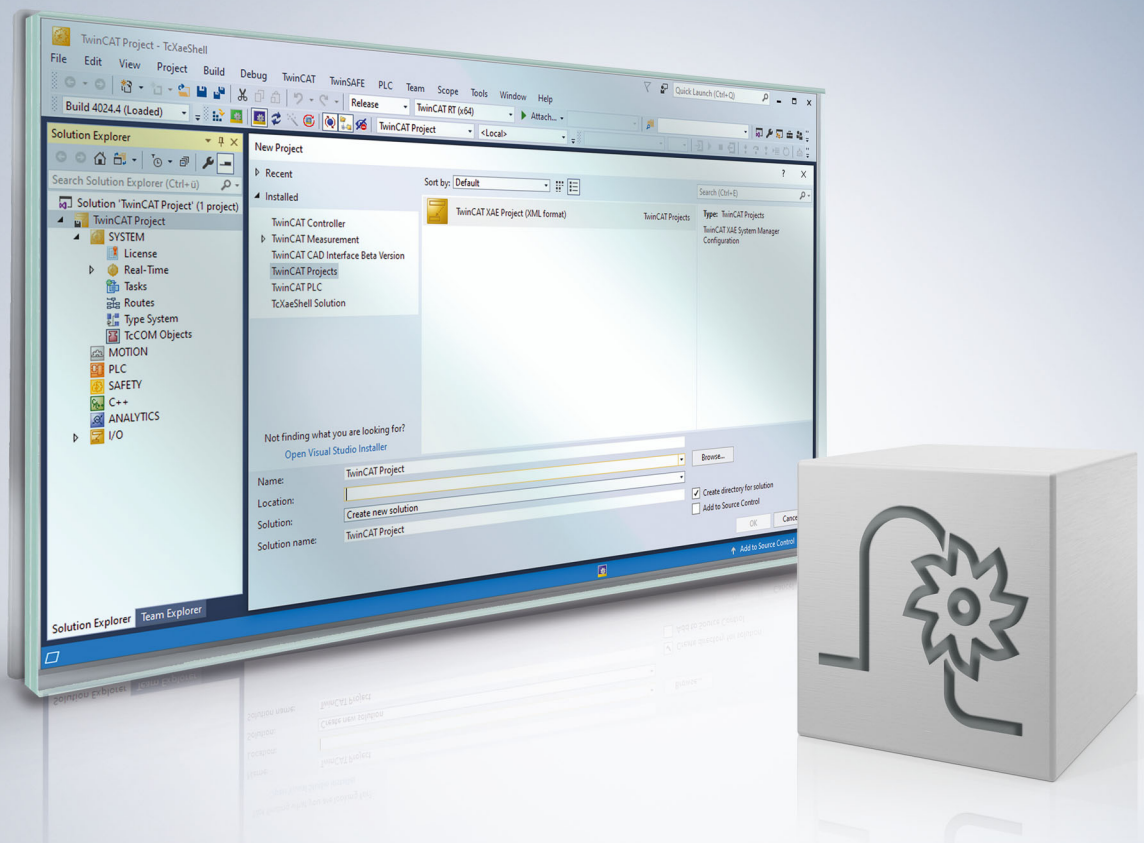
BECKHOFF New Automation Technology

マニュアル | JA

TF5200

TwinCAT 3 CNC - プログラミング手順



2024-02-27 | バージョン: 1.01

目次

1 取扱説明書に関する注記

この説明書は対応する国内規格を熟知した、トレーニングを受けた制御、オートメーションエンジニアリングの有資格者のみの使用を対象としています。
本製品の設置およびコミッショニングの際は、必ず以下の注意事項と説明に従ってください。
有資格者は、常に最新版のドキュメントを参照する管理義務があります。

本製品を使用する上での責任者は、本製品の用途および使用方法が、関連するすべての法律、法規、ガイドラインおよび規格を含む、安全に関するすべての要件を満たしていることを確認してください。

免責事項

この取扱説明書の記載内容は、一般的な製品説明および性能を記載したものであり、場合により記載通りに動作しないことがあります。
製品の情報· 仕様は予告なく変更されます。
この説明書に記載されているデータ、図および説明に基づいて、既に納品されている製品の変更を要求することはできません。掲載されている写真やイラストと、実際の製品は異なる場合があります。この説明書は最新でない可能性があります。必ず最新バージョンの説明書を参照してください。

商標

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS®, XPlanar® は、Beckhoff Automation GmbH の登録商標です。
この取扱説明書で使用されているその他の名称は商標である可能性があり、第三者が独自の目的のために使用すると所有者の権利を侵害する可能性があります。

特許

EtherCAT Technologyについては、欧州特許
EP1590927、EP1789857、EP1456722およびEP2137893、ドイツ特許DE102015105702
に記載されていますが、これらに限定されるものではありません。

EtherCAT®は、Beckhoff Automation GmbHの登録商標および特許技術です。

著作権

© Beckhoff Automation GmbH & Co.KG, Germany.
明示的な許可なく、本書の複製、配布、使用、および他への内容の転載は禁止されています。
これに違反した者は損害賠償の責任を負います。ベッコフは、特許、実用新案、意匠の付与に関するすべての権利を留保しています。

2 簡単な説明

DIN 66025規格に沿った言語、あるいはカスタム設計・拡張された言語に従って制御処理を行っています。

- テキスト指向のプログラム
- 非常に広い範囲に及ぶパラメータの計算処理(ローカルおよびグローバルパラメータ)
- プレーンなテキスト名称の変数(V.A.変数、V.G.変数)による、位置データ、計測データ、工具データ、オフセットデータといったコントロール内部データへのアクセス
- プレーンなテキスト名称の変数(V.L.変数、V.S.変数、V.P.変数)で、NCプログラム内で自由にパラメータを指定
- C言語に基づいた制御ブロック。例えば、
  - 条件分岐: $IF, $ELSEIF, $ELSE, $ENDIF,
    $SWITCH, $CASE, $DEFAULT, $ENDSWITCH, $BREAK
  - カウンタ付きループ: $FOR, $ENDFOR, $CONTINUE, $BREAK
  - 実行条件付きループ: $WHILE, $ENDWHILE, $CONTINUE, $BREAK
  - 実行条件なしループ: $DO, $ENDDO, $CONTINUE, $BREAK
  - 同一のNCプログラムレベル内でのジャンプ: $GOTO
- グローバルサブプログラム(すべてのメインプログラムからアクセス可能)とローカルサブプログラム(特定のメインプログラムからのみアクセス可能)の使い分け
- 数式:
  - 標準的な演算子: +、-、*、/、**、MOD
  - 関数: ABS、SQR、SQRT、EXP、LN、DEXP、LOG
  - 三角関数: SIN、COS、TAN、ASIN、ACOS、ATAN
  - 変換関数: INT、FRACT、ROUND
- 技術項目(各機能を有効化する設定が可能):
  - 追加ファンクション(MO ~ M65535)
  - 補助ファンクション(H0 ~ H65535)
  - スピンドルファンクション(S、M3、M4, M5, M19)
  - 工具ファンクション(T、D)
- 座標表記A・B・C ~ U・V・W (他の用途で使用されていない表記に限る)、文字列のプログラミング(X_ACHSE、SPINDEL_1...)のいずれかまたは組み合わせて利用可能。

より詳細の概要は「NCコマンドの付録」記載されています [▶ 562]**!**

3 プログラミングの基本原理

3.1 構文の表記

この取扱説明書に記載されているNCコマンドの構文には、以下の表記を使用します。

太字	NCコマンドの必須構文要素。
[ ]	オプションの再指定しないイベント
{ }	0回または複数回のイベント
|	複数記号に対する論理和(「or」)
< >	数式 (数式 [▶ 24]の章に準拠)または文字列

3.2 安全に関する指示事項

安全に関する注意事項

この取扱説明書に記載された安全に関する指示や注意事項はよくお読みになり、必ず指示に従ってください。

納入仕様

すべての製品は、用途に適した特定のハードウェア構成およびソフトウェア構成を有する状態で供給されます。ハードウェアまたはソフトウェアに取扱説明書に記載されている以外の変更を加えることは許可されていません。許可されていない変更を加えると、Beckhoff Automation GmbH & Co. KGの保証の対象外となります。

使用者の資格

この説明書は関連する国内法規を熟知した、制御およびオートメーションエンジニアリングの専門家の使用を目的としています。

安全記号の説明

この取扱説明書では、安全に関する指示や注意事項とともに以下の安全記号を使用します。安全に関する指示事項はよくお読みになり、必ず指示に従ってください。

⚠ 危険

重大な人的傷害の危険

この記号が付いた安全に関する注意事項に従わないと、人命および健康に直ちに危害を及ぼします。

⚠ 警告

人的傷害の危険

この記号が付いた安全に関する注意事項に従わないと、人命および健康に危険を及ぼします。

⚠ 注意

人的傷害の恐れ

この記号が付いた安全に関する注意事項に従わないと、人命および健康に危険を及ぼす恐れがあります。

注記

物的損害と環境汚染

この記号が付いた安全に関する注意事項に従わないと、物的損害と環境汚染をもたらす恐れがあります。

ヒントまたはアドバイス

この記号が示す情報により、さらに理解が深まります。

3.3 文字書式および数値書式

3.3.1 文字セットおよびファイル形式

ASCII形式で記述されたNC制御データを、解読処理します。コントローラに付属のプログラミングソフトや工ディタで、NC制御データを用意できます。

CNCは、以下の文字を処理します。

文字 = 半角英字 および/または
数字 および/または
特殊文字 および/または
制御文字

半角英字	{ A B C ... Z a b c ... z }
数字	{ 1 2 3 ... 9 0 }
特殊文字	{ ! @ # $ % & * - + _ = ~ ( ) [ ] , ; : " < > / \ ? ' }
制御文字 = HT 水平タブ	TAB
中間スペース	SP
キャリッジリターン	CR
ラインフィード	LF

3.3.2 数値入力

数値の入力は、内部的な数値表現設定によって制限されます。

この数値表現設定ですが、解像度が0.1 µmの場合に200 mの範囲内の動作させることができます。

仕様ファイル内の内部変換係数を変更することで、位置や送りなどの入力をmmやインチ以外の単位でも行うことが可能になります。

- 整数または小数点の数値入力。小数点の入力は、基本的に「.」で区切ります。先頭の0は省略できます。長さおよび位置の値の入力は、設定やプログラミングによりmmまたはインチで行うことができます。角度の入力は、度またはグラード(grade, gon)で行えます。

例:

値: [µm]	値: [mm]
0.1	0.0001または.0001
1	0.001または.001
10	0.01または.01
100	0.1または.1
1000	1.0または.1
10000	10.0または10
100000	100.0または100
1000000	1000.0または1000

- **16進数**の数値入力。「16#」、「0x」、「H」のいずれかを先頭として始めます。値の先頭の0や空白は省略できます。数値以外に6つの値として、A～FおよびA～fの文字を使用できます。

書式: **「16#**<A...F, a..f, 0..9>**」**または**「0x**<A...F, a..f, 0..9>**」**または**「H**<A...F, a..f, 0..9>**」**

例:

「16#FA1B」 - 10進数値64027と同一です。

「0x0ED2」 - 10進数値3794と同一です。

「H1869f」 - 10進数値99999と同一です。

- **2進数**の数値入力。「2#」、「02#」、「B」のいずれかを先頭として始めます。値の先頭の0や空白は省略できます。「0」か「1」の2つのいずれかを値には利用します。

書式: **「2#**<0..1>**」**または**「02#**<0..1>**」**または**「B**<0..1>**」**

例:

「2#1010011」 10進数値83と同一です。

「02#010011」 10進数値19と同一です。

「B11101010」 10進数値234と同一です。

- **8進数**の数値入力。「8#」、「08#」のいずれかを先頭として始めます。値の先頭の0や空白は省略できます。値には、0から7の数値を使用します。

書式: **「8#**<0..7>**」**または**「08#**<0..7>**」**

例:

「8#12345」　10進数値5349と同一です。

「08#0107302」　10進数値36546と同一です。

3.4 NC制御データの構造

説明:

NCプログラムに工具データや座標オフセットデータなども加えたものが、NC制御データの構成要素です。NCプログラムは、処理を順番に実行するステップフローとして記述されます。

区切り文字により、一連の文字列が終了したと判別されます。解読処理により、ブロック終了文字、スペース、タブ、ASCII表記で「/0」が付加された文字、ファイル終了文字、およびコメントを区切り文字として認識します。

コメントには解読できないASCII情報も含めることができ、文字「(」および「)」の間、または文字「;」の後ろに記述をしてください。もしコメント内に「(」が含まれる場合には、対となる「)」が必要です。ブロック終了文字およびファイル終了文字によっても、コメントが終了します。

数式は数値、パラメータ、演算子、関数などで構成されます(2.4.1項「数式」を参照)。これらは内蔵の演算機能によって判別されます。

例:　「100」、「100+20」、「100+P10」、「100+PP10」、「100*[2+P3]」、「DEXP[2]」、

「100+SIN[P10+PP20]」

NCプログラムは、以下で構成されます。

- コメント
- 以下で構成されるローカルサブプログラム
  - 文字列「%L」とそれに続く最大14文字のサブプログラム名。サブプログラム名はブロック終了文字、空文字、タブ、ASCII表記/0が付加された文字、またはファイル終了文字で終了します。
  - 一連のNCプログラムブロック
  - サブプログラムの終了を示すコード(M17またはM29)
- 以下で構成されるメインプログラム
  - 文字列「%」とそれに続くプログラム名。プログラムはブロック終了文字、空文字、タブ、ASCII表記/0が付加された文字、またはファイル終了文字で終了します。
  - 一連のNCプログラムブロック
  - メインプログラムの終了を示すコード(M02またはM30)

コメント以外で、最初の文字が区切り文字でも「%」でもない通常の文字の場合には、プログラム名のないメインプログラムと判別されます。そのため、「%」の前にはブロック番号も記述しないようにしてください。

3.5 NCブロック構造

NCブロックは、以下で構成されます。

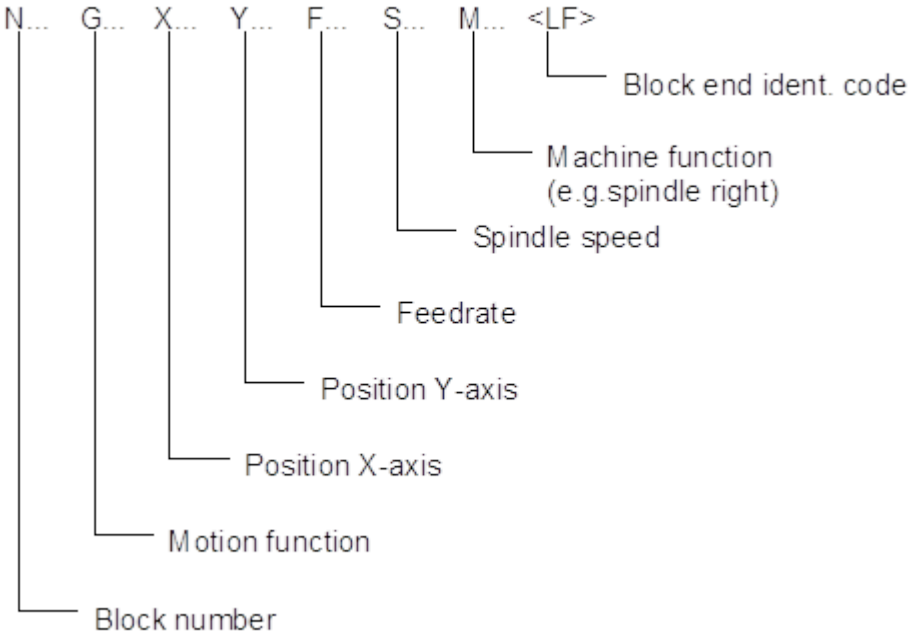
- ブロック番号(オプション)
- 一連のワード
- ブロック終了識別子

11項で説明する特殊コマンド(#<Statement>)を利用した場合、プログラムミングにNCプログラムのワードを使うことができなくなります(NCブロック番号を除く)。

通常、各ブロックは「N」をつけたブロック番号で始まり、その後に数式が続きます。解読処理の個別の設定に対応できるように、この表記は整数表記に変更して解読されます。

プログラムフローでは、ブロック番号は重要ではありません。このため、ブロック番号を昇順でプログラムする必要はありません。

ブロック構造の例:



DIN 66025準拠のNCコマンドは、空文字またはタブによって必ずしも区切る必要はありません。DIN規格に準拠していないテキストコマンドによるプログラミング(制御ステートメント、特殊関数など)では、NCプログラムの記述に区切り文字が必要であり便利です。

NCプログラム構造の例:

番号	部分的に	全てに
なしの場合	番号を付けた場合	番号を付けた場合
% 100	% 100	% 100
"Block 1"	N10 "Block 1"	N10 "Block 1"
"Block 2"	"Block 2"	N20 "Block 2"
"Block 3"	N20 "Block 3"	N30 "Block 3"
.	"Block 4"	N40 "Block 4"
.	.	.
.	.	.
M30	M30	N700 M30

重要度に応じて、ワードは以下で区別されます。

- 形状に関する情報(位置など)
- 技術的な情報(主軸速度、送り速度、主軸回転の右回り指令など)
- プログラム順序の制御に関する情報(カウントループなどのいわゆる制御ブロック)
- 計算に関する情報(変数計算やパラメータ計算など)

複数のワードを1つのブロックに含めることができます(第11章の特殊関数に記載されている特殊コマンドは除く)。ブロック内における一連の制御処理は、そのブロック内を通して適用されます。そのため、プログラマは任意の順序でNCブロックの各々のワードを入力することができます。この入力順序は処理内容に影響を及ぼしません。例外については、プログラミングマニュアルに特記事項として記載されています。

ブロック終了コードは必ず入力します。任意の制御文字をブロック終了コードとすることができますが、慣例的に制御文字「CR」が使用されます。

3.5.1 NCブロックのスキップ「/」

NCプログラム内でも計測ループ、テストブロック、ダミーのようなオプションプログラムにはスキップブロックを適用させることができます。

3.5.1.1 標準スキップ

文字の前に「/」を付けること(「/N3412...」など)で、特定のNCブロックをスキップできます。メインプログラムを開始する前に操作画面(HMI)やPLCのコマンドによって「スキップブロック」機能を有効にしておくと、NCブロックを無視して動作します。NCプログラムを実行中にスキップブロックの設定変更をした場合には、次のメインプログラムを開始した際に設定変更が有効になります。

/ N..

3.5.1.2 拡張スキップ(スキップレベル)

この機能はV300.3021.00 (TwinCAT 3)以降で使用可能です。

スラッシュ(「/」)と数字を組み合わせることで、NCプログラム内で最大10個の異なるスキップレベルを使用できます(/5 N3412など...)。スキップレベルはメインプログラムが開始する前に、操作画面(HMI)またはPLCのコマンド(32ビット変数)によって有効になります。複数のスキップレベルを同時に使うことができます。NCプログラムを実行中にスキップブロックの設定変更をした場合には、次のメインプログラムを開始した際に設定変更が有効になります。

詳細情報は、[FCT-M6]を参照してください。

番号のついていない「/」、および「/1」は、スキップレベルは同じです。

/<1 - 10> N..

プログラミング例

```
%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
```

3.5.2 コメントブロックの記述

NCプログラムのほぼすべての個所に、説明のためのコメントを挿入できます。プログラムの先頭にも挿入できます。コメントはNC機械加工の動作に影響を及ぼしません。

コメントは文字「(」で始まります。コメントがブロック終端まで続く場合は、文字「(」のみで十分です。NCブロックの中間にあるコメントの場合は、対となる文字「)」が必要です。

コメントはセミコロン「;」で始めることもできます。この場合、コメントは必ずブロック終端まで続きます。

ブロックに対する最大文字数を超えなければ、コメントの長さは任意です。

コメントを入れ子構造にすることも可能です。

プログラミング例

```
% 100    (Comment in complete brackets)


N200 ... (Comment only with open bracket
N300     (Comment (Nested comment))

N500 X10 (Comment within a block) Y20

N700 ... ;Comment behind semicolon

N999 M30
```

11.5項に記載されている特殊コマンド*#COMMENT BEGIN / END*を使用すると、複数行に渡るコメントを記述できます。

3.5.3 NCブロックの改行「\」

NCコマンドによっては、NCブロックの構文シーケンスに改行「\」を挿入して複数のセグメントに分割し、読みやすくすることができます。改行文字「\」は、構文用語が完了した後にのみ配置できます。

分割されたブロックでは、対応する構文シーケンスのみ継続して利用可能です。

例外:

分離されたブロックの先頭に新しいブロック番号N... をプログラムすることができます。

通常、改行「\」は以下の場合に使用できます。

- 軸のプログラミング X[...]
- 主軸のプログラミング S[...]
- PLCopenコマンド S[MC_...]
- サイクルプログラミング L CYCLE [...]
- DIN/ISOプログラミング G01 X10 ...
- ユーザによる配列変数の定義

注記

追加ファンクショングループ(# commands)のNCコマンドは、改行で**分割できません**。例外がある場合は個別に記載されています。

プログラミング例

```
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
```

3.6 ワード構造

ワードはアドレス文字、数式、およびテキスト<string>で構成されます。個々のアドレス文字の意味は、以下の章に記載されています。概要は2.4.2項 「アドレス文字の割り当て」に記載されています。

3.6.1 数式

以下で構成されます。

- 数値は以下に分類されています。
  - 整数<int>
  - 小数値<float>
- 演算式<expr>は、以下で構成されます。
  - 数値
  - 演算子
  - 関数
  - パラメータ

- 変数
- 記号文字群

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

3.6.1.1 整数<int>

整数は小数点なしで指定されます。内部的に許容される値の範囲は、プログラミングのC言語のlong int変数と同じです。これは、-2.14*10^9から+2.14*10^9です。mm単位での値の入力に対して内部的に0.1 µmの単位を使用して計算する場合、数値範囲は–2.14*10^5 ~ +2.14*10^5となります。これは、400メートル以上の移動範囲に対応します。位置コントローラの内部制限により、移動範囲はその半分に制限されることがあります。つまり、200メートル以上になります。負の数の場合、先頭に文字「-」が付きます。正の数の場合、文字「+」は不要です。

3.6.1.2 小数値<float>

小数値は小数点、または指数表現で指定します。値の範囲は整数値の範囲に対応します。内部的に0.1 µmの単位を使用して計算する場合、mm単位の入力値に対して小数点第4位まで利用できます。負の数の場合、先頭に文字「-」が付きます。正の数の場合、文字「+」は不要です。

小数値の例:

123.3456 0.6789 .6789 -345.56 +78.987

12E5 +2.5E6 -4E7 -523.6E-3

3.6.1.3 演算式<expr>

数式の処理では、一般的な演算規則が使用されます。

- 加減の前に乗除を行う
- 括弧規則。角括弧[ ]を使用する

数式ではパラメータが頻繁に使用されます。パラメータの表記は、以下の通りです。

- Pの後に整数(P12など)

数式の例:

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

記号文字セットは数式、およびその一部に割り当てられます。

文字列変数の内容が、元の文字列に続いたと判別されます。繰り返し処理も可能です。

文字列は引用符を用いて記述する必要があります。文字列の解読の際には、大文字と小文字が区別されます。文字列のネストには、引用符の前にプリセット文字「\」を記述します。ネストのためには、必ず1つの文字列内でグループ化されていることが必要です。つまり、最初の置換テキストに「[」、後ろの置換テキストに「]」を付加しても、数式は有効なものにはなりません。

プログラミング例

```
Correct:

N10  "STRING1" = "COS[\"STRING2\"]"
N20  "STRING2" = "5 * 12"
N30  "STRING3" = "SIN[89.5 + \"STRING1\"]"

N40  X[-2 * "STRING1" + "STRING2" + "STRING3"]     (Traverse after X60)

M30

Wrong:
Only complete nesting levels should be put together in the string

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

NCプログラム内で定義された文字セットは、プログラム内でどこからでも参照できます。

数式外の記号文字セットのプログラミングは、別のプログラミングマニュアル「記号文字セット」に記載されています。

使用可能な全演算子の概要

基本の計算タイプ:

加算	+	P1 = P2 + P3 + 0.357
減算	-	P1 = P2 - 0.031
乗算	*	P1 = P2 * [P3 + 0.5]
除算	/	P1 = P2 * P3 / [P5 + P6]
指数計算	**	P1 = 2**P3 (2のP3乗)
剰余計算	MOD	P1 = 11 MOD 3 (結果は 2)

数値関数:

絶対値変換	ABS [..]	P1 = ABS [P2 - P4]
2乗	SQR [..]	P1 = SQR [P2] + SQR [P3]
2乗根	SQRT [..]	P1 = SQRT [SQR[P2]+SQR[P3]]
指数関数	EXP [..]	P1 = EXP [P2 * P4]
自然対数	LN [..]	P1 = LN [P2] + LN [P3]
倍精度指数関数	DEXP [..]	P1 = DEXP [P2]
常用対数	LOG [..]	P1 = LOG [P2]

注記

LN、LOG、およびSQRTの場合、引数は必ず0よりも大きな値であることが必要です!

ビット演算子:

論理積	&	P1 = P2 & P3
論理和	|	P1 = P2 | P3
排他的論理和	^	P1 = P2 ^ P3
補集合	INV[..]	P1 = INV[P2]

注記

あらゆる正の数式や数値を引数にすることができます。負の数式や数値は使用できません。浮動小数点の引数は整数に変換されます。

論理演算子:

論理積演算子	&& / AND	`P1 = P2 && P3` `P1 = P2 AND P3`
論理和演算子	|| / OR	`P1 = P2 || P3` `P1 = P2 OR P3`
排他的論理和演算子	XOR	`P1 = P2 XOR P3`
否定演算子	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)`

注記

あらゆる正の数式や数値を引数にすることができます。負の数式や数値は使用できません。

値が0.5以上ならば、浮動小数点数がTRUE (1)に評価されます。

比較演算子

ループ構造内では(10項を参照)、比較演算子が必要となります。以下のように使用することができます。

等式	==	$IF P1 == 5
不等式	!=	$IF P1 != 5
次の値以上	>=	$IF P1 >= 10
次の値以下	<=	$IF P1 <= 10
次の値よりも小さい	<	$IF P1 < 10
次の値よりも大きい	>	$IF P1 > 10

演算子の優先度

使用可能な演算子の優先度を降順で記載します。優先度は10が最高、1が最低です。

優先度	演算子	説明
10	**	指数計算
9	*, /, %	乗算、除算、剰余計算
8	+、-	加算、減算
7	&	ビット論理積
6	^	ビット排他的論理和
5	|	ビット論理和
4	<=、>=、==、<、>、!=	比較演算子
3	&&、AND	論理積演算子
2	XOR	排他的論理和演算子
1	||、OR	論理和演算子

以下の真偽値が使用可能です。

真	TRUE	$IF V.A.MERF.X == TRUE
偽	FALSE	$WHILE V.G.WZ[2].OK == FALSE

注記

真偽値の使用:
TRUE制御のために値1が内部的に使用されます。
FALSE制御のために値0が内部的に使用されます。

三角関数(度単位での角度の指定):

正弦	SIN [..]	P1 = SIN [P2 * 30 +10]
余弦	COS [..]	P1 = COS [P2 * 30 +10]
正接	TAN [..]	P1 = TAN [P2 * 30 +10]
逆正弦	ASIN [..]	P1 = ASIN [P2 * 10]
逆余弦	ACOS [..]	P1 = ACOS [P2 * 10]
逆正接	ATAN [..]	P1 = ATAN [P2 * 10]
引数が2つある 逆正接	ATAN2 [*y,x*]	`P1 = ATAN2[100,100]` `(-> 結果は45°)`

注記

三角関数ASINおよびACOSでは、引数は必ず-1と+1の間の値であることが必要です。
関数TANの場合、引数には値 -90、90、270°を設定しないでください。
関数ATAN2は、xが0以外ならば正しい象限でX軸からの角度を計算します。
例外: ATAN2[0,0] (x = y = 0)ならば、結果は常に0になります。

変換関数

小数点以下の桁を切り捨て	INT [..]	`P1 = INT [123.567] (P1 = 123)`
整数部を削除	FRACT [..]	`P1 = FRACT [123.567] (P1 = 0.567)`
整数に四捨五入	ROUND [..]	`P1 = ROUND [123.567] (P1 = 124)`

定数:

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

特殊関数:

以下の 存在を確認 ・ 変数 (V.P. 、V.L. 、V.S. 、V.E.) ・ パラメータ ・ M/Hファンクション	EXIST [*<variable/* *parameter/* *M function/* H function>]	`$IF EXIST[V.P.MYVAR] == TRUE` `$IF EXIST[P1] != TRUE` `$IF EXIST[M55] == TRUE` `$IF EXIST[H20] == TRUE`
以下の サイズを 判別 ・ 配列変数の (V.P.、V.L.、V.S.、V.E.) の次元 ・ パラメータ	SIZEOF [<array_name>, <dimension>] または、一次元の配列なら SIZEOF [<array_name>]	`$IF SIZEOF` `[V.P.MYARRAY,2] == 3` `P1 = SIZEOF[P10,2]`
変数の より小さな値	MIN [*x,y*]	`P1 = MIN [P2, P3]`
変数の より大きな値	MAX [*x,y*]	`P1 = MAX [P2, P3]`

3.6.2 文字列の操作

使用可能な全操作の概要

文字列操作:

文字列の追加	「+」によって2つの文字列を結合します。 `V.E.str = "Hello" + " world!"` `(-> 結果は "Hello world!")`
+	

左側から文字列の一部を取得	LEFTは、文字列の一番左を先頭文字と判別します。文字列*str*に対して、先頭文字から*nbr*の文字数分を取得します。
LEFT[*str, nbr*]	`V.E.str = LEFT["Hello world!", 5]` `(-> 結果は "Hello")`

文字列の中間の一部を取得	MIDは、文字列の一部を指定します。文字列*str*に対して、posの位置の文字を先頭文字と判別し、先頭文字から*nbr*の文字数分を取得します。
MID[*str, nbr, pos*]	`V.E.str = MID["How are you?", 3, 5]` `(-> 結果は "are")`

右側から文字列の一部を取得	RIGHTは、文字列の一番右の終端文字を、取得する最初の文字と判別します。文字列*str*に対して、終端文字から*nbr*の文字数分を取得します。
RIGHT[*str, nbr*]	`V.E.str = RIGHT["Hello world! How are you?", 12]` `(-> 結果は "How are you?")`

文字列長の判定	LENは文字列長(文字数)を判定します。
LEN[*str*]	`P1 = LEN["Hello world! How are you?"]` `(-> 結果は25)`

FIND[..] は大文字と小文字を区別します!

部分的な文字列の検索	FINDは文字列*str1*内で文字列*str2*を検索します。*str1*内で最初に一致した*str2*の位置が判定結果となります。
FIND[*str1, str2*]	`V.E.str1 = "Hello world! How are you?"` `V.E.str2 = "How"` `P1 = FIND[V.E.str1, V.E.str2]` `(-> 結果は14)` 文字列*str1*内に文字列*str2*が存在しない場合、FINDの判定結果は値0となります。 `V.E.str1 = "Hello world! How are you?"` `V.E.str2 = "today"` `P1 = FIND[V.E.str1, V.E.str2]` `(-> 結果は0)`

部分的な文字列の削除	DELETEは位置*pos*の文字を開始位置として、文字列*str*に対して文字数*nbr*分、削除します。
DELETE[*str, nbr, pos*]	`V.E.str = DELETE["Hello world! How are you?", 5, 7]` `(-> 結果は "Hello !How are you?")`

部分的な文字列の挿入	INSERTは位置*pos*の文字の**後ろを開始位置**として、文字列*str1*に対して文字列*str2*を挿入します。
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]` `(-> 結果は "Hello world!How are you?")`

部分的な文字列の置換	REPLACEは位置*pos*の文字から開始し、文字列*str1*に対して文字数*nbr*を文字列*str2*に置換します。
REPLACE[*str1, str2, nbr, pos*]	`V.E.str1 = "What is your name?"` `V.E.str2 = "age"` `V.E.str = REPLACE[V.E.str1, V.E.str2, 4, 14]` `(-> 結果は "What is your age?")`

比較演算子

比較演算子は大文字と小文字を区別します!

等式	
==	V.E.str1 = "Peter" V.E.str2 = "Peter" $IF V.E.str1 == V.E.str2 #MSG ["%s is equal to %s!", V.E.str1, V.E.str2] $ELSE #MSG ["Strings are not equal!"] $ENDIF (-> **結果は** "Peter is equal to Peter")

不等式	
!=	V.E.str1 = "Peter" V.E.str2 = "Steve" $IF V.E.str1 !=V.E.str2 #MSG ["%s is not equal to %s!", V.E.str1, V.E.str2] $ELSE #MSG ["Strings are equal!"] $ENDIF (-> **結果は** "Peter is not equal to Steve")

次の値よりも大きい / 次の値以上	
>	V.E.str1 = "Peter" V.E.str2 = "Peter" $IF V.E.str1 > V.E.str2 #MSG ["%s is greater than %s!", V.E.str1, V.E.str2] $ELSEIF V.E.str1 >= V.E.str2 #MSG ["%s is greater than or equal to %s!", V.E.str1, V.E.str2] $ENDIF (-> **結果は** "Peter is greater than or equal to Peter!")
>=	V.E.str1 = "Peter" V.E.str2 = "Bob" $IF V.E.str1 > V.E.str2 #MSG ["%s is greater than %s!", V.E.str1, V.E.str2] $ELSEIF V.E.str1 >= V.E.str2 #MSG ["%s is greater than or equal to %s!", V.E.str1, V.E.str2] $ENDIF (-> **結果は** "Peter is greater than Bob!")

次の値よりも小さい / 次の値以下	`V.E.str1 = "Peter"` `V.E.str2 = "Peter"` `$IF V.E.str1 < V.E.str2` `  #MSG ["%s is less than %s!", V.E.str1, V.E.str2]` `$ELSEIF V.E.str1 <= V.E.str2` `  #MSG ["%s is less than or equal to %s!", V.E.str1, V.E.str2]` `$ENDIF` `(-> 結果は "Peter is less than or equal to Peter!")`
<	
<=	`V.E.str1 = "Bob"` `V.E.str2 = "Tim"` `$IF V.E.str1 < V.E.str2` `  #MSG ["%s is less than %s!", V.E.str1, V.E.str2]` `$ELSEIF V.E.str1 <= V.E.str2` `  #MSG ["%s is less than or equal to %s!", V.E.str1, V.E.str2]` `$ENDIF` `(-> 結果は "Bob is less than Tim!")`

変換関数:

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

3.6.3 割り当てられたアドレス文字

以下のアドレス文字には、固定の意味が割り当てられています。

技術要素に関連するアドレス文字:

D、d	*<int, float, expr>*	工具補正
E、e	*<int, float, expr>*	ブロック終端での送り速度(使用不可)
F、f	*<int, float, expr>*	ブロック先頭での送り速度
H、h	*<int, float, expr>*	補助機能
M、m	*<int, float, expr>*	切り替え機能
S、s	*<int, float, expr>*	主軸速度、同期比率など
T、t	*<int, float, expr>*	工具番号

形状に関連するアドレス文字:

G、g	*<int, float, expr>*	準備機能
I、i	*<int, float, expr>*	1番目の軸の補間パラメータ
J、j	*<int, float, expr>*	2番目の軸の補間パラメータ
K、k	*<int, float, expr>*	3番目の軸の補間パラメータ
R、r	*<int, float, expr>*	円の半径

プログラムフローに関連するアドレス文字:

L、l	<string>	グローバルサブプログラムの呼び出し
LL、ll	<string>	ローカルサブプログラムの呼び出し
N、n	<int, float, expr>	ブロック番号
O、o	<int, float, expr>	未使用
$		制御ブロックとして使用するコード
#		拡張言語要素として使用するコード

計算に関連するアドレス文字:

P、p	<int, float, expr>	パラメータ

軸指定用のアドレス文字は、各機械構成に応じて割り当て可能です。通常は、直交空間軸となる3つの直線軸の名称にはX、Y、Zの文字が使用されます。回転軸の名称にはA、B、Cがよく使用されます。それ以外のすべての大文字および小文字の英字も軸の名称に使用できます。

軸名は複数の文字(文字列)で構成することもあります(X_ACHSE、Y22、ZA3)。この場合、軸名を座標の値と区別するために文字「=」を使用します(X1=120.345など)。

このプログラミングマニュアルの以下の章では、最初に命名され、よく使用されるアドレス文字を使用します。

3.6.4 プログラム例

今までに紹介した内容を明確にするために、以下の項にプログラミング例を記載します。

以下のアドレス文字を使用します。

準備機能:

G00 「早送り移動」

G01 「直線補間」

主軸:

S1000 「主軸速度1000 rpm」

送り速度:

F5 「送り速度5 mm/分」

数値軸:

X、Y、Z 「3つの直交軸」

マシンファンクション:

M03 「プログラム指定速度での主軸右回転」

M05 「主軸の停止」

M30 「プログラムの終了」

プログラミング例

```
% 100 (program example)

N10 G00 X100 Y100             (Rapid traverse mode of the X- and Y-axes)
                              (to position 100)
N20 Z100                      (Rapid traverse mode of the Z-axis to 100.)
                              (G00 remains active till deselection by)
                              (another G-function.)
N30 G1 Z50 F5 S1000 M3        (Spindle right-hand side rotation (1000 rpm)
                              (and linear interpolation with feedrate)
                              ((5 m/min) to position Z=50)
N40 Z100                      (Linear interpolation to Z=100)
N50 G0 X200 Y200 Z200         (Rapid traverse to position X=Y=Z=200)
N50 M5                        (Spindle stop)
N60 M30                       (Program end)
```

4 経路情報

4.1 軸指令

軸名を設定することができますが、軸名設定の固有の方法に沿って命名する必要があります。大文字と小文字が区別されて解読処理がなされます。

以下の軸名を使用できます。

- 個々のアドレス文字: {A、B、C、U、V、W、X、Y、Z、Q}
- 等号を使用して値を代入する場合と混同しないように、1つのアドレス文字での軸名のプログラミングの場合には、値と次の文字の間にスペースを入れる必要があります。

例:

軸名として「X」および「X50P1」がNCチャンネルに存在し、軸「X」を位置「50」まで移動する必要がある場合。

X50P1=7	(誤)	軸「X50P1」が位置7まで移動します。
X50 P1=7	(正)	軸「X」が位置50まで移動します。

- 文字列(X_SCHLITTEN、X1、Y22、Z_ACHSEなど)
- 文字列の先頭文字は、予約済みアドレス文字のいずれかと一致する必要があります(上記参照)。それ以降の文字には数字0～9も使用できます。軸名の文字列長は最大許容長(固定値)を超えてはなりません。超えると工ラーメッセージが出力されます。混同を避けるため、1文字以上のすべての軸名の後、「位置」の数値の前に等号(=)を追加する必要があります。特に0～9のいずれかの数値で終了する軸名の場合、等号の追加は必須です。

X1 = <int, float, expr>

例:

X1 = 100.0

X22 = 0.001

X_SCHLITTEN = SIN [30]

Z_ACHSE = SQRT [2]/2

その他、および宣言:

- 各軸名は、「チャンネルパラメータ[1]-5」内に設定する必要があります。
- 軸名の後には、数値または数式が続く必要があります。

X <int, float, expr>	例: X 100.0 Y 0.001 Z SIN [30] A SQRT [2]/2 B 4 * R1/R2

プログラミング例

```
(Used axis designations)
(  Y               )
(  Y50             )
(  Y_ACHSE_SCHL_1  )
(  Z7              )

N010 G01 F1500
N020 Y50 = 51                      (axis Y50 on position 51)
N030 Y52                           (axis Y on position 52)
N040 Y50 Z7 = 54                   (axis Y on position 50 and)
                                   (axis Z7 on position 54)
N050 Y 70 Z7 = 55                  (axis Y on position 70 and)
                                   (axis Z7 on position 55)
N060 Y = 71 Z7 = 56                (axis Y on position 71 and)
                                   (axis Z7 on position 56)
N070 Y[2+3]                        (axis Y on position 5)
N080 Y50 = [4*3]                   (axis Y50 on position 12)
N090 Y_ACHSE_SCHL_1 = 23           (axis Y_ACHSE_SCHL_1 on position 23)
N100 Y50 = P1                      (axis Y50 on position P1)
N110 M30
```

本プログラミングマニュアルでは、一般的に使用される名称X、Y、Zを直交する3つの直線軸に対して使用し、その他の2つの軸にはA、Bを使用しています。

4.2 数値単位系の入力および精度範囲

位置、角度、および送り速度の指定に使用する数値単位系は、ユーザ設定に応じて適用されます。

長さおよび位置:	mmまたはインチ
移動速度:	mmまたはインチ / 秒または分
角度:	度またはグラード(ゴン)
角速度:	度またはグラード(ゴン) / 秒または分

プログラマは、パラメータ計算機能を使うことでユーザ定義した数値単位系を使用することが可能です。

長さの指定および移動速度は、すべて0.1 µmの標準精度で計算されます。この分解能での最大移動範囲は、-214 m ~ +214mとなります。プログラミングにおいて、数値入力が-214 000.0000 mm ~ +214 000.0000 mmの範囲を超えてはなりません。数式の結果が許容範囲内となる場合、数式の個別の要素(パラメータなど)はこの範囲を超えても構いません。オフセットおよび補正を組み合わせた上で数値範囲を超えてはいけないということも考慮してください。

例外: 円の半径の定義は、最大10^9 mmまで可能です。ただし、円弧の終点が常に軸の最大移動範囲(-2.14 ~ +2.14) * 10^5 mm内に存在する必要があります。

機械データとして、軸のタイプを入力しておく必要があります。回転軸または主軸の場合には、角度は0.0001度の分解能で処理されます。つまり、n * 360度(n=11900)の範囲をプログラムできます

例えば、0.0001°の分解能の場合、角度は以下の範囲になります。

(-2.14 ~ +2.14) * 105 / 360 = 2 * 594　　回転

4.3 座標系

機械基準を設定することで、制御のための機械原点、または機械座標が定まります。X100などといったNCプログラムで指定されるプログラム座標(*インデックスp*)は、絶対座標(*インデックスa*)と一致しています。

$$x_a = x_p$$

$$y_a = y_p$$

ワーク座標系の定義することで、オフセットが設定されます。ワーク座標系の位置は、実際の機械軸によって定義された座標系とは異なります。単一軸内でプログラムされており一定である直動オフセットと、キネマティックトランスフォーメーション(シリンダ⇔直交など)や形状のトランスフォーメーション(工具径補正、ミラーリングなど)といった通常複数の軸に影響を与える動的オフセットとを、区別する必要があります。

例えば、ゼロオフセット機能(NPV-G54 ~ G59)によってワーク原点Wまたはワーク座標系から自由に選択し、機械原点Mから原点をシフトすることができます。絶対座標は、プログラム座標にNPVを追加したものになります。

$$x_a = x_{NPV} + x_p$$

$$y_a = y_{NPV} + y_p$$

ゼロオフセット機能によって指定されるこれらのNPV以外でも、オフセットタイプのG92 X... Y... Z...などを使用することでプログラム内で明示的にプログラムすることができます。

この座標プリセット(BPV)は元々のNPVに追加されます。これにより、絶対座標は以下のように特定できます(図3-1)。

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

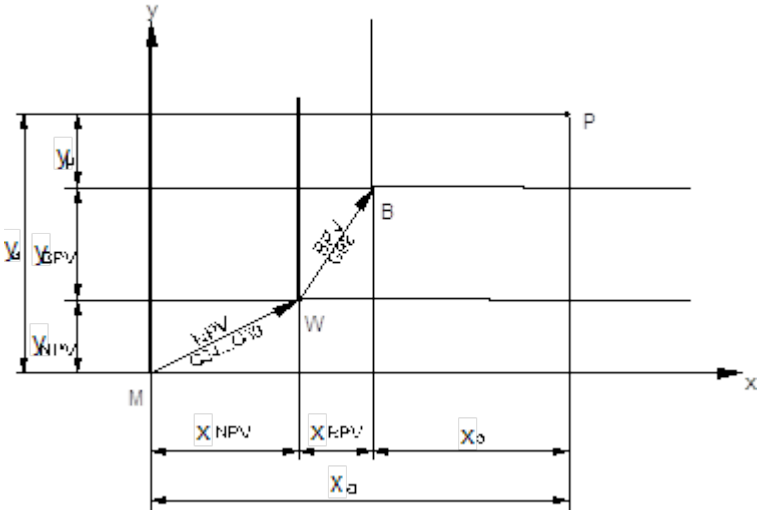


図 1: 図3-1: 座標シフトで利用できるものの説明

x_a, y_a	絶対座標
x_p, y_p	プログラム座標
x_{NPV}, y_{NPV}	ゼロオフセット
x_{BPV}, y_{BPV}	座標プリセット
M:	機械原点
W:	ワーク原点
B:	参考座標点
P:	位置

MDIの座標表示で確認できる物理機械軸(ACS)とワーク座標(PCS)の差によって、有効なオフセット量が判別できます。ただし、機械座標やワーク座標を、工具径補正やミラーリングなどで操作した結果生じるオフセットの場合は、座標の差はなくなります。

以下の表に、オフセット付加機能の概要を記載します。より詳細の情報を確認できます。

考慮すべき事項:

有効化および**無効化**は、座標変更により座標が変わり、MDI上にオフセットが表示されたタイミングです。ただし、有効化や無効化を行った後に、最初に物理的に動作させて初めて適用されます。例えば、プログラム終端での無効化は、以降のプログラムの最初の動作から適用されます。

プログラムで指定できるオフセット(直線、一定)

番号	説明	定義	有効な場合 ACS – PCS に差があるか	有効化	無効化	一時的な機能停止
1	座標プリセット	NCプログラム	あり	NCブロック „*G92 X.. Y..*“ G90/91を考慮する必要あり	NCブロック „*G92 X0 Y0.*“または NCプログラムをスタート	„#SUPPRESS OFFSETS“、„#MCS ON“
2	ゼロオフセット	パラメータリスト、NCプログラム	あり	NCブロック „*G54...G59*”	NCブロック„G53“ または NCプログラムをスタート	„#SUPPRESS OFFSETS“、„#MCS ON“
3	クランプ位置オフセット	パラメータリスト	あり	NCプログラムをスタート、NCプログラムによる変更不可	NCプログラムをスタート、NCプログラムによる変更不可	„#SUPPRESS OFFSETS“、„#MCS ON“
4	工具オフセット	パラメータリスト、NCプログラム、外部入力	あり	NCブロック„*D..*“	NCブロック„D0“ または NCプログラムをスタート	„#SUPPRESS OFFSETS“、„#MCS ON“
5	位置プリセット	NCプログラム	あり	NCブロック „#PSET...“	NCブロック „#PRESET...“ または NCプログラムをスタート	„#SUPPRESS OFFSETS“、„#MCS ON“
6	CSオフセット	NCプログラム	あり	NCブロック „*#CS ON [vx,vy,vz,..*”	NCブロック „#CS OFF“ または NCプログラムを終了	„#MCS ON“
7	ACSオフセット	NCプログラム	あり	NCブロック „*#ACS ON[vx,vy,vz,..*“	NCブロック „#ACS OFF“ または NCプログラム終了	„#MCS ON“

形状のトランスフォーメーションによるオフセット(直線、動的)

番号	説明	定義	有効な場合 ACS – PCS の差があるか	有効化	無効化	一時的な機能停止
8	CS	NCプログラム	あり	NCブロック „#CS ON[..,..,..]“	NCブロック „#CS OFF” NCプログラムを終了	„#MCS ON“
9	ACS	NCプログラム	あり	NCブロック „#ACS ON[..,..,..]“	NCブロック „#ACS OFF” NCプログラムを終了	„#MCS ON“
10	形状回転	NCプログラム	なし	NCブロック „#ROTATION ON[ANGLE..]”	NCブロック „#ROTATION OFF”	„#MCS ON“
11	ミラーリング	NCプログラム	なし	G21/G22/G23後のNC動作ブロック	G20後のNC動作ブロック	不可
12	工具径補正	NCプログラム	なし	NCブロックG41/G42	NCブロックG40	不可
13	キネマティックトランスフォーメーション	NCプログラム	なし	NCブロック „#RTCP ON“ プログラムをスタート自動	NCブロック „#RTCP OFF”	„#MCS ON“

特殊機能によるオフセット

番号	説明	定義	有効な場合 ACS – PCS の差があるか	有効化	無効化	一時的な機能停止
14	並行補間を用いた手動モードでのオフセット	手動ハンドル、NCプログラム	あり	NCブロック „G201“	NCブロック„G202“	不可
15	計測によるオフセット	NCプログラム	あり	NCブロック „G101“	NCブロック„G102“	„#SUPPRESS OFFSETS“、„#MCS ON“
16	原点復帰後のオフセット	NCプログラム	なし	NCブロック „G74 X.. Y.. Z.. “	不可	不可

NCコマンド#SUPPRESS OFFSETSはモーダルではないため、現在のNCブロックにのみ影響を及ぼします。

NCコマンド#MCS ONは、コマンド#MCS OFFがプログラムされるまであらゆるオフセットを無効にします。

各(A)CSにおいて、タイプが1、2、および5のオフセットは「ローカル」に保存されます。

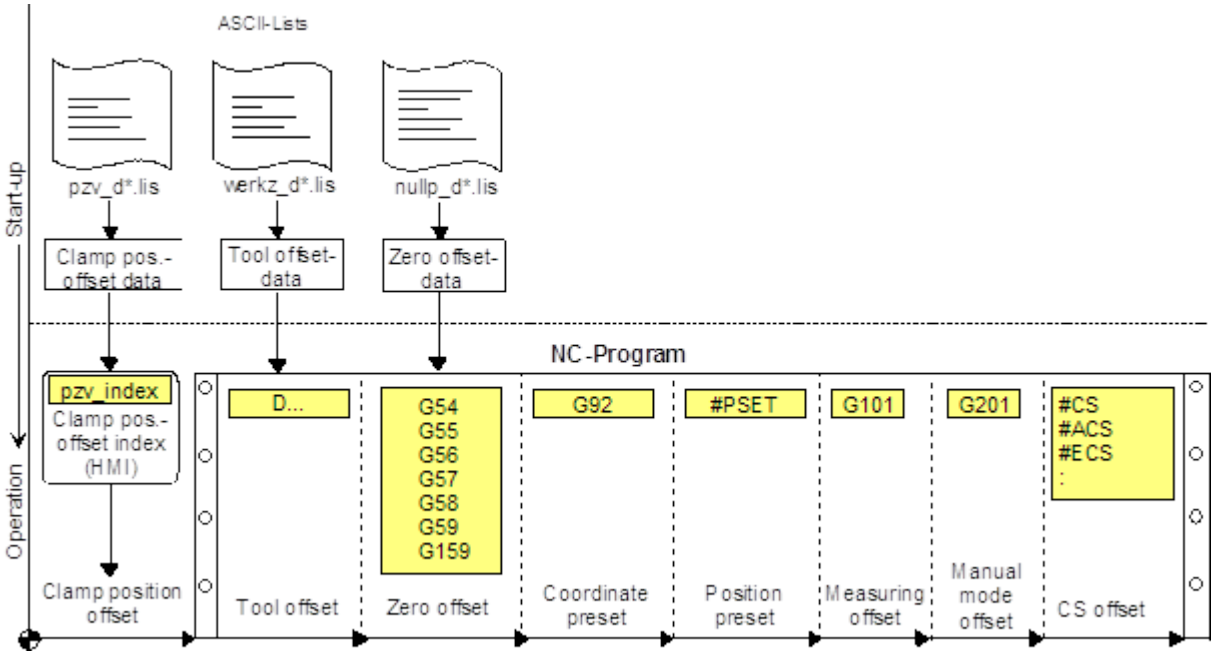


図 2: 図3-2 : 座標系へのオフセット付加の概要

5 Gファンクション

いわゆる「G」ファンクションにより、移動のタイプ、補間のタイプ、計測方法、時間に影響する項目、および特定機能の有効化などを指定することができます。構文は、以下の通りです。

G *<expr>*

Gファンクションには、様々に異なる特性があります。

効果: プログラミングすると対応するブロックに対してのみ有効になるGファンクション(ノンモーダル)と、最初のプログラミングの後に明示的に選択が解除されるまで有効になるGファンクション(モーダル)があります。

例外: 相互に排他的なGファンクションもあります。例えば、G01 (直線補間)とG02 (円弧補間)は同時に選択できません。このため、こういったグループに含まれるGファンクションは、1つのNCブロック内でプログラムしてはなりません。

後ろのブロックで同一グループの他のファンクションが指令された場合、モーダルファンクションは自動的に解除されます。

プログラミング例

```
:
N50 G01 X100 Y200                       (Linear interpolation active)
N60 G41 X200 Y200                       (Linear interpolation active)
N50 X300 Y250                           (Linear interpolation active)
N50 X100 Y50                            (Linear interpolation active)
N50 G02 X100 Y50 I100                   (Circular interpolation active)
:
```

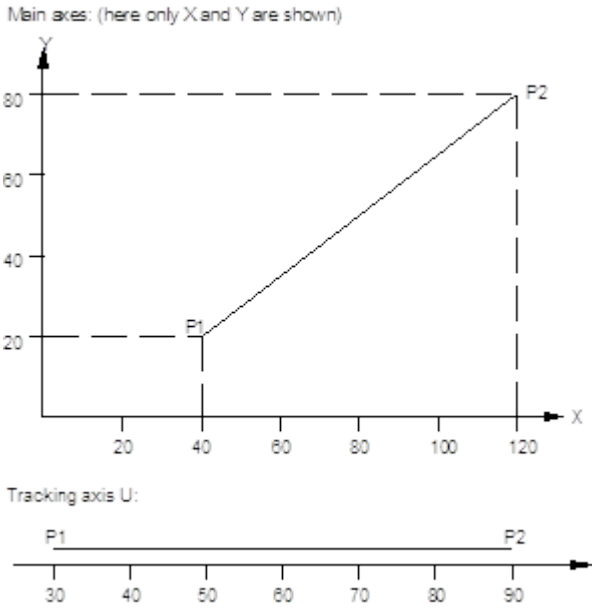
基本設定: 制御コントローラは切り替え中、RESET後、またはプログラム終端で、Gファンクションを基本設定に戻します。この基本設定では、明示的に指令しなくてもいくつかのGファンクションが既に有効になっています。

5.1 経路準備ファンクション

5.1.1 早送りG00

G00	早送りでの直線補間	(モーダル)

G00が選択されている場合、機械パラメータ内で指定された軸の早送り速度が使用されます。この場合、早送りで移動する複数の軸の条件によって最終的な軸の速度が決まります。直交座標系(X、Y、Z)において任意の数の直線動作をプログラムすることができます。メインとなる軸の動作開始および動作終了と同時になるように、プログラムされている全ての従軸が直線速度で動作します。



アブソリュート指令入力:

```
Nnnnn G00 G90 X120 Y80 U90        (traverse from P1 to P2)
```

インクリメンタル指令入力:

```
Nnnnn G00 G91 X80 Y60 U60         (traverse from P1 to P2)
```

図4-1: 早送りでの位置決め

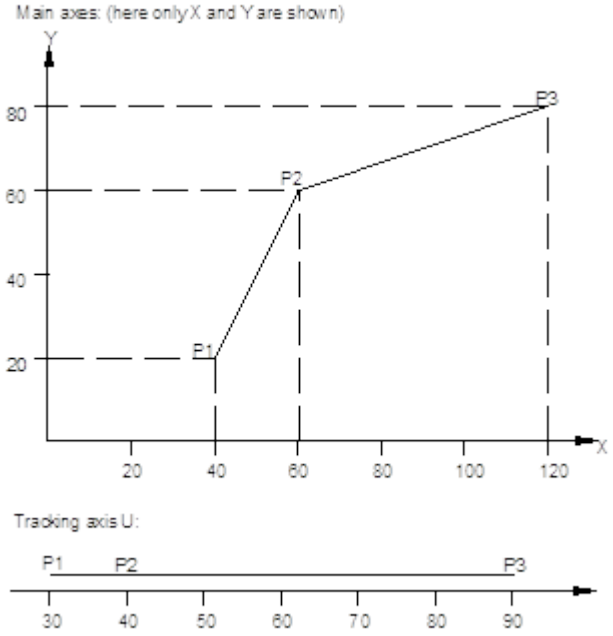
特殊な例: G00 G90を使用する回転軸の割り出し

回転軸がG00でプログラムされ、G90 (G90: 絶対プログラミング)が有効な場合、プログラムされた目標位置は剰余で計算されます。つまり、回転軸は最大で半周回転します。

5.1.2 直線補間G01

G01	プログラムされた送り速度での直線補間	(モーダル)

G01が選択されていると、Fワード(設定されているmm/分などの単位)によって指定されている送り速度で、プログラムされた目標位置まで直線的な経路で移動します。直交座標系(X、Y、Z)において任意の数の直線動作をプログラムすることができます。メインとなる軸の動作開始および動作終了と同時になるように、プログラムされている全ての従軸が直線速度で動作します。



アブソリュート指令入力:

```
Nnn G01 G90 X60 Y60 U40 F1000      (traverse from P1 to P2 feedrate)
                                   (1000mm/min)
Nnn X120 Y80 U90              (traverse from P2 to P3 feedrate 1000mm/min)
```

インクリメンタル指令入力:

```
Nnn G01 G91  X20 Y40 U10 F1000     (traverse from P1 to P2 feedrate)
                                   (1000mm/min)
Nnn X60 Y20 U50               (traverse from P2 to P3 feedrate 1000mm/min)
```

図4-2: 直線補間(G01)

5.1.3 円弧補間(G02/G03)

G02	円弧補間(右回り、CW)	(モーダル)
G03	円弧補間(左回り、CCW)	(モーダル)

G02 (右回り、CW)またはG03 (左回り、CCW)が選択されている場合、Fワードで指定された送り速度で、プログラムされた目標位置まで円弧移動します。円弧移動は、空間座標系(XY、ZX、YZ平面)の3つの指定平面で実行できます。指定平面の選択は、ファンクションG17、G18、G19を使用して行います(「平面指定 (G17/G18/G19) [▶ 77]」の章も参照)。

メインとなる軸の動作開始および動作終了と同時になるように、プログラムされている全ての従軸が直線速度で動作します。

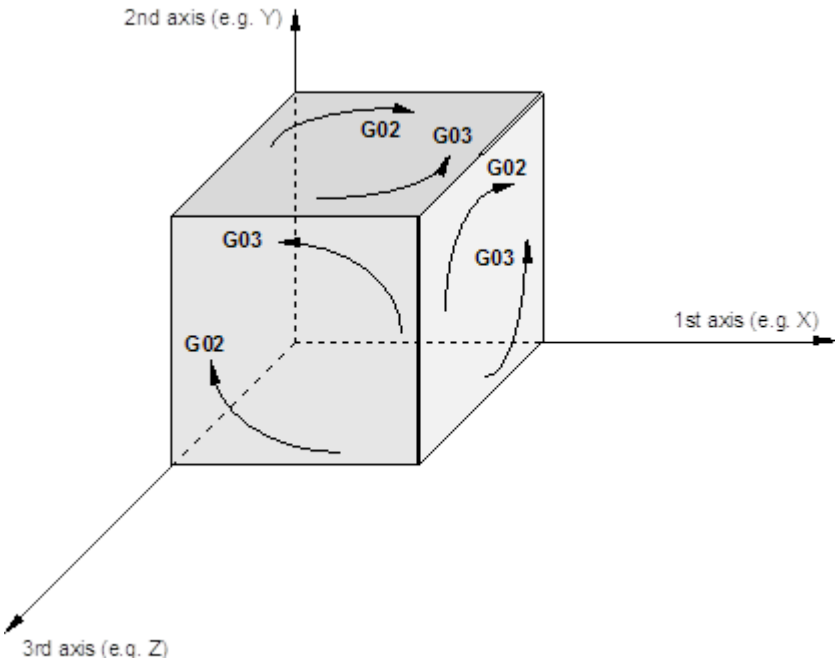


図 3: 図4-3: 円弧機能G02およびG03の説明

円弧の定義には、円弧「Ka」の起点(前のブロックで特定)、円弧「Ke」の終点、および円弧中心「Km」が使用されます。G162が有効である場合には、円弧起点からの相対位置を示す補間パラメータI、J、Kにより、円弧の中心点が指定されます。G161が有効な場合には、絶対位置となります。

G162: (初期設定)

I - 円弧起点からKmへの相対位置(X軸方向)

J - 円弧起点からKmへの相対位置(Y軸方向)

K - 円弧起点からKmへの相対位置(Z軸方向)

G161:

I - Kmの絶対位置(X軸)

J - Kmの絶対位置(X軸)

K - Kmの絶対位置(Z軸)

中心点補正(G165)がオフの場合、円弧の中心点の定義が間違っていると工ラーメッセージが出力されます。G165が有効な場合には、円弧を移動できるように中心点が設定されます。補間パラメータがプログラムされていない場合には、円弧の中心点補正はI、J、K = 0の状態から開始することになります。また、円弧の中心点補正の機能は「ノンモーダル」になります。

G02/G03を用いて、円弧の終点を指定せずにプログラムした場合、一周円が移動します。

円弧の最大許容半径は、10^9 mmです。
ただし、円弧の終点が軸の最大移動範囲である$\pm 2.14*10^5$ mmを超えてはなりません。

G17平面の場合の記述例:

G02 | **G03** [**X***<expr>***Y***<expr>***] **I***<expr>***J***<expr>*** | **R***<expr>*

G02 | G03 円弧補間CW / CCW

X*<expr>* Y*<expr>*　　XY平面での終点

I*<expr>* J*<expr>*　　XY平面での円弧補間の中心点位置(I　-> X軸方向、J -> Y軸方向)。G161/G162により指定値が変わります。

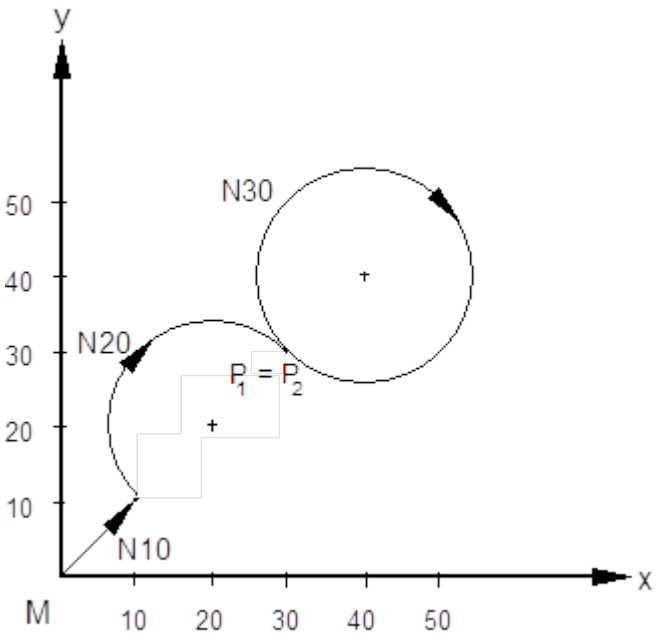
R*<expr>*　　円弧の半径(I、Jの代わりに指定)

プログラミング例

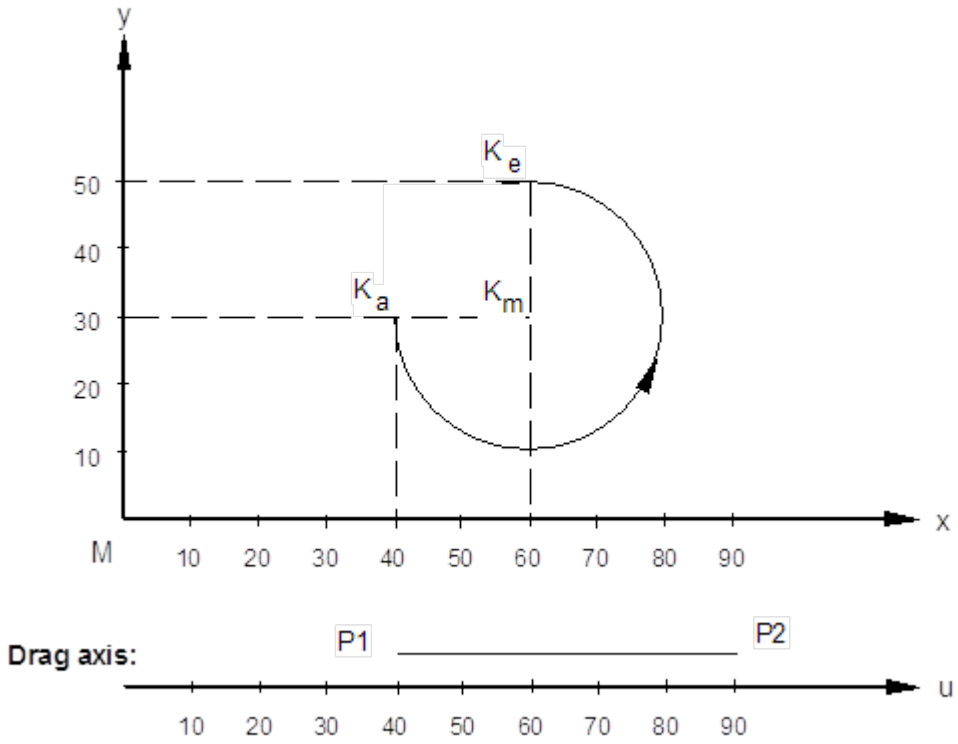
```
N10  G01  X10  Y10
N20  G02  X30  Y30  I10  J10    (Semicircle, circle end point X30 Y30)
N30                 I10  J10    (Full circle)
N40       X50  Y50              (Error message, since no center point or)
                                 (radius was specified)
```

指定平面ごとの記述方法:

平面	円弧補間タイプ	平面内の終点	中心点/半径
G17	G02/G03	X..Y..	I..J../R
G18	G02/G03	Z..X..	K..I../R
G19	G02/G03	Y..Z..	J..K../R



Main axes (here only X and Y):



アブソリュート指令入力:

```
Nnn G90 F1000                          (Absolute dimension, feedrate)
Nnn G17                                (Selection of  X-Y-plane)
Nnn G03 G161 X60 Y50 I60 J30 U90       (Circle: Ka -> Ke and)
                                       (linear interpolation: P1 -> P2)
```

インクリメンタル指令入力:

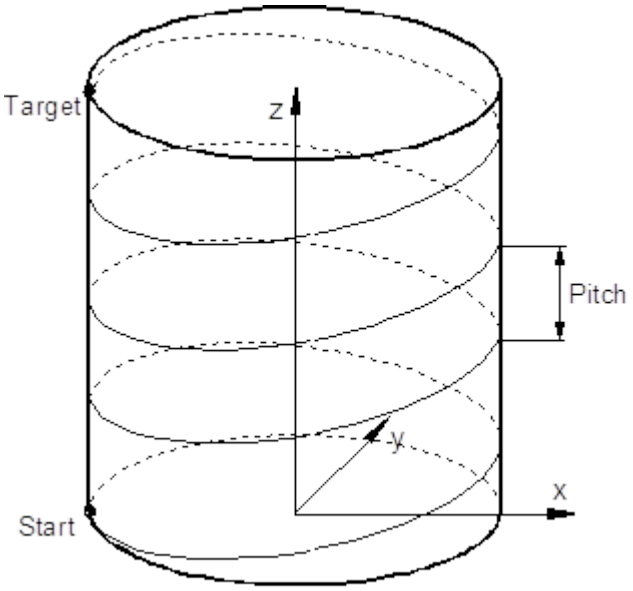
```
Nnn G91 F1000                          (Incr. dimension, feedrate)
Nnn G17                                (Selection  X-Y-plane)
Nnn G03 G162 X20 Y20 I20 U50           (Circle : Ka -> Ke and)
                                       (linear interpolation: P1 -> P2)
```

図4-4: 円弧補間の例

円弧は半径指定によってプログラムすることも可能です。半径での指定方法は、G163=「 」としたり、アドレス文字R=「半径値」としたりすることにより可能です。また、R1=「半径値」によって定義しても同一の結果となります(「円弧定義の中心点指定(G161/G162) [▶ 130]」、「半径プログラミング(R、G163) [▶ 130]」、「 中心点オフセット制御(G164/G165) [▶ 133]」の章も参照)。

5.1.4 ヘリカル補間

ヘリカル補間は、第1軸および第2軸の円弧補間と、第3軸の直線補間とを重ね合わせた動作です。その結果、一定のピッチでらせん状の動作が実行されます。ピッチは、選択した平面に応じて定まる円弧補間の3番目のパラメータによってプログラムします。



G17平面の場合の記述例:

G02 | **G03X***<expr>***Y***<expr>***Z***<expr>***I***<expr>***J***<expr>* | **R***<expr>***K***<expr>*

G02 | G03	円弧補間CW / CCW
X*<expr>* Y*<expr>*	XY平面の終点
Z*<expr>*	XY平面と直交するZ軸のらせん終点における座標値
I*<expr>* J *<expr>*	XY平面での円弧補間の中心点位置(I -> X軸方向、J -> Y軸方向)。G161/G162により指定値が変わります。
R*<expr>*	円弧の半径(I、Jの代わりに指定)
K*<expr>*	Z軸方向のらせんピッチ(通常は**符号なし**の値)

指定平面ごとの記述方法:

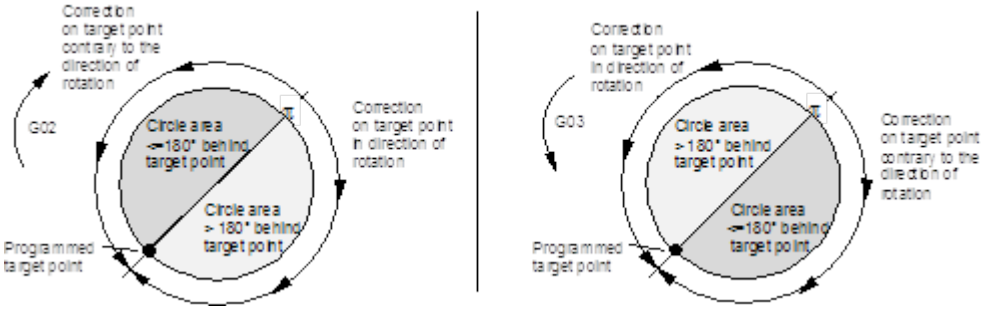
平面	円弧補間タイプ	目標経路が乗る指定平面	目標経路が沿うらせん軸	円弧中心/半径	ピッチ
G17	G02/G03	X..Y..	Z..	I..J../R	K
G18	G02/G03	Z..X..	X..Y..	K..I../R	J
G19	G02/G03	Y..Z..	X..	J..K../R	I

プログラムした目標点にらせんが正確に到達するようにピッチを定義する必要はありません。このような場合、NCカーネルによって固定点である起点と目標点に対して、可能な限り近くなるように補正したピッチが計算されます。

この計算を行うために、まず最初にプログラムしたピッチに基づいてらせんの目標点が計算されます。この計算された目標点がプログラムした目標点と異なる場合、補正が必要になります。補正の基準は、プログラムした目標点から計算された目標点への、回転方向に沿った距離です。

間隔がπ(180°)以下の場合、計算されたらせんの目標点は回転の**逆方向**に沿ってプログラムした目標点に移動します。つまり、ピッチが増加します。

間隔がπ(180°)を超える場合、計算されたらせんの目標点は回転の**順方向**に沿ってプログラムした目標点に移動します。つまり、ピッチが減少します。



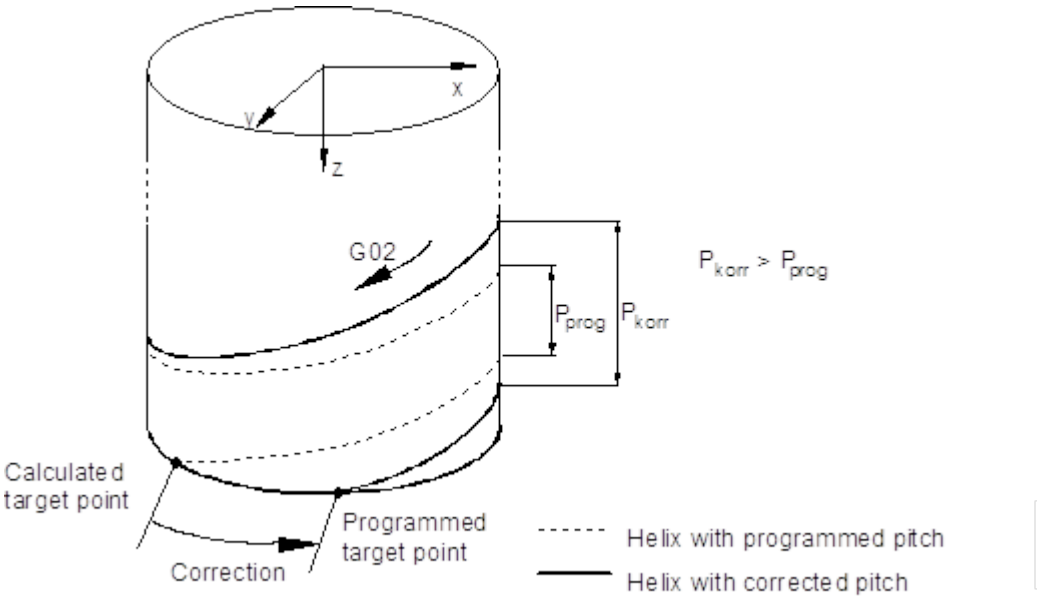
回転方向によるらせんピッチの補正方法

例1:

右回りのらせんの補正方法(G02)

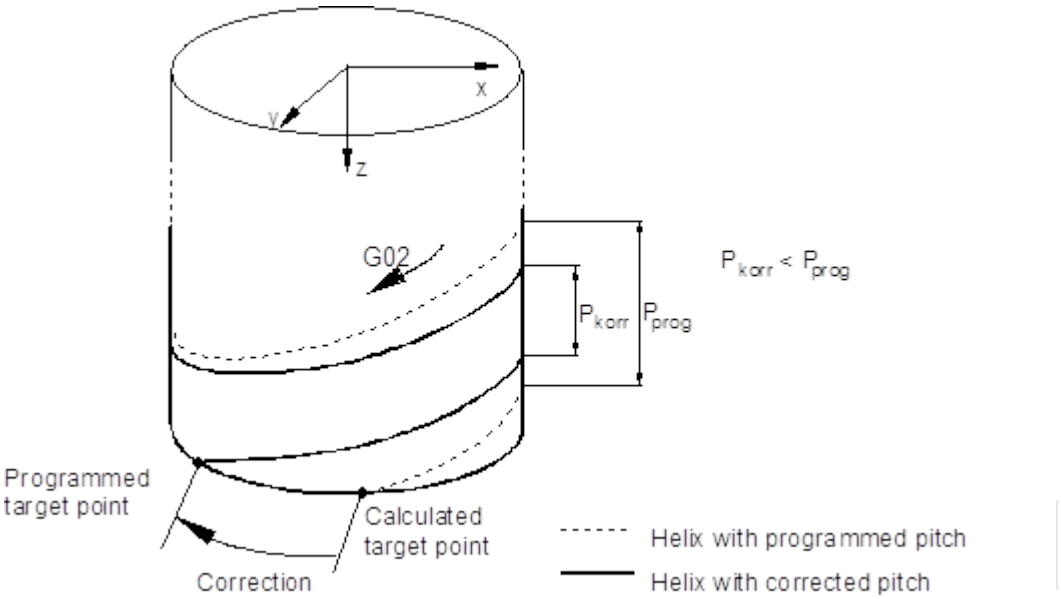
ケース1:

プログラムした目標点に対して、プログラムしたピッチP_{prog}を使用して計算された目標点が、前方180°以内の範囲に存在しています。補正する場合は、ピッチP_{korr}を増加させる必要があります。



ケース2:

プログラムした目標点に対して、プログラムしたピッチP_{prog}を使用して計算された目標点が、後方180°以内の範囲に存在しています。補正する場合は、ピッチP_{korr}を減少させる必要があります。



例2:

XY平面に対する右回りのヘリカル補間らせん:

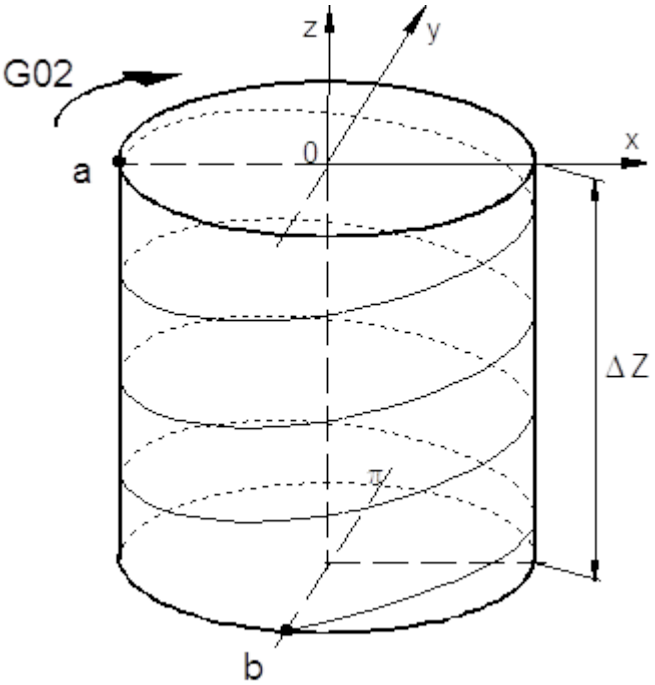
起点a: X-10 Y0 Z0

目標点b: X0 Y-10 Z-20

らせんの中心I、J: 原点

らせんピッチK: 変数

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



最小回転: 3/4 → ピッチK=26.66

ピッチ K ≥ 26.66 の場合:

最大許容値であるピッチK = 26.66に制限され、aからbまでのらせん経路は¾回転分のみ実行されます。

ピッチ K < 26.666 の場合:

プログラムされた ピッチK (単位mm)	aからbまでの らせん回転	補正後の ピッチK (単位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

5.1.4.1 単純なヘリカル補間

単純なヘリカル補間では、ピッチは定義されず、目標点のみが定義されます。目標点により動作が変わり、最大で1回転するらせん動作となります。

G17平面の場合の記述例:

G02 | **G03X***<expr>***Y***<expr>***Z***<expr>***I***<expr>***J***<expr>* | **R***<expr>*

G02 | G03	円弧補間CW / CCW
X*<expr>* Y*<expr>*	XY平面の終点
Z*<expr>*	XY平面と直交するZ軸のらせん終点における座標値
I*<expr>* J *<expr>*	XY平面での円弧補間の中心点位置(I -> X軸方向、J -> Y軸方向)。 G161/G162により指定方法が変わります。
R*<expr>*	円弧の半径(I、Jの代わりに指定)

指定平面ごとの記述方法:

平面	円弧補間タイプ	目標経路が乗る 指定平面	目標経路に沿う らせん軸	円弧中心/半径
G17	G02/G03	X..Y..	Z..	I..J../R
G18	G02/G03	Z..X..	X..Y..	K..I../R
G19	G02/G03	Y..Z..	X..	J..K../R

例:

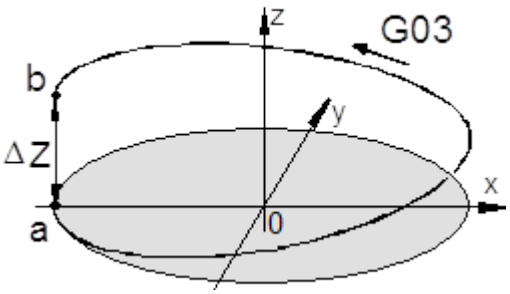
XY平面に対する左回りのヘリカル補間らせん:

起点a: X-10 Y0 Z0

目標点b: Z20

らせんの中心I、J: 原点

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



5.1.5 ドウェル(G04)、(#TIME)

G04 *<1.main_axis><expr>* | *<expr>* | *<main_spindle><expr>*	(ノンモーダル)

G04	ドウェル時間
<1.main_axis><expr>	ドウェル時間は**最初の軸**の名前で秒単位で指定します。
<expr>	または、ドウェル時間を秒単位で直接指定します。
<main_spindle><expr>	または、**スピンドル軸**の名前と回転数によって指定します(V2.11.2023.02以降)。

ドウェルは、形状切削やMファンクションなどで必要です。

ドウェルはNCブロック内でのみプログラム可能です(例外: ブロック番号)。

プログラミング例

```
N10 G04 X4.5            (wait for 4.5 seconds)

or

N20 G04 3.0             (wait for 3.0 seconds)


or

N30 G04 S5              (wait for 5 roundings)
```

その他の方法として、#TIMEという拡張機能でもドウェルを使用できます。

#TIME *<expr>*	(ノンモーダル)

#TIME ドウェル時間

<expr> ドウェル時間の値は秒です。

プログラミング例

```
N10 #TIME 2.5           (wait for 2.5 seconds)
```

5.1.6 プログラムでの原点復帰(G74)

G74	原点復帰の実行	(ノンモーダル)

G74を使用すると、NCプログラムによって原点復帰の実行が可能です。原点は軸の基準となる位置です。また、軸の原点復帰を実行する順番を指定する必要があります。軸名と共に指定した値により、原点復帰を行う順番を決まります。

指定した値が同じ軸に対しては、原点復帰が同時に開始されます。

基準点復帰や原点復帰についてのより詳細の情報は、機能説明「原点復帰」[FCT-M1]に記載されています。

プログラミング例

軸ごとに順番に実行するコマンド:

```
N10 G74 X2 Y3 Z1      Sequence of homing: Z-> X -> Y
```

同時に実行するコマンド:

```
N10 G74 X1 Y1 Z1      Sequence of homing: X,Y,Z at the same time
```

5.1.7 座標プリセット(G92)

G92	座標プリセット	(ノンモーダル)

G92を使用すると、任意の軸に対してプログラムから値を指定して、ゼロオフセット機能に追加する形でオフセットすることができます。G90/G91の設定によって、絶対値としてオフセット値を設定するか、現在のオフセット設定値に追加するかが決まります。

G92の指令はノンモーダルです。ただし、座標プリセット自体はもちろん、次のG92がプログラムされるまで有効です。

プログラミング例

```
N10 G90                (Absolute dimensional specification)
N20 G92 X10 Z30        (Displaces the programmed and the absolute)
 .                     (coordinates by 10 in X, 30 in Z)
 .
 .
Nnn G92 X0 Z0          (Reset of coordinate preset to 0)
```

G91モードでの座標プリセットの変更:
N10 G92 X10 Y20

N20 G00 X0 G91
上記のプログラムでは、X軸は動作しません(相対移動距離が0と指定されているため)。座標プリセットは、例えばアブソリュートモード(G90)で次に軸が動作した際に、初めて有効になります。

5.1.8 マイナス側のソフトウェアリミットスイッチの設定(G98)

G98	マイナス側のソフトウェアリミットスイッチの設定	(ノンモーダル)

G98は、プログラムされたすべての軸に対して、マイナス側のソフトウェアリミットスイッチの座標を設定します。G90/G91の設定によって、ソフトウェアリミットスイッチの座標を絶対座標として設定するか、以前の設定値に累積するように加えるかが決まります。

マイナス側のリミットスイッチの値は、各軸に対応した変数に保存されます。

V.A.-SWE.X、V.A.-SWE.Y、V.A.-SWE.Zなど。

(「軸特有の変数(V.A.) [▶ 421]」の章も参照)。

G98の指令は「**ノンモーダル**」です。ただし、ソフトウェアリミットスイッチの機能自体はもちろん、プログラム中は常に有効です。

機械が起動すると、軸パラメータP-AXIS-00177の値が初期設定値として有効になります。

プログラムでG98を指令することで、このリミット値をさらに限定することはできますが、広げることはできません。つまり、軸パラメータリスト内で定義されたリミット値は、G98では広げられません。

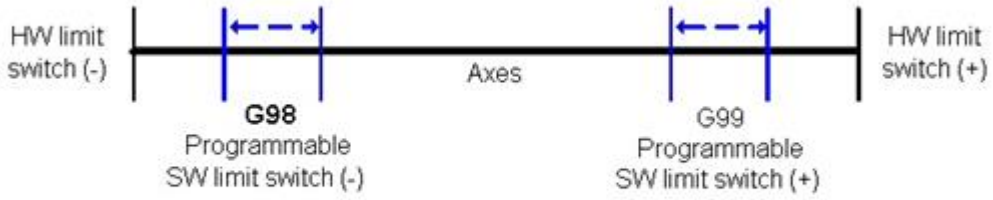
途中で軸構成を変更することのなく動作させている場合には、変更したリミット値はプログラム終端まで有効であり、次に実行するNCプログラム内でも有効になります。CNCリセットを実行し、その後NCプログラムをスタートさせた場合のみ、再度初期値が有効になります。

途中で軸構成を変更して動作するような場合には、チャンネルに対して軸構成を書き換えた際に、初期値へのリセットが実行されます。

G98の機能により影響を与えるのは、独立軸または単一軸が移動する経路のみが対象となります。手動モードの移動範囲には影響がありません。手動モードの移動範囲はコマンド#MANUAL LIMITS […] [▶ 156]で定められます。

プログラミング例

```
N10  G90
...
N100 G98 X-1000 Y-2000              Sets negative software limit switch in
                                    X to -1000 and in Y to -2000
```



5.1.9 プラス側のソフトウェアリミットスイッチの設定(G99)

G99	プラス側のソフトウェアリミットスイッチの設定	(ノンモーダル)

G99は、プログラムされたすべての軸に対して、プラス側のリミットスイッチの座標を設定します。G90/G91の設定によって、ソフトウェアリミットスイッチの座標を絶対座標として設定するか、以前の設定値に累積するように加えるかが決まります。

プラス側のリミットスイッチの位置は、各軸に対応した変数に保存されます。

V.A.+SWE.X、V.A.+SWE.Y、V.A.+SWE.Zなど。

(「13.1 軸固有の変数」の章も参照)。

G99の指令は「**ノンモーダル**」です。ただし、ソフトウェアリミットスイッチの機能自体はもちろん、プログラム中は常に有効です。

機械が起動すると、軸パラメータP-AXIS-00178の値が初期設定値として有効になります。

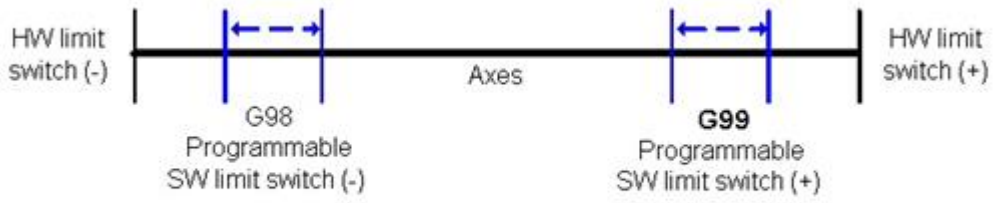
プログラムでG99を指令することで、このリミット値をさらに限定することはできますが、広げることはできません。つまり、軸パラメータリスト内で定義されたリミット値は、G99では広げられません。

変更したリミット値はプログラム終端まで有効であり、次に実行するNCプログラム内でも有効になります。CNCリセットを実行し、その後NCプログラムをスタートさせた場合のみ、再度初期値が有効になります。

G99の機能により影響を与えるのは、独立軸または単一軸が移動する経路のみが対象となります。手動モードの移動範囲には影響がありません。手動モードの移動範囲はコマンド#MANUAL LIMITS [...] [▶ 156]で定められます。

プログラミング例

```
N10  G90
...
N10 G99 X+1000 Y+2000              Sets positive software limit switch in
                                   X to +1000 and in Y to +2000
```



5.1.10 計測機能

制御コントローラを起動すると、チャンネルパラメータリストのP-CHAN-00057で初期設定されている計測タイプが有効になります。プログラム内において#MEAS MODE [▶ 241]または#MEAS [TYPE..] [▶ 241]を使用すると、いつでも計測タイプを変更することができます。

NCプログラム内では、変数V.G.MEAS_TYPE [▶ 425]にて現在有効な計測タイプを確認することができます。以下の7種類の計測タイプが利用できます。

計測タイプ	内容
1*	1つ以上の複数軸に対して、動作させながら計測します。 計測する送り速度は、プログラム内でFワードにより指定します。
2*	1つの軸に対してのみ、動作させながら計測します。 計測する送り速度は、P-AXIS-00215にて指定します。
3	1つ以上の複数軸に対して、動作させながら計測します。 計測する送り速度は、プログラム内でFワードにより指定します。 計測後も終点まで移動する設定も可能です。
4	最大3軸に対して、動作させながら計測します。 計測する送り速度は、プログラム内でFワードにより指定します。
5	1つ以上の複数軸に対して動作させながら計測し、計測により中断分岐します。 計測する送り速度は、プログラム内でFワードにより指定します。
6	1つ以上の複数のSERCOS軸に対して動作させながら計測し、計測により中断分岐します。 計測する送り速度は、プログラム内でFワードにより指定します。
7*	1つ以上の複数軸に対して動作しながら計測し、軸停止の発生を検出します。計測する送り速度は、プログラム内でFワードにより指定します。

* この計測タイプでは、独立軸 [▶ 543]の計測も可能です。

NCプログラムでは、計測に関連する以下の変数を使用できます(「軸固有の変数(V.A.)」 [▶ 421]の章も参照)。

V.A.MERF.*<axis>*	計測動作が完了しているか?
V.A.MESS.*<axis>*	すべてのオフセットを含む、現在の座標系で計測された座標値
V.A.MOFFS.*<axis>*	計測オフセット値
V.A.MEIN.*<axis>*	組み込まれている計測オフセット値

バージョンV2.11.2020.07以降では、軸固有の変数V.A.MEAS.ACS.VALUEおよびV.A.MEAS.PCS.VALUEを変数V.A.MESSの代わりに使用できます。追加されたこれらの変数により、「すべてのオフセットを含めた座標系における座標値」と「オフセットを含めないプログラミング座標系における座標値」の両方を利用できます。

V.A.MEAS.ACS.VALUE.*<axis>*	すべてのオフセットを含む、現在の座標系で計測された座標値
V.A.MEAS.PCS.VALUE.*<axis>*	オフセットを含まない、プログラミング座標系で計測された座標値

制限事項:

以下の場合、計測動作のプログラミングはできません。

- 多項式曲線での形状切削(G261、G61)が有効である。
- スプライン補間(AKIMA、Bスプライン)が有効である。
- HSC機能が有効である。

5.1.10.1 複数軸に対する計測(G100) (タイプ1)

G100 <*axis_name*><*expr*> { <*axis_name*><*expr*> } [**F**<*expr*>]	(ノンモーダル)

G100 計測動作の指定

<axis_name><expr> 計測動作する軸の目標点

F*<expr>* 計測する送り速度

あらゆる軸に対して計測動作が可能です。計測ブロックで指定されたすべての軸は、計測軸に設定(P-AXIS-00118)されていることが必要です。計測方法(タイプ1)が、P-CHAN-00057で指定されている必要があります。

計測ブロック内での計測中に、計測信号が発生すると記録がなされます。始点およびNCコマンドにて指定した目標点との間で、直線補間が実行されます(G01と同様です)。本マニュアルでは、計測ブロックでの経路速度を「計測する送り速度」と呼びます。1つ以上の軸が計測動作に指定されている必要があります。計測する送り速度は、Fワードによって指定します。計測ブロックでの動作距離は、0よりも大きな値である必要があります。

プログラミング例

```
%G100_Type_1

N10 G00 X0 Y0
N20 G100 X10 Y20 F200  ;X10/Y20 Target point of measuring traverse
...
```

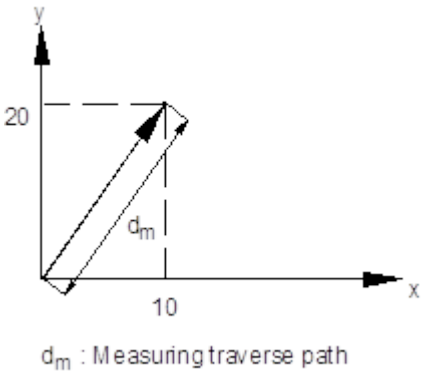


図 4: 図4-5: 計測機能のプログラミング

装置は、計測信号が確認されると停止します。計測ブロックの残りの動作経路は、出力されません。

プログラミング例

```
%Meas_traverse

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

図4-6は、実行結果の経路を示しています。

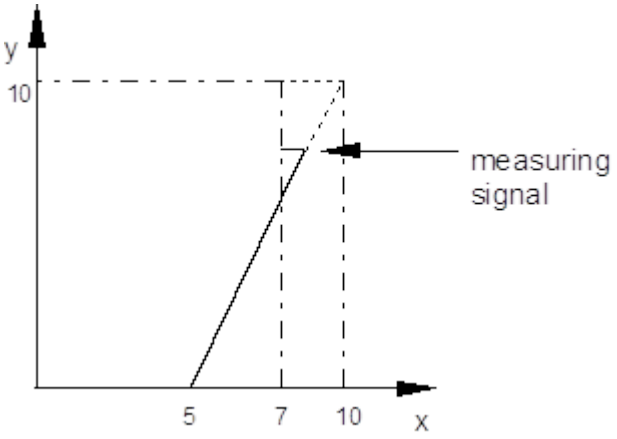


図 5: 図4-6: プログラムされた経路

5.1.10.2 単一軸での計測(G100) (タイプ2)

G100 <*axis_name*><*expr*>	(ノンモーダル)

G100 計測動作の指定

<axis_name><expr> 計測動作する軸の目標点

1つの軸でのみ、計測ブロックの移動動作が可能です。計測ブロックで指定された軸は、計測軸として設定(P-AXIS-00118)されていることが必要です。計測方法(タイプ2)が、P-CHAN-00057で指定されている必要があります。

計測ブロック内での計測中に、計測信号が発生すると記録がなされます。始点およびNCコマンドにて指定した目標点との間で、直線補間が実行されます(G01と同様です)。本マニュアルでは、計測ブロックでの経路速度を「計測する送り速度」と呼びます。軸1つのみが、計測動作に使用されている必要があります。P-AXIS-00215で計測される送り速度を指定します。計測ブロック内の動作距離は、0よりも大きな値である必要があります。

プログラミング例

```
%G100_Type_2

N10 G00 X0 Y0
N20 G100 X10    ;X10 Target point of measuring traverse
...
```

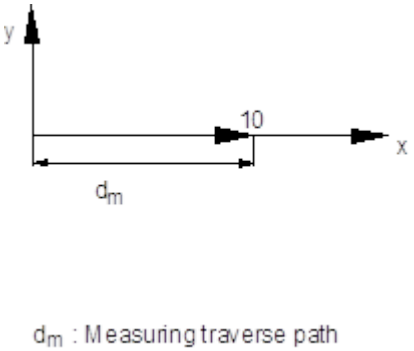


図 6: 図4-7: 計測機能のプログラミング

装置は、計測信号が確認されると停止します。計測ブロックの残りの動作経路は、出力されません。計測信号が計測ブロック内で検出できない場合には、エラーメッセージが出力されます。

