT Z)

Die Verwendung von #CS SHIFT Z ist nur sinnvoll bei einer 2-Pfadkonfiguration (z.B. Drahterodieren). Ein bereits definiertes Koordinatensystem kann hierbei entlang dem Draht auf eine neue Höhe in Z verschoben werden. Der Durchstoßpunkt wandert entlang des Drahtes, aber die Koordinaten innerhalb des Koordinatensystems selbst ändern sich nicht. Die Verschiebung kann nur für das Primär-CS kommandiert werden.

Syntax Verschieben eines CS in Z:

#CS SHIFT Z [<*Name*>**] [** <*Neue Höhe*> **]**

<*Name*>	Name des zu verschiebenden CS mit maximal 8 Zeichen.
<*Neue Höhe*>	Neuer Abstand zum Sekundär-CS des vorhergehenden Koordinatensystems in [mm, inch].

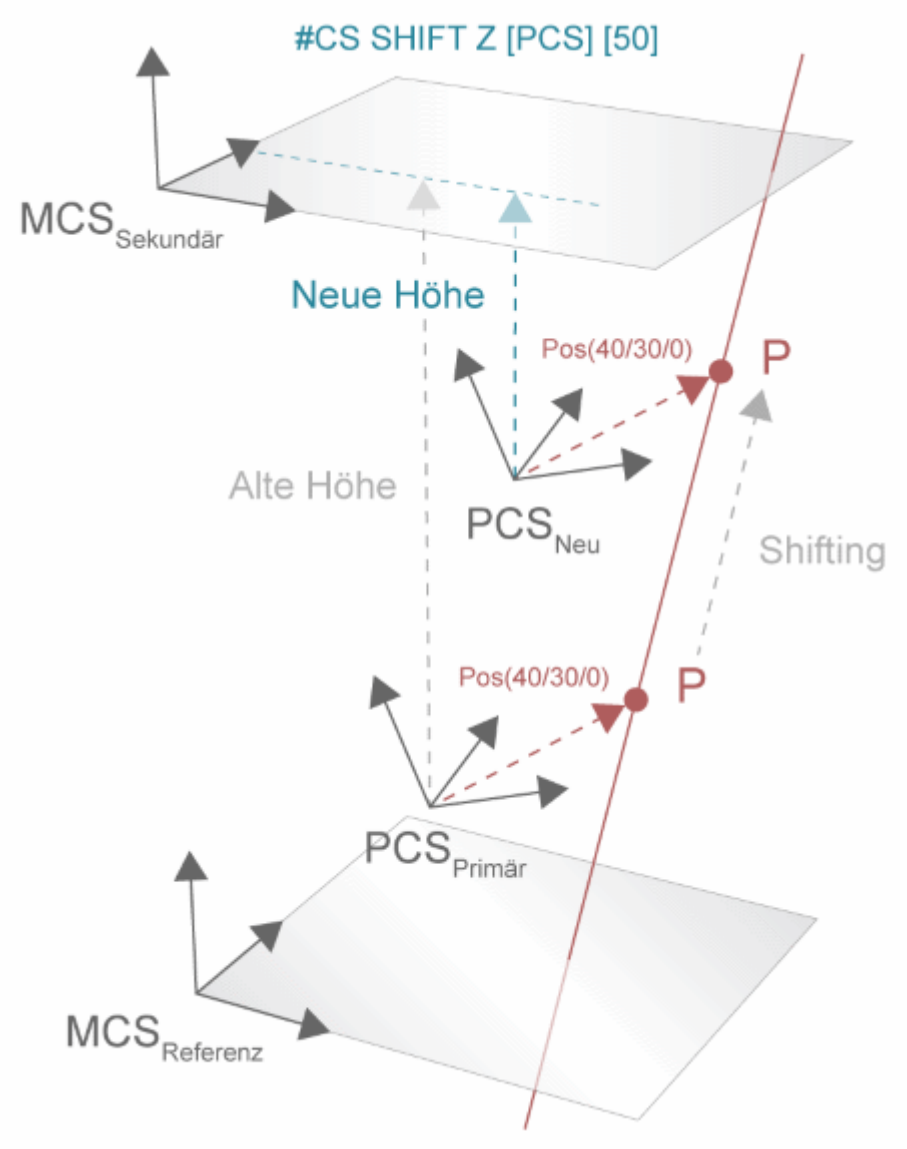


Abb. 213: Verschieben eines PCS mit #SHIFT CS Z

Hinweis

#CS SHIFT Z und #CS SET haben bezüglich der Verschiebungen in einem Koordinatensystemstapel die gleiche Wirkung. Jedes Koordinatensystem oberhalb des verschobenen CS setzt wieder relativ auf den resultierenden Verschiebungen von #CS SHIFT Z auf.