プログラミング例

```
%Meas_traverse

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

図4-8は、実行結果の経路を示しています。

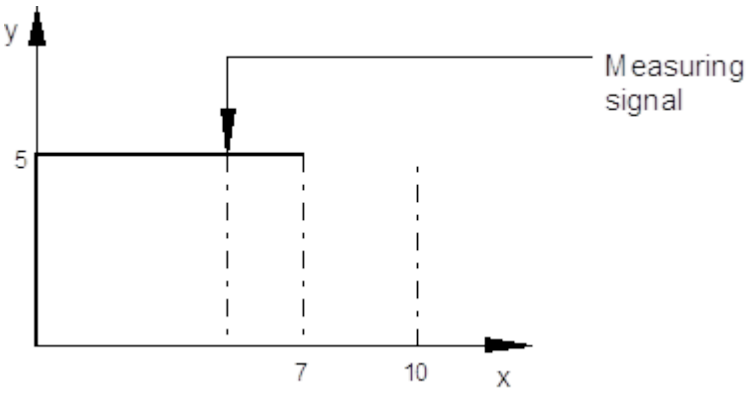


図 7: 図4-8: プログラムされた経路

5.1.10.3 目標点までの動作に対する計測(G100/G106) (タイプ3)

G100 <*axis_name*><*expr*> { <*axis_name*><*expr*> } [**G106**] [**F**<*expr*>] (ノンモーダル)

G100	計測動作の指定
<axis_name><expr>	計測動作する軸の目標点
G106	目標点まで動作させることの指定
F*<expr>*	計測する送り速度

あらゆる軸に対して計測動作が可能です。計測ブロックで指定された軸は、計測軸として設定(P-AXIS-00118)されていることが必要です。計測方法(タイプ3)が、P-CHAN-00057で指定されている必要があります。

計測ブロック内での計測中に、計測信号が発生すると記録がなされます。始点およびNCコマンドにて指定した目標点との間で、直線補間が実行されます(G01と同様です)。本マニュアルでは、計測ブロックでの経路速度を「計測する送り速度」と呼びます。1つ以上の軸が計測動作に使用されている必要があります。計測する速度は、Fワードによって指定します。計測ブロックでの動作距離は、0よりも大きな値である必要があります。

プログラミング例

```
%G100_Type_3

N10 G00 X0 Y0
N20 G100 X10 Y20 F200 G106  ;X10/Y20 Target point of measuring traverse
...
```

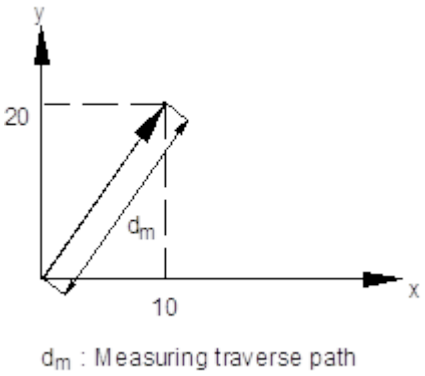


図 8: 図4-9: プログラムされた計測機能

G106がプログラムされている場合は、計測信号を検出した後、計測ブロックの目標点まで進みます。G106がプログラムされていない場合は、計測信号を検出した後に減速し、残りの動作経路は出力されません(計測タイプ1と同じ動作)。

プログラミング例

```
%Meas_traverse

N10 G00 X0 Y0 Z0
N20 G01 X5 F500
N30 G100 G106 X10 Y10  ;After measuring signal traverse to target point
N40 G01 X7
N50 M30
```

図4-10は、実行結果の経路を示しています。

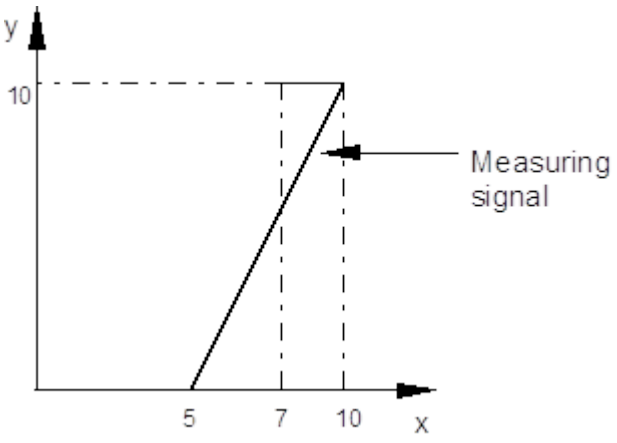


図 9: 図4-10: プログラムされた経路

5.1.10.4 メインとなる軸での計測(G100) (タイプ4)

G100 <*axis_name*><*expr*> { <*axis_name*><*expr*> } [F<*expr*>] (ノンンモーダル)

G100	計測動作の指定
<axis_name><expr>	計測動作する軸の目標点
F*<expr>*	計測する送り速度

メインとなる3つの軸に対して、計測ブロックでの移動動作が可能です。計測ブロックで指定された軸は、計測軸として設定(P-AXIS-00118)されていることが必要です。計測方法(タイプ4)が、P-CHAN-00057で指定されている必要があります。

計測ブロック内での計測中に、計測信号が発生すると記録がなされます。始点およびNCコマンドにて指定した目標点との間で、直線補間が実行されます(G01と同様です)。本マニュアルでは、計測ブロックでの経路速度を「計測する送り速度」と呼びます。最大で、メインとなる3つの軸で計測動作が可能です。計測する送り速度は、Fワードによって指定します。計測ブロックでの動作距離は、0よりも大きな値である必要があります。

プログラミング例

```
%G100_Type_4

N10 G00 X0 Y0
N20 G100 X10 Y20 F200  ;X10/Y20 Target point of measuring traverse
...
```

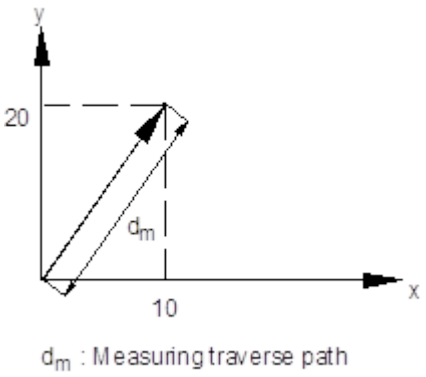


図 10: 図4-11: 計測機能のプログラミング

装置は計測信号が確認されると停止します。計測ブロックの残りの動作経路は出力されません。

プログラミング例

```
%Meas_traverse

N10 G00 X0 Y0 Z0
```

```
N20 X5
N30 G100 X10 Y10 F500
N40 G01 X7
N50 M30
```

図4-12は、実行結果の経路を示しています。

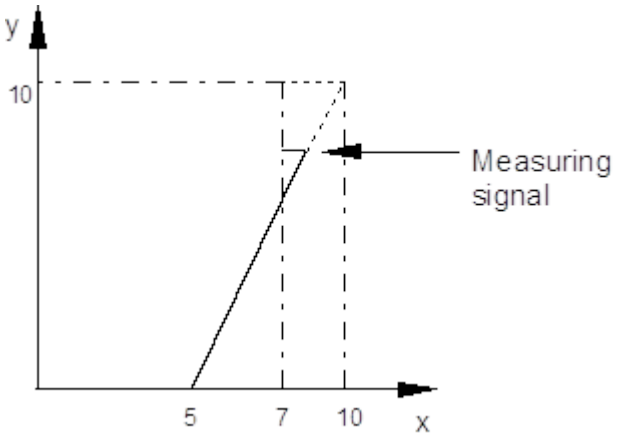


図 11: 図4-12: プログラムされた経路

5.1.10.5 中断分岐が発生する計測(G310) (タイプ5、6)

G310 [G00 | G01] <*axis_name*><*expr*> {<*axis_name*><*expr*>} [**$GOTO**<*Label*>] (ノンモーダル)

G310	中断分岐する計測動作を指定
G00 | G01	補間モードを指定
<*axis_name*><*expr*>	計測動作する軸の目標点
$GOTO<*Label*>	計測により中断された場合の、移動ジャンプ先のラベル

あらゆる軸に対して計測動作が可能です。計測ブロックで指定された軸は、計測軸として設定(P-AXIS-00118)されていることが必要です。計測方法(タイプ5、6)が、P-CHAN-00057で指定されている必要があります。

この計測方法では、計測信号によって計測動作を中断できます。動作内容は同一ブロック内で明示的にプログラムされている必要があります。計測信号によって動作が中断されると、G310ブロックで指定しているラベルに分岐して移動します。ブロック中で計測信号が発生しない場合は、NCプログラムは次のNCブロックを継続して実行します。

プログラミング例

```
N10 G00 X0 Y0
N20 G310 G01 F100 X100 Y200 $GOTO[N_LABEL] (If interrupt. jump to N_LABEL)
```

```
N30 G01 X200
N40 $GOTO[ENDE]
N50 [N_LABEL] X0 Y0
N60 [ENDE] M30
```

プローブ信号によって動作が中断されると、チャンネル内のすべての計測軸に対して、プログラムした目標点の座標値が現在の座標値に置換されます。次に、指定されたブロックにジャンプします。

信号が受信されない場合、プログラムした目標点まで移動します。その後は、ジャンプは発生せずに、次のブロックが実行されます。

ラベルがプログラムされていない場合は、次のブロックが実行されます。

動作が中断された際の現在の座標値は、V.A.MESS[..]変数によってNCプログラムから読み取ることができます。

TRC機能 (G41/G42)が有効中には、中断できません。エラーメッセージが出力されます。

5.1.10.6 軸停止の計測動作(G100) (タイプ7)

軸停止の発生を計測する動作中は、計測に関係するすべての動作軸に対してトルク制限が有効となっている必要があります。また、軸が動作する際に位置偏差をモニタリングする機能も解除されている必要もあります。

計測に関係する軸のいずれかがの軸停止が検出されると、計測が終了します。

プログラミング例、および軸停止の計測動作に必要な設定に関するより詳細な情報は、機能説明「計測 (C4)」([FCT-C4])に記載されています。

5.1.10.7 計測オフセットの計算(G101/G102)。

計測オフセットとは、記録された計測位置と目標位置との間のオフセット量です(図4-13を参照)。これは、以下のように計算されます。

計測オフセット = 計測位置 – 目標位置

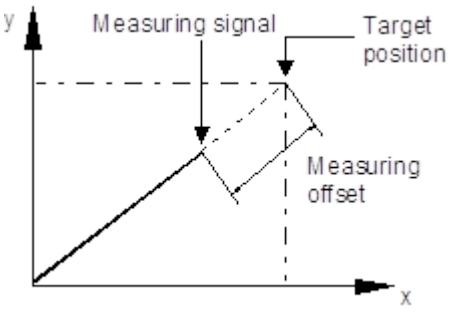


図 12: 図4-13: 計測オフセット

G101 <*axis_name*><*expr*> { <*axis_name*><*expr*> } (ノンモーダル)

G101 シフト量への計測オフセット値の組み入れ

<axis_name><expr> 軸に対して、計測オフセットの組み入れる際の係数

計測タイプ2を実行した場合であっても、事前に個別の計測動作が各軸に対して実行されている場合には、複数の軸に対して計測オフセットを指定することが可能です。

計測値から定められる計測オフセット値が、プログラムされた座標と絶対座標との間のシフト付加量として、プログラムされた座標に対して組み入れられます。計測値が事前に取得されていない場合は、エラーメッセージが出力されます。軸名の後の数値は、組み入れる際の係数を表します。

計測オフセットによるシフトは、G102によって解除されるまで有効です。

プログラミング例

プログラムされた座標値と絶対座標値とのシフト量として、X軸に対して係数1で、Y軸に対して係数7で計測オフセット値を組み入れます。

```
N10 G101 X1 Y7
```

G102 { *<axis_name><dummy_expr>* } (ノンモーダル)

G102 計測オフセット値のシフト量からの解除

<axis_name><dummy_expr> 計測オフセットを抽出する必要がある軸。

G100によって取得され、G101によってシフト量として組み入れられた計測値は、以下のルールに従って解除することができます。

1つ以上の軸に対してプログラムした場合、プログラムしたすべての軸の計測シフト量が解除されます。軸が何も指定していない場合、すべての軸の計測シフト量が解除されます。計測値が事前に組み入れられていない場合は、エラーメッセージが出力されます。

G101と異なり、軸名の後ろの値は意味がありませんが、構文上の理由から必要です。必ず、G101によって指定されているシフト量は解除されます。

プログラミング例

```
N10 G102 X1     (Extract measuring shift from X-axis    )
N20 G102        (Extract measuring shift from all axes  )
```

5.1.10.8 エッジバンディング(G108)

木材加工や家具製造においては、正確なベニヤ材を製造していくために、エッジバンディング機能が必要です。ベニヤ材を製造する際、ワークが異なるとエッジの開始位置が数ミリ単位で異なる場合があります。このため、NCブロックの指定位置とは異なる位置でベニヤ材を切断する必要があります。これはXY平面上で発生する2次元上の問題であり、直線移動と円弧移動のどちらも関係することがあります。

ベニヤ材のエッジの位置は、計測プローブによって先行して計測します(図4-14)。計測した値は、ベニヤ材を切断しなければならない正確なエッジの位置を決定するために使用されます。

以下の動作が行われる必要があります。

計測プローブにより測定されると、補間計算処理がなされ、その先の経路を定義して出力します。その後、装置はすべての軸が制御できるようになるまで待機します。次に、MNE_SNS P-CHAN-00027で設定した同期タイプのMファンクションが実行され、PLCに伝達されます。これによりベニヤ板が切断されていきます。応答の確認ができたら、プログラムした目標点まで補間計算処理が継続されます。実行できるMNE_SNS機能は1つのみです。つまり、1つのNCブロック内で複数のプログラムは実行できません。ただし、MNE_SNS機能は、他の同期タイプのMファンクションとは同時にプログラムすることができます。

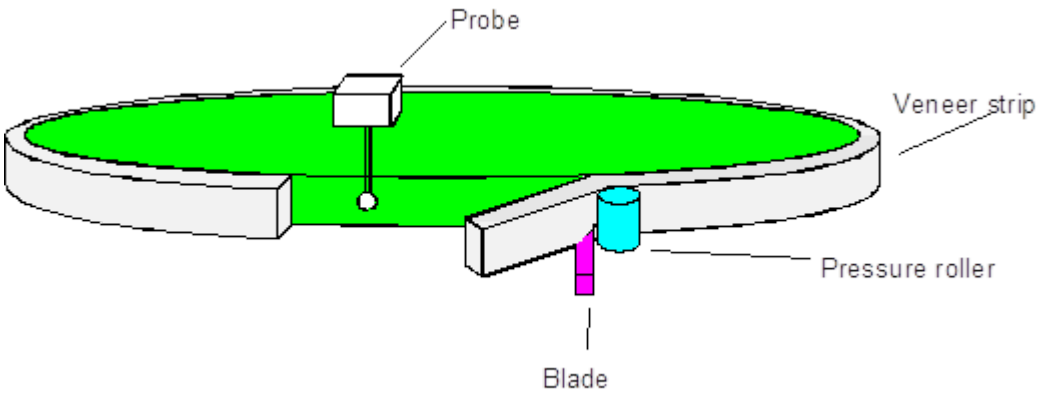


図 13: 図4-14: ベニヤ板の製造とエッジバンディング

G108	エッジバンディング	(モーダル)

エッジバンディングはモーダルファンクションG108によって有効になります。チャンネルパラメータリストP-CHAN-00029での設定により、次の2つの方法が可能です。

CNC軸1つだけを移動させる場合は、G108も主軸モーションなしで使用することができます。この特殊なケースでは、計測についても1つの軸のみで実行できます。

5.1.10.8.1 1つのブロックで動作するエッジバンディング(方法1)

G108による計測選択はモーダルであり、P-CHAN-00027にて設定されたMNE_SNSタイプのMファンクションが実行されるまで効果が継続します。Mファンクションにより計測が開始されます。このMファンクションによりプログラムされているため、モーションブロックの実行中に計測信号が発生します。

計測信号が検出された後、装置はP-CHAN-00030の設定に従い指定された残りの経路移動を続けます。G108は無効になります。

プログラミング例

```
N05 X0 Y0
N10 G108                    (Activate edge banding)
N20 G01 X90 Y90 F20
N30 M97 G01 X150 Y150 F8    (M97 is of MNE_SNS type, start measuring)
                            (Continue by residual path after measuring)
                            (signal)
M30
```

注記

計測中に計測信号が検出されなかった場合、ブロックの終端でエラーメッセージが発生します。

5.1.10.8.2 複数のブロックで動作するエッジバンディング(方法2)

CADシステムなどによって形状が作成された場合には、多数の短いNCブロックによってプログラムされることがあります。このような形状に対するエッジバンディングを実行する場合、NCブロックごとにベニヤ板の正確な切断位置を特定するのは困難です。

G108はエッジバンディングを有効化するとともに、モーダルとして動作します。チャンネルパラメータリストP-CHAN-00029によるパラメータ設定により、Mファンクション後の複数のブロックでベニヤ板の切断を実行できます。まず、Mファンクションが実行されます。その後、計測信号が与えられます。計測信号が検出された後、装置はP-CHAN-00030の設定に従い、複数のNCブロックの指定に沿って残りの経路移動を続けます。G107により計測の終了を明示的に示します。その時点までに計測信号が検出される必要があります。

G107	エッジバンディングの解除	(モーダル)

プログラミング例

```
N05 X0 Y0
N10 G108                    (Activate edge banding)
N20 G01 X90 Y90 F20
N30 M97 G01 X100 Y100 F8    (M97 is of MNE_SNS type, start measuring)
N40 X110 Y110
N50 X120 Y120
N60 X130 Y130
N70 X140 Y140
N80 X150 Y150               (<- last block of measuring!)
N90 G107                    (End of measuring for edge banding)
N80 G00 X200 Y200
M30
```

動作ブロックと同じブロックでの計測も可能です。ただし、このケースでは方式1でより簡単にプログラミングできます。

注記

計測中に計測信号が検出されなかった場合、G107でエラーメッセージが発生します。

5.1.10.8.3 残りの経路のプログラミング

V.G.RW	エッジバンディングでの残りの経路	(読み取り/書き込みアクセス)

軸グループに定義されている「V.G.RW」変数を使用して、エッジバンディングの残りの経路を再設定できます。残りの経路とは、計測信号が発生した後に残っている経路です。初期設定状態では、必要なデータがチャンネルパラメータリストP-CHAN-00030から取得され、残りの経路が設定されます。

プログラミング例

```
N010 G91 G00 X0 Y0 Z10         (Linear interpolation
N020 V.G.RW = 5D               (Definition of a new resid. path)
N030 G108                      (Activate edge banding)
N040 G01 X10 Y10 F300          (Linear interpolation)
N050 M97 G01 X30 Y10 F200      (M97 is of the MNE_SNS type,)
                               (Measurement with lin. interpol., continue)
                               (by residual path after  measuring signal)
```

注記

エッジバンディングのMファンクションは、MNE_SNS type P-CHAN-00027により定義されます。

5.2 送り速度の適用(G08/G09/G900/G901)

「送り速度の適用」機能では、線形的に速度が変化する動作の場合のみ有効となります。

G08	ブロックの先頭から加速	(モーダル、初期設定)
G09	ブロックの終端で減速	(ノンモーダル) (効果はG901/G900に依存)
G901、**G900**	ブロックの**終端後に**減速 G09により各ブロックごとにキャンセル可能 ブロックの**終端で**減速。G09には依存せずモーダルで動作。	(モーダル) (モーダル)

連続する2つのNCブロックが異なる送り速度でプログラムされている場合、ブロックの境界で「ソフトに」送り速度が変化します。G08を指定すると、ブロック先頭から必ず加速が行われます。G09を指定すると、現在のブロックの終端時点で、次のブロックの送り速度にまで減速しているように動作します。

G09はプログラム開始時の初期設定です。G901では、ブロック終了後の減速するように初期設定を変更できます。反対に、G900はそのスイッチを戻す動作となり、チャンネルを初期設定のモードに戻します。

注記

G00からG01、G02、またはG03への続く動作は、基本的にG09は有効となります。つまり、ブロック終了時点で次のブロックの速度まで減速するように動作します。

G901が有効になっていても経路を動作する速度が限界値に達してしまうと、ブロック終端時点ですでに次のブロックの速度となってしまうことがあります。この場合、G901は有効に動作しません。

Example:
N10 G01 X500 F400
N20 X900 F1000

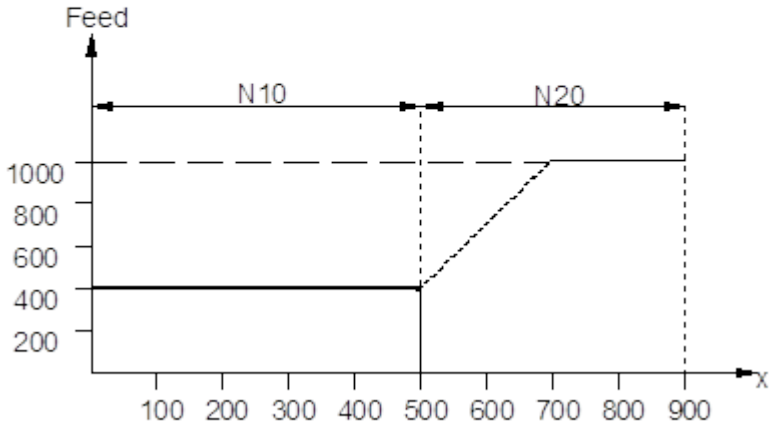
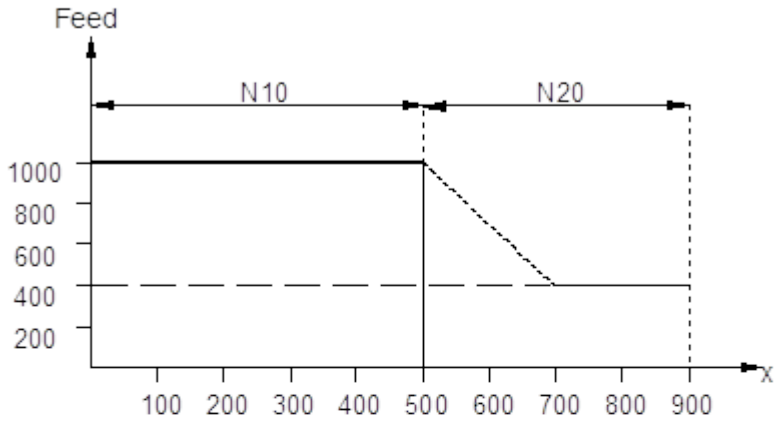


図 14: 図4-15: 初期設定におけるブロック遷移時の加速(G08の設定)

Example:
N10 G01 G901 X500 F1000
N20 X900 F400



Example:
N10 G01 G900 X500 F1000
N20 X900 F400

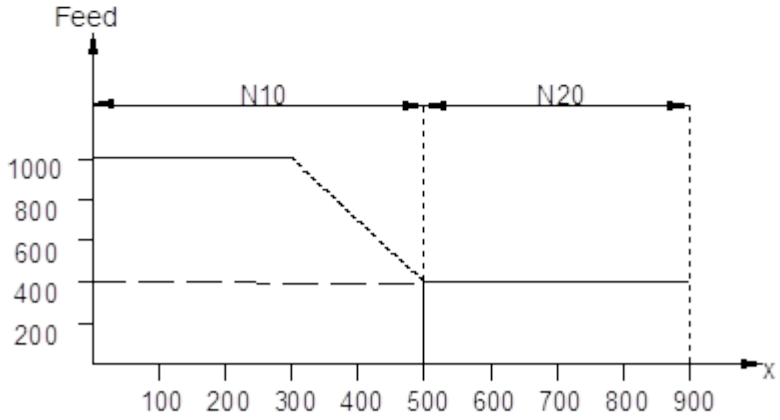


図 15: 図4-16: G901およびG900でのブロック遷移時の減速

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

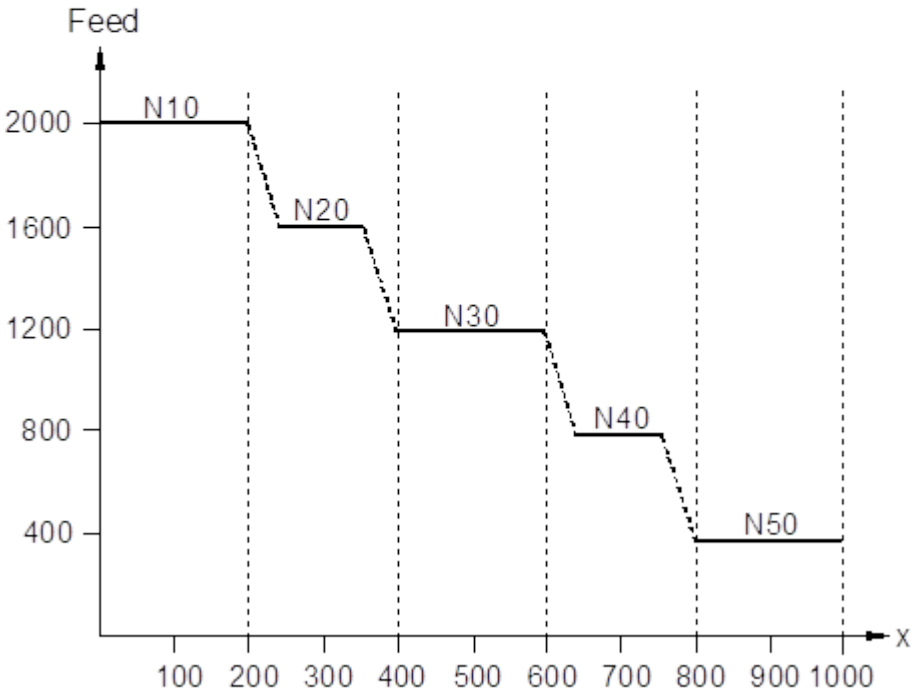


図 16: 図4-17: G09と、G901およびG900の組み合わせ

5.3 経路/時間で送り速度を補間する(G193/G293)

「経路で送り速度を補間する」機能は、速度が線形に変化する動作の場合、またはスロープタイプ3 (HSCスロープ)の場合にのみ有効です。

「時間で送り速度を補間する」機能は、速度が線形に変化する動作の場合のみに有効です。

G193	経路で送り速度を補間する	(ノンモーダル)
G293	時間で送り速度を補間する	(ノンモーダル)

G193/G293では最初の速度と、プログラムされた最終的な速度との間で、直線的に送り速度が補間されます。

プログラミング例

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

N10 / N20のブロック間では、経路で送り速度を補間する計算が実行され、N20でプログラムされた経路の全体をとおして、直線的に最終速度まで減速していく動作となります。

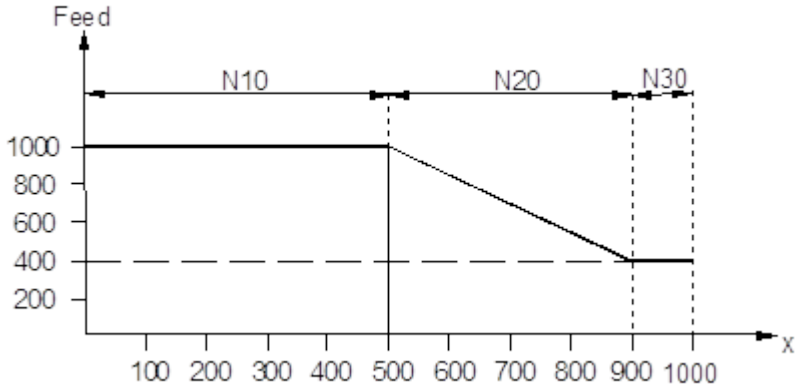


図 17: 図4-18: G193による経路で送り速度を補間する機能

軸の加減速設定値の許容範囲内であれば、オーバライドの変更も重ね合わせて設定することができます。

G193はG293、G08、またはG09と同一NCブロックではプログラミングできません。

プログラミング例

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

N10 / N20のブロック間では、時間で送り速度を補間する計算が実行され、ブロックの動作時間全体をとおして、直線的に最終速度まで減速していく動作となります。

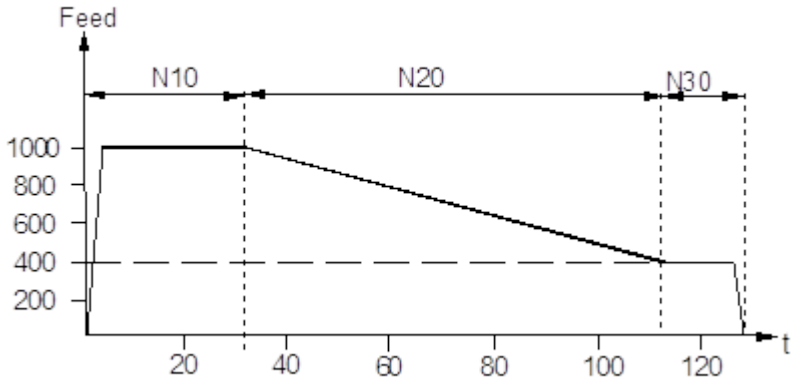


図 18: 図4-19: G293による時間で送り速度を補間する機能

G293はG193、G08、またはG09と同一NCブロックではプログラミングできません。

5.4 平面指定(G17/G18/G19)

G17	XY平面	(モーダル、初期設定)
G18	ZX平面	(モーダル)
G19	YZ平面	(モーダル)

G17、G18、またはG19をプログラムするミことで、工具径補正および円弧補間(「円弧補間(G02/G03) [▶ 45]」の章も参照)が有効となる平面が決定されます。

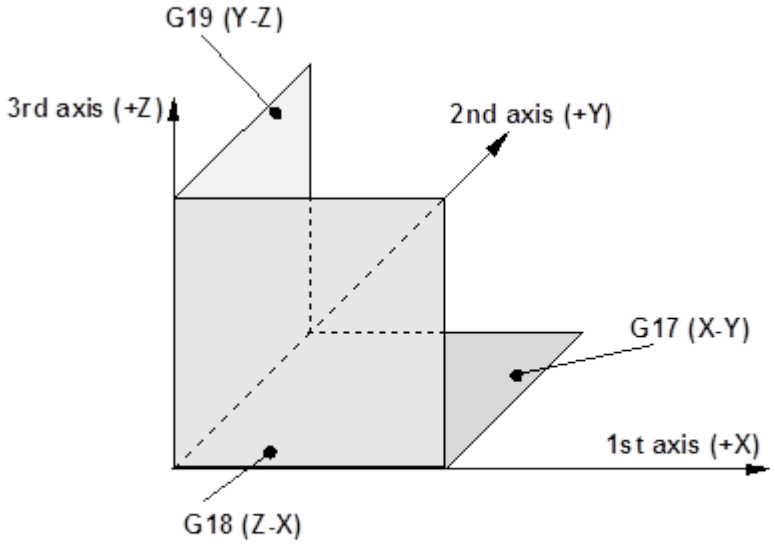


図 19: 図4-20: 平面指定の説明

工具径補正:

工具軸補正は、最初の2つの軸に対して有効になることに注意が必要です。

円弧補間:

G17、G18、またはG19により指定平面を変更すると、DIN規格に準拠した割り当てが再度有効になり、X、Y、ZにI、J、Kに割り当てられます。以下の表に、指定した平面に対応する構文を記載します。

平面指定	円弧補間タイプ	平面内の終点	円弧中心点/半径
G17	G02/G03	X..Y..	I..J../R
G18	G02/G03	Z..X..	K..I../R
G19	G02/G03	Y..Z..	J..K../R

5.5 ミラーリング(G21/G22/G23/G20)

ミラーリングの際には、「メンタル座標」が定義され導入されます。ミラーリングの前、あるいはミラーリングの解除後には、プログラム座標とメンタル座標は同じになります。ミラーリングを有効にする際には、プログラマはメンタル座標の種類を指定します。これによりプログラム座標はミラーリングされ、すなわち座標が移動します。

以下のミラーリングファンクションを使用できます(XY平面など)。

G21	Y軸を中央としたプログラムされた経路のミラーリング	(モーダル)
	$x_m = - x_p$	
	$y_m = y_p$	

G22	X軸を中央としたプログラムされた経路のミラーリング	(モーダル)
	$x_m = x_p$	
	$y_m = - y_p$	

G23	G21およびG22の重ね合わせ	(モーダル)
	$x_m = - x_p$	
	$y_m = - y_p$	

G20	ミラーリングの解除	(モーダル)

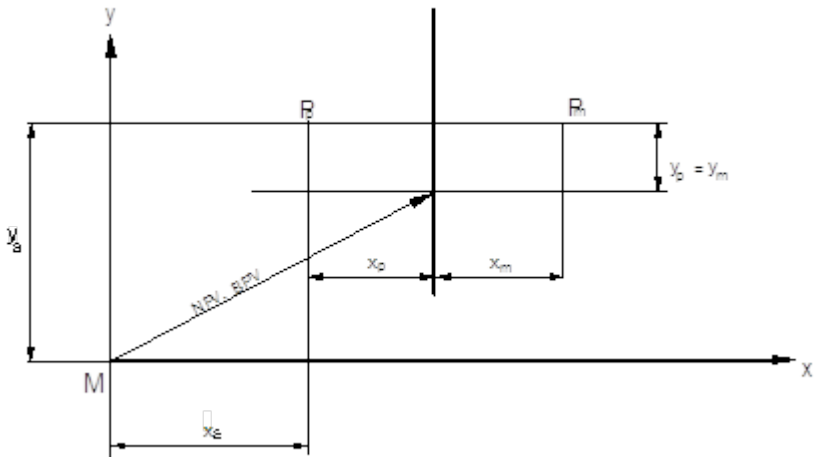


図 20: 図4-21: メンタル座標とプログラム座標のG21での関係性

x_m、y_m メンタル座標
x_p, y_p プログラム座標
x_a, y_a 絶対座標

ミラーリング形状は、サブプログラムとして指定してください。このサブプログラムは、ミラーリングファンクションを実行後に呼び出してください。ミラーリングはモーダルです。

工具径補正中にミラーリングがプログラミングされた場合、補正方向が変化します。つまり、G41が有効な場合、ミラーリング後には補正量と等距離分、形状の右側へプログラム経路が計算されます(G42: 形状の右側への工具径補正(TRC))。これは、進行方向自体が変化した場合でも同様です。

プログラミング例

```
%L DREIECK                  (Subroutine)
N10  G90  X10  Y20          (Determination of the (mental)coordinates)
N20  G91  X10  Y-10         (to be mirrored)
N30       X-10 Y-10
N40            Y20
N50  M29

%SPIEGELUNG                 (Main program)
N10  G92  G90  X60  Y40     (Coordinate preset)
N20  G01  F500              (Linear interpolation with feed 500)
N30  G21                    (Mirroring on the Y-axis)
N40  LL DREIECK             (Deselection of mirroring)
N60  G92  G90  X0  Y0       (Withdrawal of coordinate preset)
N70  M30
```

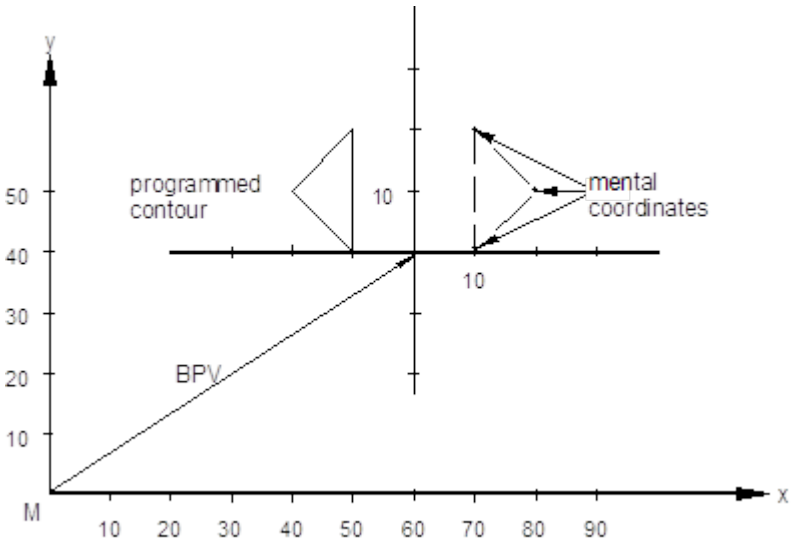


図 21: 図4-22: ミラーリングの例

注記

円弧のミラーリング

通常はプログラムされた座標のみがミラーリングされます (つまり全円の場合には、中心点座標のみがミラーリングします)。ミラーリング後には、円弧の終点がミラーリングされて移動します。円弧の起点はミラーリングされません。
結果: 動作ブロック内では、終点のみミラーリングされ、軌跡は完全にはミラーリングされません(図4-23)。
効果: 動作ブロックが一周円であり、起点/終点が0/0ではない場合、中心点座標がミラーリングされることにより、新しい円弧が作成されて形状が変化します(図4-24)!

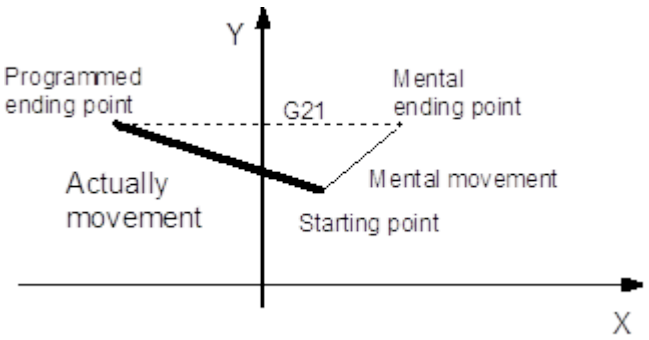


図 22: 図4-23: 動作ブロックにおける終点のミラーリング

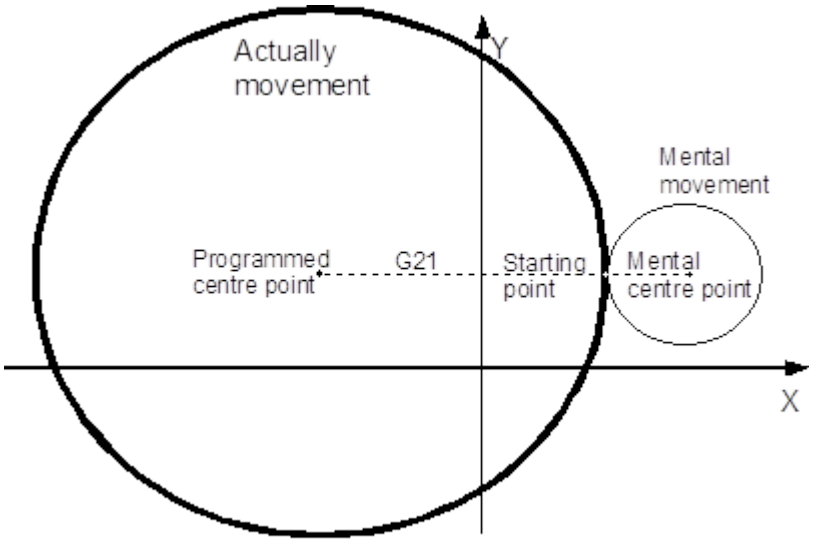


図 23: 図4-24: 一周円のミラーリング時の形状変化

3次元での検討:

XY平面(G18、G19)以外でミラーリングファンクションが選択される場合、選択した平面によっては円弧補間の回転方向が変わる可能性があることに注意が必要です。また、ミラーリングはZ座標には影響を与えないため、メンタル座標とプログラム座標のZ軸座標上の値は常に同一になります。

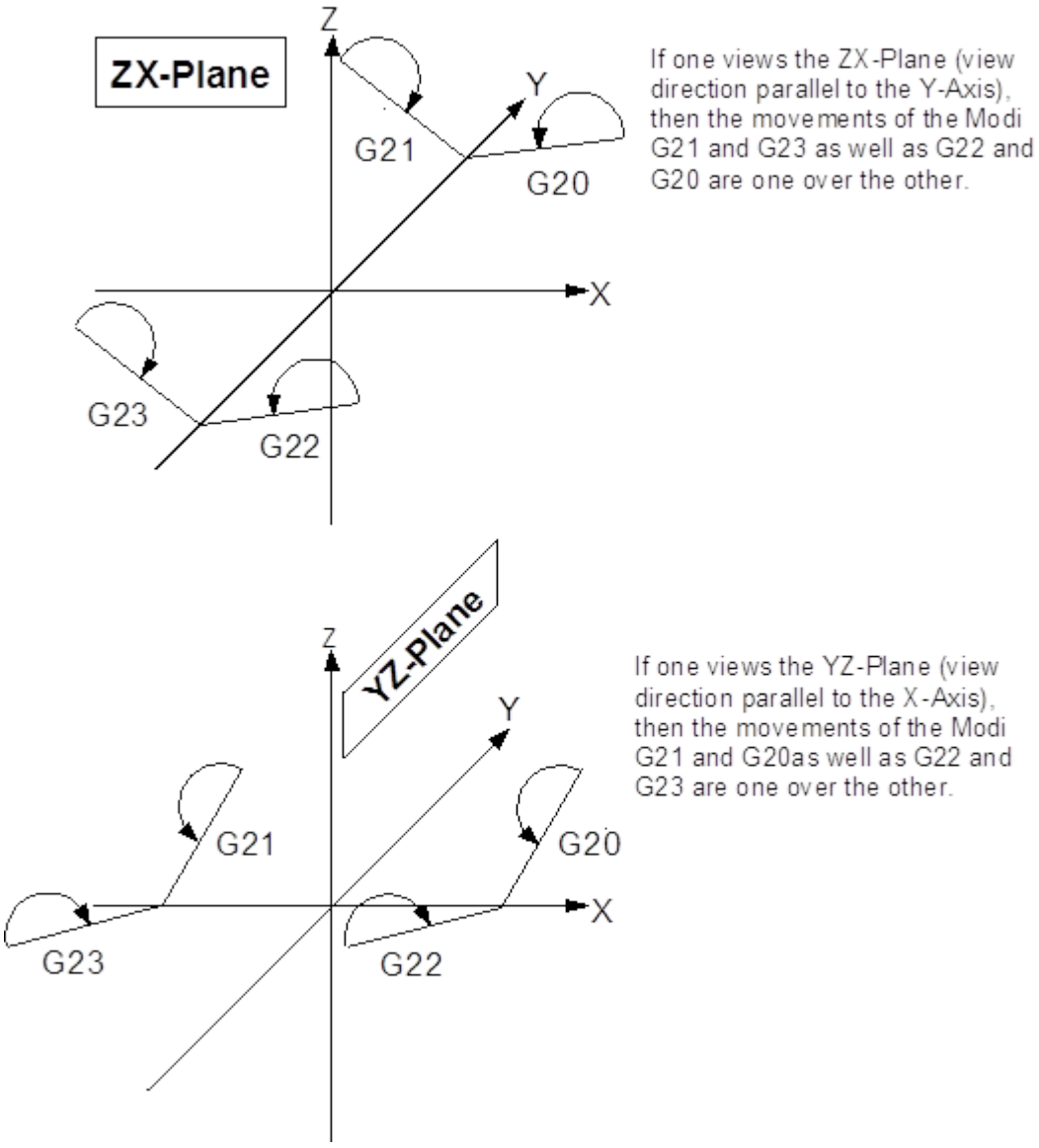


図 24: 図4-25: XY平面とは異なる平面上における円弧回転方向へのミラーリングファンクションの効果

5.6 軸情報のミラーリング(G351)

GファンクションG20～G23は、DIN 66025に準拠した形でミラーリングを使用することができます。さらに、以下の拡張機能により、ミラーリングを行う軸を明示してプログラムすることができます。

G351 <*axis_name*> [[**+**]	**-**] **1** { <*axis_name*> [[**+**]	**-**] **1** } (ノンモーダル)

G351 ミラーリングの軸ごとの選択。GファンクションG351はノンモーダルですが、このファンクションを使用してプログラムした軸のミラーリングはモーダルです。

<axis_name> 座標はアドレス文字と、それに続く座標値計算の数式から成ります。

座標値 -1: ミラーリングを選択

座標値1または+1: ミラーリングをキャンセル

プログラミング例

```
G351 X-1 Y1 Z+1     (Selects mirroring of the X axis and cancels mirroring)
                    (of the Y and Z axes)
```

- ミラーリングされる軸は、NCブロック内の**どの個所にでも**プログラム可能です。
- 1つ以上の軸をG351と共にプログラムする必要があります。
- ファンクションG351は、NCブロック内でG351のみでプログラムされる必要があります。ブロック番号Nはこの規則の例外です。
- 軸のミラーリングを繰り返し選択、またはキャンセルすることができます。ただし、同一NCブロック内で繰り返しプログラムされると、エラーメッセージが出力されます。
- 同期操作で先行軸(マスタ軸)に対してミラーリングが選択されても、ミラーリングは従軸(スレーブ軸)に対しては**自動的に選択されません**。ただし、スレーブ軸は常にマスタ軸の移動動作に従います。このため、マスタ軸のミラーリングの結果としての移動動作が、常にスレーブ軸に影響を与えます。
- プログラムの開始およびリセット時に、すべての軸のミラーリングがキャンセルされます。軸の交換時、変更された軸のミラーリングがキャンセルされます。
- 1番目または2番目に指定したメイン軸のミラーリングは、円弧補間または工具径補正における移動方向に影響を与えます。
- 工具径補正中にミラーリングをプログラムすると、G41/G42の選択がデコーダによって自動的に交換されます。これは直線ブロックでのみ可能です。この動作はコントローラーの仕様として定められています。

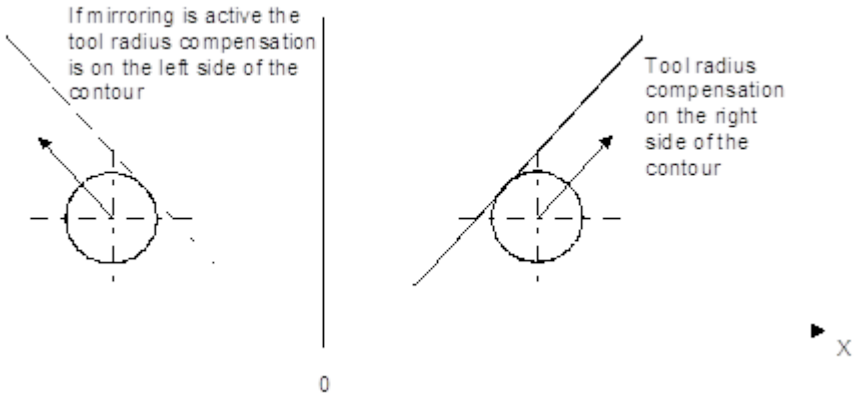


図 25: 図4-26: ミラーリング中の工具径補正で選択される補正方向

- ミラーリングされる軸に対して基準点オフセットがプログラムされている場合、基準点オフセットの座標もミラーリングされます。軸のミラーリングが選択されると、基準点オフセットもミラーリングされます。

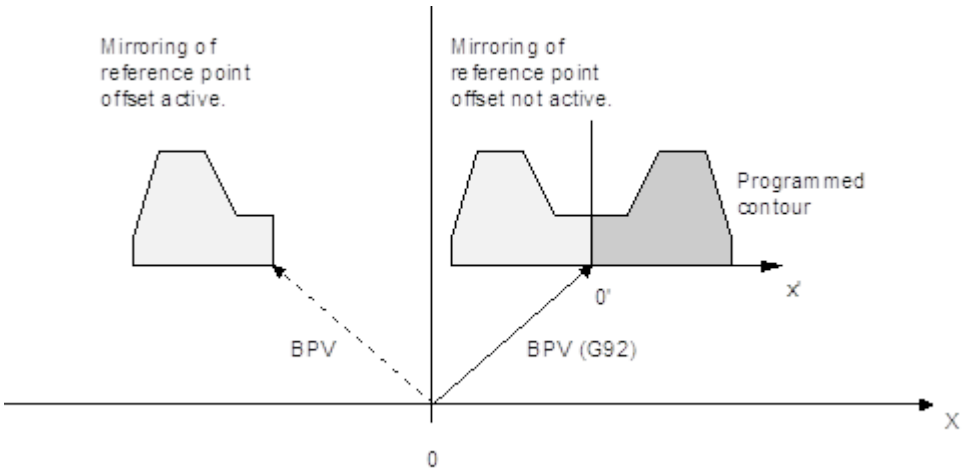


図 26: 図4-27: 基準点オフセットのミラーリング

- 円弧の中心点I、J、Kの座標もミラーリングされます(第4.5章)。
- 面取りおよび丸み付け(G301/G302)を挿入する場合は、I文字列を使用して面取りや丸み付けの値を入力します。このため、ここではミラーリングを考慮する必要はありません。
- G21～G23は軸名に関わらず、最初の2つのメイン軸をミラーリング対象として選択します。
- G20は最初の2つのメイン軸のミラーリングのキャンセルのみを行います。

以下は、G351ファンクションの使用方法を示しています。ここでは軸X、Y、およびZが、それぞれ1番目、2番目、3番目の主軸であると仮定しています。

プログラミング例

```
N10 G351 X-1            (Selects mirroring of the X axis (G21))

N20 G351 Y-1            (Selects mirroring of the Y axis (G22))

N30 G351 X-1 Y-1        (Selects mirroring of the X and Y axes (G23))

N40 G351 X1 Y+1         (Cancels mirroring of the X and Y axes (G20))

N50 X1 G351 Y-1 Z1      (Selects mirroring of the Y axis and cancels)
                        (mirroring of the X and Z axes)
```

5.7 単位(G70/G71)

G70	インチでのデータ入力	(モーダル)
G71	メートルでのデータ入力	(モーダル、デフォルト)

G70またはG71は、すべての経路および座標値に対して動作します。

例外: 工具パラメータ(V.G.WZxx.Pxx.)およびキネマティックパラメータ(V.G.WZxx.KIN_PARAMxx)は、G70/G71で定められた現在の単位とは関係なく、直接定められた単位で常に動作します。

5.8 暗黙的なサブプログラム呼び出し(G80～G89/G800～G819)

G80～G89 [[*<val1>*,*<val2>*, ～ ,*<val50>*]]	サブプログラム呼び出し	(ノンモーダル)
または、		
G800～G819 [[*<val1>*,*<val2>*, ～ ,*<val50>*]]	サブプログラム呼び出し	(ノンモーダル)

G8x / G8xx G80～G89およびG800～G819をプログラミングすることにより、割り当てられたグローバルサブプログラムが暗黙的に実行されます。これらのサブプログラムのデフォルト設定は、チャンネルパラメータP-CHAN-00160～P-CHAN-00169、またはP-CHAN-00187、またはNCプログラム実行中にコマンド#FILE NAME [▶ 328]によって定義することが可能です。

G80～G89、およびG800～G819がプログラムで指令された際に、何も割り当てがされていない場合には、エラーメッセージP-ERR-20131 「Unknown G function」が生成されます。

グローバルサブプログラムは一度だけ呼び出されます。つまり、G80～G89、およびG800～G819にはモーダル効果はありません。

<val1>, ～ ,*<val50>* サブプログラム(サイクル)の後ろにオプションで、定められた順番で最大50のパラメータ(REAL型の数式)を括弧で囲んで付加することが可能です。パラメータはカンマで区切ります。パラメータの空白は、連続するカンマ「, ,」で表す必要があります。

サブプログラム内でのノーマルサイクルプログラミングと同様、パラメータは@Pxアクセスによって読み取れます。呼び出しパラメータと@Pxでの読み取りアクセスは、固定で割り当てられます。 (例えば、@P1はパラメータ値1を読み取り、@P2はパラメータ値2を読み取ります)。これらのサブプログラムでは、文字「@」が利用した追加の拡張サイクル構文を使用できます。パラメータがプログラムされた(有効)かどうかは、サブプログラム(サイクル)内で変数 V.G.@P[i].VALID [▶ 425]によってチェックできます。

G80～G89、およびG800～G819は、ブロック内での最後の動作として実行されます。つまり、同一NCブロック内でプログラムされた軸動作は、必ずグローバルサブプログラム呼び出しの前に実行されます。

プログラミング例

```
For G80 the global sub progr. - g80_cycle.nc - shall be called:
..
N10 #FILE NAME[ G80="g80_cycle.nc" ]
Nx ..
N30 G80               Call of g80_cycle.nc as global subprogram
:
M1000 M30

For G815 the global sub progr. - g815_cycle.nc - shall be called:
```

```
..
N10 #FILE NAME[ G815="g815_cycle.nc" ]
Nx ..
N30 G815              Call of g815_cycle.nc as global subprogram
:
G85 calls the global sub progr. - cycle_tst.nc - with parameters:
N10 #FILE NAME[ G85="cycle_tst.nc" ]
Nx ..
N30 G85 [10,2, ,15,-3, ,5]  Call of cycle_tst.nc as global subprogram
:
```

5.9 アブソリュート指令、またはインクリメンタル指令(G90/G91)

G90	アブソリュート指令	(モーダル、デフォルト)
G91	インクリメンタル指令	(モーダル)

アブソリュート指令入力(G90)の場合、座標値は座標軸の原点が基準となります。つまり、動作ブロック内の指令数値として、座標系上の座標値を指定します。

インクリメンタルプログラミングの場合は、座標値は前の動作ブロックでの到達位置が基準となります。つまり、動作ブロック内での指令値として、移動する距離を指定します。

5.9.1 排他的プログラミング

初期設定では、NCブロック内ではアブソリュート指令とインクリメンタル指定の**どちらか一方**のみを選択できます。同一NCブロック内で重複してプログラミングしたり、G90とG91両方をプログラミングすることはできません。このため、NCブロック内でのG90/G91の位置は意味を持ちません。

プログラミング例

```
:
N10 X10 Y10 (Absolute dimensioning G90 is selected, Default)
N20 G91 X20 Y20 (Deselection of absolute- and selection of relative)
                (programming)
N30 X30 G90 Y30 (Deselection of relative- and selection of absolute)
                (programmig)
:
N100 G90 Z30 G90     (Error message: Redundant programming of G90/G91)
N110 G91 X10 G90 Z30 (Error message: Redundant programming of G90/G91)
```

5.9.2 複合プログラミング

チャンネルパラメータP-CHAN-00116を使用すると、**補間軸**への排他的なプログラミング指令を無効にできます。これにより、同一NCブロック内でアブソリュート指令とインクリメンタル指令の**両方を**プログラミングできます。さらに、同一NCブロック内でG90とG91を繰り返し使用することも可能になります。この場合、NCブロック内のG90/G91の位置は意味を持ちます。直近で指令されたG90/G91がそれ以降のすべての軸指令に対して有効になり、それは次のG90/G91まで続きます。

プログラミング例

```
:
N10 X10 Y10 (Absolute dimensioning G90 is selected, Default)
N20 G91 X20 G90 Y20 (Relative for X axis, absolute for Y axis)
N30 X30 G91 Y30 Z20 (Absolute for X axis, relative for Y/Z axes)
N30 G90 X30 G91 Y30 G90 Z20 (Absolute for X/Z axes, relative for Y axis)
:
N100 G90 G91 Z30 (Relative for Z axis)
N110 G91 X10 G90 Z30 (Relative for X axis, absolute for Z axis)
```

G90/G91は、円弧補間のパラメータI、J、K、またはヘリカル補間には影響を与えません。また、G90/G91は、G161/G162指令の際には排他的に動作します。

5.10 イグザクトストップ(G60/G360/G359)

G60	イグザクトストップ	(ノンモーダル)
複数のブロックに対するイグザクトストップ:		
G360	イグザクトストップの選択	(モーダル)
G359	イグザクトストップの解除	(モーダル)

G60/G360により、限界まで精度良く、正確に目標位置に到達できるようになります。送り速度はブロック終了までにゼロになるまで減速され、ポジション偏差エラーもなくなります。

精度良く停止させることで、エッジを正確に加工する場合や、方向反転時に目標位置に正確に到達する必要がある場合に使用することができます。指定された制御範囲(P-AXIS-00236)にポジションが到達すると、次の目標位置への移動へ進みます。

5.11 多項式輪郭加工(G61/G261/G260)

G61	多項式輪郭加工(ブロック終了時)	(ノンモーダル)
複数のブロックに対する多項式輪郭加工:		
G261	多項式輪郭加工の選択(ブロック終了時)	(モーダル)
G260	多項式輪郭加工の解除	(モーダル)

「多項式輪郭加工」では、2つの移動ブロックの曲率および方向が連続します。このために、プログラムされた移動ブロックは短縮されます。ブロック間に形状曲線が追加されます。この処理で、直線と直線、直線と円弧、円弧と円弧間の輪郭加工が可能になります(図4-28)。これによって特定の平面だけでなく、3次元空間の任意の場所にある曲線間の輪郭加工ができます。

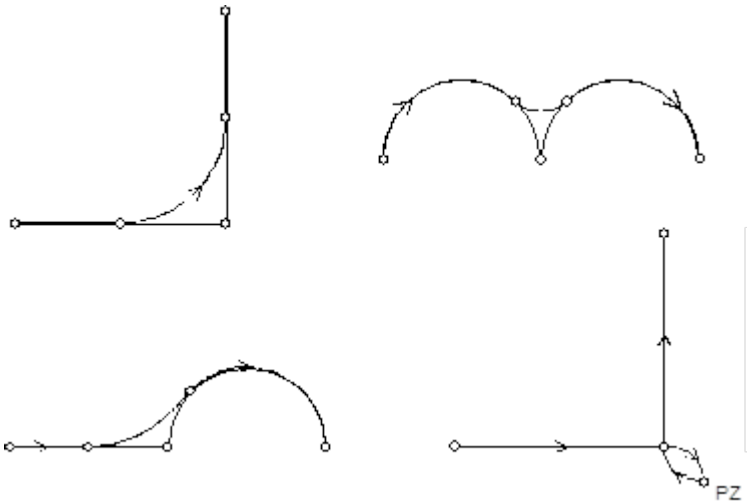


図 27: 図4-28: 多項式輪郭加工の例

輪郭加工には、以下のタイプが用意されています。

- コーナーの最大偏差を使用した自動輪郭加工
- 速度を使用した自動輪郭加工
- コーナー距離輪郭加工
- コーナー偏差輪郭加工

- 暫定点輪郭加工

輪郭加工のタイプに応じて、形状曲線を特徴づける異なるパラメータを提供できます。輪郭加工処理が完全に実行されるまで、パラメータは有効な状態が継続します。プレブロックとポストブロック間の形状パラメータが変更されると、次の輪郭加工処理のみ影響を受けます。

5.11.1 用語定義

本プログラミングマニュアルでは、以下の用語を使用します。

多項式輪郭加工:	曲率および方向が連続する2つの移動ブロックの結合。
形状曲線:	軸に対して4番目の2つの多項式からなる曲線。
ブロック長:	移動ブロックに対応する曲線のブロック長。
コーナー距離:	形状曲線の開始/終了から移動ブロックのプログラムした目標点/起点までの距離(以下の図を参照)。コーナー距離は、必ずブロック長の半分までに制限されます。円弧ブロックでは、コーナー距離は形状曲線の起点から円弧のプログラムした終点までの弧の距離です。

図 28: 図4-29: コーナー距離の定義

プレブロック:	形状曲線の前の移動ブロック。
ポストブロック:	形状曲線の後の移動ブロック。
プレ距離:	プリブロックのコーナー距離。
暫定点:	形状曲線の2つの部分曲線の教会が接する点。
コーナー偏差:	プログラムしたコーナー点と形状曲線の暫定点間の距離(以下の図を参照)。

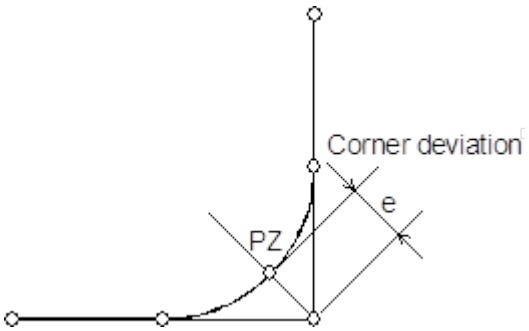


図 29: 図4-30: コーナー偏差の定義

プログラミング例

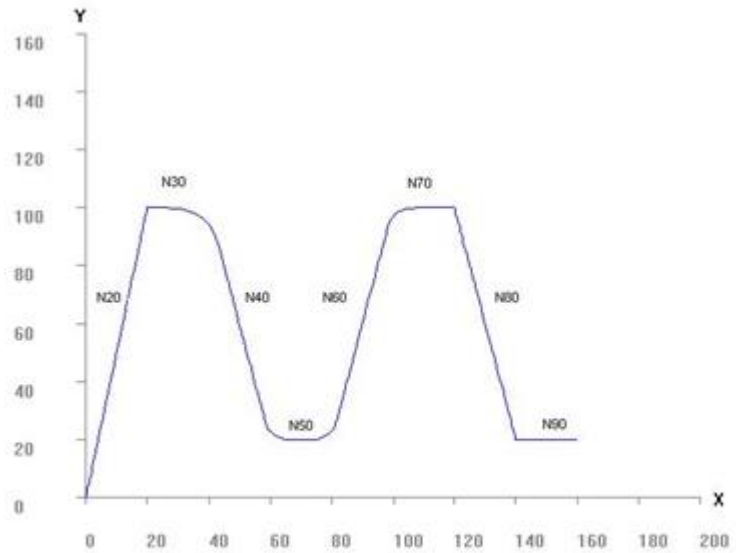
G61とG261/G260のプログラミングの比較:

この3つのNCプログラムは、すべて以下の図で示される同一の形状を作成します。

```
%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
N90       X160 Y20
N100 M30
```



5.11.2 一般的なプロパティ

形状的な多項式輪郭加工の処理は、3つのすべての次元での主軸の経路形状に基づいて行われます。コーナー距離は、コーナー偏差や経路速度の割合のようなあらかじめ定義された設定で決まります。これは、元の経路上に存在する多項式の起点および目標点が事前に分かっていることを意味します。

計算された主軸のコーナー距離から、従軸のコーナー距離も決定します。主軸と同様に、従軸でもプレブロックとポストブロックのコーナー距離間で軸の最大加速を考慮し、曲率と方向が連続する多項式が挿入されます。

コーナー偏差に対して指定可能な制限は、主軸しか考慮しません。このため、従軸の偏差に対して個別に制限を指定することができます。従軸がこの偏差を理論的に超過する場合、形状曲線が減衰します(コーナー距離の減速)。

以下の場合、多項式輪郭加工が自動的に抑制されます。

- すべての軸の移動が正接しているか、または直接ミラーリングされている。
- 主軸の移動のみが正接していて、従軸の最大偏差が指定されていない(値 = 0)。
- G61後に、ポストブロックのないプログラム終端に達した。さらに、警告メッセージが発生した。

5.11.2.1 最大コーナー偏差および残りの最小ブロック長

多項式曲線の「変形」を避けるために、以下の制限も有効です。

- コーナー距離は、元のブロック長の50%を超えてはなりません。50%を超えると、プレブロックとポストブロックのコーナー距離がそれぞれ制限されます。ブロックの開始およびブロックの終端でのコーナー距離が元のブロック長の50%であると、このブロックは完全にスキップされます。
- 輪郭加工のパラメータ設定中に、最小の残りのブロック長を0%から100%の範囲で設定できます。これは、50%から0%の範囲で変化する最大コーナー距離に対応します。プログラムの開始時は、初期設定の最小の残りブロック長は0% (ブロックの完全な輪郭加工)です。例えば、最小の残りのブロック長を10%とすると、このブロックのコーナー距離は最大で元のブロック長の(100% - 10%) / 2 = 45%となります。
- 円弧ブロックでは、最大コーナー距離は円弧角90°に制限されます。

5.11.2.2 関連するブロック長

コントロールの数学的分解能(REAL値の有効桁数およびプロシージャによって取得)により、最小曲線長は制限されます。初期設定では、この値は31.7 µmです。経路セグメントがこの最小曲線長よりも小さいと工ラーメッセージが出力され、輪郭加工が中止されます。

また、形状にはプログラミングシステム(CAD/CAM)や工具径補正によって生成される非常に短い補正ブロックが含まれることがあります。補正後には、このブロックは連続する経路の形状を確実に描画する必要があります。

この短いブロックで輪郭加工が中止されないように、多項式輪郭加工が次のブロックに関連する最小ブロック長を定義することが可能です。有効な輪郭加工中はこれよりも短いブロックは無視され、輪郭加工はそれ以降のブロックを考慮します。

主軸の移動距離の制限値や、従軸の移動距離の制限値を定義できます。主軸の移動距離と従軸の移動距離の両方が、定義された制限値を下回るとブロックが完全にスキップされます。多項式輪郭加工は、前後のブロック曲線と方向を結合します。基本ブロックは、互いに接してはなりません(形状が連続してはなりません)。

ブロックがスキップされる場合、初期設定では主軸と従軸の最大コーナー偏差は正確には考慮されません。つまり、輪郭加工の偏差に関連するスキップブロックは無視できると仮定されています。

プログラミング例

```
#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
```

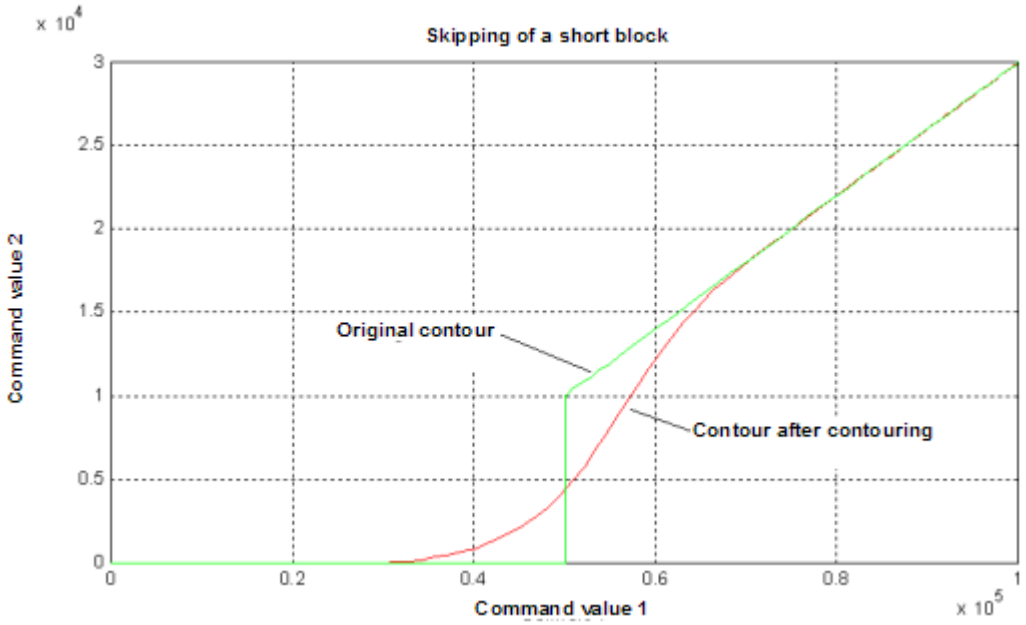


図 30: 図4-31: 輪郭加工中の短いブロックN05のスキップの例

特殊なケース1: ブロック遷移後に連続する複数の短いブロック

複数の連続するブロック(N20、N30、N40)が最小移動距離よりも短い場合、目標位置から最後の関連する終了位置(N10)までの距離が定義された最小移動距離よりも短いと、これらのブロックはスキップされます。スキップブロックの目標位置が開口部のない球体の外にある場合、定義された最小長よりも短くてもこのブロック(N40)が形状曲線の計算に使用されます。この方法により、複数の連続するブロックがスキップされる場合でも、元の形状の小さな偏差を表すことができます。

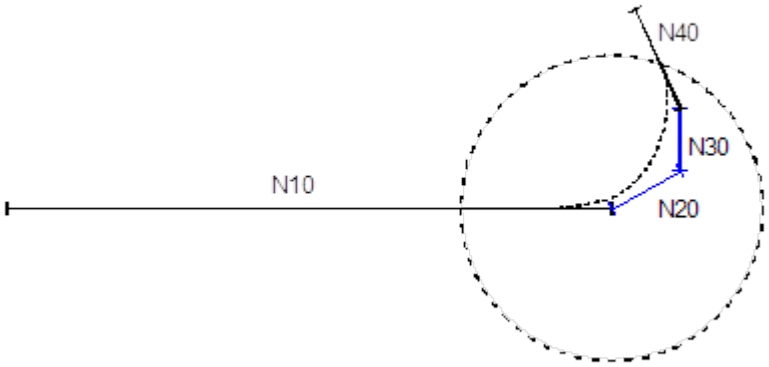


図 31: 図4-32: 複数の単一ブロック(N20、N30、およびN40)は短すぎますが、目標位置は最小ブロック長の範囲外にあります。

特殊なケース2: ブロック遷移後に連続する複数の短いブロック(最後のブロックが極端に短い)

特殊なケースとして、ブロックN40自体を輪郭加工に必要な、システム固有の最小長(約15 µm)より短くできます。このケースでは、最後の終了位置と新しい目標位置が直線ブロックによって接続されます。その後、この新しい直線ブロックN20は形状曲線の計算に使用されます。

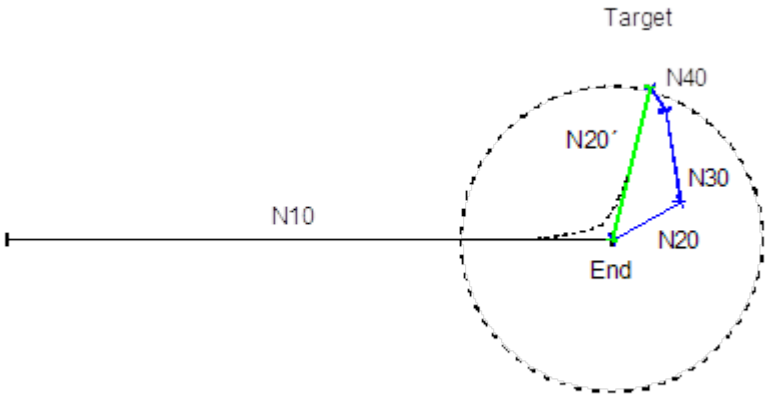


図 32: 図4-33: 複数の単一の短いブロック(N20、N30、およびN40)は短すぎますが、すべてのブロックの合計が最小ブロック長を超過しています。

特殊なケース3: ブロック遷移前の短いブロック

輪郭加工の開始時(ブロック遷移前)に、ブロックがシステムによって指定された最小長よりも既に短い場合、これらのブロックはスキップされます。これらのブロックは、最後の有効な位置と現在の目標位置間の距離が最小ブロック長を超過しているとスキップされます。このケースでは、最後の終了位置と現在の目標位置が直線ブロックN10によって接続されます。その後、この直線ブロックは輪郭加工の開始ブロックとして使用されます。

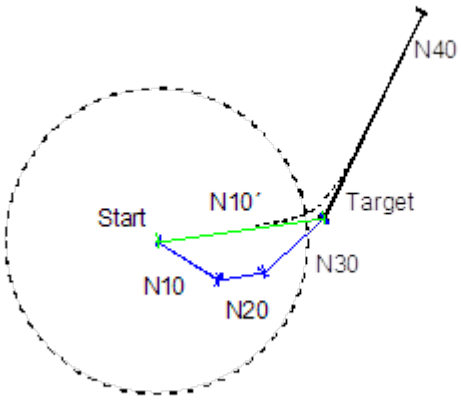


図 33: 図4-34: 複数のブロック(N10、N20、およびN30)は短すぎますが、すべてのブロックの合計がシステムによって指定された最小ブロック長を超過しています。

特殊なケース4: 輪郭加工の解除、またはパラメータ設定の変更

ブロックのスキップ中に輪郭加工が解除されたり、輪郭加工の基本条件が変更されたりした場合、現在の輪郭加工は解除、またはパラメータ変更までしか継続できません。その後、輪郭加工は新しいパラメータで継続できます。

```
#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…
```

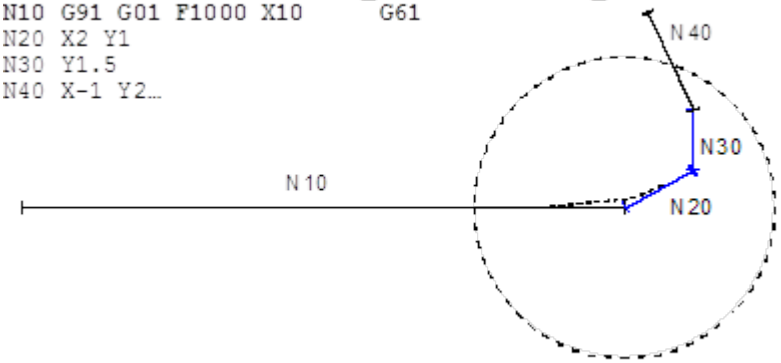


図 34: 図4-35: 複数の単一ブロック(N20、N30、およびN40)は短すぎますが、輪郭加工はブロックN20から解除されます。

5.11.2.3 追加ブロックの実行

ブロック終了(N10～N20)時の移動ブロックに加えて、形状情報を持たないコマンド(MVS_SNS同期するMファンクションなど)がプログラムされる場合、このコマンドを形状曲線の前、形状曲線の途中、または形状曲線の後に実行できます。

例:

```
N10       X100 G61 M25
N20       Y100
```

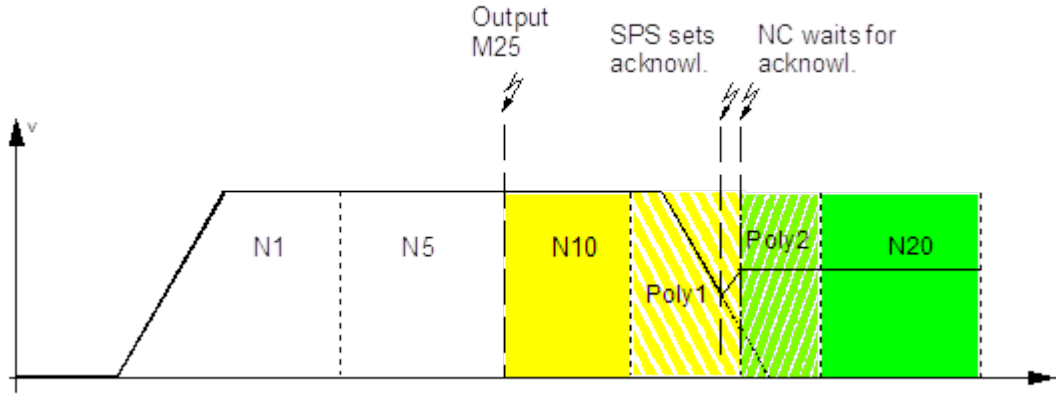


図 35: 図4-36: 輪郭加工中の形状に関連する動作のない同期

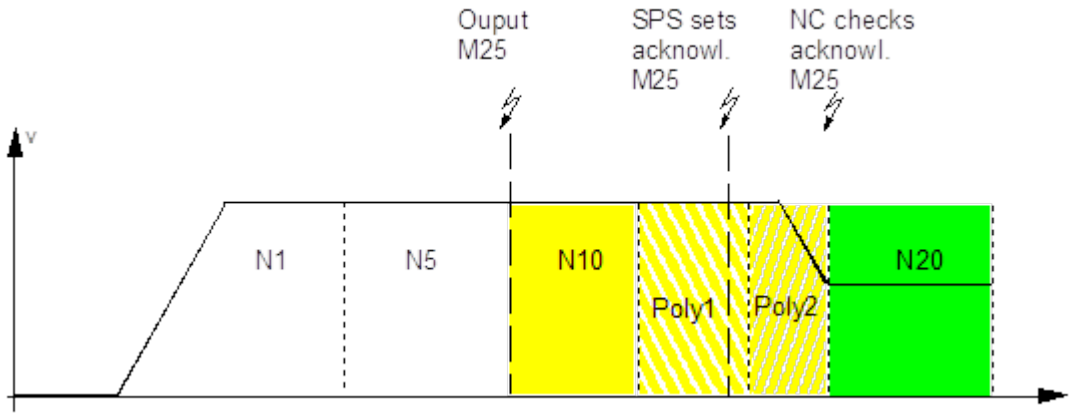


図 36: 図4-37: 輪郭加工後の形状に関連する動作のない同期

このコマンドの実行では、以下の3つの可能性が存在します。

- プレブロック(N10)の直後、および1番目の形状多項式の前
- 1番目と2番目の形状多項式の間
- 2番目の形状多項式の後、およびポストブロック(N20)の前

5.11.2.4 多項式内の加加速度

多項式の曲率は、軌道に交わる軸の加加速度となります。この加加速度は、軸の最大動的パラメータ(P-AXIS-00199)でチェックされます。加加速度が強すぎる場合、経路速度が低減されます。ユーザ定義のケースでは、最大加加速度による減速が望ましくない場合があります。このため、NCコマンド#CONTOUR MODE内でいくつかの特定の制御コマンドを使用できます。これらの制御コマンドはチャンネルパラメータリストP-CHAN-00110の事前定義を上書きし、プログラム終了までモーダル状態が続きます。

以下の例では、N6からN7へのブロック遷移が加加速度を考慮した多項式によって輪郭加工されます。N7からN8への遷移も輪郭加工されますが、経路形状上で加加速度は考慮されません。

プログラミング例

```
%poly_jerk.nc
( Default setting in channel list : 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
```

5.11.2.5 輪郭加工部分での速度特性

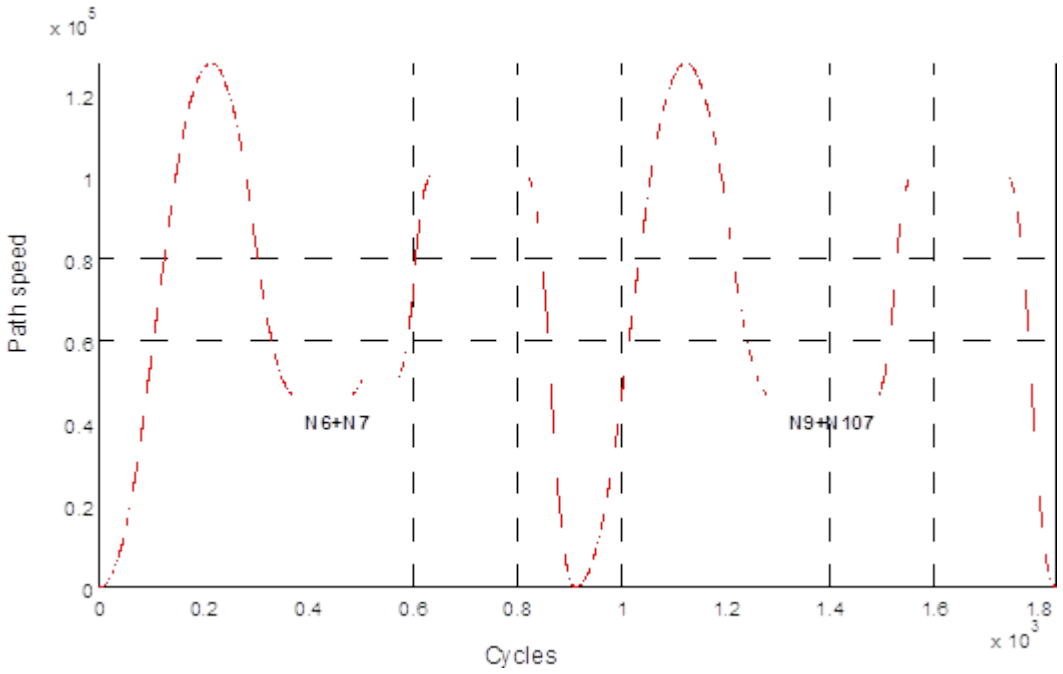
軸のパラメータ設定および用途に応じて、輪郭加工部分での速度特性を変更する必要が発生する場合があります。初期設定の定義では、輪郭加工部分が最大許容経路速度で移動します。これにより、軸のダイナミクスが大幅に異なる場合、許容範囲を超えた機械の振動が発生することがあります。これは、輪郭加工部分で経路速度を適用する必要があるためです。

輪郭加工部分の特性は、NCコマンド#CONTOUR MODE内のいくつかの制御コマンドによって変更できます。

以下の例では、N6からN7へのブロック遷移が加加速度を考慮した多項式によって輪郭加工されます。この多項式では、形状部分を最大速度で移動します。これは、速度調整が異なる軸ダイナミクスで行われることを意味します。N9からN10への遷移も輪郭加工されますが、速度調整は行われません。つまり、輪郭加工部分では経路速度は一定です。

プログラミング例

```
%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
```



5.11.3 NCプログラム内での輪郭加工モードのパラメータ設定

多項式輪郭加工(G61 / G261)を実際に行う前に、NCコマンド#CONTOUR MODEによってさまざまなオプションを設定します。

輪郭加工のタイプに応じて、パラメータ設定でいくつかの特定のキーワードを使用できます。コマンドは、以下の基本構文要素で構成されます。

#CONTOUR MODE[

<contouring_type> PATH_DEV TRACK_DEV VEL PRE_DIST POST_DIST

<1.st_main_ax_name> *<2.nd_main_ax_name>* *<3.rd_main_ax_name>*

RELEVANT_PATH RELEVANT_TRACK REMAIN_PART *<動作>*

CHECK_JERK MAX_ANGLE ACC_MAX ACC_MIN

RAMP_TIME]

<輪郭加工のタイプ>:	AUTO_DEV	最大コーナー偏差を使用した自動輪郭加工(タイプ1)
	AUTO_VEL	減速を使用した自動輪郭加工(タイプ2)
	DIST	コーナー距離輪郭加工(タイプ3)
	DEV	コーナー偏差輪郭加工(タイプ4) **注記 プログラム開始後の初期設定**
	POS	暫定点輪郭加工(タイプ5)
	DIST_SOFT	形状によって制限する加加速度を使用した輪郭加工(タイプ6)

<action>:	PRE_ACTION	形状曲線の**前での**アクション(M/H)の実行
	INTER_ACTION	形状曲線の**途中での**アクション(M/H)の実行
	POST_ACTION	形状曲線の**後での**アクション(M/H)の実行

5.11.4 NCプログラム内での輪郭加工モードの有効化

対応する輪郭加工モードのパラメータ設定後に、GファンクションG61 (ブロック)またはG261 (モーダル)によって輪郭加工が有効になります。

バージョンV2.11.2022.13以降では、オン/オフのプログラミングに加えて、コマンド#CONTOUR MODEを組み合わせて輪郭加工を直接有効および無効にできます。このため、G261/G260のプログラミングは不要です。

コマンドは、以下の構文要素で構成されます。

#CONTOUR MODE ON | OFF [*<輪郭加工パラメータ>*]

プログラミング例

```
%ContourG61

#CONTOUR MODE [AUTO_DEV PATH_DEV=1.0 RELEVANT_PATH=0 RELEVANT_TRACK=0]
N1  G90 G01 X0 Y0 Z0 A0 C0 F60
N10 G1 G61 X99.999 Y0.001
N20 G1     X100            ;error message: 120216, Block to short
                             ;for contouring, contouring stopped

#CONTOUR MODE [AUTO_DEV PATH_DEV=1.0 RELEVANT_PATH=0.1 RELEVANT_TRACK=0]
N2  G90 G01 X0 Y0 Z0 A0 C0 F60
N30 G1 G61 X99.999 Y0.001  ;no error message: 120216
N40 G1     X100            ;Block is skipped, Contouring N30 -> N3

#CONTOUR MODE [AUTO_DEV PATH_DEV=1.0 RELEVANT_PATH=0 RELEVANT_TRACK=0]
N3  G90 G01 X0 Y0 Z0 A0 C0 F60
N50 G1 G61 X100
N60 G1     X200 Z00 C-32.667  ;tangential transition of main axes
                               (contouring is suppressed)

#CONTOUR MODE [AUTO_DEV PATH_DEV=1.0 RELEVANT_PATH=0 RELEVANT_TRACK=0.1]
N4  G90 G01 X0 Y0 Z0 A0 C0 F60
N70 G1 G61 X100
N80 G1     X200 Z00 C-32.667   ;tangential transition of main axes
                              ;contouring is not suppressed because
                              ;maximum deviation of slave axes
N10000 G1 G61 X300 Z0        ;error message: 120206, program end
                             ;during active contouring
N0210 M30



%Contour_on_off

N10 G90 G01 X0 Y0 Z0 A0 C0 F60
N20 #CONTOUR MODE ON [DEV PATH_DEV=1.0]    ;Parameterization and
                                           ;activation (= G261)
N30 X100
N40 Y100
N50 X0
```

```
N60 Y0
N70 #CONTOUR MODE OFF                    ;Deactivation (= G260)
N80 M30
```

5.11.4.1 最大コーナー偏差を使用した自動輪郭加工(タイプ1)

コーナー距離が自動的に決定され、プレブロックとポストブロックで同一になります。必要なコーナー距離の推定値を決定するために、軸データの最大空間加速度と必要な(プログラムされた)移動速度が加味されます。

コーナー距離は、以下の境界条件に従って決定されます。

- 形状経路は、最大速度が低減せずにできるだけ遠くを移動できる必要があります。
- 指定された**最大コーナー偏差**を超えてはなりません。

コーナー偏差(タイプ4)での輪郭加工に対する主な差分は、軸ダイナミクスを考慮しています。プログラムされた速度での移動が最大多項式長を必要としない場合、その多項式は短縮されます。

この処理は、以下に適しています。

- プログラムされた速度に応じて、コーナー距離を変更できます。つまり、多項式形状はプログラムされた異なる速度で変わる可能性があります。
- 推定値に変換が含まれないため、キネマティック/直交変換は有効になりません。

以下のステートメントを使用してパラメータ設定が行われます。

```
#CONTOUR MODE [ AUTO_DEV  [PATH_DEV<expr>]  [RELEVANT_PATH<expr>]
                [RELEVANT_TRACK<expr>]  [TRACK_DEV<expr>]  [REMAIN_PART<expr>]
                [<action>]  [CHECK_JERK<expr>]  [MAX_ANGLE<expr>]  [CONST_VEL<expr>] ]
```

AUTO_DEV　　最大コーナー偏差を使用した自動輪郭加工

PATH_DEV<*expr*>　　プログラムされた形状からのmm単位での最大偏差

初期値: 1 mm

RELEVANT_PATH<*expr*>　　関連するポストブロックのmm単位での最小経路長

初期値: 0 mm

RELEVANT_TRACK<*expr*>　　関連するポストブロックに対する従軸のmm単位での最小経路長

初期値: 0 mm

TRACK_DEV*<expr>* 結合軸のmm単位での最大偏差

初期値: 1 mm

REMAIN_PART*<expr>* 元のブロックのパーセント単位[0; 100]での残りの部分

初期値: 0 %

<action> 追加動作の実行時間

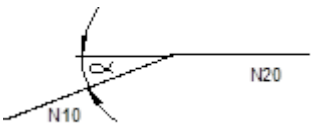
有効なキーワード	優先度
PRE_ACTION	形状曲線の**前での**実行。
INTER_ACTION	形状曲線の**途中での**実行(初期設定)。
POST_ACTION	形状曲線の**後での**実行。

CHECK_JERK*<expr>* 多項式の曲率によって生じる加加速度のモニタリング(P-CHAN-00110)。

値	優先度
0	加加速度モニタリングなし(初期設定)。
1	形状的なランプ時間に基づいた加加速度モニタリング(P-AXIS-00199)。経路速度が低減される可能性あり。
2	ランプ時間に基づいた加加速度モニタリング(P-AXIS-00195)。非直線速度プロファイルのP-AXIS-00198。

MAX_ANGLE*<expr>* 輪郭加工が有効になるまでの、2つの直線ブロック間の遷移の最大形状角度。いずれかの遷移ブロックが円弧ブロックの場合は、形状角度は考慮されません。

初期値: 0° (すべての経路要素が輪郭加工されます)



CONST_VEL*<expr>* 輪郭加工領域での一定の経路速度。

プログラミング例

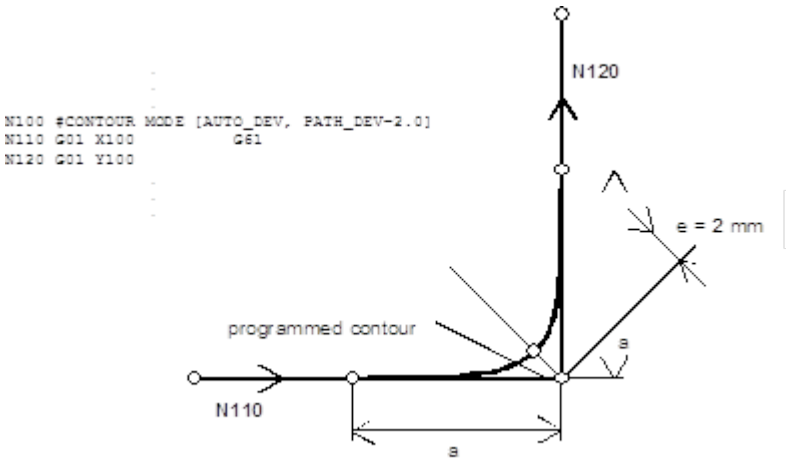


図 37: 図4-38: 自動輪郭加工(タイプ1)

5.11.4.2 特定の速度での自動輪郭加工(タイプ2)

このタイプの輪郭加工では、プレブロックのプログラムされた速度の指定されたパーセント値vb_prozで曲線を移動できるように、形状曲線が決定されます。例えば、vb_proz = 50%の場合、プレブロック内でプログラムされた速度の半分で形状曲線を移動します。

ただし、形状の曲率、および軸の最大加速度により、目的の速度には達しません。

以下のステートメントを使用してパラメータ設定が行われます。

```
#CONTOUR MODE [ AUTO_VEL  [VEL<expr>]  [RELEVANT_PATH<expr>]
                [RELEVANT_TRACK<expr>]  [TRACK_DEV<expr>]  [REMAIN_PART<expr>]
                [<action>]  [CHECK_JERK<expr>] [MAX_ANGLE<expr>] [CONST_VEL<expr>] ]
```

AUTO_VEL 特定の速度での自動輪郭加工

VEL<*expr*> 速度のパーセンテージ[0; 100]

初期値: 100 %

RELEVANT_PATH<*expr*> 関連するポストブロックのmm単位での最小経路長

初期値: 0 mm

RELEVANT_TRACK<*expr*> 関連するポストブロックに対する従軸のmm単位での最小経路長

初期値: 0 mm

TRACK_DEV<*expr*> 結合軸のmm単位での最大偏差

初期値: 0 mm

REMAIN_PART*<expr>* 元のブロックのパーセント単位[0; 100]での残りの部分

初期値: 0 %

<action> 追加動作の実行時間:

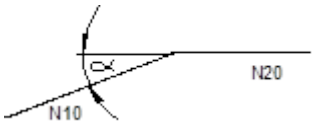
有効なキーワード	優先度
PRE_ACTION	形状曲線の**前での**実行。
INTER_ACTION	形状曲線の**途中での**実行(初期設定)。
POST_ACTION	形状曲線の**後での**実行。

CHECK_JERK*<expr>* 多項式の曲率によって生じる加加速度のモニタリング(P-CHAN-00110)。

値	優先度
0	加加速度モニタリングなし(初期設定)。
1	形状的なランプ時間に基づいた加加速度モニタリング(P-AXIS-00199)。経路速度が低減される可能性あり。
2	ランプ時間に基づいた加加速度モニタリング(P-AXIS-00195)。非直線速度プロファイルのP-AXIS-00198。

MAX_ANGLE*<expr>* 輪郭加工が有効になるまでの、2つの直線ブロック間の遷移の最大形状角度。いずれかの遷移ブロックが円弧ブロックの場合は、形状角度は考慮されません。

初期値: 0° (すべての経路要素が輪郭加工されます)



CONST_VEL*<expr>* 輪郭加工領域での一定の経路速度。

プログラミング例

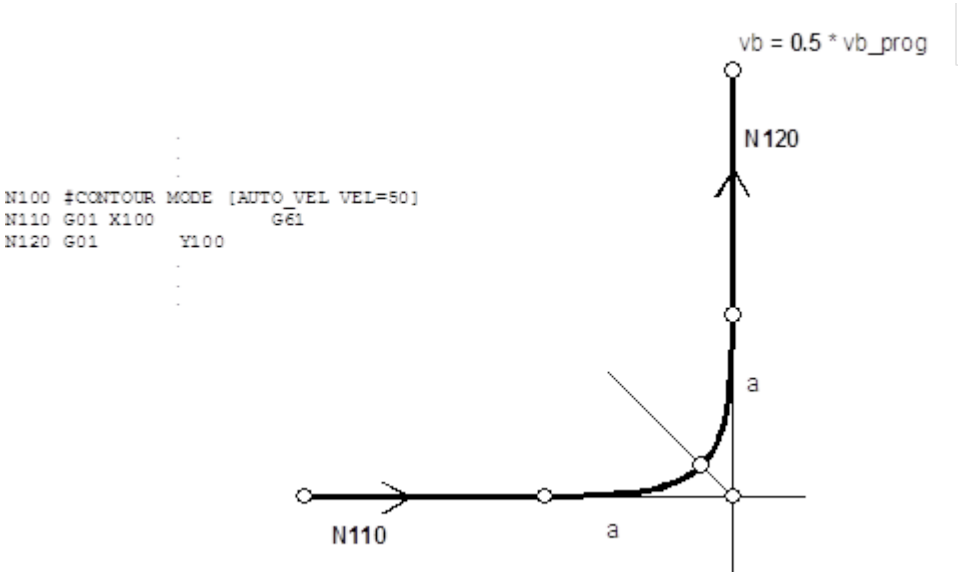


図 38: 図4-39: 自動輪郭加工(タイプ2)

5.11.4.3 コーナー距離輪郭加工(タイプ3)

元の形状が残っている可能性がある位置が分かっている場合、ユーザは連続する移動ブロックが短縮されるプレブロックとポストブロックのコーナー距離を明示的に指定できます。

プレブロックとポストブロックの経路長の45%を超える場合は、角距離が制限されます。

- コーナー距離aおよびbの定義が同じ場合、経路長の45%に制限されるとともに、両方のコーナー距離も対称に制限されます。
- コーナー距離aおよびbの定義が異なる場合、制限時によって遠い距離のみが短縮されます。経路長が非対称な場合、これは形状の「変形」を招く可能性がありますが、意図的に行われることもあります(エキスパートモード)。

以下のステートメントを使用してパラメータ設定が行われます。

```
#CONTOUR MODE [ DIST   [PRE_DIST<expr>]   [POST_DIST<expr>]   [RELEVANT_PATH<expr>]
                       [RELEVANT_TRACK<expr>]  [TRACK_DEV<expr>]  [REMAIN_PART<expr>]
                       [<動作>]  [CHECK_JERK<expr>]  [MAX_ANGLE<expr>]  [CONST_VEL<expr>] ]
```

DIST　　コーナー距離輪郭加工

PRE_DIST<*expr*>　　コーナーまでの距離(mm)。元の形状からの偏差点

初期値: 1 mm

POST_DIST<*expr*>　　コーナー以降の距離(mm)。元の形状の回帰点

初期値: 1 mm

RELEVANT_PATH*<expr>* 関連するポストブロックのmm単位での最小経路長

初期値: 0 mm

RELEVANT_TRACK*<expr>* 関連するポストブロックに対する従軸のmm単位での最小経路長

初期値: 0 mm

TRACK_DEV*<expr>* 結合軸のmm単位での最大偏差

初期値: 0 mm

REMAIN_PART*<expr>* 元のブロックのパーセント単位[0; 100]での残りの部分

初期値: 0 %

<action> 追加動作の実行時間:

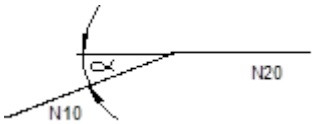
有効なキーワード	優先度
PRE_ACTION	形状曲線の**前での**実行。
INTER_ACTION	形状曲線の**途中での**実行(初期設定)。
POST_ACTION	形状曲線の**後での**実行。

CHECK_JERK*<expr>* 多項式の曲率によって生じる加加速度のモニタリング(P-CHAN-00110)。

値	優先度
0	加加速度モニタリングなし(初期設定)。
1	形状的なランプ時間に基づいた加加速度モニタリング(P-AXIS-00199)。経路速度が低減される可能性あり。
2	ランプ時間に基づいた加加速度モニタリング(P-AXIS-00195)。非直線速度プロファイルのP-AXIS-00198。

MAX_ANGLE*<expr>* 輪郭加工が有効になるまでの、2つの直線ブロック間の遷移の最大形状角度。いずれかの遷移ブロックが円弧ブロックの場合は、形状角度は考慮されません。

初期値: 0° (すべての経路要素が輪郭加工されます)



CONST_VEL<*expr*>　　輪郭加工領域での一定の経路速度。

プログラミング例

```
.
.
.
N100 #CONTOUR MODE [DIST   PRE_DIST=10   POST_DIST=5]
N110 G01 X100                    G61
N120 G01          Y100
.
.
.
```

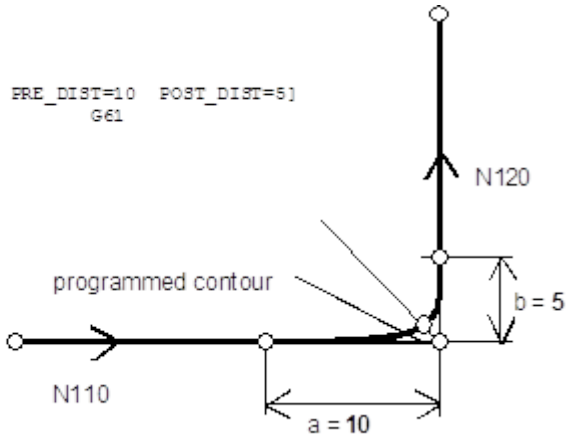


図 39: 図4-40: コーナー距離輪郭加工(タイプ3)

5.11.4.4　コーナー偏差輪郭加工(タイプ4)

輪郭加工タイプ4の初期設定は、プログラム開始後の多項式輪郭加工の初期設定でもあります!

連続する移動ブロックが短縮されるコーナー距離は、ユーザによって指定されるコーナー偏差を超えないように、形状的考慮によって自動的に決定されます。コーナー距離はブロック長の半分に制限されますが、両方の距離が対称に制限されます。タイプ1とは異なり、プログラムされた速度は多項式形状の影響は受けません。

以下のステートメントを使用してパラメータ設定が行われます。

```
#CONTOUR MODE [ DEV   [PATH_DEV<expr>]   [RELEVANT_PATH<expr>]
                      [RELEVANT_TRACK<expr>]  [TRACK_DEV<expr>]  [REMAIN_PART<expr>]
                      [<動作>]  [CHECK_JERK<expr>]  [MAX_ANGLE<expr>]  [CONST_VEL<expr>] ]
```

DEV　　コーナー偏差輪郭加工

PATH_DEV*<expr>* プログラムされた形状からのmm単位での最大偏差

初期値: 1 mm

RELEVANT_PATH*<expr>* 関連するポストブロックのmm単位での最小経路長

初期値: 0 mm

RELEVANT_TRACK*<expr>* 関連するポストブロックに対する従軸のmm単位での最小経路長

初期値: 0 mm

TRACK_DEV*<expr>* 結合軸のmm単位での最大偏差

初期値: 0 mm

REMAIN_PART*<expr>* 元のブロックのパーセント単位[0; 100]での残りの部分

初期値: 0 %

<action> 追加動作の実行時間:

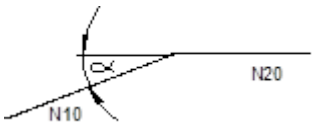
有効なキーワード	優先度
PRE_ACTION	形状曲線の**前での**実行。
INTER_ACTION	形状曲線の**途中での**実行(初期設定)。
POST_ACTION	形状曲線の**後での**実行。

CHECK_JERK*<expr>* 多項式の曲率によって生じる加加速度のモニタリング(P-CHAN-00110)。

値	優先度
0	加加速度モニタリングなし(初期設定)。
1	形状的なランプ時間に基づいた加加速度モニタリング(P-AXIS-00199)。経路速度が低減される可能性あり。
2	ランプ時間に基づいた加加速度モニタリング(P-AXIS-00195)。非直線速度プロファイルのP-AXIS-00198。

MAX_ANGLE*<expr>* 輪郭加工が有効になるまでの、2つの直線ブロック間の遷移の最大形状角度。いずれかの遷移ブロックが円弧ブロックの場合は、形状角度は考慮されません。

初期値: 0° (すべての経路要素が輪郭加工されます)



CONST_VEL*<expr>* 輪郭加工領域での一定の経路速度。

プログラミング例

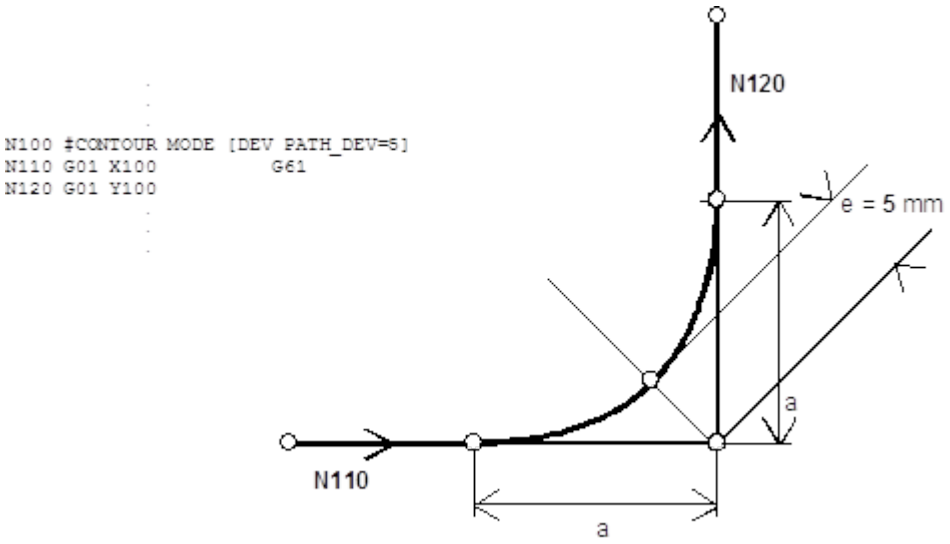


図 40: 図4-41: コーナー偏差輪郭加工(タイプ4)

5.11.4.5 暫定点輪郭加工(タイプ5)

ここでは、ユーザはコーナー距離だけでなく、2つの多項式曲線が互いに接する暫定点PZも指定します(エキスパートモード)。この処理により、プログラムされた形状を維持し、コーナー距離を0に指定することでダイナミクスを完全に活用できます。つまり、コーナー距離が対称であってはならないことを意味します。

以下のステートメントを使用してパラメータ設定が行われます。

```
#CONTOUR MODE [ POS   [PRE_DIST<expr>]   [POST_DIST<expr>]
                      [<1.Hauptachsname><expr>]   [<2.Hauptachsname><expr>]
                      [<3.Hauptachsname><expr>]   [<action>]   [CHECK_JERK<expr>]
                      [CONST_VEL<expr>] ]
```

POS 暫定点輪郭加工

PRE_DIST*<expr>* コーナーまでの距離(mm)。元の形状からの偏差点。

初期値: 1 mm

POST_DIST*<expr>* コーナー以降の距離(mm)。元の形状の回帰点。

初期値: 1 mm

<1.st_main_ax_name><expr> 1番目の主軸のmm単位での暫定点の位置

<2.nd_main_ax_name><expr> 2番目の主軸のmm単位での暫定点の位置

<3.rd_main_ax_name><expr> 3番目の主軸のmm単位での暫定点の位置

<action> 追加動作の実行時間:

有効なキーワード	優先度
PRE_ACTION	形状曲線の**前での**実行。
INTER_ACTION	形状曲線の**途中での**実行(初期設定)。
POST_ACTION	形状曲線の**後での**実行。

CHECK_JERK*<expr>* 多項式の曲率によって生じる加加速度のモニタリング(P-CHAN-00110)。

値	優先度
0	加加速度モニタリングなし(初期設定)。
1	形状的なランプ時間に基づいた加加速度モニタリング(P-AXIS-00199)。経路速度が低減される可能性あり。
2	ランプ時間に基づいた加加速度モニタリング(P-AXIS-00195)。非直線速度プロファイルのP-AXIS-00198。

CONST_VEL*<expr>* 輪郭加工領域での一定の経路速度。

プログラミング例

```
N100 #CONTOUR MODE [POS PRE_DIST=2 POST_DIST=3 X110 Y-10 Z0]
N110 G01 X100 G61
N120 G01 Y100
```

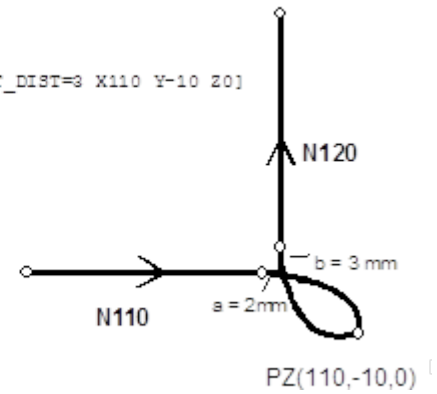


図 41: 図4-42: 暫定点輪郭加工(タイプ5)

5.11.4.6 動的に最適化された輪郭加工(タイプ6)

現在のところ、多項式輪郭加工タイプ1～5は、方向および曲率が連続する2つの移動ブロックの結合を定義します。軸を基準にして、形状曲線が加速度を変化させる場合があります。

軸に関連した動的データ(加速度、加加速度)を考慮し、輪郭加工曲線は関連する軸の**一定の加速度**(最小加加速度)で決定されます。輪郭加工の**時間**とともに軸の最大加速度を使用して、輪郭加工が減らされます。

以下のステートメントを使用してパラメータ設定が行われます。

```
#CONTOUR MODE [ DIST_SOFT  [PATH_DIST<expr>]  [TRACK_DIST<expr>]
                           [ACC_MAX<expr>]  [ACC_MIN<expr>]  [RAMP_TIME<expr>]
                           [DIST_WEIGHT<expr>]]
```

DIST_SOFT　　動的に最適化される輪郭加工

PATH_DIST<*expr*>　　元の形状からの偏差が許容される、プレブロックとポストブロック(対称)のコーナー距離(mmまたは度)。この定義は、送り軸の移動距離に対応しています。

初期値: 1 mmまたは1 °

モニタリングオフ: -1 mmまたは-1 °

TRACK_DIST<*expr*>　　元の形状からの非送り軸(抗力軸)の偏差が許容される、プレブロックとポストブロック(対称)のコーナー距離(mmまたは度)。

初期値: プログラム開始から明示的に定義されていない場合、この値はPATH_DISTから自動的にマッピングされます。

モニタリングオフ: -1 mmまたは-1 °

ACC_MAX*<expr>* 形状によって使用されるパーセント単位での最大加速度(機械データ)。

初期値: 100 %

ACC_MIN*<expr>* 形状によって使用されるパーセント単位[0; 100]での最小加速度(機械データ)。この加速度が特定のコーナー距離を維持するには小さすぎる場合(SYM_SISTを参照)、使用される加速度が最大値(ACC_MAX)まで増加されます。

初期値: 50 %

RAMP_TIME*<expr>* ランプ時間の重み付けのパーセンテージ[0; 10000]。

初期値: 100 %

DIST_WEIGHT*<expr>* 前後のブロックに対するコーナー距離の重み付けのパーセンテージ[0; 100]。

初期値: 0 %

制限事項:

- 動的に最適化される輪郭加工は、直線ブロックの遷移で使用できます。円弧ブロックが遷移に含まれる場合、計算が標準的なコーナー距離計算(動的最適化なし、タイプ3に準拠)に切り替わります。
- 計算には、各軸に対して1つのランプ時間のみが使用されます(個々のランプ時間の最大値)。
- キネマティック変換は考慮されません。いずれかの変換が有効な場合、計算も自動モードから標準モードに切り替わります(タイプ3に準拠)。

多くの場合、パラメータDIST_WEIGHTによる前後のブロックに応じたコーナー距離の重み付けにより、利用可能なブロック長を最適に使用できます。

軸特有の輪郭加工中は、前後のブロックのコーナー距離は基本的に同じ(対称)になります。さらに、最大コーナー距離がブロック移動距離の半分に制限されている場合には、移動距離が前後の移動距離よりも長いと、輪郭加工部分および輪郭加工速度が小さくなります。

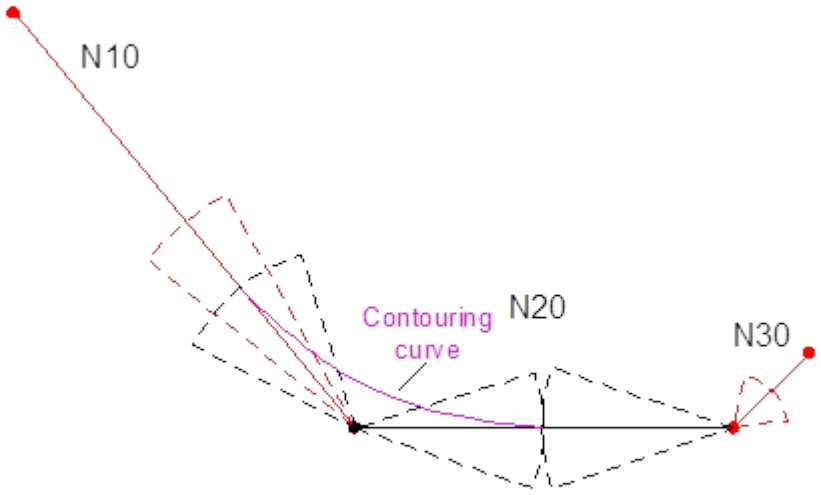


図 42: 図4-43: N10およびN20のブロック長に依存しないブロックN20の最大コーナー距離(DIST_WEIGHT = 0 %)

最大コーナー距離の計算に前後のブロック長が考慮される場合、輪郭加工部分を増やすことができます。

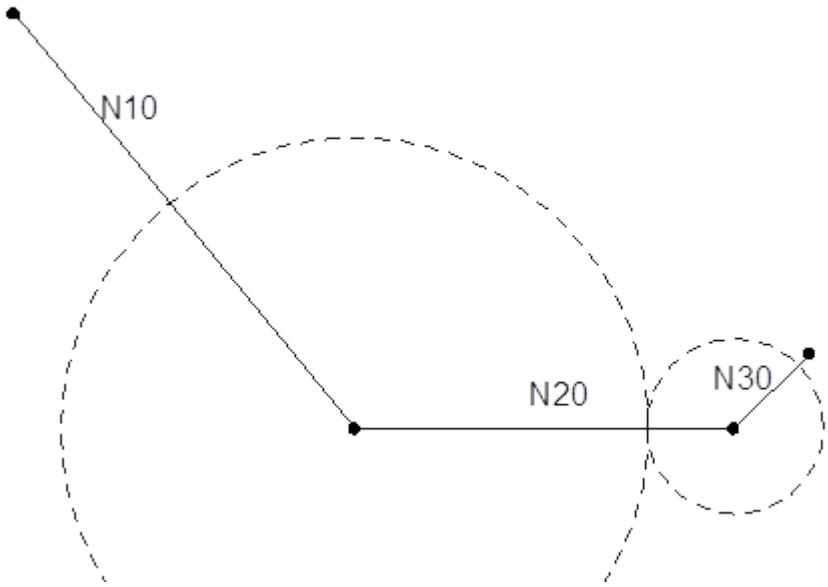


図 43: 図4-44: N10およびN30の割合で除算されるブロックN20の最大コーナー距離(DIST_WEIGHT = 100 %)

プログラミング例

4分の1の角度(90°)の輪郭加工方法の比較:

- 半径の明示的な挿入(G302):

```
P1 = 150 (Eckenabstand
P4 = 2000 (Anfahrposition
N10 X-P4 YP4
N20 X0 Y0
N25 G302 IP1
N30 XP4 YP4
N40 M30
```

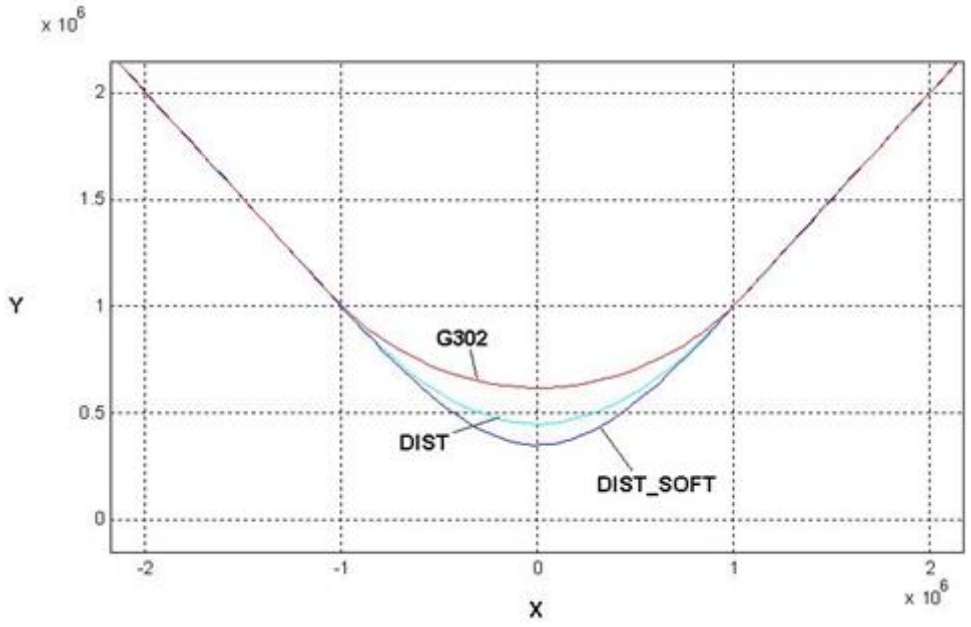
- コーナー距離の定義を使用した通常の輪郭加工(DIST):

```
#SLOPE[TYPE=TRAPEZ]
P1 = 150 (Eckenabstand
P4 = 2000 (Anfahrposition
#CONTOUR MODE [DIST PRE_DIST=P1 POST_DIST=P1]
N10 X-P4 YP4
N20 X0 Y0 G61
N30 XP4 YP4
N40 M30
```

- 動的に最適化される輪郭加工(DIST_SOFT):

```
#SLOPE[TYPE=TRAPEZ]P1 = 150 (Eckenabstand
P4 = 2000 (Anfahrposition
#CONTOUR MODE [DIST_SOFT PATH_DIST=P1 ACC_MAX=100 ACC_MIN=50 RAMP_TIME=100]
N10 X-P4 YP4
N20 X0 Y0 G61
N30 XP4 YP4
N40 M30
```

所定の3つのモードの比較:



5.11.5 例

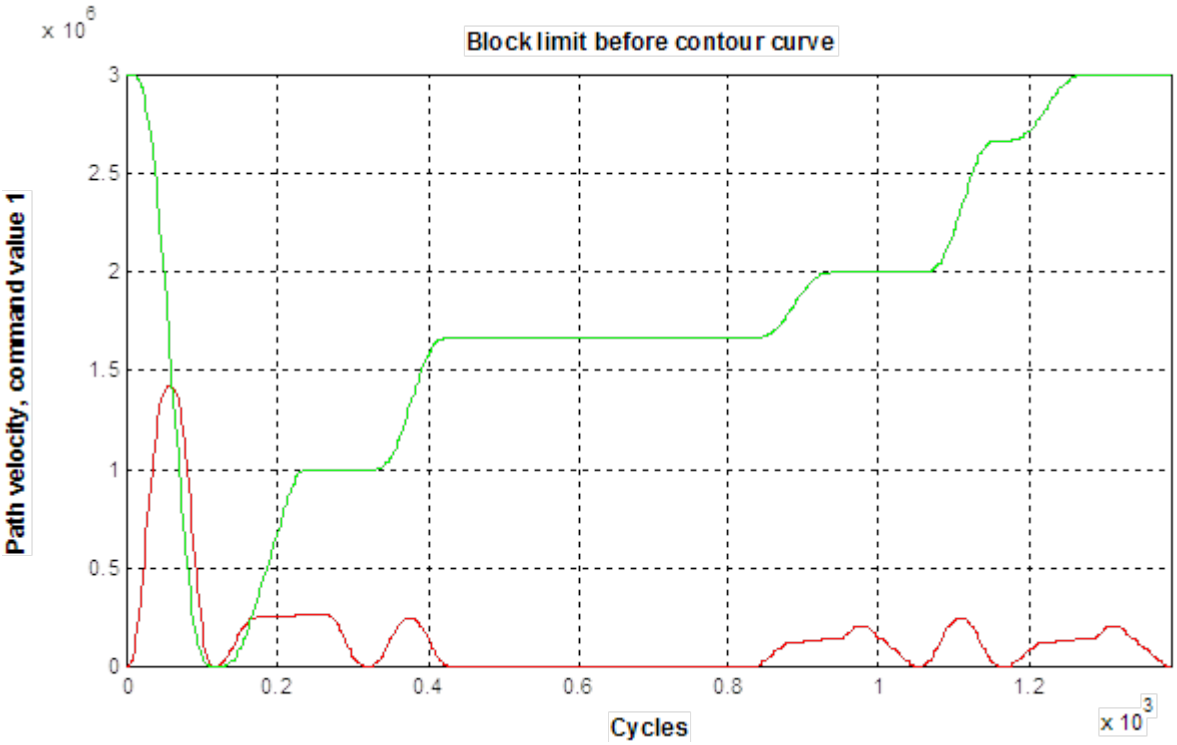
プログラミング例

```
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
N23 M30
```

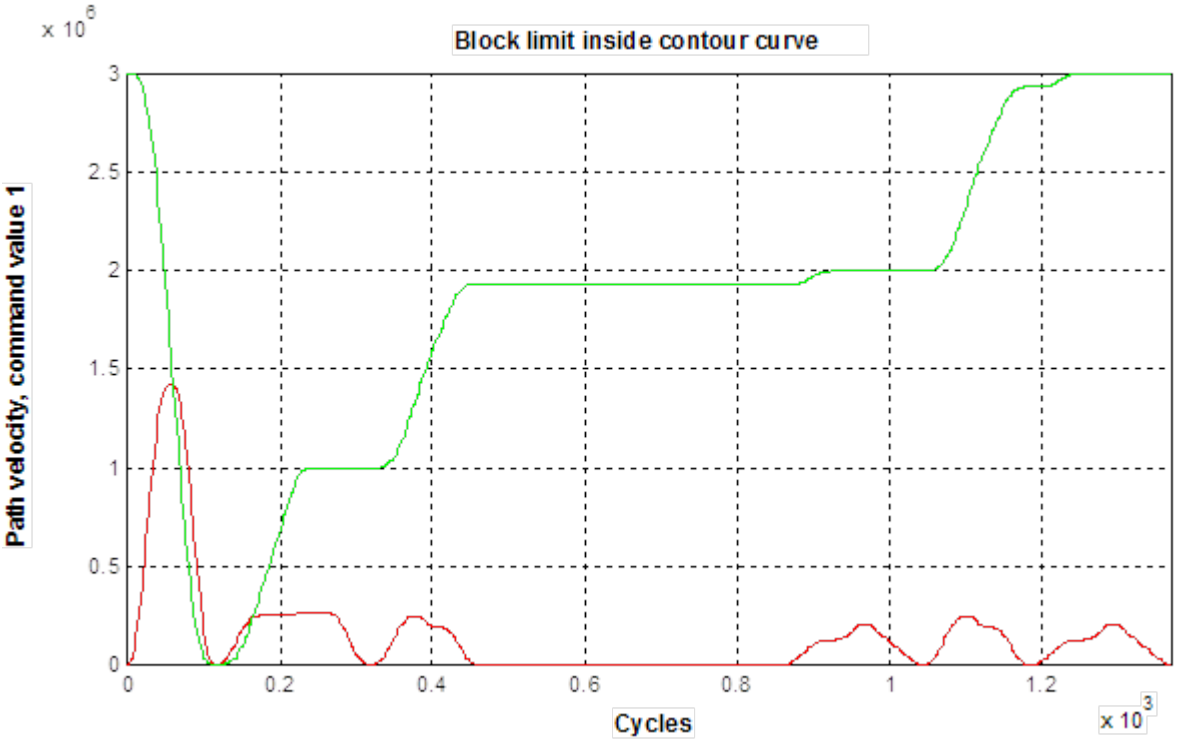
多項式形状曲線の前での出力:

N01 #CONTOUR MODE [DEV PATH_DEV=10.0 **PRE_ACTION**]



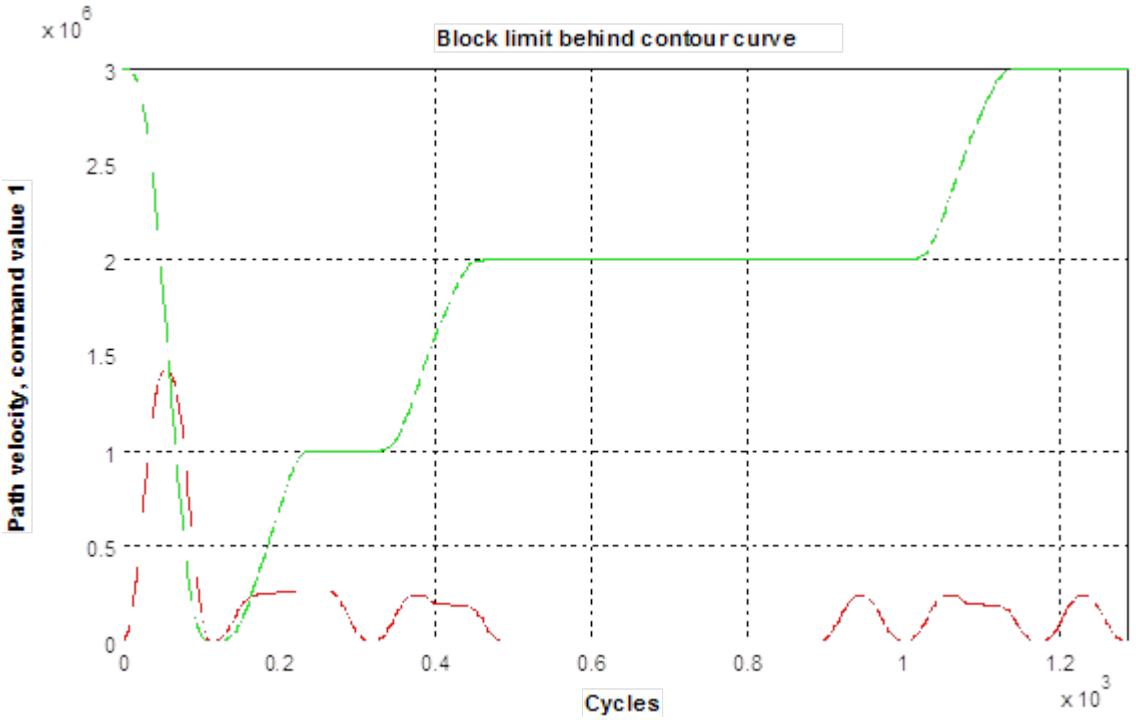
多項式形状曲線の途中での出力:

N01 #CONTOUR MODE [DEV PATH_DEV=10.0 **INTER_ACTION**]

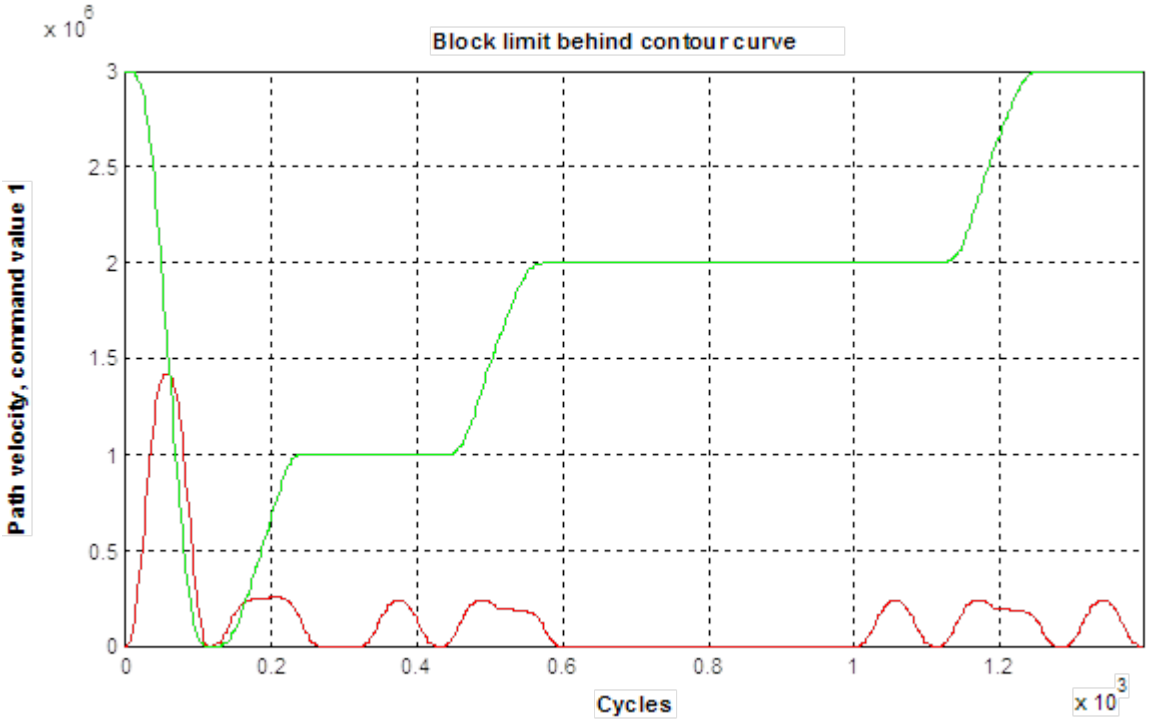


多項式形状曲線の後での出力:

N01 #CONTOUR MODE [DEV PATH_DEV=10.0 **POST_ACTION**]



M25の応答確認が遅延すると、動作が形状曲線の後で停止し、PLCの応答確認を待機します。



プログラミング例

輪郭加工中の制限角度の変更:

```
#CONTOUR MODE [DEV PATH_DEV=0.50 RELEVANT_PATH=0.1 TRACK_DEV=2 RELEVANT_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
```

結果:

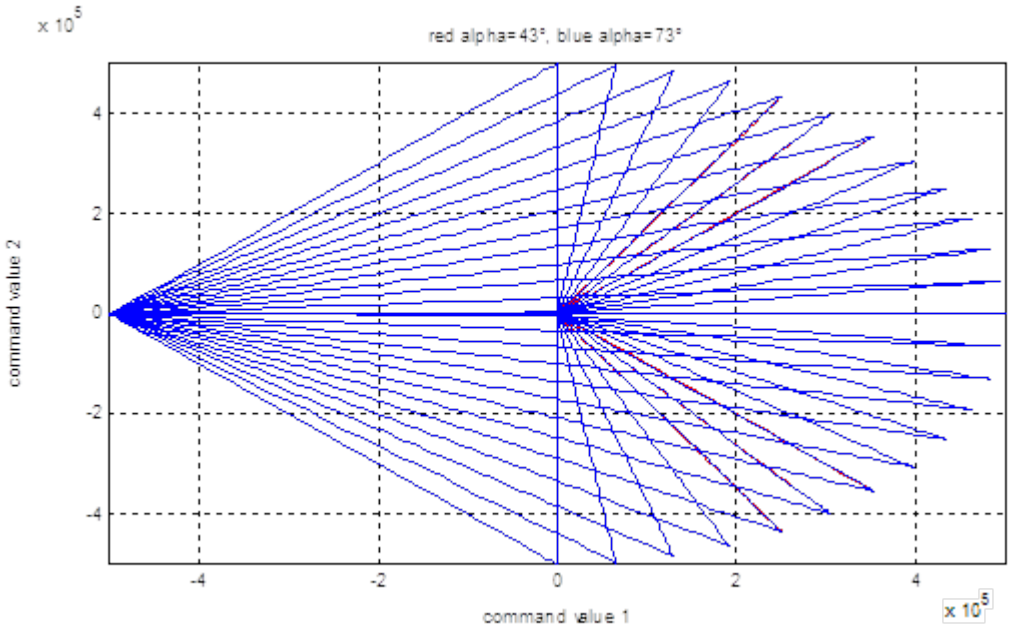
ブロックN<i>の輪郭加工は、必ず前のブロックN<i-5>の制限角度で実行されます。

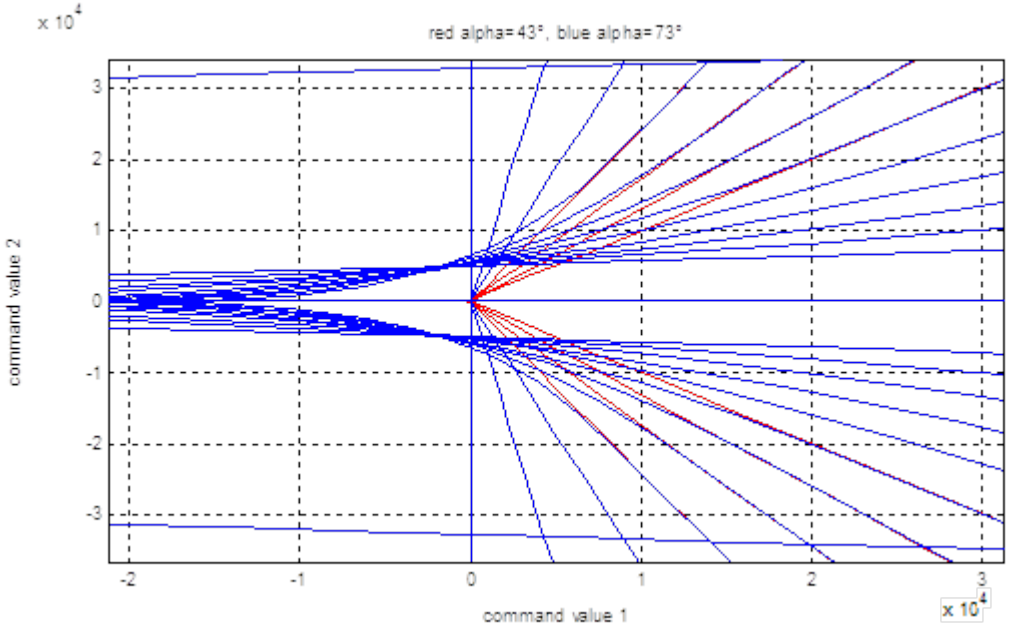
プログラミング例

一定の制限角度の形状角度の種類:

```
#CONTOUR MODE [DEV PATH_DEV=0.50 RELEVANT_PATH=0.1 TRACK_DEV=2 RELEVANT_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
```



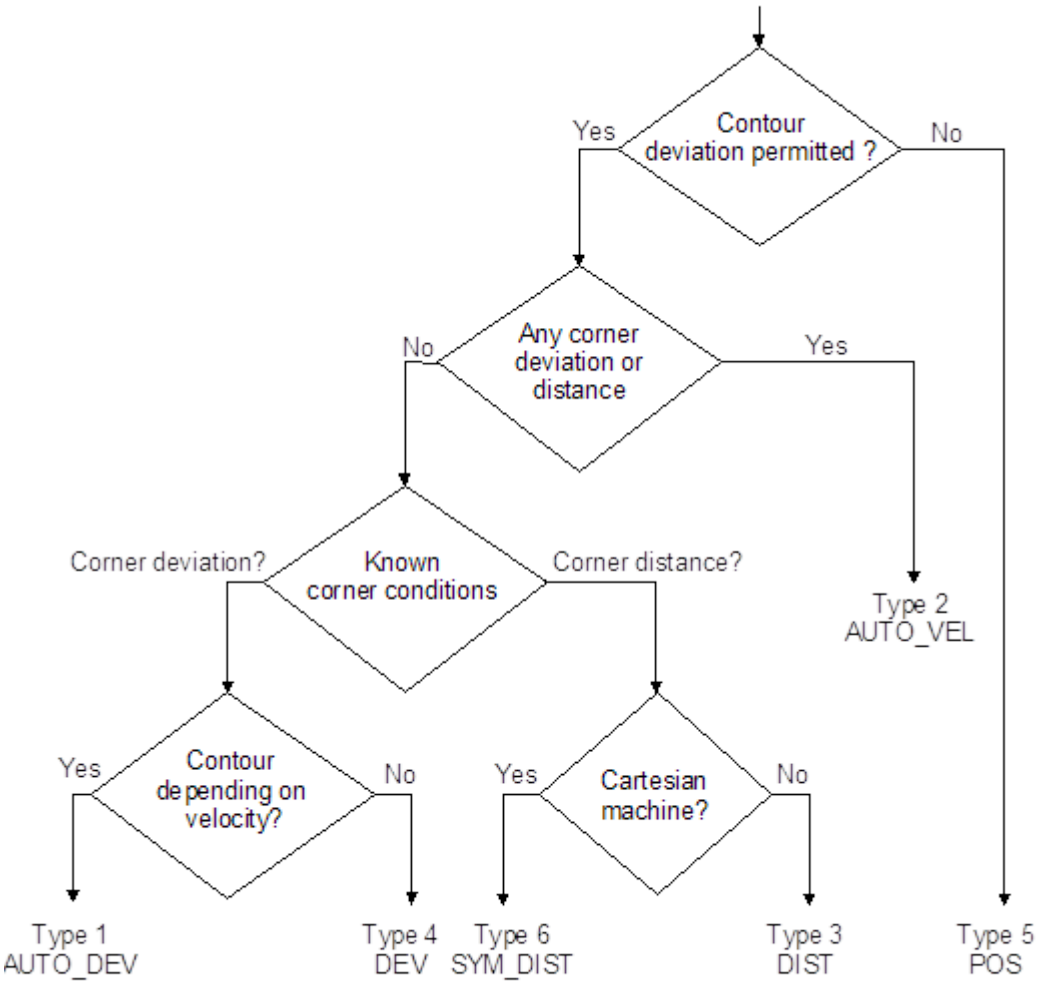


5.11.6 プロセスの概要

以下の表に、使用可能なプロセス、および輪郭加工パラメータの決定方法の概要を記載します。

輪郭加工パラメータ	自動輪郭加工 1	自動輪郭加工 2	距離輪郭加工	コーナー偏差輪郭加工	暫定点輪郭加工	動的輪郭加工
プレ距離/ポスト距離	自動	自動	ユーザによる	自動	ユーザによる	ユーザによる
コーナー偏差	ユーザによる	パラメータ設定不可	パラメータ設定不可	ユーザによる	パラメータ設定不可	パラメータ設定不可
暫定点	自動	自動	自動	自動	ユーザによる	自動

以下の図は、用途に最適な輪郭加工タイプの特定に役立ちます。いずれのケースでも、多項式輪郭加工は**元の形状**と**速度の低減**(曲率の連続)が考慮されます。



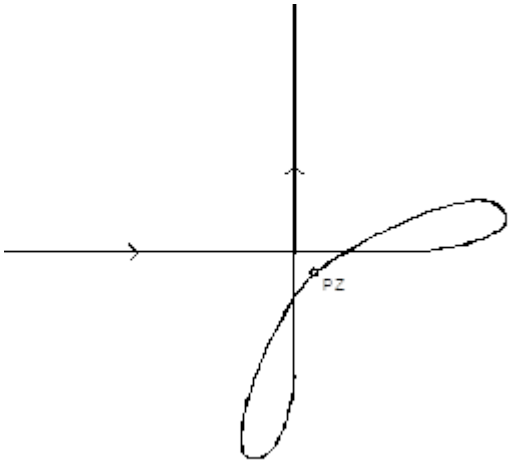
5.11.7 注意

- G61またはG261/G260 (ブロック終了時の多項式輪郭加工)のプログラミング後に軸が解放される、または呼び出されると、輪郭加工の手順が実行されない可能性があります。

プログラミング例

```
N10 G01  X100  Y0  Z0  F1000
N20 G01  X50   Y50  G61
N30 #PUT AX [Z]       ( --> Contouring procedure will not be executed)
N40 G01  X100
N50 M30
```

- 暫定点輪郭加工では、曲線形状は暫定点の選択によって変化します。以下の曲線形状が考えられます。



5.12 コーナー減速

切削加工では、内コーナー付近において、除去体積V_zおよび必要なスピンドル出力が増加します。それはスライス状に順番に加工する過程で工具がすでに材料を取り除いているからです(図4-45)。

機械加工が常にスピンドルの性能限界で行われる場合、スピンドルが内コーナー付近でも十分に動作するように、コーナー領域での送り速度が低減される必要があります。スピンドルの性能限界を超えないようにするために、NCプログラムで経路上の点を指定します。この点以降は速度が低減されます。このコーナー減速には、パラメータ値の設定、有効化、および無効化の3つのNCコマンドを使用できます。

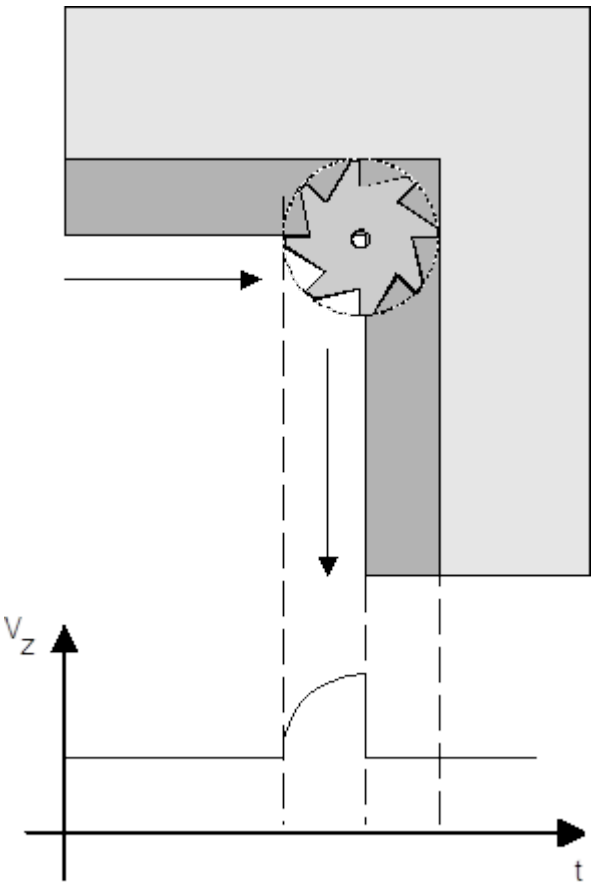


図 44: 図4-45: 内コーナー90°および一定送り速度における除去体積V_zと時間の変化

5.12.1 コーナー減速パラメータ値の設定

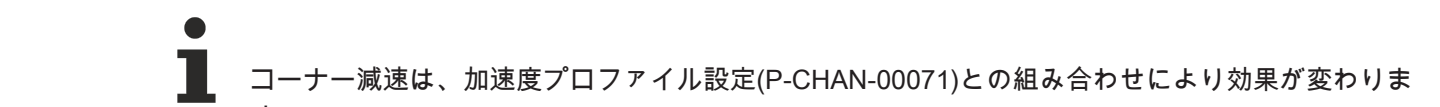
バージョンV2.11.2010.02からは、コマンド#CORNER PARAM [...]をコマンド#SET CORNER PARAM [...]の代わりに使用します。互換性の理由から、このコマンドは当分の間使用できますが、新しいNCプログラムでは使用しないでください。

#CORNER PARAM [DIST<*expr*> UNIT<*ident*> FEED<*expr*>]	(モーダル)

DIST<expr>	コーナー間の距離[mm、インチ]
UNIT<*ident*>	コーナー*送り速度*の単位使用可能な項目:
	FWORD F指令に設定している単位
	PERCENT パーセント[%]
FEED<expr>	コーナー送り速度、[単位はUNIT<*ident*>で設定]

コーナー間の距離を指定することで、送り速度を低減する直線区間を計算して定義することができます。ここでは、以下の定義が適用されます。

プログラムされるコーナー間の距離は、元のプログラム上の距離ではなく、補正された経路上の距離となります。



コーナー減速は、加速度プロファイル設定(P-CHAN-00071)との組み合わせにより効果が変わります。

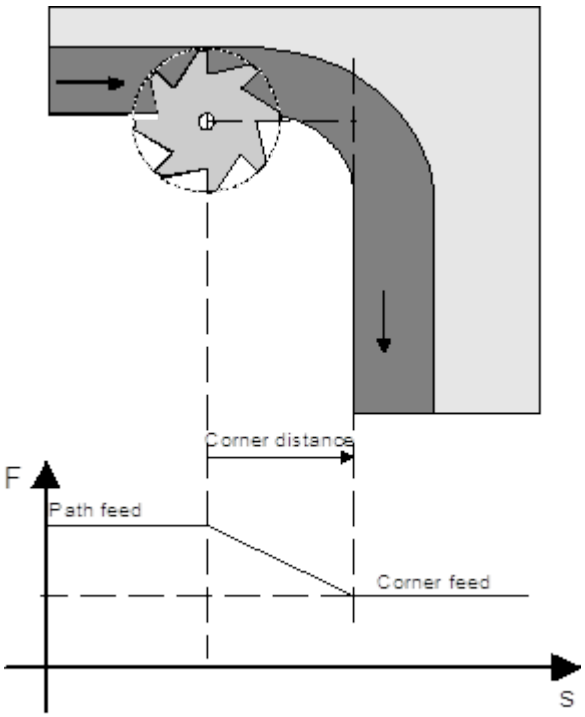


図 45: 図4-46: コーナー円弧形状での送り速度の図

5.12.2 コーナー減速の選択および解除(G12/G13)

G12	コーナー減速の解除	(モーダル、デフォルト)
G13	コーナー減速の選択	(モーダル)

あらかじめコーナー減速パラメータを定義せずにG13がプログラムされると、エラーメッセージが出力されます。

NCブロック内で同時にG12とG13がプログラムされると、エラーメッセージが出力されます。

これらのGファンクションはモーダルです。システム起動後の初期状態は、「コーナーでの減速が無効」(G12)です。

コーナー減速が選択されている場合(G13)にプログラム終了(M30)しても、エラーメッセージは出力されません。プログラム開始時には、デフォルトであるG12が選択されます。

コーナー間の距離が0の場合は、エラーメッセージが出力されます。

プログラミング例

```
N010 G01 G90 X10 Y10 F1000
N015 #CORNER PARAM[DIST=10 UNIT=PERCENT FEED=50]  (Corner distance 10mm)
                                                  (Corner feed 50%)
N020 G13                            (Activation)
N030 X50                            (Corner deceleration active)
N040 Y50                            (Corner deceleration active)
N050 G12                            (De-selection)
N060 X30 Y30
N050 M30
```

ブロックN030の前にはコーナー減速が有効にならないことに注意してください。N010からN030へのコーナー遷移では、コーナー減速の影響を受けません。

同様に、コーナー減速はブロックN040内でも解除されないため、有効です。

コーナー減速は、コーナー減速が有効になるべきではないNCブロックの前、または**そのブロック内**で解除される必要があります。

5.13 工具径補正(G40/G41/G42)

G40	工具径補正(TRC)の解除	(モーダル、デフォルト)
G41	形状左側へのTRCの選択	(モーダル)
G42	形状右側へのTRCの選択	(モーダル)

工具径補正は、G17、G18、またはG19で選択された平面で有効になります。

5.14 工具径補正方法(G138/G139/G237/G238/G239/G05)の選択

TRCの選択/解除には、5つのオプションがあります。

G138	TRCの直接選択/解除	(モーダル)
G139	TRCの間接選択/解除	(モーダル、デフォルト)
G237	TRCの垂直方向の選択/解除	(モーダル)
G238	TRCの内角コーナー選択	(モーダル)
G239	ブロックを使用しないTRCの直接選択/解除	(モーダル)
G05	TRCの接線選択/選択解除	(ノンモーダル)

5.15 工具径補正の遷移ブロックの選択(G25/G26)

G25	直線遷移	(モーダル、デフォルト)
G26	円弧遷移	(モーダル)

工具径補正が実行されている場合、外角コーナーはバイパスして接続されている必要があります。このために、工具径補正はいわゆる遷移ブロックを挿入します。G25およびG26では、直線ブロックの挿入と円弧ブロックの挿入を選択できます。

5.16 工具径補正時の送り速度の適用(G10/G11)

G10	一定の送り速度	(モーダル、デフォルト)
G11	最適化された送り速度	(モーダル)

G11が有効な場合、工具径補正によって挿入された遷移ブロックをより高速で移動します。円弧ブロックでは、工具接点でプログラムされた送り速度が有効になります(図4-47を参照)。

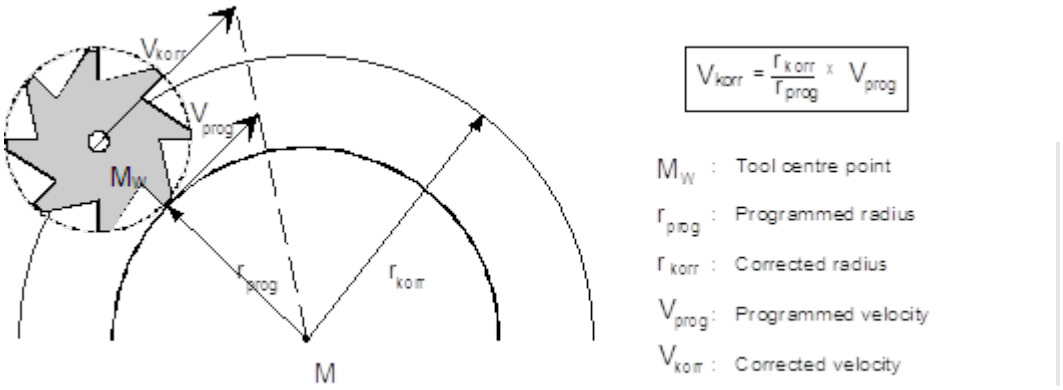


図 46: 図4-47: 補正された円弧補間での送り速度の適用

5.17 工具径補正時の形状マスキングの選択/解除(G141/G140)

G140	形状マスキングの解除	(モーダル、デフォルト)
G141	形状マスキングの選択	(モーダル)

大径工具での荒加工中に工具径補正が実行されると、工具直径よりも短い距離の形状要素がある場合には、形状に削り込んでしまう可能性があります。TRCの利用時にG140/G141の使用することで、形状に沿って正しく移動できるようになります。形状エレメントをマスキングすることにより、形状への削り込みが発生するケースに対して、形状の破損を防止することができます。

プログラム終了時に形状マスキングが有効な状態でも、エラーメッセージは出力されません。プログラム開始時には、自動的にデフォルトであるG140が有効になります。

注記

工具径補正がG40によって解除されても形状マスキングは選択されたままです。工具径補正が再度選択されると、形状マスキングも再度有効になります。

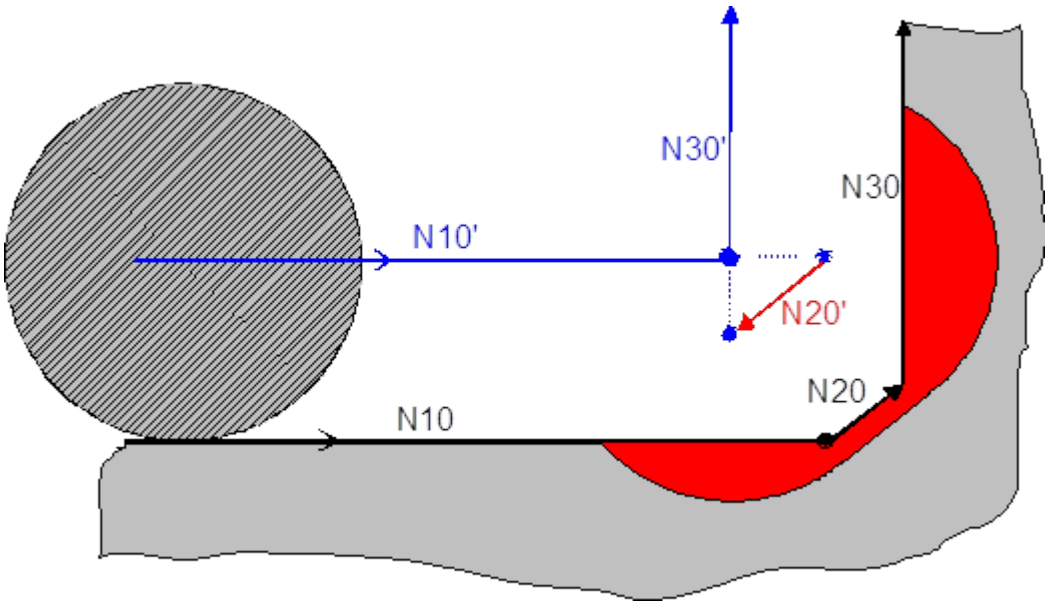


図 47: 図4-48: 形状への削り込みを回避するN20のマスキング

5.18 ゼロオフセットNPV (G53/G54/...G59)

本マニュアルでは、「NPV」はゼロオフセットを意味します。

G53	ゼロオフセットの解除*	(モーダル、デフォルト)*
G54	1番目のゼロオフセットの選択	(モーダル)
G55	2番目のゼロオフセットの選択	(モーダル)
G56	3番目のゼロオフセットの選択	(モーダル)
G57	4番目のゼロオフセットの選択	(モーダル)
G58	5番目のゼロオフセットの選択	(モーダル)
G59	6番目のゼロオフセットの選択	(モーダル)

* G53の設定データの意味は、P-ZERO-00001によって決まります。G53によりNPVが解除されるのか、NPVにG53の設定データが追加されるのかが変わります。

起動時に自動的に有効になるデフォルトのNPVは、P-ZERO-00002で指定することができます。

対応するゼロオフセットテーブル欄を使用することで、G54～G59のゼロオフセットを設定できます。ブロック内でG53、G54、～がプログラムされていると、ゼロオフセットはただちに有効となります。ただし、軸動作を指定しないと、移動動作は実行されません!

初期状態(G53)では、どのゼロオフセットも有効ではありません。

G91モード時のゼロオフセットの選択に注意してください。
N10 G54 X0 G91

のプログラミングは、X軸の相対的な移動量として0を指定しているため、X軸は移動しません。

したがって、**次の** 動作指令が**絶対**座標(G90)でプログラミングされた際に、ゼロオフセットが機能します!

5.18.1 高度なゼロオフセット変数

変数	意味	読み取り	書き込み
V.G.NP[j].ALL	全軸アドレスへのゼロオフセット	可	可
V.G.NP_AKT.V[i]	現在の(現在有効な)軸へのゼロオフセット	可	可
V.G.NP_AKT.ALL	現在の(現在有効な)全軸アドレスへのゼロオフセット	可	可
V.G.NP_DEFAULT	初期状態で有効なゼロオフセット	可	可

V.G.NP[j]およびV.G.NP_DEFAULT変数への書き込みアクセスは、リストが次に更新されるまで有効になります。一方、V.G.NP_AKTによる書き込みアクセスは、デコーダ内に一時的に格納されるゼロオフセットにのみ影響を与えます。以前にV.G.NP_AKTによって書き込まれたゼロオフセットが再度選択されると、リストから取得された元の値が有効になります。解除後もV.G.NP_AKTへの書き込みアクセスを維持する場合は、V.G.NP[j].ALLによる割り当てで現在のデータをリスト内に保存する必要があります。

プログラミング例

```
Saving of the currently active zero offset values:
V.G.NP[12].ALL=V.G.NP_AKT.ALL
```

5.18.2 オフセットの追加/削除

V.G.NP[j].ALL変数により、既存のゼロオフセットの値を足し合わせることにより、新しいゼロオフセットデータを新規に作成することができます。すべての軸のオフセットは、それぞれ個別に計算されます。ゼロオフセットの設定には、表記「+=」、「-=」、および「=」を使用できます。

プログラミング例1

G54 (NPV1)は、G55 (NPV2)およびG57 (NPV4)を足し合わせたものになります。

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

プログラミング例2

以下は同じ操作です。G54に値を*追加*する操作です。

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

または

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

注記

V.G.NP[j].ALL変数に対して、軸を指定したV.G.NP[j].V[i]変数や定数により値を割り当てることはできません!

プログラミング例

G54 (NPV1)は、G55 (NPV2)、G57 (NPV4)のオフセット、および補正値の組み合わせで定義されます。

誤:

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

正:

割り当ては2つのステップで行う必要があります。

```
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
```

5.18.3 現在のゼロオフセットへのアクセス

デコーダ内の現在有効なゼロオフセットには、V.G.NP_AKT.V[i]変数によってアクセスします。オペレータは、現在選択されているゼロオフセット(例えばインデックスなど)を認識している必要はありません。

プログラミング例

X軸の現在のゼロオフセットは200になります。

```
N10 V.G.NP_AKT.V.X = 200
```

現在のすべての軸のゼロオフセットは、G55 (NPV2)のオフセット分、値が追加されます。

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

または

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

現在のゼロオフセットの変更は、プログラムに対して影響を与えません。一方で、ゼロオフセットを追加で割り当てることにより、ユーザが他のプログラムで現在のゼロオフセットを使用できるようになります。

プログラミング例

現在のゼロオフセットはG54 (NPV1)に保存し、後で他のプログラム内で使用できます。

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

5.18.4 初期ゼロオフセット

プログラム開始後に有効になるNPVは、P-ZERO-00002で指定できます。

オペレータはNCプログラム内でV.G.NP_DEFAULT変数を使用し、初期ゼロオフセットにアクセスできます。変更は、プログラム内でグローバルに有効になります。

プログラミング例

このプログラム行以降、新しくG55 (NPV2)が初期ゼロオフセットとなります。つまり、次のプログラムの開始時にはG55が自動的に有効になります。

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

5.18.5 ゼロオフセットグループの作成

ゼロオフセットの作成規則により、ゼロオフセットグループを定義します。これは、特定のゼロオフセットの組み合わせによって表されることを意味します。ゼロオフセットグループは記号と文字列と組み合わせて、NCプログラム内で定義されるか、チャンネルパラメータリスト[1]-1内で初期設定として定義されます。記号と文字列の使用方法は、「記号文字列(マクロ) [▶ 505]」の章で説明されています。ゼロオフセットグループは、必ず3つのステップで有効にします。

プログラミング例

ステップ1:

NCプログラム内で、VERSCH_1というマクロ名でゼロオフセットグループを定義する例

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

チャンネルパラメータリスト[1]-1で定義する例

```
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
```

ステップ2:

ゼロオフセットの作成(新しくNPV1のデータへ割り当て)

```
N20 " VERSCH_1 "
```

ステップ3:

ゼロオフセットの選択(ここではNPV1、つまりG54)

```
N30 G54
```

5.18.6 拡張ゼロオフセット(G159)

G159 = *<index>*	(モーダル)

G159	拡張ゼロオフセット
<index>	ゼロオフセットデータのインデックス

NPVデータレコード[3]へのアクセスができます。G53～G58のデータレコードへのアクセスも可能です。拡張ゼロオフセットグループの最大数は、[6]-6.12で指定できます。

プログラミング例

```
G159 = 0        corresponds to G53
G159 = 1        corresponds to G54
G159 = 2        corresponds to G55
```

```
G159 = 3        corresponds to G56
G159 = 4        corresponds to G57
G159 = 5        corresponds to G58
G159 = 6        corresponds to G59
G159 = 7        access to data record with list index 7
:
G159 = 10       access to data record with list index 10
:
```

5.18.7 ゼロオフセットの軸ごとの有効化/無効化(G160)

G160 = *<index> <axis_name><flag>*	(モーダル)

G160 ゼロオフセットの軸ごとの有効化

<index> ゼロオフセットデータのインデックス

<axis_name><flag> 軸ごとのゼロオフセットの有効化のフラグ

フラグ	意味
0	軸のゼロオフセットが計算に含められます。
1	軸のゼロオフセットが計算に含められません。

各ゼロオフセットデータレコード*<index>*にパラメータP-ZERO-00004を事前に設定しておくと、計算時に各軸*<axis_name>*に対してオフセットを含めるか、または含めないか定義できます。つまり、個別の軸に対してオフセットを解除または設定できます。

この個別の軸に対するゼロオフセットの有効化は、NCプログラム内でG160を使用して変更できます。

プログラミング例

G55 (インデックス2のゼロオフセットデータレコード)を選択する前に、XおよびZ軸のオフセットを解除し、Y軸のオフセットを設定します。

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

これにより次のブロックでは、G55の中でも有効な軸オフセット(Y軸)のみが動作に含められます。

5.19 円弧定義の中心点指定(G161/G162)

G161	絶対値での円弧中心点	(モーダル)
G162	相対値での円弧中心点	(モーダル、デフォルト)

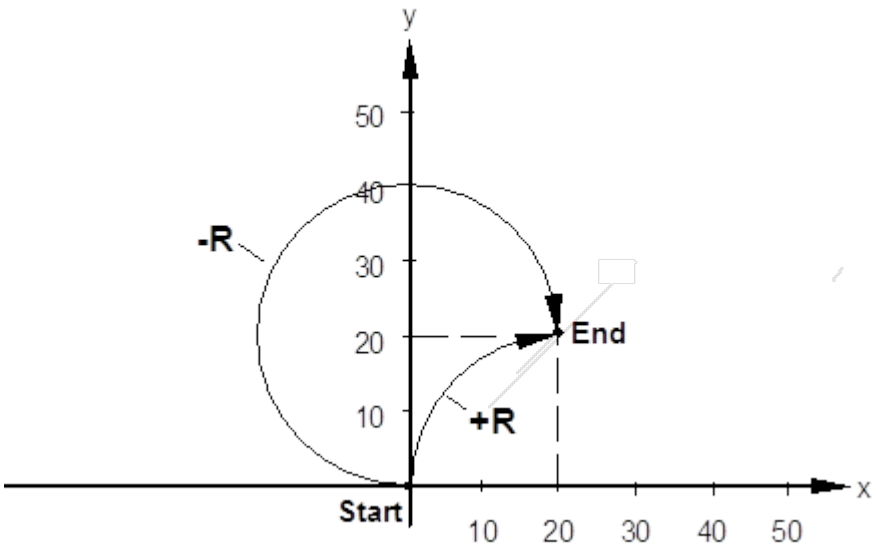
G162が有効な場合は、中心点がI、J、およびKによって円の起点に対して相対的に定義されます。G161が有効な場合は、I、J、およびKがプログラムされた座標系内の絶対座標として指定されます。

5.20 半径プログラミング(R、G163)

円弧中心点(I、J、およびK)が指定されていない場合は、半径プログラミングを行います。文字Rに続く値は、半径として解釈されます。NCブロック内で円弧プログラミングをすることで、この解釈は自動的に実行されます。半径ステートメントはモーダルであり、半径が同一の複数の円弧移動動作の場合は半径を繰り返し指定する必要はありません。

円弧の最大許容半径は、10^9 mmです。
ただし、円弧の終点が軸の最大移動範囲である$\pm 2.14*10^5$ mmを超えてはなりません。

半径が正の値である場合は最短の円弧、半径が負の値である場合は最長の円弧が決定します(以下の図を参照)。



インデックスを使用した半径プログラミングの場合(これを行うにはパラメータプログラミングでRワードが使用されていない必要があります。第9章「パラメータおよびパラメータ計算」を参照。)、インデックスには1以外は使用できません。

インデックスを用いて半径プログラミングを使用する場合は、インデックスを数式としてプログラムしてはいけません。

プログラミング例

```
1.)                N10 P2  = 1     3.)                   N10 R1 = 5 P2 = 1
    permissible:   N20 RP2 = 5          not permissible:  N20 P3 = RP2

2.)                N10 R1 = 5
    permissible:   N20 P2 = R1
```

プログラミング例

```
All these examples, produce semi-circles with radius 50.

1. N10  G90 G01 X0 Y0 F500
   N20  G02  X100  R50
   N30  G03  X200  R50

2. N10  G90 G01 X0 Y0 F500
   N20  G02  R=50             (In this block still no motion)
   N30  X100                  (Motion block)

3. N10  G90 G01 X0 Y0 F500
   N20  R1=50
   N30  G02  X100
   N40  G03  X200

4. N10  G90 G01 X0 Y0 F500
   N20  G02  X100  R1=50
   N30  G03  X200

The following programming results in an error message, since R1 is interpreted as radius with
value 1.
   N10  G90 G01 X0 Y0 F500
   N20  R1=50
   N30  G02  X100  R1
```

R = 「半径」またはR1 = 「半径」での円弧指定の代わりに、以下のGコードを使用して円弧の径を指定できます。

G163= *<半径>*	(モーダル)

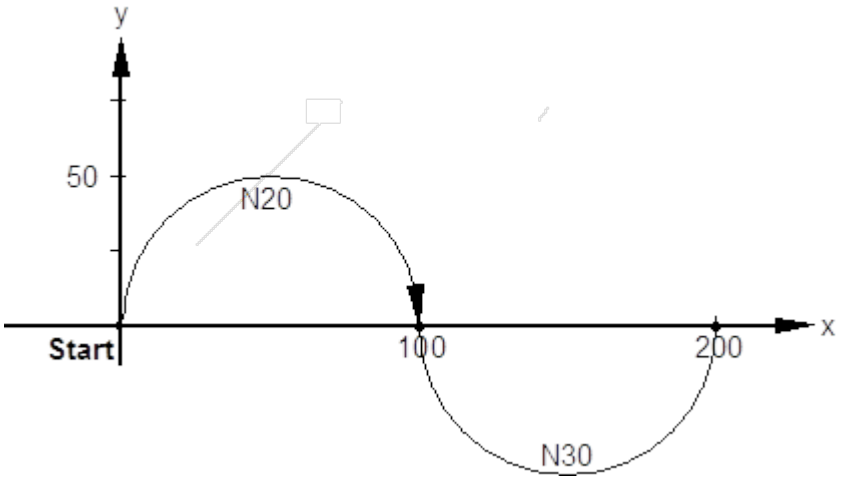
G163 円弧の半径指定

<半径> 円弧の半径値

円弧の半径指定により円弧を定義した場合は、I、J、Kを用いて新しく円弧が定義または解除されるまで有効となります。

プログラミング例

```
(N10: Movement to origin)
(N20: Semi circle cw with end point X100 under preset)
(of circle radius by G163 (radius defintion is modal))
(N30: Semi circle ccw with end point X200 and a modal radius,)
(which has been defined in N20)

%Radiusprogramming_G163

N10 G90 G01 X0 Y0 F1000
N20 G02 G163=50 X100
N30 G03  X200
N40 M30
```



注記

「R」、「R1」、または「G163」でプログラムされた円弧の起点および終点が同一であると、エラーメッセージが出力されます。このため、一周円を移動する場合はI/J/Kを使用してプログラムする必要があります。

5.21 中心点オフセット制御(G164/G165)

G164	中心点補正OFF	(モーダル)
G165	中心点補正ON	(モーダル、初期設定)

G165が有効な場合、Iステートメント、Jステートメント、およびKステートメントによってプログラムされた円弧が、指定された円弧方向(G02/G03)、指定された開始位置(前のブロックの終了位置)、および指定された終了位置(円弧ブロック内の座標)で円弧が補間されるように補正されます。これにより、I、J、およびKによってプログラムされた中心点が移動する可能性があります! より正確に中心点を示すことで、中心点のオフセットは小さくなります。

半径Rの定義を使用して円弧をプログラムすると中心点が必ず正確に計算されるようになるため、中心点補正は無効になります。

プログラムされた中心点から移動した中心点までの偏差について、2つのリミット値P-CHAN-00059およびP-CHAN-00060がモニタリングされます。これらのリミット値を超えると、エラーメッセージが出力されます。

mittelpkt_diff: 10^{-4} mmの許容オフセット

mittelpkt_faktor: 0.1 %のパーセンテージオフセット

中心点オフセットΔmが絶対値「mittelpkt_diff」よりも大きいかどうかが検証されます。

Δm > mittelpkt_diff ?

また、中心点オフセットΔmが「mittelpkt_faktor/1000」と補正された半径「radius」の積よりも大きいかどうかが検証されます。

Δm > mittelpkt_faktor/1000 * radius ?

つまり、Δmの上限は計算された半径に依存します。中心点オフセットΔmと計算された半径「radius」の関係は、図4-49に記載されています。

例:

「mittelpkt_faktor」 = 5は、プログラムされた中心点座標と移動した中心点座標が、補正された円弧の半径の最大で0.5 %となる必要があることを意味します。

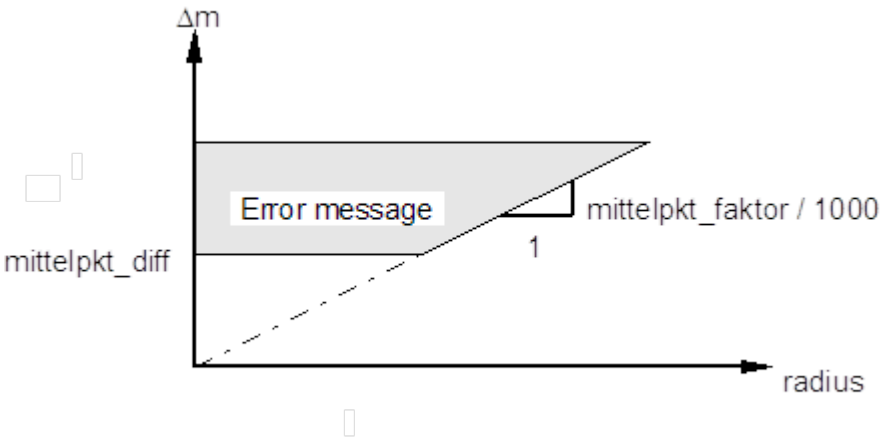


図 48: 図4-49: 中心点オフセットΔmと計算された半径「radius」の関係

このため、プログラムされた中心点座標は移動した円弧の中心点の周囲の外接円上に存在する必要があります。外接円の半径は、許容される中心点オフセットΔmに対応します。これは2つのパラメータ「mittelpkt_diff」および「mittelpkt_faktor」で設定可能です(図4-50)。

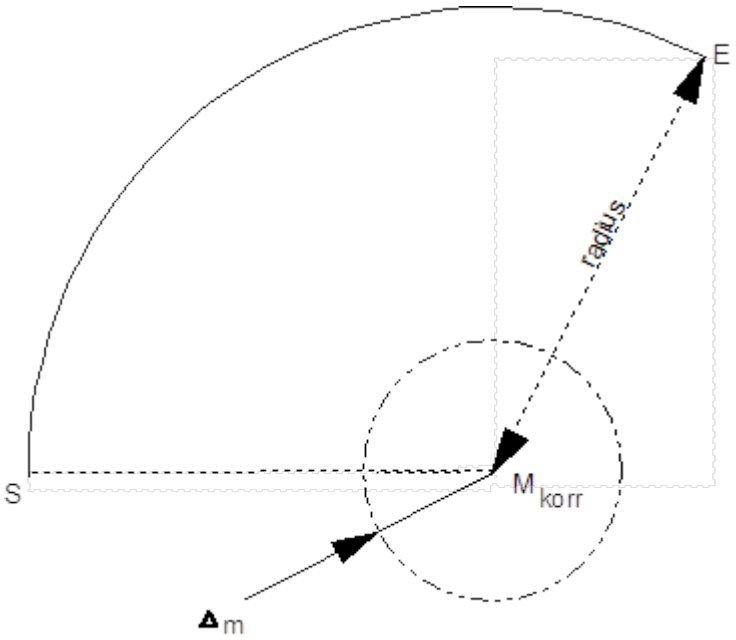


図 49: 図4-50: 許容されるプログラムされた中心点の範囲

ポログラムされた円弧の中心点の補正方法

NCプログラム内で起点S、終点E、および中心点Mによって指定される円弧は、形状が冗長です。この問題は、円弧の中心点座標と円弧の半径が以下のアルゴリズムで補正されると解決します。

以下の2つの次元ベクトルが必要です。

- 絶対終点ベクトル(v_{ende})
- 相対(起点ベクトルに対して)中心点ベクトル(r_{start})
- 絶対起点ベクトル(v_{start})

図4-51のように、選択される方法は、絶対値の相加平均が中心点から起点または終点までのベクトル、または逆方向のベクトルの両方から取得できるという事実に基づいています。このため、新しい中心点はこの平均長さの2つのベクトルから割り出されます。

形状的には、これはプログラムされた中心点が楕円形の経路を楕円と楕円の短軸との交点に移動することを意味します。時間がかかるケース分け(理論上は2つの中心点が存在)を避けるため、以下のアルゴリズムが実行されます。

円弧の弦のベクトルを形成します。

$$v_{sehne} = v_{ende} - v_{start}$$

プログラムされた円弧の中心点のローカルベクトルを形成します。

$$v_{mittel} = v_{start} + r_{start}$$

円弧の中心点から円弧の終点までの相対ベクトルを取得します。

$$r_{ende} = v_{ende} - v_{mittel}$$

これにより、半径の決定が可能

$$radius = 0.5 * (|r_{start}| + |r_{ende}|)$$

弦のベクトル上に開始半径ベクトルを投影(スカラー積)することで、弦の中心点にベクトルが直交する点を取得できます。

$$v_{fusz} = (v_{sehne} * r_{start}) / |v_{sehne}|^2$$

これらのデータにより、中心点に直交するベクトルを決定できます。

$$v_{lot} = r_{start} - v_{fusz}$$

これで、プログラムされた弦のベクトルを使用して、このベクトルを新しい半径ベクトルの投影された長さに正規化できます。

$$v_{lot\_neu} = v_{lot} * sqrt [radius^2 - (|0.5 * v_{sehne}|)^2] / |v_{lot}|$$

新しい相対半径ベクトルは、以下のようになります。

$$r_{start\_neu} = 0.5 * v_{sehne} + v_{lot\_neu}$$

2つの円を順に移動し、半径が1番目の円弧に対してのみプログラムされていて、かつ中心点補正を実行する必要がある場合、2番目の円弧は1番目の円弧の補正された半径ではなく、プログラムされた半径に基づいて計算されます。

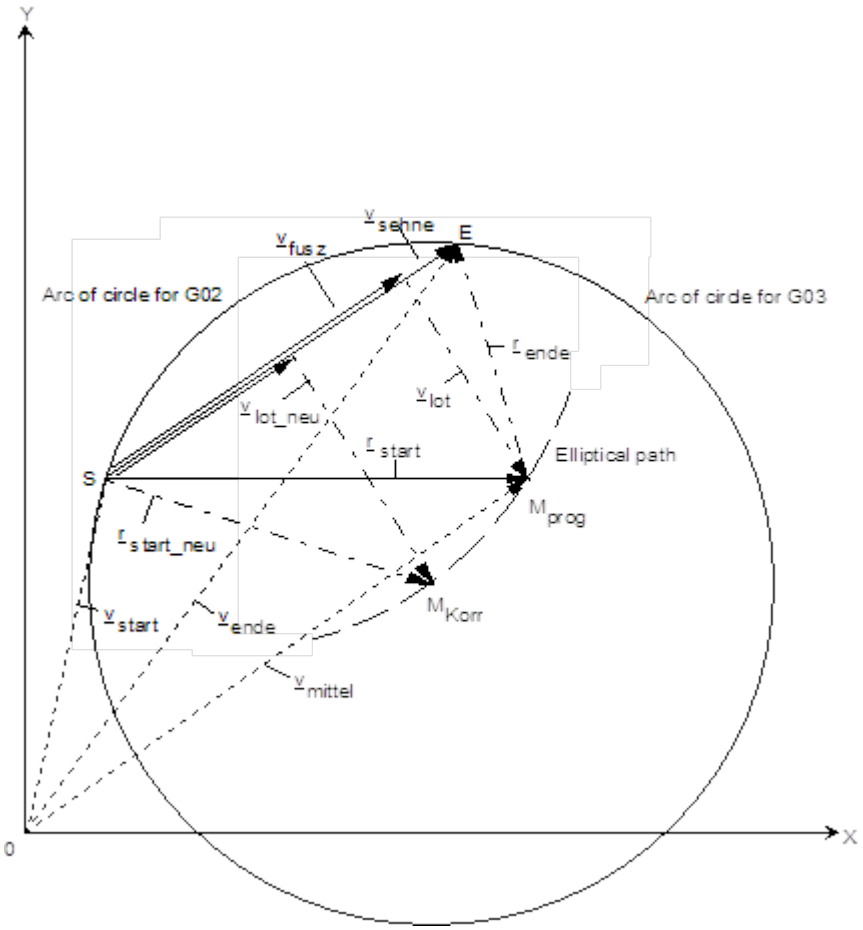


図 50: 図4-51: プログラムされた円弧の半径の補正

5.21.1 特殊ファンクション: G164と組み合わせた円弧径補正

特定の状況では、円弧の中心点補正(G165)によりプログラムされた円弧の中心点、および円弧の位置が不適切に移動することがあります。このような不適切な状況は、円弧の起点と目標点が近すぎ、円弧とプログラムされた一周円とほぼ同じである場合に発生します。このような目標点が指定され、円弧の中心点が補正された、またはされていない(G165/G164)円弧のプログラム例を以下に記載します。円弧の中心点補正が有効な状態でポストプロセッサの分解能誤差をシミュレーションするために、それぞれのケースで円弧の目標点を起点と比較してXおよびY方向に0.1 µm移動します。G165を使用した円弧は起点を中心に回転し、補正された円弧の中心点M_Kはプログラムされた中心点Mと比較すると大幅に移動します。

```
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
```

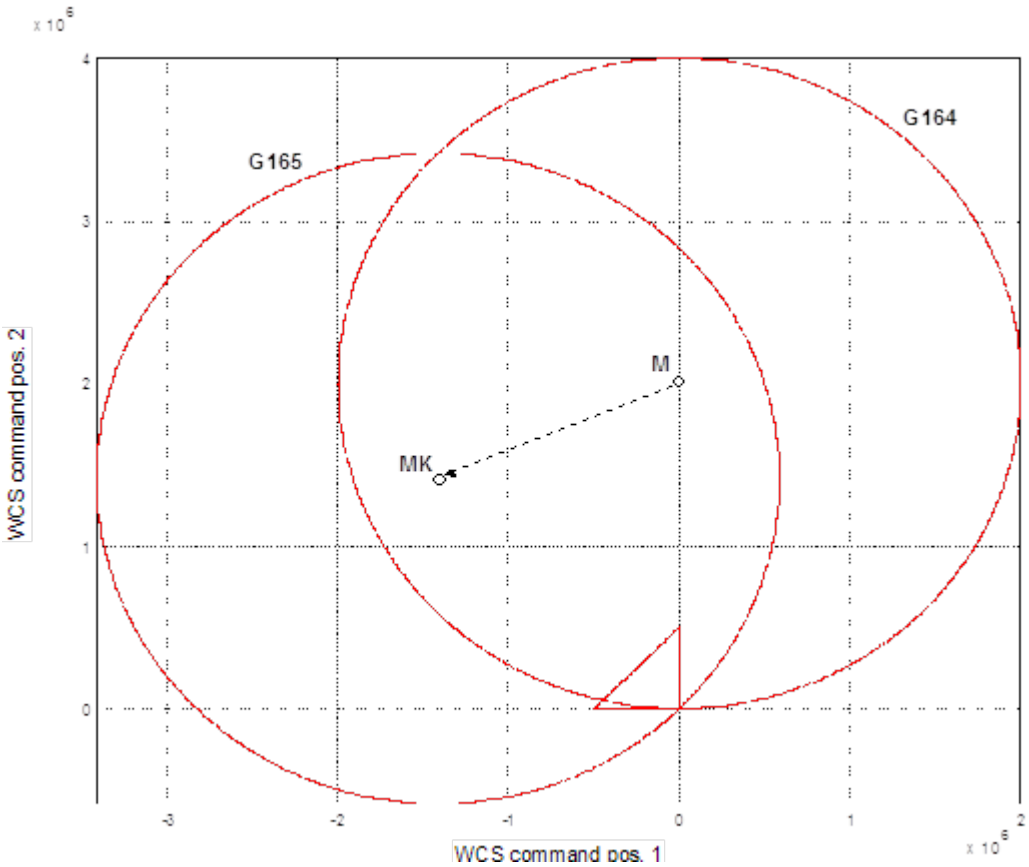


図 51: 図4-52: G165の場合の円弧の中心点移動

このような場合、一般には円弧の中心点補正を無効にし、チャンネルパラメータ「max_radius_diff_circle」> 0および「max_proz_radius_-diff_circle」を設定すると、より良い結果が得られます。プログラムされた円弧の中心点はこのファンクションでは変更されません。また、円弧の半径の差異は円弧の角度によって開始半径から目的の半径まで直線に移行します。

円弧の半径の偏差が分解能精度順に存在する場合、一般には円弧の歪みや動的効果を無視できます。

一周円の場合は、ここで円弧の起点および目標点が同一である必要もあります!

5.22 フィードフォワード制御(G135/G136/G137)

G135	フィードフォワード制御の選択	(モーダル)
G136 <*axis_name*><*expr*> { <*axis_name*><*expr*> }	ウエイトの指定	(モーダル)
G137	フィードフォワード制御の解除	(モーダル)

速度および加速度のフィードフォワード制御を使用することで、動作経路の誤差を低減させることができます。

G135を用いたプログラミングにより、軸グループ全体に対して機能を有効化できます。G136により、軸ごとにフィードフォワード制御のウエイトをパーセンテージで設定することができます。これはすべての軸に対して100%までに制限されています。

G137は軸グループ全体に対して、フィードフォワード制御を無効にします。フィードフォワード制御を行わない軸に対しては、G135を使用して全体を選択した後で、G136を用いて0%のウエイトをプログラムする必要があります。フィードフォワード制御の選択とウエイトを、単一のブロック内に含めることも可能です。

各プログラムの開始時には、フィードフォワード制御がオフとなっており、ウエイトは100%に設定されています。

NCプログラム内でフィードフォワード制御のオン/オフを切り替えた場合でも、ウエイトはG136で設定された値が維持されます。G136がプログラムされていない場合は、ウエイトは100%となります。

注記

軸の交換指令がなされたすべての軸に対しては、G136で設定したウエイトは100%にリセットされます。

プログラミング例

```
G135    (Selection of feedforward control: weighting for all axes 100%)
G136 X80 Y95 Z0 (Weighting; here Z-axis receives no feedforward control)
```

5.23 最大速度ウエイト(G128)

G128 = <*expr*>	(モーダル)

G128	軸グループに対する最大速度ウエイト
<*expr*>	**すべて**の軸に対するパーセントでのウエイト指定

ファンクションG128を使用すると、最大速度を変更することが可能です。

最大速度の変更は、パーセント単位での指定により対応する速度値を変更することで可能です。

プログラムの中でG128を使用すると、プログラムされていないすべての軸、または今後プログラムされる軸は100%に設定されます。その後、実際に軸が選択されると、以前のプログラムとは関係なく、100%として動作します。つまり、形状データ処理時には必ず、デフォルトのP-AXIS-00212で設定したパーセント値でウエイトされています。

50%が正しく2回プログラムされると、設定は25%ではなく、50%と設定されます。

最大速度ウエイトでの送り設定は、原点復帰、手動モード、または独立軸などの単一軸動作には効果がありません。

注記

G128 > 100%の場合、最大速度は最大値パラメータのP-AXIS-00212に制限されます。

G128 = 0の場合、最大速度は最小値1µm/sに制限されます。

プログラミング例

```
N10 G128 = 10           (Axis group-specific weighting of vb_max)
                        (vb_max of all axes to 10%)

N20 G01 X100 Y150 F1000     (Linear interp., movement with 10% of vb_max)


Special feature:
N50 G128 = 10 X10 Y20   (In the same block with G128 also axis positions)
                        (can be programmed)
```

5.24 早送り速度ウエイト(G129)

G129 = <*expr*>	(モーダル)

G129	軸グループに対する早送り速度ウエイト
<*expr*>	**すべて**の軸に対するパーセントでのウエイト指定

ファンクションG129を使用すると、早送り速度G00を変更することが可能です。

早送り速度の変更は、パーセント単位での指定により対応する速度値を変更することで可能です。

プログラムの中でG129を使用すると、プログラムされていないすべての軸、または今後プログラムされる軸は100%に設定されます。その後、実際に軸が選択されると、以前のプログラムとは関係なく、100%に設定されます。つまり、形状データ処理には必ず、デフォルトのP-AXIS-00209で設定したパーセント値でウエイトされています。

50%が正しく2回プログラムされると、設定は25%ではなく、50%と設定されます。

早送りウエイトは、早送りNCブロック(G00)に対してのみ効果があります。原点復帰、手動モード、または独立軸などの単一軸動作には効果がありません。

注記

G129 > 100%の場合、最大速度は最大値パラメータのP-AXIS-00212に制限されます。

G129 = 0の場合、最大速度は最小値1µm/sに制限されます。

プログラミング例

```
N10 G129 = 70           (Axis group-specific rapid traverse weighting)
                        (Rapid traverse velocity of all axes to 70%)

N20 G00 X100 Y150       (Linear interp., rapid traverse movement with 70%)


Special feature:
N50 G129 = 70 X10 Y20   (In the same block with G129 also axis positions)
                        (can be programmed)
```

5.25 加速度プロファイルのパラメータ設定

5.25.1 加速度ウエイト(G130/G131/G231)

G130 <*axis_name*><*expr*> { <*axis_name*><*expr*> }	(モーダル)
G131 = <*expr*>	(モーダル)
G231 = <*expr*>	(モーダル)

G130	軸ごとの加速度ウエイト
<axis_name><expr>	指定の軸に対するパーセントでのウエイト指定
G131	G01、G02、G03使用時の軸グループ全体に対する加速度ウエイト
<expr>	すべての軸に対するパーセントでのウエイト指定
G231	G00使用時の軸グループ全体に対する加速度ウエイト
<expr>	すべての軸に対するパーセントでのウエイト指定

ファンクションG130/G131/G231を使用すると、加速度ランプを変更できます。

加速度の変更は、パーセント単位での指定により対応する加速度値を変更することで可能です。加加速度のプロファイルの設定は、軸パラメータP-AXIS-00001およびP-AXIS-00002で可能です。

プログラムの中でG130/G131/G231を使用すると、プログラムされていないすべての軸、または今後プログラムされる軸は、100%に設定されます。その後、実際に軸が選択されると、以前のプログラムとは関係なく、100%に設定されます。つまり、形状データ処理には必ず、初期値[2]-1 および [2]-2で設定したパーセント値でウエイトが設定されます。

50%が正しく2回プログラムされると、設定は25%ではなく、50%で設定されます。100%以上については、最大軸加速度P-AXIS-00008までのウエイト設定が可能です。

注記

軸交換指令がなされたすべての軸に対しては、G130でのウエイト設定は100%にリセットされます。

加速度ウエイトは、送り速度による補間動作中(G01、G02、G03)のみ効果があります。原点復帰、手動モード、または独立軸などの単一軸動作では**効果がありません**。

プログラミング例

```
N10 G130 X70          (Axis-specific acceleration weighting:)
                      (Acceleration of X-axis is restricted to 70%)
N20 G01 F1000 X100    (Linear interpolation)
N30 G130 Y60          (Acceleration of  Y-axis is restricted to 60%)
                      (Acceleration of X-axis remains set to 70%)
N40 Y100              (Linear interpolation)
N50 G131 = 100        (Axis group-specific acceleration weighting:)
                      (G01,G02,G03 acceleration of all axes to 100%)
N60 G231 = 80         (Axis group-specific acceleration weighting:)
                      (G0 acceleration of all axes to 80%)
N70 G00 X200          (Rapid feed)

Special feature:
N50 G131 = 100 X10 Y20  (In the same block with G131/G231 also axis)
                        (positions can be programmed)
```

5.25.2 ランプ時間ウエイト(G132/G133/G134/G233)

G132 <*axis_name*><*expr*> { <*axis_name*><*expr*> }	(モーダル)
G133 = <*expr*>	(モーダル)
G134 = <*expr*>	(モーダル)
G233 = <*expr*>	(モーダル)

G132	軸ごとのランプ時間ウエイト
<*axis_name*><*expr*>	指定の軸に対するパーセント指定でのウエイト
G133	G01、G02、G03使用時の軸グループ全体へのランプ時間ウエイト
<expr>	すべての軸に対するパーセント指定でのウエイト
G134	軸グループ全体への形状加工時のランプ時間ウエイト
<expr>	すべての軸に対するパーセント指定でのウエイト
G233	G0使用時の軸グループ全体へのランプ時間ウエイト
<expr>	すべての軸に対するパーセント指定でのウエイト

ファンクションG132/G133/G233を使用すると、[2]-1にて説明されている非直線的な立ち上がりに対して、軸加速度のランプ時間を変更できます。(直線的な立ち上がりに対する軸加速度のランプは、ステップ的な形になります(図11-5を参照))。

ファンクションG134を使用すると、P-AXIS-00199での非直線的な立ち上がりにおいて、形状的なところに対してもランプ時間を変更できます。

このランプ時間の変更は、パーセント指定にて対応するランプ時間を変更することで可能です。

プログラムの中でG132/G133/G134/G233を使用すると、プログラムされていないすべての軸、または今後プログラムされる軸は、100%に設定されます。その後、実際に軸が選択されると、それまでのプログラミングに関係なく、100%に設定されます。つまり、形状データ処理には必ず、デフォルトで設定されているパーセント値がウエイトされます。これにより、50%を2回プログラムしても、25%ではなく50%に設定されます。

注記

軸交換指令がなされたすべての軸に対して、G132のウエイトは100%にリセットされます。

ランプ時間ウエイトは、送り速度による補間動作中(G01、G02、G03)のみ効果があります。原点復帰、手動モード、または独立軸などの単一軸のランプには効果がありません。

プログラミング例

```
N10 G132 X200        (Axis-specific ramp time weighting:)
                     (Ramping time of X-axis is increased by 200 %)
N20 G01 F1000 X100   (Linear interpolation)
N30 G132 Y50         (Ramping time of Y-axis is decreased by 50%)
                     (Ramping time of X-axis remains at 200%)
N40 Y100             (Linear interpolation)
N50 G133 = 100       (Axis group-specific ramp time weighting:)
                     (G01,G02,G03 ramping time of all axes to 100%)
N60 G134 = 50        (Axis group-specific acceleration time weighting:)
                     (Geometrical ramping time of all axes to 50%)
:
N70 G233 = 80          (Axis group-specific ramp time weighting:)
                       (G00 ramping time of all axes to 80%)
N80 G00 X200           (Rapid feed)

Special feature:
N50 G133 = 100 X10 Y20  (In the same block with G133/G233 also axis positions)
                        (can be programmed)
N60 G134 = 50 X10 Y20   (In the same block with G134 also axis positions)
                        (can be programmed)
```

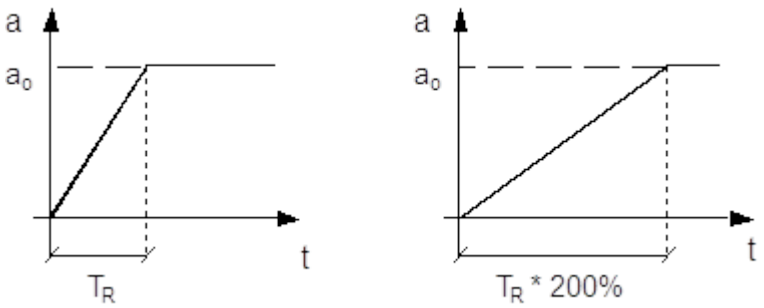


図 52: 図4-53: G132/G133/G233によるランプ時間ウエイトの例

以下に、G134を使用した場合を図示します。円弧移動(P1 --> P2)中にフィードが増加する(v1 --> v2)際における、経路に直交する加速度(遠心加速度a1--> a2)への影響を示しています。

G134のランプ時間を増加することで遠心加速度の変化を低減すると、加速がより滑らかになり、終端加速度であるa2に到達するのはより先の点P3においてとなります。

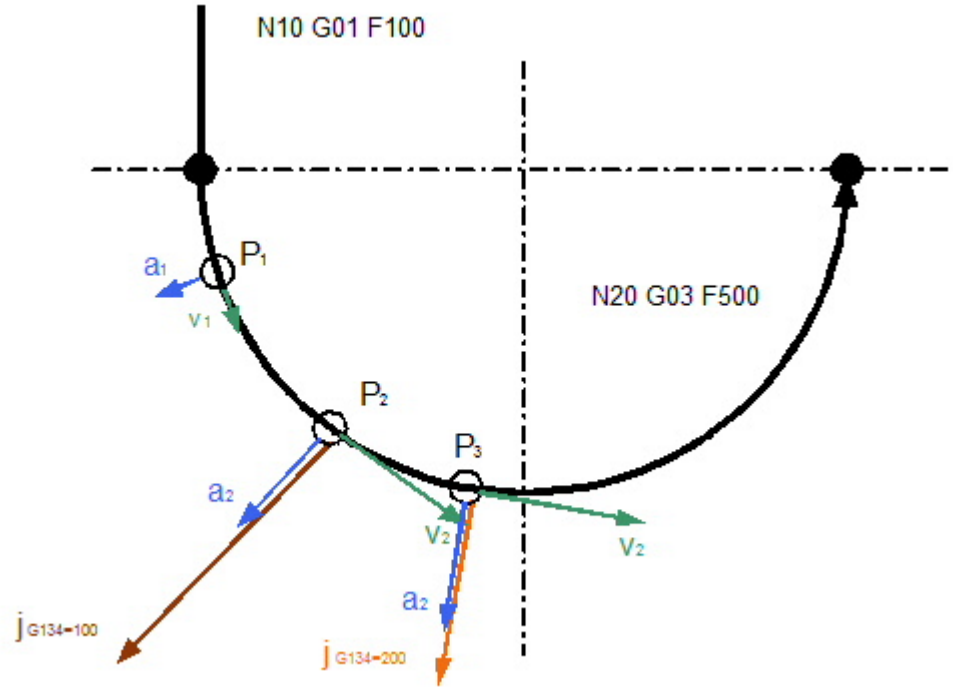


図 53: 図4-54: 円弧補間でのG134によるランプ時間ウエイトの例

5.26 加工時間または送り速度(G93/G94/G95/G194)

G93	加工時間 (秒)	(モーダル)
G94	送り速度 (mm/分)	(モーダル、初期設定)
G95	送り速度 (mm/回転)	(モーダル)
G194	軸送り最大送り速度にウエイトを考慮した送り速度計算	(モーダル)

GファンクションG93、G94、G95、およびG194により、Fワードでの動作をオプションにより変更できます。

G94における通常の単位:

- 回転軸のFワード (度/分)
- 直線軸のFワード (mm/分)

G194とFワードを合わせて使用することで、最大許容送り速度のパーセント指定でウエイトを設定することができます。軌跡上での最大許容送り速度は、P-AXIS-00212によって軸ごとに決まります。その後は、ウエイトを考慮した最大速度で移動します。ウエイトは100%以下の値のみ使用可能です。

プログラミング例