Programmierbeispiel

Verschieben eines CS mit #CS SHIFT Z

```
:
N10 #CS ADD [PCS][...] [...]
N20 #CS SELECT [PCS]
N30 G0 X40 Y30          ;Fahre auf PCS-Position P(40,30)
:
N50 #CS SHIFT Z [PCS] [50] ;PCS verschieben auf neue Abstandshöhe 50
:
M30
```

23 Anhang

23.1 Befehlsübersicht

23.1.1 G- Funktionen (G..)

G00	Geradeninterpolation im Eilgang	S. [54]
G01	Geradeninterpolation	S. [55]
G02	Kreis im Uhrzeigersinn (cw)	S [56].
G03	Kreis im Gegenuhrzeigersinn (ccw)	S. [56]
G02 Z.. K..	Helikalinterpolation (cw)	S. [65]
G03 Z.. K..	Helikalinterpolation (ccw)	S. [65]
G04	Verweilzeit	S. [89]
G05	Tangentiale An/Abwahl der WRK	S. [564]
G08	Beschleunigung am Satzanfang	S. [114]
G09	Verzögerung am Satzende	S. [114]
G10	Vorschub der WRK: konstant	S. [567]
G11	Vorschub der WRK: angepasst	S. [567]
G12	Abwahl Eckenverzögerung	S. [134]
G13	Anwahl Eckenverzögerung	S. [134]
G17	X-Y-Ebene	S. [119]
G18	Z-X-Ebene	S. [119]
G19	Y-Z-Ebene	S. [119]
G20	Abwahl der Spiegelung	S. [120]
G21	Spiegeln der programmierten Wege an der Y-Achse	S. [120]
G22	Spiegeln der programmierten Wege an der X-Achse	S. [120]
G23	Überlagerung von G21 und G22	S. [120]
G25	Geradenübergänge bei der WRK	S. [506]
G26	Kreisübergänge bei der WRK	S. [506]
G33	Gewindeschneiden, gleichbleibende Steigung	S. [659]
G40	WRK/SRK Abwahl	
	• Werkzeugradiuskorrektur (WRK)	S. [506]
	• Schneidenradiuskorrektur (G40/G41/G42)	S. [652]
G41	WRK/SRK links der Kontur	
	• Werkzeugradiuskorrektur (WRK)	S. [506]
	• Schneidenradiuskorrektur (G40/G41/G42)	S. [652]
G42	WRK/SRK rechts der Kontur	
	• Werkzeugradiuskorrektur (WRK)	S. [506]
	• Schneidenradiuskorrektur (G40/G41/G42)	S. [652]
G51	Anwahl Durchmesserprogrammierung	S. [650]
G52	Abwahl Durchmesserprogrammierung	S. [650]
G53	Abwahl der Nullpunktsverschiebung	S. [135]
G54 - G59	Anwahl der Nullpunktsverschiebungen	S. [135]
G60	Genauhalt	S. [130]
G61	Polynomüberschleifen	S. [131]
G63	Gewindebohren	S. [665]
G66	Taktsynchronisierung am Satzende	S. [188]
G68	Anwahl Konturrotation	S. [189]
G69	Abwahl Konturrotation	S. [189]

G310	Messen mit Unterbrechung und Sprung (G310) (Typ 5, 6)	S. [106]
G331	Gewindebohren mit Angabe der Steigung	S. [667]
G332	Gewindebohren Rückzug mit Angabe der Steigung	S. [667]
G333	Beschleunigungsgewichtung bei Feedhold, achsspezifisch	S. [150]
G334	Beschleunigungsgewichtung bei Feedhold, achsgruppenspezifisch	S. [150]
G338	Rampenzeitgewichtung bei Feedhold, achsspezifisch	S. [152]
G339	Rampenzeitgewichtung bei Feedhold, achsgruppenspezifisch	S. [152]
G351	Spiegelung mit Achsangabe	S. [124]
G359	Abwahl Genauhalt	S. [130]
G360	Anwahl Genauhalt	S. [130]
G800 - G819	Implizite Unterprogrammaufrufe (zusätzlich)	S. [127]
G900	Verzögerung am Satzende	S. [114]
G901	Verzögerung nach Satzende	S. [114]

23.1.2 M- Funktionen (M..)

M00	Programmierter Halt	S. [▶ 193]
M01	Wahlweiser Halt	S. [▶ 193]
M02	Programmende	S. [▶ 193]
M03	Spindeldrehung cw	
	• in DIN-Syntax	S. [▶ 642]
	• in spindelspezifischer Syntax	S. [▶ 691]
M04	Spindeldrehung ccw	
	• in DIN-Syntax	S. [▶ 642]
	• in spindelspezifischer Syntax	S. [▶ 691]
M05	Spindel stoppen	
	• in DIN-Syntax	S. [▶ 642]
	• in spindelspezifischer Syntax	S. [▶ 691]
M06	Aufruf eines Werkzeugwechselprogramms	S [▶ 195].
M17	Unterprogrammende	S. [▶ 193]
M19	Spindel positionieren	
	• in DIN-Syntax	S. [▶ 643]
	• in spindelspezifischer Syntax	S. [▶ 693]
M29	Unterprogrammende	S. [▶ 193]
M30	Programmende	S. [▶ 193]
M40 - 45	Anwahl der Spindelgetriebestufen	S. [▶ 647]

23.1.3 Nach DIN reservierte Funktionen und ISG-Erweiterungen

D	Werkzeuggeometriekorrektur (Werkzeugkorrekturdaten)	S. [▶ 499]
E	Vorschubgeschwindigkeit am Satzende	S. [▶ 201]
F	Vorschubgeschwindigkeit im Satz	S. [▶ 201]
M/H	Anwenderspezifische Technofunktionen (M/H)	
	• in DIN-Syntax	S. [▶ 192]
	• in spindelspezifischer Syntax	S. [▶ 696]
L	Globale Unterprogramme	S. [▶ 207]
LL	Lokale Unterprogramme	S. [▶ 206]
L , LL CYCLE	Zyklen als globale oder lokale Unterprogramme	S. [▶ 212]
L SEQUENCE	Aufruf von Satzfolgen	S. [▶ 219]
N	Satznummer	S. [▶ 204]
P	Parameter und Parameterrechnung	S. [▶ 228]
R	Radiusprogrammierung	S. [▶ 56]
S	Spindeldrehzahl	
	• in DIN-Syntax (S-Wort)	S. [▶ 645]
	• in spindelspezifischer Syntax (REV)	S. [▶ 695]
S.POS	Spindel positionieren	
	• in DIN-Syntax	S. [▶ 643]
	• in spindelspezifischer Syntax (POS)	S. [▶ 693]
S.OFFSET	Gewindeversatzwinkel bei mehrgängigen Gewinden	S. [▶ 656]
T	Werkzeugplatzanwahl	S. [▶ 200]
/	Ausblenden von Sätzen	S. [▶ 27]
\	Zeilenumbruch im NC-Satz	S. [▶ 29]
"..."	Makroprogrammierung	S. [▶ 729]

23.1.4 Steuersatzanweisungen ($..)

$BREAK	BREAK- Anweisung	S. [249]
$CONTINUE	CONTINUE- Anweisung	S. [250]
$DO **$ENDDO**	DO- Schleife	S. [247]
$REPEAT **$UNTIL**	REPEAT- Schleife	S. [247]
$FOR **$ENDFOR**	FOR- Schleife	S. [245]
$GOTO	GOTO- Anweisung	S. [240]
$IF **$ELSE** **$ELSEIF** **$ENDIF**	IF- ELSE- Verzweigung	S. [236]
$SWITCH **$CASE** **$DEFAULT** **$ENDSWITCH**	Sprungverteiler	S. [239]
$WHILE **$ENDWHILE**	WHILE- Schleife	S. [247]

23.1.5 Zusatzfunktionen (#..)

A

#ACS ON/OFF	Definition/ Aktivierung eines Aufspannlagenkorrektur-KS	S. [▶ 759]
#ADD	Zusatzinformation am Satzende	S. [▶ 375]
#AKIMA STARTVECTOR	Definition Starttangente	S. [▶ 300]
#AKIMA ENDVECTOR	Definition Zieltangente	S. [▶ 300]
#AKIMA TRANS	Übergangsart Splinekurve - Bewegungssatz	S. [▶ 299]
#ANG	Konturzugprogrammierung	S. [▶ 75]
#AX DEF	Definition einer Achskonfiguration (erweiterte Syntax)	S. [▶ 335]
#AX DEF DEFAULT	Laden der Defaultachskonfiguration (erweiterte Syntax)	S. [▶ 335]
#AX LINK ON/OFF/OFF ALL	Programmierung von Achskopplungen	
	• erweiterte Syntax	S. [▶ 359]
	• Erweiterung "SOFT-GANTRY"	S. [▶ 361]
#AX LOCK/UNLOCK ALL	Sperren einer Achsbewegung bei PTP	S. [▶ 748]
#AX REQUEST	Anfordern von Achsen (erweiterte Syntax)	S. [▶ 325]
#AX RELEASE	Abgeben von Achsen (erweiterte Syntax)	S. [▶ 332]
#AX RELEASE ALL	Abgeben aller Achsen (erweiterte Syntax)	S. [▶ 332]