```
N10 G90 F1000 X100      (Feed 1000 mm/min (G94 Default))
N20 G194 F90            (Weighting 90% of max. axis feed velocity)
Nxx X200                (Feed e.g. 9000 mm/min with vb_max=10000 mm/min)
N80 G94 X50             (Feed 1000 mm/min valid from N10)
Nxx X.. Y.. Z..         (Interpolation)
N120 G93 F20            (Machining time 20 s)
Nxx  X.. Y.. Z..        (Interpolation)
N160 G94 F1500 X150     (Feed 1500 mm/min)
Nxx  X.. Y.. Z..        (Interpolation)
N200 M30
```

同じ軌道経路内に直線軸および回転軸が存在し動作する場合には、G93が選択されていると、両方の軸タイプでFワードの大きさは同じになります。

パラメータ値およびブロックでの動作形状が原因で加工時間を超過すると、エラーメッセージが出力されます。

ファンクションG95については、「1回転当たりの送り速度(G95) [▶ 460]」に記載されています。

5.27 面取りおよび丸み付けの挿入(G301/G302)

G301	面取りの挿入	(どちらのファンクションも
		2つの移動ブロック間で
G302	丸み付けの挿入	有効になります)

「丸み付けの挿入」とは、特定の半径の2つの隣接する形状要素に対する正接方向に交わる円弧の挿入です。「面取り」では、隣接する形状要素に対して傾斜角が同一な直線の挿入が行われます。

丸み付けの挿入にはファンクションG302、面取りの挿入にはファンクションG301が用意されています。これらのファンクションはノンモーダルで有効であり、正確に1つの形状要素(円弧または直線)の挿入を生成します。G302ブロックまたはG301ブロックは、グループ「a」の有効なGファンクション、つまりG05以外のG00、G01、G02/G03でのみブロック間に書き込むことができます。同一のNCブロック内にプログラムされたIワードは、挿入要素の半径または面取り幅を定義します。このIワードは半径または面取り幅が同一の以降のG302/G301でも有効のまま維持されるため、Iワードを再度プログラムする必要はありません。G301/G302の最初のプログラミング時に、ゼロ以外のIワードをNCブロック内でプログラムしないとエラーメッセージが出力されます(このエラーによりデコードが中止されます)。

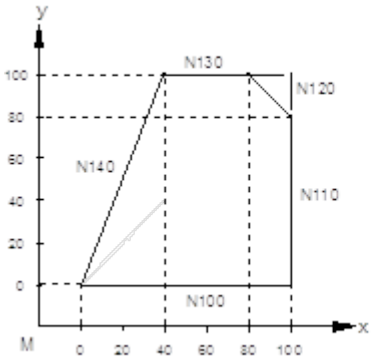
G301/G302での送り速度の有効性:

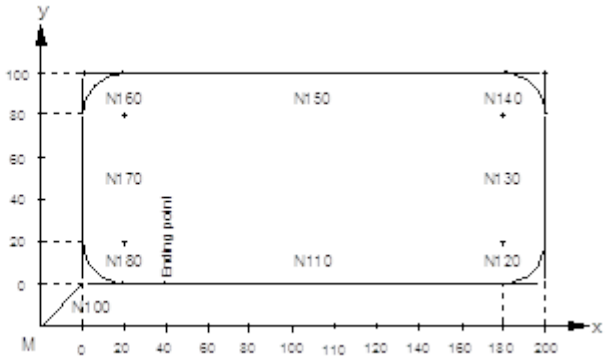
- G00が続く場合、円弧も最大許容速度で実行されます。
- G01/G02/G03が続き場合、プログラムされた送り速度が有効になります。
- G301/G302を含むブロックでは、送り速度も指定できます。
- G11およびG41/G42が有効な場合、送り速度も一致します。

プログラミング例

面取り: 90度のコーナーおよび2本の直線(面取りでは、20×45度、I=20が指定されます)

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





丸み付け: コーナー半径20 mm、長さ200 mm、幅100 mmの長方形のポケット

```
N100  G00  G42  X0    Y0    F200
N110  G01  G11  X200
N120  G302     I20         F150
N130                 Y100  F200
N140  G302                 F150
N150           X0          F200
N160  G302                 F150
N170                 Y0    F200
N180  G302                 F150
N190  G10      X40         F200
```

面取りまたは丸み付けは、必ず2番目の動作ブロックがプログラムされている平面に追加されます。

例A:

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

例B:

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

例AおよびBでは、YZ平面の形状が同じになります。

5.27.1 面取りの挿入

G301では、Iワードはプログラムされた形状のコーナー点と、挿入された直線の対応する交差点間の距離を形状要素で定義します。一方または両方の形状要素が円弧である場合、距離は弦の長さとなります。

挿入半径または面取りのサイズにより、一方または両方の形状要素の方向が逆転する場合、エラーメッセージが出力されます。

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

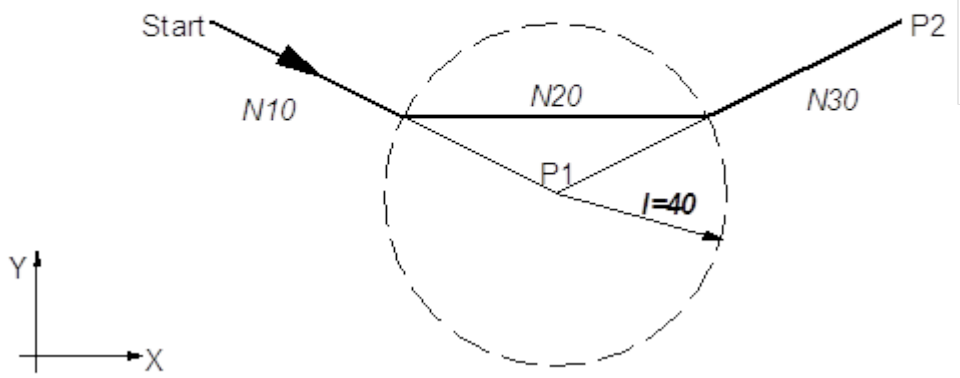


図 54: 図4-55: 2つの直線間への面取りの挿入

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

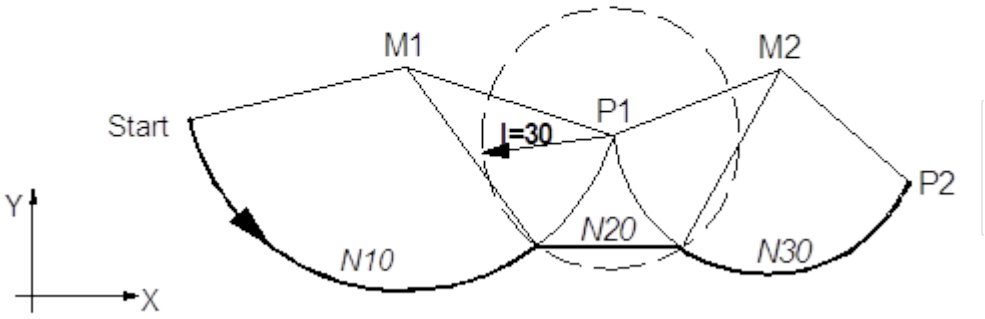


図 55: 図4-56: 2つの円弧間への面取りの挿入

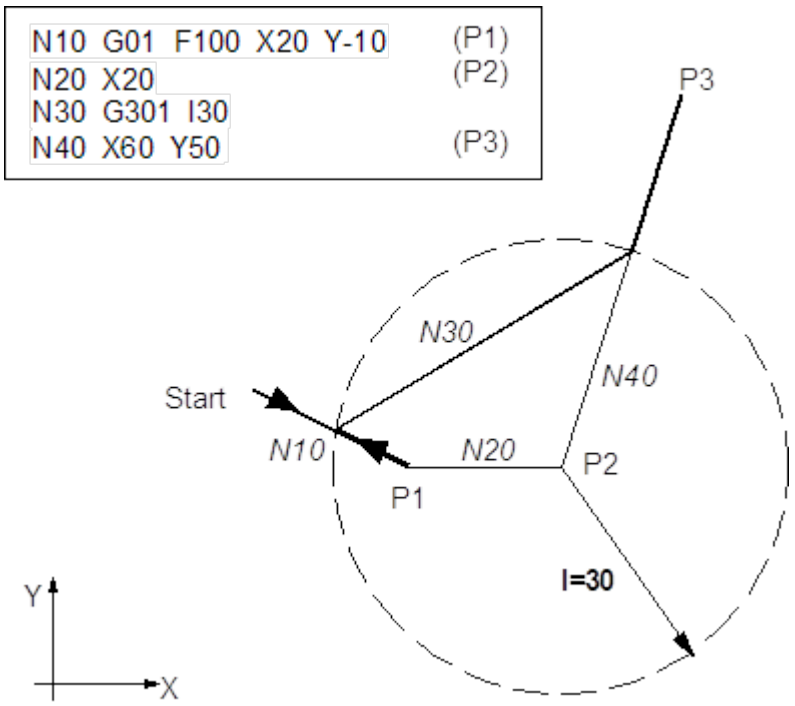


図 56: 図4-57: 方向の逆転によるエラー

5.27.2 丸み付けの挿入

G302では、Iワードは挿入された円弧の半径を定義します。円弧の中心点は、プログラムされた経路までの等距離線の交点です(間隔I)。等距離線の位置は、プログラムされた形状を丸み付け時に可能な限り遠くに配置する必要がある場合は固定されます。

正接ブロック遷移では、規則上、形状半径を挿入できません。

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

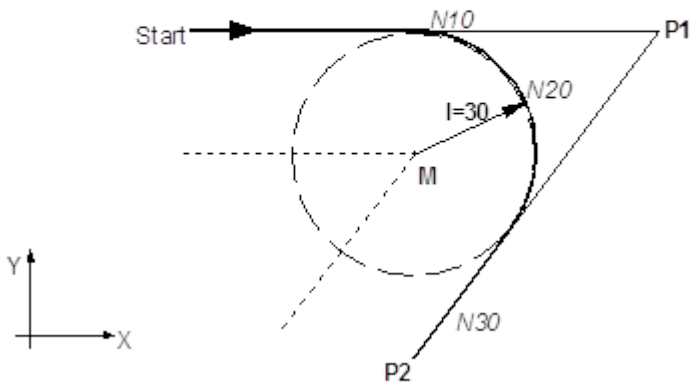


図 57: 図4-58: 2つの直線間への円の挿入

```
N10 G02 G91 F100 X80 I40      (P1)
N20 G302 I10
N30 G02 X50 I25 J-15          (P2)
```

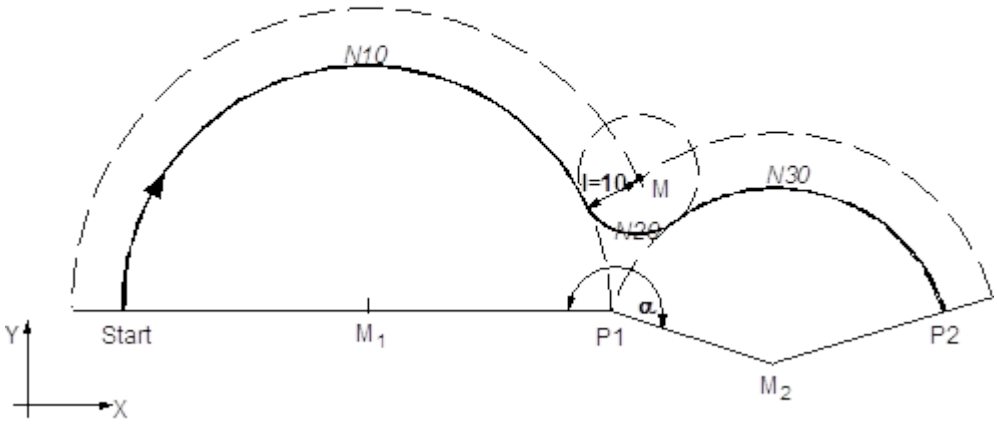


図 58: 図4-59: 2つの円弧間への円の挿入(角度α ≥ 180°)

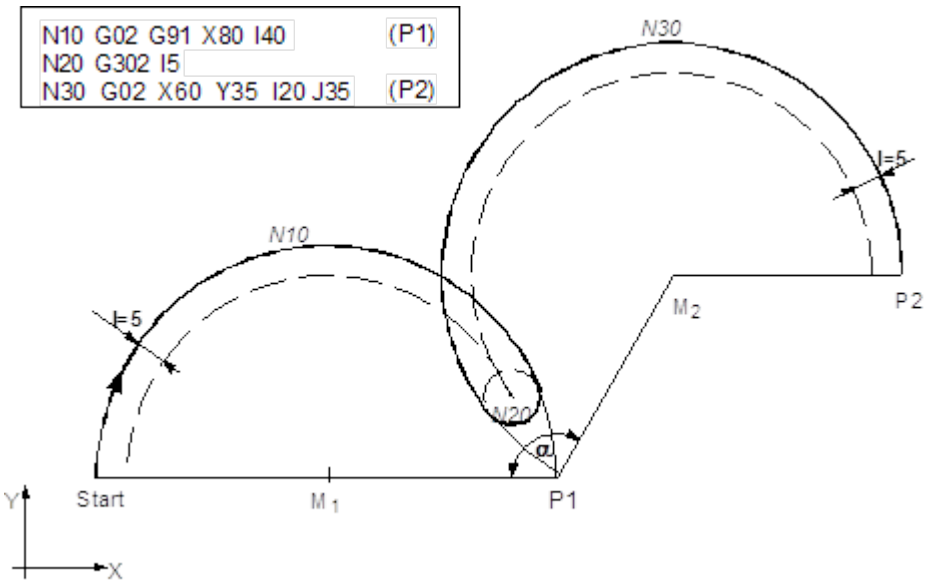


図 59: 図4-60: 2つの円弧間への円の挿入(角度α < 180°)

5.28 手動モード

手動モード(HB)により、手動操作の物理要素(ハンドホイール、インチングキー、ジョイスティック)によって軸を個別に外部制御できるようになります。補間実行中、つまりNCプログラム実行中に、オペレータが設定値を経路に追加できます。以下の手動操作モードが用意されています。

ハンドホイールファンクション: 任意の経路および速度。
連続ジョグモード: 任意の経路、パラメータ設定可能な速度。
インクリメンタルジョグモード: 事前指定された経路、パラメータ設定可能な速度。
中断可能なジョグモード: インクリメンタルジョグモードに準じる。ただし、移動の中断が可能。

これらの手動操作モードは、操作パネルから有効にできます。分解能、速度、ジョギング距離といった対応するパラメータは、適切なNCコマンド(「#」コマンド、「G」コマンド)によってプログラムします。手動モードにより、手動操作軸、手動操作要素、およびパラメータを集中管理できます。

使用中の手動操作モード、および物理手動操作要素によって制御されている軸は、いつでも変更可能です。物理手動操作要素により、複数の軸を同時に複数のNCチャンネル内で制御できます。1つの軸は単一の手動操作要素でのみ制御できます。また、複数の手動操作モードで同時に実行することはできません。

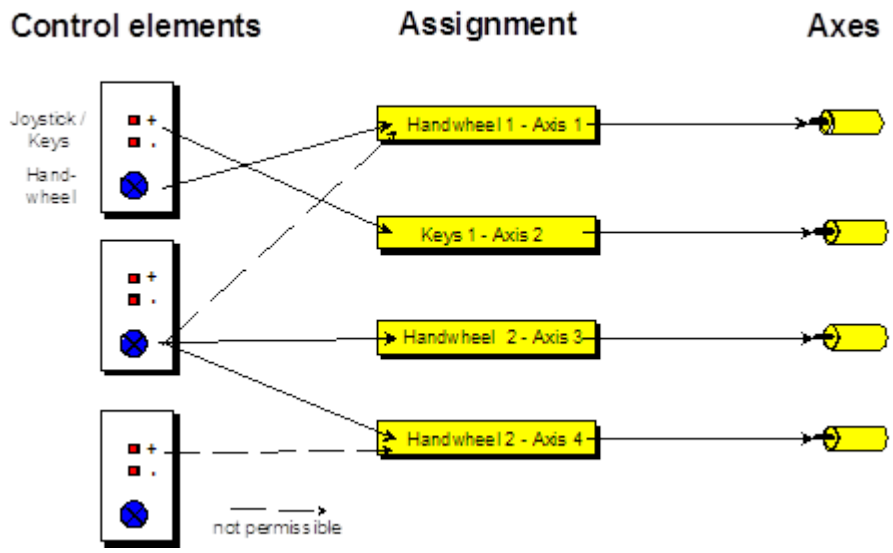


図 60: 図4-61: 手動操作とその設備

5.28.1 並行補間の選択/解除(G201/G202)

インターポレータと手動操作モード間の内部インターフェイスが、プログラムされた軸に対して有効になります。スピンドルはプログラムできません。

バージョンV2.11.2010.02以降では、軸X.. Y.. のプログラミングがコマンド#ACHSE [...]に代わります。互換性の理由から、このコマンドは当分の間使用できますが、新しいNCプログラムでは使用しないでください。

G201では、複数の特定の軸に対して手動操作が選択されます。その後、この軸に対してはG202によって解除されるまで、並行補間による手動動作が有効になります。

G201	*<axis_name><dummy_expr>* {*<axis_name><dummy_expr>*}	(モーダル)

<axis_name><dummy_expr>	複数の特定の軸に対する手動操作の選択。軸指定子以降の値は意味がありませんが、構文上の理由から必要です。選択には軸名が必要です。

G202では、すべての軸、または複数の特定の軸に対して手動操作が解除されます。

G202	**すべての**軸に対する手動操作の解除	(モーダル、初期設定)

または複数の特定の軸に対して

G202 *<axis_name><dummy_expr>* {*<axis_name><dummy_expr>*}

<axis_name><dummy_expr>	複数の特定の軸に対する手動操作の解除。軸指定子以降の値は意味がありませんが、構文上の理由から必要です。

プログラミング例

```
.....
G00 X100 Y100
G201 X1 Y1            (Status transition of X/Y axis into manual mode)
P1 = 0
$WHILE P1 == 0
  (Manual operation with X/Y axis)
  (Program can be continued by setting P1 to 1 through operating console)
$ENDWHILE
G202                  (Status transition of all axes into normal mode)
G01 Y200 F500
.....
.....
```

5.28.2 並行補間を行わない選択(G200)

インターポレータと手動操作モード間の内部インターフェイスが、プログラムされた軸に対して有効になります。スピンドルはプログラムできません。

バージョンV2.11.2010.02以降では、軸X.. Y.. のプログラミングがコマンド#ACHSE [...]に代わります。互換性の理由から、このコマンドは当分の間使用できますが、新しいNCプログラムでは使用しないでください。

G200によって、並行補間を**行わない**手動操作を選択できます。

G200	**すべての**軸に対する手動操作	(ノンモーダル)

または複数の特定の軸に対して

G200 *<axis_name><dummy_expr>* {*<axis_name><dummy_expr>*}	(ノンモーダル)

<axis_name><dummy_expr>	複数の特定の軸に対する手動操作の選択。軸指定子以降の値は意味がありませんが、構文上の理由から必要です。

G200がプログラムされている場合、現在のNCプログラムの処理がインターポレータ内で中断されます。この手動操作モードの有効化、変更、および解除が可能です。手動操作時の軸の移動後、制御ステートメント「Continue motion」によってインターポレータにNCプログラムの継続を要求できます。

この動作モードでは、オフセットリミット値が自動的にソフトウェアリミットスイッチ位置に設定されるため、手動モードでソフトウェアリミットスイッチ範囲全体を移動できるようになります。手動操作モードの解除後は、以前のオフセットリミット値が再度有効になります。

G200では、手動操作中の軸の並行補間はできません。ステートメント「Continue motion」の後では、すべての軸および操作モードが解除されるため、手動操作はできなくなります。

G200の解除後、デコーダはインターポレータからのすべての手動モードオフセットおよびコマンド位置を要求し、現在のコマンド位置をNCチャンネル内のすべての関連装置に送信します。手動モードオフセットは解読処理変数V.A.MANUAL_OFFSETSに保管され、NCプログラム内にアドレス指定できます(第13.1章を参照)。インターポレータ内の手動モードオフセットは削除されます。

軸名なしでG200がプログラムされると、このコマンドは存在するすべての軸に影響を与えます。

プログラミング例

```
.....
G00 X 100 Y 100
G200 X1                (Manual operation for X axis)
G01 X200 Y200 F600
G01 Y 200 F500
G200                   (Manual operation for all axes)
G01 Y500 Z500
.....
```

5.28.3 プログラムの終了(M02、M30)

NCプログラムの終了時、手動操作で軸を移動できてはなりません。このため、コマンドM30またはM02はG202が追加でプログラムされているかのように処理されます。

5.28.4 操作モードのパラメータ設定

割り当てられた軸に対して手動モードが解除されている(G202)場合にのみ、パラメータ設定用の#コマンドをプログラムできます。

5.28.4.1 操作モードハンドホイール

バージョンV2.11.2010.02からは、コマンド#HANDWHEEL [...]をコマンド#SET HR [...]の代わりに使用します。互換性の理由から、このコマンドは当分の間使用できますが、新しいNCプログラムでは使用しないでください。

#HANDWHEEL [AX<*axis_name*> | AXNR<*expr*> RES1<*expr*> [RES2<*expr*> RES3<*expr*>]] (モーダル)

AX<*axis_name*>	手動モード軸の名前
AXNR<*expr*>	手動モード軸の論理番号(正の整数)
RES1<*expr*>、	分解能ステージ(最大3)、
RES2<*expr*>、	[mm/ハンドホール回転、インチ/ハンドホイール回転、grd/ハンドホイール回転]
RES3<*expr*>	

プログラミング例

```
With programming of axis name:
...
G202 X1
#HANDWHEEL [AX=X RES1=0.1 RES2=0.2 RES3=0.5]
...
G201 X1
...

..or with programming of logical axis number:
...
G202 X1
#HANDWHEEL [AXNR=1 RES1=0.1 RES2=0.2 RES3=0.5]
...
G201 X1
...
```

5.28.4.2 操作モード連続ジョグ

バージョンV2.11.2010.02からは、コマンド#JOG CONT [...]をコマンド#SET TIP [...]の代わりに使用します。互換性の理由から、このコマンドは当分の間使用できますが、新しいNCプログラムでは使用しないでください。

#JOG CONT [AX<*axis_name*> | AXNR<*expr*> FEED1<*expr*> [FEED2<*expr*> FEED3<*expr*>]] (モーダル)

AX<*axis_name*>	手動モード軸の名前
AXNR<*expr*>	手動モード軸の論理番号(正の整数)
FEED1<expr>、 FEED2<expr>、 FEED3<expr>	速度ステージ(最大3)、[mm/分、インチ/分、grd/分]

プログラミング例

```
With programming of axis name:
G202 X1
...
#JOG CONT [AX=X FEED1=1.0 FEED2=1.5 FEED3=2.0]
...
G201 X1
...

..or with programming of logical axis number:
G202 X1
...
#JOG CONT [AXNR=1 FEED1=1.0 FEED2=1.5 FEED3=2.0]
...
G201 X1
...
```

5.28.4.3 操作モードインクリメンタルジョグ/中断可能なジョグ

バージョンV2.11.2010.02からは、コマンド#JOG INCR [...]をコマンド#SET JOG [...]の代わりに使用します。互換性の理由から、このコマンドは当分の間使用できますが、新しいNCプログラムでは使用しないでください。

#JOG INCR [AX<*axis_name*> | AXNR<*expr*> DIST1<*expr*> FEED1<*expr*> [DIST2<*expr*> FEED2<*expr*> DIST3<*expr*> FEED3<*expr*>]] (モーダル)

AX<*axis_name*>	手動モード軸の名前
AXNR<*expr*>	手動モード軸の論理番号(正の整数)
DIST1<expr>	
FEED1<expr>、	(インクリメンタル幅; 速度)のペア(最大3)、
DIST2<expr>	[mm/(mm/分)、インチ/(インチ/分)、grd/(grd/分)]
FEED2<expr>、	
DIST3<expr>	
FEED3<expr>、	

プログラミング例

```
With programming of axis name:
...
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
...

..or with programming of logical axis number:
...
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
...
```

5.28.5 オフセットリミット値のプリセット

バージョンV2.11.2010.02からは、コマンド#MANUAL LIMITS [...]をコマンド#SET OFFSET [...]の代わりに使用します。互換性の理由から、このコマンドは当分の間使用できますが、新しいNCプログラムでは使用しないでください。

#MANUAL LIMITS [AX<*axis_name*> | AXNR<*expr*> NEGATIVE<*expr*> POSITIVE<*expr*>]

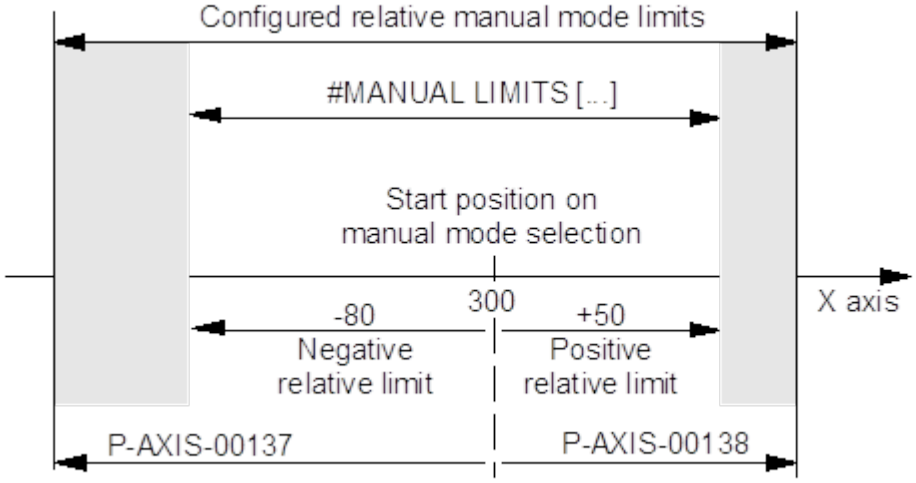
AX<*axis_name*>	オフセットリミット値を有効にする軸名
AXNR<*expr*>	オフセットリミット値を有効にする軸の論理番号 正の整数
NEGATIVE<*expr*>	負のオフセット値。0 [mm、インチ]以下にプログラムする必要があります。
POSITIVE<expr>	正のオフセット値。0 [mm、インチ]以上にプログラムする必要があります。

このコマンドを使用すると、許容される相対手動移動の正負のリミット値を各軸に対して設定できます。

正負の相対オフセットリミット値は、手動モード選択時の開始位置に対応します。

プログラム例

```
N090 G01 F1000 X300
N100 #MANUAL LIMITS [AX=X NEGATIVE=-80 POSITIVE=50]
N110 G200
Nxx ..
```



回転モジュール軸またはスピンドルとして動作する軸(M3、M4、M5、M19、S...)は、このコマンドでは考慮されません。これらのオフセットリミット値は、すべての手動操作モードに対して有効です。これらの値は、G202などで事前にプログラミングすることなく、いつでも上書き可能です。プログラムされた値の符号のチェックのみが実行されます。例えば、許容範囲超過チェックなどの検証は実行されません。オフセット値は、軸のパラメータデータレコードP-AXIS-00138およびP-AXIS-00137を使用して設定することも可能です。

5.28.6 手動モードでの1つの軸に対するパラメータ設定例

プログラミング例

```
%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
...
```

コマンドを使用

#HANDWHEEL [AX=X RES1=0.1 RES2=0.2 RES3=0.5]

X軸のハンドホイール分解能ステージが指定されます。分解能ステージは、0.1、0.2、および0.5 mm/ハンドホイール回転です。

コマンドを使用

#JOG CONT [AX=X FEED1=1.0 FEED2=1.5 FEED3=2.0]

X軸の連続ジョブモードの移動速度が指定されます。連続ジョグ操作の速度ステージは、1.0、1.5、および2.0 mm/分です。

コマンドを使用

#JOG INCR [AX=X DIST1=0.1 FEED1=1.0 DIST2=0.2 FEED2=1.5 DIST3=0.5 FEED3=2.0]

X軸のインクリメンタルジョグモードおよび中断可能なジョグモードがパラメータ化されます。インクリメンタル幅および速度はペアでプログラムされます。

ステージ1:	インクリメンタル幅0.1 mm、	速度1.0 mm/分
ステージ2:	インクリメンタル幅0.2 mm、	速度1.5 mm/分
ステージ3:	インクリメンタル幅0.5 mm、	速度2.0 mm/分

コマンドを使用

#MANUAL LIMITS [AX=X NEGATIVE=-5 POSITIVE=5]

負の手動操作オフセットリミット値は-5 mmに設定され、正のオフセットリミット値は+5 mmに設定されます。

X軸に対して以下のコマンドを使用すると、手動モードが選択/解除されます。G202が単独でプログラムされると、すべての軸に対して手動操作が解除されます。

G201 X1

G202 X1

G202

5.29 オフセット、コマンド、および現在値の要求

以下のNCコマンドは、NCプログラムインタプリタの動作データにインターポレータの手動モードオフセット、コマンド、および現在値を含めるジョブをトリガします。このデータには、NCプログラム内の変数によってアクセスできます。

補間軸のオフセット、コマンド、および現在値は、要求しかできません。回転モジュー口軸またはスピンドルとして動作する軸(M3、M4、M5、M19、S...)は、以下のコマンドでは考慮されません。

NCプログラムのインタプリタとインターポレータの同期は、記載されているすべてのNCコマンドで同一です。すべてのデータがNCプログラムのインタプリタの作業データに受け入れられて初めてジョブが完了し、NCプログラムのコーディングが継続します。

コマンドの概要と特性

コマンド	NCチャンネルの初期化	IPO内の手動オフセット	設定		
			V.A.MANUAL_OFFSETS	V.A.ABS	Pパラメータ、V.S./ V.P./ V.L.
#GET MANUAL OFFSETS	なし	保存されない	あり	なし	なし
#CHANNEL INIT [CMDPOS]	あり	削除される	なし	あり	なし
#CHANNEL INIT [ACTPOS]	あり	削除される	なし	あり	なし
#GET CMDPOS	なし	保存されない	なし	なし	あり
#GET ACTPOS	なし	保存されない	なし	なし	あり

5.29.1 現在の手動モードオフセットの要求、およびV.A.MANUAL_OFFSETS[ ]への代入

バージョンV2.11.2010.02以降では、コマンド#GET MANUAL OFFSETSがコマンド#GET IPO OFFSETに取って代わります。互換性の理由から、このコマンドは当分の間使用できますが、新しいNCプログラムでは使用するべきではありません。

#GET MANUAL OFFSETS

このコマンドは、G201/G202 (並行補間が行われる手動モード)と組み合わせると便利です。デコーダはインターポレータに割り当てられたすべての軸の現在の手動モードオフセットを要求します。手動モードオフセットは取得後にデコーダ変数V.A.MANUAL_OFFSETSに保管され、NCプログラム内にアドレス指定できま

す。NCチャンネルの位置は、暗黙的に初期化**できません**。インターポレータ内の手動モードオフセットはこのジョブによっては削除されず、変数V.A.ABSは更新されません(「軸特有の変数(V.A.) [▶ 421]」の章を参照)。

プログラミング例

```
X100
G201
..... Moving in manual mode
G202
#GET MANUAL OFFSETS
G01 X[100 + V.A.MANUAL_OFFSETS.X] F500 (Initialization of positions)
                                       (in NC channel after)
                                       (deselection of manual mode)
```

5.29.2 現在のコマンド位置の要求、およびV.A.ABS[ ]への代入

バージョン**V2.10.1504**以降では、コマンド**#CHANNEL INIT [CMDPOS]**がコマンド**#SET DEC LR SOLL**に取って代わります。互換性の理由から、このコマンドは当分の間使用できますが、新しいNCプログラムでは使用するべきではありません。

#CHANNEL INIT [CMDPOS]

NCプログラムのインタプリタはインターポレータにすべての補間軸の現在のコマンド位置を要求して、データをNCプログラムのインタプリタ操作データにファイルし、これらの位置でNCチャンネルを初期化します。変数V.A.ABSをプログラミングすると、操作データ内の現在のコマンド位置にアクセスできます。このコマンドにより、インターポレータ内に保存されている手動モードオフセットを削除できます。変数V.A.MANUAL_OFFSETSに代入された値は、このジョブでは更新されません(「軸特有の変数(V.A.) [▶ 421]」の章を参照)。

このコマンドはNCブロック内で排他的であることが必要です。

プログラミング例

```
%channel_init_cmd

X100
G201
..... Moving in manual mode
G202
#CHANNEL INIT [CMDPOS]
#MSG ["X command pos:%F",V.A.ABS.X]
.....
```

5.29.3 現在の実位置の要求、およびV.A.ABS[ ]への代入

#CHANNEL INIT [ACTPOS { AX<*axis_name*> | AXNR<*expr*> }]

AX< *axis_name*> 現在値が要求される軸の名前。

AXNR<*expr*> 現在値が要求される軸の論理番号。正の整数。

NCプログラムのインタプリタはインターポレータにすべて、または複数の特定の補間軸の現在位置を要求して、データをNCプログラムのインタプリタ操作データにファイルし、これらの位置でNCチャンネルを初期化します。変数V.A.ABSをプログラミングすると、操作データ内の現在のコマンド位置にアクセスできます。このコマンドにより、インターポレータ内に保存されている手動モードオフセットを削除できます。変数V.A.MANUAL_OFFSETSに代入された値は、このジョブでは更新されません(第13.1章を参照)。

プログラムされている軸が**ない**場合、NCチャンネル内の**すべての**補間軸に対して現在位置が要求されます。

プログラムされた軸の場合には現在位置が要求され、プログラムされていない軸の場合にはコマンド位置が要求されます。

注記

現在位置をコマンド位置として受け入れると、軸動作が発生する場合があります。特にNCプログラムループ内では、これによってドライブのドリフトが発生することがあります。

プログラミング例

```
%channel_init_act

G01 F1000 X100 Y200

#CHANNEL INIT [ACTPOS AX=X AX=Y ]       (Request of actual value by)
                                        (axis name..)
#CHANNEL INIT [ACTPOS AXNR=1 AXNR=2 ]   (..or logical axis number..)

#CHANNEL INIT [ACTPOS]                  (..or for all path axes)

#MSG ["Actpos X:%F, Y:%F ",V.A.ABS.X, V.A.ABS.Y]

M30
```

5.29.4 特定の軸の現在のコマンド位置の要求、およびユーザ定義変数やパラメータへの代入

バージョンV2.11.2010.02からは、コマンド#GET CMDPOS [...]をコマンド#SET IPO SOLLPOS [...]の代わりに使用します。互換性の理由から、このコマンドは当分の間使用できますが、新しいNCプログラムでは使用するべきではありません。

#GET CMDPOS [V.S.*<string>* | **V.P.***<string>* | **V.L.***<String>* | **P***<expr>***=** *<axis_name>*

{ **,V.S.** *<string>* |**V.P.** *<string>* | **V.L.***<String>* |**P***<expr>* **=** *<axis_name>* } **]**

インターポレータに格納されている特定の軸の現在のコマンド位置がデコーダによって要求され、指定されたユーザ定義変数(V.S./V.P./V.L.)、またはパラメータ(P..)に代入されます。NCチャンネルの位置は、暗黙的に初期化**できません**。このコマンドでは、インターポレータ内に保存されている選択された軸の手動モードオフセットは削除されません。NCプログラムからアドレス指定可能な変数V.A.ABSおよびV.A.MANUAL_OFFSETSは、このジョブでは更新されません。

プログラミング例

```
.....
#GET CMDPOS [P1 = X, P2 = Y, V.P.POS1 = Z, V.P.POS2 = C]
G01 XP1 YP2
G01 ZV.P.POS1  CV.P.POS2
.....
```

5.29.5 特定の軸の現在位置の要求、およびユーザ定義変数やパラメータへの代入

TwinCAT CNC: V2.11.2022.05以降

その他: V263以降

#GET ACTPOS [V.S.*<string>*| **V.P.** *<string>*| **V.L.***<String>* | **P***<expr>***=** *<axis_name>*

{ **,V.S.** *<string>* |**V.P.** *<string>* | **V.L.***<String>* |**P***<expr>* **=** *<axis_name>* } **]**

解読処理によって、インターポレータに格納されている特定の軸の現在位置が要求され、指定されたユーザ定義変数(V.S./V.P./V.L.)またはパラメータ(P..)に代入されます。NCチャンネルの位置は、暗黙的に初期化**できません**。このコマンドでは、インターポレータに格納されている選択された軸の手動モードオフセットは削除されません。NCプログラムからアドレス指定可能な変数V.A.ABSおよびV.A.MANUAL_OFFSETSは、このジョブでは更新されません。

プログラミング例

```
.....
#GET ACTPOS [P1 = X, P2 = Y, V.P.POS1 = Z, V.P.POS2 = C]
G01 XP1 YP2
G01 ZV.P.POS1  CV.P.POS2
.....
```

5.30 ギア切替(G112)

G112	ギア切替	(ノンモーダル)

ファンクションG112を使用すると、補間軸ごとにギアを切り替えることができます。G112はスピンドル軸にも使用できますが、この場合は第14.2.7章に記載されている特性を考慮する必要があります。

実ギアがない軸については、この機能を軸パラメータ各種として使用することもできます。これにより、例えば重切削加工を行う場合において、コントローラの機能を拡張してポジションエラーをモニタリングしてパラメータセットを個別に設定したり、ギア番号でNCプログラムを呼び出したりすることが可能です。

注記

G112は、同一のNCブロック内で軸準備ファンクションと一緒にプログラムすることはできません。

プログラミング例

```
Nnn G112 X4 Y8        (Gear stage for X-axis: 4, Gear stage for  Y-axis: 8)
```

5.31 先読み機能の効果(G115/G116/G117)

G115	通常の先読み機能の効果	(モーダル)
G116	ブロック遷移速度の計算の効果	(モーダル)
G117	先読み機能のリセット	(モーダル、デフォルト)

先読み機能の概要

先読み機能として、最大限の動作速度を実現するための様々なファンクションを実装しているおり、それにより軸速度および軸加速度を常に維持しておくことができます。

以下のシングルファンクションが、速度プロファイルの維持に効果があります。

(1) チャンネルパラメータリストP-CHAN-00071で指定された最大軸速度を守り、最大経路速度を計算する。

これにより、プログラムされた経路上の速度は、動作に関係する各軸がそれぞれの最大軸速度を超えないように制限されます。

(2) 最大弦誤差を守り、曲線経路上の最大経路速度を計算する。

これにより、各曲線の中で最も曲率が厳しいものに対して、曲線動作を決定することができます。この円上では、生成された点が始点および終点に対応するように、円弧部分も定義できます。実際の機械軸が各弦を移動していくという状況であるため、誤差値も含めてそれぞれの円弧の高さ設定することができます。

(3) チャンネルパラメータリストP-CHAN-00071で指定された最大軸加速度を守り、曲線経路上の最大経路速度を計算する。

割線誤差に基づき計算される最大経路速度は、トレース時間が短く円弧運動が小さい場合に特に速すぎる場合があり、その結果生じる円中心方向の加速度が軸加速度の許容値を超過する可能性があります。

(4) チャンネルパラメータリストP-CHAN-00071で指定された軸のリミット値およびそれらの軸の動作を考慮したブロック毎の形状遷移に基づき、遷移する経路速度を計算する。

ブロック遷移中は通常、全体の動作の中でそれぞれの軸の動作比率は変化します。ただし、この軸状況の変化は直線動作などの新規経路を最初にトレースするサイクルでのみ発生するものです。生成される加速度なしで遷移が行われることが予測できる場合(例えば傾斜ファンクションなど)は、その結果生じる軸加速度を推測できます。これにより、遷移時に最大軸加速度を超過しないように、ブロック遷移時の最大経路速度の計算が可能になります。

一般的なケース:

先読みを使用した通常操作

プログラム開始後、すべての先読み機能がオンに切り替わります(G117)。これにより例えば、軸データとして与えられた最大軸加速度のようなパラメータが必ず守られるようになります。

特殊なケース:

G115を用いたシングルファンクションごとの先読み機能の切り離し

プログラミング例

```
Nnn G115 = 0          (Switching off the single functions (2),(3) and(4))
                      (of Look-ahead-funct.)
```

以下の表で、組み合わせ可能なシングルファンクションについて説明します。

選択または解除されるファンクションは、常にすべての軸が対象になります。

G115で使用できる識別番号(ID)の概要:

シングルファンクション ID	(1)	(2)	(3)	(4)
0	*			
2	*			*
4	*	*		
6	*	*		*
8	*		*	
10	*		*	*
12	*	*	*	
14	*	*	*	*

*	シングルファンクションが有効		シングルファンクションが無効

プログラミング例

```
Nnn G115 = 2     (Switching off individual functions (2) and (3))
Nnn G115 = 12    (Switching off individual function (4))
Nnn G115 = 14    (Switching on of all individual functions after)
                 (program start, see also G117)
```

特殊なケース:

G116、G117による先読み機能の効果

G116により、**個々の**プログラム可能な軸に対してブロック遷移速度の計算をオフにできます。軌跡内での「コーナー」では、ブロック遷移速度(経路速度)の減衰は発生しません。

この場合、軸のダイナミックデータ(軸加速度)は通常、超過してしまいます。

プログラミング例

```
N10 G116 X1 Y2                  (No decrease in block transition rate of)
                                (different axis speeds of the axes X or Y)
N20 G01 G91 X100 Y-100 F1000
N30 X-100
```

この例では、設定された速度を超過することなくブロック遷移N20/N30を移動します。軸の経路により、コマンドによる指令値がブロック遷移時にジャンプする形で変更されます。

注記

座標値は構文上の理由でのみ必要で、意味はありません。

G117を使用すると、先読み機能の**すべての**シングルファンクションが有効(プログラム開始後の初期設定)になります。

プログラミング例

```
Nnn G117              (Switching on all single functions (1),(2),(3),(4))
                      (of Look-ahead-function)
```

5.32 オーバライド(G166)

G166	オーバーライド設定 100%ド	(ノンモーダル)

ファンクションG166を使用すると、補間軸に対して外部から設定されたオーバライドの影響がオフになります。また、プログラム内のブロック [▶ 344]で、あるいは個別の軸 [▶ 555]に対して、設定されたオーバライド値の影響も抑制されます。

プログラミング例

```
%override_G166_extern
;Assumption: External override 50%

N10 G00 G90 X0 Y0 Z0            ;Rapid feed 50%
N20 G01 X10 F2000               ;Feed F1000
N30 X20 Y20                     ;Feed F1000
N40 G166 Z30                    ;Feed F2000, override 100%
N50 X30                         ;Feed F1000
N60 G166 Y30                    ;Feed F2000, override 100%
N70 G166 G00 X0 Y0 Z0           ;Rapid feed 100%
N80 G01 F3000
N90 X10 Y20 Z30                 ;Feed F1500
...
M30


%override_G166_path

;Path specific override: G01 120%, G00 75%
N10 #OVERRIDE [FEED_FACT=120 RAPID_FACT=75]

N20 G01 X10 Y10 Z10 F1000       ;Feed F1200
N30 G166 X20 Y20 Z20            ;Feed F1000, override 100%
...
N50 G00 X50                     ;Rapid feed 75%
N60 G166 Y50 Z50                ;Rapid feed 100%
...
M30


%override_G166_ax

;Axis specific override for X: G01 20%, G00 60%
N10 X[OVERRIDE FEED_FACT=20 RAPID_FACT=60]

N20 G00  X10                    ;Rapid feed in X with override 60%
N30      Y10 Z10                ;Rapid feed in Y/Z with 100%
N40 G166 X20                    ;Rapid feed in X with override 100%

N50 G01 X30 F2000               ;Feed in X with F400, override 20%
N60     Y20 Z20                 ;Feed in Y/Z with F2000 (100%)
N70 G166 X40                    ;Feed in X with F2000, override 100%
...
M30
```

6 切り替えおよび補助ファンクション(M / H / T)

6.1 ユーザ定義のM/Hファンクション

通常、M/Hファンクションは「プログラマブルロジックコントローラ」(PLC)のプログラミングによって決定されます。現在のNCブロックでプログラムしたM/Hファンクションにより、(「WindowからPLCへの」)インターフェイス情報を介して制御をおこないます。さらに、同期条件を設定できるファンクションが追加されており、NCとPLC間の同期を制御することができます。

M *<expr>*
または
H *<expr>*

M/Hファンクションは、一般的な数式でプログラム可能です。これにより、例えばMP10といったパラメータによる値の指定が可能になります。計算された数値は、M/Hファンクションにアクセスする前に四捨五入され、整数に変換されます。最小許容値は0 (ゼロ)です。負の値の場合にはエラーメッセージが生成されます。

1つのNCチャンネルに対するM/Hファンクションの最大数 ([6]-8.1/-8.2) 、および1つのNCブロックに対するM/Hファンクションの最大数 ([6]-8.3) は固定値で指定されます。

EXISTファンクション(「演算式<expr> [▶ 25]」の章を参照)は、M/Hファンクションが使用できるかどうかをチェックします。

プログラミング例

最初の手順として、ユーザ指定のMファンクションの呼び出し前にEXIST要求をすることにより、NCチャンネルリストP-CHAN-00027内でそのMファンクションが定義されているかどうかがチェックされます。

```
...
N10 G90 Y0
N20 $IF EXIST[M80] == TRUE
N30   X0 Y0 Z0 M80      (M80 is available)
N40 $ELSE
:   …
N60 $ENDIF
...
M30
```

以下のMファンクションは、NCプログラム内ではあらかじめ定義された固定の意味を持ち、自由に使用することができません。同期モードでのみ、P-CHAN-00027にてユーザによって設定可能です。

M00	プログラム停止
M01	オプショナルストップ
M02	プログラムの終了、機械の停止

M17	サブプログラムの終了
M29	サブプログラムの終了
M30	プログラムの終了、機械の停止

6.1.1 プログラム停止(M00)

M00により、例えば計測やチップ廃棄を実行するために、実行中のNCプログラムが中断されます。「CONTINUE MOTION」ボタンが押されると、加工が継続します。移動命令があるブロック内にM00が存在する場合、この移動命令の実行後に中断が発生します。

6.1.2 オプションの停止(M01)

M01はM00と同様に動作しますが、動作時にM01を有効にしておく必要があります。

6.1.3 プログラムの終了(M02/M30)

M02およびM30は同一の動作を行い、メインプログラムおよびサブプログラムに対して、プログラムの実行をリセットし、また、NCブロックの初期位置への移動を行います。

さらに、設定によってはPLCに信号を送信することも可能です。PLCはこの信号を使用し、例えばワーク交換動作の同期などを行います。

6.1.4 サブプログラムの終了(M17/M29)

M17およびM29はサブプログラムを適切に終了します。

さらに、設定によってはPLCに信号を送信することも可能です。

6.2 軸特有のM/Hファンクション

通常のDIN規格に沿ってプログラムされたユーザ定義のM/Hファンクションは一般的に、チャンネルごとに処理および実行されます。

M/Hファンクションによりユーザーがどうしても軸特有の動作をさせたいという場合には、軸特有の動作をするように、チャンネルパラメータリストP-CHAN-00039およびP-CHAN-00025に追加定義するオプションが用意されています。

m_default_outp_ax_name[*<m_expr>*]	*<axis_name>*
または	
h_default_outp_ax_name[*<h_expr>*]	*<axis_name>*

<m_expr> 軸特有の効果を持つユーザ定義のMファンクション。

<h_expr> 軸特有の効果を持つユーザ定義のHファンクション。

<axis_name> M/Hファンクションが動作する軸の名前。このケースでは、補間軸およびスピンドル軸が対象となります。

プログラミング例

チャンネルパラメータリスト[1]から抽出:

```
:
# Definition of the axis spezific M-Functions
# ==========================================
m_default_outp_ax_name[20]                          S2
m_default_outp_ax_name[21]                          S3
m_default_outp_ax_name[22]                          Z
:
# Definition of the axis spezific H-Functions
# ==========================================
h_default_outp_ax_name[10]                          X
h_default_outp_ax_name[11]                          Y
h_default_outp_ax_name[12]                          Z
:

:
Nn S1000 M3 M20 M21 M22 H10   (S and M3 effect on the main spindle axis,
                               M20 effects on the S2-spindle axis
                               M21 effects on the S3-spindle axis and
                               M22 effects on the Z-axis
                               H10 effects on the X-axis)
:
```

軸特有のM/Hファンクションに対して、括弧表現のプログラムでスピンドル軸特有のM/Hファンクションとして指定した場合、初期設定が無視され、スピンドル軸でM/Hファンクションが動作します。

さらに、補間軸に対しては、**NCプログラム**に以下の括弧表現を使用することで軸特有のM/Hファンクションとして強制的に利用するオプションも用意されています。

<axis_name>[M*<expr>* H*<expr>* { M*<expr>* H*<expr>* }] { *<axis_name>*[...] }

<axis_name> M/Hファンクションが動作する軸の名前。このケースでは、補間軸のみが対象となります。

M*<expr>* 軸特有の動作を行うユーザ定義のMファンクション。

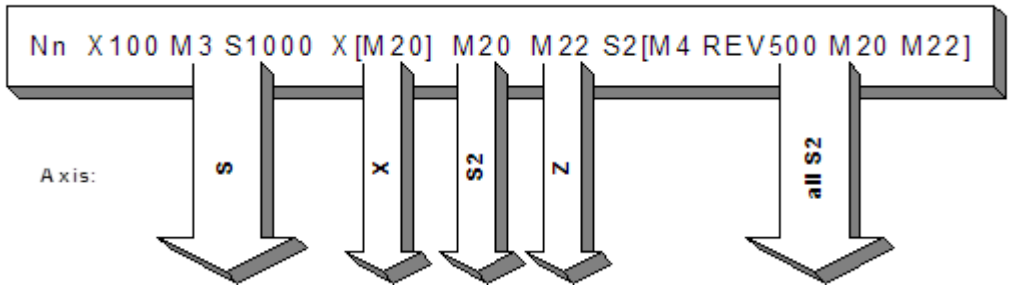
H*<expr>* 軸特有の動作を行うユーザ定義のHファンクション。

プログラミング例

```
:
N10 S1000 M3 X100 X[M20 H12] Y[H10] (S, M3 act on the main spindle axis)
                                    (M20, H12 act on the X-axis)
                                    (H10 acts on the Y-axis)
:
```

これによりユーザ定義のM/Hファンクションを、軸特有とするかチャンネル特有とするか、プログラミングによってユーザーが柔軟に実行できます。

プログラミング例



6.3 M/Hファンクションについてのオプションの追加情報

NCプログラムで指定するM/Hファンクションでは、オプションの追加指定値をプログラムすることができます。M/Hファンクションと組み合わせて、PLCのインターフェイスとして利用可能です。

スピンドル軸ファンクションであるM03、M04、M05、M19、およびギヤ切替ファンクションであるM40～M45を除き、ユーザ定義のすべてのM/Hファンクション、および内部Mファンクションであるm00、M01、M02、M17、M29、M30に対して追加指定値をプログラムできます。M/Hファンクションの追加指定値は、チャンネル特有および軸特有のいずれのプログラミングでも制限なく使用できます。

M<*expr*> [= <*additive_expr*>]
または
H<*expr*> [=<*additive_expr*>]

M<*expr*> ユーザ定義のMファンクション

H<*expr*> ユーザ定義のHファンクション

<*additive_expr*> オプションの追加指定値。直接または一般的な数式のプログラムにより、負または正の整数をプログラムすることが可能です。

プログラミング例

```
%m_h_add_fct

(M functions with additional value)
N10 M52=-345
N20 M12=123     (with channel parameter m_default_outp_ax_name[12]  Z)
N30 M10=321     (with channel parameter 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 functions with additional value)
N110 H5=-345
N120 H6=123     (with channel parameter h_default_outp_ax_name[6]  Z)
N130 H9=321     (with channel parameter 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

(Mixed M/H functions with additional value)
N200 X[M52=-345 H4 H8=-789 M54=567 H5=345] H3=333 M54=444 H7=567 M63

(M/H functions with additional value in axis speczific function(INDP))
N05 X[INDP G90 G01 FEED=2000 POS=555 M54=151 H8=-181]

N999 M30=111
```

6.4 工具選択(Tワード)

T *<expr>*	工具の選択	(モーダル)

工具コマンドは、加工のための工具を決定します。工具番号はプログラマブルロジックコントローラ(PLC)に転送され、工具交換のために工具マガジン内の指定された工具を呼び出します。Tワード自体は、工具形状(工具データ)とは関係なく、制御内部での計算には影響を与えません。この目的では、Dワードが使用されます。P-CHAN-00014を指定しておくことによって、Tワードに対応するDワードが自動的に有効となります。

Tワードだけでは工具交換は開始しません。通常、工具交換はマシンファンクションM06によって開始します。

工具データ ([5]、[6]-9.7) は、CNCの工具データテーブル ([6]-9.1) に保存されます。また、オプションとして(外部の)工具データ管理システムとの通信機能もあります。この工具データ管理システムは、ユーザが提供したり、配信されたり、追加インターフェイス経由で要求したりして利用します。内部工具データまたは外部工具データへのアクセスは、P-CHAN-00016で指定可能です。

工具データの読み出しや転送の待ち時間の回避のために、外部から読み出される工具に対しては通常、実際に工具データが組み込まれるよりも1つ以上前の動作ブロックで要求されます (P-CHAN-00087)。複数の異なるT番号が先頭のT番号に対応するD番号の前にプログラムされている場合は、最後にプログラムされているT番号よりも前のT番号は無視されます。つまり、工具のD番号が最後にプログラムされているT番号と一致しない場合、外部の工具管理機能によって提供されるデータは無視されます。

7 速度 (F-、E-)

F *<expr>*	ブロック内での送り速度	(モーダル)
E *<expr>*	ブロック終了での送り速度	(ノンモーダル)

補間タイプG01、G02、G03、G100、G105では、プログラムされた軸がFワードで宣言された経路速度で移動します (設定により、直進軸の場合はプリセット値mm/分、m/分、インチ/分、mm/秒、インチ/秒などが使用され、回転軸の場合は°/分が使用されます)。このFワードはモーダルです。

ユーザコマンド「G93」を使用すると、Fワードによる送り速度の代わりに、加工時間も指定できます。このコマンドの説明は、以下の項に記載されています。「Gファンクション」

ブロック終了時の速度は、Eワードによってプログラムします。値はブロック固有であり、モーダルではありません。Eワードが宣言されていないか、またはプログラムされた値がFワードよりも大きい場合、EワードはFワードの値に設定されます。

EワードはG94と組み合わせて使用する場合のみ有効です!

G301、G302、G41、G42、G261、#HSC [OPMODE 1…]などの形状変更ファンクションによって作成され、挿入された形状要素は、前の移動ブロックのE送り速度を引き継ぎ新しいF送り速度として使用します。その他のすべてのスプラインおよびHSCファンクションの場合には、E送り速度の使用を避けることをお勧めします。

値は直接、または各パラメータによりFワードおよびEワードに割り当てられます。10進数(REAL型)で指定が可能です。単位はチャンネルパラメータP-CHAN-00108で設定可能です。

プログラミング例

```
N10 X200 G01 F1000 G90 ;F-Feedrate 1000 mm/min
N20 X300
N30 X350 F800          ;Lowering of feed velocity from 1000 to 800 mm/min
```

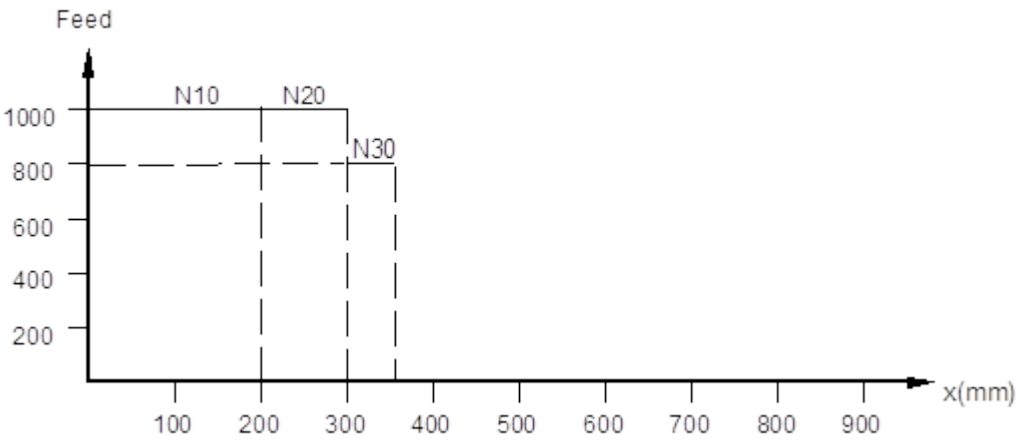


図 61: 図6-1: Fワードを使用した送りのプログラミング

プログラミング例

```
N05 G01 G94 G90
N10 X200 F1000 E500        ;F-Feedrate 1000 mm/min, E-Feedrate 500mm/min
N20 X250       E400        ;F-Feedrate 1000 mm/min, E-Feedrate 400mm/min
N30 X350 F800  E300        ;F-Feedrate 800 mm/min, E-Feedrate 300mm/min
N40 X400                   ;F-Feedrate 800mm/min, E-Feedrate 800mm/min
```

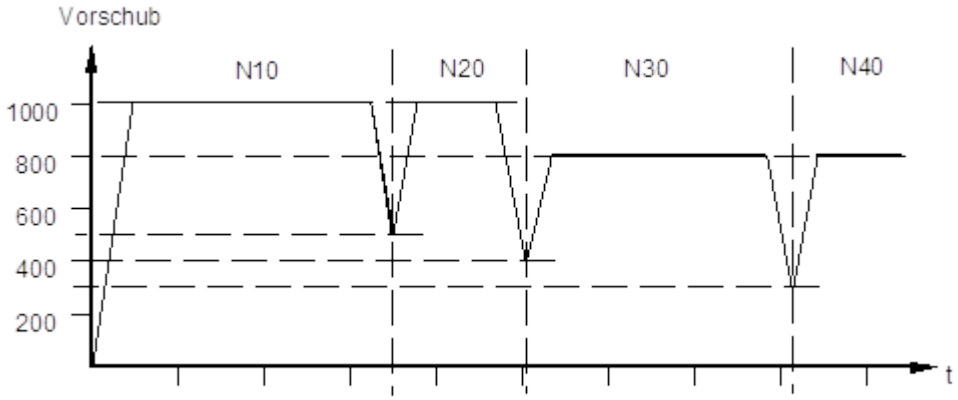


図 62: 図6-2: FワードおよびEワードを使用した送りのプログラミング

プログラミング例

```
N10 G01 G94 G90 G261
N20 X100 F1000 E200        ;F-Feedrate 1000 mm/min, E-Feedrate 200mm/min
N30 Y100       E200        ;F-Feedrate 1000 mm/min, E-Feedrate 200mm/min
N40 X0                     ;F-Feedrate 1000 mm/min
N50 Y0         E200        ;F-Feedrate 1000 mm/min, E-Feedrate 200mm/min
N60 X50
N70 G260
```

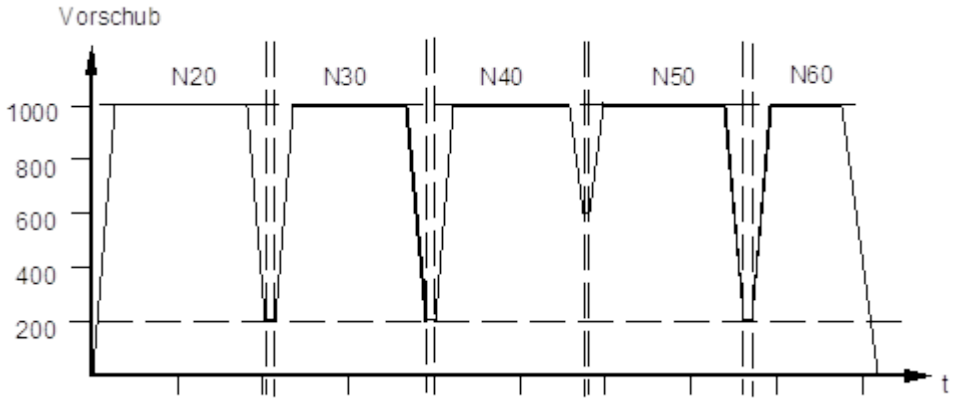


図 63: 図6-3: 挿入された形状要素(ここではG261)へのEワードの適用

8 Nファンクション

NCブロック番号は、アドレス文字「N」を使用してプログラム可能です。設定されている場合は、この番号が表示データとエラーメッセージの両方に登録されます(または、ファイル内のオフセット値を使用することもできます)。

N <expr>

ブロック番号は、一般的な数式でプログラム可能です。これにより、例えばプログラムループ内において、実行パラメータなどの値を割り当てることが可能になります。計算された数値は内部で自動的に四捨五入され、整数に変換されます。

プログラミング例

```
N10 G01 X200 F500
N20 $FOR P1=1, 10, 1
N[P1*100] XP1
    .
    .
N30 $ENDFOR
```

プログラムフローでは、ブロック番号は重要ではありません。このため、ブロック番号を昇順で使用する必要はありません。

9 サブプログラム手法

似たような繰り返される動作やファンクションフローをサブプログラムとして実行させることができます。

サブプログラムには2つのタイプがあります。

- ローカルサブプログラム
- グローバルサブプログラムまたはサイクル

入れ子構造にした複数のサブプログラム(ローカルまたはグローバル)を呼び出すことができます。そのため、使用されるサブプログラム数、つまりネストの深さを[6]-6.24で定義します。

メインプログラムからサブプログラムへの変数の伝送は、Pパラメータによって間接的に行うか、サイクリックに呼び出される特定の伝送パラメータによって直接的に行います。

9.1 ローカルサブプログラム(LL <string>呼び出し)

ローカルサブプログラムの呼び出しは、以下で実行されます。

LL *<string>*	(注意:「LL」と<string>の間には空白文字が必要です。)

<string>	ローカルサブプログラム名

ローカルサブプログラム(LUP)はメインプログラムと共に共通のデータファイルに記述されます。共通のデータファイルでは、すべてのサブプログラムが実際のメインプログラムの前に記述される必要があります。

ローカルサブプログラムは、同一データファイル内のメインプログラムからのみ呼び出せることに注意してください。

ローカルサブプログラムは、「%L」およびプログラム名で始まります。「L」とプログラム名の間には、1つ以上の区切り文字が必要です。

サブプログラムの終了は、M17またはM29を記述します。M02およびM30によってプログラムを中断できます。その後、警告が出力されます。

これらのプログラムの終了コードが記述されていない場合、「%」(次のメインプログラムの先頭文字)がサブプログラムの終わりとなります。

プログラミング例

```
Structure of a data file, consisting of NC main program and local sub-routines:

%L UP1                      (1st local sub-routine)
N1 .....
N2 .....
N9 M17                      (M17 can also be left out)
```

```
%L UP2                      (2nd  local sub-routine)
N11 .....
N12 .....
N19 M29                     (M29 can also be left out)


%100                        (Main  program)
N100 .....
N105 .....
N200 LL UP1                 (Call of 1st LUP)
     .
N250 LL UP2                 (Call of 2nd LUP)
     .
N300 M30                    ( Main program end)
```

プログラム定義の前に、NCコマンドを配置することはできません。これは、NCブロック内のプログラム実行の順序は重要ではないという規則の例外です(「NCブロック構造 [▶ 20]」の章を参照)。

9.2 グローバルサブプログラム(呼び出しL <string>)

グローバルサブプログラムの呼び出しは、以下で実行されます。

L*<string>*または L *<string>*

<string> グローバルサブプログラム名

グローバルサブプログラム(GUP)は、個別のデータファイル内に独立したプログラムユニットとして存在します。グローバルサブプログラムの呼び出しは、サブプログラム名ではなく、ローカルサブプログラムおよびメインプログラムで構成されるデータファイルの名前で実行されます。グローバルサブプログラム名の指定(%<Prog_Name>)は、ローカルサブプログラムと終了とメインプログラム(ここではグローバルサブプログラムとして使用)の開始を示すこと以外には必要ありません。そのため、データファイル内にローカルサブプログラムが存在する場合には省略できます。

同様に、メインプログラムの呼び出しも他のデータファイル内に独立したプログラムユニットとして保存されます。グローバルサブプログラムは、すべてのメインプログラムから呼び出せます。

プログラミング例

```
Call of local and global sub-routines

%L LUP                  (Local sub-routine)
N11 .....
N12 .....
     .
     .
N19 M17                 (M17 can be left out)


%333                    (Main program)
N100 .....
N105 .....
N110 LL LUP             (Call up of local sub-routines LUP)
     .
N200 L GUP_FILE         (Call up of global sub-routines via the name of the)
     .                  (data file, in which this  GUP is stored)
N300 M30
```

次の例では、ローカルサブプログラムが存在しないデータファイル名によって呼び出しを行うため、グローバルサブプログラム名を省略できます。

プログラミング例

```
Structure of data file GUP_FILE with global sub-routine:
 (globale subroutine GUP)

N10 .....
    .....
N90 M17               (Here an M17 or M29 is mandatory)
```

プログラム定義およびプログラム呼び出しの概要

メインプログラム

定義:	%PROG_NAMEまたは% PROG_NAME	プログラム定義は、ローカルサブプログラムが存在する場合にのみ必要です。
呼び出し:	操作コンソール経由でのファイル名指定	

グローバルサブプログラム

定義:	メインプログラムと同様	
NCプログラムからの呼び出し:	LFILE_NAMEまたはL FILE_NAME	

ローカルサブプログラム

定義:	%L PROG_NAME	「L」とプログラム定義の間に空白文字がない場合は、メインプログラムの定義とみなされます。
NCプログラムからの呼び出し:	LL PROG_NAME	「LL」とプログラム呼び出しの間に空白文字がない場合は、グローバルサブプログラムの呼び出しとみなされます。

ファイルのコメント外で、先頭文字が区切り文字でも「%」でもない場合、この文字は命名されていないメインプログラムの先頭文字として評価されます。これにより、「%」の前にブロック番号がプログラムされていない可能性も考えられます。

9.3 パラメータによるサブプログラム呼び出し(LL / L V.E. …)

ローカルサブプログラムおよびグローバルサブプログラムは、固定名の代わりに外部変数によって呼び出せます。これにより、パラメータによるNCプログラムフローを外部から制御できます。外部変数は、「String」および「String array」タイプであることが必要です(「外部変数(V.E.) [▶ 443]」の章も参照)。

ローカルサブプログラムの呼び出しは、以下で実行されます。

LL V.E. …	(注意:「LL」とV.E. …の間には空白文字が必要です)

V.E. … 外部変数によって定義されたローカルサブプログラム名

グローバルサブプログラムの呼び出しは、以下で実行されます。

LV.E. … *または* **L V.E. …**

V.E. … 外部変数によって定義されたグローバルサブプログラムを含むファイル名

それ以外は、「ローカルサブプログラム(LL <string>呼び出し) [▶ 176]」および「グローバルサブプログラム(呼び出しL <string>) [▶ 177]に記載されている規則と同様の規則が適用されます。

プログラミング例1

```
Call of sup programs via external variables of type “string”
%L TASCHE               (Local sub-routine)
N10 .....
     .
N99 M17


%MAIN                   (Main program)
N100 .....
N105 .....
N110 LL V.E.LUP         (Call of local sub-routine via external variable)
                        (V.E.LUP, including the string TASCHE)
     .
N200 L V.E.GUP          (Call of global sub-routine via external variable)
                        (V.E.GUP, including the string of a file name)
N300 M30
```

プログラミング例2

```
Call of sup programs via external variables of type “string array”
%L TASCHE_1             (Local sub-routine 1)
N10 .....
     .
N99 M17
%L TASCHE_2             (Local sub-routine 2)
N10 .....
     .
N99 M17
%L TASCHE_3             (Local sub-routine 3)
N10 .....
     .
N99 M17

%MAIN                   (Main program)
N100 .....
N105 $FOR P1 = 1,3,1
N110  LL V.E.LUP[P1]    (Call of sub-routines via external variables)
                        (V.E.LUP[1], V.E.LUP[2],V.E.LUP[3] including)
                        (the strings TASCHE_1, TASCHE_2, TASCHE_3)
N120 $ENDFOR
     .


N205 $FOR P2 = 1,5,1
N210  L V.E.GUP[P2]     (Call of sub-routines via external variables)
                        (V.E.GUP[1], V.E.GUP[2]..., including the)
                        (strings of the file names)
```

```
N220 $ENDFOR

N300 M30
```

9.4 プログラム開始時の暗黙的なグローバルサブプログラム呼び出し

他のNCメインプログラムの開始中に、同一の定期的なデータ初期化の実行が必要な場合が時々あります。この初期化がチャンネルパラメータP-CHAN-00119を使用してグローバルサブプログラム内で要約されている場合、サブプログラムをNCプログラムのスタート時の最初の動作[CHAN]として暗黙的に実行できます。

このサブプログラム内では、NCプログラムの対象ファンクションをすべて使用できます。つまり、メインプログラム内のLワードを使用したサブプログラム呼び出しと同様です。

さらに、この機能はオフセット、Gファンクション、パラメータ値、変数などの初期化、定義およびプリセットなどに使用すると便利です。ここでは、操作の具体的な手順はプログラムしてはなりません。

9.5 プログラム終了時の暗黙的なグローバルサブプログラム呼び出し

他のNCメインプログラムの終了(M02、M30)時に、同一の定期的なデータ初期化(変換、オフセット、ファンクションの解除など)の実行が必要な場合が時々あります。この初期化がチャンネルパラメータP-CHAN-00252を使用してグローバルサブプログラム内で要約されている場合、サブプログラムをM02またはM30の**前の**最後の動作として暗黙的に実行できます。

このサブプログラム内では、NCプログラムの対象ファンクションをすべて使用できます。つまり、メインプログラム内のLワードを使用したサブプログラム呼び出しと同様です。

さらに、このファンクションはオフセット、Gファンクション、パラメータ値、変数などのリセットに使用すると便利です。ここでは、操作の具体的な手順はプログラムしないでください。

9.6 グローバルサブプログラムまたはローカルサブプログラムとしてのサイクル(呼び出しL | LL CYCLE)

NCカーネル内ではグローバルサブプログラムまたはローカルサブプログラムとしてサイクルを使用でき、深穴加工やポケット加工などの特殊加工操作が可能になります。サイクル内で定義される加工タスクは、一般的な書式で記述します。サイクルが呼び出されると、伝送パラメータの割り当てでデータが提供されます。

サイクルは現在有効な平面(G17、G18、G19)、およびNCチャンネル内で設定された軸に関係なくプログラムされます。サイクル内では、カプセル化されたユーザ独自のパラメータグループにアクセスできます。これらは、サイクルが呼び出された際に割り当てられる値です。

このために、文字「@」を使用する特殊な構文が使用できます。サイクルのプログラミングでは、「@」を以下と組み合わせて使用します。

@Pxx サイクル呼び出しおよびサイクル内の伝送パラメータ

@X、@Y、@Z サイクル内の主軸

@I、@J、@K サイクル内の中心点座標

@S サイクル内のメインスピンドル

サイクルは、他のNCコマンドがない他のNCブロック内でプログラムされる必要があります。構文は、サイクルに依存する伝送パラメータが指定されたグローバルサブプログラムまたはローカルサブプログラムで構成されます。

L | LL CYCLE [NAME=<*cycle*> [MODAL_MOVE/ _BLOCK] @P1=<*expr*> … @P50=<*expr*> { \ }]

NAME=<cycle> サイクル名(ファイル名)

MODAL_MOVE モーダルなサイクル呼び出し。動作コマンドを含むメインプログラム内のNCブロックの後に、

(モーダル、古い構文) サイクルが暗黙的に必ず再実行されます。

MODAL_BLOCK モーダルなサイクル呼び出し。メインプログラム内のNCブロックの後に、サイクルが暗黙的に必ず再実行されます。

以下のNCコマンドでは、暗黙的なブロックモーダルサイクル呼び出しが停止します。

- 空白行、コメント行

- サブプログラム呼び出し(L、LL、M6、G8xx)

- $ コマンド($GOTO、$IF、$FORなど...)

- プログラム終了Mファンクション(M2、M30、M17、M29)

MODAL_MOVE / _BLOCKのモーダル効果は、NCコマンド#DISABLE MODAL CYCLEによってキャンセルされます。

@P1<expr>... 伝送パラメータのリスト

... @P50<expr> 最大で50のREAL型の@Pxxパラメータを転送できます。サイクル内では書き込みおよび読み取りアクセスのみ許可されます。@Pxxパラメータは、値、任意の千数、Pパラメータ、および数式によって直接割り当て可能です。

\ 複数行にわたるコマンドの明確なプログラミングのための区切り記号(「バックスラッシュ」)。

括弧内では、キーワードおよび伝送パラメータの順序は**前後しても構いません**。プログラミング中は、ユーザは呼び出されるサイクルに対してどの@Pxxパラメータを割り当てる必要があるかのみを認識している必要があります。

不要なパラメータは省略できます。プログラムされていない@Pxxパラメータは、呼び出し時に0 (ゼロ)で初期化されます。パラメータがプログラムされたか(有効であるか)どうかは、サイクル内で変数V.G.@P[i].VALID [▶ 425]によってチェックできます。伝送パラメータは、他のサイクルの**プログラムされた呼び出し**まで保持されます。

サイクル内でプログラムされているが、そのサイクル内では使用されない伝送パラメータは無視されます。

現在のサブプログラムまたはプログラムレベルがサイクルかどうかは、変数V.G.CYCLE_ACTIVE [▶ 425]でチェックできます。

注記

サイクルは、加工タスクが定義されている独立したNCプログラムユニットです。伝送パラメータを複数設定してしまう恐れがあるため、サイクルの呼び出しは入れ子構造にしないことを推奨します。

実行中のNCプログラム内、または単一ブロックモードのサイクル実行中に、初期設定でサイクルの実行行が表示されるかどうかは、バージョンによって異なります。表示がオフの場合には、サイクル呼び出しのみが表示されます。

TwinCATシステムでは、すべてのサイクル行がデフォルトでディスプレイに表示されます。

この機能は、チャンネルパラメータP-CHAN-00211で変更可能です。

サイクル呼び出し前に必要な定義

- サイクル呼び出し前に有効なモーダルなGファンクション、円弧形状、および有効な送り速度(Fワード)は、サイクル終了後も保持されます。このファンクションは、チャンネルパラメータP-CHAN-00210で変更可能です。
- 軸名と組み合わせてプログラムされるモーダルなGファンクション(G92、G98、G99、G100、G112、G130など)は、サイクル内でプログラムされていなければ復元されません。

- 加工平面(G17、G18、G19)は、サイクル呼び出し前に上位のNCプログラム内で定義する必要があります。ドリルサイクルでは、この平面に直交する軸が掘削操作が実行される軸です。研削サイクルの場合は、この軸が深度を決定するフィード軸です。
- 工具形状補正(長さ補正など)は、サイクルが呼び出される前に選択する必要があります。
- サイクル内に対応する伝送パラメータが存在しない場合、送り速度で必要な値、スピンドルの速度、およびスピンドルの回転方向は、上位のNCプログラム内で定義する必要があります。
- サイクル内でプログラムされたスピンドルコマンドは、必ずNCチャンネルの有効なメインスピンドルを基準にします。サイクル呼び出しの前に、このメインスピンドルが定義されているかを確認します。
- 対応するドリル操作または研削操作の開始位置、および工具の向きは、必ず上位のNCプログラム内でサイクルが呼び出される前に移動する必要があります。

モーダルなサイクルのキャンセル

モーダルで動作するサイクル(サイクル呼び出し内のキーワードMODAL_MOVEまたはMODAL_BLOCK)は、以下のNCコマンドでキャンセルされます。このコマンドは、NCブロック内で独立してプログラムする必要があります。

```
#DISABLE MODAL CYCLE
```

使用可能なサイクル

加工タスク用

- ドリル加工
- ねじのタッピング
- ヘリカルミル加工
- ポケット加工

パラメータ定義可能なサイクルが用意されています。プログラミングおよび使用に関する詳細情報は、機能説明「サイクル(C21)」([FCT-C21])を参照してください。

プログラミング例

以下ではドリルサイクル(drill.cyc)の呼び出し例を用いて、さまざまなパラメータ割り当ての方法を示します。

ドリルサイクル「drill.cyc」は、以下の伝送パラメータを必要とします。

@P1 -> 戻り平面の位置(絶対値)

@P2 -> 加工平面の位置(絶対値)

@P3 -> 安全距離(符号なし)

@P4 -> 最終ドリル加工奥行き(絶対値)

@P5 -> 加工平面に対する最終ドリル加工奥行き(符号なし)

定数でのサイクル呼び出し

```
..
Nxx L CYCLE [NAME=drill.cyc @P1=110 @P2=100 @P3=4 @P4=40]
..
or with specification of a relative drilling depth @P5:
..
Nxx L CYCLE [NAME=drill.cyc @P1=110 @P2=100 @P3=4 @P5=60]
..
```

変数でのサイクル呼び出し

```
The variables must be defined and assigned values before the cycle call.
..
#VAR
 V.L.RPL = 110
 V.L.WPL = 100
 V.L.SDST = 4
 V.L.DEP = 50
#ENDVAR
Nxx L CYCLE [NAME=drill.cyc @P1= V.L.RPL @P2=V.L.WPL @P3=V.L.SDST @P4=V.P.DEP]
..
```

Pパラメータでのサイクル呼び出し

```
The parameters must be defined and assigned values before the cycle call.
..
Nxx P10 = 110
Nxx P11 = 100
Nxx P15 = 4
Nxx P17 = 50

Nxx L CYCLE [NAME=drill.cyc @P1= P10 @P2=P11 @P3=P15 @P4=P17]
..
```

数式でのサイクル呼び出し

```
..
Nxx P20 = 100

Nxx L CYCLE [NAME=drill.cyc @P1= 10+P20 @P2=2*50 @P3=5-1 @P4=P20/2]
..
```

定数でのサイクル呼び出し、カッコ内のパラメータの順序は任意

```
..
Nxx L CYCLE [@P4=40 NAME=drill.cyc @P2=100 @P3=4 @P1=110]
..
```

定数でのサイクル呼び出し、サイクルはモーダル効果を持つ

```
..
Nxx L CYCLE [NAME=drill.cyc @P1=110 @P2=100 @P3=4 @P4=40 MODAL_MOVE]
..
```

例

```
%drill_main
N05 T1 D1
N10 M06
N15 X0 Y0 Z0 G0 G90 F200 S300 M3 G53 G17
N20 Z110
N30 X40 Y40 (drilling 1)
N40 L CYCLE [NAME=drill.cyc MODAL_MOVE @P1=110 @P2=100 @P3=2 @P4=55]
N50 X60 Y60 (drilling 2 and implicit cycle call because it is modal)
N60 X100 Y60 (drilling 3 and implicit cycle call because it is modal)
N70 X100 Y20 (drilling 4 and implicit cycle call because it is modal)
#DISABLE MODAL CYCLE
```

```
N80 X0 Y0  M5
N100 M30
```

サイクル作成時の注意

サイクルは可能な限り有効、かつ現在NCチャンネルおよび平面定義で使用されている軸名に依存しないようにプログラムする必要があります。これを行うために、サイクル内で先頭の3つの主軸に対して平面に依存しない「中立な軸名」@X、@Y、および@Zを使用する方法があります。意味:

@X -> 常に1番目の主軸を意味します

@Y -> 常に2番目の主軸を意味します

@Z -> 常に3番目の主軸を意味します

例: サイクル内の軸

```
Nxx G91 @X=@P1 @Y=@P2 @Z=@P3 F1000 G01
```

同様に、円弧のプログラミング時にはいわゆる「中立中心点座標」を使用できます。意味:

@I -> 常に1番目の主軸の中心点座標を意味します

@J -> 常に2番目の主軸の中心点座標を意味します

@K -> 常に3番目の主軸の中心点座標を意味します

例: サイクル内の円弧

```
Nxx G91 G02 @X=@P1 @Y=@P2 @I=@P4 @J=@P5 F1000
```

スピンドルのプログラミング中に、設定されたスピンドル名に依存しないようにするために、サイクル内でスピンドルを常に中立なスピンドル名@Sでアドレス指定できます。

@S -> 常にメインスピンドルを意味します

例: サイクル内のスピンドル

```
Nxx @S=1000 M3 (main spindle cw with 1000 rpm)
```

9.7 ブロックシーケンスの呼び出し(L SEQUENCE)

同一のNCプログラムまたはグローバルサブプログラム内のブロックシーケンスは、以下の記述で呼び出します。

L SEQUENCE [[NAME=<*string*>] N<*expr*> [N<*expr*>] REPEAT<*expr*>]

NAME=<*string*>	同一の名前、またはブロックシーケンスが実行されるグローバルサブプログラムの名前です。名前がプログラムされていない場合、ブロックシーケンスは現在のNCプログラム内で実行されます。
N<*expr*>	実行される最初のブロックの番号です(開始番号、ブロックシーケンスの開始)
N<*expr*>	実行される最後のブロックの番号です(戻り番号、ブロックシーケンスの終了)。両方のブロック番号が同一の場合は、このブロックのみが実行されます。
REPEAT<*expr*>	ブロックシーケンスの繰り返しの回数(1以上の正の整数)です。

コントローラは指定されたNCプログラム(コマンドを呼び出す同一プログラムの場合もあります)内でプログラムされたN (ブロック)番号を研削します。2つのN番号が、ブロックシーケンス内で実行される先頭および最後のNCブロックをマークします。このブロックシーケンス外のNCブロックは実行されません。

プログラム時に、開始番号と戻り番号をコマンド内で交換することもできます。ただし、NCプログラム内ではブロックシーケンスは必ず小さなN番号から順に実行されます。

開始番号または戻り番号が見つからない場合には、エラーメッセージが出力されます。

ブロックシーケンスが複数回実行される場合(REPEAT > 1)、ブロックシーケンスの終端で開始番号から再度プログラムが開始します。すべての実行が終了すると、プログラムはサブプログラムから戻り、残りのプログラムシーケンスが続行されます。

コマンド内でN番号が指定されている場合、この行のみが実行されます。これは、2つの同一のN番号による呼び出しの場合も同様です。

ブロックシーケンスをすべて呼び出すことは、サブプログラムを呼び出しと同じことです。このため、ネストの深さについてはグローバルサブプログラムと同じ規則が適用されます。

注記

コンテキスト評価

サブプログラム内のプログラムコンテキストは、ブロックシーケンスの最初のNC行が実行されるとセットアップされます。それまでに実行されたすべてのNC行は評価されません。事前に定義された変数、座標系、パラメータ、モーダルステートメントなども、作成や初期化がされません。このため、これらはブロックシーケンス内では認識されず、使用できません。
特に、制御ブロック構造($IF-$ELSE-$ENDIF, $SWITCH,...)を使用したブロックシーケンスの実行の場合、エントリと戻り点によってユーザ自身がこれらを競合なく実行できる必要があります。

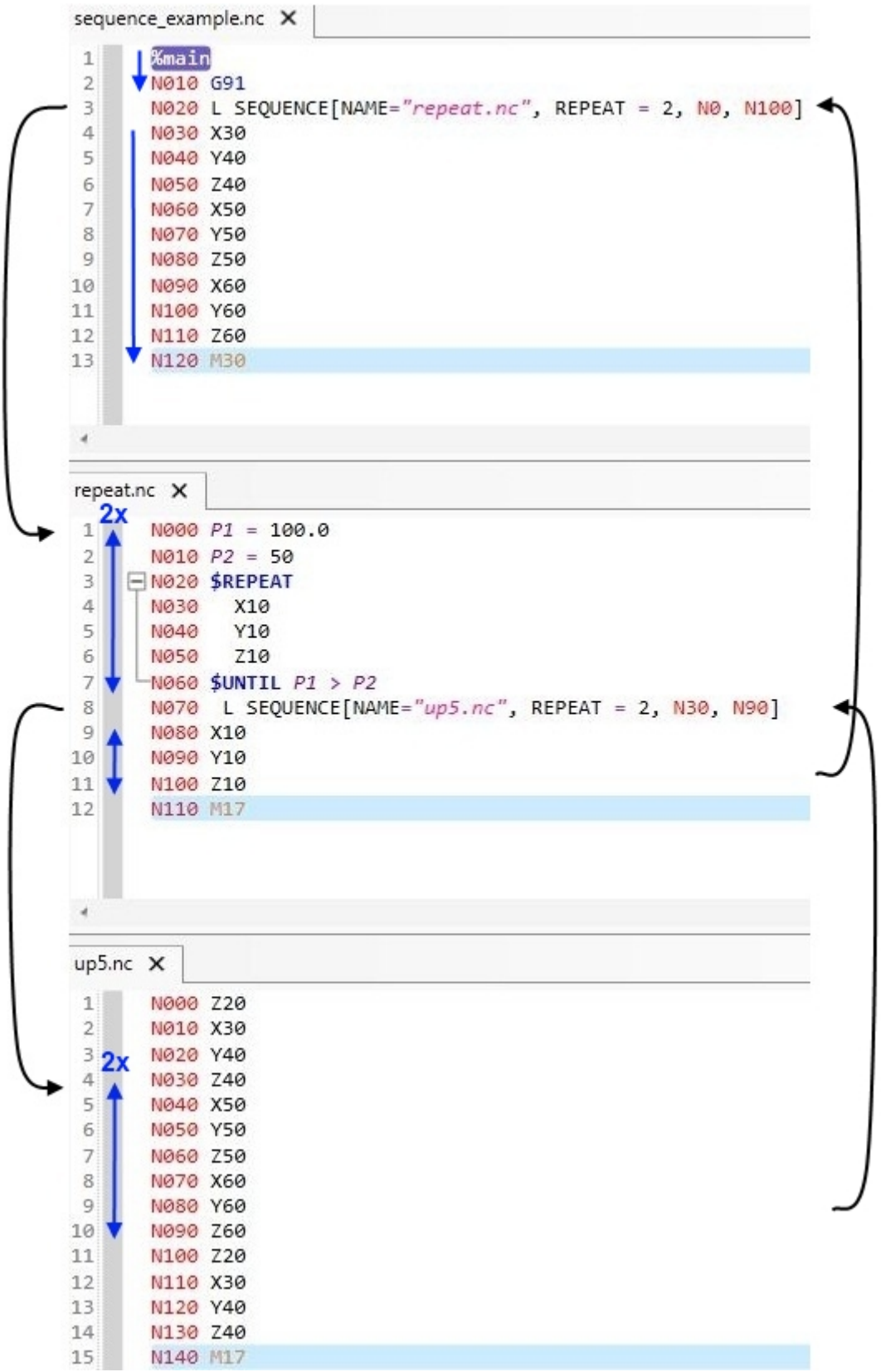
プログラミング例

```
%sequence_example.nc
N010 G91
N020 L SEQUENCE[NAME="sequence.nc", REPEAT = 2, N0, N100]
N030 X30
N040 Y40
N050 Z40
N060 X50
N070 Y50
```

```
N080 Z50
N090 X60
N100 Y60
N110 Z60
N120 M30
```

シーケンス

- メインプログラム「sequence_example.nc」がN20まで実行されます。
- N0からN100までrepeat.ncによって2回繰り返されるシーケンスコマンドが呼び出されます。
- サブプログラム「repeat.nc」がN0からN60まで実行されます。
- $REPEATループの後で、N30からN90まで2回実行される「up5.nc」のシーケンスコマンドが呼び出されます。
- サブプログラム「up5.nc」がN30からN90まで2回実行されます。
- 2回の実行後、プログラムは「repeat.nc」N70まで戻り、残りの行N80からN100までが実行されます。
- プログラムがN0まで戻り、それ以降はプログラムは実行されません。
- 「repeat.nc」の実行が2回完了すると、プログラムは「sequence_example.nc」まで戻り、N30からN120までが実行されます。


sequence_example.nc
%main
N010 G91
N020 L SEQUENCE[NAME="repeat.nc", REPEAT = 2, N0, N100]
N030 X30
N040 Y40
N050 Z40
N060 X50
N070 Y50
N080 Z50
N090 X60
N100 Y60
N110 Z60
N120 M30
repeat.nc
2x
N000 P1 = 100.0
N010 P2 = 50
N020 $REPEAT
N030 X10
N040 Y10
N050 Z10
N060 $UNTIL P1 > P2
N070 L SEQUENCE[NAME="up5.nc", REPEAT = 2, N30, N90]
N080 X10
N090 Y10
N100 Z10
N110 M17
up5.nc
N000 Z20
N010 X30
2x
N020 Y40
N030 Z40
N040 X50
N050 Y50
N060 Z50
N070 X60
N080 Y60
N090 Z60
N100 Z20
N110 X30
N120 Y40
N130 Z40
N140 M17

10 パラメータおよびパラメータ計算

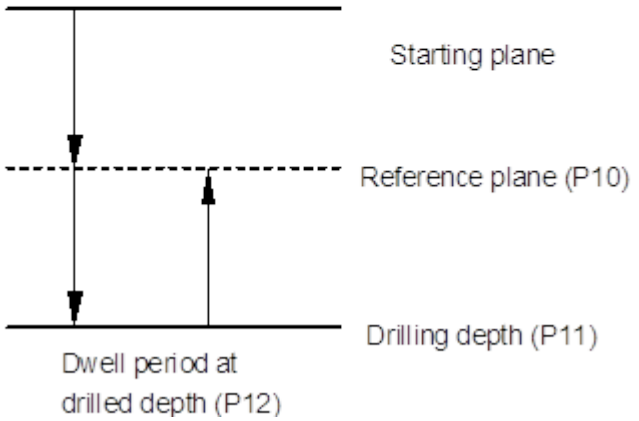
NCプログラム内ではパラメータを、数値の保存場所として使用できます。パラメータを使用する利点は、プログラムフロー中にパラメータの値は変更が可能なことです。これにより、フレキシブルなNCプログラム開発が可能になります。

RワードとPワードのどちらが使用されるかは、パラメータ名[6]-6.44の設定によって異なります。ここに示す例では、Pパラメータを使用します。

プログラミング例

ドリルサイクルなどのサブプログラム内では、座標値(ドリルの奥行き、ドリルのフィード、ドウェルなど)の代わりにパラメータが使用されます。対応する呼び出しメインプログラム内の最終値がパラメータに割り当てられます。

For the global sub-routine
%4712(drilling, face countersinking)
the following parameters are to be defined:
P10 -Reference plane=withdrawal plane
P11 -Drilling depth
P12 -Dwell period



メインプログラム内の呼び出しは、以下のように表現されます。

```
:
N100 P10=20.5 P11=12.6 P12=1.2
N110 L4712
:
```

図9-1: パラメータ計算の適用例

P *<expr>*	標準パラメータ

P*<expr>* パラメータインデックスには任意の値を使用できますが、必ず0よりも大きな値であることが必要です。使用されるパラメータの最大数は、[6]-6.19で設定できます。

通常のパラメータに加えて、パラメータ配列(例えばP100[50])も使用できます。配列の次元は、[6]-6.20で設定できます。

P *<expr>* **[** *<expr>* **]** { **[***<expr>***]** }	パラメータ配列

チャンネルパラメータP-CHAN-00067は、Pパラメータがプログラム全体で有効かどうかを指定します。

パラメータはNCプログラム内で、**#VAR**で始まり**#ENDVAR**で終わる宣言ブロック内で作成/初期化することも、最初の書き込みアクセスによって暗黙的に作成/初期化することも可能です。ただし、パラメータ配列は必ず宣言ブロック内で作成される必要があります。読みやすくするために、パラメータ配列の初期化は文字「\」を使用して複数のNCブロックに渡って記述することができます。作成には以下の構文を使用します。

#VAR	宣言ブロックの開始
:	
:	宣言および初期化の部分
:	
#ENDVAR	宣言ブロックの終了

プログラミング例

```
#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
:
```

パラメータ配列へのアクセスは、インデックス0から開始します! 上記の例では、P10[0][5]にアクセスすると値15が取得されます。

パラメータおよびパラメータ配列は、NCプログラム内で削除することも可能です。削除するには、#DELETEコマンドを以下の構文で使用します。

#DELETE P *<expr>* {,**P***<expr>*}

プログラミング例

```
#DELETE P10, P20, P100, P200, P201
```

さらに、SIZEOFおよびEXISTファンクション(「演算式<expr> [▶ 25]」の章を参照)を使用して、パラメータ配列の大きさを決定したり、パラメータの存在をチェックしたりできます。

パラメータはNCプログラムを使用して割り当てます(例えば、P12=0.12)。パラメータにより、以下のような制御に依存する値、またはプロセスに依存する値にも対応できます。

- 現在の軸回転数
- ドライブ内のトルク
- 外部計測デバイスの値
- 熱および力量センサの値
- 操作メニューからのキーボード入力
- など

数値を直接割り当てる代わりに、連鎖した数式を使用することも可能です(「数式 [▶ 24]」の章も参照)。入力には、以下のような一般的な計算規則が適用されます。

- 加減の前に乗除を行う
- 括弧規則。ただし、角括弧[]を使用する

10.1 パラメータによる座標のプログラミング

パラメータを使用した軸名による座標のプログラミング構文は、以下の通りです。

<axis_name> **P***<expr>*

<axis_name> 軸名

P<*expr*> 割り当てられたパラメータ

P<*expr*>の代わりに数式も使用できます。

プログラミング例

```
X P1*SIN [P2*30]
```

10.2 間接パラメータ

数式および割り当てでは、直接パラメータと同様に間接パラメータも使用されます。モード[6]-6.44に応じて、以下のケースが区別されます。

ケース1: パラメータにPワードのみ使用できる

直接プログラミング(Pnn)、間接プログラミングともにPワードを使用して排他的に行います。間接パラメータ使用時は、以下が適用されます。

PPnnがパラメータPnnをポイントする

初期化中に、PnnのアドレスがPPnnに割り当てられます。PPP...の使用も可能です。

例:

P120=10の場合、値10がパラメータ120にロードされます。ただし、ステートメントPP120=123.456は、アドレスがP120内に存在するパラメータ(P10)にこの値を割り当てます。したがって、PP121=SQRT[2,0]の結果は以下のようになります。

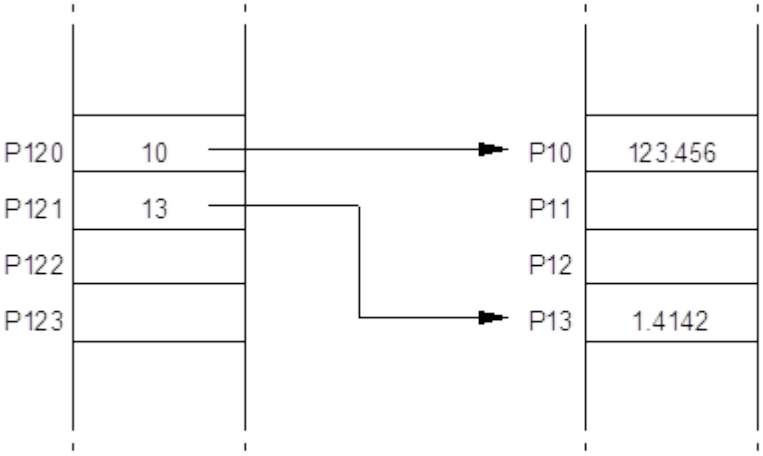


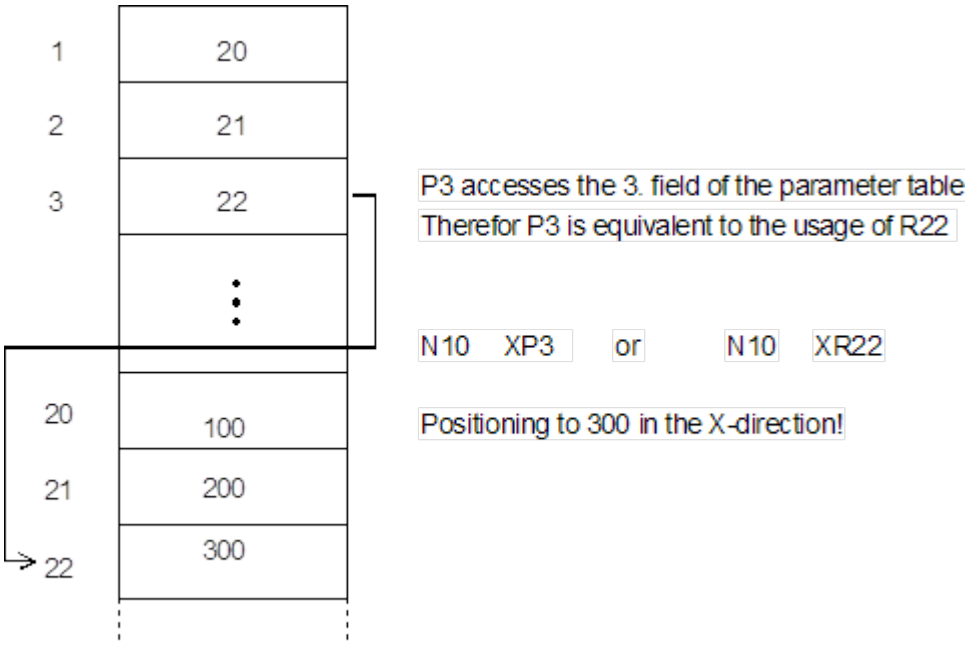
図 64: 図9-2: 間接Pパラメータの効果

ケース2: 直接パラメータとしてRワードを使用し、間接パラメータとしてPワードを使用する

直接プログラミングはRワード(Rnn)を使用して行い、間接プログラミングはPワードを使用して行います。このモードでは、先にパラメータテーブルがRワードによって割り当てられていなければ、Pワードを使用して間接的にアクセスできません。Pワードは排他的に間接的に動作するため、ここではプログラミングをPPで行う必要はありません。

DINに準拠し、「Pnn」はパラメータテーブルのフィールド「nn」にインデックスが格納されているパラメータをポイントします。

例:



間接パラメータを使用すると、パラメータのすべてのフィールドへの割り当てが可能です。

プログラミング例

PパラメータP20およびP40に50を割り当てる

```
N110 P1 = 20  P2 = 40
N120 PP1 = 50
N130 PP2 = PP1
```

PパラメータP15およびP25に0.0を割り当てる

```
N110 $FOR P1 = 10,20,1
N120 P[P1 + 5] = 0.0
N130 $ENDFOR
```

パラメータ(PPPP ...nn)の間接アドレス指定中のネストの深さは、設定によって異なることに注意が必要です。

11 NCプログラムのフローシーケンスに関係するステートメント

制御ブロックステートメントの構文は以下の通りです。

$<*statement*>

<statement> 以下に基づく制御ブロック文字列。$と*<statement>*の間には空白文字を入力してはならないことに注意が必要です。

プログラムフロー(制御ブロック)に影響を与えるステートメントによって、以下が可能になります。

- 条件ジャンプ: 例えばトリガの場合、計測値に応じて加工ステップが変化します。
- インクリメンタルなデジタルループ: 例えば、直線研削や円弧穴のドリル加工中の複数の繰り返される加工ステップのプログラミングを簡略化できます。
- 実行条件付きループ: 中止条件が満たされるまで、複数の加工ステップを繰り返せます。例えば、工具の送り込みや加工操作が、定義された座標値に達するまで実行されます。実行条件がない、または満たされない場合、ループはエンドレスループとしてプログラムされます。

制御ブロックステートメントの使用では、以下の規則が適用されます。

- 1つのNCブロック内では、記述できる制御ステートメントは1つのみです!
- 制御ブロックステートメントを入れ子構造にできます。ネストの深さは指定されています。
- 制御ブロックステートメントの前には、ブロック番号と「/」以外はプログラムできません。
- 制御ブロックステートメントの無効な分岐では、ブロック番号およびその他の(入れ子構造にした)制御ブロックステートメントの構文チェックが行われます(IF/ELSE分岐の例を参照)。

プログラミング例

無効な分岐での構文チェック:

```
N10 $IF 0
N20 XY              (Here no syntax check takes place)
N30 $ENDIF

N10 $IF 0
N20 ...
N30 $IF XY          (Syntax-check because of nested control-block-)
                    (statement; Error message because of the unknown term)
N40 ...
N50 $ENDIF
N60 $ENDIF

N10 $IF 0
```

```
NXY                (Syntax check of block number;)
                   (Error message because of unknown term)
N30 $ENDIF
```

計算やパラメータの内部表現に誤りがある場合、制御ブロックステートメント内の対応する操作が誤った結果となる可能性があります(「演算式<expr> [▶ 25]」の章を参照)。このため、誤りが疑われる場合は、値の正確さではなく許容範囲をチェックする必要があります。

プログラミング例

```
Wrong:
N10  $FOR P1 = 0, 10, 1
N20    P2 = P2 + 0.01
N30  $ENDFOR
N40  $IF P2 == 0.1             (Due to inaccuracies of calculation P2)
N50    ...                     (can be unequal to 0.1, so that the $ELSE-)
N60  $ELSE                     (branch is executed)
N70    G04  X20
N80  $ENDIF

Correct:
N10  $FOR P1 = 0, 10, 1
N20    P2 = P2 + 0.01
N30  $ENDFOR
N40  $IF [P1 - 0.1] <= .000001  (Verification of a tolerance range non-)
                                (problematical for NC manufacture;)
N50    G04  X5                  ($IF-branch will be executed)
N60  $ELSE
N70    ...
N80  $ENDIF
```

11.1 条件分岐

11.1.1 IF/ELSE分岐

IF/ELSE条件分岐では、以下の制御ステートメントを使用します。

$IF, $ELSE, $ELSIF, $ENDIF.

分岐は必ず

$IF *<expr>*

で始まり、以下のステートメントで終了します。

$ENDIF

制御ステートメント

$ELSE

および

$ELSEIF

はオプションで、複数の分岐の記述に使用します。

注記

$IF 制御ブロック内の条件は、数式の「真」または「偽」(TRUEおよびFALSE)を確認することでチェックします。DECIMAL型の変数も使用できるようにするために、

.. 数式の絶対値が0.5以上の場合に、ジャンプ条件が満たされます。

プログラミング例

```
N10 ...
N20 $IF P1      Only if |P1| is greater or equal to 0,5, the statements
N30 ...         N30 to  N50 will be executed.
N40
N50
N60 $ENDIF
```

以下も可能です。

```
N10 ...
N20 $IF P1 >= 0.5     Only if P1 is greater or equal to 0.5 , the state-
                      ments N30 to N50 will be executed.
N30 ...
N40
N50
N60 $ENDIF
```

または

```
N10 ...
N20 $IF P1 > P2  Only if P1 is greater than P2, the statements N30 to N50
                 will be executed, otherwise the statements N70 to N90
N30 ...
N40 ...
N50 ...
N60 $ELSE
N70 ...
N80 ...
N90 ...
N100 $ENDIF
```

ELSEIFにより、以下が許可されます。

```
N10 ...
N20 $IF P1 == 0     Only if P1 is equal to 0, the statements N30 to N50
                    will be executed, otherwise the $ELSEIF-condition will
                    be checked, whether P2 >= 0.5 and accordingly N70 to
                    N90 or N110 to N130 will be executed.