B

#BACKWARD STORAGE CLEAR	Rückwärtsfahrspeicher löschen	S. [▶ 460]
#BCS DEF	Definition und Speicherung eines BCS	S. [▶ 767]
#BCS ON/OFF	Definition, Speicherung und Aktivierung eines BCS	S. [▶ 767]
#BLOCKSEARCH LOCKED/ RELEASED	Sperren von Programmbereichen für den Satzvorlauf	[FCT-C6]

C

#CACHE LOAD/CLEAR/ALL	NC-Programme in lokalen Speicher laden	[FCT-C23]
#CALL AX	Anfordern von Achsen	S. [▶ 316]
#CAX	Anfordern der Spindelachse für C-Achsbearbeitung	S. [▶ 670]
#CAX OFF	Abgabe der C-Achse an die Spindel	S. [▶ 670]
#CAXTRACK ON/OFF	Automatische Achsnachführung	S. [▶ 416]
#CHANNEL INIT	Kanal mit aktuellen Soll-/Istpositionen initialisieren	
	• Kanal mit Sollpositionen initialisieren	S. [▶ 179]
	• Kanal mit Istpositionen initialisieren	S. [▶ 180]
#CHANNEL INTERFACE ON/OFF	Dynamisches CS über Kanalinterface	[FCT-C30]
#CHANNEL SET	Setzen funktionsspezifischer Parameter im Kanal	
	• Vorschubprogrammierung bei Mikrostegen	[FCT-C1]
	• Zeit-Offsets bei Vorausberechnung	[FCT-C34]
#CHF	Einfügen von Fasen und Radien: Fasenlänge	S. [▶ 156]
#CHR	Einfügen von Fasen und Radien: Fasenbreite	S. [▶ 156]
#CLEAR CONFIG	Löschen einer gesicherten Konfiguration	S [▶ 314].
#COMMAND WR	Nicht synchronisiertes Schreiben von SERCOS-Kommandos	S. [▶ 383]
#COMMAND WR SYN	Synchronisiertes Schreiben von SERCOS-Kommandos	S. [▶ 384]
#COMMAND WAIT	Nicht synchronisiertes Warten auf SERCOS-Kommandos	S. [▶ 385]
#COMMAND WAIT SYN	Synchronisiertes Warten auf SERCOS-Kommandos	S. [▶ 386]
#COMMENT BEGIN/END	Satzübergreifender Kommentar	S. [▶ 345]
#CONTOUR MODE	Überschleifen	
	• Glättungsverfahren	S. [▶ 251]
	• Parametrierung Polynomüberschleifen	S. [▶ 274]
#CONTROL AREA BEGIN/END	Definition eines Kontrollbereichs	S. [▶ 449]
#CONTROL AREA ON/OFF	An-/ Abwahl eines Kontrollbereichs	S. [▶ 452]
#CONTROL AREA CLEAR	Löschen eines Kontrollbereichs	S. [▶ 453]
#CORNER PARAM	Parametrierung der Eckenverzögerung	S. [▶ 133]
#CS DEF	Definition und Speicherung eines CS	S. [▶ 752]
#CS ON/OFF	Definition, Speicherung und Aktivierung eines CS	S. [▶ 752]
#CS MODE ON/OFF	Änderung der Drehreihenfolge der CS-Achsen	S. [▶ 752]
#CS ADD/SELECT/SET/ DEL	Erweiterte Programmierung von Koordinatensystemen	S. [▶ 780]
#CS TRACK	Nachführen eines Koordinatensystems	S. [▶ 786]
#CS SHIFT Z	Verschieben eines Koordinatensystems	S. [▶ 875]
#CYL	Mantelflächenbearbeitung	
	• Anwahl	S. [▶ 677]
	• Anwahl 3/4 achsige Rund-/Profilrohrbearbeitung	[FCT-M5]
#CYL 2ROT	Mantelflächenbearbeitung mit 2 rotatorischen Achsen	
#CYL OFF	Mantelflächenbearbeitung	S. [▶ 682]
	• Abwahl	S. [▶ 677]
	• Abwahl 3/4 achsige Rund-/Profilrohrbearbeitung	[FCT-M5]
#CYL ORI LATERAL	5/6 achsige Rundrohrbearbeitung	[FCT-M5]
#CYL ORI PROFILE	5/6 achsige Profilrohrbearbeitung	[FCT-M5]

D

#DELETE	Löschen eigendefinierter Variablen oder Parameter	S. [▷ 620]
#DEL DIST2GO	Restweg verwerfen	[FCT-C28]
#DISABLE MODAL CYCLE	Abwahl eines modalen Zyklus	S. [▷ 212]
#DIST CTRL ON/OFF	An-/ Abwahl 3D-Abstandsregelung	[FCT-M3]
#DIST TO GO BEGIN/END	Restweganzeige in einem Programmabschnitt	S. [▷ 496]
#DISTANCE PROG START ON/ OFF/ CLEAR	Zurückgelegter Fahrweg ab Programmstart	[FCT-C6]
#DRIVE WR SYN	Schalten von Antriebsfunktionen: Synchrones Schreiben	S [▷ 465].
#DRIVE WAIT SYN	Schalten von Antriebsfunktionen: Synchrones Warten auf Quittierung	S [▷ 465].
#DYNAMIC WEIGHT ON/OFF	An-/ Abwahl der Dynamikgewichtung	S. [▷ 485]

E

#ECS ON/OFF	An-/ Abwahl Effektor-Koordinatensystem	S. [▷ 771]
#EDGE MACHINING ON/OFF	An-/ Abwahl Eckenbearbeitung	S. [▷ 483]
#EDM ON/OFF	Umschalten Auflösung externe Geschwindigkeitsschnittstelle	S. [▷ 498]
#ENABLE AX LINK	An-/ Abwahl von Achskopplungen	S. [▷ 365]
#DISABLE AX LINK		
#ERROR	Benutzerdefinierte Fehlerausgabe	S. [▷ 421]
#EXPL SYN	Explizite Synchronisierung	S. [▷ 711]
#EXPORT VE	Export von V.E.-Variablen in Strukturen für eine SPS Integration	[FCT-C22]
#EXTCOMP ON/OFF	An-/Abwahl externer Kompensation	[FCT-C38]

F

#FACE	Anwahl Stirnflächenbearbeitung	S. [▷ 671]
#FACE 2ROT	Anwahl Stirnflächenbearbeitung mit 2 rotatorischen Achsen	S. [▷ 676]
#FACE OFF	Abwahl Stirnflächenbearbeitung	S. [▷ 672]
#FF	Gewichtung des externen Vorschubes	S. [▷ 486]
#FGROUP	Definition einer Vorschubgruppe	S. [▷ 426]
#FGROUP ROT	Vorschubberechnung mit Rundachse	S. [▷ 426]
#FGROUP WAXIS	Schwächste Achse als Vorschubachse	S. [▷ 426]
#FILE NAME	Definition von Dateinamen	S. [▷ 438]
#FILE RENAME	Umbenennen einer Datei	S. [▷ 440]
#FILE DELETE	Löschen einer Datei	S. [▷ 442]
#FILE EXIST	Prüfen der Existenz einer Datei	S. [▷ 443]
#FILTER ON/OFF	An-/Abwahl FIR-Filter und Parametrierung	S. [▷ 262]
#FLUSH	NC-Kanal leeren mit unterbrochener Bewegung	S. [▷ 341]
#FLUSH CONTINUE	NC-Kanal leeren mit kontinuierlicher Bewegung	S. [▷ 341]
#FLUSH WAIT	Synchronisierung von Decodierung und Interpolation	S. [▷ 341]
#FRC	Einfügen von Fasen und Radien: Vorschub im Fasen- oder Kreissegment	S. [▷ 156]
#FREE TOOL CHANGE ON/OFF	Werkzeugwechsel bei aktivem Synchronbetrieb	S. [▷ 461]

G

H

I

J

K

L

M

#MACHINE DATA	Schreiben von Maschinendaten	S. [▶ 434]
#MAIN SPINDLE	Wechsel der Hauptspindel	S. [▶ 706]
#MANUAL LIMITS	Vorgabe der Offsetgrenzen im Handbetrieb	S. [▶ 173]
#MCS ON/OFF	Temporärer Übergang in das Maschinenkoordinatensystem	S. [▶ 773]
#MCS TO WCS	Abbildung Maschinenkoordinaten in Werkstückkoordinaten	S. [▶ 774]
#MEAS MODE	Umschalten des Messtyps	S. [▶ 351]
#MEAS	Erweiterte Programmierung von Messoptionen	S. [▶ 352]
#MEAS DEFAULT	Erweiterte Messoptionen zurücksetzen	S. [▶ 352]
#MSG	Senden einer Meldung aus dem NC-Programm	S. [▶ 368]
#MSG INFO	Senden einer Meldung mit zusätzlichen Informationen für den Empfänger	S. [▶ 371]
#MSG SAVE	Schreiben einer Meldung in eine Datei	S. [▶ 373]