N30 ...
N40
N50
N60 $ELSEIF P2>=0.5   The $ELSEIF-condition serves for building up
                      branching nesting into one another.
N70 ...
N80
N90
N100 $ELSE
N110 ...
N120
N130
N140 $ENDIF
```

注記

1.) プログラミング言語Cに準拠し、以下に構文の違いが存在します。

割り当て: P5 = 3

および

比較: $IF P5 == 3

2.) バージョン2.3以前の場合: 数式が常にシーケンスであるため…

演算子 -> 項 -> 演算子 -> 項 -> など

…が比較演算に必要な場合、マイナス記号とそれらの式は括弧で囲む必要があります("-" は演算子として解釈されます)。

プログラミング例

```
$IF P1 >= -5        wrong, because term->operator->operator->term

$IF P1 >= [-5]      right, because term->operator->term->operator->term
```

11.1.2 CASE分岐

CASE分岐により、数式の関数として異なるNCプログラムの処理が可能になります。

CASE分岐では、以下の制御ステートメントを使用します。

$SWITCH, $CASE, $DEFAULT, $ENDSWITCH .

CASE分岐は必ず

$SWITCH *<expr1>*

で始まり、複数の

$CASE *<expr2>*

$BREAK

オプションで、以下が続きます。

$DEFAULT

で始まり、以下のステートメントで終了します。

$ENDSWITCH

プログラミング例

```
N100 $SWITCH P1=INT [P1*P2/P3]  If the result of the arithmetic expression
                                is equal 1, the blocks after $CASE 1 will
                                be executed (N120 bis N140)
N110   $CASE 1
N120 ... N130
N140   $BREAK
N150   $CASE P2            If the result is equal to P2, tzhe blocks
                           N160 ..N170 will be executed.
N160 ...
N170   $BREAK
N300   $CASE n
N320 ...
N330   $BREAK
N350   $DEFAULT            The $DEFAULT block is optional and serves to
                           process the NC-blocks.
N360 ...                   N360 to N380, if the result of the $SWITCH-
                           Block has not matched any of the
N370
N380
N390 $ENDSWITCH
```

式*<expr1>*および*<expr2>*の比較は、内部的なREAL型で実行されます。ここでは、値の差が0.001未満の場合、両方の式が等しいと評価されます。

式*<expr1>*および*<expr2>*は負の値でも構いません。

11.1.3 $GOTOステートメント

サブプログラム手法や制御ブロック命令($IF, $FORなど)の他にも、他のプログラムセクションに分岐する方法もあります。NCプログラム内にラベルを記述し、GOTOコマンドを使用することでプログラム内の任意の場所に制御を移動できます。

ジャンプステートメントには2つの使用方法があります。

式 - ラベル:

N *<block_number>*:	定義

$GOTO N** *<block_number>* \| **$GOTO N*<expr>*	ジャンプ呼び出し

文字列 - ラベル:

[*<string>*]	定義

$GOTO [*<string>* **]**	ジャンプ呼び出し

特性:

- 呼び出し$GOTOは、NCプログラム内のラベル定義の前後に関係なく記述できます。ラベルはプログラム内で上下両方向に検索されます。
- ラベルは、必ず同一プログラムレベルで(プログラム内部でローカルに)呼び出しと定義が行われる必要があります。メインプログラムとサブプログラム、およびサブプログラム間のジャンプは許可されていません(図10-1を参照)。
- メインプログラムとサブプログラムで同一のラベルを定義できます。
- NCプログラム内の複数の箇所から同一のラベルにジャンプできます。
- $IFステートメントは、同一NC行内の$GOTOと組み合わせることができます。この場合、対応する$ELSE/$ENDIF構文はプログラムできません。
- 同一NC行内の$GOTOコマンドの前後に、他のNCコマンドをプログラムできます。ただし、ジャンプはNC行内の最後の動作です。

- $IF-$ELSE-$ENDIF制御ブロックの任意のレベル、およびこれらのレベル内、およびレベル間での外部ジャンプが可能です。ただし、このジャンプインレベルは有効なレベルです(条件が「真」、「**プログラミング例**」を参照)。
- $WHILE, $FOR, $DO, $REPEAT内のジャンプは許可されて**いません**。
- すべてのレベルの$GOTOによって制御ブロックからいつでも終了することができます。
- コメント内のラベル(#COMMENT BEGIN, #COMMENT END)は認識されません。
- 文字列ラベル内では、大文字と小文字は区別されません。
- 解読中に検出されたラベルはすべて保存されます。保存可能な数式ラベル[6]-6.41と文字列ラベル[6]-6.42の最大数、および文字列ラベルの長さ[6]-6.43は固定です。

- ジャンプが呼び出される度に、ジャンプラベルが既に認識されているか、および保存されているかがチェックされます。既に認識されており、保存されている場合は、ジャンプが実行されます。ジャンプラベルが認識されていない場合、現在のNCブロックの有効なプログラムレベルからプログラム終端(M29/M30)まで検索されます。ラベルが見つからない場合には、エラーメッセージP-ERR-208が出力されます。
- 保存可能なラベルの最大数に達すると、新規に検出されたラベルは保存できません。これらは警告P-ERR-20829およびP-ERR-20831によって表示されます。認識されていないラベルでの新しいジャンプラベル呼び出しが行われる度に、現在のプログラムレベルの始点から再度検索が開始します。NCプログラムが非常に大きな場合は、ジャンプの実行に時間がかかります。

プログラミング例

```
%goto

N05  P1=1
N06  P2=1

N10  G74 X1 Y2 Z3
N11      X0 Y0 Z0

N15   $IF P1==1 $GOTO N40: -> Jump from outside to N40 into a control
                                block

N20  X10
N25  Y10

N30  $IF P1==2
N35    X20
N40:    $IF P2==1
N45    X30
N50:    Y30 $GOTO N65: -> Jump to N65 between control block levels
                           (IF-ELSE)
N55  $ELSE
N60    Y40
N65:    X40
N70  $ENDIF

N80  Z99

N999  M30
```

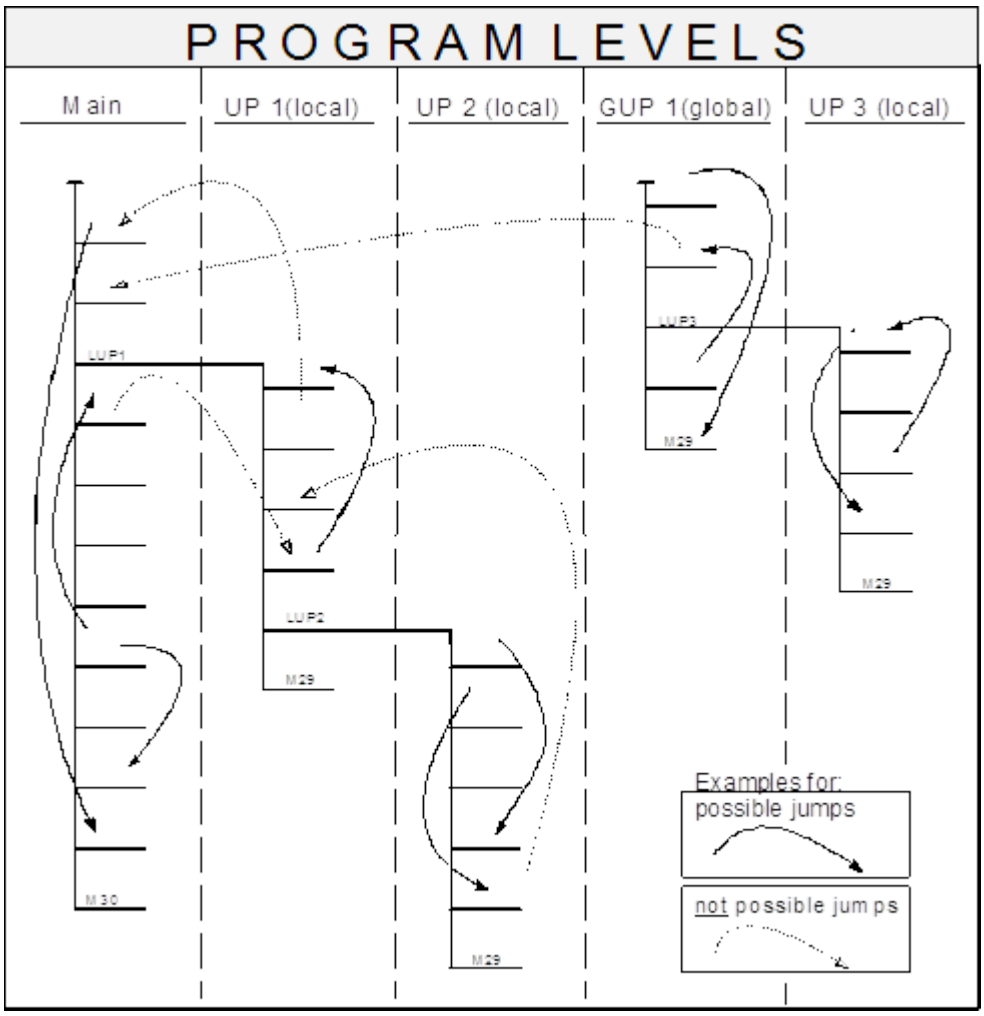


図 65: 図10-1: $GOTOジャンプコマンドの略図

プログラミング例

```
N10 G1 XY
N20:  X100                                (Label Definition N20: )
$IF V.L.dummy_1 <100 $GOTO N20           (Jump to Label N20 (or N20: ))
$IF V.L.dummy_1 >200
$GOTO [LABEL_1]                           (Jump to 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]    (Jump to 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 ...
...
```

11.1.3.1 パラメータによるジャンプ呼び出し

$GOTOコマンドでは、パラメータによってラベルをプログラムすることもできます。これにより、NCプログラムフローを外部(例えばPLCなど)から制御できます。

数式ラベルのジャンプ呼び出しでの形式がブロック番号<*expr*>の場合、パラメータ、ローカル変数、グローバル変数、および外部変数といった記述可能なすべての数式タイプが許可されます。

$GOTO N<*expr*>	ジャンプ呼び出し

文字列ラベルのジャンプ呼び出しは、String型およびString型配列の外部変数を使用してプログラムすることも可能です(「外部変数(V.E.) [▶ 443]」の章も参照)。ここでは、ラベル名が外部変数に格納されます。

$GOTO V.E. ...	ジャンプ呼び出し

プログラミング例

```
N10 ...
:
N50 $GOTO NV.E.JUMP_EXPR     (Jump e.g. to N200 via ext. variable)
                            (V.E.JUMP_EXPR including the value 200)
:
:
N100 $GOTO V.E.JUMP_STR      (Jump e.g. to [CONTINUE) via ext. variable)
                            (V.E.JUMP_STR including the value 200)
:
:
N200: ...
:
N500:...
:
:
:
[CONTINUE]...
:
N...
```

11.2 ループカウント

ループカウントによりn回のステートメント処理が可能になります。ループ回数はカウント変数によってチェックされます。

カウントループの構文は以下で始まり

$FOR P<*expr*> = <expr1>, <expr2>, <expr3>

必ず以下で終了します。

$ENDFOR

ここでは、P<*expr*>がカウント変数です。開始値は<*expr1*>、終了値は<*expr2*>、カウントインクリメントは<*expr3*>で指定されます。

カウント変数には整数値以外は使用できません。

小数を使用している場合には、加算時に丸め誤差が加算されるため、正確に増分(例外: 2のべき乗)を表すことは一般的に可能ではありません。これによって、少なすぎる1つのループを通して実行することにつながる可能性があります。

Pパラメータの位置には、書き込みアクセス可能な変数(「V」)、または[6]-6.44が設定されている場合はRパラメータ(「パラメータおよびパラメータ計算 [▶ 189]」)も使用できます。

カウントインクリメントが負の数であれば終了値を下回るとループが中止され、カウントインクリメントが正の数であれば終了値を超過するとループが中止されます。カウントインクリメントに0をプログラミングするとエンドレスループになり、警告が出力されます。

プログラミング例

```
N100 $FOR P1= 10, 100, 2   P1 is pre-allocated with 10 at loop beginning.
                           The counting loop is traversed till P1 has
                           exceeded the value 100, whereas P1 is
                           incremented by 2 at the end of every loop pass.
N110    X SIN [P1 * 5]     Within the counting loop, the NC blocks N110
                           to N130 are executed..
N120    Y COS [P1 * 5]
N130 ...
N150 $ENDFOR
```

プログラミング例

```
Negative step width:
N100 $FOR P1= 100, 10, -2  P1 is pre-allocated with 100 at the loop
                           beginning. The counting loop is traversed till
N110    X SIN [P1 * 5]     P1 has fallen below the value 10, whereas P1 is
                           decremented by 2 at the end of every loop pass.
                           Within the counting loops the NC blocks N110 to
                           N130 are executed.
N120    Y COS [P1 * 5]
N130 ...
N150 $ENDFOR
```

```
N100 $FOR P1= 100, 10, 1   P1 is pre-allocated with 100 at the loop
                           beginning. The counting loop is traversed till
                           P1 has exceeded the value 10, i.e. only once.
N110    X SIN [P1 * 5]     Within the counting loops the NC blocks N110 to
                           N130 are executed.
N120    Y COS [P1 * 5]
N130 ...
N150 $ENDFOR

Endless loop
N100 P2=20
N110 $FOR P1= 100, 10, 0   Endless loop
N120   $IF P2 == 50
N130     $BREAK
N140   $ENDIF
N150 $ENDFOR
```

11.3 実行条件付きループ

11.3.1 ループ開始時の実行条件の検証

このループの構文は以下で始まります。

$WHILE *<expr>*

必ず以下で終了します。

$ENDWHILE

各ループの先頭で、記述されたパラメータが検証されます。式<expr3>の値の範囲がFALSE (-0.5 <expr> 0.5)の場合には、ループが中止されます。

プログラム例

```
N90  P1 = 100.0
N100 $WHILE P1 > 0.5       P1 > 0.5 is tested for FALSE at the loop
N110 P1 = P1 - 1.5 Y P1    beginning. The loop is traversed till P1 ful-
N120 $ENDWHILE             fills the aborting condition
N130 ...
```

11.3.2 ループ終了時の実行条件の検証

使用可能な2種類のループがあります。

DOループの構文は以下で始まり

$DO

以下で終了します。

$ENDDO *<expr>*

各ループの終端で、記述されたパラメータが検証されます。式<expr>の値の範囲がFALSE (expr < 0.5)の場合、ループが中止されます。

REPEATループの構文は以下で始まります。

$REPEAT

以下で終了します。

$UNTIL *<expr>*

各ループの終端で、記述されたパラメータが検証されます。式<expr>の値の範囲がTRUE (expr > 0.5)の場合、ループが中止されます。

$WHILEおよび$FORループと比較し、$DOおよび$REPEATは必ず1回以上実行されます。

プログラミング例

```
N10 X0 Y0 Z0
N20 P2=10 P1=0
N30 $DO
N40   P1=P1+1
N50   XP1
N60 $ENDDO P1 <= P2         P1 is tested for FALSE at the end of the loop.
                            The loop is passed till P1 does not fulfill the
                            condition any longer.
N99 M30

N10 X0 Y0 Z0
N20 P2=10 P1=0
N30 $REPEAT
N40   P1=P1+1
N50   XP1
N60 $UNTIL P1 > P2          P1 is tested for TRUE at the end of the loop.
                            The loop is passed till P1 does fulfill the
                            condition.
N99 M30
```

11.4 ループフローシーケンスの効果

11.4.1 $BREAKステートメント

$BREAK

ループを中止基準によって抜けるのは、必ずしも推奨されません。プログラム実行とは別に、$SWITCHステートメントの$CASE文(「CASE分岐 [▶ 198]」の章を参照)内でキーワード$BREAKを使用してループの実行を中止できます。

例えば、深く入れ子構造にしたループの最も内部の実行を中止する場合に便利です。

プログラミング例

```
N10 $WHILE <expr1>          The loop is terminated if
N20 ...                     expr1 "not valid" or
N30                         expr2 "valid"
N40 $IF <expr2>
N50     $BREAK
N60 $ENDIF
N70 ...
N80
N90 $ENDWHILE
N100 ...
```

11.4.2 $CONTINUEステートメント

$CONTINUE

$BREAKとは異なり、ループは$CONTINUEステートメントでは中止されませんが、ループの先頭に分岐されます。このため、$CONTINUEの後に記述されたすべてのステートメントが実行されません。

プログラミング例

```
N10 $FOR <expr1>            N70 and N80 are executed if
N20 ...                     expr2  is "not valid
N30
N40   $IF <expr2>
N50     $CONTINUE
N60   $ENDIF
N70 ...
```

```
N80
N90 $ENDFOR
N100 ...
```

12 特殊関数

特殊関数の構文は以下の通りです。

```
#<string> <spezific additional syntax>
```

#<string> いわゆる文字列コマンド。#と<string>の間には空白文字を入力してはならないことに注意が必要です。

<spezific additional syntax> 直接、または割り当てられたカッコ内にプログラムされるコマンド固有の構文要素。

注記

#コマンドは、1つのブロック内に1つのみしかプログラムしてはなりません。例外は個別に記載します。

特に記載がない場合には、個別の追加構文カンマ「,」と等号「=」はオプションです。これらは、NCパートプログラムを読みやすくする目的のみに使用します。

例:

#STRING [VALUE_A 10 VALUE_B 20] ←→ #STRING [VALUE_A=10, VALUE_B=20]

12.1 軸交換コマンド

この項では、以下のNCコマンドについて記載します。

- 軸の要求
- 軸の解放
- 軸設定の決定

が記載されています。

各プログラムが開始中に、指定された[1]-5軸設定が格納されます。この軸交換コマンドは、現在の軸グループ内で有効です。NCブロック内では、軸の複数の要求や応答が可能です。これらの操作は、半並行状態で行われます。

- 同期操作や手動操作が有効な軸は交換できません。
- 工具径補正(TRC)が有効な場合は、先頭の2つの主軸は交換してはなりません。
- 軸交換コマンドは、1つのNCブロック内に1つのみしかプログラムしてはなりません。
- 軸交換コマンドは補間軸および結合軸にのみ使用できます。スピンドル軸は無視されます。スピンドル軸の解放と要求には、特殊コマンド [▶ 482]が用意されています。

バージョンV2.6以降では、既存の軸交換コマンドが改良され、機能とメカニズムが拡張されています。「標準軸交換コマンド」の章では、旧バージョンの既存の標準構文について説明されています。この構文は下位互換性があり、今後も使用できます。

「拡張された軸交換コマンド」の章では、新しい構文について説明されています。この構文は既存の対象機能に対する下位互換性がありますが、いわゆる論理スイッチや各種軸交換シーケンス定義方法により、より柔軟なプログラミングが可能です。

12.1.1 標準構文

12.1.1.1 軸の要求

NCチャンネルの軸グループ内に既に存在している軸を要求しようとしても、軸管理機能から要求することはできません。

このNCコマンドを使用すると、軸管理機能から軸を要求できます。

#CALL AX[<*mode*>] **[<axis_name>,<axis_number>,<axis_index>** {,<*options*>}]
{ **[<axis_name>,<axis_number>,<axis_index>** {,<*options*>}] } (ノンモーダル)

<*mode*> インターポレータからの軸位置要求、および軸交換中のNCチャンネル位置の初期化の有効/無効

モード	NCチャンネル内の軸の要求
	初期設定: インターポレータから設定値を要求し、NCチャンネルの位置を初期化します。
FAST	インターポレータから設定値を要求とNCチャンネルの位置を初期化を行いません。

<axis_name> 文字A、B、C、Q、U、V、W、X、Y、およびZで始まる文字列を使用できます。異なる軸に対して同じ名前を割り当てる(論理軸番号による識別)とエラーメッセージが出力され、NCプログラムが中止されます。

<axis_number> 論理軸番号によって、軸を物理的に割り当てます。数式が使用できます。軸管理で論理軸番号が認識されている必要があります。認識されていない論理軸番号、または複数の同一の論理軸番号を要求するとエラーメッセージが出力され、NCプログラムが中止されます。

<axis_index> 軸インデックスは、NCチャンネルの現在の軸グループ内の軸位置を決定します。軸インデックスは、主軸および同期軸を定義します(以下の表を参照)。結果が[0～最大軸番号-1]の値の範囲となる数式を使用できます。ただし、軸インデックスは軸内に割り当ててはなりません。他の軸に割り当てられているインデックスの割り当てを要求するとエラーメッセージが出力され、NCプログラムが中止されます。

インデックス	軸設定
0	1. 加工平面内の主軸
1	2. 加工平面内の主軸
2	3. 通常、加工平面に直交する主軸
3	1. 同期軸
~n	(n-2)。同期軸。

注記

プログラミングを簡略化するために、**結合軸**の軸インデックスの入力は省略できます(#CALL AX...の場合のみ)。この場合、インデックス3の次に空いている軸インデックスがこの結合軸に自動的に割り当てられます。

主軸の場合には、このインデックスを**常に**明示的にプログラミングする必要があります。

ただし、結合軸のインデックスは様々な機能があるため重要です。例えば、キネマティックトランスフォーメーション(RTCP)の場合、主軸との隙間を作らないようにすべての変換軸を配置する必要があります。このような場合、結合軸に対しても軸インデックスを明示的にプログラムする必要があります。

<options> オフセットが軸ごとに保持されます。ここでは、以下が含まれます。

- 基準点オフセット
- 機械テーブルオフセット
- 工具オフセット
- ゼロオフセット
- 計測オフセット
- 手動操作オフセット
- プリセットオフセット

軸の要求中に、以下の表のキーワードを使用してオフセットを考慮するかどうかを制御できます。

工具オフセット以外のすべてのオフセットは、軸の解放または呼び出し中に、必ず論理軸番号に紐付けられます(以下のプログラミング例「Zero_offset」を参照)。

キーワード	軸の要求
	オフセットを考慮しない(初期設定)
ALL	すべてのオフセットを考慮する *
BPV	基準点オフセットを考慮する
PZV	機械テーブルオフセットを考慮する
WZV	工具オフセットを考慮する *
NPV	ゼロオフセットを考慮する
MOFFS	計測オフセットを考慮する
SOFFS	手動操作オフセットを考慮する
PSET	プリセットオフセットを考慮する

注記

* #CALL AXの場合は、工具がキャンセルされた場合は工具オフセットの考慮のみ有効になります。チャンネル内で工具が有効になるか、または工具交換後に#CALL AXが選択された場合、適用されている工具オフセットが現在の工具のオフセットで置換されます。

工具選択に関しては、オフセットが常に工具データにインデックスを付けてシーケンスに準拠した軸の計算に置かれているという事実に注意を払います。

プログラミング例

```
%Zero_Offset
N010 X200
N015 G54
N015 V.G.NP_AKT.V.X = 11
N016 X0                      (Final machine X-axis position is 11)
N020 #PUT AX [X]             (Release axis X with log. axis number 1)
.....
N130 #CALL AX [X1,1,0,NPV]   (Call log. axis number 1 under new name X1)
N140 X1=100                  (Final machine X1-axis position is 111)
M30
```

例:

プログラム開始時の軸名、論理軸番号、および軸インデックスの割り当て

軸名	論理軸番号	軸インデックス
X	1	0
Y	2	1
Z	3	2

プログラミング例

```
%ACHSTAUSCH1

N10 #CALL AX FAST [X1,7,4]      (X1-axis without set value -request)
                                (and output of Init.-function block)
```

軸要求後の軸名、論理軸番号、および軸インデックスの割り当て

軸名	論理軸番号	軸インデックス
X	1	0
Y	2	1
Z	3	2
		3
X1	7	4

プログラミング例の続き:

```
:
N100#CALL AX [Y1,8,6] [C,9,3]   (request for Y1- and C-axis, axis index
                                of C-axis is determined automatically)
```

2番目の軸要求後の軸名、論理軸番号、および軸インデックスの割り当て

軸名	論理軸番号	軸インデックス
X	1	0
Y	2	1
Z	3	2
C	9	3
X1	7	4
		5
Y1	8	6

プログラミング例の続き:

```
:
N1000 #CALL AX FAST [Z1,13,5,ALL]   (Inclusion of all shifts)
N1010 #CALL AX [C1,11,7,NPV MOFFS]  (Inclusion of zero offsets)
                                    (and of measurement offsets)
```

3番目の軸要求後の軸名、論理軸番号、および軸インデックスの割り当て

軸名	論理軸番号	軸インデックス
X	1	0
Y	2	1
Z	3	2
C	9	3
X1	7	4
Z1	13	5
Y1	8	6
C1	11	7

12.1.1.2 軸の解放

これらのNCコマンドを使用し、NCチャンネルの現在の軸グループの軸を軸管理に戻せます。存在しない軸を戻すことはできないため、戻すとエラーメッセージが表示されます。

#PUT AX [<*axis_name*>{,<*axis_name*>}**]**	(ノンモーダル)

<axis_name> 文字A、B、C、Q、U、V、W、X、Y、およびZで始まる文字列を使用できます。

このNCコマンドを使用し、NCチャンネルの現在の軸グループ内に存在する**すべての**軸が軸管理に戻されます。

#PUT AX ALL	(ノンモーダル)

例:

プログラム開始時の軸名、論理軸番号、および軸インデックスの割り当て

軸名	論理軸番号	軸インデックス
X	1	0
Y	2	1
Z	3	2

プログラミング例

```
%ACHSTAUSCH1

N10 #PUT AX [ X , B ]   (Release X-axis; B-axis not present,)
                        (nevertheless no error message is output.)
```

軸解放後の軸名、論理軸番号、および軸インデックスの割り当て

軸名	論理軸番号	軸インデックス
		0
Y	2	1
Z	3	2

プログラミング例の続き:

```
...
N100 #PUT AX ALL      (Releasing of all axes of this group.)
```

2番目の軸解放後の軸名、論理軸番号、および軸インデックスの割り当て

軸名	論理軸番号	軸インデックス
		0
		1
		2

12.1.1.3 軸設定の定義

このNCコマンドを使用して、既存の軸設定に代わる新しい軸設定を定義できます。つまり、NCコマンド内でプログラムされた軸によって、NCチャンネルの新しい軸設定が構成されます。

#SET AX[<*mode*>] **[<axis_name>,<axis_number>,<axis_index>** {,<*options*>} **]**
{ **[<axis_name>,<axis_number>,<axis_index>** {,<*options*>} **]** } （ノンモーダル）

<mode > インターポレータからの軸位置要求の有無、および軸交換中のNCチャンネル位置の初期化。

モード	NCチャンネル内の軸の置換
	初期設定: インターポレータから設定値を要求し、NCチャンネルの位置を初期化します。
FAST	インターポレータから設定値を要求しません。NCチャンネルの位置を初期化します。

<axis_name > 文字A、B、C、Q、U、V、W、X、Y、およびZで始まる文字列を使用できます。異なる軸に対して同じ名前を割り当てる(論理軸番号による識別)とエラーメッセージが出力され、NCプログラムが中止されます。

<axis_number > 論理軸番号によって、軸を物理的に割り当てます。数式が使用できます。軸管理で論理軸番号が認識されている必要があります。認識されていない論理軸番号、または複数の同一の論理軸番号を要求するとエラーメッセージが出力され、NCプログラムが中止されます。

<axis_index > 軸インデックスは、NCチャンネルの軸グループ内の軸位置を決定します。これにより、軸インデックスは主軸および同期軸を定義します(以下の表を参照)。結果が[0～最大軸番号-1]の値の範囲となる数式を使用できます。ただし、軸インデックスは軸内に割り当ててはなりません。他の軸に割り当てられているインデックスの割り当てを要求するとエラーメッセージが出力され、NCプログラムが中止されます。

インデックス	軸設定
0	1. 加工平面内の主軸
1	2. 加工平面内の主軸
2	3. 通常、加工平面に直交する主軸
3	1. 同期軸
...	...
n	(n-2)。同期軸。

<options> オフセットが軸ごとに保持されます。ここでは、以下が含まれます。

- 基準点オフセット
- 機械テーブルオフセット
- 工具オフセット
- ゼロオフセット
- 計測オフセット
- 手動操作オフセット
- プリセットオフセット

軸の要求中に、以下の表のキーワードを使用してオフセットを考慮するかどうかを制御できます。

キーワード	軸の交換
	オフセットを考慮しない(初期設定)
ALL	すべてのオフセットを考慮する *
BPV	基準点オフセットを考慮する
PZV	機械テーブルオフセットを考慮する
WZV	工具オフセットを考慮する *
NPV	ゼロオフセットを考慮する
MOFFS	計測オフセットを考慮する
SOFFS	手動操作オフセットを考慮する
PSET	プリセットオフセットを考慮する

注記

*** ツールが選択**されている場合は、#SET AXでの工具オフセット適用時に次の事項を考慮する必要があります。

- #SET AXによってのみ軸が交換される場合(内部軸交換)、または補助軸が指定されていないか、または要求されていない場合には、すべてのオフセット(工具オフセットを含む)も交換され、有効な状態が継続します。オフセット適用にキーワードを指定しても効果はありません。
  その後に新しい工具が選択された場合、オフセットも新しい工具のオフセットに置換されます。
- #SET AXによって軸の解放、または軸の要求がトリガされると(外部軸交換)、工具データ内にインデックスされた軸シーケンスの計算に再度工具オフセットが含まれるようになります。このため、適用された工具オフセットがすべて現在の工具オフセットで置換されます! 対応する軸内の元の工具オフセットが引き続き適用される場合は、オフセットが新しい軸配置に適用されるように新しい工具を選択する必要があります。

⇨ このため、解除された工具で#SET AXを実行し、新しく選択された工具の記録内で適切にパラメータ設定することで工具オフセットを正しく割り当てることを推奨します。

例:

工具データ内の工具オフセットのインデックス	[0]	[1]	[2]	[3]
パラメータ化された工具オフセット(例えばT1)	50	0	70	20
プログラム開始時での軸設定	X	Y	Z	---
T1選択後に適用される工具オフセット	50	0	70	---
「内部的な」#SET AX {Z, X, Y}:	**Z**	**X**	**Y**	---
工具オフセットが交換されます。	*70*	*50*	*0*	---
「外部的な」#SET AX {Z, X, Y, B}:	**Z**	**X**	**Y**	**B**
T1に対応する工具オフセットが新しく再計算されます。	*50*	*0*	*70*	*20*

例:

プログラム開始時の軸名、論理軸番号、および軸インデックスの割り当て

軸名	論理軸番号	軸インデックス
X	1	0
Y	2	1
Z	3	2

プログラミング例

```
(Setting of axis configuration:

(X-axis remains in its place;)
(Y-axis is released;)
(Z-axis is resorted acc. to Index 4;)
(Y1- and Z1-axis are requested)

%ACHSTAUSCH1

N10 #SET AX [X,1,0][Y1,4,2][Z1,5,3][Z,3,4]
```

N10後の軸名、論理軸番号、および軸インデックスの割り当て

軸名	論理軸番号	軸インデックス
X	1	0
		1
Y1	4	2
Z1	5	3
Z	3	4

12.1.2 拡張構文

このファンクションを使用できるかどうかは、設定やバージョンの具体的な範囲によって異なります。

拡張構文により、マクロ定義(第15 [▶ 505]章)やString型の外部変数(V.E....)で軸交換シーケンスをプログラミングできます。これはチャンネルが複数ある機械およびシステムで、チャンネル間で静的な軸グループを交換する必要がある場合に特に便利です。この軸グループは、例えば軸交換コマンド内で使用するマクロ内で定義可能です。

さらに、拡張構文でいわゆる論理スイッチを設定することにより、エラーメッセージや警告を出力せずに、競合の内部処理が可能になります。オプションのこの論理スイッチは、軸交換コマンド内に追加でプログラムできます。論理スイッチがプログラムされていない場合には、標準的な評価が有効になります。これにより、妥当性が競合した場合にエラーメッセージが出力され、軸交換シーケンスの評価が中止されます。

すべての軸交換コマンドで同じ論理評価が使用され、**プログラムされた軸交換シーケンス内**での妥当性、および**NCチャンネル内で既に有効な軸に対する**妥当性の両方がチェックされます。

論理スイッチは、マクロや文字列変数経由で定義された軸変換シーケンスで特に便利です。これは、重複や冗長なプログラミングが内部的に解消されるためです。

12.1.2.1 軸の要求

このNCコマンドを使用すると、軸管理からの軸を要求できます。

#AX REQUEST [<*mode*>] [**NAM**, **NBR**, **IDX**] [<*axis_exchange_sequence*> {,<*options*>}]
{ [<*axis_exchange_sequence*> {,<*options*>}] } (ノンモーダル)

<*mode*> インターポレータからの軸位置要求の有無、および軸交換中のNCチャンネル位置の初期化。

モード	NCチャンネル内の軸の要求
	初期設定: インターポレータから設定値を要求し、NCチャンネルの位置を初期化します。
FAST	インターポレータから設定値を要求しません。NCチャンネルの位置を初期化します。

NAM、NBR、IDX 競合を処理するための論理スイッチ。

ID	優先度
NAM	冗長な軸名の処理
NBR	冗長な軸番号の処理
IDX	冗長な軸インデックスの処理

論理スイッチは、個別でも組み合わせてもプログラム可能です!

<*axis_exchange_sequence*>は、以下で構成されます。

<*axis_name*> 文字A、B、C、Q、U、V、W、X、Y、およびZで始まる文字列を使用できます。

プログラムされた軸交換シーケンス内に競合が存在する場合:

冗長な軸名 → エラー、NCプログラムが中止

NCチャンネル内の既存の軸と競合する場合:

軸名が同じで軸番号が異なる → エラー、NCプログラムの中止

論理スイッチNAMが有効な場合、競合は以下のように解決されます。

軸がその軸パラメータリストP-AXIS-00297からデフォルト名を取得します。 ユーザは、リスト内の既定の軸名の明確な定義を確実に行う必要があります。

<axis_number> 論理軸番号によって、軸を物理的に割り当てます。数式が使用できます。軸管理で論理軸番号が認識されている必要があります。

プログラムされた軸交換シーケンス内に競合が存在する場合:

冗長な軸名 → エラー、NCプログラムが中止

NCチャンネル内の既存の軸と競合する場合:

NCチャンネル内に軸番号が既に存在する → 警告

論理スイッチNBRが有効な場合、競合は以下のように解決されます。

軸要求が無視され、実行されないことを意味します!

<axis_index> 軸インデックスは、NCチャンネルの現在の軸グループ内の軸位置を決定します。軸インデックスは、主軸および同期軸を定義します(以下の表を参照)。結果が[0 ~ 最大軸番号-1]の値の範囲となる数式を使用できます。ただし、軸インデックスは軸内に割り当ててはなりません。

インデックス	軸設定
0	1. 加工平面内の主軸
1	2. 加工平面内の主軸
2	3. 通常、加工平面に直交する主軸
3	1. 同期軸
~n	(n-2)。同期軸。

注記

プログラミングを簡略化するために、**結合軸**の軸インデックスの入力は省略できます。この場合、インデックス3の次に空いている軸インデックスがこの結合軸に自動的に割り当てられます。

主軸の場合には、このインデックスを**常に**明示的にプログラミングする必要があります。

ただし、結合軸のインデックスは様々な機能があるため重要です。例えば、キネマティックトランスフォーメーション(RTCP)の場合、すべての変換軸を主軸との隙間を作らないように配置する必要があります。このような場合、結合軸に対しても軸インデックスを明示的にプログラムする必要があります。

プログラムされた軸交換シーケンス内に競合が存在する場合:

冗長な軸名 → エラー、NCプログラムが中止

NCチャンネル内の既存の軸と競合する場合:

軸インデックスがNCチャンネル内で既に使用されていて、異なる軸名 → エラー、NCプログラムが中止

論理スイッチIDXが有効な場合、競合は以下のように解決されます。

軸に対して、NCチャンネルの軸設定の次の空きインデックスが自動的に特定されます。

<options> オフセットが軸ごとに保持されます。ここでは、以下が含まれます。

- 基準点オフセット
- 機械テーブルオフセット
- 工具オフセット
- ゼロオフセット
- 計測オフセット

- 手動操作オフセット
- プリセットオフセット

軸の要求中に、以下の表のキーワードを使用してオフセットを考慮するかどうかを制御できます。

工具オフセット以外のすべてのオフセットは、軸の解放または要求中に必ず論理軸番号に紐付けられます。

キーワード	軸の要求
	オフセットを考慮しない(初期設定)
ALL	すべてのオフセットを考慮する *
BPV	基準点オフセットを考慮する
PZV	機械テーブルオフセットを考慮する
WZV	工具オフセットを考慮する *
NPV	ゼロオフセットを考慮する
MOFFS	計測オフセットを考慮する
SOFFS	手動操作オフセットを考慮する
PSET	プリセットオフセットを考慮する

注記

* #AX REQUESTで、工具がキャンセルされた場合は、工具オフセットのみが考慮されます。チャンネル内で工具が有効になる、または工具交換後に#AX REQUESTが選択された場合、適用されている工具オフセットが現在の工具のオフセットで置換されます。

工具選択に関しては、オフセットが常に工具データにインデックスを付けてシーケンスに準拠した軸の計算に置かれているという事実に注意してください。

例1:

標準機能の使用

プログラム開始時の軸名、論理軸番号、および軸インデックスの割り当て

軸名	論理軸番号	軸インデックス
X	1	0
Y	2	1
Z	3	2

プログラミング例

```
%Axis_exchange1
N10 #AX REQUEST [X1,7,4]        (Request for X1-axis)
```

軸要求後の軸名、論理軸番号、および軸インデックスの割り当て

軸名	論理軸番号	軸インデックス
X	1	0
Y	2	1
Z	3	2
		3
X1	7	4

プログラミング例の続き:

```
:
N100 #AX REQUEST [Y1,8,6] [C,9, ]   (Request for Y1- and C-axis, C-axis)
                                    (is set automatically on index 3)
```

2番目の軸要求後の軸名、論理軸番号、および軸インデックスの割り当て

軸名	論理軸番号	軸インデックス
X	1	0
Y	2	1
Z	3	2
C	9	3
X1	7	4
		5
Y1	8	6

プログラミング例の続き:

```
:
N1000 #AX REQUEST FAST [Z1,13,5,ALL]     (Inclusion of all shifts)
N1010 #AX REQUEST [C1,11,7,NPV MOFFS]    (Inclusion of zero offsets)
                                         (and of measurement offsets)
```

3番目の軸要求後の軸名、論理軸番号、および軸インデックスの割り当て

軸名	論理軸番号	軸インデックス
X	1	0
Y	2	1
Z	3	2
C	9	3
X1	7	4
Z1	**13**	**5**
Y1	8	6
C1	**11**	**7**

例2:

追加構文の使用(論理スイッチ)

チャンネル1でのプログラム開始時の軸名、論理軸番号、および軸インデックスの割り当て

軸名	論理軸番号	軸インデックス
X	1	0
Y	2	1
Z	3	2

チャンネル2でのプログラム開始時の軸名、論理軸番号、および軸インデックスの割り当て

軸名	論理軸番号	軸インデックス
X	7	0
Y	8	1
Z	9	2

プログラミング例

チャンネル1内: チャンネル2 (論理軸番号7および8)からのX軸およびY軸の要求。

```
%Axis_exchange
N10 #AX REQUEST NAM [X,7,3] [Y,8,4]   (Request of X/Y-axes)
```

チャンネル1内の論理スイッチNAMにより、軸リストの初期設定名(X2およびY2など)で新しくXおよびYが追加されます。

軸要求後の軸名、論理軸番号、および軸インデックスの割り当て

軸名	論理軸番号	軸インデックス
X	1	0
Y	2	1
Z	3	2
X2	7	3
Y2	8	4

プログラミング例の続き:

チャンネル1内: チャンネル2 (論理軸番号9)からのZ軸の要求。

```
:
N100 #AX REQUEST NAM IDX [Z,9,2]   (Request of Z-axis)
```

チャンネル内の論理スイッチNAMおよびIDXにより、 軸リストの初期設定名(Z2など)で次の空いているインデックス(インデックス5など)に新しくZ軸が追加されます。

2番目の軸要求後の軸名、論理軸番号、および軸インデックスの割り当て

軸名	論理軸番号	軸インデックス
X	1	0
Y	2	1
Z	3	2
X2	7	3
Y2	8	4
Z2	9	5

例3:

追加構文の使用(論理スイッチおよび文字列としての軸交換シーケンス)

チャンネル1でのプログラム開始時の軸名、論理軸番号、および軸インデックスの割り当て

軸名	論理軸番号	軸インデックス
X	1	0
Y	2	1
Z	3	2

チャンネル2でのプログラム開始時の軸名、論理軸番号、および軸インデックスの割り当て

軸名	論理軸番号	軸インデックス
X_1	7	0
Y	8	1
Z	9	2
A	10	3
B	11	4

プログラミング例

チャンネル1内: チャンネル2 (論理軸番号7、8および11)からのX_1軸、Y軸およびB軸の要求。軸交換シーケンスはマクロ内に格納されます。

```
%Axis_exchange
N05 "ACHSEN_KANAL2" = "[X_1,7,0] [Y,8,1] [B,11,2]"
:
N10 #AX REQUEST NAM IDX "ACHSEN_KANAL2"    (Request axes)
```

チャンネル1内の論理スイッチNAMおよびIDXにより、新しい追加軸が正常に組み込まれます。

軸要求後の軸名、論理軸番号、および軸インデックスの割り当て

軸名	論理軸番号	軸インデックス
X	1	0
Y	2	1
Z	3	2
X_1	7	3
Y2	8	4
B	11	5

12.1.2.2 軸の解放

これらのNCコマンドを使用し、NCチャンネルの現在の軸グループの軸を軸管理に戻せます。存在しない軸を戻すことはできないため、戻すとエラーメッセージが表示されます。

#AX RELEASE [< *axis_name* >{,< *axis_name* >} **]**	(ノンモーダル)

<axis_name> NCチャンネル内に現在存在している軸の軸名。

論理スイッチNBRを使用すると、軸名から論理軸番号に変更できます(解放時に軸名が認識されていない場合など)。

#AX RELEASE [NBR] [<*expr*> {,<*expr*> }] (ノンモーダル)

<*expr*> 論理軸番号。

このNCコマンドを使用し、NCチャンネルの現在の軸グループ内に存在する**すべての**軸が軸管理に戻されます。

#AX RELEASE ALL (ノンモーダル)

例:

プログラム開始時の軸名、論理軸番号、および軸インデックスの割り当て

軸名	論理軸番号	軸インデックス
X	1	0
Y	2	1
Z	3	2
A	4	3
B	5	4

プログラミング例

```
%Axis_exchange
N10 #AX RELEASE[X, A]      (Release X/A-axes)
```

軸解放後の軸名、論理軸番号、および軸インデックスの割り当て

軸名	論理軸番号	軸インデックス
Y	2	1
Z	3	2
B	5	4

プログラミング例の続き:

```
...
N100 #AX RELEASE NBR[2]    (Release Y-axis)
```

2番目の軸解放後の軸名、論理軸番号、および軸インデックスの割り当て

軸名	論理軸番号	軸インデックス
Z	3	2
B	5	4

プログラミング例の続き:

```
...
N100 #AX RELEASE ALL   (Release all present axes from this NC channel)
```

3番目の軸解放後の軸名、論理軸番号、および軸インデックスの割り当て

軸名	論理軸番号	軸インデックス

12.1.2.3 軸設定の定義

このNCコマンドを使用して、既存の軸設定に代わる新しい軸設定を定義できます。つまり、NCコマンド内でプログラムされた軸が、NCチャンネルの新しい軸設定を構成します。

#AX DEF <*mode*>] [**NAM**, **NBR**, **IDX**] [<*axis_exchange_sequence*> {,<*options*>}]
{ [<*axis_exchange_sequence* > {,<*options*>}] } (ノンモーダル)

<*mode*> インターポレータからの軸位置要求の有無、および軸交換中のNCチャンネル位置の初期化。

モード	NCチャンネル内の軸の置換
	初期設定: インターポレータから設定値を要求し、NCチャンネルの位置を初期化します。
FAST	インターポレータから設定値を要求しません。NCチャンネルの位置を初期化します。
OPT_FAST	高速オプション。高速プログラム開始が命令されている場合にのみ、効果があります。それ以外の場合は、設定値が要求されます。

NAM、NBR、IDX 競合を処理するための論理スイッチ。

ID	優先度
NAM	冗長な軸名の処理
NBR	冗長な軸番号の処理
IDX	冗長な軸インデックスの処理

論理スイッチは、個別でも組み合わせてもプログラム可能です!

<*axis_exchange_sequence*>は、以下で構成されます。

<*axis_name*> 文字A、B、C、Q、U、V、W、X、Y、およびZで始まる文字列を使用できます。

プログラムされた軸交換シーケンス内に競合が存在する場合:

冗長な軸名 → エラー、NCプログラムが中止

NCチャンネル内の既存の軸と競合する場合:

軸名が同じで軸番号が異なる → エラー、NCプログラムの中止

論理スイッチNAMが有効な場合、競合は以下のように解決されます。

軸はその軸パラメータリストP-AXIS-00297からデフォルト名を取得します。 ユーザは、リスト内の既定の軸名の定義を明確かつ確実に行う必要があります。

<*axis_number*> 論理軸番号によって、軸を物理的に割り当てます。数式が使用できます。軸管理で論理軸番号が認識されている必要があります。

プログラムされた軸交換シーケンス内に競合が存在する場合:

冗長な軸名 → エラー、NCプログラムが中止

NCチャンネル内の既存の軸と競合する場合:

NCチャンネル内に軸番号が既に存在する → 警告

論理スイッチNBRが有効な場合、競合は以下のように解決されます。

軸要求が無視され、実行されないことを意味します!

<axis_index> 軸インデックスは、NCチャンネルの現在の軸グループ内の軸位置を決定します。軸インデックスは、主軸および同期軸を定義します(以下の表を参照)。結果が[0～最大軸番号-1]の値の範囲となる数式を使用できます。ただし、軸インデックスは軸内に割り当ててはなりません。

インデックス	軸設定
0	1. 加工平面内の主軸
1	2. 加工平面内の主軸
2	3. 通常、加工平面に直交する主軸
3	1. 同期軸
...	...
n	(n-2)。同期軸。

注記

プログラミングを簡略化するために、**結合軸**の軸インデックスの入力は省略できます。この場合、インデックス3の次に空いている軸インデックスがこの結合軸に自動的に割り当てられます。

主軸の場合には、このインデックスを**常に**明示的にプログラミングする必要があります。

ただし、結合軸のインデックスには様々な機能があるために重要です。例えば、キネマティックトランスフォーメーション(RTCP)の場合、すべての変換軸は主軸との隙間を作らないように配置する必要があります。このような場合、結合軸に対しても軸インデックスを明示的にプログラムする必要があります。

プログラムされた軸交換シーケンス内に競合が存在する場合:

冗長な軸名 → エラー、NCプログラムが中止

NCチャンネル内の既存の軸と競合する場合:

軸インデックスがNCチャンネル内で既に使用されていて、異なる軸名 → エラー、NCプログラムが中止

論理スイッチIDXが有効な場合、競合は以下のように解決されます。

軸に対して、NCチャンネルの軸設定の次の空きインデックスが自動的に特定されます。

<options> オフセットが軸ごとに保持されます。ここでは、以下が含まれます。

- 基準点オフセット
- 機械テーブルオフセット
- 工具オフセット
- ゼロオフセット
- 計測オフセット
- 手動操作オフセット
- プリセットオフセット

軸の要求中に、以下の表のキーワードを使用してオフセットを考慮するかどうかを制御できます。

キーワード	軸の交換
	オフセットを考慮しない(初期設定)
ALL	すべてのオフセットを考慮する *
BPV	基準点オフセットを考慮する
PZV	機械テーブルオフセットを考慮する
WZV	工具オフセットを考慮する *
NPV	ゼロオフセットを考慮する
MOFFS	計測オフセットを考慮する
SOFFS	手動操作オフセットを考慮する
PSET	プリセットオフセットを考慮する

注記

***ツールが選択**されている場合は、#SET AXでの工具オフセット適用時に次の事項を考慮する必要があります。

- #SET AXによってのみ軸が交換される場合(内部軸交換)、または補助軸が指定されていないか、または要求されていない場合には、すべてのオフセット(工具オフセットを含む)も交換され、有効の状態が継続します。オフセット適用にキーワードを指定しても効果はありません。
  その後に新しい工具が選択された場合、オフセットも新しい工具のオフセットに置換されます。
- #AX DEFによって軸の解放、または軸の要求がトリガされると(外部軸交換)、工具データ内にインデックスが付けられた軸シーケンスの計算に、再度工具オフセットが含まれるようになります。このため、適用された工具オフセットがすべて現在の工具オフセットで置換されます! 対応する軸内の元の工具オフセットが引き続き適用される場合は、オフセットが新しい軸配置に適用されるように新しい工具を選択する必要があります。

⇨ このため、解除された工具で#AX DEFを実行し、新しく選択された工具の記録内で適切にパラメータ設定することにより、工具オフセットを正しく割り当てることを推奨します。

例:

工具データ内の工具オフセットのインデックス	[0]	[1]	[2]	[3]
パラメータ化された工具オフセット(例えばT1)	50	0	70	20
プログラム開始時での軸設定	X	Y	Z	---
T1選択後に適用される工具オフセット	50	0	70	---
「内部的な」#AX DEF {Z, X, Y}:	**Z**	**X**	**Y**	---
工具オフセットが交換されます。	*70*	*50*	*0*	---
「外部的な」#AX DEF {Z, X, Y, B}:	**Z**	**X**	**Y**	**B**
T1に対応する工具オフセットが新しく再計算されます。	*50*	*0*	*70*	*20*

DEFAULTをプログラミングすることにより、チャンネルパラメータリスト[1]-5に対応する初期設定を強制できます。論理スイッチと追加の軸要求を組み合わせることも可能です。

# AX DEF DEFAULT	(ノンモーダル)
または	
# AX DEF DEFAULT [NAM, NBR, IDX] { [<*axis_exchange_sequence*> {,<*options*>}] }	(ノンモーダル)

例:

プログラム開始時の軸名、論理軸番号、および軸インデックスの割り当て

軸名	論理軸番号	軸インデックス
X	1	0
Y	2	1
Z	3	2

プログラミング例

```
(Setting of axes configuration)
(X-axis remains on ist place; Y-axis is released;)
(Z-axis is resorted acc. to Index 4;(Y1- and Z1-axis are)
(requested;)

%Axis_exchange
N10 #AX DEF [X,1,0][Y1,4,2][Z1,5,3][Z,3,4]
:
```

N10後の軸名、論理軸番号、および軸インデックスの割り当て

軸名	論理軸番号	軸インデックス
X	1	0
		1
Y1	**4**	**2**
Z1	**5**	**3**
Z	**3**	**4**

初期軸設定の復元:

```
%Axis_exchange
N10 #AX DEF DEFAULT
:
```

N10後の軸名、論理軸番号、および軸インデックスの割り当て

軸名	論理軸番号	軸インデックス
X	1	0
Y	2	1
Z	3	2

12.2 ドウェル - 機能

#TIME *<expr>*	(ノンモーダル)

「ドウェル(G04)、(#TIME) [▶ 54]」の章を参照してください。

12.3 NCチャンネルのフラッシング

注記

有効な工具径補正、多項式輪郭加工、HSCモードなどの形状変更ファンクションの使用中は、NCチャンネルのフラッシングに関するコマンド(#FLUSH、#FLUSH CONTINUE、#FLUSH WAIT)を使用できません。

バージョンV2.10.1503.08以降、有効な多項式輪郭加工中のNCチャンネルのフラッシングに関するコマンドを使用することが許可されています。この場合、輪郭加工は移動ブロックでNCチャンネルのフラッシングに関するコマンドの前後に停止します。

#FLUSH (ノンモーダル)

コマンド#FLUSHにより、現在NCチャンネル内に格納されているすべてのNCブロックが実行されます。これは、NCチャンネルのバッファ効果が除去されることを意味します。#FLUSHの前のプログラムされた最後の移動ブロックが同時に実行されます。

通常はルックアヘッド内に次の移動ブロックが存在している場合でも、この移動ブロックの終端で動作が停止します。ただし、パートプログラムの解読は停止せずに継続します。

プログラミング例

ステートメント#FLUSHがプログラムされていない場合、ルックアヘッドは次の移動ブロックが出現するまで直線動作の解釈を待機します。ただし、その後にWHLEループにジャンプするため、移動動作の実行が遅延する可能性があります。このため、#FLUSHによってルックアヘッドの効果が無効になり、WHILEループへのジャンプ前に移動ブロックが実行されます。

```
N10   G01  X20
N20   #FLUSH
N30   $WHILE 1
...
Nnn   $ENDWHILE
M30
```

#FLUSH CONTINUE (ノンモーダル)

CONTINUEを追加すると、移動動作のブロック終端での停止を回避できます。ルックアヘッド内に次の移動ブロックが存在する場合、動作は中断せずに実行されます。次の移動ブロックが存在しない場合には、動作が停止します。

プログラミング例

移動ブロックN10とN200の間に、動作に関連しない多くのNCブロックがプログラムされています(さまざまなパラメータ計算、変数制限など)。そのため、移動動作の実行がルックアヘッド内で遅延し、動作がN10の終端で停止する可能性があります。#FLUSH CONTINUEにより、これを回避できます。N10の終端に次の移動ブロック(N200)が存在する場合、動作が中断せずに実行されます。

```
N10   G01  X200
N20   #FLUSH CONTINUE
N30   P1 = [P2 * P3] + V.E...
...
N200  X250
M30
```

12.4 解読処理と補間の同期

#FLUSH WAIT	(ノンモーダル)

「#FLUSH」とは異なり、プログラム実行がコマンド「#FLUSH WAIT」によって中断されます。その後、NCチャンネルのバッファ効果も除去されます。NCチャンネルが空で、補間が同期されている場合、その他のパートプログラムの実行が自動的に開始します。

プログラミング例

補間が位置X20に到達するまで、ブロックN20内で解読処理が停止します。その後、N30の解読処理が自動的に開始します。

```
N10 G01 X20
N20 #FLUSH WAIT
N30 X40
...
M30
```

12.5 内部ブロックが有効なコメント

#COMMENT BEGIN	コメントブロックの先頭
:	
:	無効な領域。すべての行が
:	評価が行われずに
:	NCカーネルが管理されます。
:	
#COMMENT END	コメントブロックの終端

両方のコマンドが自身のNC行内でプログラムされる必要があります。つまり、同一ブロック内の他のNCコマンドは許可されません。「(」と「)」に囲まれた1行ごとのコメントは例外です。

コメントブロックはNCブロックの内外を問わずに定義できますが、現在のファイルに限定されます。コメントに含まれるグローバルサブプログラムからの呼び出し、および戻り値は評価されません。

コメントのネストは、アプリケーションごとに含めることができます。

プログラミング例

ネストが有効:

```
:
#COMMENT BEGIN  Beginning ofcomment block 1
...
   #COMMENT BEGIN      Beginning of comment block 2
   ...
   #COMMENT END End of comment block 2
   ...
#COMMENT END    End of comment block 1
:
```

ネストが無効:

```
:
#COMMENT BEGIN  Beginning ofcomment block 1
...
   #COMMENT BEGIN      Beginning ofcomment block 2
   ...
#COMMENT END    End of comment block 1 and 2
:
```

以下のケースでは、NCカーネルがエラーメッセージを生成します。

- 「End of File」がコメントブロック内で読み取られる。これにより、次のメッセージが出力されます。「read in File end before the end of comments.」
- コメントの開始がプログラムされていないにも関わらず、*#COMMENT END*が読み取られる。メッセージ:「Comment end programmed without corresponding comment begin」
- コメントコマンド後に、ブロック内に他のNCコマンドがプログラムされている。メッセージ:「After #COMMENT BEGIN/END no further NC commands allowed」

12.6 イベントの待機

#WAIT FOR *<expr>*	(ノンモーダル)

このコマンドにより、数式がTRUEまたは0.5以上になるまで、NCプログラムの解読処理が停止します。

プログラミング例

```
N10 #WAIT FOR V.E.EXT1 == 5  (Decoding of NC program is stopped at this)
                             (point, till the value of external variable)
                             (is 5)
....
N100 #WAIT FOR V.E.EXT2      (Decoding of NC program is stopped at this)
                             (point, till the value of external variable)
                             (is ≥ 0.5.)
N200 G01 X200 F1000
....
```

注記

コントロールのルックアヘッドメカニズムによって、#WAIT FORの前にプログラムされている移動関連の一定のNCブロックをNCチャンネル内に保持できます。これらの保持される動作ブロック自体がイベント生成に関連している場合にはデッドロックが発生し、明確な原因がない状態でCプログラムの実行が停止します。

#WAIT FORの前に直接#FLUSHや#FLUSH CONTINUEをプログラムするときには、前のすべての動作ブロックを強制的に実行してデッドロックを回避できます。

プログラミング例

```
....
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          ;Forced execution of all buffered
                                        ;motion blocks
Nxx #WAIT FOR V.E.XX>123    ;Decoding of NC program is stopped at this
                             ;point, till the value of PLC variable
                             ;is > 123
....
```

12.7 正接送り速度適用のための最小半径

バージョンV2.11.2010.02からは、コマンド#TANGFEED [...]をコマンド#SET TANGFEED RMIN [...]の代わりに使用します。互換性の理由から、このコマンドは当面の間使用できますが、新しいNCプログラムでは使用しないでください。

このコマンドは、G10/G11でプログラムされた正接送り速度の適用を補完します。送り速度を適用するために、識別子を含む円弧ブロックに最小半径が考慮されます。送り速度は、プログラムされた円弧形状半径が最小半径以上の場合のみに適用されます。

#TANGFEED [RMIN<*expr*>]	(モーダル)

RMIN<*expr*> 正接送り速度での最小形状半径Rminは適用されません。[mm、インチ]

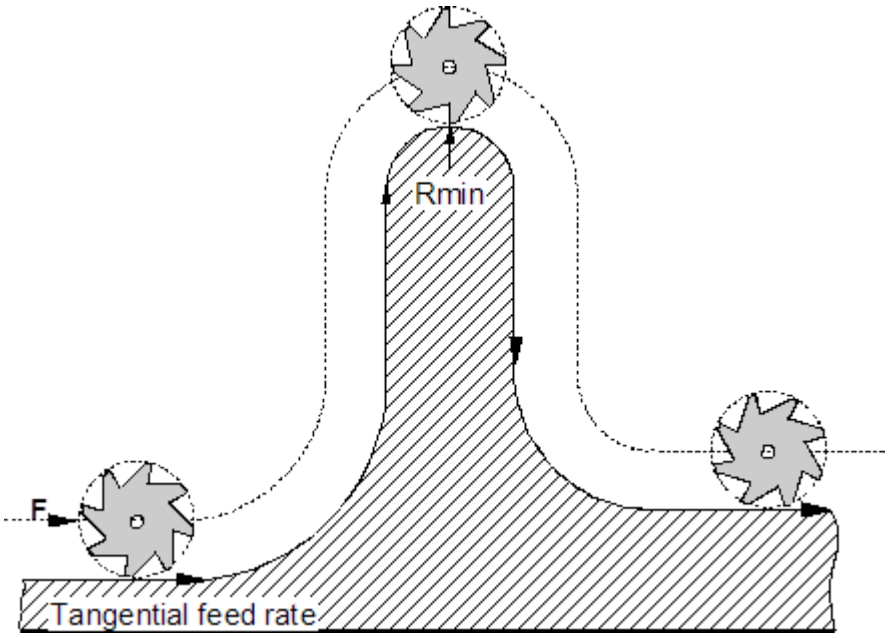


図 66: 図11-1: 正接送り速度のプログラミング

- 新しくプログラムされた最小半径は、次の関連する動作ブロックで考慮されます。
- 最小半径の宣言はオプションです。#TANGFEED [RMIN…]がプログラムされていない場合や、半径がゼロに設定されている場合も、正接送り速度が適用されます。

プログラミング例

```
(Contour with TRC and tangential feedrate adaptation)

N10  V.G.WZR=5
N20  #TANGFEED [RMIN=3]          (Set minimum radius rmin to 3mm)
N30  G41 G01 X0 Y20 G11 F600     (Selection of TRC and feedrate adaptation)
N40  X20 Y20
N50  G03 X40 R10                 (Feedrate adaptation)
```

```
N60  G01 Y40
N70  G02 X44 R2                (No feedrate adaptation → rprg < rmin)
N80  G01 Y20
N20  #TANGFEED [RMIN=5]        (Set new minimum radius rmin to 5mm)
N90  G03 X57 R4                (No feedrate adaptation → rprg < rmin)
N100 G01 Y50
N110 X60 G40 G10           (Deselection of TRC and feedrate adaptation)
N120 X0 Y0
N999 M30
```

12.8 条件解釈

プログラミング言語Cのプリコンパイラステートメントと同様に、NCプログラムのパートは#IF-#ELSE-#ENDIF構文で表現できます。このため、これらのプログラムパートの解釈は数式の関数として制御できます。

条件解釈は、必ず以下で始まり、

#IF *<expr>*

以下で終了します。

#ENDIF

ステートメント

#ELSE

はオプションで、分岐の構築も行います。

注記

$IF制御ブロック内の条件は、数式の「真」または「偽」(TRUEおよびFALSE)を確認することでチェックします。DECIMAL型の変数も使用できるようにするために、

..数式の絶対値が0.5以上の場合にジャンプ条件が満たされます。

条件解釈の使用では、以下の規則が適用されます。

- #IF、#ELSE、および#ENDIFの前では、「/」のみプログラムできます。ブロック番号は使用できません。
- 条件解釈のネストが可能です。ネストの深さは固定です。
- 条件解釈の無効な分岐は構文チェックされませんが、完全にスキップされます。

プログラミング例

```
#IF  V.E.START_VALUE == TRUE
N100 F100 …
:
#ELSE
N200 F40 …
:
#ENDIF

N300 …
:
:
N999 M30
```

12.9 軸オフセットの停止

#SUPPRESS OFFSETS <*axis_name*><*expr*> {<*axis_name*><*expr*>} (ノンモーダル)

<axis_name><expr> オフセットのない新しい軸位置

このコマンドを使用して同一のNCブロック内でプログラムされた位置が、変更されることなくNCチャンネルに渡されます。つまり、有効な「従来の」座標オフセット(ゼロオフセット、追加オフセット、位置プリセット、工具オフセットなど)は考慮されません。

コマンド#SUPPRESS OFFSETSは、現在のNCブロック内でのみ有効です。

有効なキネマティックトランスフォーメーションや直交フォーメーション(CS、ACS)によるオフセットは、このコマンドでは**停止しません**。すべての種類のオフセットを停止するには、コマンド#MCS ON/OFF [▶ 527]を使用する必要があります。

プログラミング例

```
:
N10 G92 X10 Y20                    (BPV X10 Y20)
N20 X0 Y0                          (Position X=10 Y=20)
N30 #SUPPRESS OFFSETS X0 Y0        (Position X=0 Y=0)
:
```

12.10 計測の設定

12.10.1 計測モードの切り替え

コマンド#MEAS MODE[<expr>]を使用すると、NCプログラムにより計測タイプを設定できます。

#MEAS MODE [[*<expr>*]] (モーダル)

<expr> 「計測機能 [▶ 59]」に準拠した計測タイプ。

パラメータを**使用せずに**#MEAS MODEをプログラムすると、チャンネルパラメータリストP-CHAN-00057で指定された初期設定の計測タイプを選択します。

プログラミング例

```
N10 #MEAS MODE[3]        (Measurement type 3)
N20 G100 X150            (Measurement traverse)
..
N100 #MEAS MODE          (Default- measurement type)
..
M30
```

12.10.2 拡張プログラミング

#MEAS MODEコマンドの代わりに以下のコマンドを使用すると、計測タイプの定義以外の追加調整が可能です。プログラムされたパラメータ設定は、プログラム終了まで有効です。ただし、プログラム開始後は設定リストに基づいた初期設定が有効になります。軸の計測パラメータの変更は、軸が計測軸としてマークされる(軸パラメータP-AXIS-00118が1に設定される)ことを意味します。

#MEAS [[[**TYPE***<expr>*] | [**SIMU_OFFSET***<expr>*] | [**TRIGGER**]] | [{**AX***<axis_name>* / **AXNR***<expr>*} [**SIGNAL***<ident>*] [**EDGE***<ident>*] [**INPUT***<expr>*] [**G107** | **G108**]] **]** (モーダル)

TYPE*<expr>*

第4.1.10章に準拠した新しい計測タイプ。この計測タイプは、次の新しい設定またはプログラム終了まで有効です。

SIMU_OFFSET*<expr>*

このキーワードは、軸パラメータP-AXIS-00112 *= 4*の場合の計測シミュレーション中の特定のケースでのみ有効です。この値は、プログラムした目標点に基づいたパス動作に関するシミュレーションされた初期設定の計測ポイントを移動します。

計測タイプ2に対してSIMU_OFFSETを使用すると、初期設定の計測ポイントを正または負の方向に移動できます。この場合、軸パラメータP-AXIS-00114の追加オフセットは考慮されません。

TRIGGER

プログラムされた計測信号をトリガします。必ずファンクションブロックのグローバルエッジバンディングG107/G108 [▶ 70]と組み合わせて使用します。P-CHAN-00257が有効な場合にのみ有効です。

TYPE、SIMU_OFFSET、TRIGGERは排他的であるため、他のキーワードと組み合わせてプログラムすることはできません。

TYPE、SIMU_OFFSET、TRIGGERは排他的であるため、他のキーワードと組み合わせてプログラムすることはできません。

AX<*axis_name*>

計測軸の名前です。

AXNR<*expr*>

計測軸の論理軸番号(正の整数)です。

SIGNAL<*ident*>

計測に使用される計測信号のソースの名前です(s.P-AXIS-00516)。

有効値	優先度
PLC	PLCによる計測信号です。
DRIVE	ドライブ位置ラッチによる計測信号です。
FIXED_STOP	軸停止の移動による計測信号です。
PLC_FIRST_EVENT	PLCによる計測信号です。1つの軸が計測イベントを受信すると計測が終了します。
PLC_EXT_LATCH_CONTROL	外部ハードウェアの計測インターフェイスで計測します(s. [HLI//外部プローブハードウェアでの計測])。

EDGE<*ident*>

関連する計測エッジです。P-AXIS-00518も参照。

有効値	優先度
POS	立ち上がり計測エッジ
NEG	立ち下がり計測エッジ

INPUT<*expr*>

計測に使用されるドライブ計測入力の名前(P-AXIS-00517を参照)。

計測信号DRIVE:

有効値	優先度
1	1番目の 計測入力
2	2番目の 計測入力

計測信号PLC_EXT_LATCH_CONTROL:

有効値	優先度
1 .. 255	アドレス指定された外部プローブハードウェアのチャンネル番号です。

G107 この軸のエッジバンディングファンクションを解除します。エッジバンディング中は、この軸に対して計測値はラッチされません。

G108 この軸に対するエッジバンディングファンクションを選択します。この軸の軸パラメータリスト内で、エッジバンディングファンクションが有効であることが前提条件です(P-AXIS-00098を参照)。

#MEAS DEFAULT [[{AX<*Achsname*> / AXNR<*expr*>}]] (モーダル)

DEFAULT #MEASコマンド(SIGNAL、EDGE、INPUT、G107/G108)によって変更された、**軸特有の**(AX、AXNR)パラメータ設定をリセットします。軸パラメータの計測設定が再度有効になります。

計測エッジ(EDGE)および計測入力(INPUT)は、位置ラッチが内蔵されたSERCOSドライブでは変更できません(SIGNAL=DRIVE)。これは、ドライブ内でもパラメータ変更が必要なためです。

プログラミング例

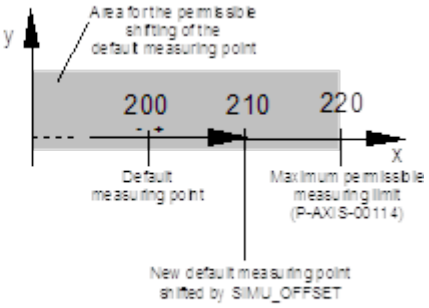
他の計測タイプの選択:

```
N100 #MEAS [TYPE=2]
```

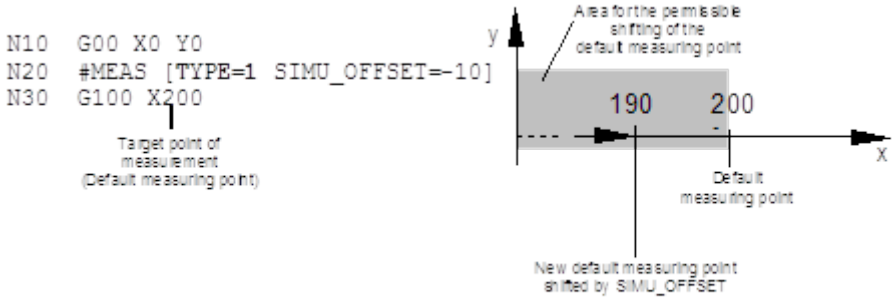
計測タイプ2での計測シミュレーション用の計測位置設定:

```
N10  G00 X0 Y0
N20  #MEAS [TYPE=2 SIMU_OFFSET=10]
N30  G100 X200
```

Target point of measurement (Default measuring point)



他のすべての計測タイプでは、負の方向(パス動作と反対方向)への移動のみ可能です。



XおよびY軸に対する軸停止の移動による計測の有効化:

```
N100 #MEAS [AX=X AX=Y SIGNAL=FIXED_STOP]
```

立ち下りエッジPLCによる計測信号の有効化:

```
N100 #MEAS [AXNR=1 SIGNAL=PLC EDGE=NEG]
```

YおよびZ軸に対するエッジバンディングファンクションの無効化:

```
N100 #MEAS [AX=Y AX=Z G107]
```

軸パラメータに基づいた、すべての補間軸の計測設定の復元:

```
N100 #MEAS DEFAULT
```

軸パラメータに基づいた、すべての補間軸の計測設定の復元:

```
N100 #MEAS DEFAULT [AX=X]
```

12.11 位置プリセット

新規位置プリセットは、以下のようにプログラムします。

#PSET <*axis_name*><*expr*> {<*axis_name*><*expr*>}	(ノンモーダル)

<axis_name><expr> 軸の新しい現在位置

位置は絶対値でプログラムします。このコマンドにリンクする移動動作は**ありません**。

新しいプログラミング位置プリセットがプログラムされていない軸は、現在のプログラミング位置プリセットを保持します。つまり、対応するオフセットはこの軸については変更されません。

#PSETコマンドは何回でも繰り返し使用できます。

プリセットファンクションはノンモーダルです。

機械基準検索(G74)により、#PSETコマンドによって発生した座標系のオフセットが除去されます。

12.11.1 位置プリセットの解除

#PRESET *{<axis_name> <dummy_expr>}* (ノンモーダル)

<axis_name><dummy_expr> 軸の現在値オフセットがリセットされます。座標値は構文上の理由でのみ必要です。

軸を指定せずに#PRESETをプログラムすると、すべての軸のプログラミング位置プリセットオフセットが除去されます。

工具径補正、ミラーリング、または直径プログラミングが使用されている場合は、#PSETまたは#PRESETをプログラムしてはなりません。

(#PSETおよび#PRESET コマンドのみが「ノンモーダル」です。プログラミング位置のプリセットのオフセット自体は、 再度プログラムされるかまたは#PRESETで選択解除されるまではもちろん有効です)

プログラム例

```
N10 X50  Y10  Z0
N20 #PSET X20 Y20     (Preset position for X and Y)
N30 X10
N40 #PRESET X10       (Reset X-Position; value 10 not relevant)
N50 X30
N60 #PRESET           (Reset all axes)
N70 M30
```

以下は、対応するNCブロック実行後の機械座標内でのX軸の位置です。

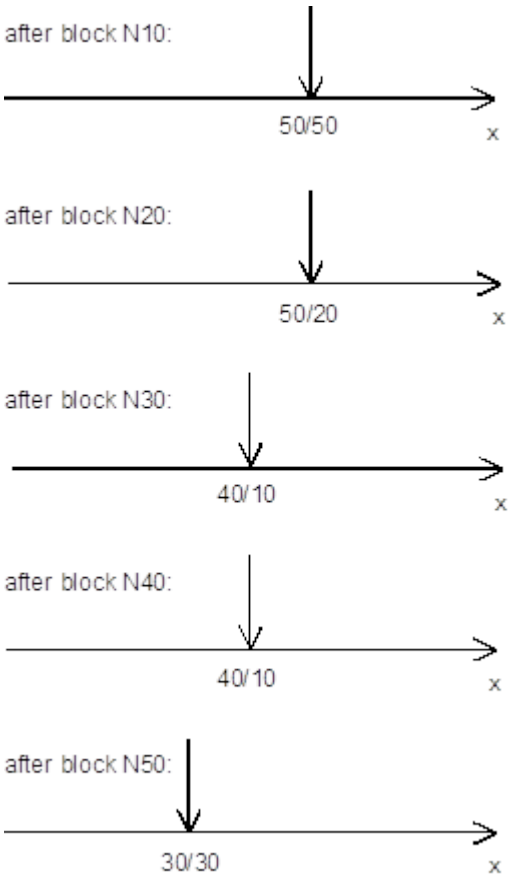


図 67: 図11-2: 機械座標/プログラム座標内でのX軸の位置(この例では、他の座標変換は選択されていません)

(この例では、他の座標変換は選択されていません)

12.12 同期操作

12.12.1 軸結合のプログラミング

以下のコマンドをNCプログラム内で使用し、軸結合を定義できます。

#SET AX LINK [*<coupling_group>*, <s*lave*>=<*master*> {,<*slave*>=<*master*>}]
または
#AX LINK [NBR] [*<coupling_group>*, <s*lave*>=<*master*> {,<*slave*>=<*master*>}]

<coupling_group> 結合グループ数 (1)

<slave> 結合ペアiの従軸の名前、または論理軸番号 (2)

<master> 結合ペアiの主軸の名前、または論理軸番号 (2)

NBR 論理スイッチNBRを使用すると、評価を論理軸名から軸番号に変更できます。変更後は、軸結合(および結合ペア)を論理軸番号で定義する必要があります。定義をする際は、軸がNCチャンネル内に存在してはなりません。NCチャンネル内での存在チェックは、結合グループを有効にする時にのみ行われます!

各結合グループ内で、1つ以上のマスタ/スレーブ結合ペアが定義されている必要があります。
スピンドルの結合は、「同期スピンドル操作」の章でより詳細に記載されています。

一般的な処理および操作方法:

- 結合グループの定義は、プログラム全体に適用されます。これは、以降のNCプログラム内でもこの結合グループを再選択できることを意味します(「軸結合の選択/解除 [▶ 252]」の章も参照)。
- 結合番号が「0」の結合グループは、NCプログラム内でプログラムできません。これはチャンネルパラメータリスト[1]-2で定義されています。
- 無効にされている結合グループは、いつでも変更できます。以前に定義された結合は上書きされます。
- 有効な結合グループは変更できません。
- 軸の再帰結合はできません。結合グループ内の主軸を同時に従軸にすること、または従軸を同時に主軸にすることはできません。従軸を同時に他の結合グループ内の主軸にすること、または主軸を同時に他の結合グループ内の従軸にすることはできません。
- 結合ペアの主軸と従軸を同一にすることはできません。
- 主軸は従軸にできません。
- 従軸は1つの主軸にしか割り当てられませんが、主軸は複数の従軸のオーナになることができます。
- 同期操作が有効な場合、すべての主軸および従軸がNCチャンネルの軸グループに含まれているかどうかがチェックされます。
- 主軸および従軸は同一の軸タイプであることが必要です。また、同一の軸モードで使用されている必要があります。
- 結合グループのプログラミングに使用するNCコマンドは、NCブロック内の単一の命令であることが必要です。
- 結合番号も数式でプログラムできます。数式の結果は正の整数となる必要があります。
- 従軸(#CAXTRACK)は主軸にも従軸にもすることができません。

(1) 1 ... [結合グループの最大数–1]、[6]-2.11を参照

(2) 結合ペアの最大数、[6]-2.12を参照

プログラミング例

```
N10 #SET AX LINK[1, Z2=Z]          Z2 is coupled as slave to main axis Z

N20 #SET AX LINK[2, Y2=Y, X2=X]    Coupling Y2 with Y and X2 with X

N30 #SET AX LINK[3, X1=X, X2=X, X3=X]  Coupling of X1, X2, X3 as slaves to the same master axis X
...

or alternative

N10 #AX LINK[1, Z2=Z1]

N20 #AX LINK NBR[2, 8 = 2, 9 = 3]     Coupling via log. axis numbers
```

12.12.2 軸結合の拡張プログラミング(SOFT-GANTRY)

このファンクションを使用できるかどうかは、設定やバージョンの具体的な範囲によって異なります。

輪郭加工軸に加えて、スピンドルも同期モードで動作させることが可能です。詳細説明は「同期スピンドル操作 [▶ 490]」に記載されています。

輪郭加工軸は、いわゆるガントリー軸としても操作できます。通常の同期操作とは異なり、位置偏差を考慮したモニタリングメカニズムが有効になり、特殊な誤差応答が適用されます。機械構造により、メカニカル(および静的)なガントリー動作中に軸が相互に恒久的に結合され、機械の設定によって定義されます。制御システム起動後のガントリー結合の動的変更はできません。

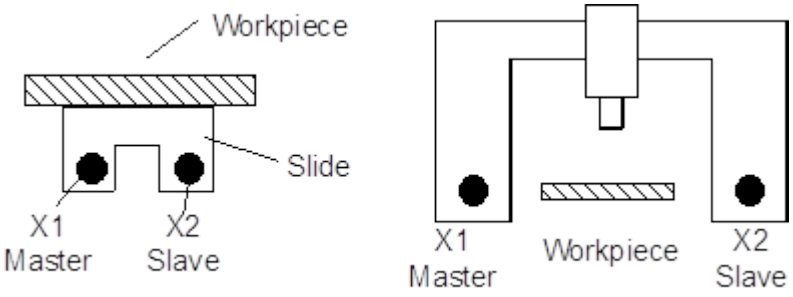


図 68: 図11-3: メカニカルなガントリー操作

2つのスライドが独立した研削機械など、基本構造によってメカニカルなガントリー操作ができない機械は、プログラミングによるガントリーモードで操作できます。例えば、大きなワークのクランプや加工を行うために、スライドを互いに結合する必要がある場合にプログラミングによるガントリーモードが必要になります。

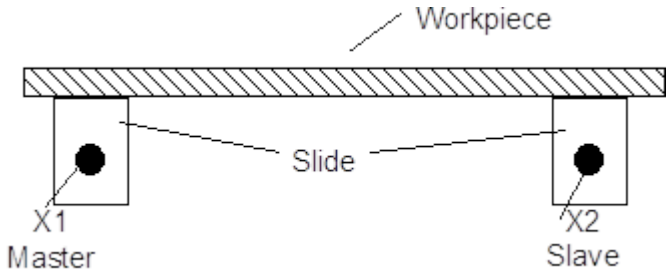


図 69: 図11-4: プログラム可能なガントリー操作(「Soft-Gantry」)

このために、「軸結合のプログラミング [▶ 246]」の章で説明されている#SET AX LINKおよび#AX LINKコマンドを拡張構文で使用できます。

#SET AX LINK [*<coupling_group>*, **[** *<slave>*=*<master>*,**G [**,*<limit_1>*, *limit_2*>**]]**
{, **[** *<slave>*=*<master>*,**G [**,*<limit_1>*, *limit_2*>**]] }]**
または
#AX LINK [NBR] [*<coupling_group>*, **[** *<slave>*=*<master>*,**G [**,*<limit_1>*, *limit_2*>**]]**
{, **[** *<slave>*=*<master>*,**G [**,*<limit_1>*, *limit_2*>**]] }]**

<coupling_group> 結合グループ数 (1)

<slave> 結合ペアiの従軸の名前、または論理軸番号 (2)

<master> 結合ペアiの主軸の名前、または論理軸番号 (2)

NBR 論理スイッチNBRを使用すると、評価を論理軸名から軸番号に変更できます。変更後は、軸結合(および結合ペア)を論理軸番号で定義する必要があります。定義をする際は、軸がNCチャンネル内に存在してはなりません。NCチャンネル内での存在チェックは、結合グループを有効にする時にのみ行われます!

G 「ガントリー結合」のキーワード

ガントリー結合を使用する場合、2つのステージで以下の値がガントリー軸の許容位置偏差のモニタリングに使用されます。[mm]で指定されます。正の実数:

<limit_1> 1番目のモニタリングリミット値。このリミット値を超過すると動作が中止され、制御システムはエラー状態であると見なします。初期設定では、位置偏差はRESETにより消去されます。用途に応じて、偏位動作も使用できます。

<limit_2> 2番目のモニタリングリミット値。このリミット値を超えると、リセットできないエラーが出力されます。制御システムの電源をオフにし、位置偏差を手動で修正する必要があります。

モニタリングリミット値がプログラムされていない場合は、従軸の軸パラメータレコードP-AXIS-00072およびP-AXIS-00071の初期設定が適用されます。

(1) 1 ... [結合グループの最大数–1]、[6]-2.11を参照

(2) 結合ペアの最大数、[6]-2.12を参照

通常処理および操作原理の補足事項

- ガントリー結合は、結合選択時(第11.12.3章を参照)の軸位置で行われます。オフセットは位置セットポイントにより位置コントローラで内部的に計算されるため、NCコマンド内でオフセットを指定する必要はありません。
- 従軸の動的データは、輪郭加工動作に考慮されます。
- 安全上の理由から、リセット時またはプログラム終了時に有効な状態の結合は、次回のプログラム開始時に暗黙的に復元されます。この挙動はパラメータ(P-CHAN-00104、P-CHAN-00105)で定義できます。

プログラミング例

```
:
N10 #SET AX LINK[1, [Y2 = Y1,G,0.01,0.25]]  Gantry coupling of Y1 as
                                            master axis and Y2 as slave
                                            axis. First limit is 10µm,
                                            second limit is 250 µm.
N20 #SET AX LINK[2, [Y2 = Y1,G]]   Gantry coupling of Y1 (master) and
                                   Y2 (slave). The monitoring limits
                                   of the axis parameter record of the
                                   Y2-axis are valid.
N30 #SET AX LINK [3,[Y2 = Y1]]     Standard coupling of Y2 and Y1.
                                   No gantry operation.

or alternative

N10 #AX LINK[1, [Y2 = Y1,G,0.01,0.25]]

N20 #AX LINK NBR[2, [8 = 2,G]]     Gantry coupling via log. axis numbers
```

形状が対称、またはサイズが異なるワークの並行加工を#SET AX LINKおよび#AX LINKコマンドの拡張構文によってプログラムできます。これらのモード(ミラーリングやスケーリング)では、位置偏差はモニタリングされません。

#SET AX LINK [*<coupling_group>* **, [** *<slave>*=*<master>*,*<numerator>*, *<denominator>*]
{, [*<slave>*=*<master>*,*<numerator>*,*<denominator>*] }]

または

#AX LINK [NBR] [*<coupling_group>*, [*<slave>*=*<master>*,*<numerator>*, *<denominator>*]
{, [*<slave>*=*<master>*,*<numerator>*,*<denominator>*] }]

<coupling_group> 結合グループ数 (1)

<slave> 結合ペアiの従軸の名前、または論理軸番号 (2)

<master> 結合ペアiの主軸の名前、または論理軸番号 (2)

NBR 論理スイッチNBRを使用すると、評価を論理軸名から軸番号に変更できます。変更後は、軸結合(および結合ペア)を論理軸番号で定義する必要があります。定義をする際は、軸がNCチャンネル内に存在してはなりません。NCチャンネル内での存在チェックは、結合グループを有効にする時にのみ行われます!

<numerator>、<denominator> 整数。これらは主軸と従軸間の結合係数の計算に使用されます。

計算によって導き出される結合係数に以下が適用されます。

-1 : ミラー結合

1 : 標準結合、従来の構文と同等。

0 : エラーメッセージの出力

注記

純粋なスケーリング(係数 ≠ 1)、またはミラーリングと同時に行われるスケーリング(係数 ≠ -1)となる係数は、現在は使用できません。警告が出力され、結合は標準結合として処理されます。つまり、**結合係数としては–1と1**しか使用できません(例を参照)。

(1) 1 ... [結合グループの最大数–1]、[6]-2.11を参照

(2) 結合ペアの最大数、[6]-2.12を参照

プログラミング例

```
:
N10 #SET AX LINK[1, [Y2 = Y1,1,-1]]   Mirroring coupling (Factor -1)
N20 #SET AX LINK[1, [Y2 = Y1,-1,1]]   Mirroring coupling (Factor -1)
N30 #SET AX LINK[1, [Y2 = Y1,-2,2]]   Mirroring coupling (Factor -1)
N40 #SET AX LINK[1, [Y2 = Y1,1,1]]    Standard coupling
N50 #SET AX LINK[1, [Y2 = Y1,2,2]]    Standard coupling
N60 #SET AX LINK[1, [Y2 = Y1,0,1]]    Error, Program abortion
N70 #SET AX LINK[1, [Y2 = Y1,1,0]]    Error, Program abortion
N80 #SET AX LINK[1, [Y2 = Y1,1,2]]    Warning (Factor 0.5), Standard cpl.
```

```
N90 #SET AX LINK[1, [Y2 = Y1,2,3]]    Warning(Factor 0.666), Standard cpl.
N100 #SET AX LINK[1, [Y2 = Y1,3,2]]   Warning (Factor 1.5), Standard cpl.
N110 #SET AX LINK[1, [Y2 = Y1,-1,2]]  Warning (Factor -0.5), Standard cpl.
N120 #SET AX LINK[1, [Y2 = Y1,-3,2]]  Warning(Factor -1.5), Standard cpl.
or alternative
N40 #AX LINK[1, [Y2 = Y1,1,1]]    Standard coupling
N50 #AX LINK NBR[1, [8 = 2,2,2]]  Standard coupling via log. axis numbers
```

12.12.3 軸結合の選択/解除

以下のNCコマンドで結合グループを有効にできます。

#ENABLE AX LINK [*<coupling_group>* **]**	
または	
#ENABLE AX LINK	*(結合グループ0、チャンネルパラメータリストで定義)*
または	
#AX LINK ON [*<coupling_group>***]**	
または	
#AX LINK ON	*(結合グループ0、チャンネルパラメータリストで定義)*

以下のNCコマンドで有効な結合グループを無効にできます。

#DISABLE AX LINK [*<coupling_group >* **]**	
または	
#DISABLE AX LINK	*(最後に有効にされた結合グループの解除)*
または	
#AX LINK OFF [*<coupling_group >* **]**	
または	
#AX LINK OFF	*(最後に有効にされた結合グループの解除)*
#AX LINK OFF ALL	*(最後に有効にされた結合グループの解除)*

処理および操作方法:

- NCカーネルを初期位置でスタートアップすると、結合グループは有効になりません。軸結合は、NCプログラム内で有効にされると有効になります。有効な場合は、初期設定ではプログラム終端(M30、M02)で終了します。有効な軸結合が次のプログラムでも有効である(プログラムグローバルである)必要がある場合は、特定のチャンネルパラメータP-CHAN-00105を設定する必要があります。
- 複数の結合グループを同時に有効にできます。

- 割り当てられていない結合グループは有効にできません。1つ以上の有効なマスタ/スレーブ結合ペアが定義されている場合、結合グループが割り当てられていると見なされます。
- NCコマンドはNCブロック内の単一の命令であることが必要です。
- 結合グループの番号も数式でプログラムできます。
- 同期モードの選択または解除の前に、工具径補正(TRC)が解除されている必要があります。
- 同期モードが選択されている場合、並行補間による手動モード(G201)を従軸に対して有効にできません。
- 同期モードが有効な場合、従軸の位置はプログラムできません。

プログラミング例

使用される軸名: 主軸システムX、Y、Z、C

従軸システムY_S、Z_S、C_S

```
(Initialization program)
%L UP_INIT_ACHS_KOPPL
(initialize axis coupling 1)
N10 #SET AX LINK[1, Y_S=Y, Z_S=Z, C_S=C]
(or #AX LINK[1, Y_S=Y, Z_S=Z, C_S=C]
N20 M17

(tool changing program)
%L UP_WZ
N30 #DISABLE AX LINK (or #AX LINK OFF)
(Approach tool changing position)
N40 G01 G90 Y1000 Z100 C0 Y_S=1000 Z_S=100 C_S=0
(Tool change; T10 contains all tool axis offsets and the tool lengths of master and slave tools; or
these values are explicitly included in the  calculation.)
N50 T10 D10
.....................
(Further commands for physical tool change)
.....................
(Approach old coupling position. The coupling position may also be defined
via parameter programming, and then be used by the subroutine.)
N80 G01 G90 X20 Y20 Z40 C50 Y_S=20 Z_S=40 C_S=50
N90 #ENABLE AX LINK[1] (or #AX LINK ON[1])
N110 M17


(Subroutine for contour machining)
%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


(Main program; initial condition: Both tools have been changed in.)
(Move both axis systems to coupling position first.)
N300 G01 G91 X20 Y20 Z40 C50 Y_S=20 Z_S=40 C_S=50 F300
(Start synchronous operation)
N310 #ENABLE AX LINK[1] (or #AX LINK ON[1])
N320 LL UP1
.....................................
.....................................
N400 #DISABLE AX LINK (or #AX LINK OFF)
N410 M30
```

12.12.4 変数経由での結合状態および結合番号の取得

NCプログラム内の変数経由で、現在の結合状態、および現在の、または最後の有効な結合グループにアクセスできます。このために、軸グループに関連する以下の変数が用意されています。

現在の、または最後の有効な結合グループの取得:

V.G.AX_LINK.NR

現在の結合状態の取得:

V.G.AX_LINK.ACTIVE

これらの変数には、読み取りアクセスのみ可能です。書き込み操作をすると エラーメッセージが出力され、解読処理が中止します。

12.13 NCプログラムからのメッセージ

この機能を使用することで、NCプログラムからのメッセージを任意のシステムデバイスに出力できます。この機能は、オペレーティングシステムの「printf」ファンクションに対応します。

NCプログラム内のメッセージをプログラムし、例えばオペレータにプロセスやコントローラの現在の状態を知らせることが可能です。これにより、メッセージ表示はオプションで現在の処理状態と同期することができます。これは、インターポレータがNCプログラム内の特定の箇所に到達した場合にのみメッセージが表示されることを意味します。

メッセージテキストはASCII文字、パラメータ、および変数で構成できます。

メッセージは同一NCブロック内の他のNCコマンドと共にプログラムすることはできません(例外: #ADD)。

メッセージはASCII形式で送信されます。

12.13.1 メッセージのプログラミング

#MSG [<*mode*>] [<*receiver*>] [**"**<*message_text*>**"**]

<mode> モードによりメッセージ出力の時間が定義されます。インターポレータの処理状態と同期してメッセージが出力される場合には、SYNまたはSYN_ACKモードが指定されている必要があります。モードが指定されていない場合、またはACKが指定されている場合には、解読処理直後にメッセージが出力されます。

モード	メッセージ出力の時間
SYN	インターポレータ内の処理状態と同期。
SYN_ACK	受信機(PLCなど)による応答確認で、インターポレータ内の処理状態と同期。応答確認が終了すると、インターポレータ内の処理が継続します。
---	解読処理直後(初期設定)
ACK	受信機(PLCなど)による応答確認で、解読処理直後。応答確認が終了すると、解読処理が継続します。

<receiver> メッセージのレシーバを通信デバイスIDで指定します。レシーバの指定はオプションです。レシーバがプログラムされていない場合、メッセージは初期設定されているレシーバ[6] *に送信されます。

レシーバID	優先度
ISG_DIAG_BED	メッセージはCNC故障診断インターフェイスに送信されます。
HMI	メッセージはシステム固有のユーザインターフェイスに送信されます。
PLC	メッセージはシステム固有のPLCに送信されます。

* TwinCAT環境では、故障診断インターフェイスahmi_ads.exeが初期設定のレシーバです。

注記

メッセージが対応するレシーバによって読み取られない場合、メッセージを10回送信した後にCNCが停止します。

<message_text> メッセージテキストは二重引用符「"..."」で囲む必要があります。

プログラミング例

```
#MSG HMI ["Text_1"]

#MSG SYN HMI ["Text_2"]

#MSG ["Text_3"]

#MSG SYN ["Text_4"]

#MSG SYN_ACK ["Text_5"]
```

解読直後、Text_1がシステム固有のユーザインターフェイスに送信されます。Text_2の文字列は、インターポレータ内の処理と同期して同一のデバイスに送信されます。

解読直後、Text_3が初期設定されたレシーバに送信されます。Text_4の文字列は、インターポレータ内の処理と同期して同一のデバイスに送信されます。

Text_5の文字列もインターポレータ内の処理と同期して同一のデバイスに送信されますが、確認応答が受信されるまで処理が停止します。

数値の出力、および文字列の出力では、以下のフォーマット要素を使用できます。

%dまたは%D	10進数(符号付き整数)の出力
%uまたは%U	10進数(符号なし整数)の出力
%fまたは%F	実数(浮動小数点数)の出力
%sまたは%S	文字列(<string>、マクロ、V.E.STRING)の出力

% xxを使用して、最大10のパラメータを出力できます。フォーマット文字の数が、後に続くパラメータの数と一致する必要があります。

プログラミング例

フォーマット要素を使用したメッセージのプログラミング:

```
#MSG [ "Message text_%d and message text_2", 1 ]
Sended string: Message text_1 and message text_2

#MSG [ "Current measurement value: %f", V.A.MEAS.ACS.VALUE.X]
Sended string: Current measurement value: 3.4567800000E+001

#MSG [ "Current state: %s", "End of roughing"]
Sended string:  Current state: End of roughing
```

文字「%」を出力するには、制御シーケンス「%%」をプログラムする必要があります。また、二重引用符の前には文字「\」が必要です。

「%」および二重引用符を使用したメッセージのプログラミング:

```
#MSG [ "Text with %%-character"]
#MSG [ "Text in quotation marks: \"TEXT\" "]
```

この例では、以下の文字列を出力します。

```
Sended string: Text with %-character and Text in quotation marks: "TEXT"
```

メッセージテキストはパラメータ、および変数で指定します。

パラメータおよび変数を使用したメッセージのプログラミング:

```
P10      = 1
V.P.BSP  = 2
#MSG SYN ["Text_%D and Text_%D", P10, V.P.BSP]
```

この例では、インターポレータ内の処理と同期して、Text_1およびText_2文字列を初期設定されたレシーバに送信します。

12.13.2 メッセージ情報のプログラミング

以下のコマンドを使用し、ユーザに関する情報のプログラミングや、メッセージ処理のパラメータ表示が可能です。この情報、およびパラメータも、ASCII文字で送信されます。

#MSG INFO [<*mode*>] [<*receiver*>] **["**<*message_information*>**"]**

メッセージの場合、通常はメッセージのレシーバによるASCII文字列の表示のみであるのに対して、メッセージ情報のASCII文字列には、レシーバ自身によってエンコードされるユーザ関連の情報が含まれています。

インターポレータの処理状態と同期してメッセージがレシーバに送信される場合には、**SYN**が指定されている必要があります。モードが指定されていない場合は、解読処理直後にメッセージが出力されます(「メッセージのプログラミング [▶ 254]」の章を参照)。

レシーバの指定には、#MSGコマンドと同一の要件が適用されます。

メッセージ情報は、二重引用符「"..."」で囲む必要があります。数値の出力には、#MSGコマンドと同一の書式要素を使用できます(「メッセージのプログラミング [▶ 254]」の章を参照)。

プログラミング例

```
P1 = 20
#MSG INFO ["RED,%d,BOLD ,P1]
```

この例では、文字列「RED,20,BOLD」がメッセージ情報として初期設定されているレシーバに送信されます。

12.13.3 「記号文字列」機能の組み入れ

このファンクションを使用できるかどうかは、設定やバージョンの具体的な範囲によって異なります。

複数のプログラム間でのメッセージ管理を行うために、メッセージを記号文字列として定義することも可能です。これは、繰り返し使用されるメッセージや複数のプログラムで使用されるメッセージで特に便利です。

例: チャンネルパラメータリスト[1]内での記号文字列の定義:

```
makro_def[1].symbol             Message_1
makro_def[1].nc_code            #MSG ["Message text "]
```

これにより、以下のメッセージをNCプログラムに出力できます。

```
"Message_1"           (The "Message text" string is transmitted to the)
                      (default receiver of messages)
```

12.13.4 ファイルへのメッセージの書き込み

コマンド#MSG SAVEを使用し、データをNCプログラム以外のストレージメディア(ハードディスクなど)のファイルに直接保存します。メッセージテキストの構成方法とデータ出力の方法は、コマンド#MSGや#MSG INFOと同一です。

#MSG SAVE [EXCLUSIVE] ["*<message_text>*"]

モード(SYN)やレシーバID (HMIなど)のプログラミングは不要です。メッセージはNCプログラムでの評価直後にレポートファイルに書き込まれるため、プログラミングするとエラーが発生します。

メッセージテキストは二重引用符「"..."」で囲む必要があります。ファイルが既に存在する場合には、#MSG SAVEを呼び出す度にメッセージテキストがファイルの末尾に追加されます。新規にレポートファイルを作成するには、NCプログラムの開始前にユーザが既存のファイルを削除する必要があります。

#MSG SAVE ["message text "]を使用すると、メッセージテキストがレポートファイルに以下の形式で保存されます。

< time stamp > “message text“

#MSG SAVE EXCLUSIVE ["message text "]を使用すると、メッセージテキストがレポートファイルに以下のタイムスタンプが**付加されない**形式で保存されます。

“message text“

レポートファイル名は、コマンド#FILE NAMEを使用して事前に定義できます。このコマンドを使用しない場合、#MSG SAVEは初期設定の名前**message.txt**のファイルに書き込みます。

レポートファイルのパスは、スタートアップリストP-STUP-00020内で定義できます。レポートファイルのパスが入力されていない場合には、制御プラットフォームに応じて初期設定のパスが使用され、レポートファイルがコントロールのメインディレクトリに保存されます。

プログラミング例

```
:
#FILE NAME [MSG="timer.txt"]
:
#TIMER START [ID=10]                Start counter 10
#TIMER START SYN [ID11]             Start counter 11 (IPO-level)
:
:
#TIMER READ [ID10]                  Save counter value in V.G.TIMER[10]
#TIMER READ SYN [ID11]              Save counter value in V.G.TIMER[11]
#MSG SAVE["T10= %d",V.G.TIMER[10]]   Store counter value in timer.txt
#MSG SAVE["T11= %d",V.G.TIMER[11]]   Store counter value in timer.txt
#TIMER STOP [ID10]                  Counter 10 is stopped.
#TIMER CLEAR [ID10]                 Reset counter 10.
:
```

12.13.5 ブロック終端での追加情報の出力

#ADDコマンドを使用し、NCブロックに追加情報をプログラムできます。追加情報の構成方法は、#MSGコマンドと同一です。ただし、#MSGコマンドとは異なり、#ADDコマンドの**前に**他のNCコマンドをプログラムできます。このため、#ADDは必ずNCブロック終端の最後のコマンドとしてプログラムする必要があります。以下のコメントが可能です。

#ADD [<*receiver*>] [**"***<additional_information>***"**]

モード(SYN)のプログラミングは不要です。このメッセージは必ずインターポレータ内の処理状態と同期して出力されるため、プログラミングするとエラーが発生します。

レシーバの指定には、#MSGコマンドと同一の要件が適用されます。

追加情報は二重引用符「"..."」で囲む必要があります。

プログラミング例

```
%add_block_info
N05 P1=20
```

```
N10 G00 X0 Y0 Z0
N15 T1             #ADD["Tool T=%d active", V.G.T_AKT]
N20 G01 F2000 X10 #ADD["Move to X-position"] (Comment)
N30 YP1            #ADD["Y-position=%d", P1]
N40 Z30            #ADD["Z-position"]
N50 Z33            #ADD["Z-position"] X11 Y22  ->Error 21509
N999 M30
```

12.14 加加速度制限傾斜

この傾斜ファンクションは、指定された許容速度、加速度、および加加速度[2]-1を維持しながら、プログラムされた経路上の速度を決定します。以下のモードを使用できます。

- 加加速度がモニタリングされずに、加速度が制限される階段状の加速度プロファイル
- 加加速度がモニタリングされる台形の加速度プロファイル
- 加加速度がモニタリングされる方形波/正弦波の加速度プロファイル

選択された傾斜ファンクションに応じて加速度曲線が生成されます。

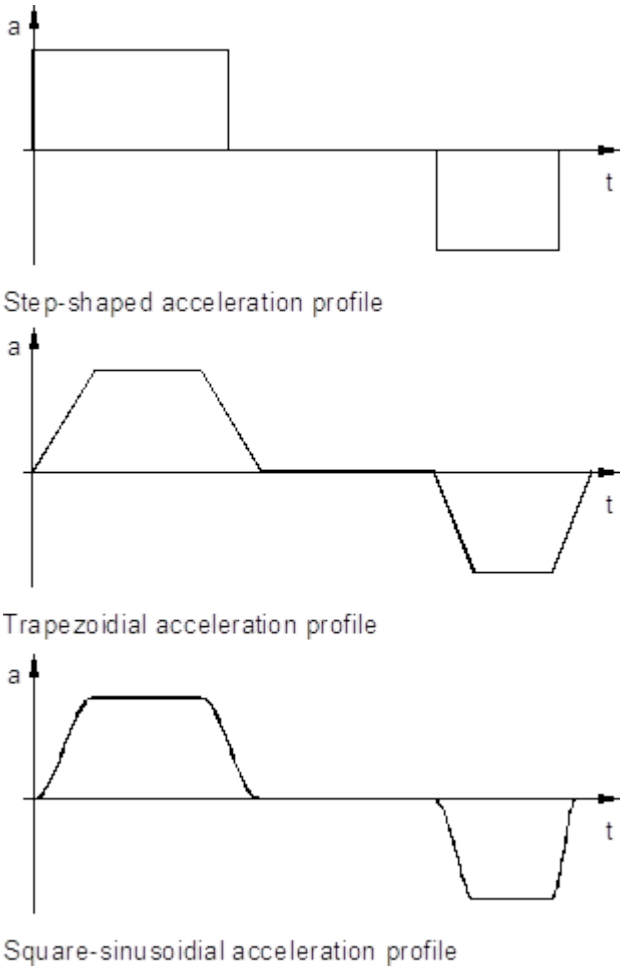


図 70: 図11-5: プログラムされた経路上の加速度

加速度プロファイルは、加速度およびランプ時間によって軸ごとに指定されます[2]-1。

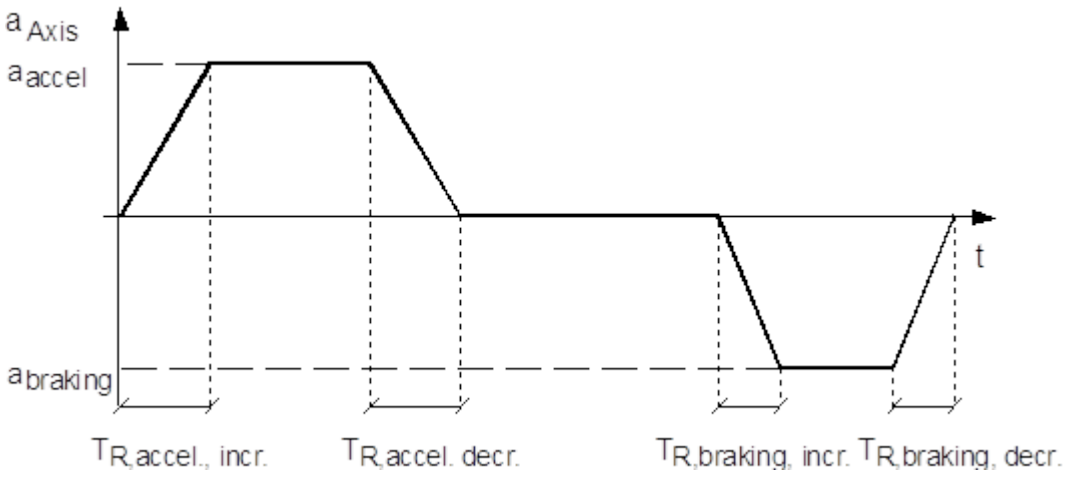


図 71: 図11-6: 加速度プロファイルのパラメータ

12.14.1 操作モードの選択

バージョンV2.11.2010.02以降は、コマンド#SLOPE [...]をコマンド#SET SLOPE PROFIL [...]の代わりに使用します。互換性の理由から、このコマンドは当分の間使用できますが、新しいNCプログラムでは使用しないでください。

#SLOPE [TYPE<*ident*>]	(モーダル)

TYPE<*ident*> 加速度プロファイルのタイプ。使用可能なプロファイルのタイプ:

STEP	階段状の加速度プロファイル(初期設定)
TRAPEZ	台形の加速度プロファイル
SIN2	方形波/正弦波の加速度プロファイル
HSC	「拡張HSCプログラミング [▶ 274]」で推奨されるHSC傾斜 *

* プロファイルタイプ3 (HSC傾斜)を選択してこの機能を使用するには、拡張パッケージ「HSC」のライセンスが必要です。この機能は、標準ライセンスの範囲には含まれていません。

ランプ時間(G132/G133)および加速度(G130/G131)の特定の重み付けは、コマンド#SLOPE [...]ではサポートされていません。重み付けはすべてのランプ時間と加速度に対して常に有効です(初期設定)。

#SLOPE DEFAULT	(モーダル)

#SLOPE DEFAULTをプログラミングすると、基本状態(起動後など)を復元します。これは、初期設定でチャンネルパラメータP-CHAN-00071の傾斜タイプが使用されることを意味します。

プログラミング例1

```
N10  G01  X50  Y10  Z0  F1000          (step-shaped acceleration profile)
N20  #SLOPE[TYPE=TRAPEZ]               (trapezoidal acceleration profile)
N30  X10  Y30
N40  #SLOPE[TYPE=SIN2]                 (sinusoidal acceleration profile)
N50  X15
N60  Y50
N70  M30
```

プログラムされた経路上では、以下の加速度進行が取得されます。

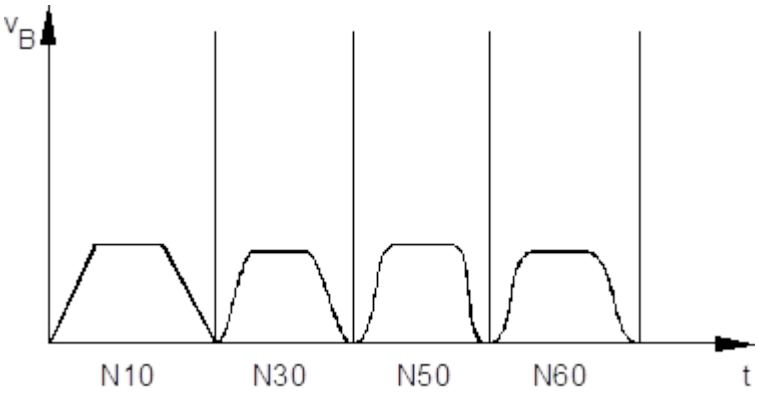


図 72: 図11-7: プログラム経路によって変化する加速度

12.15 Akimaスプライン軸補間

この機能を使用するには、拡張パッケージ「Spline」のライセンスが必要です。このライセンスは、標準ライセンスの範囲には含まれていません。

スプライン軸補間を選択すると、プログラムされた頂点を通るAkimaスプラインが生成されます。

12.15.1 スプラインタイプの選択

バージョンV2.11.2010.02以降では、コマンド#SPLINE TYPE AKIMAがコマンド#SET SPLINETYPE AKIMAに取って代わります。互換性の理由から、このコマンドは当面の間使用できますが、新しいNCプログラムでは使用しないでください。

#SPLINE TYPE AKIMA	(モーダル)

スプラインを使用する場合は、このスプラインタイプが初期設定です。ただし、スプラインタイプは明示的に指定することを推奨します。

スプラインタイプの選択直後から、前の動作ブロックからスプライン曲線への正接遷移ができなくなります。

12.15.2 スプライン軸補間の選択

バージョンV2.11.2010.02以降では、コマンド#SPLINE ONをコマンド#SET SPLINE ONの代わりに使用します。互換性の理由から、このコマンドは当面の間使用できますが、新しいNCプログラムでは使用しないでください。

#SPLINE ON	(モーダル)

互換性の理由から、G151コマンドで選択することも可能です。

最後に選択されたモードでスプライン軸補間が選択されます。スプライン曲線は、最後にプログラムされた目標位置で開始します。この位置は、スプライン曲線の最初の頂点です。

このコマンドは2番目の頂点と同一のステートメント、または以前のステートメントにプログラムできます。

12.15.3 スプライン軸補間の解除

バージョンV2.11.2010.02以降では、コマンド#SPLINE OFFをコマンド#SET SPLINE OFFの代わりに使用します。互換性の理由から、このコマンドは当面の間使用できますが、新しいNCプログラムでは使用しないでください。

#SPLINE OFF	(モーダル)

互換性の理由から、G150コマンドを使用して解除することも可能です。

1つのステートメント内でコマンドと位置がプログラムされている場合、この位置はスプライン曲線の一部では**ありません**。

3つ以上の頂点がプログラムされている場合に限り、解除が可能です。すべての接線(曲線の両端での正接遷移、または**すべての**接線の明示的なプログラミング)が指定されている場合は、2つのプログラムされた頂点の後に解除できます。

12.15.4 頂点のプログラミング

頂点のプログラミングは、目標点が頂点として使用される直線ブロック(G00、G01)で行います。

12.15.5 遷移タイプの指定

バージョンV2.11.2010.02以降は、コマンド#AKIMA TRANS [...]をコマンド#SET ASPLINE MODE [...]の代わりに使用します。互換性の理由から、このコマンドは当面の間使用できますが、新しいNCプログラムでは使用しないでください。

#AKIMA TRANS [[START<*ident*> END<*ident*>]]	(モーダル)

START<*ident*>

END<*ident*>

スプライン曲線と隣接する(直線または円弧)動作ブロック間の正接遷移のタイプ。遷移タイプの指定はオプションです。**指定されていない**場合は、遷移タイプAUTOが初期設定として使用されます。使用可能な遷移タイプ

AUTO　　遷移時のスプライン曲線の接線が自動的に計算されます。

TANGENTIAL　前後の直線または円弧ブロックへの正接遷移。

USER　　遷移時のスプライン曲線の接線を明示的に指定します。

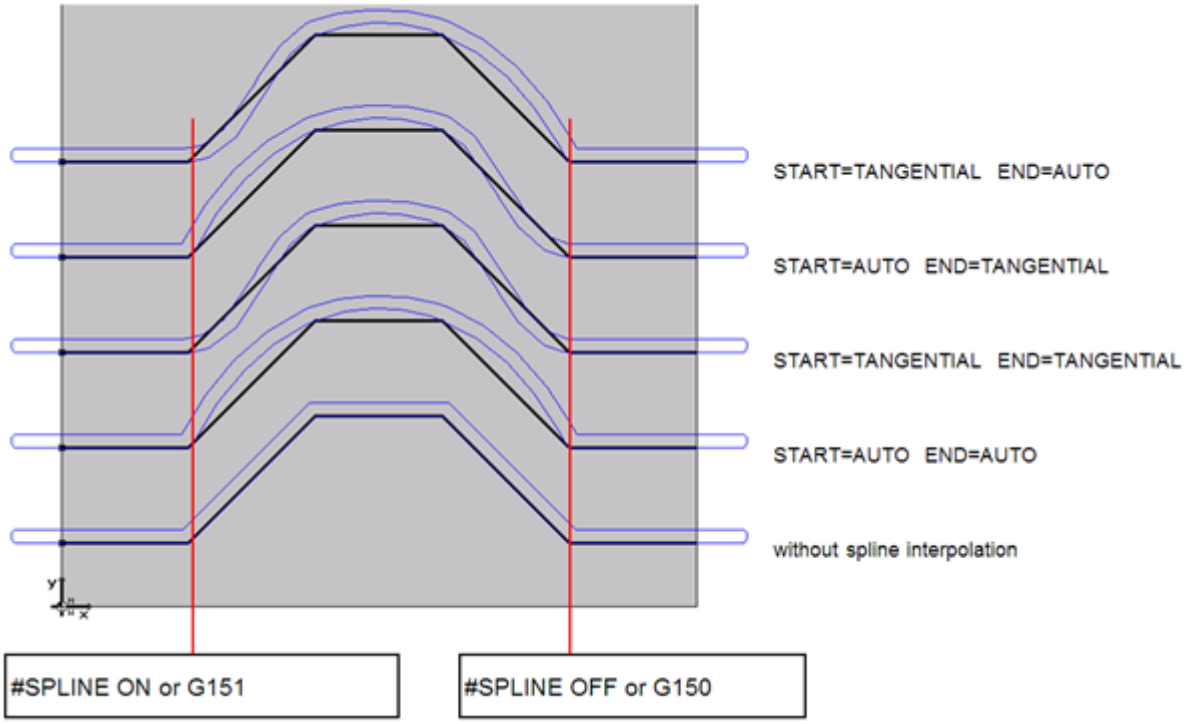


図 73: 図11-8: 遷移タイプ1と2の組み合わせの例

特定の遷移に対して接線が明示的に指定されている場合(遷移タイプ3)、および特定の軸に対して接線がプログラムされていない場合には、この接線は自動的に決定されます(遷移タイプ1)。

12.15.6　開始接線の定義

バージョンV2.11.2010.02以降は、コマンド#AKIMA STARTVECTOR ...をコマンド#SET ASPLINE STARTTANG...の代わりに使用します。互換性の理由から、このコマンドは当面の間使用できますが、新しいNCプログラムでは使用しないでください。

#AKIMA STARTVECTOR {<*axis_name*><*expr*>}	(モーダル)

<*axis_name*><*expr*>	接線ベクトルのコンポーネント(実数値)

軸名を使用して、方向が接線の方向と一致するベクトルを指定します。このため、ベクトルを正規化する必要はありませんが、必ず主軸のすべてのコンポーネントを指定する必要があります。

従軸のベクトルコンポーネントは、従軸がカバーする距離と経路移動距離の割合から計算されます。

ベクトルコンポーネント$_{track}$ = S_{track} / S_{path} $S_{path} = \sqrt{S_x^2 + S_y^2 + S_z^2}$

12.15.7 終了接線の定義

バージョンV2.11.2010.02以降は、コマンド#AKIMA ENDVECTOR ...をコマンド#SET ASPLINE ZIELTANG ...の代わりに使用します。互換性の理由から、このコマンドは当面の間使用できますが、新しいNCプログラムでは使用しないでください。

#AKIMA ENDVECTOR {<*axis_name*><*expr*>}	(モーダル)

<*axis_name*><*expr*>	接線ベクトルのコンポーネント(実数値)

目標接線の定義。開始接線の定義と同様です。

プログラミング例

```
N10 G01 X20 Y0 F1000            (Becomes the first vertice point of the)
                                (spline curve, due to the following G151
N20 #AKIMA TRANS[START=USER END=AUTO]  (Transition type with explicit )
                                      (specifiction of start tangent +)
                                      (automatic determination of the)
                                      (end tangent)
N30 #AKIMA STARTVECTOR X1 Y1 Z0       (Pre-setting of start tangent)
N40 G151                              (Selection of spline-interpolation)
```

```
N50 G01  X40 Y20
N60 X60
N70 Y0
N80 X80
N90 Y10                          (Becomes the last vertice point of the)
                                 (spline curve due to the following G150)
                                 (Support point of the spline-curve)
N100 G150                        (Deselection of spline-interpolation)
N110 X70
N120 M30

The following NC program provides the same result, however, uses for selection and deselection of
the spline interpolation the second variation.
N10 G01 X20  Y0  F1000
N20 #AKIMA TRANS[START=USER END=AUTO]     (Transition type with explicit)
                                          (pre-setting of start tangent +)
                                          (automatic determination of the)
                                          (end tangent)

N30 #AKIMA STARTVECTOR X1 Y1 Z0        (Pre-setting of start tangent)
N40 G151 G01  X40 Y20                  (Selection of spline interpolation)
N50 X60
N60 Y0
N70 X80
N80 Y10
N90 G150 X70                        (Deselection of spline-interpolation)
N100 M30

Caution: Block No. 80 contains the end point of the spline !
```

プログラムにより以下の形状が生成されます。

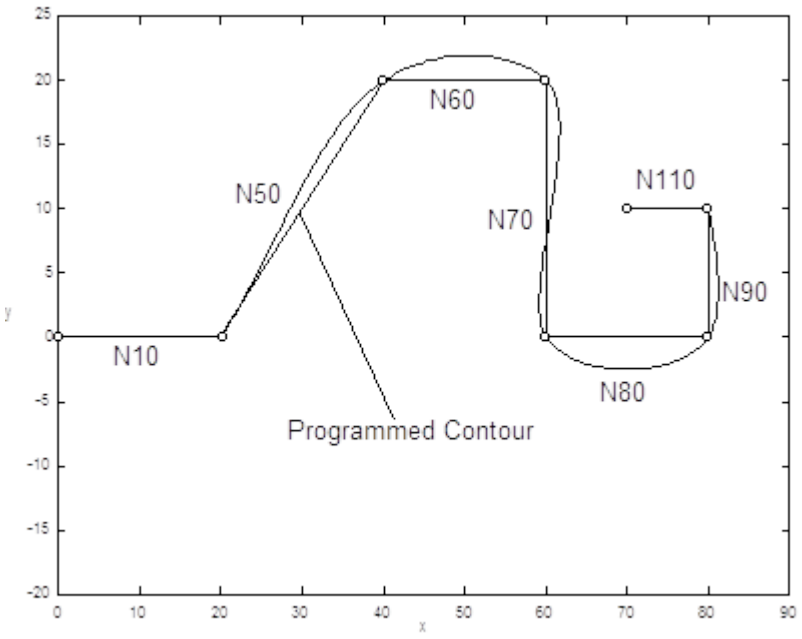


図 74: 図11-9: プログラム例で生成される形状(番号は最初のプログラム例に対応)

ブロックN50に対応する曲線の開始部分(スプライン曲線の開始部分)に、1のプログラムされた傾斜があることが分かります。スプライン曲線の終了部分(ブロックN90の終端)で、傾斜が自動的に計算されます。

円弧ブロック(G02またはG03)が挿入されると円弧ブロックの前にスプライン曲線が途切れ、次の直線ブロックに到達すると新しいスプライン曲線が自動的に開始されます。円弧ブロックへの遷移、および円弧ブロックからの遷移は接線方向に行われます。

直線ブロックが主軸を固定し従軸を移動してプログラムされている場合には、スピンドル曲線も中断されます。曲線の終端で接線が自動的に特定されると、スプライン補間は選択解除されます。従軸は、プログラムされている主軸を移動して直線ブロックまで直線的に補間されます。この場合、スプライン補間が自動的に再選択されます。スプライン曲線への遷移は、主軸と従軸の両方に対して接線方向に起こります。

頂点として機能する直線ブロック間では、その他のファンクション(Mファンクションなど)もプログラムすることができます。ただし、全体で5連続の頂点間でプログラムされたこれらのファンクション数は、設定によって制限されます。

12.16 Bスプライン軸補間

この機能を使用するには、拡張パッケージ「Spline」のライセンスが必要です。このライセンスは、標準ライセンスの範囲には含まれていません。

スプライン軸補間を選択すると、プログラムされた形状点に対してBスプラインが生成されます。

12.16.1 スプラインタイプの選択

バージョンV2.11.2010.02以降では、コマンド**#SPLINE TYPE BSPLINE**をコマンド**#SET SPLINETYPE BSPLINE**の代わりに使用します。互換性の理由から、このコマンドは当面の間使用できますが、新しいNCプログラムでは使用しないでください。

#SPLINE TYPE BSPLINE	(モーダル)

Bスプラインタイプのみが存在する場合には、このタイプが自動的に選択されます。ただし、スプラインタイプは必ず明示的に選択することを推奨します。

12.16.2 スプライン軸補間の選択

バージョンV2.11.2010.02以降では、コマンド#SPLINE ONをコマンド#SET SPLINE ONの代わりに使用します。互換性の理由から、このコマンドは当面の間使用できますが、新しいNCプログラムでは使用しないでください。

#SPLINE ON	(モーダル)

互換性の理由から、G150コマンドで選択することも可能です。曲線は、最後にプログラムされた目標位置で開始します。

12.16.3 スプライン軸補間の解除

バージョンV2.11.2010.02以降では、コマンド#SPLINE OFFをコマンド#SET SPLINE OFFの代わりに使用します。互換性の理由から、このコマンドは当面の間使用できますが、新しいNCプログラムでは使用しないでください。

#SPLINE OFF	(モーダル)

互換性の理由から、G150コマンドで解除することも可能です。

1つのステートメント内でコマンドと位置がプログラムされている場合、この位置はスプライン曲線の一部では**ありません**。

4つ以上の頂点がプログラムされて場合に限り、解除が可能です。

12.16.4 制御点のプログラミング

制御点のプログラミングは、目標点が制御点として使用される直線ブロック(G00、G01)で行います。

曲線が始点と終点でのみ制御点を通るようにする必要があります。

プログラミング例

```
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
```

以下の図は、**プログラミング例**で生成される形状を示しています。

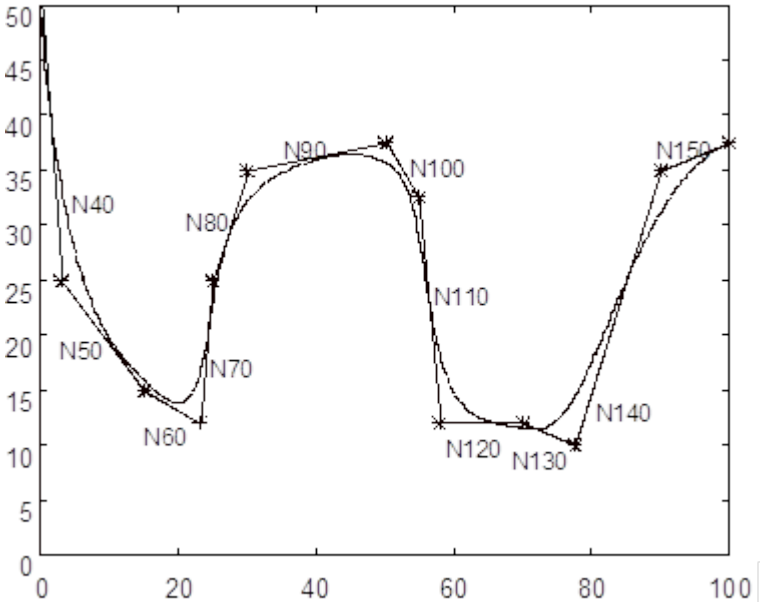


図 75: 図11-10: プログラミング例で生成される形状

この図は、特にブロックN120/N130でのBスプライン曲線の平滑化特性を示しています。この図から分かるように、この曲線は制御点を通りません。ただし、制御点をつないでできるポリゴンにより、曲線の実際の形状が分かります。

Bスプライン曲線の場合は、曲線の終端で接線を直接指定することはできません。ただし、曲線の接線は曲線終端の対応する移動ブロックの接線と同一であるため、曲線終端で移動ブロックを適切にプログラミングすることにより、接線を取得できます。

12.17 自由形状面操作

一般的な自由形状面のプログラムは、CADシステムや試料ワークのデジタル化によって作成されます。通常、これらの表面はブロック長が0.01 mmから数mmの多数のブロックで構成されます。速度変化のない高速な送り速度でプログラムされた形状を移動するのが理想です。サイクルタイムが長くても、形状誤差は僅かです。表面の品質について、特定の状況で経路曲線がより平滑になる場合、形状誤差は許容できます。これらの条件下での要件プロファイルは、以下のようにまとめることができます。

境界条件	制御要件
多くの短いブロックによる形状表現	個々のNCブロックの高スループット、高速な加工速度。
速度変化のない高速な送り速度。	データ供給速度に合った経路速度、インターポレータ内のブロックの迅速な提供。
最小形状誤差	プログラムされた形状と可能な限り等しくなるように補間されます。
高品質な表面	スプラインを使用して頂点シーケンスを平滑化し、発生する形状誤差を監視します。

通常、自由形状面のプログラムではブロック長が短いと、処理が必要なデータ量が多くなります。サイクル時間を増加すると補間頂点間の距離が長くなり過ぎて、複数のブロックがスキップします。このため、論理的であると思われる冗長なブロックを少なくとも部分的に除去します。これは、データ削減のタスクです。計算時間を短縮することでデータスループットが向上し、場合によっては処理時間が短縮されます。

12.17.1 標準HSCプログラミング

#HSC [**ON** | **OFF**] [[**OPMODE** *<expr>*] [**CONTERROR** *<expr>*]] (モーダル)

ON 自由形状モードの有効化

OFF 自由形状モードの無効化

OPMODE *<expr>* 操作モードの設定:

有効な値	優先度
1	遷移多項式の挿入
2	補間スプライン曲線の生成

CONTERROR *<expr>* 最大許容形状誤差の指定:

有効な値	優先度
>=0.0	最大形状誤差「ε」 有効なOPMODE 1との組み合わせでのみ有効です。 OPMODE 2に設定されたパラメータは、OPMODE 1への変更後に有効になります!

#HSC ONコマンドがパラメータなしでプログラムされている場合は、以下の初期値が有効になります。

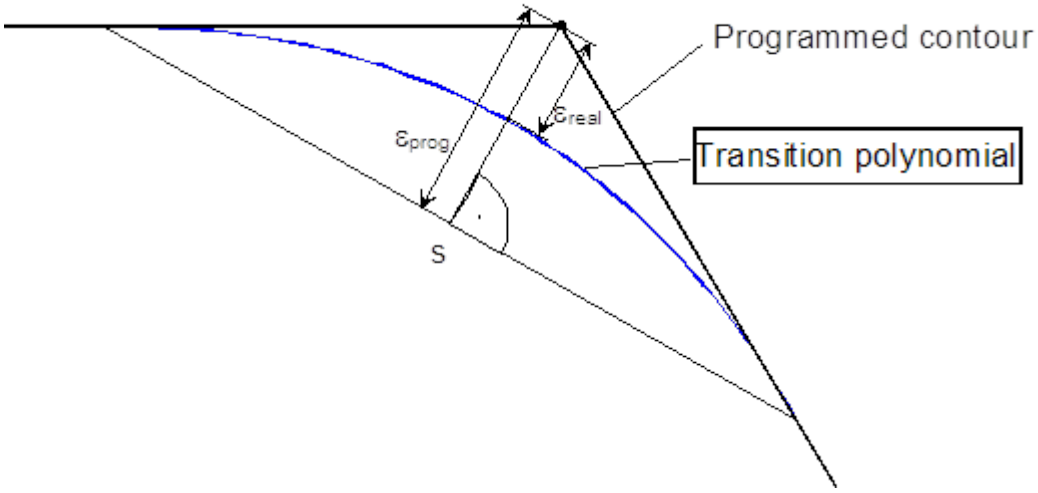
OPMODE	1
CONTERROR	0.1 mm

- パラメータは個別に変更できます。つまり、自由形状操作モード(「OPMODE」)を選択して自由形状モードを有効(「ON」)にできますが、最大形状誤差(「CONTERROR」)は後で変更します。
- 自由形状操作モードは、自由形状モードが有効な場合には変更できません。変更を試みると、エラーメッセージが出力されます。

12.17.1.1 使用可能な操作モード

OPMODE 1: 遷移多項式の挿入

自由形状操作モード1では、動作ブロック遷移時にブロックが短縮され、遷移多項式が挿入されます。



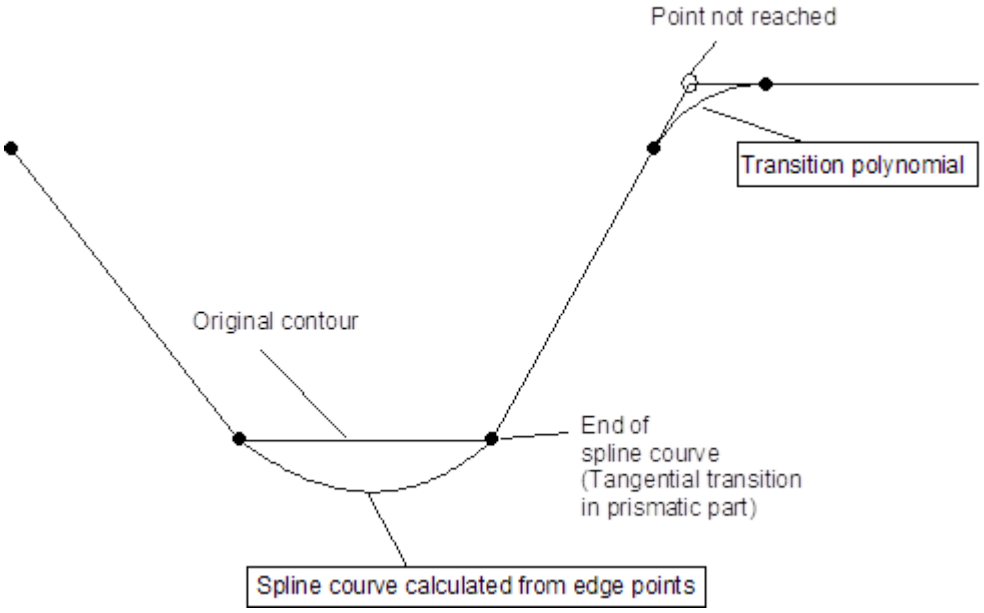
多項式遷移の生成には、最大形状誤差CONTERROR (「ε」)が必要です。つまり、挿入された多項式で実際の形状誤差が低減されます(εreal ≤ εprog)。

膝角度が178°を超える場合、HSCモードが自動的に解除されます。

OPMODE 2: 自由形状部分に対するスプライン曲線の生成

自由形状操作モード2では、指定されたエッジ点でスプライン曲線が生成されます。角柱部の開始時にスプライン生成が自動的に解除され、正接遷移が実行されます。

第11.17.1.2章の追加パラメータに基づいて検出された角柱部でのブロック遷移で、モードが自動的に操作モード1 (OPMODE 1)に自動的に変化します。これは、遷移多項式が挿入されることを意味します。挿入されたスプライン曲線では、頂点間の形状誤差はモニタリングされません。



プログラミング例1

```
...
 #HSC[OPMODE 1  CONTERROR 0.01] (Free form operation mode 1)
                                (Max. contour error <= 0,01 mm)
...
 #HSC ON                    (Enable free form mode)
...
 #HSC OFF                   (Disable free form mode)
...
 #HSC ON                    (Enable free form mode)
                            (Previously selected parameter are active)
                            (Free form operation mode 1)
                            (Max. contour error <= 0,01 mm)
...
 #HSC OFF                   (Disable free form mode)
...
 #HSC ON [OPMODE 2  CONTERROR 0.002]  (Enable free form mode with)
                                      (operation mode 2)
                                      (Maximum contour error <= 0,002 mm)
...
 #HSC[CONTERROR 0.005]      (Change maximum contour error to <= 0,005 mm)
...
 #HSC OFF                   (Disable free form mode)
```

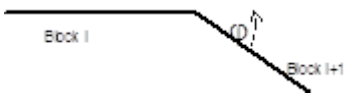
プログラミング例2

```
...
#HSC[OPMODE 1  CONTERROR 0.01]      (Select free form operation mode 1)
                                    (Maximum contour error <= 0,01 mm)

#HSC ON                             (Enable free form mode)
...
#HSC[OPMODE 2]                      (Error message!)
                                    (Changing of operation mode is)
                                    (not allowed while free form mode)
                                    (is active.)
```

12.17.1.2 追加パラメータ

通常、必要となるのは前述のパラメータのみです。しかし、自由形状モード2用に内部的に使用されている設定にアクセスするために、以下の追加パラメータが用意されています。

#HSC [[COS_PHI_MIN<*expr*>] [FACT_BLOCK_LEN<*expr*>]]

HSCパラメータ	タイプ	有効範囲 (初期設定)	説明
COS_PHI_MIN	Real	-1.0 ~ 1.0 (0.7)	 cos(φ)の最小値。値が小さいとスプライン曲線の生成が解除されます。
FACT_BLOCK_LEN	Real	> 0.0 (3.0)	ブロック長が平均ブロック長を超える可能性がある最大係数 *。値が大きいとスプライン曲線の生成が解除されます。 * 平均ブロック長は、前の5つの動作ブロックのブロック長から算出されます。

12.17.2 拡張HSCプログラミング

この機能を使用するには、拡張パッケージ「HSC」のライセンスが必要です。このライセンスは、標準ライセンスの範囲には含まれていません。

注記

拡張HSCプログラミングでプログラムされた制御点では、Bスプラインが作成されます。この種類のHSC操作では、コマンド#SLOPE [TYPE…] [▶ 261]によって先にプロファイルタイプHSC (傾斜3)を選択することを推奨します。

加工タスクに応じて、HSCの選択/解除とパラメータ割り当てには2つの操作モードが使用できます。

モード1は、形状に沿った単一の移動(トリミング)に適しています。これに関連して、形状は高速な送り速度で移動する短い多くのブロックで構成されます。

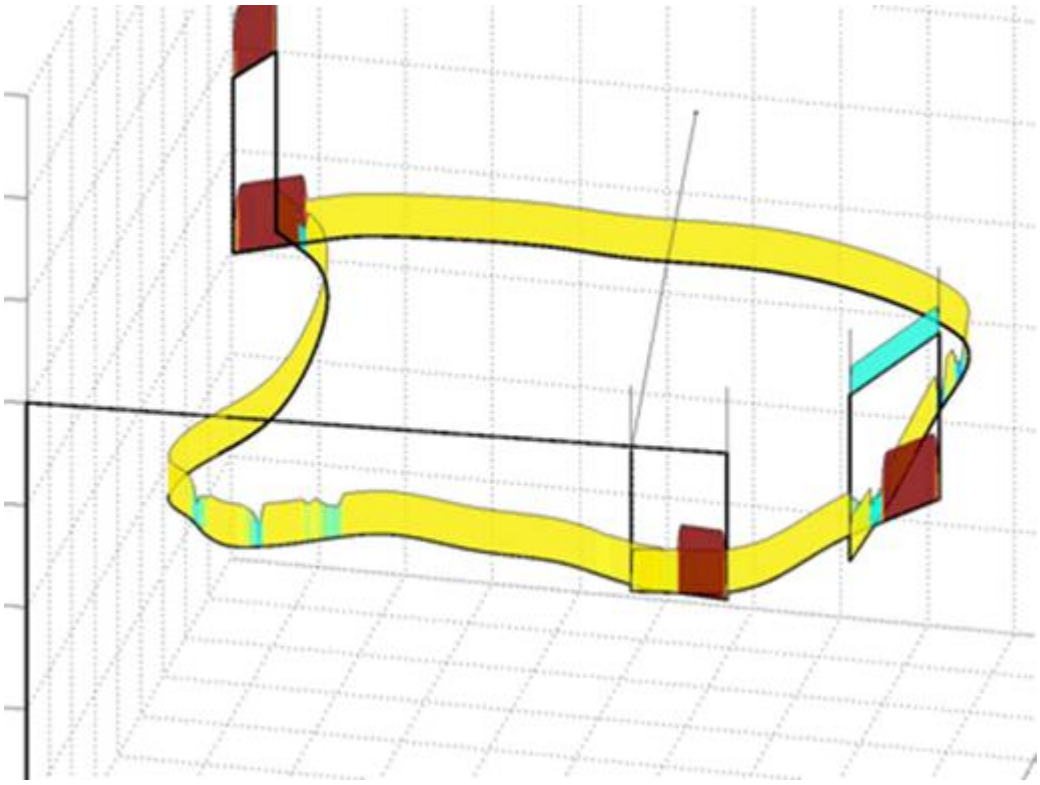


図 76: 図11-11: 形状のトリミング

モード2は、特に自由形状サーフェースの加工に適しています。CADシステムで生成されるこのNCプログラムでは、通常、多くのパスでワークが加工されます(1ライン毎、またはヘリカル)。可能な限り短い処理時間で高品質なサーフェス加工を行うために、いくつかの特殊なアルゴリズムが使用されます(サーフェスオプティマイザ)。

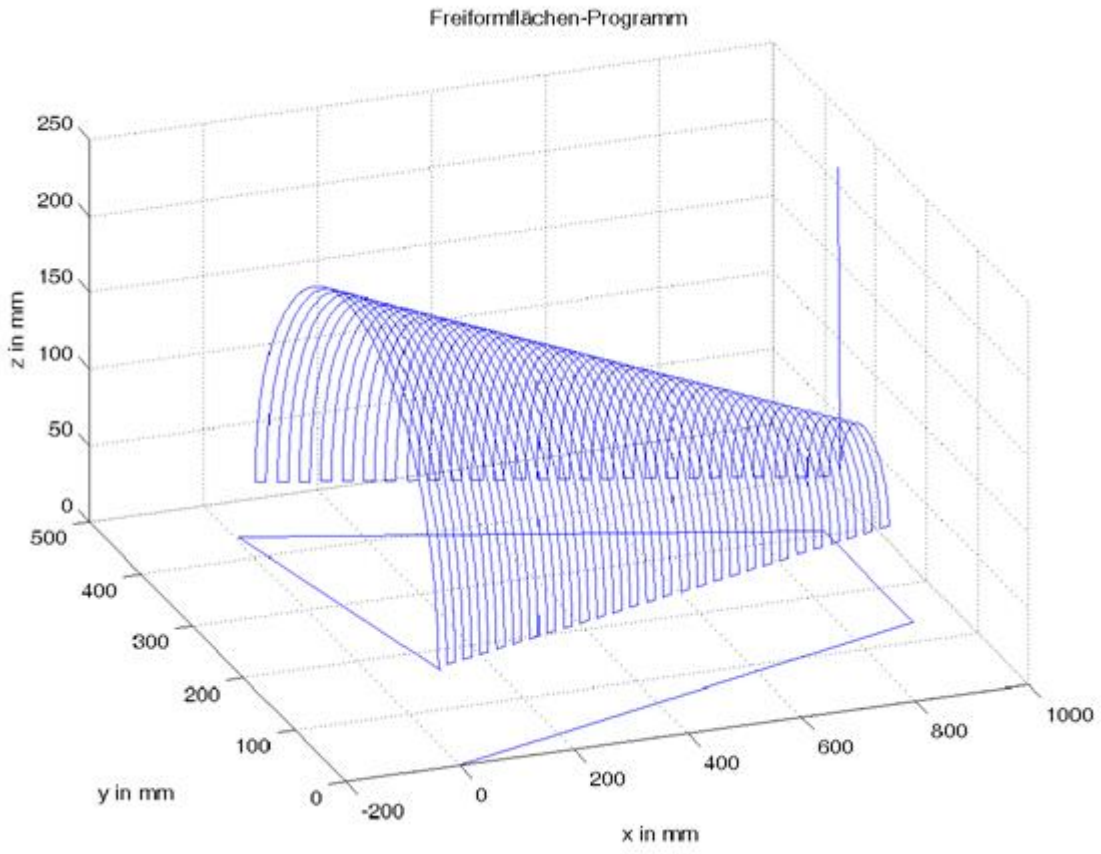
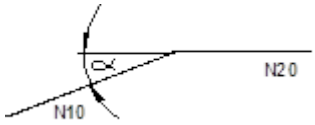


図 77: 図11-12: サーフェスの1ライン毎の研削

12.17.2.1 形状のトリミング

#HSC [ON | OFF] [[BSPLINE [PATH_DEV<*expr*>] [TRACK_DEV<*expr*>] [MERGE<*expr*>] [AUTO_OFF_PATH<*expr*>] [AUTO_OFF_TRACK<*expr*>] [AUTO_OFF_G00<*expr*>] [MAX_PATH_LENGTH<*expr*>] [MAX_ANGLE<*expr*>]]] (モーダル)

ON	自由形状モードが有効。
OFF	自由形状モードが無効。
BSPLINE	BスプラインによるHSC操作の識別。必ず**1番目に**プログラムされるキーワードであること。
PATH_DEV<*expr*>	プログラムされた経路形状からのmm単位でのBスプラインの最大偏差 。この偏差を超過すると、Bスプラインが自動的に解除されます。最大偏差を0に定義すると、経路偏差のモニタリングが実行されません。 初期値: 0.2 mm
TRACK_DEV<*expr*>	従軸の最大偏差(mmまたは度)。最大偏差を0に定義すると、抗力軸のモニタリングが実行されません。 初期値: 5 mmまたは5度
MERGE<*expr*>	ブロックの結合。PATH_DEVおよびTRACK_DEVからの値に従って最大偏差が決定されます。 0: ブロックの結合なし(初期設定) 1: ブロックの結合
AUTO_OFF_PATH<*expr*>	のプログラムされたBスプラインの偏差(PATH_DEV)を超過すると、自動的にブロックが分割されます。 0: 偏差が過大な場合に解除されず(初期設定)、ブロックを分割 1: 偏差が過大な場合に解除
AUTO_OFF_TRACK<*expr*>	抗力軸のプログラムされたBスプラインの偏差(TRACK_DEV)を超過すると、自動的にブロックが分割されます。 0: 偏差が過大な場合に解除されず(初期設定)、ブロックを分割 1: 偏差が過大な場合に解除
AUTO_OFF_G00<*expr*>	G00ブロックに対するBスプライン軸補間を自動的に解除します。 0: 高速動作ブロックにより暗黙的に解除されない(初期設定) 1: 高速動作ブロックにより暗黙的に解除される
MAX_PATH_LENGTH<*expr*>	関連するポストブロックの最小経路長(mm)。設定された長さよりもブロック長が長いと、Bスプラインが自動的に解除されます。 初期値: 0 mm (ブロック長が発生しないため暗黙的に解除される)
MAX_ANGLE<*expr*>	Bスプラインが挿入されるまでの、2つの直線ブロック間の遷移の最大形状角度。この2つの直線ブロック間の角度がこのリミット値を超過すると、Bスプラインが内部的に解除されます。 初期値: 0° (形状角度により暗黙的に解除されない)



形状点のプログラミングが、直線ブロック(G00およびG01)によって行われます。目標点が形状点として使用されます。始点と終点でのみ曲線が制御点を通って直線になることが考慮されています。

パラメータは複数のステップで変更できます。つまり、例えば最初のステップで最大形状偏差(「PATH_DEV」)、2番目のコマンドで最大経路長(「MAX_PATH_LENGTH」)およびBスプライン軸補間の選択(「ON」)を定義できます。

注記

Bスプライン軸補間が有効な場合は、パラメータ設定を変更できません。

プログラミング例

```
The spline curve is based on the control positions N40 - N155, but the spline curve only is going
straight through at N20 and N50.

N20 G00  X0   Y0   Z0 F10000
N30 #HSC ON [BSPLINE PATH_DEV=0.2 MERGE=1..] Parametrization + Selection
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                                Deselection
N170 M30
```

または

```
N20 G00  X0   Y0   Z0 F10000
N25 #HSC [BSPLINE PATH_DEV=0.2 MERGE=1..]    Parametrization
N30 #HSC ON                                  Selection
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                                Deselection
N170 M30
```

12.17.2.2 サーフェスオプティマイザでのサーフェス加工

#HSC [ON | OFF] [[SURFACE [PATH_DEV<*expr*>] [TRACK_DEV<*expr*>] [PATH_DEV_G00<*expr*>] [TRACK_DEV_G00<*expr*>] [MAX_ANGLE<*expr*>] [CHECK_JERK<*expr*>] [AUTO_OFF_G00<*expr*>]]] (モーダル)

ON	自由形状モードが有効。
OFF	自由形状モードが無効。
SURFACE	サーフェスオプティマイザによるHSC操作の識別。必ず**1番目に**プログラムされるキーワードであること。
PATH_DEV<*expr*>	最大形状誤差の定義。
PATH_DEV_G00<*expr*>	G0-G0遷移の最大形状誤差の定義。 0.0よりも大きな値: 最大経路偏差(mm) 初期値: 0.2 mm **注記 経験則的に、形状誤差は割線誤差よりも2～3倍大きく設定する必要があります。これは、CAMシステムによるNCプログラム生成時に定義されます。G0動作中は、工具が接触しません。これにより、工具ピース精度に影響を与えずに、許容値をPATH_DEFよりも非常に大きく設定できます。**
TRACK_DEV<*expr*>	最大方向誤差の定義。
TRACK_DEV_G00<*expr*>	G0-G0遷移の最大方向誤差の定義。 0.0以上の値: 最大経路偏差(mm) 初期値: 2° **注記 ボール研削カッターを使用する場合、この値をPATH_DEVよりも非常に大きく(例えば10倍に)設定できます。G0動作中は、工具が接触しません。これにより、工具ピース精度に影響を与えずに、許容値をTRACK_DEFよりも非常に大きく設定できます。**
MAX_ANGLE<*expr*>	Bスプラインが挿入されるまでの、2つの直線ブロック間の遷移の最大形状角度。この2つの直線ブロック間の角度がこのリミット値を超過すると、Bスプラインが内部的に解除されます。 0.0以上の値: 最大経路角度(°) 初期値: 160° 

CHECK_JERK<*expr*> 多項式の曲率によって発生する加加速度のモニタリング(P-CHAN-00110)。

このパラメータは、チャンネルパラメータP-CHAN-00110 (check_jerk_on_poly_path)によって定義される初期設定を上書きします。

0: 加加速度モニタリングなし

1: 形状的なランプ時間P-AXIS-00199に基づいた加加速度モニタリング。経路速度が減衰される可能性があります。

2: 非直線速度プロファイルP-AXIS-00195 ~ P-AXIS-00198のランプ時間に基づいた加加速度モニタリング。

AUTO_OFF_G00<*expr*> G00ブロックに対するBスプライン軸補間を自動的に解除します。

0: 高速動作ブロックにより暗黙的に解除されない(初期設定)

1: 高速動作ブロックにより暗黙的に解除される

自由形状面加工用の#HSC ONコマンドがパラメータなしでプログラムされている場合は、以下の初期値が有効になります。

PATH_DEV	0.2mm
TRACK_DEV	2°
PATH_DEV_G00	0.2mm
TRACK_DEV_G00	2°
MAX_ANGLE	160°
CHECK_JERK	チャンネルパラメータが有効 P-CHAN-00110 (check_jerk_on_poly_path、初期設定は1)
AUTO_OFF_G00	0

パラメータは複数のステップで変更できます。つまり、例えば最初のステップで最大形状偏差(「PATH_DEV」)、2番目のコマンドで加加速度モニタリング(「CHECK_JERK」)およびHSC面補間の選択(「ON」)を定義できます。

注記

平滑化が有効な場合は、パラメータ設定を変更できません。

注記

このファンクションをチャンネル内で使用する必要がある場合は、対応する起動リスト内でこのファンクションを有効にする必要があります。

スタートアップリストの設定例:

configuration.channel[].path_preparation.function FCT_DEFAULT|FCT_SURFACE

プログラミング例

```
N20 G00  X0   Y0   Z0 F10000
N30 #HSC ON [SURFACE PATH_DEV=0.02 CHECK_JERK=0] Parametrization + Selection
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                                  Deselection
N170 M30
```

または

```
N20 G00  X0   Y0   Z0 F10000
N25 #HSC [SURFACE PATH_DEV=0.02 CHECK_JERK=0]    Parametrization
N30  #HSC ON                                     Selection
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                                  Deselection
N170 M30
```

12.17.3 フィルタプログラミング

自由形状サーフェスの加工時に高品質のサーフェス仕上げを実現するためには、可能な限り機械振動の発生を抑える必要があります。HSCフィルタを使用すると、ドライブに対して命令された軸値を平滑化し、機械の加振を低減することが可能です。この結果、プログラムされた形状から乖離しますが、これはユーザによって制限できます。従来のローパスフィルタとは異なり、HSCフィルタを使用すると形状に歪みが生じません。つまり、径方向の並行パスが並行を維持します。

注記

この機能を使用するには、軸特有のフィルタパラメータ[2]-3を完全かつ正確に設定する必要があります。

軸リストの設定例:

filter[0].order 0

filter[0].prototype 5 #Typ HSC-Filter

filter[0].type 1

filter[0].fg_f0 20

filter[0].guete 100

filter[0].share_percent 100

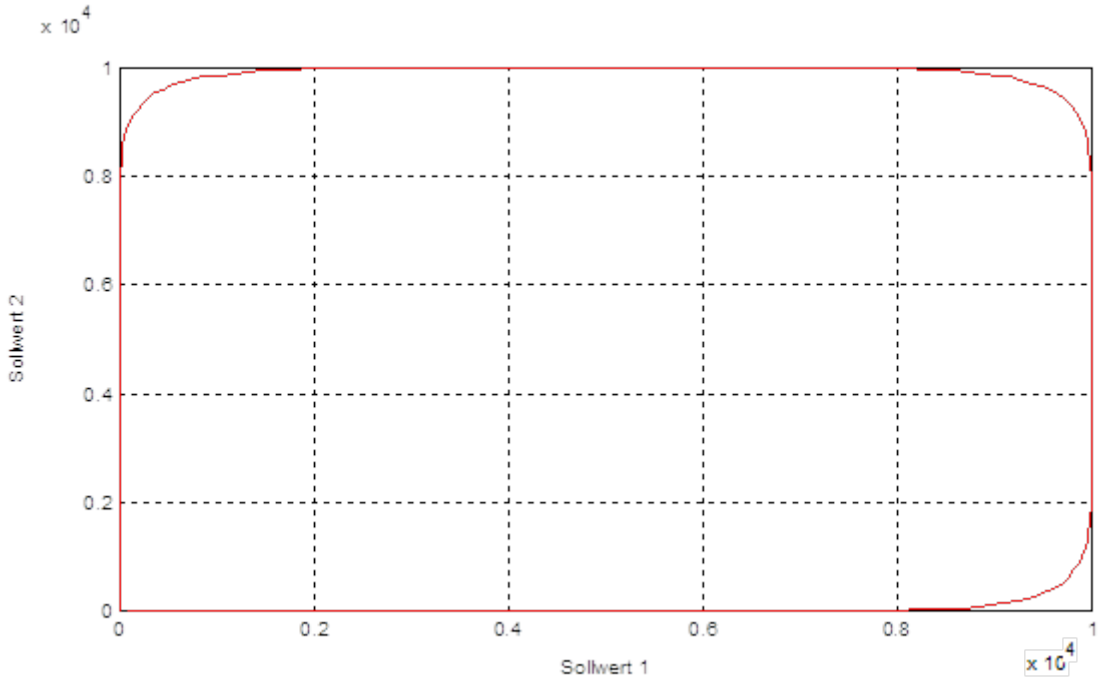
フィルタプログラミングの選択には、以下のNCコマンドを使用できます。

#FILTER [ON | OFF] [HSC]	(モーダル)

HSC HSCフィルタを選択するためのキーワード。

プログラミング例

```
N1111 #FILTER ON [HSC]
N10 X0 Y0
N20 X0 Y1
N30 X1 Y1
N50 X1 Y0
N40 X0 Y0
N4001 #FILTER OFF [HSC]
```



設定値= コマンド値

12.18 SERCOSパラメータおよびコマンドの書き込み/読み取り

12.18.1 SERCOSパラメータ

SERCOSパラメータの書き込み/読み取りには、以下のNCコマンドを使用します。従来のSERCOS形式[4]は、プログラミングを簡略化するために使用します。

ドライブ内のパラメータ識別番号を正しく処理するには、軸名や論理軸番号、コード、および属性などの詳細情報が必要です。この情報が、同一のNCコマンド内の識別番号と紐付けてプログラムされます。

「非同期」とは、解読中にコマンドが実行されることを意味します。

「同期」とは、補間レベルの処理に同期したコマンドの実行を意味します。

12.18.1.1 パラメータの非同期書き込み

```
#IDENT WR [ AX<axis_name> / AXNR<expr> ID<ident_nr> VAL<expr> TYP<expr>
            DEC<expr> <drive_type> ]
```

AX*<axis_name>* SERCOS軸名

AXNR*<expr>* SERCOS軸の論理軸番号(正の整数)

ID*<ident_nr>* SERCOS形式の識別番号(例えば*S-0-0047*や*P-0-0129*)

VAL*<expr>* 書き込まれる値(実数値)

TYP*<expr>* 値のデータ型(2または4バイト長):

有効な値	優先度
2	データ長2バイト
4	データ長4バイト

DEC*<expr>* 小数点以下の値(正の整数)

<drive_type> ドライブタイプ

有効なID	優先度
SERC	SERCOSドライブ(現状ではドライブのみ使用可能)

プログラミング例

```
:
#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]
:
```

注記

論理軸番号、識別番号、プログラムされた属性データ型、および小数点の位置に関する妥当性チェックは行われません。オペレータが正しく入力をする必要があります。

12.18.1.2 パラメータの非同期読み取り

#IDENT RD [AX*<axis_name>* **| AXNR***<expr>* **ID***<ident_nr>* **P =***<variable>* **TYP***<expr>*
DEC*<expr>* **<***drive_type>* **]**

AX*<axis_name>* SERCOS軸名

AXNR*<expr>* SERCOS軸の論理軸番号(正の整数)

ID*<ident_nr>* SERCOS形式の識別番号(例えば*S-0-0047*や*P-0-0129*)

<variable> 読み取った値が保存される変数。
Pパラメータや V.P.、V.L.または V.S.変数など。

TYP*<expr>* 値のデータ型(2または4バイト長):

有効な値	優先度
2	データ長2バイト
4	データ長4バイト

DEC*<expr>* 小数点以下の値(正の整数)

<drive_type> ドライブタイプ

有効なID	優先度
SERC	SERCOSドライブ(現状ではドライブのみ使用可能)

プログラミング例

```
:
#IDENT RD [AX X  ID S-0-0104  P=P1  TYP 4  DEC 2  SERC] or


#IDENT RD [AXNR 1  ID S-0-0104  P=V.P.KV_WERT  TYP 4  DEC 2  SERC]
:
```

注記

論理軸番号、識別番号、プログラムされた属性データ型、および小数点の位置に関する妥当性チェックは行われません。オペレータが正しく入力をする必要があります。

12.18.1.3 パラメータの同期書き込み

#IDENT WR SYN [AX*<axis_name>* **| AXNR***<expr>* **ID***<ident_nr>* **VAL***<expr>* **TYP***<expr>*
DEC*<expr>* *<drive_type>* **[NO_WAIT]]**

AX*<axis_name>* SERCOS軸名

AXNR*<expr>* SERCOS軸の論理軸番号(正の整数)

ID*<ident_nr>* SERCOS形式の識別番号(例えば*S-0-0047*や*P-0-0129*)

VAL*<expr>* 書き込まれる値(実数値)

TYP*<expr>* 値のデータ型(2または4バイト長):

有効な値	優先度
2	データ長2バイト
4	データ長4バイト

DEC*<expr>* 小数点以下の値(正の整数)

<drive_type> ドライブタイプ

有効なID	優先度
SERC	SERCOSドライブ(現状ではドライブのみ使用可能)

NO_WAIT パラメータが正常に書き込まれるまで待機しません。

このキーをプログラミングしない場合は、必ずパラメータが正常に書き込まれてから補間が行われます。

プログラミング例

```
:
#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]
:
```

注記

論理軸番号、識別番号、プログラムされた属性データ型、および小数点の位置に関する妥当性検査は行われません。オペレータが正しく入力をする必要があります。

12.18.2 SERCOSコマンド

以下のNCコマンドにより、SERCOSコマンドを開始します。従来のSERCOS形式[4]は、プログラミングを簡略化するために使用します。

ドライブ内のコマンド識別番号を正しく処理するには、軸名や論理軸番号、コードなどの詳細情報が必要です。この情報が、同一のNCコマンド内の識別番号と紐付けてプログラムされます。

「非同期」とは、解読処理中にコマンドが実行されることを意味します。

「同期」とは、補間レベルの処理に同期したコマンドの実行を意味します。

12.18.2.1 コマンドの非同期書き込み

#COMMAND WR [AX<*axis_name*> | AXNR<*expr*> ID<*ident_nr*> <*drive_type*>]

AX<*axis_name*> SERCOS軸名

AXNR<*expr*> SERCOS軸の論理軸番号(正の整数)

ID<*ident_nr*> SERCOS形式でのコマンドの識別番号。

例: *S-0-0148*など(ドライブ制御のリファレンシング)

または*S-0-0170*(トレースサイクル)

<*drive_type*> ドライブタイプ

有効なID	優先度
SERC	SERCOSドライブ(現状ではドライブのみ使用可能)

プログラミング例

```
:
#COMMAND WR [AX=X  ID S-0-0148  SERC]

#COMMAND WR [AXNR 1, ID S-0-0148, SERC]
:
```

12.18.2.2 コマンドの同期書き込み

#COMMAND WR SYN [AX<*axis_name*> | AXNR<*expr*> ID<*ident_nr*> <*drive_type*>]

AX<*axis_name*> SERCOS軸名

AXNR<*expr*> SERCOS軸の論理軸番号(正の整数)

ID<*ident_nr*> SERCOS形式でのコマンドの識別番号。

例: *S-0-0148*など(ドライブ制御のリファレンシング)

または*S-0-0170*(トレースサイクル)

<*drive_type*> ドライブタイプ

有効なID	優先度
SERC	SERCOSドライブ(現状ではドライブのみ使用可能)

プログラミング例

```
:
#COMMAND WR SYN [AX Y, ID S-0-0170, SERC]

#COMMAND WR SYN [AXNR 2  ID S-0-0170  SERC]
:
```

注記

論理軸番号、および識別番号に関する妥当性検査は行われません。オペレータが正しく入力をする必要があります。

12.18.2.3 コマンドの非同期待機

#COMMAND WAIT [AX *<axis_name>* **/ AXNR***<expr>* **ID***<ident_nr>* *<drive_type>* **]**

AX*<axis_name>* SERCOS軸名

追加ID	優先度
ALL	システムのすべてのSERCOS軸のチェック

AXNR*<expr>* SERCOS軸の論理軸番号(正の整数)

ID*<ident_nr>* SERCOS形式でのコマンドの識別番号。

例: *S-0-0148*など(ドライブ制御のリファレンシング)

または*S-0-0170* (トレースサイクル)

識別番号がプログラムされていない場合は、すべての

オープンコマンドに対して待機が有効になります。

<drive_type> ドライブタイプ

有効なID	優先度
SERC	SERCOSドライブ(現時点ではドライブのみ使用可能)

プログラミング例

```
...
#COMMAND WAIT [AX X, ID S-0-0148, SERC]
:
#COMMAND WAIT [AX ALL  ID S-0-0148  SERC]
...
```

注記

論理軸番号、および識別番号に関する妥当性検査は行われません。オペレータが正しく入力をする必要があります。

12.18.2.4 コマンドの同期待機

#COMMAND WAIT SYN [AX *<axis_name>* **/ AXNR***<expr>* **ID***<ident_nr>* *<drive_type>* **]**

AX*<axis_name>* SERCOS軸名

追加ID	優先度
ALL	システムのすべてのSERCOS軸のチェック

AXNR*<expr>* SERCOS軸の論理軸番号(正の整数)

ID*<ident_nr>* SERCOS形式でのコマンドの識別番号。

例: *S-0-0148*など(ドライブ制御のリファレンシング)

または*S-0-0170* (トレースサイクル)

識別番号がプログラムされていない場合は、すべてのオープンコマンドに対して

待機が有効になります。

<drive_type> ドライブタイプ

有効なID	優先度
SERC	SERCOSドライブ(現時点ではドライブのみ使用可能)

プログラミング例

```
...
#COMMAND WAIT SYN [AX=X, ID=S-0-0148, SERC]
:
#COMMAND WAIT SYN [AX ALL, ID S-0-0148, SERC]
...
```

注記

論理軸番号、および識別番号に関する妥当性検査は行われません。オペレータが正しく入力をする必要があります。

SERCOSコマンドを待機するために、動作は完全には停止しません。ただし、SERCOSコマンドが動作ブロックの終端で終了しない場合、新しいNCブロックは処理されず、動作が停止します。

プログラミング例

コマンドS-0-0148が終了するまで、ブロックN120の終端で待機します。

```
..
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
...
```

注記

WAITコマンドは、同レベルの処理で命令されたコマンドでのみチェックできます(コンテキストの解読、または補間レベルでの同期)。例えば、同期されたコマンド(「SYN」)のみ補間レベルでチェックできます。

プログラミング例

コマンドS-0-0148は依然として有効ですが、解読レベルでは有効なコマンドS-0-0148が検出されないため、ブロックN110では待機が開始されません。補間レベルでのコマンドの実際の状態のみが、ブロックN120で検出できます。

```
...
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]
 ...
```

12.19 チャンネル同期

マルチチャンネルコントロールの操作時(特にn > 2の場合)、チャンネル間で特定の実行シーケンスの順守を必要とする状況が発生する場合があります。

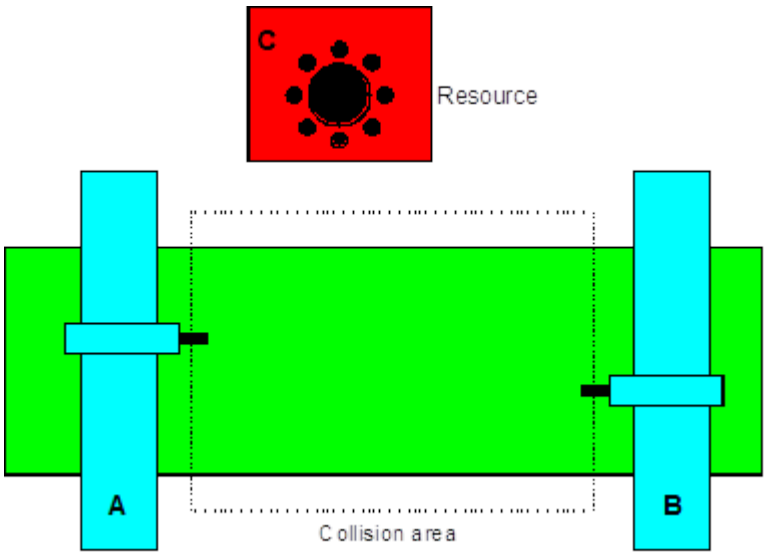


図 78: 図11-13: 使用例、工具チェンジャを搭載したダブルカラム加工機

図11-13は、2つの機械ユニット(AおよびB)が共通の加工スペースを共有する使用例を示しています。同様に、両方の機械が工具チェンジャリソース(C)を使用します。このような機械構成で衝突を回避するために、制御システムの各種チャンネルのNCプログラムを互いに特定のポイントで同期する必要があります。上記の例では、ユニットAは、衝突スペース内にユニットBがある場合には、そこに進入できません。同様に、ユニットAが工具チェンジャにアクセスしている場合、ユニットBは工具チェンジャを使用できません。

図11-14に示された制御システムの2つのチャンネルでのタイムシーケンスは、工具チェンジャへのアクセスのインスタンスになります。

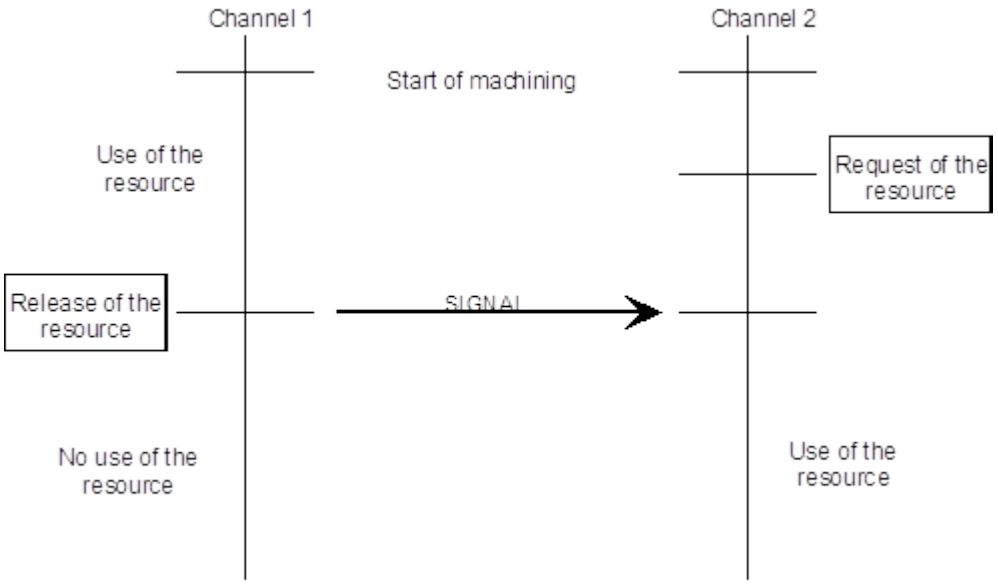


図 79: 図11-14: リソースへの共有アクセスのシーケンス

必要な同期は信号の送信および待機に基づいており、下記のNCプログラムに記載されたNCコマンドによって実行されます。

12.19.1 信号の送信

基本的に、レシーバの指定をしないブロードキャスト信号と、レシーバとしてチャンネルが明示的に指定される非ブロードキャスト信号が区別されます。信号は固有の番号によって識別されます。ここでは、同一のシグナル番号の信号送信が許可されます。

非ブロードキャスト信号の場合、1つ以上のNCチャンネルが明示的にレシーバとして指定されます。1つの信号に対して複数のレシーバが指定されている場合、個別のチャンネルへの同一信号の複数送信と同様に動作します。

例:

```
#SIGNAL [ID4711 CH1 CH2 CH3]
```

以下と同様に動作します。

```
#SIGNAL [ID4711 CH1]
#SIGNAL [ID4711 CH2]
#SIGNAL [ID4711 CH3]
```

これらの信号は、アドレス指定されたレシーバのWAITに対してのみ有効で、レシーバチャンネルのWAITに対して消費カウンタ(COUNT)が指定されていない場合は使い果たされます。消費カウンタが指定されている場合は、それに応じて有効期間を延長できます。

この状況とは対照的に、ブロードキャスト信号は任意のチャンネルのWAITによって受信できます。ブロードキャスト信号に対して消費カウンタ(COUNT)が指定されていない場合は、WAITは信号を使い果たしません。つまり、明示的にクリアされるまで信号が保持されます(#SIGNAL REMOVEを参照)。消費カウンタが指定されている場合は、非ブロードキャスト信号と同様に、信号が使い果たされるまでWAITを要求できます。

#SIGNAL[<*mode*>] [**ID**<*sgn_nr*> [**COUNT**<*expr*>] { **P[**<*idx*>**]=**<*param*>} {**CH**<*chan_nr*>}] （ノンモーダル）

<*mode*>	同期モード。使用可能な同期モード: --- デコーダレベルでの同期(基本設定)。 パラメータや変数との同期が要求される場合などは、この同期が必要です。 SYN インターポレータレベルでの同期。 マルチカラム加工機の2つの加工ユニットの同期など、リアルタイム要求の場合にこの同期が必要です。
ID<*sgn_nr*>	シグナル番号(システム全体で一意、 正の整数)。
COUNT<*expr*>	消費カウンタ。#WAITによって信号を取得できる頻度を定義します。正の整数)。
P[<*idx*>] = <*param*>	信号パラメータ。この実数値<param>は、指定されたインデックスに紐付けられ、信号のパラメータ配列に格納されます。 <*idx*> パラメータの可能な最大数の範囲: 0～信号パラメータの最大数[(1)]
CH<*chan_nr*>	信号が割り当てられるチャンネル番号 正の整数)。 1～最大のチャンネル番号[(2)] チャンネル番号が指定されていない場合、システムのすべての有効なシグナルレシーバに**ブロードキャスト**信号が送信されます。

(1) [6]-6.45を参照。 (2) [6]-2.4を参照。

プログラミング例

```
(Signal 812, Synchronization on DEC-level, Broadcast)
N500 #SIGNAL [ID812]

(Signal 4711, Synchronization on DEC-level, to channnel 2)
N100 #SIGNAL [ ID4711 CH2 ]

(Signal 4711, Synchronization on DEC-level,
 for 10 #WAIT-queries, Broadcast)
N100 #SIGNAL [ ID4711 COUNT10 ]

(Signal 815, Synchronization on IPO-Ebene,
 twice to channel 2 and once to 3)
N200 #SIGNAL SYN [ ID815 CH2 CH2 CH3 ]

(Signal 911, Synchronization on DEC-level, to channel 3)
(1. signal parameter V.A.MESS.X, 2. signal parameter P200,
 3. signal parameter 944)
N260 P200 = 924
N300 #SIGNAL [ IDP100 CH3 P[0]=V.A.MESS.X P[1]=P200 P[2]=944 ]
```

12.19.2 ブロードキャスト信号のクリア

通常、信号は割り当てられたWAITで使用後にクリアされます。さらに、通常の信号すなわち非ブロードキャスト信号は、NC-Resetの場合には指定しなくてもクリアされます(「リセット処理」の章も参照)。ブロードキャスト信号の中には、使用カウンタが指定されない場合にWAITではクリアされない信号が存在するため、ブロードキャスト信号は明示的に削除する必要があります。この目的のために、追加のNCコマンドが存在します。このNCコマンドは通常の信号をクリアするためにも使用でき、その場合は、識別番号とアドレス指定されたチャンネルが対応している必要があります。

クリアで単一の信号が指定された場合、複数の同じ信号が存在する場合でも、1つの信号のみが削除されます。他方、複数の同じシグナル番号を含む信号範囲が指定された場合([ID; IDMAX])、それらが存在する場合、すべての信号が削除されます。

#SIGNAL REMOVE [<*mode*>] **[ID**<*sgn_nr*> / **IDMIN**<*sgn_nr*> [**IDMAX**<*sgn_nr*>]
{**CH**<*chan_nr*>}] (ノンモーダル)

<*mode*>	同期モード。使用可能な同期モード: --- デコーダレベルでの同期(基本設定)。 パラメータや変数との同期が要求される場合などは、この同期が必要です。 SYN インターポレータレベルでの同期。 この同期は、例えば、マルチカラム加工機の2つの加工ユニットの同期のようなリアルタイム要求の場合に必要です。
ID<*sgn_nr*>	クリアするブロードキャスト信号の番号。正の整数
IDMIN<*sgn_nr*>	クリアするブロードキャスト信号範囲の最初の信号。さらに、ID<*sgn_nr*>の代替パラメータとしても使用されます。正の整数)。
IDMAX<*sgn_nr*>	クリアするブロードキャスト信号範囲の最後の信号。正の整数)。
CH<*chan_nr*>	クリアする信号の宛先であるチャンネルの番号。 1～最大のチャンネル番号[(1)] チャンネル番号を指定しないと、対応する**ブロードキャスト**信号がクリアされます。

(1) [6]-2.4を参照

プログラミング例

```
(Deleting of a broadcast signal 812, Synchronization on DEC-level)
N500 #SIGNAL REMOVE [ID812]  oder  #SIGNAL REMOVE [IDMIN812]

(Deleting of the signal 812 to Kanal2, Synchronization on DEC-level)
N500 #SIGNAL REMOVE [ID812 CH2]

(Deleting of all broadcast signals between 812-820, Synchronization on DEC-level)
N500 #SIGNAL REMOVE [IDMIN812 IDMAX820]  oder  #SIGNAL REMOVE [ID812 IDMAX820]

(Deleting of all signals 812 to Kanal 1, Synchronization on DEC-level)
N500 #SIGNAL REMOVE [ID812 IDMAX812 CH1]

(Deleting of the broadcast signals 813, Synchronization on IPO-level)
N600 #SIGNAL REMOVE SYN [ID813]
```

12.19.3 信号の待機

信号の送信の場合と同様に、WAITコマンドを使用して対応するSIGNALを待機することができます。ブロードキャストWAITは、同じシグナル番号を持つブロードキャストSIGNALを待ちます。デコーダ/インターポレータレベルで同期されるWAITは、それぞれの場合に、1つの別のSIGNALを消費します。

#WAIT [<*mode*>] **[ID**<*sgn_nr*> { **P[**<*idx*>**] =** <*param*> } { **CH**<*chan_nr*> } [**AHEAD**] **]** (ノンモーダル)

<*mode*> 同期モード

有効なモード	優先度
---	デコーダレベルでの同期(基本設定)。 パラメータや変数との同期が要求される場合などは、この同期が必要です。
SYN	インターポレータレベルでの同期。 マルチカラム機械の2つの加工ユニットの同期など、リアルタイム要求の場合にこの同期が必要です。

ID<*sgn_nr*> それに対する同期が行われる(システムが待機する)信号の番号。正の整数)。

P[<*idx*>] = <*param*> 信号パラメータ(実数)。信号を待機している間に、異なるチャンネルの信号送信元からパラメータが送信される場合もあります。これらのパラメータは、指定されたパラメータまたは変数に割り当てられます(<*param*>)。

インデックス	優先度
<*idx*>	パラメータの可能な最大数の範囲: 0 ~ 信号パラメータの最大数(1)

待機条件の完全な確認後(すべての必要な信号の受信後)、プログラムされたすべてのパラメータの評価および保管が終了したかどうかがチェックされます。このチェックで否定応答が返された場合には、エラーメッセージが作成されます。

信号パラメータは、現在、デコーダレベルでのみ評価されます。これは、例えば、#WAIT SYN[... P[0] = ...]が許可されないことを示します。

CH<*chan_nr*> チャンネル番号:

有効値	優先度
1～最大のチャンネル番号[2]	信号の受信が予測されるチャンネルの番号。
---	チャンネル番号を指定しないと、システムは任意のユーザの信号を待機します。

AHEAD 「フライング」WAITの実行のキーワード。Look-aheadのバッファ効果(最大70の先行ブロック)による待機時間の減少のために使用されます。瞬時WAITインターポレータレベル出力の同期の以下の確認チェック(SIGNAL)では、次のブロックの移動を中断なく実行できます。

[1] [6]-6.45を参照

[2] [6]-2.4を参照

プログラミング例

```
(Wait mark 4711, Synchronization on DEC-level, SIGNAL 4711 from any channel)
N200 #WAIT [ID4711]

(Wait mark 815, Synchronization on IPO-level, SIGNAL 815 from channel 2 and 3)
N100 #WAIT SYN [ID815 CH2 CH3]

(Wait mark 911, Synchronization on DEC-level, from channel 3)
(1. signal parameter V.P.SIGNAL, 2. signal parameter P200)
N300 #WAIT [IDP100 P[0]=V.P.SIGNAL P[1]=P200 CH3]
N350 P20 = 10 * V.P.SIGNAL * P200
```

プログラミング例

パラメータの評価付きの信号の待機(チャンネル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  (Stores value from channel 1
N20 P2 = 1  (Stores value from channel 2
N30 XP1 YP2
N40 #WAIT [ID 110014 P[0] = P1 P[1] = P2  CH1 CH2]
N50 XP1 YP2
N60 M30
```

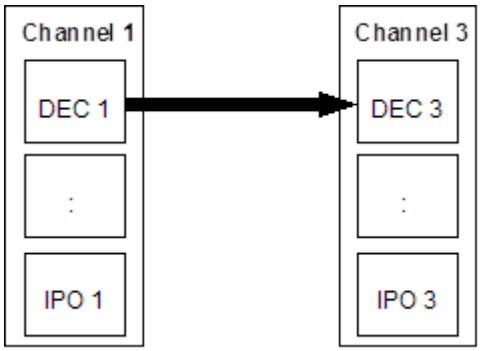
12.19.4 RESET処理

個々のチャンネルをリセットすると、該当チャンネルの同期イベントが削除されます。すなわち、該当のチャンネルによって送信されたすべての待機要求(#WAIT)と、該当のチャンネル向けのすべての非ブロードキャスト信号(#SIGNAL)が削除されます。

ブロードキャスト信号は、チャンネルによってまだ待機している場合があるため、チャンネルをリセットしても削除されません。これらのブロードキャスト信号は明示的に削除する必要があります(#SIGNAL REMOVE)。

12.19.5 同期シナリオ

2つのチャンネル上の2つのデコーダの同期



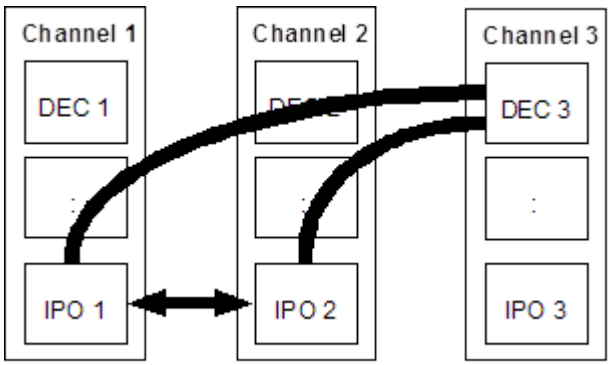
- デコーダ3はデコーダ1を待機し、デコーダ1は中断なく動作し続けます。

プログラミング例

```
% channel_1
...
(Signal P100)
(Synchronization
on DEC-level)
(Synchronization
on channel 3)
(Parameter
V.P.SYNC)

V.P.SYNC = 1000
P100 = 814

#SIGNAL [IDP100
P[0]= V.P.SYNC
CH3]
...
```

```
% channel_3
...
(Wait request 814)
(Synchronization
on DEC-level)
(Synchronization
with channel 1)
(Parameter
V.P.SIGNAL)

#WAIT [ID814 P[0]=
V.P.SIGNAL CH1]
...
```

3つのチャンネル上のデコーダおよびインターポレータ間の同期



- インターポレータ1は、インターポレータ2およびデコーダ3を待機します。
- インターポレータ2は、インターポレータ1およびデコーダ3を待機します。
- デコーダ3は、インターポレータ1およびインターポレータ2に通知します。

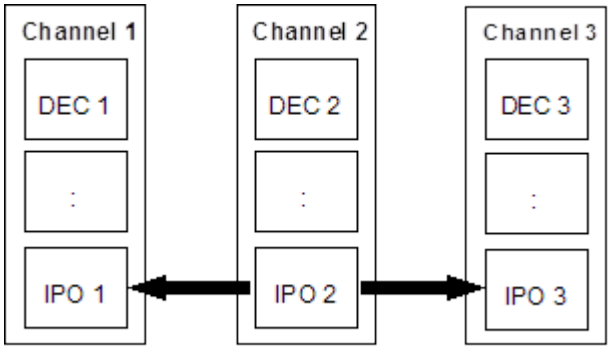
プログラミング例

```
% channel_1
...
(Wait
request 968)
(Sync. on
IPO-level)
(Sync. with
channel 2+3)

#WAIT SYN
[ID968 CH2
CH3]
```

```
%channel_2
...
(Wait
request 968)
(Sync. on
IPO-level)
(Sync. with
channel 3+1)

#WAIT SYN
[ID968 CH3
CH1]
```

```
%channel_3
...
(Signal 968)
(Sync. on
DEC-level)
(Sync. with
channel 1+2)

#SIGNAL
[ID968 CH1
CH2]
```

3つのチャンネル上のインターポレータ間の同期



- インターポレータ1は、インターポレータ2を待機します。
- インターポレータ3は、インターポレータ2を待機します。
- インターポレータ2は、インターポレータ1およびインターポレータ3に通知します。

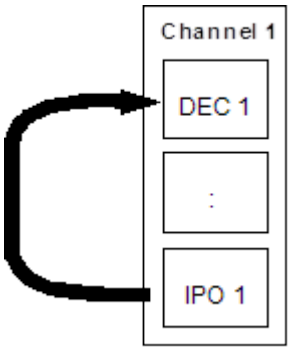
プログラミング例

```
% kanal_1
...
(Wait
request 100)
(Sync. On
IPO-level)
(Sync. With
channel 2)

#WAIT SYN
[ID100 CH2]
```

```
%kanal_2
...
(Signal 100)
(Sync. on
IPO-level)
(Sync. with
channel 1+3)

#SIGNAL SYN
[ID100 CH1
CH3]
```

```
% kanal_3
...
(Wait
request 100)
(Sync. on
IPO-level)
(Sync. with
channel 2)

#WAIT SYN
[ID100 CH2]
```

1つのチャンネルのデコーダとインターポレータ間の同期



- デコーダは、インターポレータが位置X250に達するまで待機します。
- 動作ブロック「G01 X370 Z200 F80」がNCチャンネルに既に存在し、通知後に処理されます。
- 動作ブロック「G01 X900」は、同期化後にのみ解読されます。

注記

デコーダとインターポレータ間の同期要求の場合、確認が未着であるため、NCプログラムをそれ以降に解読できない状態が発生する場合があります。ただし、NCチャンネルのバッファ効果によって、信号ブロックがインターポレータに達することがないため、その確認はインターポレータによってディスパッチされません。このような場合は、可能なデッドロックを避けるために、NCチャンネルをフラッシュする#FLUSHを指定する必要があります。

プログラミング例

```
% channel_1

G00 X100 Y500
G01 X250 F300

(Signal 88)
(Synchronization
on IPO-level)
(Synchronization
with channel 1)

#SIGNAL SYN [ID88
CH1]

(Operation)

G01 X370 Z200 F80

(Wait request 88)
(Synchronization
on DEC-level)
(Synchronization
with channel 1)
#FLUSH
#WAIT [ID88 CH1]

G01 X900
...
```

12.20 平面での回転(形状回転)

形状回転は、他のすべての形状に影響する機能の**前に**プログラムされた軸座標(形状)のみの主面で動作します。すなわち、すべてのオフセットとミラー画像処理動作は回転による影響を受けず、以前と同様に使用できます(*)。