N

#NIBBLE ON/OFF	An-/ Abwahl Nibbeln	S. [▶ 477]

O

#OPTIONAL EXECUTION ON/OFF	Ausblenden von Programmsequenzen beim Vorwärts-/Rückwärtsfahrbetrieb	S. [▶ 457]
#ORI MODE	Orientierungsprogrammierung	
	• Programmierung und Konfiguration für 5-Achskinematiken	S. [▶ 793]
	• Programmierung und Konfiguration für 6-Achskinematiken	S. [▶ 795]
#OTC ON/OFF	Online Werkzeugkompensation	[FCT-C20]
#OVERRIDE	Programmierbarer Bahnoverride	S. [▶ 464]

P

#PSET	Anwahl der Istwertverschiebung	S. [▶ 357]
#PRESET	Abwahl der Istwertverschiebung	S. [▶ 357]
#PTP ON/OFF	An-/Abwahl einer Positionierbewegung ohne Ausgleichsbewegung	S. [▶ 748]
#PUNCH ON/OFF	An-/ Abwahl Stanzen	S. [▶ 477]
#PUT AX	Abgeben von Achsen	S [▶ 320].
#PUT AX ALL	Abgeben aller Achsen	S [▶ 320].

R

#RND	Einfügen von Fasen und Radien: Radius definieren	S. [▶ 156]
#ROTATION ON / OFF	An-/ Abwahl Konturrotation	S. [▶ 402]
#RT CYCLE	Definition des Echtzeit-Zyklus im NC-Programm	[FCT-C32]

S

T

V

#VAR...#ENDVAR	Deklarationsblock für eigendefinierte Variablen oder Parameter	
	• Variablen	S. [▷ 620]
	• Parameter	S. [▷ 228]
#VECTOR LIMIT ON/OFF	Anpassung der Bahndynamikgrenzwerte	S. [▷ 430]
#VIB GUARD	Anwenden des Vibration Guard	[FCT-C36]
#VOLCOMP ON/OFF	An-/Abwahl der Volumenkompensation	[FCT-C26]

W

#WAIT	Warten auf Signale	S. [▷ 398]
#WAIT FOR	Warten auf Ereignis	S. [▷ 346]
#WAIT INDP	Warten auf asynchrone unabhängige Achse	S. [▷ 823]
#WAIT INDP ALL	Warten auf alle asynchronen unabhängigen Achsen	S. [▷ 818]
#WCS TO MCS	Abbildung Werkstückkoordinaten in Maschinenkoordinaten	S. [▷ 774]

23.1.6 Achsspezifische Zusatzfunktionen (<X>[..])

INDP_SYN	Synchrone (satzweise) unabhängige Achsbewegung	S. [▶ 823]
INDP_ASYN	Asynchrone (satzübergreifende) unabhängige Achsbewegung	S. [▶ 823]
OSC	Pendelachsen	S. [▶ 828]
COMP	Ein-/Ausschalten von Achskompensationen	S. [▶ 835]
DIST_CTRL	Abstandsregelung (Getastete Spindel)	S. [▶ 837]
OVERRIDE	Programmierbarer Achsoverride	S. [▶ 842]
DYNAMIC	Programmierbare Beschleunigungsüberlast	S. [▶ 843]
LIFT **LIFT_START / LIFT_END**	Abheben / Senken einer Achse	S. [▶ 850]
SYNC IN / OUT	Synchronisieren einer Achse auf Bahnverbund	S. [▶ 844]
POLY	Programmierung eines Achspolynoms	S. [▶ 846]

23.1.7 PLC-Open-Funktionen (<X>[MC_..])

MC_Home	Referenzpunktfahrt	S. [▶ 717]
MC_MoveAbsolute	Achsbewegung auf absolute Position	S. [▶ 718]
MC_MoveAdditive	Relative Achsbewegung zur kommandierten Position	S. [▶ 719]
MC_MoveRelative	Relative Achsbewegung zur aktuellen Position	S. [▶ 720]
MC_MoveSuperImposed	Relative Achsbewegung zu einer bereits aktiven Bewegung	S. [▶ 721]
MC_MoveVelocity	Endlose Achsbewegung mit einer Geschwindigkeit	S. [▶ 722]
MC_Stop	Anhalten einer Achsbewegung	S. [▶ 723]
MC_GearIn	Getriebekopplung mit Übersetzung	S. [▶ 724]
MC_GearOut	Lösen einer Getriebekopplung	S. [▶ 726]
MC_Phasing	Phasenverschiebung von Kopplungen	S. [▶ 727]
MC_TouchProbe	Messen einer Achsposition	S [▶ 728]