回転は、既に回転された座標系(CS、ACS)内でも適用できます。

G17/18/19による平面の変更は、自動的に有効な形状回転を選択解除し、警告が出力されます。

#ROTATION ON [[[**ANGLE***<expr>*] [**CENTER1***<expr>*] [**CENTER2***<expr>*]]]	(モーダル)
#ROTATION OFF	(モーダル)

ANGLE*<expr>* 回転角度(°)。

CENTER1*<expr>* 回転の中心に対する最初の主軸のオフセット(実数型)

CENTER2*<expr>* 回転の中心に対する2番目の主軸のオフセット(実数型)

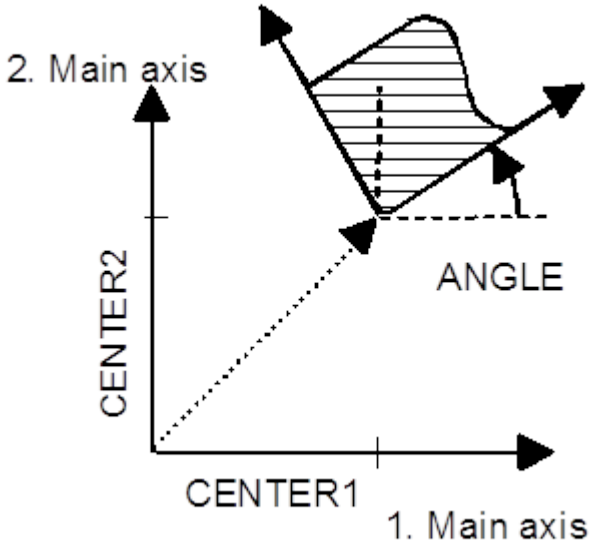


図 80: 図11-15: 主面での回転パラメータの意味

プログラムされた回転パラメータは、以下の変数から読み取ることができます。

V.G.ROT_ACTIVE 回転が有効な場合、値1を持ちます

V.G.ROT_ANGLE 回転角度

V.G.ROT_CENTER1 回転の中心に対する最初の主軸のオフセット

V.G.ROT_CENTER2 回転の中心に対する2番目の主軸のオフセット。

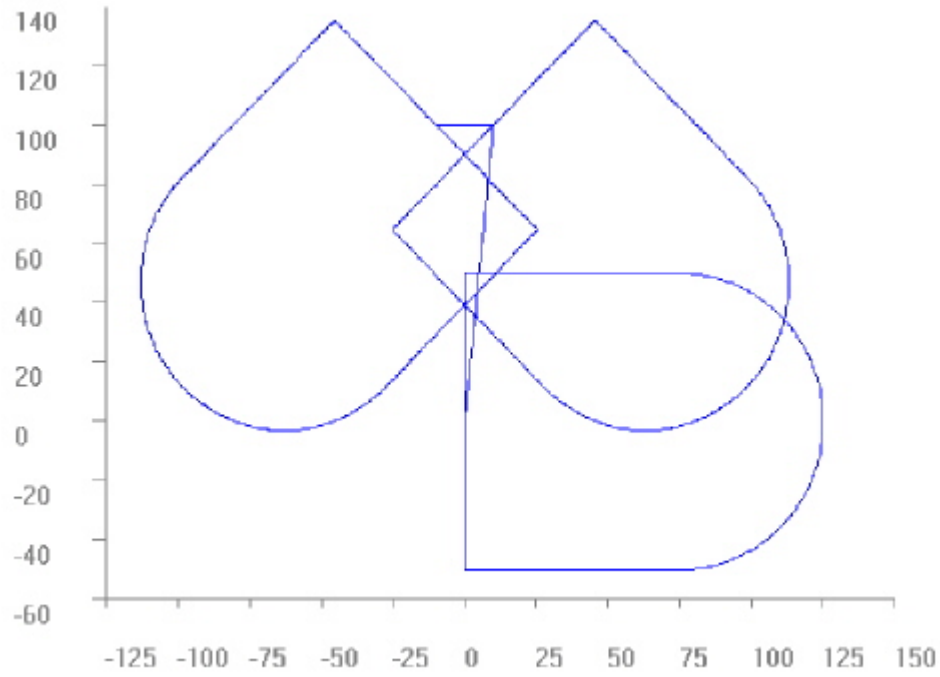
(*) シフトはROTATIONコマンドの前後にプログラムするかどうかによる違いはありません。シフトは常に機械の基本座標システム(MCS)の軸方向に向かって動作します。

また、工具オフセットは常にP-TOOL-00010からMCSの軸方向に向かって独立して動作します。

プログラミング例

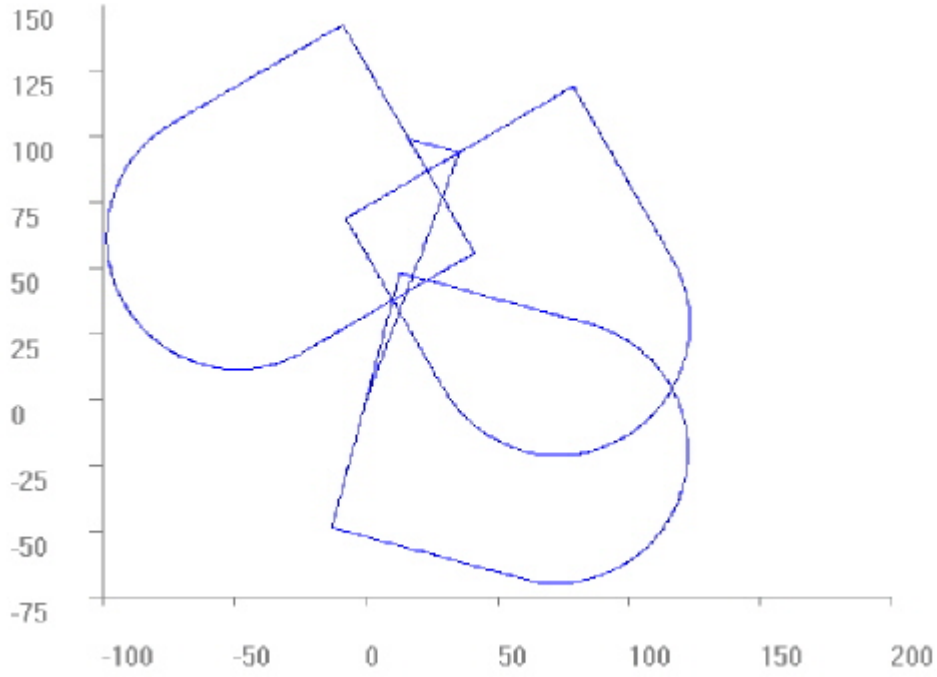
```
%L part
N10 G0 G90 X0 Y0
N30 G1 F5000 Y50
N40 X75
N50 G2 Y-50 R50
N60 G1 X0
N70 Y0
N80 M29

%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)

M30
```

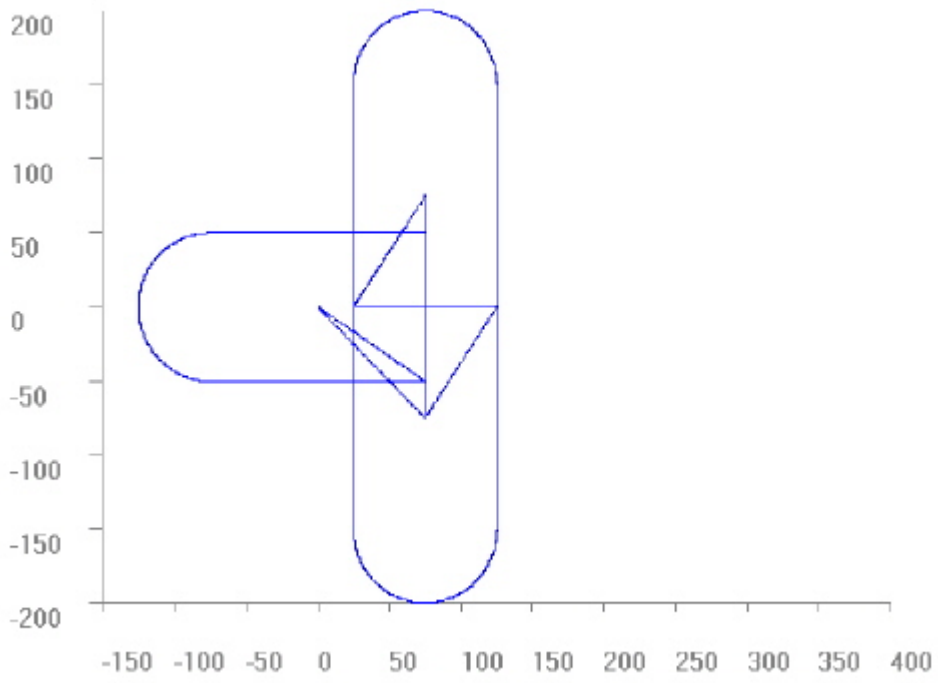


上記のパートプログラムと同一の形状、ただし、#CS–15°以内。

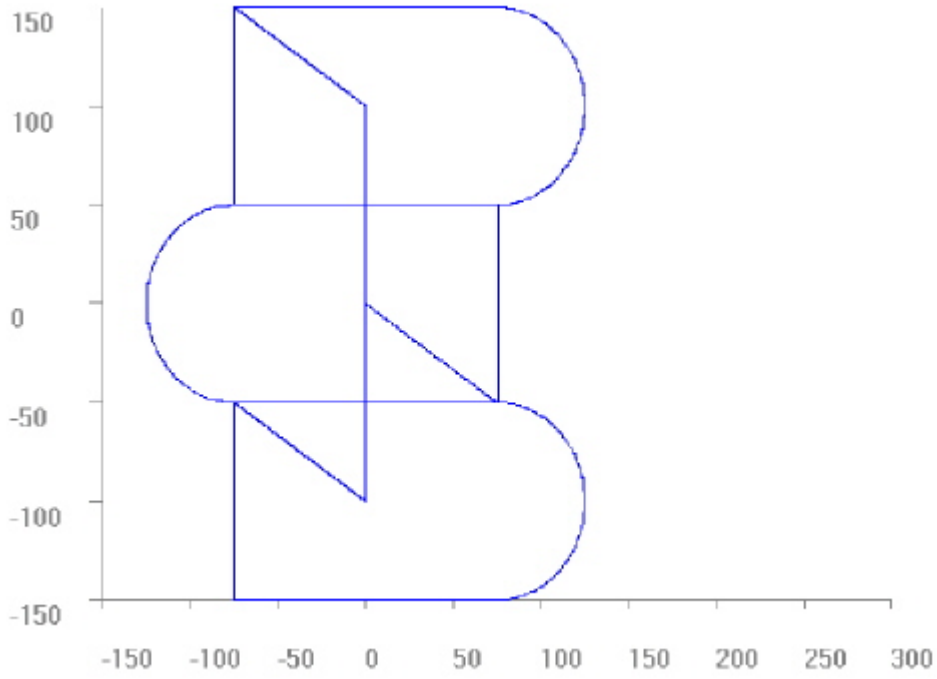
```
%L part
N10 G0 G90 X0 Y0
N30 G1 F5000 Y50
N40 X75
N50 G2 Y-50 R50
N60 G1 X0
N70 Y0
N80 M29

% 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
```

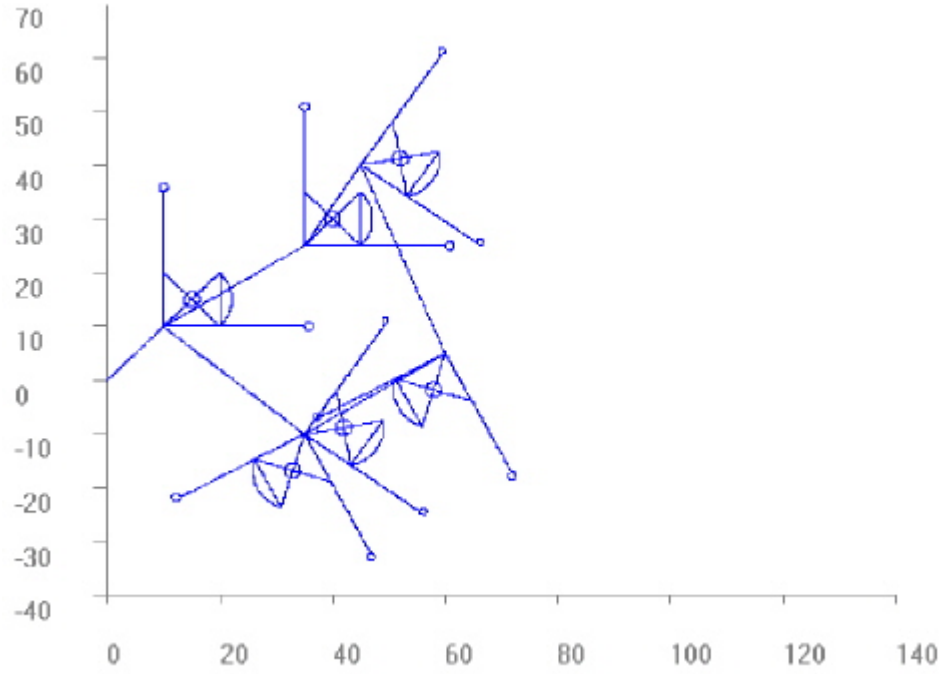



```
%L Trajectory0
N10 G54 G90 X0 Y0
N20 G1 X75 Y-50
N30 Y50
N40 X-75
N50 G3 X-75 Y-50 R50
N60 G1 X75
N70 X0 Y0
N80 M29

%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
```

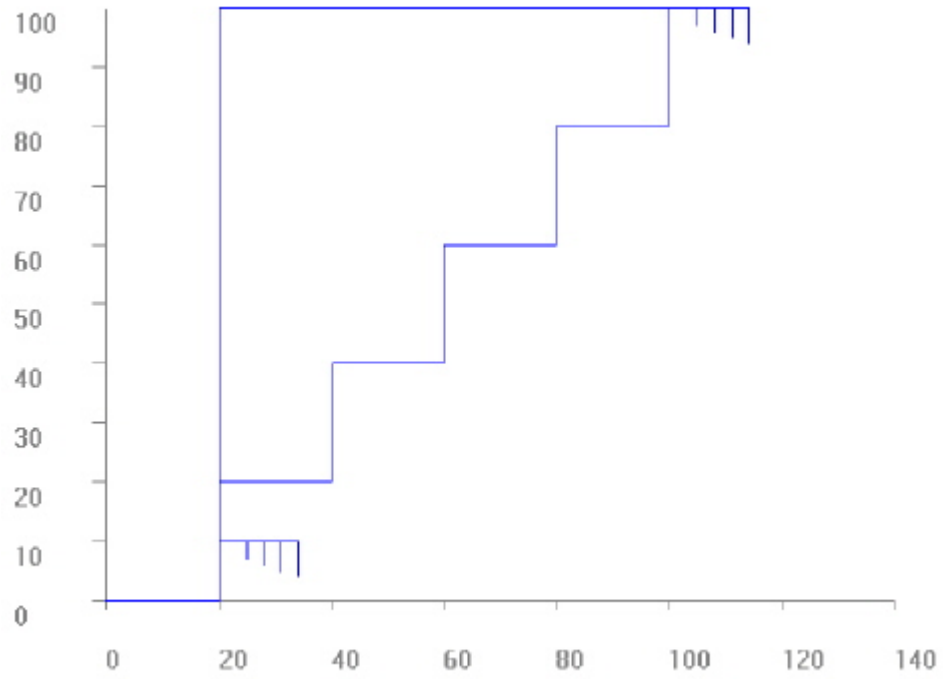


相対および絶対プログラミングのテスト:

```
%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
```

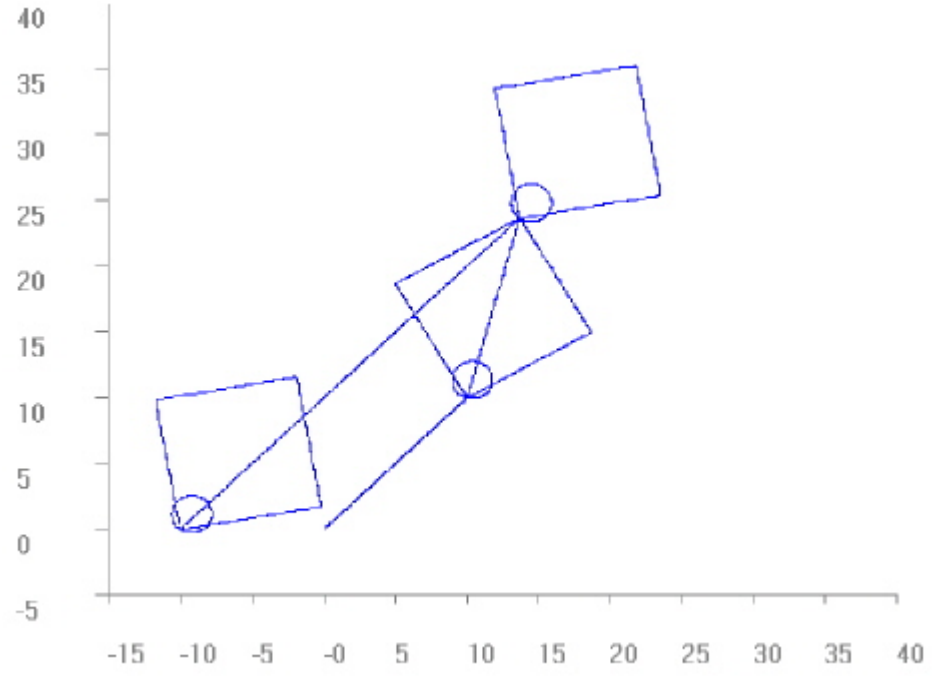
```
N5007        X0

N511 #ROTATION ON
N512 LL contour_4
N513 #ROTATION OFF

N5005 G0 G90    Y0
N5006        X0

N210 M2
```



有効な回転角度の選択後、 最初の絶対プログラミング(G90)では、最初に回転ポイントのオフセットを検討できます。

```
%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
```

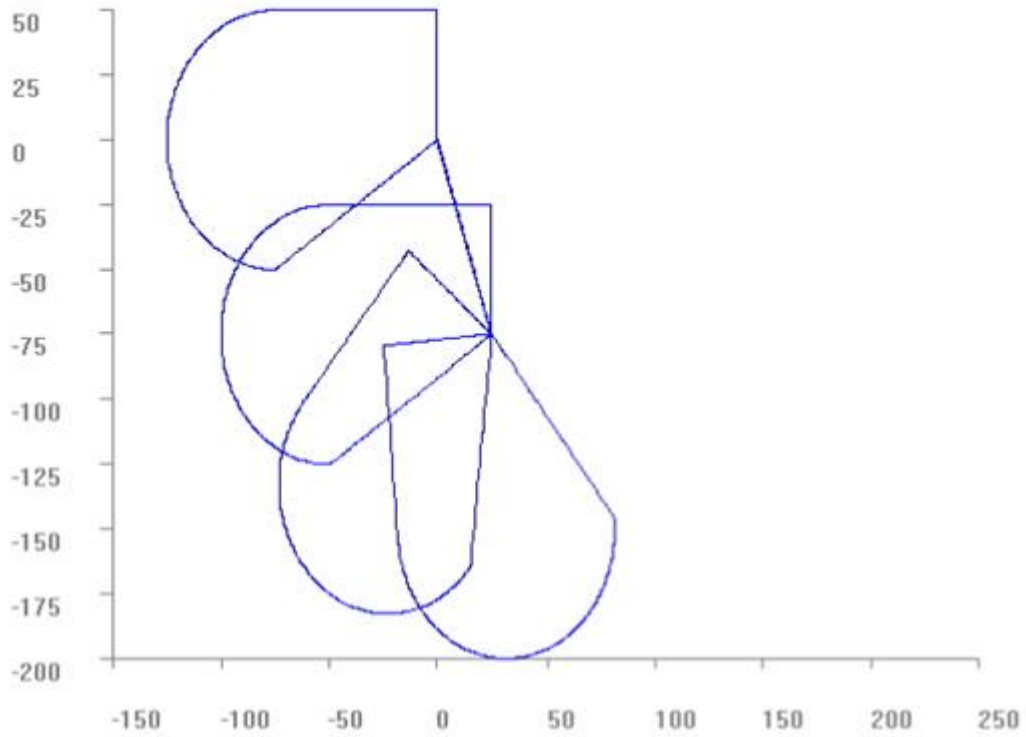
```
N250 Y0

M30
```



絶対プログラムまたは相対プログラムにされた円中心点の変換:

```
%ang_cent.nc
N10    F2000 G53

N11   G0 X0 Y0 G90
(-------------------------------------------------------------------)
( 4 times the 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 50ø (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)
N650  G0 X0 Y0 G90

(-------------------------------------------------------------------)
( The same rotated by 95ø                                           )
(-------------------------------------------------------------------)

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
```



12.21 自動軸トラッキング(C軸トラッキング)

さまざまな素材を切削する加工動作では、通常C軸に関連して使用される工具は、進むパスに対して工具を常に接線方向に保つことを可能にするガイダンスを必要とします。

その接線はパスの各点(ブレークポイント)で固有でないことに注意する必要があります。その結果、ソリューションでは、一定の接線を特徴としないブロック遷移処理の戦略が必要です。

1つの典型的な用途は、ガラス切削の技術分野です。これには、CNC工作機械によるカーバイドメタル切削ホイールの形での切削工具による加工レベル形状が含まれます。レベル加工物には、プログラムされた形状(閉じた形状、例えば楕円)に従って、加工点に少し切り込みが入れられます。この後、ガラス加工物から必要な形状を取り外すことができます。

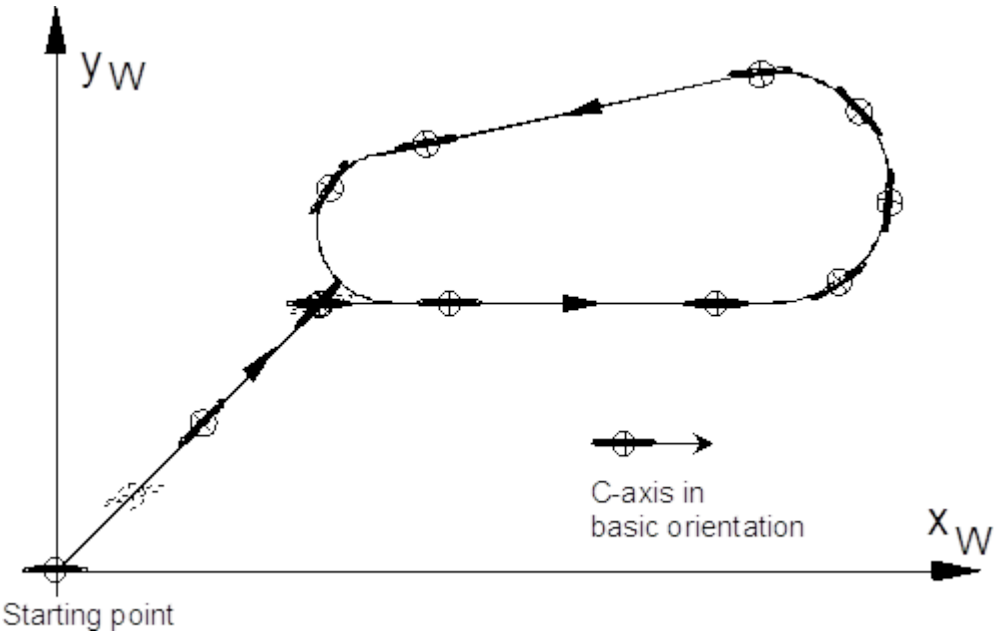


図 81: 図11-16: x-y形状に対して接線方向に向かう回転C軸のトラッキング

明示的なプログラミングによって、パスに対してC軸を接線方向にガイドすることもできます。ただし、下記のNCコマンドはプログラミングを大幅に単純化します。

#CAXTRACK ON [[[ANGLIMIT <*expr*>] [OFFSET <*expr*>] [OPTALIGN <*expr*>]	
[ROTMODE <*expr*>] [SCALEFACT <*expr*>]	
[AX <*axis_name*> / AXNR <*expr*>]]]	(モーダル)
#CAXTRACK OFF [[ANGPOS <*expr*>]]	(モーダル)

ANGLIMIT<*expr*> 制限角度(°)。

このパラメータは、非接線連続形状部分の場合のみ考慮されます。例えば、接線連続形状部分は、輪郭加工ファンクションG61によって作成されます。

ブロック遷移で形状に対する接線間の角度が制限角度を超えると、パス動作が停止し、挿入された動作ブロックによってドレッシング動作が早送り速度で実行されます。この場合、挿入された動作は、次の2番目のブロックとともに1つのユニットを形成します。これは、特にその動作(Mファンクションなど)と組み合わせたPLC同期イベントが、このモーションユニットの前後でのみ可能であることを意味します。

2番目のブロックへの遷移角度が制限角度よりも小さい場合は、2番目のブロックへの遷移時に直ちにドレッシング動作が開始されます。制限された軸加速のために、通常、2番目のブロックへのパス遷移速度は、動的なモニタリングによって減らされます。この動作を承認できない場合は、従軸の動的なモニタリングをNCコマンドによって解除でき、この場合、パス動作は影響を受けなくなります。(例: G116 C1、4.31章を参照)。

OFFSET<expr> 角度オフセット(°)。

これは、工具を形状に対して接線方向とは逆方向に向ける場合に、角度オフセットを指定するために使用します。

OPTALIGN<*expr*> 選択時、配列距離が定義された角度値よりも大きい場合、その配列が自動的に最適化されます。

このパラメータは、P-CHAN-00101、および移動領域が制限された回転直線軸(モジュロ軸なし)に従った自動配列が有効な場合にのみ考慮されます。それは、最初の形状エレメントに対する自動配列プロセス中にのみ有効です。

自動配列付きの接線追従ファンクションを選択した後、従軸の位置はオフセットなしで-180～+180°に存在します。このパラメータは、接線追従ファンクションの選択前の従軸の位置が自動配列プロセス中に考慮されることを可能にします。

このファンクションは、自動配置の選択前に従軸が最初の形状エレメントに対して既にほぼ正しい位置を持っているにも関わらず、例えば、±360°の前方向を持つ場合にのみ有効です。内部計算された配列角度値がプログラムされた角度値を超える場合、配列角度の代替ソリューションが考慮されます。この後、ソリューションの最小の配列距離によって配列角度が決まります。(*)

注記 (*) モジュロ軸の場合、自動配列プロセスは常に最短の方法で実行されます。

ROTMODE<*expr*> 従軸の割り当てを示すブール値:

0: 従軸が工具の軸である(デフォルト)。

1: 従軸が加工物の軸である。

工具の軸は、常にXY平面に対して垂直に配置する必要があります。上記の代わりに、チャンネルパラメータ(P-CHAN-00185)で従軸の割り当てを定義することもできます。

SCALEFACT<expr>	トラッキング角度のスケーリング係数(0.0 <範囲≤ 1.0)。 許容制限外の値の場合、スケーリング係数は1.0に設定されます(デフォルト)。特別な用途の場合のみ。
ANGPOS*<expr>*	選択解除で位置決め(単位(°))を行います。 選択解除中に、従軸の追加位置決めを行うことができます。回転軸の位置決めは、最短の方法で実行されます。
AX*<axis_name>*	名前による従軸の指定。この軸は、プログラムエンド(M30)まで有効です。 プログラム開始後、または従軸がプログラムされない場合、チャンネルパラメータP-CHAN-00095で定義されたデフォルト軸が有効です。
AXNR*<expr>*	論理軸番号(正の整数)による従軸の指定。この軸は、プログラムエンド(M30)まで有効です。 プログラム開始後、または従軸がプログラムされない場合、チャンネルパラメータP-CHAN-00095で定義されたデフォルト軸が有効です。

従軸は、チャンネルパラメータP-CHAN-00095で定義されます。軸の自動トラッキングは、結果としての形状遷移角度に従った最後の位置に対する正しい符号で実行されます。

パラメータ設定P-CHAN-00101に応じて、 必要な方向(通常、形状と平行な方向)への**従軸の整列**は以下のように行われます。

- 自動トラッキングを選択する前にプログラムされた整列。その位置のチェックは行われません。現在の角度位置が凍結され、従軸はその角度で配置されます。
- 自動トラッキングの選択時は、最初にプログラムされた形状エレメントに対する従軸の自動接線配置(P-CHAN-00101)。

⇨ **注意:** 多項式(G261など)によってプログラムされた形状部分が存在する場合、自動接線配置の有効化が必ず必要です。

トラッキングは、#CAXTRACK ONによる有効化後に、2番目の動作ブロックへの最初の動作ブロックの送信で開始されます。自動軸トラッキングは、円弧補間の主面(最初の主軸 + 2番目の主軸)で動作します。これは、有効化の前に定義する必要があります(G17 / 18 / 19, #PUT AX / #CALL AX / #SET AX)。

パラメータ[1]–40が有効で選択時に従軸が既に正しい方向を持つ場合、パス動作は最初の関連する動作ブロックによって中断なく継続します。

輪郭加工ファンクション(G261)が既に有効でパラメータANGLIMIT > 0の場合、円滑な動作遷移のために次の条件が必要です。

- #CAXTRACK ON [..]の前後の形状エレメントが 相互に接線ブロックであることが必要です。

同期動作では、従軸は主軸としても従軸としても動作できません!

プログラミング例

```
N10 G00 G90 X0 Y0 Z0 C0

N20 X5 Y5 C45                    ;Straight line 45° to the X axis, C axis   ;aligned parallel to
the contour

N20 #CAXTRACK ON [ANGLIMIT 3, OFFSET 0]    ;Activation of axis tracking,    ;Limit angular 3°,)
                                           ;Angular offset 0°)
N30 X10 Y10                      ;Primary movement block, C axis is
                                 ;already aligned
N40 X20                          ;Angular to the previous block: -45° >     ;Limit angular -> Block
is inserted:                ;End position of C = 0
N50 M99 X30                      ;If M-function synchronization before      ;Movement -> First
synch., then movem. C            ;on 0, then X on 30.
                                 ;If Sync. after block-> Movem. C on 0      ;then X on 30, then
Sync.
N60 X40                          ;C axis angle 0°
N70 X30                          ;C axis angle 180°
N80 Y0                           ;C axis angle -90°
N90 #CAXTRACK OFF                ;Deactivation of axis tracking

M30
```

12.22 ユーザ定義エラーの出力

NCコマンド#ERRORは、ユーザ定義エラーメッセージを出力でき、出力されたメッセージに対しては、より高度なレベルのオペレーティングシステム(GUI)によってその他の処理が加えられます。追加パラメータは、エラーをより正確に指定するオプションを提供します。

#ERROR [[[ID<*expr*>] [RC<e*xpr*>] [MID<*expr*>] {PV<*i*><*expr*>} {PM<*i*><*expr*>} {PIV<*i*><*expr*>}]] (ノンモーダル)

ID<*expr*> エラー番号。

有効値	優先度
1 ~ 1000	この数値は、出力されるお客様固有のエラー番号を決めます。

RC<*expr*> エラー(応答)クラス:

有効な値	優先度
0	警告: エラー状態への遷移なし。プログラム実行が続行される。NCカーネルのエラークラスERR_KLASSE_1に対応する。
2	重大なエラー: エラー状態への遷移あり。NC–RESETによってのみクリア可能。NCカーネルのエラークラスERR_KLASSE_3に対応する。
7	致命的なユーザエラー: エラー状態への遷移あり。NC–RESETによってクリア可能。NCカーネルのエラークラスERR_KLASSE_8に対応する。

MID*<expr>* マルチプルID。カウンタは、同一エラー番号(ID)の#ERRORコマンドがNCプログラムで複数回使用される場合、区別機能として使用されます。正の整数。

PV*<i><expr>* 実数型の最大5 (1 <= i <= 5)のお客様固有の数値(PV1 ~ PV5)をエラーメッセージで出力することもできます。

PM*<i><expr>* 最大5 (1 <= i <= 5)のPMパラメータ(PM1 ~ PM5)は、PVパラメータの意味をより正確に指定するために使用されます。

有効な値	優先度
0	IGNORE、値は意味を持たない
1	リミット値
2	現在値
3	エラー値
4	期待値
5	補正値
6	論理軸番号
7	ドライブタイプ
8	論理制御エレメント
9	状態(状況)
10	遷移
11	送信元
12	クラス
13	インスタンス
14	識別番号
15	ステータス
16	リング番号
17	ブロック番号
18	下限値
19	上限値
20	初期値
21	最終値

PIV<i><expr> 最大4 (1 <= i <= 4)のPIVパラメータ(PIV1 ~ PIV4)は、追加情報を実数型で伝送するために使用されます。

パラメータがプログラムされていない場合、以下のデフォルト値が有効です。

ID	1
RC	0
MID	0
PV1 ~ PV5	0.0
PM1 ~ PM5	1
PIV1 ~ PIV4	0.0

プログラミング例

```
:
#ERROR   Customer specific default(-standard-)error message (Warning).
:
#ERROR [ID455 RC2 PV1=5 PV2=4.999 PM1=2 PM2=3]   Fatal error 455 with
                                                  additional parameters
:
#ERROR [ID455 RC2 MID2 PV1 5 PV2 4.999 PM1 2 PM2 3]   Fatal error 455
                                                  (Multiple-identification 2)
                                                  (with additional parameters)
:
#ERROR [ID100 RC0 MID10]      Warning 100 (Multiple identification 10)
:
#ERROR [ID999 RC7]            System error 999
:
```

12.23 時間計測

NCコマンド#TIMERは、NCプログラムでの時間計測オプションを提供します。記録された時間は、ミリ秒(ms)単位で示されます。

注記

タイマカウンタは、**チャンネルでグローバル**に使用できます。これによって、チャンネル間の信号伝搬時間の計測などが可能になります。

異なるチャンネルで独立した時間計測が並列に実行されている場合、異なるIDのカウンタ番号が使用されていることを確認してください。同じIDのカウンタ番号だと、計測値が互いに影響します!

#TIMER *<action>* [*<mode>*] **[ID***<counter_nr>***]** (ノンモーダル)

<action> 指定されたカウンタ(ID)に対するアクションが決まります。

アクション	優先度
START	指定されたカウンタ(ID)の開始。
STOP	指定されたカウンタ(ID)の停止。
READ	指定されたカウンタ(ID)の読み出し。 カウントはラッチされ、割り当てられた**V.G.TIMER[ID]**変数にミリ秒(ms)単位で保存されます。
CLEAR	指定されたカウンタ(ID)のリセット。 割り当てられたV.G.TIMER変数は**削除されず**、対応するカウンタの別のREADアクションが実行されるまで保持されます。

注記

このタイマファンクションを使用して、最大1193時間を記録できます。

<mode> 同期モード:

有効なモード	優先度
---	解読レベルでのインターポレータと非同期の時間計測(基本設定)。解読後、直ちに時間計測が開始されます。
SYN	インターポレータレベルでの時間計測。指定されたカウンタは、NC機械の加工動作と同期して設定されます。インターポレータの同期読み出し(<SYN>)の場合、解読レベルでのカウントがタイマ変数に受け取られるまで、解読処理は中断されます。

プログラム動作の実行の計測の場合は、常にキーワードSYNを備えたタイマを使用することをお奨めします。

ID*<counter_nr>* カウンタ番号:

有効な値	優先度
0 ~ 127	最大128のチャンネルグローバルカウンタをプログラムできます。ただし、タイマコマンドごとには1つのカウンタ(ID)のみをプログラムできます。

プログラミング例

```
:
#FILE NAME[ MSG="C:\timer.txt" ]       File name for time recording
:
#TIMER START [ID=10]          Timer 10 is started (Decoding level)
#TIMER START SYN [ID11]       Timer 11 is started (IPO-level)
:
:
#TIMER READ [ID10]            Timer value is stored in V.G.TIMER[10]
#TIMER READ SYN [ID11]        Timer value is stored in V.G.TIMER[11]
#MSG SAVE["T10 = %d",V.G.TIMER[10]]    Recording of timer value in file
#MSG SAVE["T11 = %d",V.G.TIMER[11]]    Recording of timer value in file
#TIMER STOP [ID10]            Timer 10 is stopped
#TIMER CLEAR [ID10]           Timer 10 is reseted
:
:
#TIMER READ SYN [ID11]        Timer value is stored in V.G.TIMER[11]
#MSG SAVE["T11 = %d",V.G.TIMER[11]]    Recording of timer value in file
:
:
#TIMER READ SYN [ID11]        Timer value is stored in V.G.TIMER[11]
#MSG SAVE["T11 = %d",V.G.TIMER[11]]    Recording of timer value in file
:
:
#TIMER READ SYN [ID11]        Timer value is stored in V.G.TIMER[11]
#MSG SAVE["T11 = %d",V.G.TIMER[11]]    Recording of timer value in file
#TIMER STOP SYN [ID11]        Timer 11 is stopped
#TIMER CLEAR SYN [ID11]       Timer 11 is reseted
:
:
:
#TIMER START [ID=10, ID11]    Error, only one counter per timer command
                              permissible!
:
:
```

12.24 送り軸の定義

コマンド#FGROUPでプログラムされた軸の場合、プログラムされた送りで動作するスペース内のパスが決定されます。他のすべての軸は、従軸と同様に取り扱われます。それらは、補間軸と同様に、同時にターゲット位置に到達します。

補間軸の特徴的な機能は、送りに関して移動する距離が考慮されるということです。対照的に、従軸の移動する距離は、パス速度に対して直接の影響がありません。

送り軸がプログラムされないと、チャンネルパラメータP-CHAN-00096およびP-CHAN-00011のデフォルト設定が有効になります。

#FGROUP [[*<axis_name>* {,*<axis_name>*}]] (モーダル)

<axis_name> 送りグループのメンバである軸の名前

直線補間の場合、任意の軸を送り軸とすることができます。

円弧補間および多項式補間の場合、以下の制限事項が有効です。

- 円弧補間の場合、すべての主軸が送り軸であるか、またはすべての定義された送り軸が従軸であることが必要です。
- #FGROUPコマンドとは独立した多項式補間の場合、すべての主軸が送りグループを形成します。上記の例外は、DIST_SOFTモードでの多項式輪郭加工です。この場合、プログラムされた#FGROUPが有効です。

プログラミング例

```
N10  #FGROUP [X, Y]          (X and Y are feed axes)
:
N50  #FGROUP [A]             (Tracking axis A is feed axis)
:
N100 #FGROUP                 (Feed axes according to the default settings)
                             (in the channel parameters)
:
N999 M30
```

回転加工物軸上の円筒形加工物の処理中に、実際のプログラムされた送り[mm/min]が工具の接点で有効です。これを確保する1つの方法は、有効なキネマティックトランスフォーメーション(外側面トランスフォーメーションなど)を選択することです。その他の方法としては、NCコマンド#FGROUP ROT[...]の使用があります。このコマンドをプログラムすると、その後、回転軸の送り(Grad/min)が基準半径に応じて変換されます。回転軸のみまたは回転軸と直線軸をプログラムすると、基準半径で要求されてプログラムされた送りが発生します。

#FGROUP ROT [**AX** *<axis_name>* **REF** *<reference_radius>*] (モーダル)

AX *<axis_name>* 基準半径が有効である軸の名前。

REF *<reference_radius>* 回転軸の有効な半径。

回転軸に関する送り計算の選択解除は以下によって行われます。

#FGROUP ROT	(モーダル)

注記

軸「AX..」が実際に回転軸であるかどうかのチェックは行われません。

ファンクションはフィードブロック(G01)にのみ使用可能で、G94と組み合わせることができます。

通常、このファンクションはフライス削りのために使用されます!

旋盤の送り調整はG95とG96によってプログラムされます。

プログラミング例

基準半径R = 10mmの加工物

```
N05 G00 C0
N10 G01 C180 F1000 (Rotational speed of the workpiece 1000 degree/min)
                   (Feed at the workpiece periphery 174.67 mm/min)

N20 #FGROUP ROT[AX=C REF=10]
N30 G01 C360 F1000 (Feed at the workpiece periphery 1000 mm/min)
                   (Rotational speed of the workpiece 5727.6 degree/min)
:
Nxx #FGROUP ROT (Deselection)
```

```
:
N10 G00 X0 Y0 Z0
N15 #FGROUP ROT[AX=C REF=10]  (Feed on millers intervention area 1000 mm/min)

N60 G01 G91 X10 C57.325 F1000 (Diagonal on lateral surface)
N70 G90 X0 C0
Nxx #FGROUP ROT (Deselection)
:
```

コマンド#FGROUP WAXISは、チャンネルパラメータのデフォルト設定は別にして、実行時間が最長の軸(「最弱軸」) が、プログラムされた送り(F-Word)を備えた送り軸として自動的に動作することを定義します。他のすべての軸は、従軸として取り扱われます。

#FGROUP WAXIS	(モーダル)

プログラミング例

```
N10  #FGROUP [X, Y]              (X and Y are feed axes)
:
N50  #FGROUP WAXIS               (Weakest axis is feed axis)
:
N100 #FGROUP [X, Y, Z]           (X, Y and Z are feed axes)
:
N999 M30
```

12.25 動的なパスリミットの調整

バージョンV2.10.1507.02から、コマンド#VECTOR LIMIT ON/OFF...が、コマンド#VECTORACC ON/OFF...および#VECTORVEL ON/OFF...の代わりに使用されます。互換性の理由からこれらのコマンドはそれ以降も使用できますが、新しいNCプログラムでは使用してはいけません。

パス上の最大許容速度、加速度、および減速度は、軸特有のパラメータリストおよびプログラムされた形状で設定された動的な特性に応じて異なります。

いくつかの特定の用途(高強度レーザーまたはプラズマトーチによる切削プロセスなど)のために最適な結果を確保するために、NCプログラムで直接にパス上の動的な特性を変更/調整することができます。

これらのパスリミット値は、処理の動的な動作フェーズ中に、以下のNCコマンドによって影響を受ける場合があります。それらによって、ユーザ定義リミットの各デフォルトリミットの有効化/無効化が可能になります。

#VECTOR LIMIT ON [[ACC*<acc>*] [DEC***<dec>***] [VEL***<vel>***]]**	(モーダル)
または	
#VECTOR LIMIT ON [[ACC] [DEC] [VEL]]	(モーダル)
または	
#VECTOR LIMIT ON ALL	(モーダル)

ACC*<acc>* 加速度リミット(mm/min²)。

DEC*<dec>* 減速度リミット(mm/min²)。

選択した傾斜プロファイルが加速度および減速度パラメータ(傾斜タイプTRAPEZ [▶ 261]など)の別の定義を許可する場合は、DECが有効です。

VEL*<vel>* 速度リミット(mm/min)。

キーワードACC、DEC、VELの後にリミット値がプログラムされないか、またはコマンド#VECTOR LIMIT ON ALLが使用される場合は、チャンネルパラメータリストP-CHAN-00002、P-CHAN-00090、およびP-CHAN-00208の動的なデフォルト値が使用されます。

コマンド#VECTOR LIMIT OFF ... は、Look Aheadによる動的なリミットの計算への切り替えを行います。この切り替えは、特定のリミットとすべてのリミットの両方に対してプログラムすることができます。

#VECTOR LIMIT OFF [[ACC] [DEC] [VEL]]	(モーダル)
または	
#VECTOR LIMIT OFF ALL	(モーダル)

動的なパスリミットは、Look Aheadの有効なリミットよりも小さい場合のみ使用されます。原点復帰、手動操作、独立軸のような軸特有の動作には影響を与えず、

G01とG00の両方に影響を与えます。

注記

チャンネルパラメータP-CHAN-00097に関連して、ユーザは、プログラムされたリミットに依存した送り保持中の機械の減速度も減速の度合いを弱めることを考慮する必要があります。

G00の動的なパスリミットも、チャンネルパラメータ[1]-7で定義されたパス依存の重み付けテーブルによって影響される場合があります。

プログラミング例

```
%vec_limit

(Setting of dynamic limit data on specific values)
N10 #VECTOR LIMIT ON [ACC=3600000 DEC=4000000 VEL=3000]
N11 #VECTOR LIMIT ON [ACC=3600000 DEC=4000000]
N12 #VECTOR LIMIT ON [ACC=3600000 VEL=3000]
N13 #VECTOR LIMIT ON [ACC=3600000]
N14 #VECTOR LIMIT ON [DEC=4000000 VEL=3000]
N15 #VECTOR LIMIT ON [DEC=4000000]
N16 #VECTOR LIMIT ON [VEL=3000]

(Setting of dynamic limit data on default values)
N20 #VECTOR LIMIT ON [ACC DEC VEL]
N21 #VECTOR LIMIT ON [ACC DEC]
N22 #VECTOR LIMIT ON [ACC VEL]
N23 #VECTOR LIMIT ON [ACC]
N24 #VECTOR LIMIT ON [DEC VEL]
N25 #VECTOR LIMIT ON [DEC]
N26 #VECTOR LIMIT ON [VEL]

(Mixed setting of dynamic limit data)
N27 #VECTOR LIMIT ON [ACC=3600000 DEC]
N28 #VECTOR LIMIT ON [ACC VEL=3000]

(Setting of all dynamic limit data on default values)
N30 #VECTOR LIMIT ON ALL  (:= #VECTOR LIMIT ON [ACC DEC VEL])

(Setting of dynamic limit data by LOOK_AHEAD)
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]

(Setting of all dynamic limit data by LOOK_AHEAD)
N50 #VECTOR LIMIT OFF ALL (:= #VECTOR LIMIT OFF [ACC DEC VEL])

N999 M30
```

12.26 最小ブロック遷移速度の定義

不連続ブロック遷移(形状角度)では、パス速度は、結果としての軸加速度が軸パラメータであらかじめ定義されたリミット値を超えない程度に減少します。技術的観点からすれば、形状角度でのこの速度減少は不必要な場合があります(フレーム切削またはガス切削など)。この場合、以下のコマンドを使用して、形状角度でそれ未満にはならない最小速度を定義できます。

注記

プログラムされた最小速度は、円弧および直線ブロックの不連続ブロック遷移でのみ動作します。その結果、NCプログラムでは、輪郭加工モードもスプライン軸補間も選択できません(G261、#HSC、G151など)。

#TRANSVELMIN ON [[*<vel>*]] **#TRANSVELMIN OFF**	(モーダル)

<vel> 最小ブロック遷移速度(mm/min)。

#TRANSVELMIN ON後にリミット値がプログラムされない場合、最小遷移速度は0に設定されます。コマンド#TRANSVELMIN OFFは、Lool Aheadファンクションの速度リミットの自由な計算への切り替えを行います。

これらのファンクションは直線および非直線の傾斜を対象にしますが、非直線の傾斜の処理中は、不連続ブロック遷移用のジャークリミットが有効であることはできません(チャンネルパラメータ corr_v_trans_jerk=1)。このような場合は、ジャークリミットは事前設定します。

プログラムされた最小速度は、プログラムエンドまでのみ有効です。次のプログラム開始またはRESETでは、最小速度はゼロに設定されます。

チャンネルパラメータ[CHAN]であらかじめ行われた定義は可能ではありません。

12.27 機械データの書き込み

このファンクションを使用できるかどうかは、設定やバージョンの具体的な範囲によって異なります。

このコマンドにより、パートプログラムで軸特有の機械パラメータを変更できます。新しい値は、プログラムでグローバルに有効です。それらの値は、機械データリストの次の更新(制御のスタートアップなど)によって上書きされます。

新しいパラメータの引き継ぎが終了して値が有効になるまで、最後に有効なパス補間は停止します。結局、回転スピンドルは停止**しません**。

#MACHINE DATA [<*mode*>] **[AX**<*name*> / **AXNR**<*expr*> <*param_id*><*expr*> | **AXPARAM "**<*string*>**"** [**WAIT**] **]** (モーダル)

<*モード*> 同期モード

有効なモード	優先度
---	解読レベルでの同期(基本設定)
SYN	補間(リアルタイム)レベルでの同期

AX<*name*> 新しい軸特有のパラメータ値の対応する軸またはスピンドルの名前

AXNR<expr> 新しい軸特有のパラメータ値のチャンネル軸またはスピンドルの論理軸番号。正の整数。

注記
論理軸番号に関する妥当性検査は行われません。オペレータが正しく入力を行う必要があります。

<*param_id*><*expr*> パラメータリスト[AXIS]の単位で新しい値が割り当てられたISGスペリングの軸パラメータ(P-AXIS-xxxxx)。以下の軸パラメータは、あらかじめ定義されたキーワード(param_id)を使用して変更できます。

param_id	優先度
P-AXIS-00001	非直線速度プロファイル: 速度が増加する軸加速度
P-AXIS-00002	非直線速度プロファイル: 速度が減少する軸減速度
P-AXIS-00004	高速動作での加速度(G00)
P-AXIS-00005	直線速度プロファイル: 高速モードでの加速度ステップ1 (G00)
P-AXIS-00006	直線速度プロファイル: 高速モードでの加速度ステップ2 (G00)
P-AXIS-00008	許容軸運動: 最大許容軸加速度
P-AXIS-00011	直線速度プロファイル: 送り補間での加速度ステップ1 (G01、G02、G03)
P-AXIS-00012	直線速度プロファイル: 送り補間での加速度ステップ2 (G01、G02、G03)
P-AXIS-00045	2つの衝突軸間の最小距離
P-AXIS-00056	トラッキングモードの無効化後の最大誤差
P-AXIS-00075	ガントリー動作: 数値単位系間の誤差のために主軸と従軸間の静的オフセットを補正するための補正速度
P-AXIS-00099	P位置制御用の増幅係数k_v
P-AXIS-00103	バックラッシュ補正のサイズ
P-AXIS-00109	独立した軸およびスピンドル用の最大許容速度オーバーライド
P-AXIS-00151	正確なストップのためのウィンドウに達するための最大遷移時間
P-AXIS-00152	基準点の絶対位置
P-AXIS-00166	非直線追従誤差モニタリングの残りの偏差
P-AXIS-00167	動的な追従誤差モニタリングの係数
P-AXIS-00168	最大追従誤差
P-AXIS-00169	最小追従誤差
P-AXIS-00172	追従誤差モニタリングのタイプ
P-AXIS-00195	非直線速度プロファイル: 加速度段階的減少のランプ時間
P-AXIS-00196	非直線速度プロファイル: 加速度段階的増加のランプ時間
P-AXIS-00197	非直線速度プロファイル: 減速度段階的減少のランプ時間
P-AXIS-00198	非直線速度プロファイル: 減速度段階的増加のランプ時間
P-AXIS-00200	非直線速度プロファイル: 高速動作でのランプ時間(G00)
P-AXIS-00201	軸加加速度を制限するためのドライブの最小許容ランプ時間
P-AXIS-00208	トラッキング動作の選択解除後の補正動作の最大送り
P-AXIS-00209	早送りG00
P-AXIS-00211	直線速度プロファイル: 高速モードでのランプ1とランプ2間の切り替え速度(G00)
P-AXIS-00212	許容軸運動: 最大許容軸速度
P-AXIS-00216	スピンドルの最小許容軸速度。この速度未満の場合、位置コントローラの回転速度モニタリングは状態「速度ゼロ」を示します。
P-AXIS-00217	状態「速度値獲得」の計算および決定のための係数
P-AXIS-00218	最小原点復帰速度
P-AXIS-00219	最大原点復帰速度

param_id	優先度
P-AXIS-00221	直線速度プロファイル: 送り補間での速度ランプ1と速度ランプ2間の切り替え速度(G01、G02、G03)
P-AXIS-00236	正確なストップのための制御ウィンドウのサイズ
P-AXIS-00414	距離制御用の最大位置オフセット

AXPARAM "<*string*>" 代替構文: 完全な構造パスと、パラメータリスト[AXIS]の内部スペリングでの値を備えた軸パラメータ(例を参照)。**すべての**軸パラメータを変更できます

WAIT このキーワードの使用は、軸パラメータの同期される設定(SYN)に関してのみ許可されます。WAITがプログラムされると、さらに、プログラムの解読処理が中断され(暗黙的なFLUSH)、新しいパラメータの引き継ぎが終了し、その値がNCチャンネル全体で有効になるまで、プログラムの解読処理を待機します。

WAITは、例えば、有効な工具径補正、多項式輪郭加工、またはHSCモードと同様に、形状変更ファンクションの実行中は使用できません。

プログラミング例

```
N10  G00 X100              (Positioning with rapid traverse velocity acc.)
                           (to default settings after startup)

N20  #MACHINE DATA SYN [AX=X P-AXIS-00209=80000]     (New rapid velocity)
:
N30 ...    (In following program sequence the new rapid traverse)
           (velocity is valid)
:

Alternative in axis list syntax:
:
N20 #MACHINE DATA SYN [AX=X AXPARAM="getriebe[i].vb_eilgang  80000"]
                                                        (New rapid vel.)
:

For a spindle axis:

N20 #MACHINE DATA SYN [AX=S P-AXIS-00109=1200]       (New V-Override)


Setting of a new software limit and waiting in channel:
:
N20 #MACHINE DATA SYN [AX=X AXPARAM="kenngr.swe_pos 15000000" WAIT]
```

12.28 ファイル操作

12.28.1 ファイル名の定義

NCプログラムでコマンド#FILE NAMEを使用して、例えば、レポートファイルの作成またはNCプログラムの呼び出しのために特定のNCファンクションで使用するファイル名を定義することができます。

#FILE NAME [<*file_id*>"<*filename*>" { <*file_id*>"<*filename*>" }] (モーダル)

<*file_id*> ファイル識別

ファイル識別子	優先度
MSG	#MSG SAVE用のレポートファイルの名前です。 デフォルト名は「*message.txt*」です。
M6	NCプログラムのM6での暗黙的なプログラム呼び出し用グローバルサブプログラムの名前。その名前はM30まで有効です。M6は、それ以上、Mファンクションとしては使用されません! デフォルト名はP-CHAN-00118に設定されています。
G80 ~ G89	NCプログラムのG80 – G89での暗黙的なプログラム呼び出し用グローバルサブプログラムの名前。これらの名前はM30まで有効です。 デフォルト名は、P-CHAN-00160 - P-CHAN-00169に設定されています。
G800- G819	NCプログラムのG800 ~ G819での追加プログラムの暗黙的な呼び出し用グローバルサブプログラムの名前。これらの名前はM30まで有効です。 デフォルト名はP-CHAN-00187に設定されています。

オプション: <*file_id*>と<*filename*>の間に等号(=)をプログラムすることができます。いつでもNCプログラムで#FILE NAMEを使用して、ファイル名の定義または変更を行うことができます。

ファイル名は、RESETおよびNCプログラムの起動で、デフォルト名に設定されます。

プログラミング例

```
%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"]   Output in prog_flow.txt
N50 $ENDIF
:
Nxx #MSG SAVE["Roughing OK"]
:
Nxx #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               Call of tool change program tool_change.nc
:
M1000 M30

%example3
N10 #FILE NAME[ G80="g80_up_test.nc" ]
N20 G00 X100 Y100 Z50
N30 G80              Call of sub program g80_up_test.nc
:
M1000 M30
```

```
%example4
N10 #FILE NAME[ G800="g800_up_test.nc" G815="g815_up_test.nc"]
N20 G00 X100 Y100 Z50
N30 G800              Call of sub program g800_up_test.nc
:
N90 G815              Call of sub program g815_up_test.nc
:
M1000 M30
```

12.28.2 ファイル名の変更

コマンド#FILE RENAMEは、既存のファイル名を変更するために使用されます。すべてのパラメータを指定する必要があります。パラメータを省略すると、対応するエラーメッセージが表示されます。

#FILE RENAME [PATHOLD="<*ファイル名*>" PATHNEW="<*ファイル名*>" OVRMODE=<*expr*>]

PATHOLD<*filename*>	名前を変更するファイル(ディレクトリ指定付き)。このディレクトリパスまたはファイルが存在しない場合、NCプログラムはエラーメッセージ付きで中止されます。
PATHNEW<*filename*>	(新しいディレクトリの)ファイル(ディレクトリ指定付き)。ここで、デフォルトディレクトリと異なるディレクトリパスが指定され、そのディレクトリが存在すると、このディレクトリに新しい名前でファイルが移動します。このディレクトリが存在しない場合、対応するエラーメッセージが出力されます。
OVRMODE<*expr*>	PATHNEWによって指定されたファイルが既に存在する場合、上書きするかどうかを示すブール値。 0: ファイルを上書きできない。エラーメッセージの出力。 1: 既存のファイルを上書きできる。

オプション: 等号(=)をプログラムできます。

ディレクトリPATHNEWおよびPATHOLDに対して、ユーザは書き込みアクセス権を持つ必要があります。アクセス権を持たない場合、名前の変更でエラーが発生します。

注記

書き込み保護

名前を変更するファイルに対して書き込み保護されている (新しいディレクトリの)ファイルが既に存在し、名前を変更するファイルが書き込み保護されている場合に、エラーが生成されます。

注記

相対ディレクトリ

コマンド#FILE NAMEを使用して、ファイルを指定された相対ディレクトリと組み合わせて定義することもできます。この場合、この相対指定は、あらかじめ定義されたデフォルトディレクトリに対応します(TwinCATでは、このパスはシステムマネージャで設定されます)。
相対パス指定によって#FILE RENAMEコマンドでファイルの名前を変更する場合、この相対的なパス指定は.exeファイルを開始したカレントディレクトリに対応します。
→ 宛先ディレクトリが相対パス指定によって異なる可能性が非常に高くなります!
したがって、#FILE NAMEと#FILE RENAMEとの組み合わせで**絶対ディレクトリ**のみを使用することをお勧めします。

プログラミング例

```
%FileRename

N10 #FILE NAME[MSG="C:\Test.txt"]     ;File definition
...
N40 #MSG SAVE["Write me into file"]   ;Writing text into file

N60 #FILE RENAME[PATHOLD="C:\Test.txt" PATHNEW = "C:\NewName.txt" OVRMODE=1]
N70 M30
```

12.28.3 ファイルの削除

コマンド#FILE DELETEは、ファイルを削除するために使用されます。パラメータを指定する必要があります。パラメータを指定しないと、対応するエラーメッセージが生成されます。

#FILE DELETE [PATH="*<filename>*"]

PATH=*<ファイル名>*	削除するファイル(ディレクトリ指定付き)。このディレクトリパスまたはファイルが存在しない場合、NCプログラムはエラーメッセージ付きで中止されます。

オプション: 等号(=)をプログラムできます。

ディレクトリPATHに対して、ユーザは書き込みアクセス権を持つ必要があります。アクセス権を持たない場合、削除でエラーが発生します。

注記

書き込み保護

削除するファイルが書き込み保護されている場合に、エラーが生成されます。

注記

相対ディレクトリ

コマンド#FILE NAMEを使用して、ファイルを指定された相対ディレクトリと組み合わせて定義することもできます。この場合、この相対指定は、あらかじめ定義されたデフォルトディレクトリに対応します(TwinCATでは、このパスはシステムマネージャで設定されます)。
相対パスの指定によって#FILE DELETEコマンドでファイルを削除する場合、この相対的なパス指定は.exeファイルを開始したカレントディレクトリに対応します。
→ 宛先ディレクトリが相対パス指定によって異なる可能性が非常に高くなります!
したがって、#FILE NAMEと#FILE DELETEとの組み合わせで**絶対ディレクトリ**のみを使用することをお勧めします。

プログラミング例

```
%FileDelete

N10 #FILE NAME[MSG="C:\Test.txt"]     ;File definition
...
N40 #MSG SAVE["Write me into file"]   ;Writing text into file
…
N60 #FILE DELETE[PATH="C:\Test.txt" ] ;Delete file
N70 M30
```

12.28.4 ファイルの存在のチェック

コマンド#FILE EXISTは、ファイルシステムでの1つのファイルの可用性およびアクセスをチェックします。#FILE EXISTを実行した後、そのチェックの結果は変数V.G.FILE_EXISTに保存されます。これは、常に最新の呼び出しの結果値がV.G.FILE_EXISTに保管されることを意味します。この変数のデフォルト値はFALSEです。

#FILE EXIST [PATH="*<filename>*"]

PATH<*ファイル名* 可用性についてチェックするファイル(ディレクトリ指定付き)。
>

オプション: 等号(=)をプログラムできます。

注記

相対ディレクトリ

コマンド#FILE EXISTを使用して、指定された相対ディレクトリと組み合わせてファイルをチェックすることもできます。この相対指定は、制御の起動済み「exeファイル」の現在のディレクトリに対応します。

プログラミング例

```
%FileExist

N010 #FILE EXIST[PATH = "C:\TwinCat\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
```

12.29 軸設定および軸結合の復元

処理のために軸交換操作と同期軸を使用する複雑なマルチチャンネル機械は、NC Resetによるエラーの場合に中断されます。その後、NCチャンネルは再初期化され、中断時に有効であったチャンネル設定は失われます。工具軸のクリアランスを有効にするには、中断時に有効であったNCチャンネルの軸設定と軸結合を復元する必要があります。このために、以下のNCコマンドを使用できます。

12.29.1 現在の設定の保存

#SAVE CONFIG [[AX] [AXLINK]] (モーダル)

AX: NCチャンネルの現在の既存軸設定の保存。

AXLINK: NCチャンネルの現在の選択された軸結合の保存。AXLINKがプログラムされ、有効な結合が存在しない場合、保存済みの結合は削除されます。

通常、NCプログラムでは、コマンド#SAVE CONFIGは、常に軸設定の変更または軸結合の選択を発生させる操作後に使用されます。例えば、軸交換コマンド、または同期動作の選択用コマンドの後などです。

設定データの保存中は、NCチャンネルの設定データはインターポレータデータによって調整され、その後、非同期タスクの作業データに保管されます。このメカニズムは、現在の処理状態に同期した現在の設定データを一貫して保存することができます。

プログラミング例

```
%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]     Saving of axes configuration X,Y,Z,B,C
                                 and the active coupling group 1
N.. X.. Y.. Z..
:
N200 #AX LINK OFF ALL
N210 #AX RELEASE [C]
N220 #SAVE CONFIG [AX]     Saving of new axes configuration X,Y,Z,B.
                           The saved axes configuration up to now
                           from block N35 is overwritten!
:
N.. X.. Y.. Z..
N99 M30
```

保存済み設定はプログラムグローバルに保存されます。保存済み設定は、次の#SAVE CONFIGコマンドによって更新するか、または#CLEAR CONFIGによって削除することだけができます!

12.29.2 保存済み設定のロードまたは復元

#LOADCONFIG [[AX] [AXLINK]] (モーダル)

AX: 最新の保存済み軸設定のロード。使用できる保存済み設定が存在しない場合は、エラーメッセージが出力されます。NCチャンネルでは、存在していない軸がオフセットなしで要求されます。

AXLINK: 最新の有効な保存済み軸結合のロード。すべての軸結合が、**結合グループ1**で集約、復元、および有効化されます。使用できる保存済み軸結合が存在しない場合は、エラーメッセージが出力されます。

NC RESETの後のクリアランスプログラムでは、最新の保存済み設定を復元するために、コマンド#LOAD CONFIGが使用されます。ただし、#SAVE CONFIGコマンドの正しいプログラミングと必要な設定の正しい復元の責任は、機械のユーザが負います。両方のキーワードがプログラムされると、プログラミングの順序に関係なく、常に最初にNCチャンネルの軸設定が復元され(「FAST」なしで)、その後に保存済み軸結合が選択されます。

プログラミング例

処理の中断およびNC-RESET後のクリアランスプログラムの開始:

```
%Clearance

N10 G53
N35 #LOAD CONFIG [AX AXLINK]    Restoring of the saved axes
                                configuration and axis couplings
                                under coupling group 1
N40 #ECS ON           Definition of a EKS for the execution
                      of the retract travel
N.. X.. Y.. Z..
:
N200 #AX LINK OFF ALL
:
N.. X.. Y.. Z..
N99 M30
```

同一のNCプログラムで#SAVE CONFIGおよび#LOAD CONFIGを使用する場合、#LOAD CONFIGの開始時に、チャンネルの保存済み設定の一貫性を確保するために、常に暗黙的なFLUSHが実行されます。

プログラミング例

```
N..
Nxx #SAVE CONFIG [AX AXLINK]
:
N..
:
Nxx #LOAD CONFIG [AX AXLINK]    At first an implicit FLUSH, then
                                restoring of axes confguration,
                                then restoring of axis couplings
N.. X.. Y.. Z..
N99 M30
```

12.29.3 保存済み設定の削除

#CLEARCONFIG	(モーダル)

#CLEAR CONFIGを使用すると、最新の保存済み設定が完全に削除されます。その後に直接に#LOAD CONFIGをプログラムすると、エラーメッセージ「No restorable configuration found」が表示されます。これは、#LOAD CONFIGを使用する前に、最初に#SAVE CONFIGを使用して設定を再保存する必要があることを示します。

機械ユーザが他のNCプログラムで保存済み設定へ不適切にアクセスすることを回避するには、#CLEAR CONFIGが常に有益です。

プログラミング例

クリアランスプログラムを実行した後、次の保存済みの設定を削除します。

```
%Clearance

N10 G53
N35 #LOAD CONFIG [AX AXLINK]    Restoring of the saved axes
                                configuration and axis couplings
                                under coupling group 1
N40 #ECS ON           Definition of a EKS for the execution
                      of the retract travel
N.. X.. Y.. Z..
:
N200 #AX LINK OFF ALL
:
N.. X.. Y.. Z..
N.. #CLEAR CONFIG     After finishing the retraction travel
                      deleting of the saved configuration
N99 M30
```

12.30 作業スペース/保護スペースのモニタリング

制御領域を使用すると、特定の軸群によって、工具と機械または加工物の部分との間の衝突が生じることが回避されます。制御領域は、作業スペースまたは保護スペースとして設計できます。

制御領域は、円筒形または多角形の3次元物体として定義できます。3番目の次元は、最小および最大定数値で定義されます。軸の割り当ては、現在有効な作業平面(G17など)に応じて異なります。

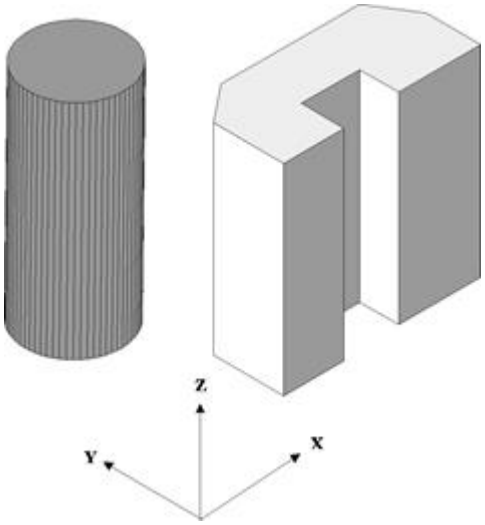


図 82: 図11-17: 3D制御領域の定義(円筒形、多角形)

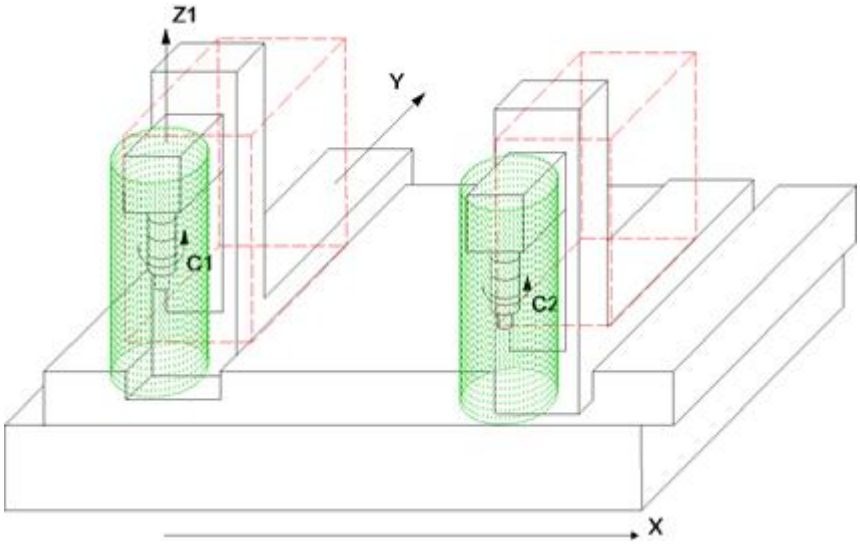


図 83: 図11-18: アプリケーション例

作業スペースの外に出ることは許可されず、保護スペースの中に入ることは許可されません。制御領域には現在の工具データが含まれ、工具ヒントの妥当性をチェックできます。

NCプログラムで、作業および保護スペースの定義、選択、および選択解除が行われます。同時に複数の制御領域を有効にすることができます。制御領域を選択解除すると、それを再定義できます。

作業スペース/保護スペースのモニタリングは、以下に関して可能です。

- 直線補間
- 円弧補間(方向G17/G18/ G19とは無関係)。
- キネマティックトランスフォーメーション(工具中心点のモニタリング)。

- 多項式輪郭加工
- ヘリカル補間

以下は許可されません。

- 直交変換(#(A)CS) ([FCT-C15]を参照)
- 手動操作モード

12.30.1 制御領域の定義

作業または保護領域の定義は、NCプログラムで直接に行う必要があります。その定義は、#-コマンドと標準動作ブロックシーケンスから構成されます。標準動作ブロックは、アブソリュート指令でプログラムする必要があります。平面内の制御領域の形状は、直線動作ブロックによって形成された閉じた多角形(ブロックシーケンスの終点と起点は同一であることが必要)か、または一周円によって定義されます。制御領域の3番目の次元の偏位とその他の特性は、割り当てられた#-コマンドで定義されます。

制御領域の定義の開始:

#CONTROL AREA BEGIN [ID *<expr>* WORK | PROT POLY | CIRC MIN_EXCUR*<expr>* MAX_EXCUR*<expr>*]	(モーダル、プログラムエンドおよびCNCリセット後も有効なまま)

ID*<expr>* 制御領域の識別番号(ID)。その定義は、プログラムエンドおよびRESEY後もグローバルに有効です。

WORK 制御領域は作業スペースです。

PROT 制御領域は保護スペースです。

POLY 制御領域の形状は、閉じた多角形として定義されます。

CIRC 制御領域の形状は、一周円として定義されます。

MIN_EXCUR<expr> 制御領域の3次元の負の方向での制限値。

MAX_EXCUR*<expr>* 制御領域の3次元の正の方向での制限値。

制御領域の定義の終了:

#CONTROL AREA END	(モーダル)

それぞれの制御領域はコマンド#CONTROL AREA ENDで閉じる必要があります。その後にのみ、それ以降の制御領域を定義できます。

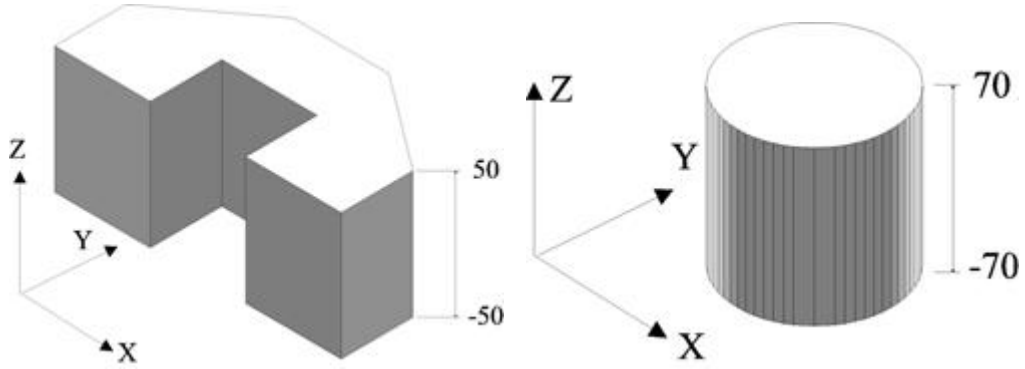
選択解除された制御領域は、同一のIDを使用する新しい定義をプログラミングすることによって再定義できます。

注記

有効な直交変換#(A)CSは、制御スペースの定義中には効果が**ありません**。作業および保護領域は、常にMCS座標系で定義されます。

プログラミング例

```
(Definition of a polygonal work space:)
:
N10 #CONTROL AREA BEGIN [ID1 WORK POLY MIN_EXCUR=-50 MAX_EXCUR=50]
N20 G01 F1000 G90 X-150 Y75   (Start point)
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   (End point identical with start point)
N140 #CONTROL AREA END
:
(Definition of a cylindrical protection space:)
:
N210 #CONTROL AREA BEGIN [ID2 PROT CIRC MIN_EXCUR=-70 MAX_EXCUR=70]
N220 G01 X100 Y0 F10000 (Start point for cylindrical prot. space)
N230 G02 G162 I50 J0
N240 #CONTROL AREA END
:
```



12.30.2 制御領域の選択解除/選択

制御領域の有効化では、TCPが有効な作業スペース内に既に存在する必要があります。同様に、現在の位置では、TCPは保護スペースの有効化で違反することはできません。

#CONTROL AREA ON [ALL] | [ID<*expr*>]	(モーダル)

ID<*expr*> 制御領域の識別番号(ID)。

ALL すべての定義済み制御領域を選択します。

プログラミング例

```
#CONTROL AREA ON [ID3] (Activate a specific control area)
#CONTROL AREA ON ALL   (Activation of all defined control areas)
```

#CONTROL AREA OFF [[ALL] | [ID<*expr*>]]	(モーダル)

ID<*expr*> 制御領域の識別番号(ID)。

ALL すべての定義済み制御領域を選択解除します。

プログラミング例

```
#CONTROL AREA OFF [ID3] (Deactivate a specific control area)
#CONTROL AREA OFF       (Deactivate the latest selected control area)
#CONTROL AREA OFF ALL   (Deactivation of all active control areas)
```

12.30.3 制御領域の削除

#CONTROL AREA CLEAR [ALL] | [ID<*expr*>]	(モーダル)

ID<*expr*> 制御領域の識別番号(ID)。

ALL すべての定義済み制御領域を削除します。

プログラミング例

```
#CONTROL AREA CLEAR [ID3] (Delete a specific control area)
#CONTROL AREA CLEAR ALL   (Deleting of all defined control areas)
```

12.31 forward/backward on pathファンクションの効果

このファンクションを使用できるかどうかは、設定やバージョンの具体的な範囲によって異なります。

これらのコマンドとforward/ backward on pathファンクションの詳細は、機能説明[FCT-C7]を参照してください。

12.31.1 プログラム部分の省略

NCプログラム内で、コマンド#OPTIONAL EXECUTIONによって、NCブロックシーケンスをforward/ backward on path中にスキップできるものとしてマークできます。forward/backward on pathが有効な場合、マークされた部分はインターポレータレベルで省略され、さらに、コマンドが省略されるとき、すべての条件の再計算は行われません。

#OPTIONAL EXECUTION ON | OFF	(モーダル)

このため、すべてのインターポレータ状態(特に、軸位置)は、いくつかのNCコマンドをスキップした後、同一であることが必要です。同一でない場合、パス、速度、加速度は連続的でなくなります。

プログラミング例

```
%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
```

ブロックの省略付き/省略なしの軸の連続パスは、CNCによってチェックされ、検出されます。その他のすべての条件はユーザ自身が確保する必要があり、CNCによってはチェックされません。

複数のコマンドoptional execution on/offのネストは考慮されません。

コマンドの実行がOPTIONAL EXECUTION ON/OFFシーケンス内にある間にシミュレーションをオンにすると、コマンドの省略は、コマンドの実行がOPTIONAL EXECUTION OFFに達するまで、自動的に遅延されます。マークされたシーケンスはすべてが省略されるか、または省略されないかのどちらかしか可能でないため、これが必要です。

OPTIONAL EXECUTIONを選択した(ON)プログラムレベルを離れる(M17、M29)前に、OPTIONAL EXECUTIONを選択解除(OFF)する必要があります。これは、メインプログラムレベルを離れる場合(M30)にも同様です。

12.31.2 バックワードストレージのクリア

NCコマンド#BACKWARD STORAGE CLEARによって、実際の使用可能なバックワードストレージを明示的にクリアできます。これによって、プログラムのクリア位置までしか戻ることができなくなります。

#BACKWARD STORAGE CLEAR	(モーダル)

プログラミング例

```
%backward-storage

N000 G01 X0 F10000
N010 X100 Y123
N020 X100
N030 X200 Y10
N040 X300 Y20

N050 #BACKWARD STORAGE CLEAR
N051 #BACKWARD STORAGE CLEAR  (test multiple clearing
N052 #BACKWARD STORAGE CLEAR  (test multiple clearing

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
```

12.32 有効な同期動作中の工具交換

セキュリティ上の理由により、有効な同期動作中は、新しい工具データ(T (暗黙的D付き)、D、または #TOOL DATA)の使用は通常許可されず、エラーメッセージ20169によってモニタされます。この禁止を次のコマンドによって解除できます。

#FREE TOOL CHANGE ON | OFF	(モーダル)

このNCコマンドは、遅くとも有効な同期動作中の最初の工具交換の前にプログラムする必要があります。禁止の解除はキーワード「ON」で行います。

同期動作および工具交換の禁止の設定は、キーワード「OFF」で行う必要があります。さらに、RESET時およびプログラムエンド時(暗黙的)に、このNCコマンドは選択解除されます。

互換性およびセキュリティ上の理由により、同期動作および工具交換の禁止はプログラム開始時にも常に有効です(デフォルト)。

エラーメッセージ20169は、同時にチャンネルパラメータP-CHAN-00100 (move_tool_offsets_directly)が有効な場合、工具オフセットの直接動作が、有効な同期動作中の従軸に機械の損傷を生じさせる場合があるため、今後はモニタされます。これは、#FREE TOOL CHANGEを使用する場合に、P-CHAN-00100を選択解除する必要があることを示します。

プログラミング例

```
%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
```

12.33 ブロック検索用のプログラム領域のロック

NCプログラムのブロック検索では、任意のプログラム領域をコマンド#BLOCKSEARCH LOCKED/RELEASEDによってロックできます。ブロック検索の開始位置がこの制限された領域の1つに存在する場合は、エラーメッセージP-ERR-21399が出力されます。

ブロック検索ロックには、対応する領域のすべてのローカルおよびグローバルサブプログラム呼び出しも含まれます。

入れ子構造にされたロック領域の場合、ブロック検索ロックには、最初の有効化から最初の無効化までの領域が含まれます(例2を参照)。

ブロック検索に関する詳細は、マニュアル[FCT-C6]を参照してください。

#BLOCKSEARCH LOCKED | RELEASED	(モーダル)

プログラミング例

例1:

ブロック検索用のプログラム領域N40–N100およびすべての含まれるサブプログラムでは、開始位置を選択できません。

```
%BLOCKSEARCH
N10 X0 Y0 Z0
N20 X10
N30 Y10
N40 #BLOCKSEARCH LOCKED
N50 X20
N60 Y20
N65 L GSP.nc
N70 Z20
N80 X30
N90 Z30
N100 #BLOCKSEARCH RELEASED
N110 Y30
N120 X40
N130 Z40
N999 M30
```

例2:

以下の入れ子構造にされた領域では、ブロック検索ロックにはN40-N75が含まれます。

```
%BLOCKSEARCH
N10 X0 Y0 Z0
N20 X10
N30 Y10
N40 #BLOCKSEARCH LOCKED
N50 X20
N55 #BLOCKSEARCH LOCKED
N60 Y20
N65 L GSP.nc
N70 Z20
N75 #BLOCKSEARCH RELEASED
N80 X30
N90 Z30
N100 #BLOCKSEARCH RELEASED
N110 Y30
```

```
N120 X40
N130 Z40
N999 M30
```

12.34 シングルステップモード用のプログラム領域のロック

シングルステップモードの場合、NCプログラムでコマンド#SINGLE STEP [DISABLE / ENABLE]によって任意のプログラム領域をロックできます。これによって、オペレータはマークされたプログラム領域を一度でスキップすることができ、分析するNC行に迅速に到達できます。

入れ子構造にされたロック領域の場合、シングルステップロックには、最初の有効化から最初の無効化までの領域が含まれます(例2を参照)。

さらに、ブロック番号に関連するシングルステップ分解能(#SINGLE STEP [RESOLUTION…])をプログラムすることによって、ユーザは、シングルステップの停止が動作するグリッドを定義できます(例3を参照)。

シングルステップブロックモードの詳細は、マニュアル[FCT-M2]を参照してください。

#SINGLE STEP [DISABLE | **ENABLE** | **RESOLUTION***<expr>***]** (モーダル)

プログラミング例

例1:

プログラム領域N40–N100およびすべての含まれるサブプログラムでは、シングルステップモードは有効ではありません。

```
%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
```

例2:

以下の入れ子構造にされた領域では、無効になったシングルステップモードにN40-N75が含まれます。

```
%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
```

例3:

ユーザによるN番号(10ステップ)と内部列挙(1ステップ処理)。黒の行は、シングルステップ停止処理を示します。灰色の領域内では、シングルステップ停止は実行されません。

```
%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

…
```

12.35 プログラム可能なパスオーバーライド

このコマンドは、必要に応じて、NCプログラムで送りブロックおよび高速送りブロックのさまざまな効果を可能とします。少なくとも1つの軸が動作している場合、パス動作用にプログラムされたオーバーライドが有効です。このファンクションによって、PLCによる送りのリアルタイム効果は影響を受けません。

2番目のファンクションとして、プログラム可能な軸オーバーライド [▶ 555]も使用できます。

軸がパス動作に含まれ、軸オーバーライドも定義されると、有効なパスオーバーライドは、両方のオーバーライド値の乗算から求められます。

ファンクションG166 [▶ 166]は、プログラムされたオーバーライド値の効果を抑制します。

#OVERRIDE [FEED_FACT<*expr*> RAPID_FACT<*expr*>]

FEED_FACT<*expr*> 送りブロックのオーバライド係数[0.1% - 200%]。

RAPID_FACT<*expr*> 高速送りブロックのオーバーライド係数[0.1% - 200%]。

プログラミング例

```
%path_override

N10 G00 G90 X0 Y0 Z0
   (Path override G01 122.765%, G00 155.7%)
N20 #OVERRIDE [FEED_FACT=122.765 RAPID_FACT=155.7]
N30 G01 X100 Y100 Z100 F1000               Effective feed=1227.65
   (Path override G01, G00 100%)
N40 #OVERRIDE [FEED_FACT=100 RAPID_FACT=100]
N50 G01 X200 Y200 Z200 F1000               Effective feed=1000

M30
```

12.36 ドライブファンクションのドライブ非依存切り替え

章11.18のSERCOS固有のコマンドと対照的に、以下のコマンドでは、ドライブパラメータの**ドライブ非依存**の設定が可能です。これらのパラメータは同期設定のみが可能で、コマンドの実行が処理時にインターポレータレベルで行われます。デフォルト設定では、#DRIVEコマンドの実行は、NCプログラムの後続の処理と並行して行われます。これは、プログラム処理が停止しないことを示します。ドライブファンクションの実行が正常に終了するまで補間を停止する場合、パラメータ「WAIT」を使用できます。

ドライブファンクションの実行が正常に終了したかどうかのチェックが一定の時間が経過した後にのみ必要な場合は、コマンド#DRIVE WAIT SYNを使用できます。

12.36.1 同期書込み

#DRIVE WR SYN [AX<*axis_name*> / **AXNR**<*expr*> [**MOTOR**<*expr*>] [**PARAM_SET**<*expr*>]
KEY<*expr*> **VAL**<*expr*> [**WAIT**]]

AX<*axis_name*> (ドライブ)軸の名前

AXNR<*expr*> (ドライブ)軸の論理軸番号、正の整数

MOTOR<expr> モータ切り替え中のドライブ増幅器の選択可能なモータ番号; 正の整数。

PARAM_SET<expr> ドライブ増幅器の新しいパラメータレコード; 正の整数

KEY<*string*> アドレス指定するファンクションの識別、[2]を参照

VAL<*expr*> 送信する値、負または正の整数

WAIT 補間の停止およびドライブファンクションの実行が正常に終了することを待ちます。

このキーワードを使用せずに、ドライブファンクションの実行が正常に終了したことをチェックするには、NCプログラムで、#DRIVE WAIT SYNを使用できます。

注記

連続回転時の有効なスピンドルに対して、モータの切り替えがなくパラメータレコードの変更のないプログラム(MOTOR=<expr>、PARAM_SET=<expr>)を実行することはできません。

12.36.2 確認の同期待機

次のコマンドを使用して、軸用のすべての前の#DRIVE WR SYNが終了したかどうかがチェックされます。ドライブですべての#DRIVE WR SYNが実行されると、インターポレータは停止します。これは、補間軸とスピンドル軸の両方に対して適用します。

#DRIVE WAIT SYN [AX*<axis_name>* / AXNR*<expr>* [SWITCH_OK]]

AX*<axis_name>* (ドライブ)軸の名前

AXNR*<expr>* (ドライブ)軸の論理軸番号、正の整数

SWITCH_OK **すべての**前の#DRIVE WR SYNが終了したかどうかをチェックします。

プログラミング例

```
Synchronous writing with immediate waiting on acknowledgement:

%TOOL_AXLINK1
N05 X0 Y0 Z0
N10 #DRIVE WR SYN [AX=X MOTOR=2 PARAM_SET=4 KEY= Torque_limit VAL=400 WAIT]
N20 X100 Y50 Z30 G01 F3000
N30 X200 Y75 Z40
N65 X300 Y100
N70 X25 Y25 Z25 C25
Nxx
N80 M30



Synchronous writing with subsequent waiting on acknowledgement:

%TOOL_AXLINK2
N05 X0 Y0 Z0
N10 #DRIVE WR SYN [AX=X MOTOR=2 PARAM_SET=4 KEY=Torque_limit 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
```

12.37 セグメンテーションによって速度が最適化されたモーション制御

いくつかの用途(奇妙な形の領域が現れる場合がある運動など)では、CNC側のプログラムされたブロックセグメンテーションを改良して、セグメンテーションによって円弧ブロック(G2/G3)をそれぞれ直線ブロック(G1)に変換することが有効になる場合があります。これは、NCプログラムで次のコマンドを使用して達成できます。

#SEGMENTATION [ON | OFF] [ALL] [[[LIN] [LENGTH<*expr*>] [CIR] [OPMODE<*expr*>] [PARAM<*expr*>]]]	(モーダル)

ON	セグメンテーションの選択
OFF	セグメンテーションの選択解除
ALL	直線および円弧ブロックのセグメンテーション
LIN	直線ブロックのセグメンテーション
LENGTH<expr>	結果としての直線ブロックの長さ[mm]
CIR	円弧ブロックのセグメンテーション
OPMODE<expr>	円弧セグメンテーションのモード: 0: 必要な弦の誤差の指定。ブロック長は自動的に計算されます(デフォルト)。 1: 必要な弦の誤差が示されます。 弦の誤差のそれぞれのブロック長はキーワードPARAMによって定義されます。
PARAM<*expr*>	弦の誤差、またはその結果としての直線ブロックの長さ、選択されたOPMODEに応じて異なります[0.1mm]

セグメンテーションの選択中に、LINおよびCIRを除くその他のパラメータ設定がプログラムされない場合、以下のデフォルト設定が有効です。

LENGTH	1mm
OPMODE	0
PARAM	0.1mm

プログラミング例

```
Selection of linear segmentation with default parameterization:

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]    ;Deselection
N100 M30


Selection of linear segmentation + parameterization:

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]     ; Deselection
N100 M30


Selection of linear - and circular segmentation + parameterization:

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      ;Deselection
N100 M30                        ;Alternative: #SEGMENTATION OFF [LIN CIR]


Combined selection of linear - and circular segmentation:

N30  #SEGMENTATION ON ALL
N40  X3     Y25
N50  X15    Y15
N60  X23    Y12
N70  X25    Y25
N80  X30    Y35
N90  #SEGMENTATION OFF ALL      ;Deselection
N100 M30
```

12.38 形状の拡大と縮小

#SCALEコマンドを使用し、軸特有の係数を指定することによって、形状または位置のスケールを拡大/縮小することができます。さまざまな係数を指定することによって、形状の拡張/圧縮を行うこともできます。

スケーリング対象を以下に示します。

- 補間軸(直線軸、例えば、X、Y、Z、U、V、W)
- 位置決め軸(独立軸、振動軸)。

オフセットは通常、オフセットが選択されたか、または#SCALE ONの前後にオフセットがプログラムされたかどうかに関係なく、スケーリングされません。

スケールされる値は、現在有効な座標系の原点を常に基準点にします。特に形状を絶対座標で記述するときは、スケーリングを選択する前に、適切なGファンクションG53～G57、G159、G92、または回転座標系#ROTATIONおよび#CSの定義用ファンクションによって、原点を形状の起点に配置することをお奨めします。

#SCALE [ON] *<axis_name><expr>* {*<axis_name><expr>*}	(モーダル)
#SCALE OFF	(モーダル)

ON	スケーリングの有効化
OFF	スケーリングの無効化
<axis_name><expr>	軸特有のスケーリング係数。その係数は> 0であることが必要です。 係数が≤ 0の場合、エラーメッセージが出力されます。

スケーリング係数の定義および選択は、同時にまたは別のステップで指定できます。これは、例えば、最初にスケーリング係数を定義した後、2番目のコマンドでスケーリングを有効にできることを示します。

プログラムされたスケーリング係数は、M30を使用したプログラムの最後まで格納されたままで、さらに#SCALE ON/OFFが多重使用されている場合には、再度使用することができます。

注記

スケーリングがキャンセルされると、スケーリング係数を変更できます。

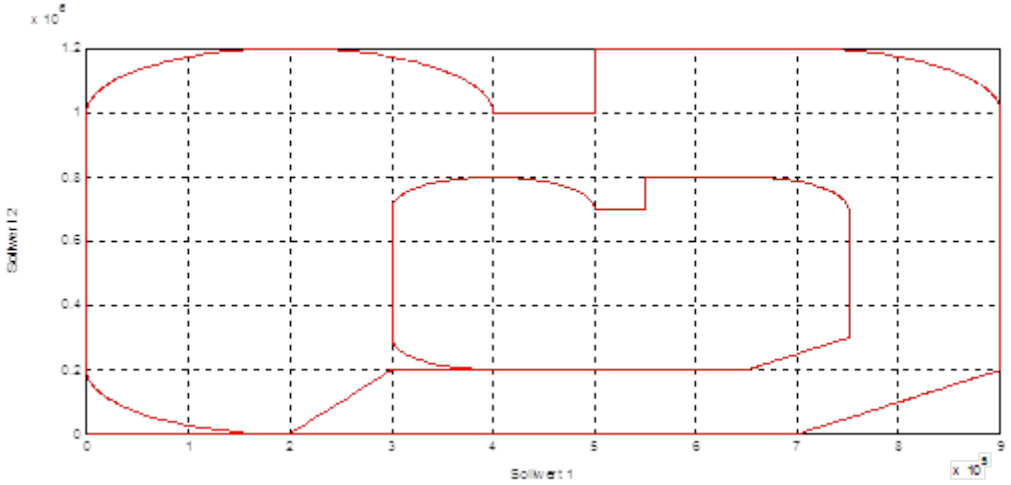
スケーリング中にスケーリング係数を変更すると、エラーメッセージが出力されます。

スケーリングと円弧プログラミングを組み合わせるときは、以下の条件を守る必要があります。

- 円弧の真のスケーリングは、円弧補間に含まれる複数軸のスケーリング係数が同一である場合のみ意味があります。これは、特に、半径がRまたはG163経由で指定された円弧のプログラミング(この場合、その半径の係数は「X」でプログラムされたスケーリング係数に基づいているため)に適用されます。
- 円弧がI、J、Kでプログラムされると、相互に異なるスケーリング係数も使用できます。ただし、この場合は通常、円弧の中心点補正でエラーが発生するか、または事実上の意味がない円弧セグメントが作成されます。

プログラミング例

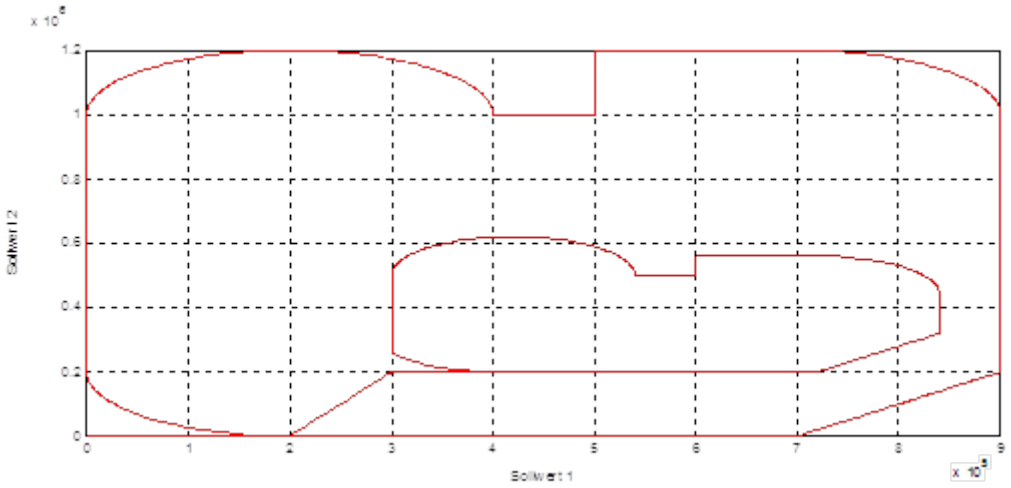
```
;Scaling of an absolute programmed contour with identical factors

%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     ; Basic contour
N075 #SCALE X0.5 Y0.5 ;Definition of scaling factors
;Definition of zero point origin of the scaled contour
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
```



設定値= コマンド値

```
;Scaling of an absolute programmed contour with different factors


%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
```

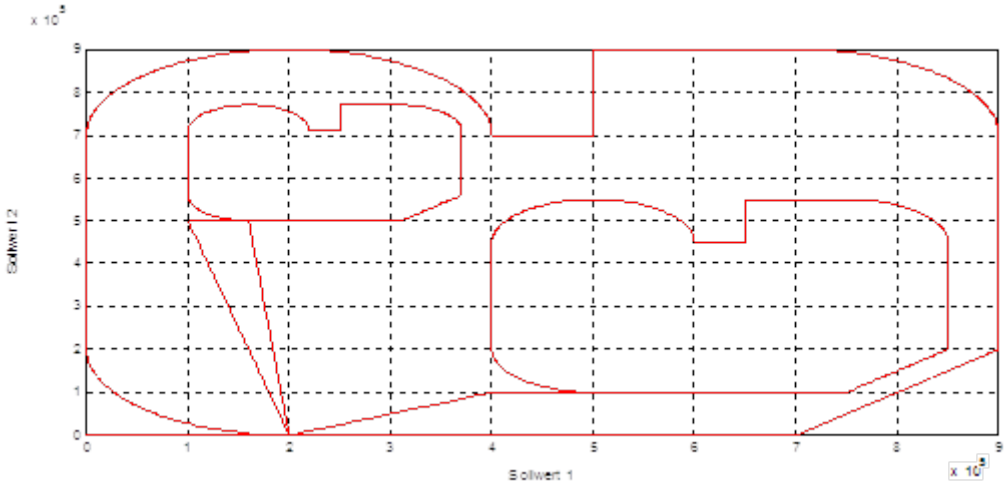
```
%scale

N015 G53
N020 G00 X0 Y0 Z0
N065 LL Cont1_abs     ;Basic contour
N075 #SCALE X0.6 Y0.3 ;Definition of scaling factors
;Definition of zero point origin of the scaled contour
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
```