23.1.8 Variablenprogrammierung (V.)

V.A. …	Achsspezifische Variablen	S. [▶ 594]
V.SPDL. … **V.SPDL_PROG. …**	Spindelspezifische Variablen	S. [▶ 600]
V.G. …	Globale Variablen	S. [▶ 602]
V.P. …	Eigendefinierte Variablen, programmglobal	S. [▶ 623]
V.S. …	Eigendefinierte Variablen, (Haupt-)programmübergreifend	S. [▶ 625]
V.L. …	Eigendefinierte Variablen, programmlokal	S. [▶ 627]
V.E. …	Externe Variablen	S. [▶ 629]
V.CYC. …	Eigendefinierte Variablen, Zyklenvariablen	S. [▶ 633]
V.TOOL. …	Werkzeug-ID Variablen	S. [▶ 816]
V.TLM. …	Werkzeugstandzeitvariablen	S. [▶ 820]

23.1.9 Sonstige Funktionen

Für die Programmierung mathematischer Ausdrücke [▶ 32] (z.B. SIN, COS, MOD, ABS, OR..) sowie die Verarbeitung von Strings [▶ 38] (z.B. LEFT, MID, INSERT..) stehen verschiedene Rechenoperationen und Funktionen zur Verfügung.

23.1.10 Migrierte NC-Befehle

Versionshinweis

In folgender Tabelle sind Befehle aufgeführt, die aufgrund funktionaler Weiterentwicklungen oder aus syntaktischen Gründen in eine neue NC-Syntax überführt wurden.

Die bisherigen Befehle sind weiterhin programmierbar (abwärtskompatibel), sollten aber bei der Erstellung neuer NC-Programme nicht mehr verwendet werden.

Alte Syntax:	Neue Syntax	ab Version
#SET DEC LR SOLL	#CHANNEL INIT [...] [▷ 179]	**V2.10.1504.00**
#VECTORVEL ON / OFF	#VECTOR LIMIT ON / OFF [...] [▷ 430]	**V2.10.1507.02**
#VECTORACC ON / OFF	#VECTOR LIMIT ON / OFF [...] [▷ 430]	**V2.10.1507.02**
#INIT MAKRO TAB	#INIT MACRO TAB [▷ 729]	**V2.11.2010.02**
G200 #ACHSE […]	G200 X.. Y.. [▷ 169]	**V2.11.2010.02**
G201 #ACHSE […]	G201 X.. Y.. [▷ 167]	**V2.11.2010.02**
G202 #ACHSE […]	G202 X.. Y.. [▷ 167]	**V2.11.2010.02**
#GET IPO OFFSET	#GET MANUAL OFFSETS [▷ 178]	**V2.11.2010.02**
#SET OFFSET [...] X	#MANUAL LIMITS […] [▷ 173]	**V2.11.2010.02**
#SET HR [...] X	#HANDWHEEL [...] [▷ 170]	**V2.11.2010.02**
#SET TIP [...] X	#JOG CONT [...] [▷ 171]	**V2.11.2010.02**
#SET JOG [...] X	#JOG INCR [...] [▷ 172]	**V2.11.2010.02**
#SET IPO SOLLPOS […]	#GET CMDPOS […] [▷ 181]	**V2.11.2010.02**
#SET SLOPE PROFIL […]	#SLOPE … […] [▷ 377]	**V2.11.2010.02**
#SET ASPLINE STARTTANG X.. Y..	#AKIMA STARTVECTOR X.. Y.. [▷ 300]	**V2.11.2010.02**
#SET ASPLINE ZIELTANG X.. Y..	#AKIMA ENDVECTOR X.. Y.. [▷ 300]	**V2.11.2010.02**
#SET ASPLINE MODE [...]	#AKIMA TRANS [...] [▷ 299]	**V2.11.2010.02**
#SET CORNER PARAM [...]	#CORNER PARAM [...] [▷ 133]	**V2.11.2010.02**
#SET TANGFEED RMIN [...]	#TANGFEED [...] [▷ 347]	**V2.11.2010.02**
#SET SPLINE ON / OFF	#SPLINE ON [▷ 297] / OFF [▷ 298]	**V2.11.2010.02**
#SET SPLINETYPE AKIMA	#SPLINE TYPE AKIMA [▷ 297]	**V2.11.2010.02**
#SET SPLINETYPE BSPLINE	#SPLINE TYPE BSPLINE [▷ 303]	**V2.11.2010.02**
#RTCP ON / OFF	#TRAFO ON / OFF [▷ 734]	**alle Versionen**

23.2 Revisionsverlauf

Version	Eintrag	Datum
1.0	Erste Version der Programmieranleitung im neuen Layout	01.10.2019
1.01	WRK: Direkte An-/Abwahl (G236) der WRK auf die Bahn	07.11.2019
1.02	Setzen einer Achsposition im Kanal	20.03.2020
1.03	TRC-Option: PERPENDICULAR_RADIUS_CHANGE und STRETCH_FACTOR	24.03.2020
1.13	Kapitel- V.CYC: neu	01.04.2020
1.14	Zahlreiche kleinere inhaltliche und redaktionelle Fehler behoben	02.04.2020
1.15	Integration von #FF	22.05.2020
	Einschränkungen bei NC-Dateinamen	27.05.2020
1.16	#CONTOUR MODE -> PATH_DIST neu	08.07.2020
	V.G.-Variablen : INVOKE_COUNT und LIST_COUNT neu	08.07.2020
1.17	Achsspezifische Programmierung: „Fahren auf Festanschlag“ neu	26.08.2020
1.18	Ausschlussliste von Befehlen bei aktiver WRK/SRK in WRK-Kapitel	26.11.2020
1.20	#GANTRY ON / OFF integriert	01.02.2021
1.211	#ORI MODE in Befehlsübersicht	03.02.2021
1.212	V.G.CAXTRACK_ACTIVE neu.	05.02.2021
1.213	LENGTH_LONG_CIR in #HSC [SURFACE..] neu	15.03.2021
1.214	Austausch von Filterprogrammierung durch „FIR-Filter programmieren“ in Glättungsverfahren.	12.04.2021
1.215	Anpassung/Ergänzung zahlreicher Überschriften an den entsprechenden Programmierbefehl nach "Übersicht Programmierbefehle".	23.04.2021
1.216	Integration von #KIN DATA[]	16.09.2021
1.217	Zahlreiche kleinere inhaltliche und redaktionelle Fehler behoben, Kapitel Makroprogrammierung ergänzt um dynamische Konfiguration, Anhangkapitel Zusatzfunktionen um Leiste für Buchstabenschnellzugriff erweitert. Buchstabenbereiche fragmentiert.	23.05.2022, Gr
1.218	#DISABLE MODAL CYCLE mit Link im Anhang ergänzt.	29.06.2022
1.219	#DIST TO GO BEGIN/END integriert	02.08.2022
1.220	Verlinkung für #DIST CTRL und #LIMIT LOAD im Anhang neu.	25.03.2024
1.221	#EDM ON/OFF neu	30.04.2024
1.222	Syntaxverweis für #KIN TCP DEF/DELETE im Anhang ergänzt.	04.09.2024

24 Literatur

[1] Dokumentation/Allgemeine Beschreibung der Kanalparameter [CHAN]

Nr.	Beschreibung
1	Elemente der Struktur makro_def[i].*
2	Elemente der Struktur synchro_data.koppel_gruppe[0].*
3	Elemente der Struktur spindel[i].*
4	Elemente der Struktur spindel[i].range_table[i].*
5	Elemente der Struktur gruppe[j].achse[i].*
6	Elemente der Struktur speed_limit_look_ahead.*
7	Elemente der Struktur dynamic_weighting[i].*

[2] Dokumentation/Allgemeine Beschreibung der Achsparameter [AXIS]

Nr.	Beschreibung
1	Elemente der Struktur getriebe[i].slope_profil.*
2	Elemente der Struktur getriebe[i].lslope_profil.*
3	Elemente der Struktur filter[i].*

[3] Dokumentation/Allgemeine Beschreibung der Nullpunktverschiebungen [ZERO]

[4] Spezifikation SERCOS Interface, IEC 61491

[5] Dokumentation/Allgemeine Beschreibung der Werkzeugdaten [TOOL]

[6] Dokumentation steuerungs-/herstellerspezifische Einstellungen von Systemparametern [SYSP]

[7] Dokumentation /Allgemeine Beschreibung der Hochlaufliste [STUP]

[8] Dokumentation/ Allgemeine Beschreibung externer Variablen [EXTV]

[9] Motion Control Plattform für PLCopen [MCP-P1]

Stichwortverzeichnis

ISG Industrielle Steuerungstechnik GmbH
STEP, Gropiusplatz 10
D-70563 Stuttgart

www.isg-stuttgart.de
support@isg-stuttgart.de