設定値= コマンド値

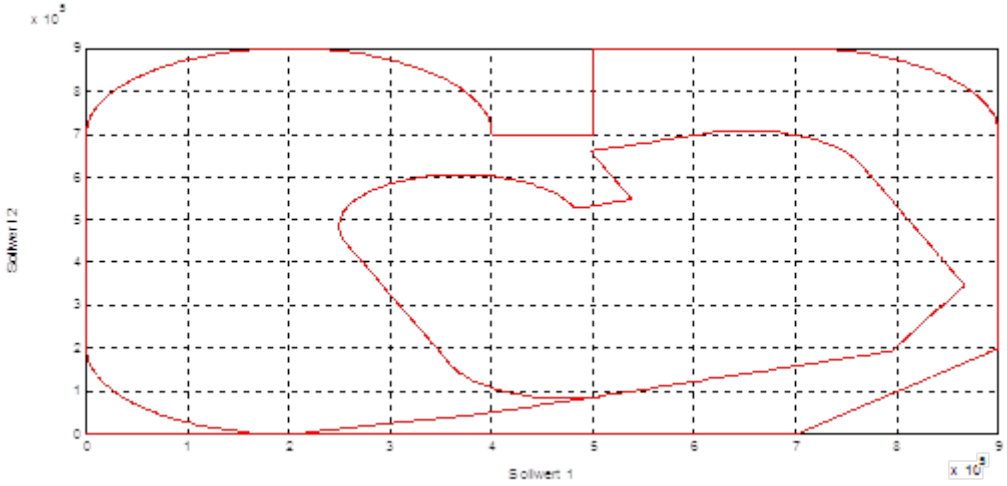
```
;Multiple different scaling of a relative programmed contour

%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
;Basic contour
N030 LL Cont1_rel
N040 #SCALE X0.3 Y0.3 ;Definition 1 of scaling factors
;Definition 1 of starting point of the scaled contour
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 of scaling factors
```

```
;Definition 2 of starting point of the scaled contour
N100 G90 G0 X40 Y10
N105 #SCALE ON
N110 LL Cont1_rel
N115 #SCALE OFF
N125 M30
```



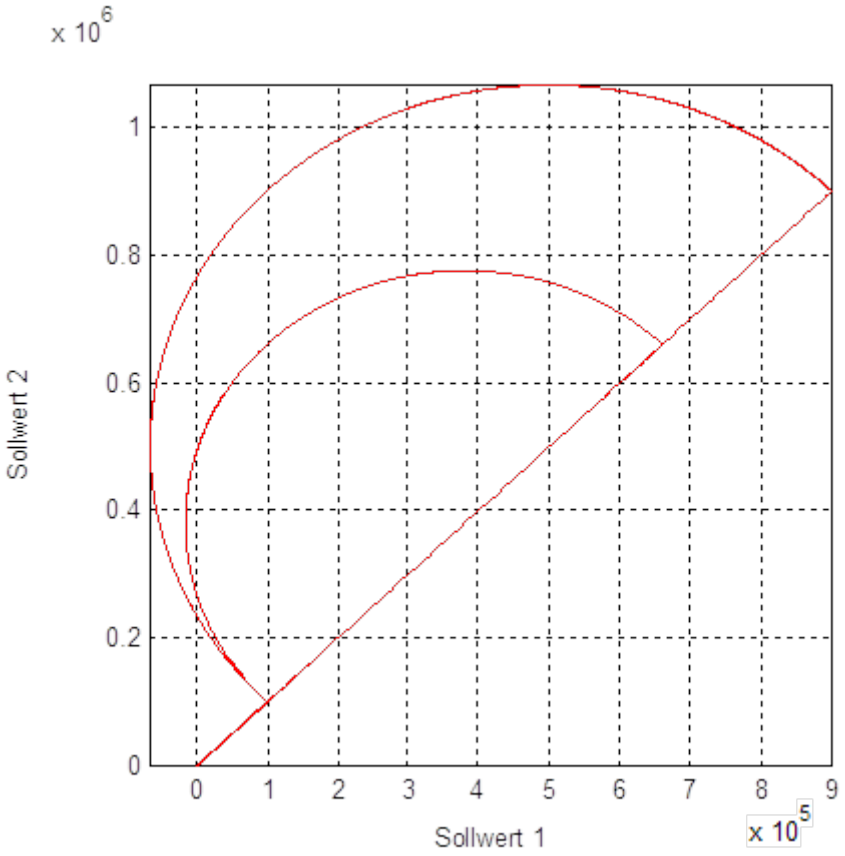
設定値= コマンド値

```
;Scaling of a contour in a coordinate system, which is shifted and rotated with #CS

%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
;Basic contour
N030 LL Cont1_rel
N040 #SCALE X0.6 Y0.6 ;Definition of scaling factors
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
```



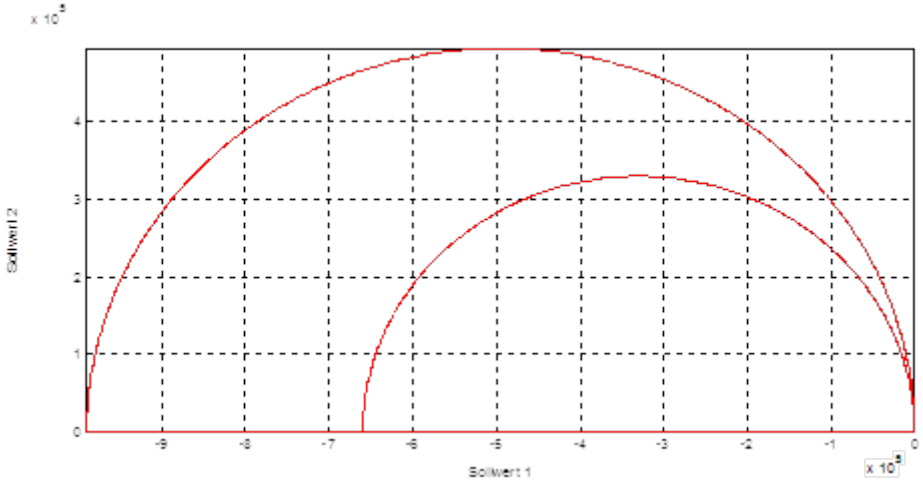
設定値= コマンド値

```
;Scaling of a half circle, programmed with I, J

%L Circle1
G02 G91 X80 Y80 I40 J40
M17

%scale

N015 G53
N020 G00 X0 Y0 Z0
N025 G01 X10 Y10 F1000
;Basic half circle
N030 LL Circle1
N040 #SCALE X0.7 Y0.7  ;Definition of scaling factors
N055 G90 G0 X10 Y10
N060 #SCALE ON
N065 LL Circle1
N070 #SCALE OFF
N075 G90 G0 X0 Y0

M30
```



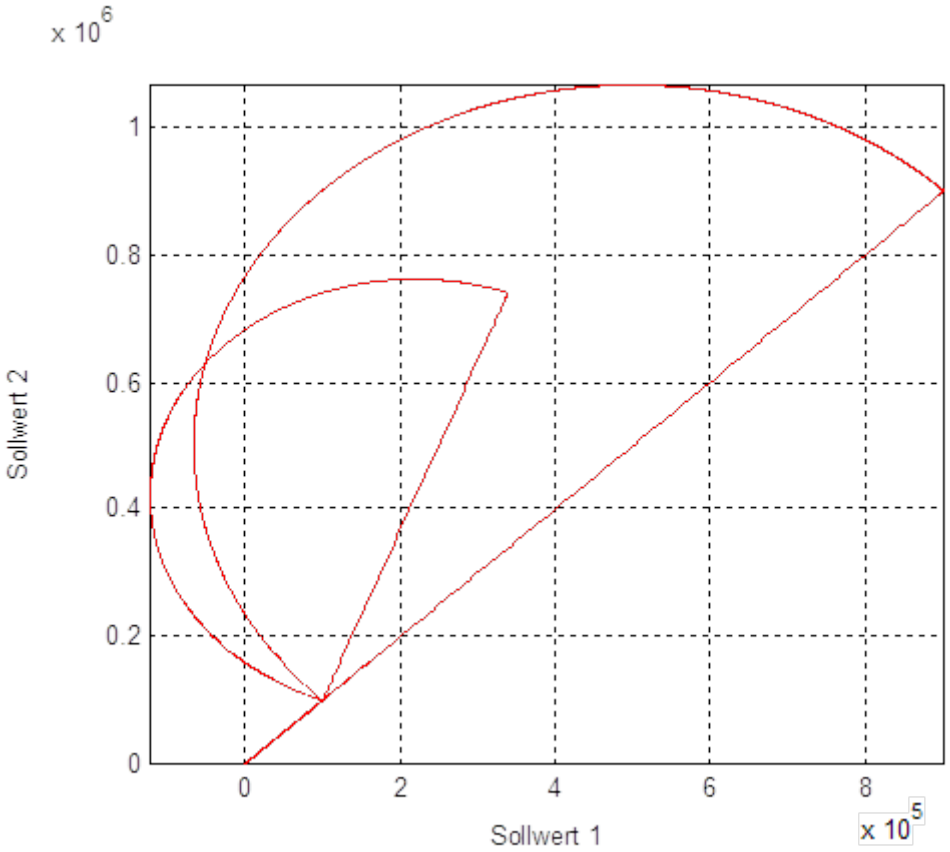
設定値= コマンド値

```
;Scaling of a half circle, programmed with R

%L Circle2
G03 R33 X-66 Y0
M17
%scale

N015 G53
N020 G00 X0 Y0 Z0 F1000
;Basic half circle
N030 LL Circle2
N040 #SCALE X1.5 Y1.5  ;Definition of scaling factors
N055 G90 G0 X10 Y10
N060 #SCALE ON
N065 LL Circle2
N070 #SCALE OFF
N075 G90 G0 X0 Y0

M30
```



設定値= コマンド値

```
;Asymmetric scaling of a half circle, programmed with I, J

%L Circle1
G02 G91 X80 Y80 I40 J40
M17

%scale

N015 G53
N020 G00 X0 Y0 Z0
N025 G01 X10 Y10 F1000
;Basic half circle
N030 LL Circle1
N040 #SCALE X0.3 Y0.8  ;Definition of scaling factors
N055 G90 G0 X10 Y10
N060 #SCALE ON
N065 LL Circle1
N070 #SCALE OFF
N075 G90 G0 X10 Y10

M30
```



設定値= コマンド値

12.39 パンチングおよびニブリング

パンチングおよびニブリング中には、自動ストローク動作がプログラム可能な部分的ステップ数で実行されます。ユーザは、ストローク動作シーケンスを自由に定義できます。ストローク動作は、有効な加工平面での有効な移動動作中にのみトリガされます。

12.39.1 移動距離の分割とプログラミング

移動距離は、部分的セグメントの*長さ*か、または部分的セグメントの*数*を指定することによって分割されます。プログラミングはモーダルです。

部分的セグメントの長さを指定することによる分割は、移動ブロックが部分的セグメントに均一にサブ分割されるように行われます。実際の各部分的セグメントの長さは、プログラムされた部分的セグメントの長さ以下です。

移動長さが部分的セグメントの数を指定することによって分割されるときは、部分的セグメント長が自動的に計算されます。

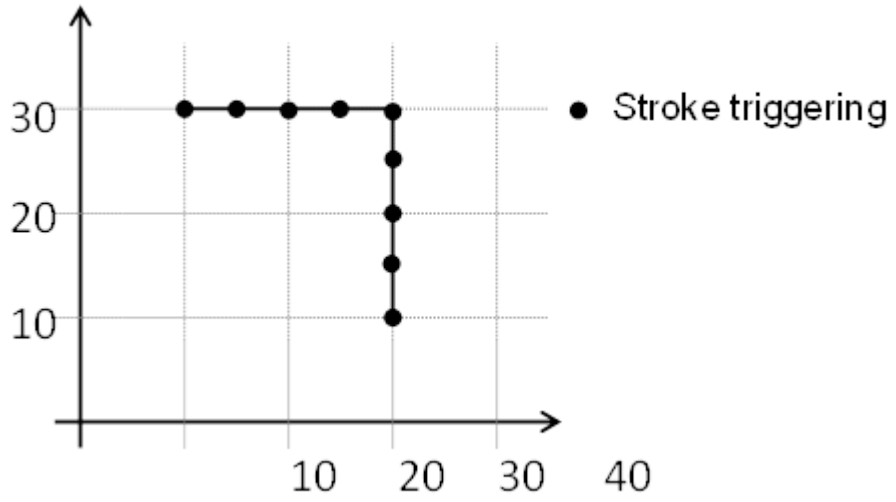


図 84: 図11-19: 直線ブロックの分割

プログラムされた円弧ブロックは部分的直線動作に変換され、移動距離の分割は円の円弧を示します。

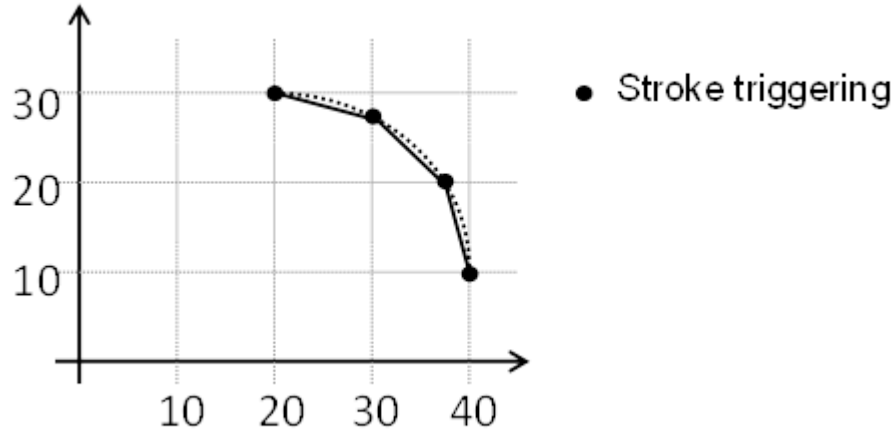


図 85: 図11-20: 円弧ブロックの分割

ストローク動作の定義:

ストローク動作を定義するには、最大*10ブロック*に制限されたシーケンス(以下の2つのコマンドで区切られます)がプログラムされます。

#STROKE DEF BEGIN	ストローク定義の開始	(モーダル)
#STROKE DEF END	ストローク定義の終了	(モーダル)

ストローク定義では、以下の制限されたファンクションの範囲のみが許可されます。

#STROKE DEF BEGIN

G0、G01、M、H、G261、G61、G260、G60、G04、#TIME、G90、G91

#STROKE DEF END

ストローク定義内で他のNCコマンドを使用すると、エラーが発生します。

パンチングおよびニブリングファンクションの有効化/無効化:

#PUNCH ON | OFF [[LENGTH*<expr>* **| NUM***<expr>***]]**	(モーダル)

#NIBBLE ON | OFF [[LENGTH*<expr>* **| NUM***<expr>***]]**	(モーダル)

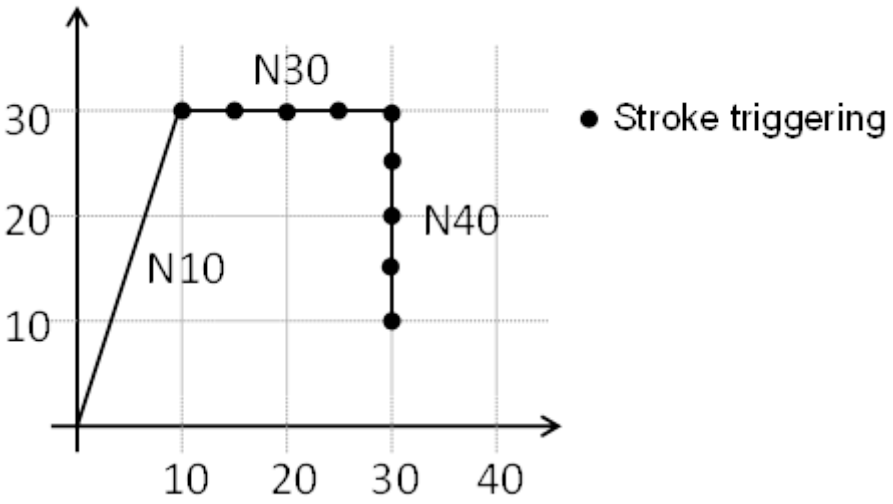
ON	パンチング/ニブリングを有効にします
OFF	パンチング/ニブリングを無効にします
LENGTH*<expr>*	部分的セグメントの長さ[mm]、この長さの後にストローク動作が自動的に挿入されます。 #NIBBLE ONの場合、(起動時でも)ストローク動作は、有効な加工平面での最初の移動動作中に実行されます。
NUM*<expr>*	1つの動作コマンド内で生成する部分的セグメントの数。ストローク動作は、すべての各部分的セグメントの後にトリガされます。

LENGTHとNUMは排他的です。すなわち分割は、部分的セグメントの長さか、または部分的セグメントの数に基づいて行われます。

プログラミング例

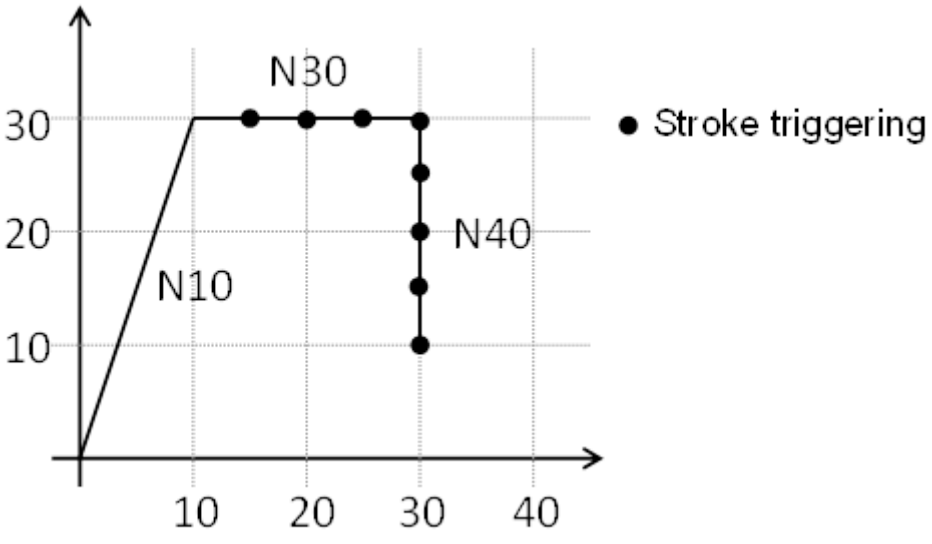
```
%Nibble
N10 G90 G17
N20 #STROKE DEF BEGIN
N30 G04 0.01
N40 G91 Z10
N50     Z-10
N60 #STROKE DEF END

N10   X10    Y30 C0
```

```
N20 #NIBBLE ON [LENGTH 5]
N30   X30    Y30 C180
N40   X30    Y10 C360
N90 #NIBBLE OFF
N100 M30
```



プログラミング例

```
%Punch
N10 G90 G17
N20 #STROKE DEF BEGIN
N30 G04 0.01
N40 G91 Z10
N50     Z-10
N60 #STROKE DEF END

G90 G17
N10   X10    Y30
N20 #PUNCH ON [LENGTH 5]
N30   X30    Y30
N40   X30    Y10
N90 #PUNCH OFF
N100 M30
```



設定:

機能を使用するには、スタートアップリスト([STUP])で以下の設定を行う必要があります。

configuration.channel[0].path_preparation.function FCT_DEFAULT | **FCT_PUNCHING**

configuration.channel[0].path_preparation.function FCT_DEFAULT | **FCT_NIBBLING**

12.39.2 その他のファンクション

ストロークトリガ時間の変更:

良好な加工結果を達成するために、ストロークトリガ時間を変更することができます。その結果、ストロークの**プリ**トリガによって、一定の信号処理むだ時間の補正が可能になります。軸の過渡応答、または、例えば、圧力パッドによるパンチング中の押下時間をストロークの**ポスト**トリガによって相殺することができます。

設定に応じて、ストロークトリガ時間の変更は以下のイベントの1つを示します。

- 補間エンドポイントに達した → G00、G01、G02、G03
- 最も遅い軸のIn-Posウィンドウ → G60

ストロークポストトリガのためには、1つのMファンクションと組み合わせたドウエル時間G04が使用されます。

プリトリガのためには、タイプMET_SVSのMファンクションが使用されます。

オフセットタイミング分解能は、最小単位としてのCNCコントローラのサイクルタイムに応じて異なります。

ストロークプリトリガの例:

ユーザ定義のMファンクションM97は、ブロックシーケンスの同期時間に達する40ミリメートル前に、PLCへ出力する必要があります

チャンネルパラメータリストからの抜粋

```
# Definition of M functions and synchronisation methods
# ======================================================
m_synch[97] 0x02000000 MET_SVS

# Setting the pre-output time, pre-output path with MET_SVS type
# ================================================================
m_pre_outp[97] 40000 in us
```

```
N10 #STROKE DEF BEGIN
N30 M97
N40 #STROKE DEF END
```

ストロークポストトリガの例:

M96は、ストロークをトリガするユーザ定義のMファンクションです。ストロークは、ストローク位置に達した後の50ミリ秒だけ行われます。

In-Posウィンドウに転送されるMファンクションの出力の例:

M96は、ストローク動作をトリガするユーザ定義のMファンクションです。ストローク動作は、すべての軸の物理的ストローク位置(実際の位置はウィンドウ内にある)に達した後の50ミリ秒だけ行われます。

```
N10 #STROKE DEF BEGIN
N20 G04 0.005
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
```

12.39.3 制限事項

有効な加工平面を傾ける回転の使用は許可されません。対照的に、XY平面が有効であるときのZ軸に関する回転は許可されます。この制限事項は、ストロークシーケンスで軸動作がプログラムされた場合のみ、適用されます。例えば、Mファンクションの出力はまだ可能です。

12.40 エッジ加工の制御

加工技術に応じて、鋭い形状(エッジ)に対する加工プロセスを特別な方法で制御することが必要になる場合があります。鋭いエッジの場合(2つの形状エレメント間の角度相違によって定義された)、あらかじめ定義されたパラメータに応じて、エッジでのパス速度が変更されます。NCブロックで、直線または円弧形状エレメントをプログラムできます。この場合、それが内面形状であるか、または外面形状であるかがチェックされません!

エッジ加工ファンクションはチャンネルパラメータリストで設定され、プログラム開始から効果があります。正確なパラメータ設定は、マニュアル[CHAN//Chapter Settings for edge machining]に記載されています。

さらに、エッジ加工ファンクションは、NCプログラムで次のNCコマンドを使用して、選択/選択解除およびパラメータ設定を行うこともできます。

エッジ加工の選択:

#EDGE MACHINING ON [[ANGLE_LIMIT<*expr*>] [WAIT_TIME<*expr*>] [PRE_DIST<*expr*>] [PRE_FEED<*expr*>] [POST_DIST<*expr*>] [POST_FEED<*expr*>] [DISABLE_FEED_ADAPTION<*expr*>]]	(モーダル)
または	
#EDGE MACHINING ON [[ANGLE_LIMIT] [WAIT_TIME] [PRE_DIST] [PRE_FEED] [POST_DIST] [POST_FEED] [DISABLE_FEED_ADAPTION]]	(モーダル)
または	
#EDGE MACHINING ON DEFAULT	(モーダル)

ON	エッジ加工の選択。
ANGLE_LIMIT<*expr*>	臨界角[度]。2つの形状エレメント間の角度がこの臨界角よりも小さい場合、この特別なエッジファンクションが有効になります。
WAIT_TIME<*expr*>	エッジでの待機時間[秒]。
PRE_DIST<*expr*>	エッジより前の距離[mm]、このためにPLC信号が作成されます。
PRE_FEED<*expr*>	エッジより前の送り[mm/min]、PRE_DISTから エッジでの停止までの動作で使用されます。
POST_DIST<*expr*>	エッジより後の距離[mm]、このためにPLC信号が作成されます。
POST_FEED<*expr*>	エッジより後の送り[mm/min]、POST_DISTまでの動作で使用されます。この後、処理は元の送りで続けられます。
DISABLE_FEED_ADAPTION<*expr*>	送り調整の切り替え、ブール値: 0: 送り調整は有効である 1: 送り調整は無効である。PRE_ /POST_FEEDは有効ではありません。
DEFAULT	ON選択と組み合わせて、チャンネルパラメータリスト[P-CHAN-221 - P-CHAN-226、P-CHAN-300]のデフォルト値が有効です。

すべてまたはいくつかの単一キーワードANGLE_LIMIT、WAIT_TIME、PRE_DIST、PRE_FEED、POST_DIST、またはPOST_FEEDの後に値をプログラムしないと、チャンネルパラメータリスト[P-CHAN-221 - P-CHAN-226、P-CHAN-300]のデフォルト値が使用されます。

コマンド#EDGE MACHINING OFFは、エッジ加工ファンクションを選択解除します。選択解除と組み合わせて、他のパラメータをプログラムすることはできません。

エッジ加工の選択解除:

#EDGE MACHINING OFF	(モーダル)

設定:

機能を使用するには、スタートアップリスト([STUP])で以下の設定を行う必要があります。

configuration.channel[0].path_preparation.function FCT_DEFAULT | **FCT_EMF**

注記

有効なエッジ加工中のパラメータの変更は許可されます。ただし、関連する形状エレメントの処理が終了した後にのみ変更を設定することを推奨します。これによって、エッジ加工の未定義の動作が回避されます。

プログラミング例

```
%edge_machining

(Selection of edge machining and setting of parameters with value)
#EDGE MACHINING ON [ANGLE_LIMIT=90 WAIT_TIME=0.2 PRE_DIST=5 PRE_FEED=800
                    POST_DIST=10 POST_FEED=1600]

(Selection of edge machining and setting of parameters on default)
#EDGE MACHINING ON [ANGLE_LIMIT PRE_DIST PRE_FEED POST_DIST POST_FEED
                    WAIT_TIME DISABLE_FEED_ADAPTION]

or

(Selection of edge machining and setting of parameters on default)
#EDGE MACHINING ON DEFAULT

(Deselection of edge machining)
#EDGE MACHINING OFF
```

12.41 動的な重み付けの切り替え

動的なリミットのパスおよび半径依存の重み付けでは、チャンネルパラメータリストにテーブルを設定できます。パラメータP-CHAN-00190およびP-CHAN-00230を使用して、これらの機能を有効にでき、機能はコントロールのスタートアップ後に有効になります。

次のNCコマンドを使用して、ランタイム中に、これらの機能をオン/オフに切り替えることができます。

#DYNAMIC WEIGHT [ON | OFF] [[PATH | CURVE]]	(モーダル)

ON	重み付けオン
OFF	重み付けオフ
PATH	パス依存の重み付け(P-CHAN-00190を参照)
CURVE	円弧半径依存の重み付け(P-CHAN-00230を参照)

有効化または無効化では、1つのキーワードPATHまたはCURVEをプログラムする必要があります。

プログラミング例

```
N20 G00 X0 Y0 Z0 F10000
N30 #DYNAMIC WEIGHT ON [PATH]   ; Activation path dependend weighting
N40 X3  Y25
N50 X15 Y15
N60 X23 Y12
N70 X25 Y25
N80 X30 Y35
N90 #DYNAMIC WEIGHT OFF [PATH]   ; Deactivation
N100 M30
```

13 工具ジオメトリ補正

工具ジオメトリは、長さ、半径、および軸シフトに関して補正されます。対応する工具ジオメトリ補正データを使用可能にする必要があります。

工具補正データは、Tnn (P-CHAN-00014)で自動的にパラメータ設定されている場合、それぞれ、Dワードを使用して選択されます。

D *<expr>*	工具補正の選択	(モーダル)

<expr> 工具補正データブロックの番号。正の整数

Dワードは、以下の値が含まれる工具データブロックを定義します。

- 工具の長さ(主面に直交する)
- 工具の半径(主面内の)
- 軸オフセット座標(すべての軸方向の)
- 数値記述の計測単位(mm/Inch)
- 工具有効性コード
- 工具動的データ

注記

工具の長さ(主面に直交する)および軸オフセット座標(軸方向の)の工具補正データの有効性の時間は、チャンネルパラメータP-CHAN-00100によってプログラムされます。

以下の2つのモードを区別できます。

- 補正動作は、移動準備ファンクションのプログラミングなしに、Dワードで直接に実行されます。
- セキュリティ上の理由により、コントロールは次の**絶対**動作ブロックで、対応する軸の補正動作のみを実行します (デフォルトモード)。

⇨ 工具補正データの追加用G91モードでは、以下が重要です。

```
N10  D16
N20  G00  X0  G91
```

プログラミングは、X軸(相対的な動作0に対応)のいかなる動作も引き起こしてはなりません。したがって、次の遷移情報がアブソリュートモード(G90)でプログラムされているときのみ、軸の工具補正データが機能します。

同時に、両方のモードに注意してください。工具半径は工具径補正に送信され、等距離の計算に影響を与えます。動作補正は、常に遷移準備ファンクションと組み合わせて行われます。

補正動作の実行の規則は、工具補正の選択解除でも有効です。

D00	工具補正の選択解除

13.1 工具長補正

次の例では、Z軸方向の工具長補正が実行されます。ブロックN30で補正データブロックD16を選択する一方で、移動データブロックと連携してN30でZ軸方向の補正動作を行います。

プログラミング例

```
N10  G01  F900  G17    (X-Y-Plane; length compensation in Z+ direction
N20  X150  Y10  Z10
N30  D16  Y40  Z15     (Selection of length compensation D16)
  .                    (Compensating movement is executed)
  .
N100  D20              (Selection of length compensation D20)
  .                    (Compensating movement occurs only with)
  .                    (next traverse block in the Z-direction)
  .
N200  G0  D0  X0  Y0  Z0     (Deselection of TLC)


   Compensation data block    D0  :   Length = 0      Radius = 0
     .                        D16 :   Length = 5      Radius = 5
     .
                              D20 :   Length = 12,5   Radius = 5
```

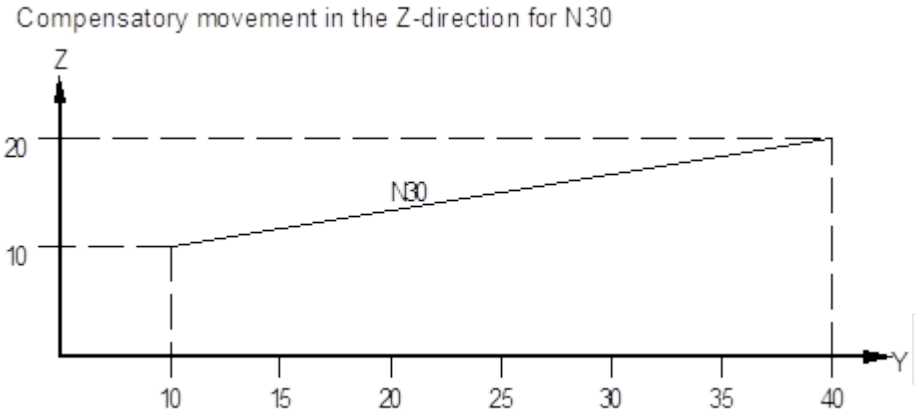


図 86: 図12-1: 工具長補正の例

13.2 工具径補正(TRC)

工具径補正(TRC)により、工具ジオメトリとは関係なく加工物の形状をプログラミングできます。TRCが選択されている場合(G41、G42)、プログラムされた工具形状から等間隔のツールパスが「工具半径」の距離で計算されます。

工具径補正は、G17、G18、またはG19で選択された平面で動作します。Dワードに保存された工具補正ブロック(「工具ジオメトリ補正 [▶ 366]」の章を参照)は、工具補正値として使用されます。

新しいDワードの選択、または変数V.G.WZ_AKT.Rの書き込みにより、TRCが有効な場合でも工具半径を変更できます。工具半径の変更は、直線ブロックと組み合わせる場合にのみ可能です。

負の工具半径を使用する場合は、TRCの選択側が自動的に変更されます。

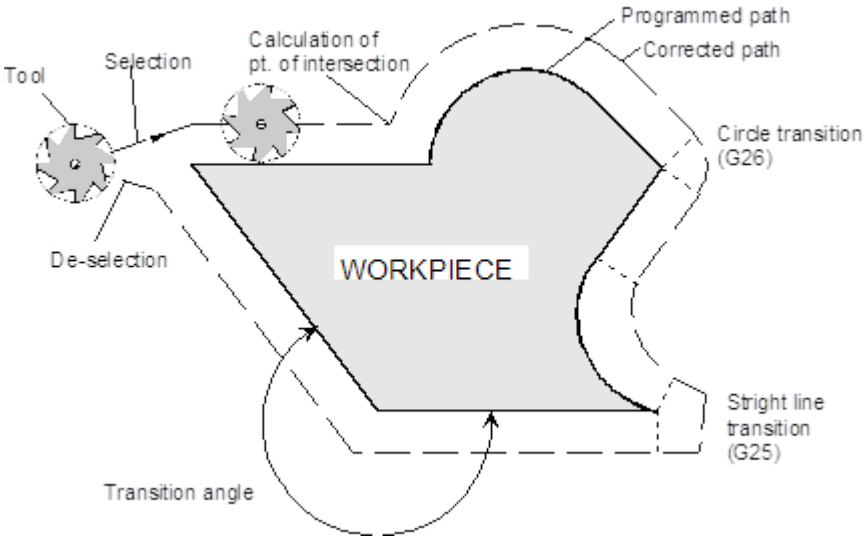


図 87: 図12-2: 工具径補正のメカニズムと用語

TRCに関連するすべてのGファンクションの概要:

選択/選択解除

G40	TRC選択解除	(モーダル、デフォルト)
G41	形状のTRC左の選択	(モーダル)
G42	形状のTRC右の選択	(モーダル)

選択/解除方法:

G138	TRCの直接選択/選択解除	(モーダル)
G139	TRCの間接選択/選択解除	(モーダル、デフォルト)
G237	TRCの直接選択/選択解除	(モーダル)
G238	TRCの内角選択	(モーダル)
G239	ブロックを使用しないTRCの直接選択/選択解除	(モーダル)
G05	TRCの接線選択/選択解除	(ノンモーダル)

注記

TRCが有効な場合、モーダル選択メソッドを直接G238に変更することはできません。

遷移の選択:

G25	直線遷移	(モーダル、デフォルト)
G26	円弧遷移	(モーダル)

送りの調整:

G10	一定送り	(モーダル、デフォルト)
G11	適合された送り	(モーダル)

形状マスキング:

G140	形状マスキングの選択解除	(モーダル、デフォルト)
G141	形状マスキングの選択	(モーダル)

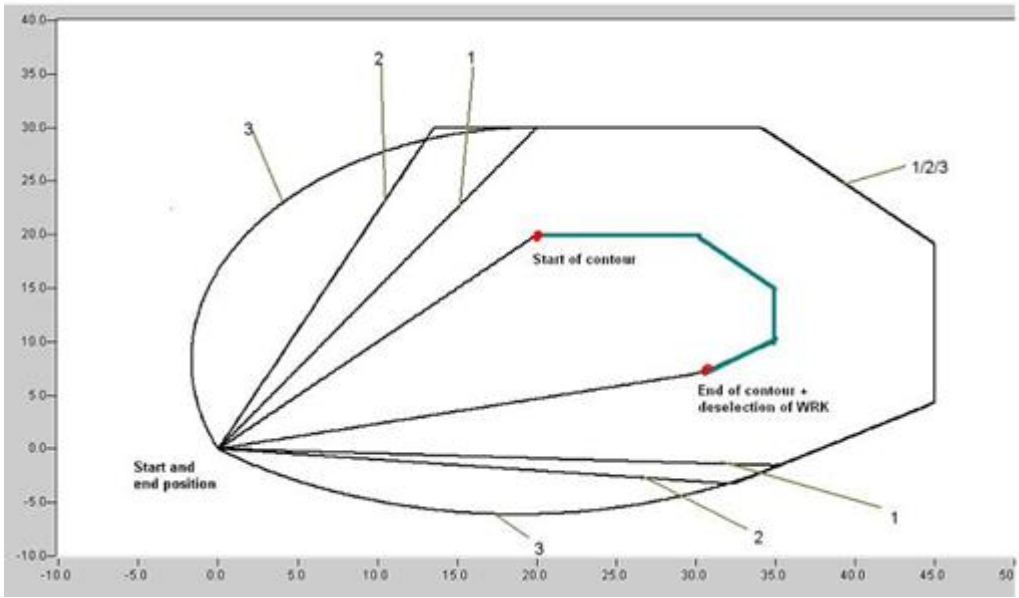
プログラミング例

工具径補正を使用した場合に可能な選択方式を記載します。直線および円弧状遷移のTRC遷移タイプ(G25/G26)と組み合わせて使用する、TRCの直接および間接選択、そして接線変化が連続的な選択/解除(G139/G138/G05)のコマンドが示されています。

テストプログラムは、工具径10 mmで実行されます。

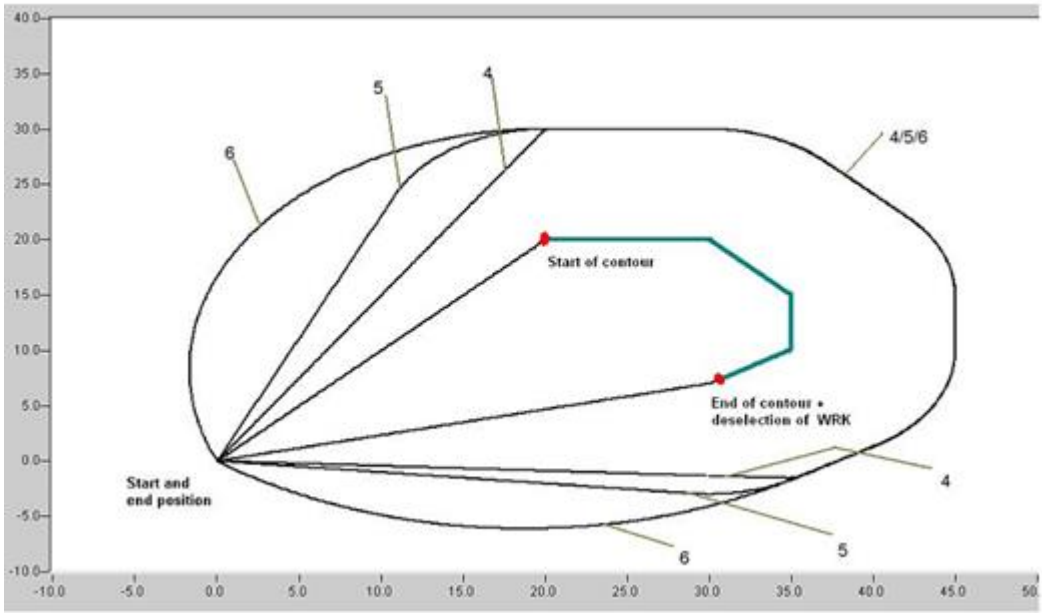
例1	G138	G139	G05
G25	**パス1**	**パス2**	**パス3**

```
%WZKG25 (Contour for G25)
N10 G00 G90 T1 D1 X0 Y0 Z0 G17

(Presentation of contour)
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

(Path 1)
N100 G138 G41               (Direct selection and TRC left of contour)
N105 G01 X20 Y20 F1000      (Compensation movement after G41)
N110 G25                    (G25 linear transition)
N115 G1 G91 X10
N120 X5 Y-5
N125 Y-5
N130 X-5 Y-3
N135 G138 G40               (Direct deselection and TRC-off)
N140 G01 G90 X0 Y0 F1000    (Compensation movement after G40)

(Path 2)
N200 G139 G41               (Indirect selection and TRC left of contour)
N205 G01 X20 Y20 F1000      (Compensation movement after G41)
N210 G25                    (G25 linear transition)
N215 G1 G91 X10
N220 X5 Y-5
N225 Y-5
N230 X-5 Y-3
N235 G139 G40               (Indirect deselection and TRC-off)
N240 G01 G90 X0 Y0 F1000    (Compensation movement after G40)

(Path 3)
N300 G05 G41                (Tangential selection and TRC left of contour)
N305 G01 X20 Y20 F1000      (Compensation movement after G41)
N310 G25                    (G25 linear transition)
N315 G1 G91X10
N320X5 Y-5
N325 Y-5
N330 X-5 Y-3
N335 G05 G40                (Tangential deselection and TRC-off)
N340 G01 X20 Y20 F1000      (Compensation movement after G41)

N999 M30
```



例2	G138	G139	G05
G26	**パス4**	**パス5**	**パス6**

```
%WZKG26 (Contour for G26)
N10 G00 G90 T1 D1 X0 Y0 Z0 G17

(Presentation of contour)
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

(Path 4)
N400 G138 G41               (Direct selection and TRC left of contour)
N405 G01 X20 Y20 F1000      (Compensation movement after G41)
N410 G26                    (G26 circular transition)
N415 G1 G91 X10
N420 X5 Y-5
N425 Y-5
N430 X-5 Y-3
N435 G138 G40               (Direct deselection and TRC-off)
N440 G01 G90 X0 Y0 F1000    (Compensation movement after G40)

(Path 5)
N500 G139 G41               (Indirect selection and TRC left of contour)
N505 G01 X20 Y20 F1000      (Compensation movement after G41)
N510 G26                    (G26 circular transition)
N515 G1 G91 X10
N520 X5 Y-5
N525 Y-5
N530 X-5 Y-3
N535 G139 G40               (Indirect deselection and TRC-off)
N540 G01 G90 X0 Y0 F1000    (Compensation movement after G40)

(Path 6)
N600 G05 G41                (Tangential selection and TRC left of contour)
N605 G01 X20 Y20 F1000      (Compensation movement after G41)
N610 G26                    (G26 circular transition)
N615 G1 G91X10
N620 X5 Y-5
N625 Y-5
N630 X-5 Y-3
N635 G05 G40                (Tangential deselection and TRC-off)
N640 G01 X20 Y20 F1000      (Compensation movement after G41)

N999 M30
```



工具データの変更:

プログラミング例

```
:
N30  G0  D0 X0 Y0 Z0 G17     (X-Y-Plane)
N40  G0  D100 X10 Y10        (Selection of TLC)
N50  G1  Z0
N60  G0  Z100
N70  G41                     (Selection of TRC with data block D100)
N80  G1  Z0
N90  G2  X10  Y10  I-15      (Full circle with radius 15)
N100 G0  Z100
N110 D2  Z200                (Other compensation data block, i.e.)
                             (other values for TLC and TRC)
N120 G1  Z0                  (Here compensatory movement of TLC occurs)
N130 G1  X20  Y20            (Compensatory movement of TRC)
N140 G02 X10  Y10  I-15      (The TRC is now executed with the data block)
N150 G0  Z100
N160 G40                     (Deselection of TRC)
N170 X0
:
```

工具径の動的変更:

変数による割り当てでも、工具径を変更することができます(「変数と変数の計算 [▶ 419]」の章も参照)。この方法により、たとえば研削工具の磨耗や断裂を移動ブロック内で補正できます。

プログラミング例

```
N00  G1  G90  X0  D6  F5
N10  G41  G26                  (Selection of TRC)
N20  X0  Y250                  (Starting point)
N30  V.P.VERSCHLEISS = 0.010

N100 $FOR V.P.LAUF = 0,100,10  (Grinding cycle)
N110   X300
N120   Y200
N130   X0
```

```
N140   Y250                      (Tool radius is always getting smaller)
N150   V.G.WZ_AKT.R = V.G.WZ_AKT.R - V.P.VERSCHLEISS
N160 $ENDFOR

N200 G40  X300                  (De-selection of TRC)
N999 M30
```

13.2.1 TRCの直接/間接選択(G41/G42)

パートプログラム内の次の移動ブロックで、対応する補正平面での工具径補正(TRC)の直接選択または間接選択中に、アプローチブロックが生成されます。

TRCの選択時には、標準のアプローチ方法(TRCの間接選択)の代わりに、G138を使用したアプローチ方法(TRCの直接選択)を選択できます。

以下の図は、許可されている形状遷移に関して、可能性のあるすべてのアプローチブロックを示しています。関連する3つの形状遷移角度に対して、2つのNCブロックNC10およびNC20がそれぞれ示されています。

13.2.1.1 直接選択(G41/G42)

表 1: 形状遷移範囲0° < ß ≤ 180°

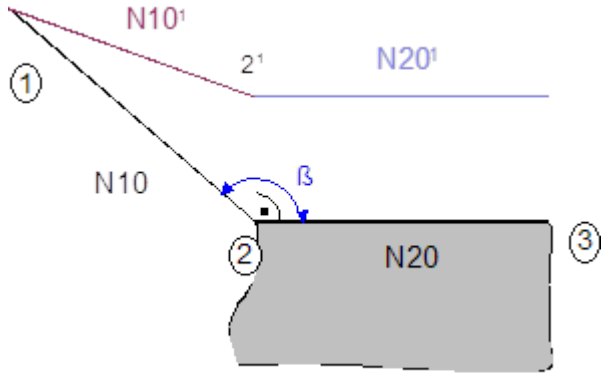
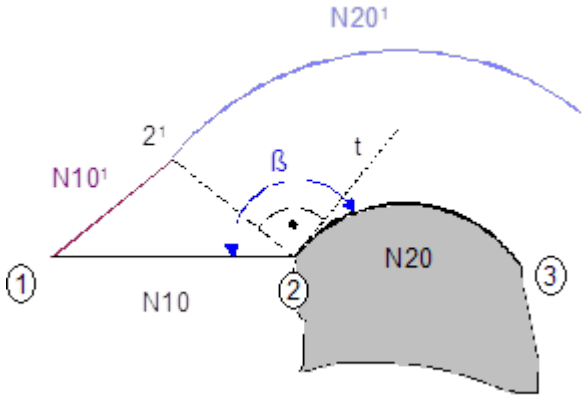
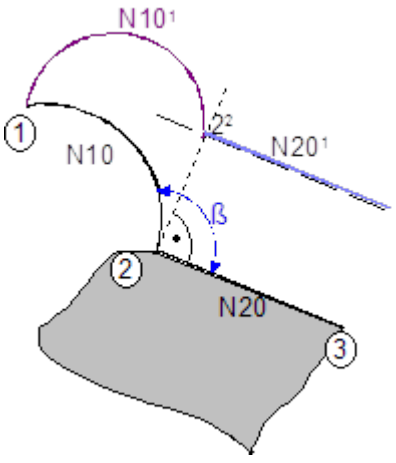
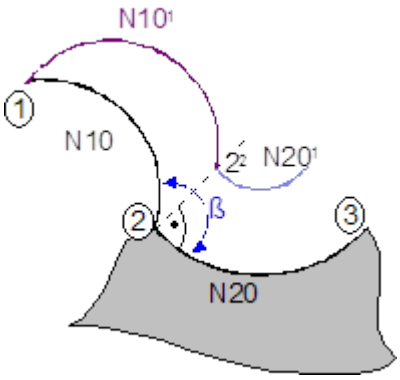
組み合わせタイプ1	組み合わせタイプ2
	
組み合わせタイプ3	**組み合わせタイプ4**
	

表 2: 形状遷移範囲180° < ß ≤ 270°

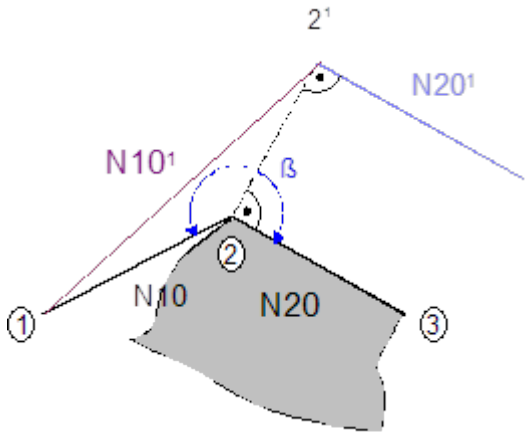
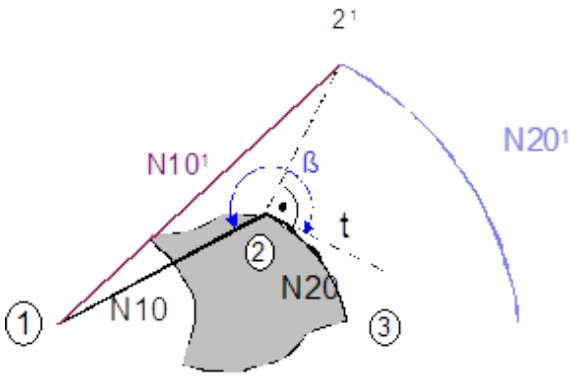
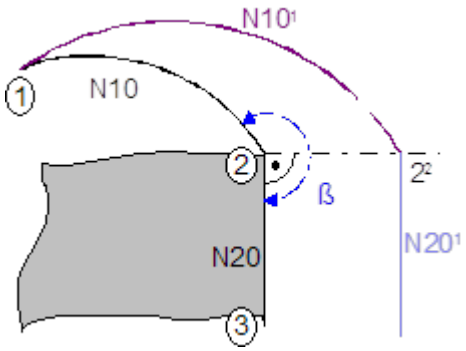
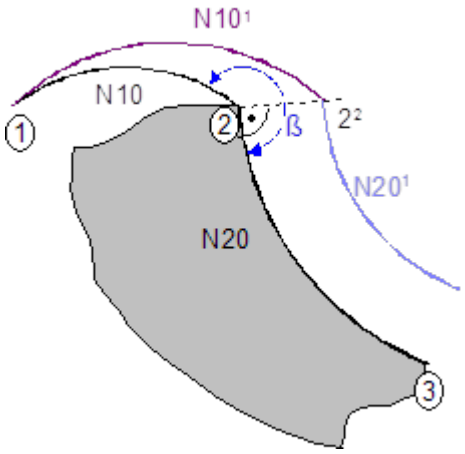
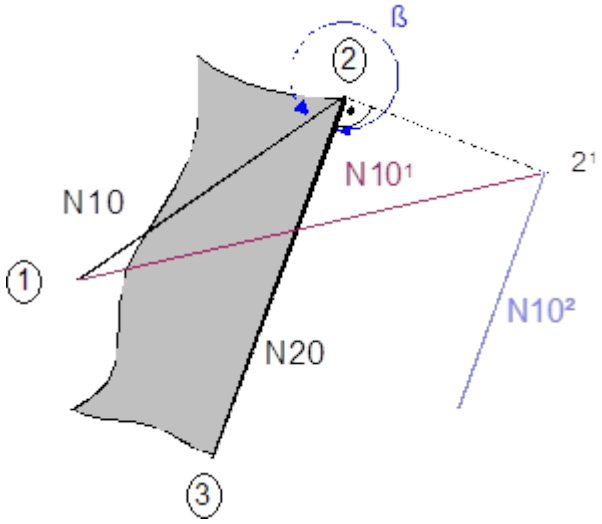
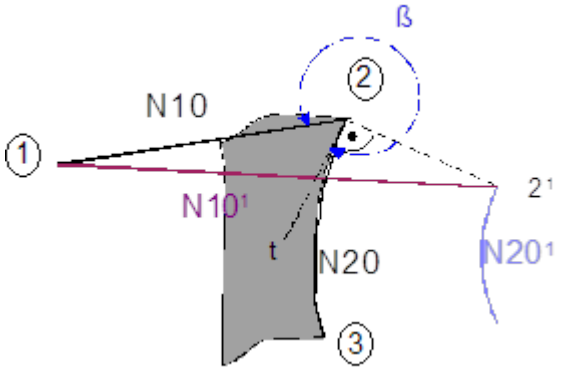
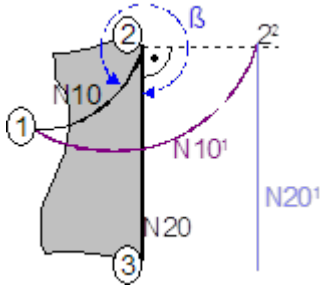
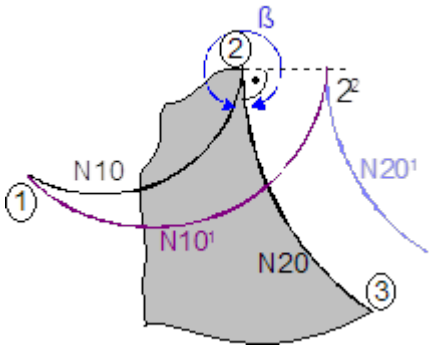
組み合わせタイプ1	組み合わせタイプ2
	
組み合わせタイプ3	**組み合わせタイプ4**
	

表 3: 形状遷移範囲270° < ß ≤ 360°

組み合わせタイプ1	組み合わせタイプ2
	
組み合わせタイプ3	組み合わせタイプ4
	

13.2.1.2 G25による間接選択(G41/G42)

表 4: 形状遷移範囲0° < ß ≤ 180°

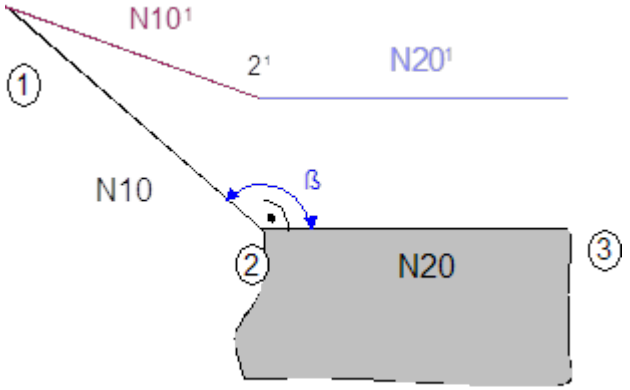
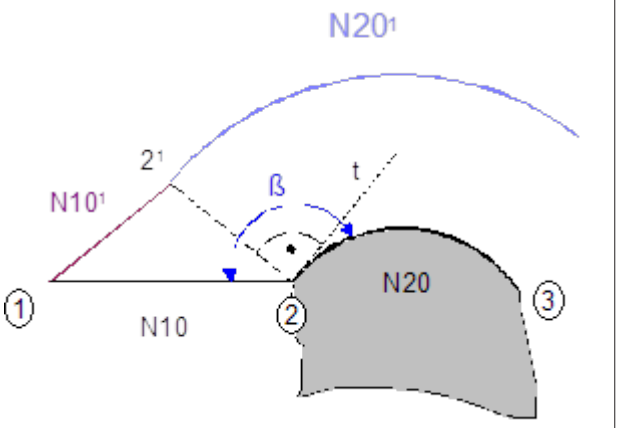
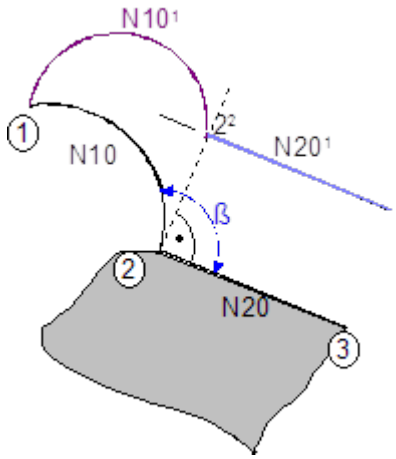
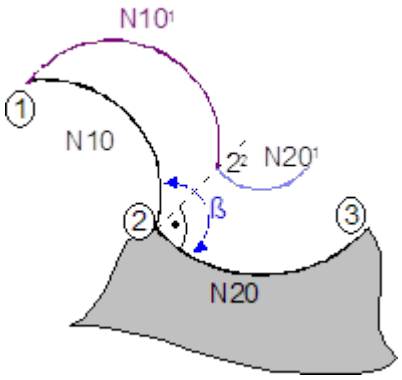
組み合わせタイプ1	組み合わせタイプ2
	
組み合わせタイプ3	**組み合わせタイプ4**
	

表 5: 形状遷移範囲180° < ß ≤ 270°

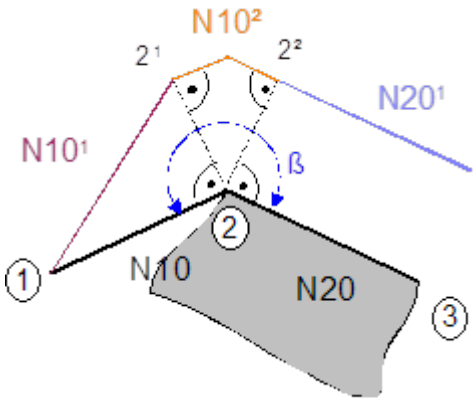
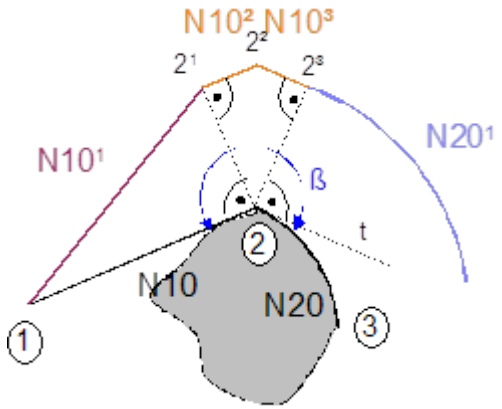
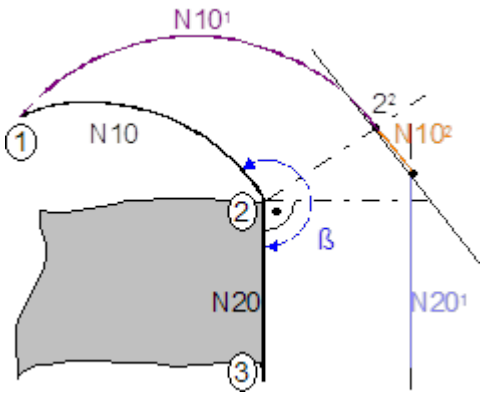
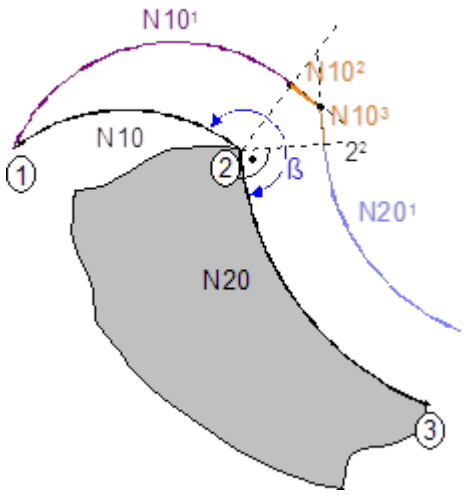
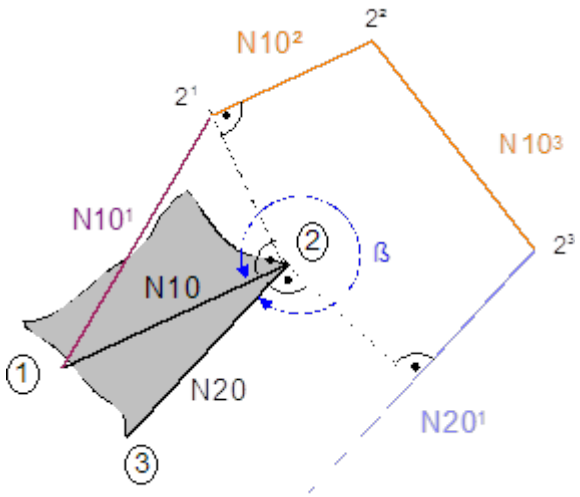
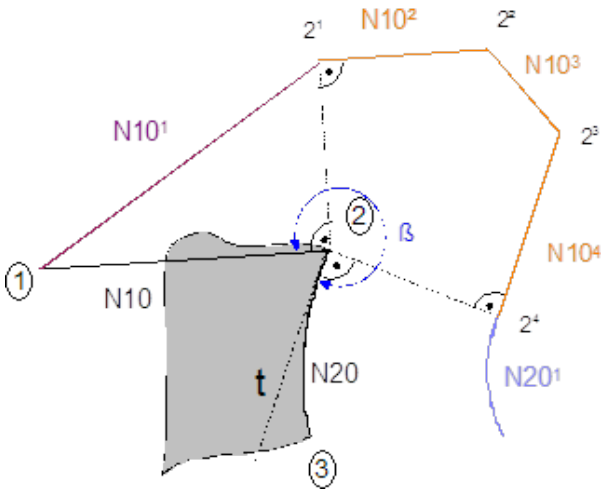
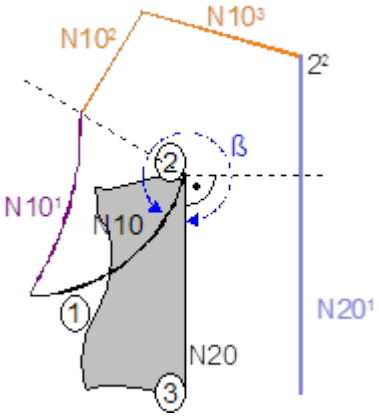
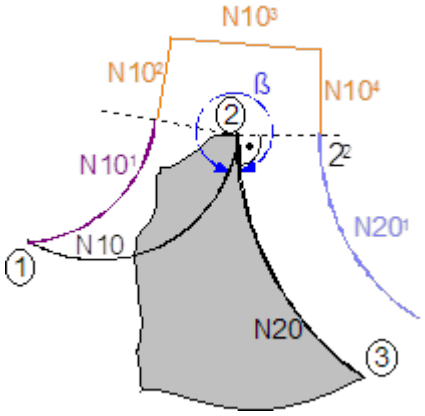
組み合わせタイプ1	組み合わせタイプ2
	
組み合わせタイプ3	組み合わせタイプ4
	

表 6: 形状遷移範囲270° < ß ≤ 360°

組み合わせタイプ1	組み合わせタイプ2
	
組み合わせタイプ3	**組み合わせタイプ4**
	

13.2.1.3 G26による間接選択(G41/G42)

表 7: 形状遷移範囲0° < ß ≤ 180°

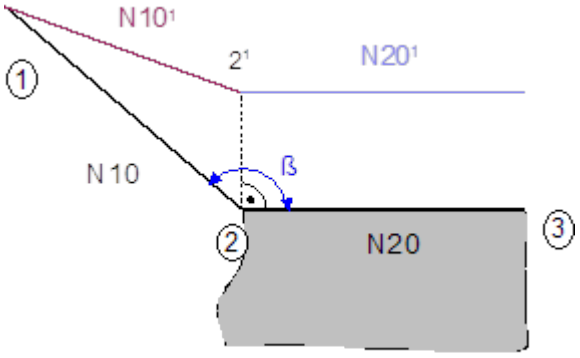
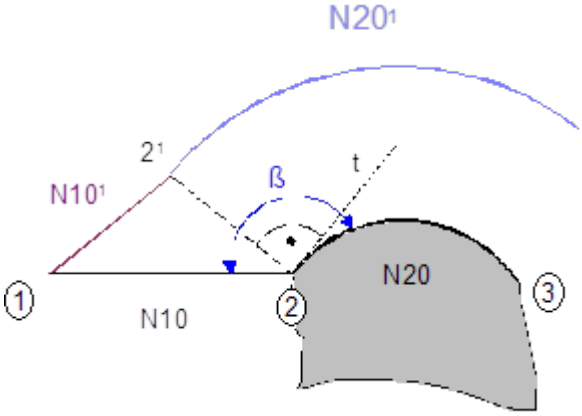
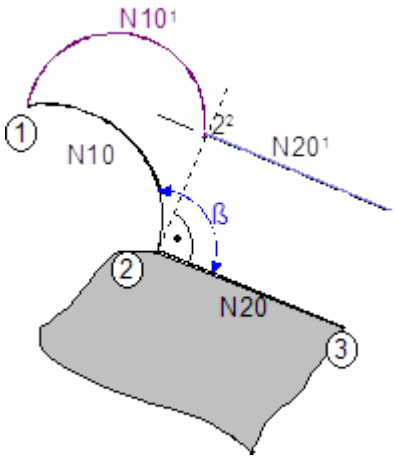
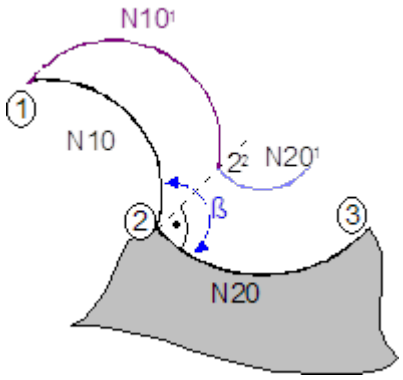
組み合わせタイプ1	組み合わせタイプ2
	
組み合わせタイプ3	**組み合わせタイプ4**
	

表 8: 形状遷移範囲180° < ß ≤ 270°

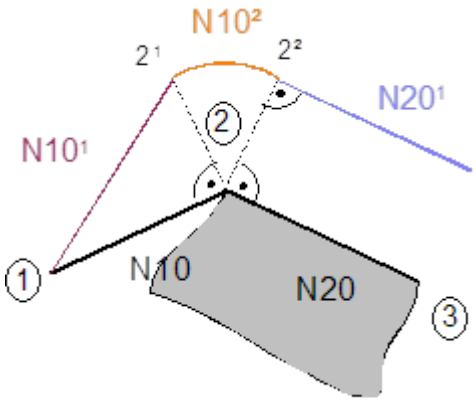
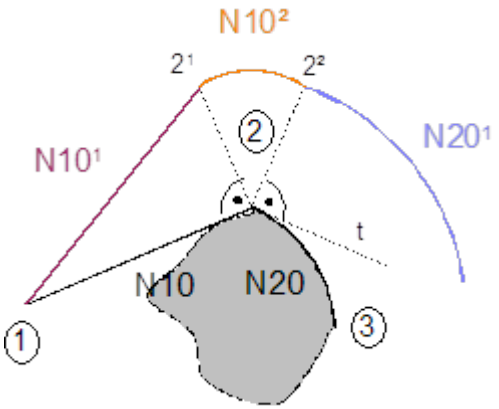
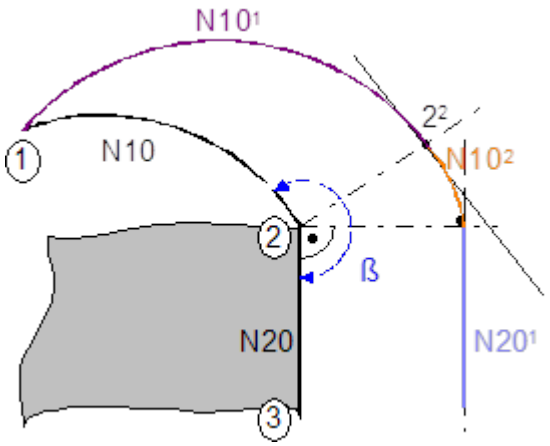
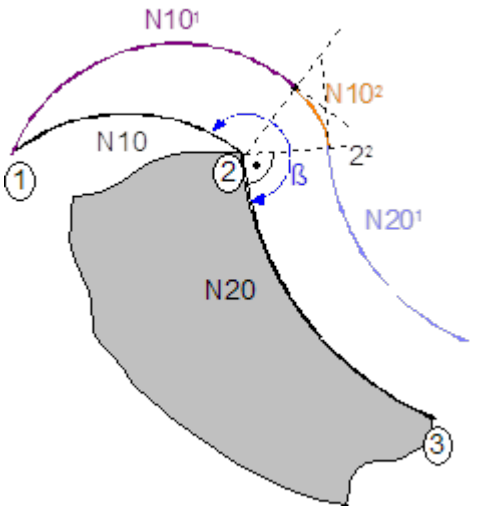
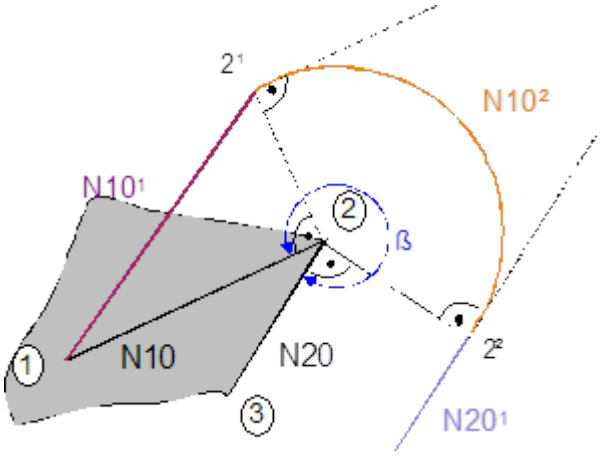
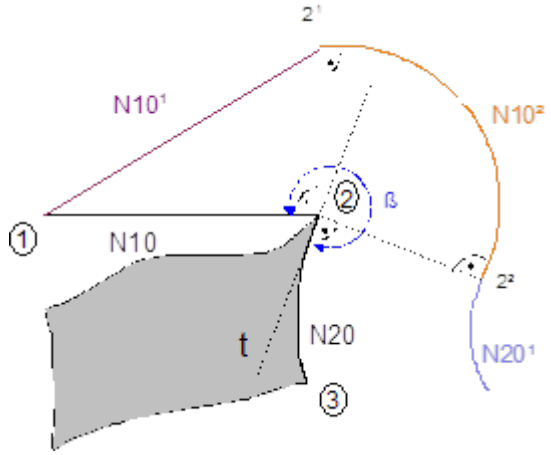
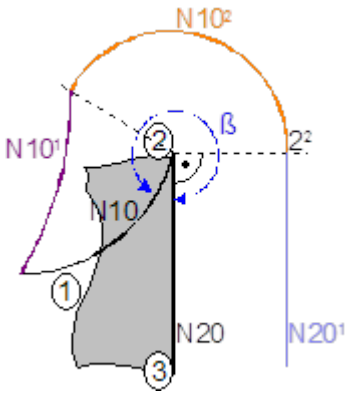
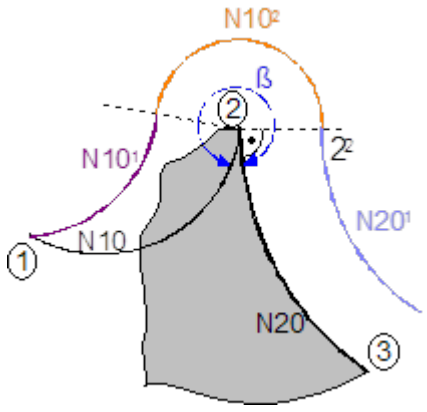
組み合わせタイプ1	組み合わせタイプ2
	
組み合わせタイプ3	組み合わせタイプ4
	

表 9: 形状遷移範囲270° < ß ≤ 360°

組み合わせタイプ1	組み合わせタイプ2
	
組み合わせタイプ3	組み合わせタイプ4
	

13.2.2 TRCの直接/間接選択解除(G40)

次の移動ブロックで、G40のプログラム後、対応する補正平面でのTRCの直接選択または間接選択中に、選択解除ブロックが生成されます。

TRCの選択解除時には、標準の選択解除方法(TRCの間接選択解除)の代わりに、G138を使用した選択解除方法(TRCの直接選択解除)を選択できます。

以下の図は、許可されている形状遷移に関して、可能性のあるすべての選択解除ブロックを示しています。関連する3つの形状遷移角度に対して、2つのNCブロックNC10およびNC20がそれぞれ示されています。

13.2.2.1 直接選択解除(G40)

表 10: 形状遷移範囲0° < ß ≤ 180°

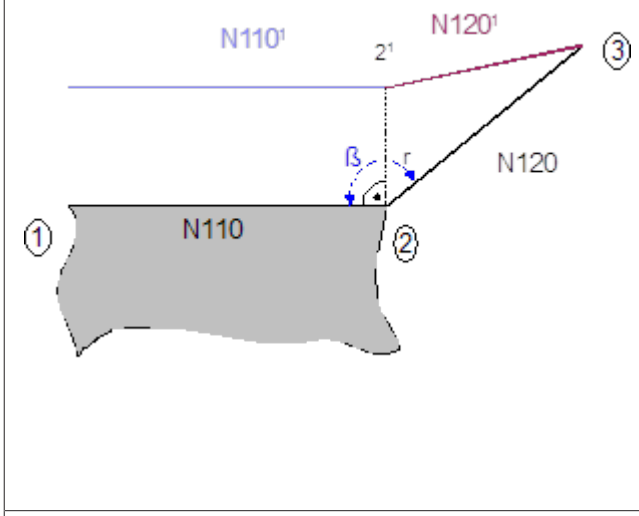
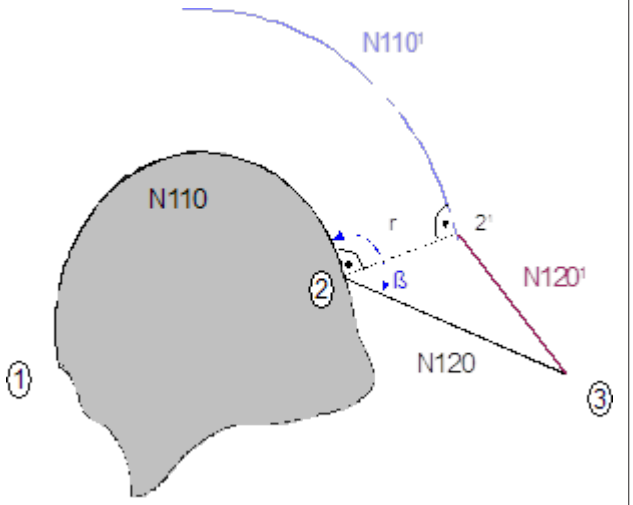
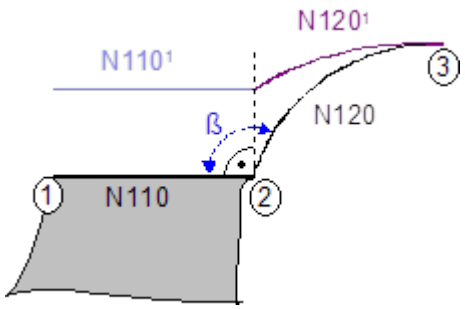
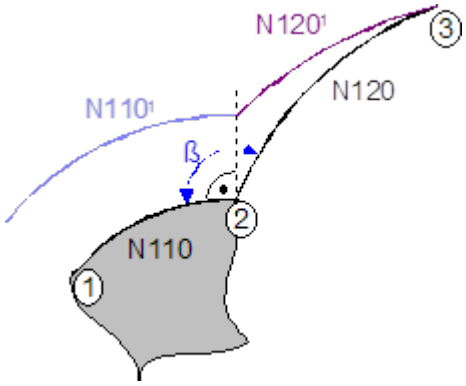
組み合わせタイプ1	組み合わせタイプ2
	
組み合わせタイプ3	組み合わせタイプ4
	

表 11: 形状遷移範囲180° < ß ≤ 270°

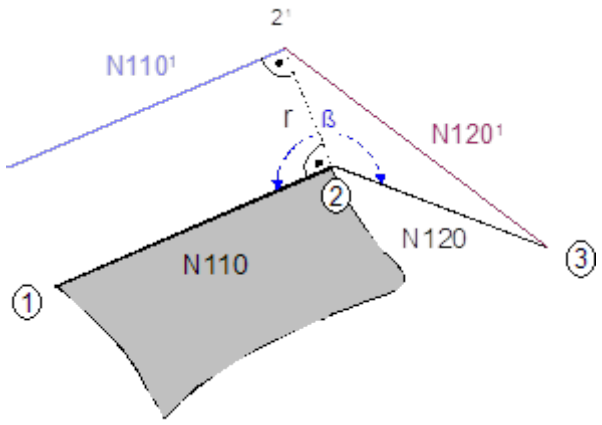
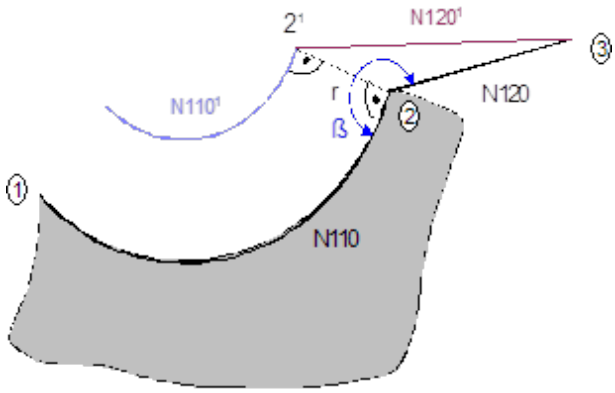
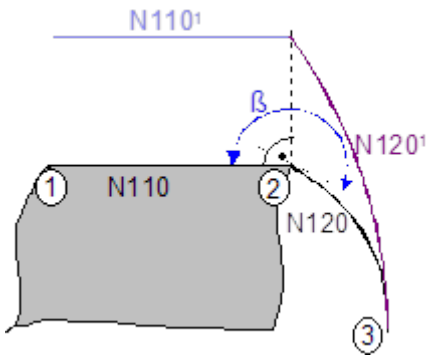
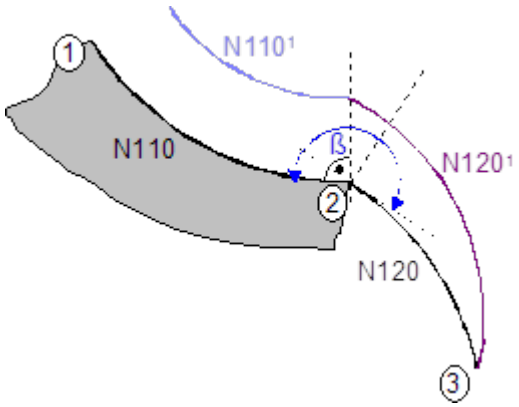
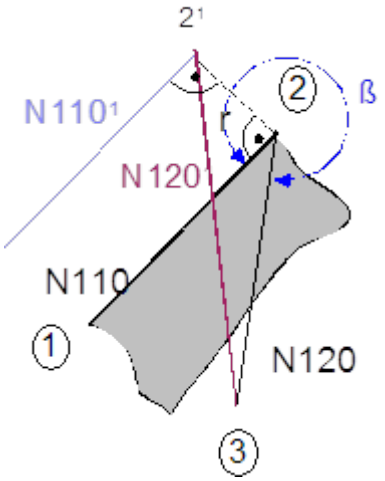
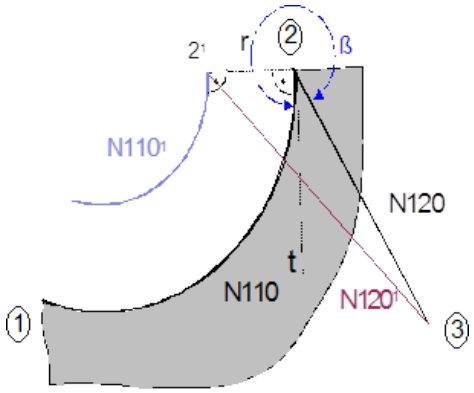
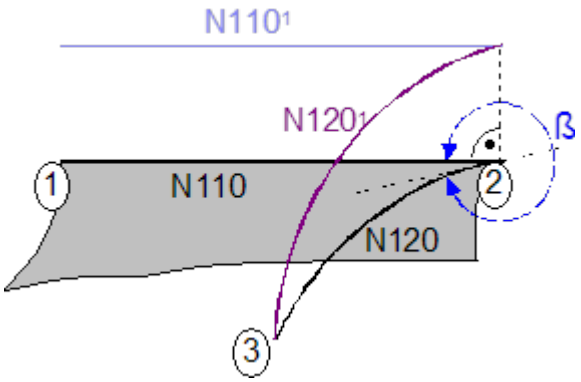
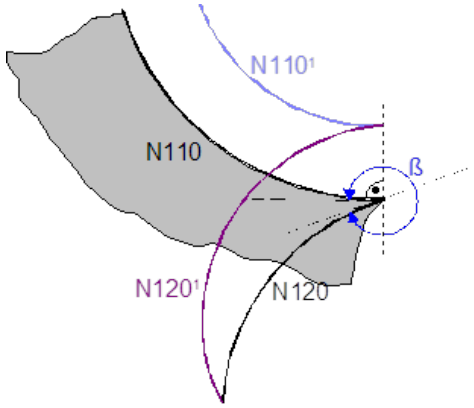
組み合わせタイプ1	組み合わせタイプ2
	
組み合わせタイプ3	**組み合わせタイプ4**
	

表 12: 形状遷移範囲270° < ß ≤ 360°

組み合わせタイプ1	組み合わせタイプ2
	
組み合わせタイプ3	組み合わせタイプ4
	

13.2.2.2 G25による間接選択解除(G40)

表 13: 形状遷移範囲0° < ß ≤ 180°

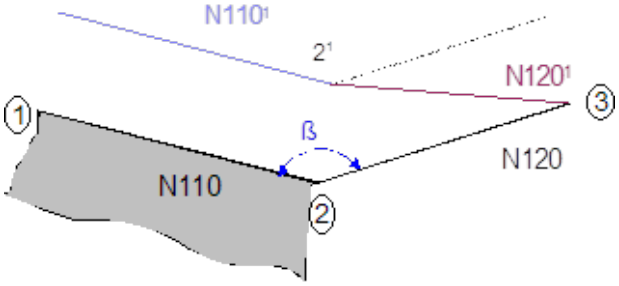
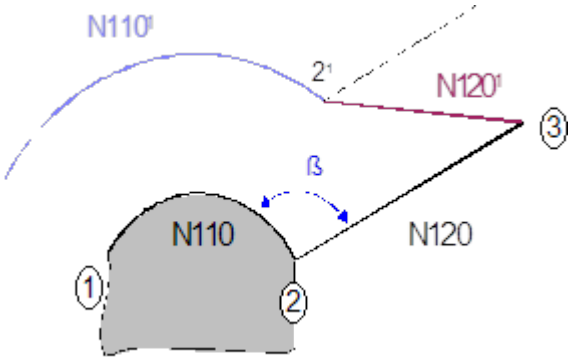
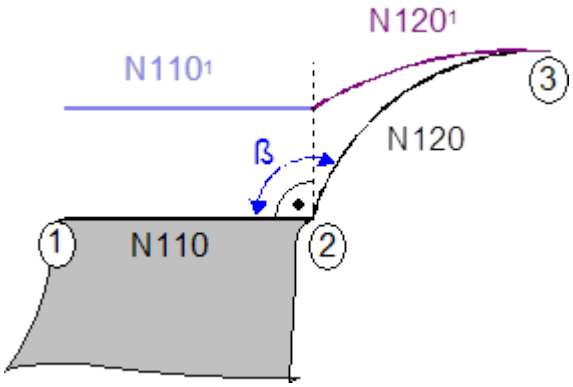
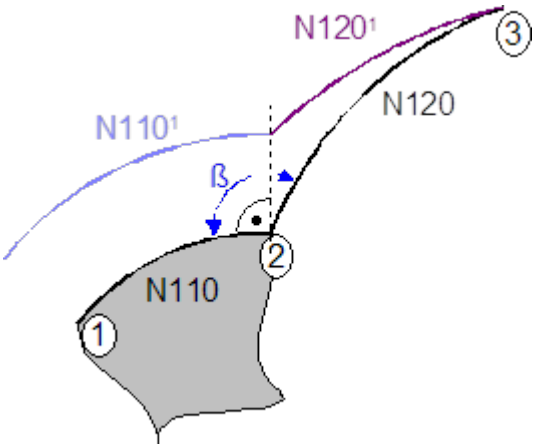
組み合わせタイプ1	組み合わせタイプ2
	
組み合わせタイプ3	組み合わせタイプ4
	

表 14: 形状遷移範囲180° < ß ≤ 270°

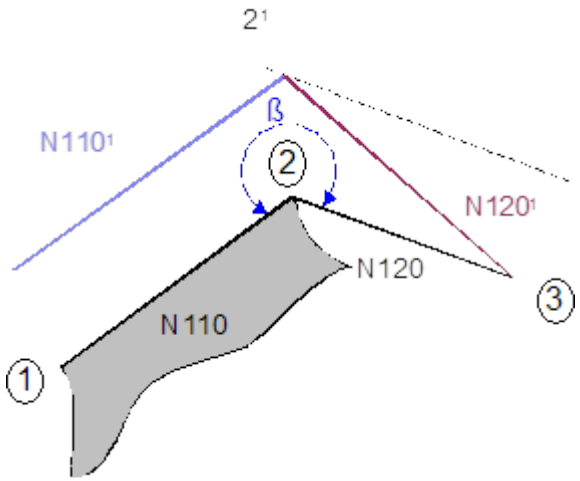
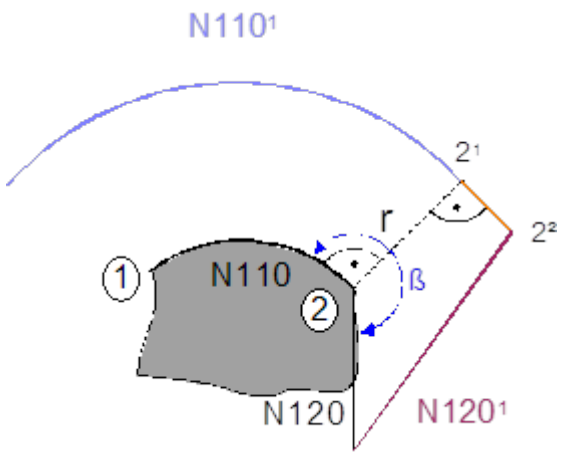
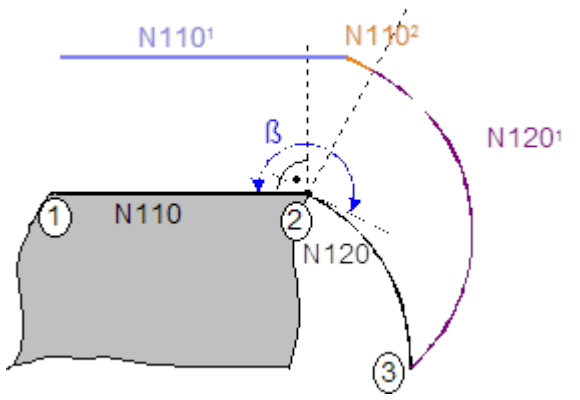
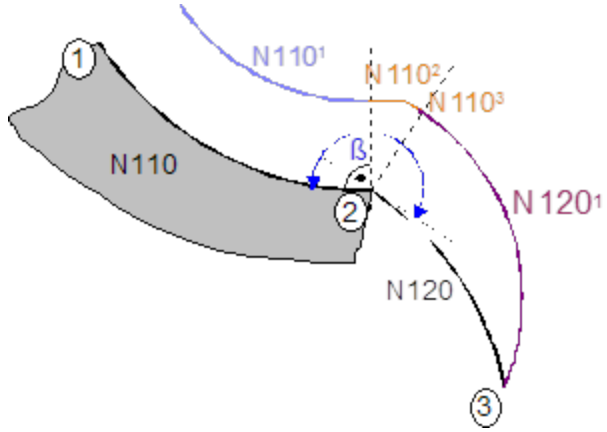
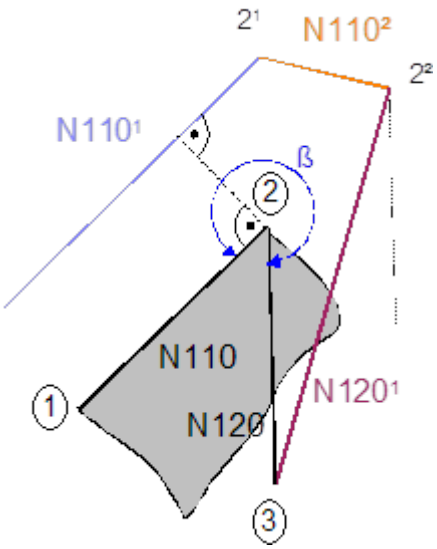
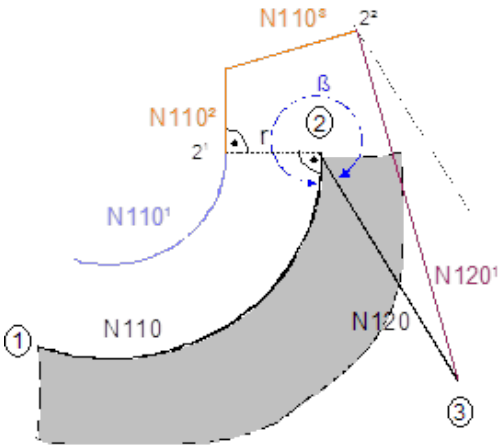
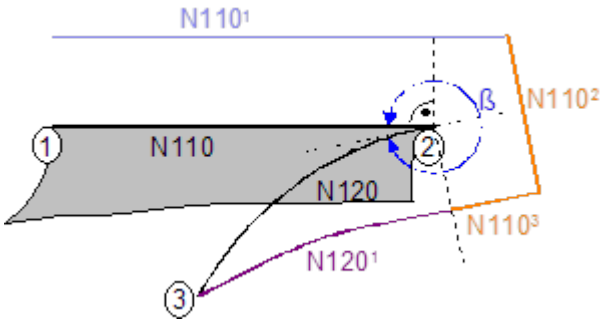
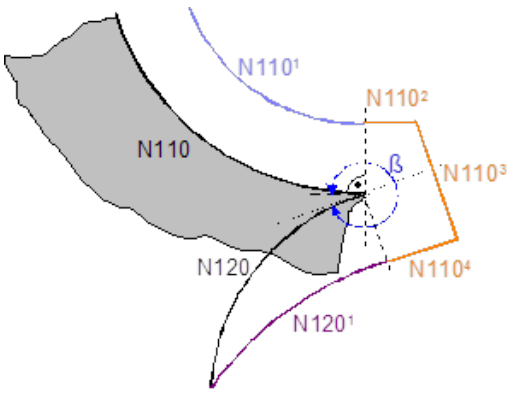
組み合わせタイプ1	組み合わせタイプ2
	
組み合わせタイプ3	組み合わせタイプ4
	

表 15: 形状遷移範囲270° < ß ≤ 360°

組み合わせタイプ1	組み合わせタイプ2
	
組み合わせタイプ3	**組み合わせタイプ4**
	

13.2.2.3 G26による間接選択解除(G40)

表 16: 形状遷移範囲0° < ß ≤ 180°

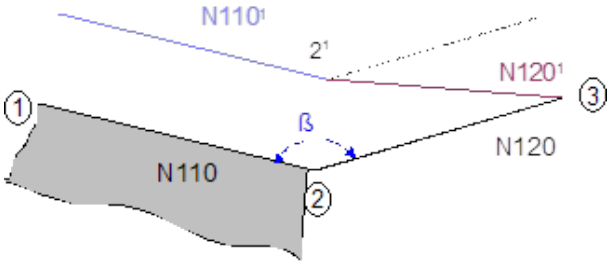
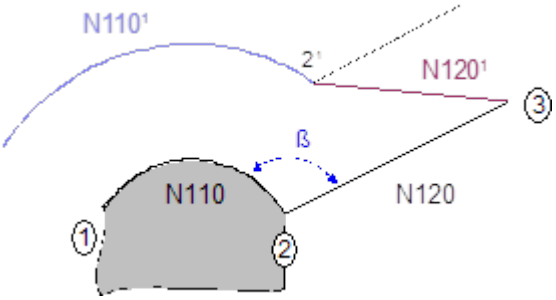
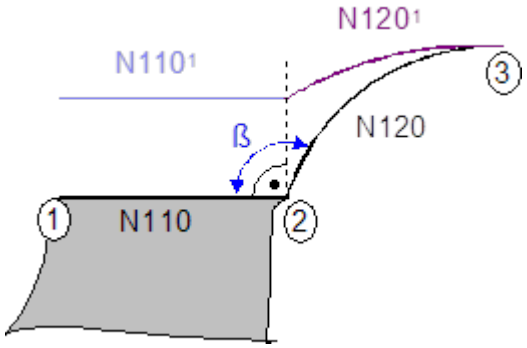
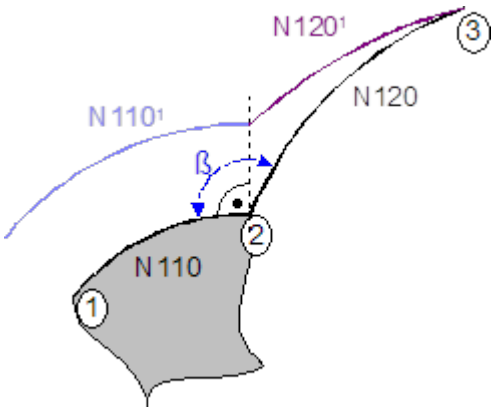
組み合わせタイプ1	組み合わせタイプ2
	
組み合わせタイプ3	**組み合わせタイプ4**
	

表 17: 形状遷移範囲180° < ß ≤ 270°

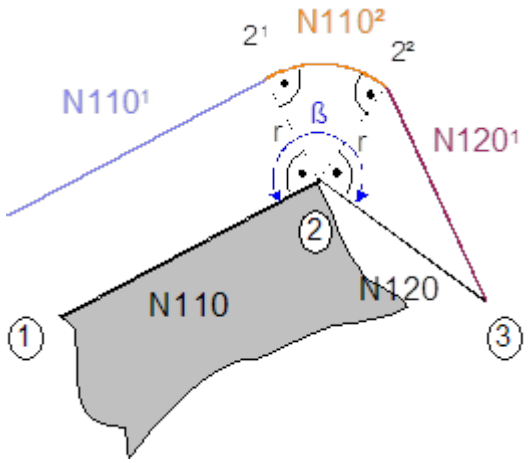
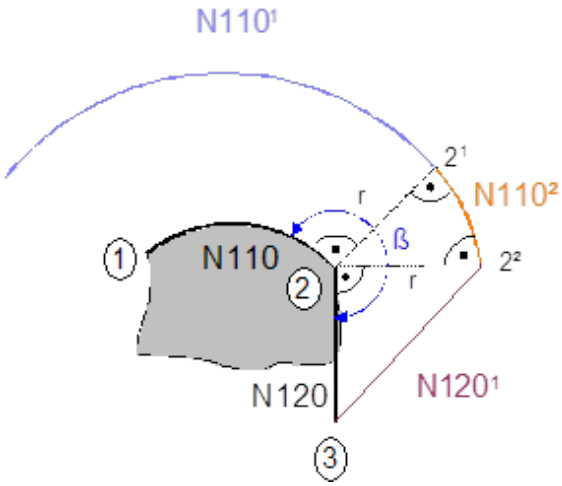
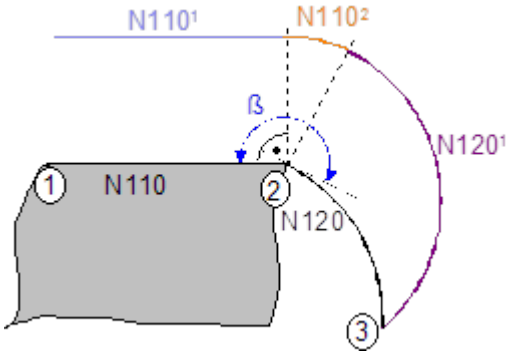
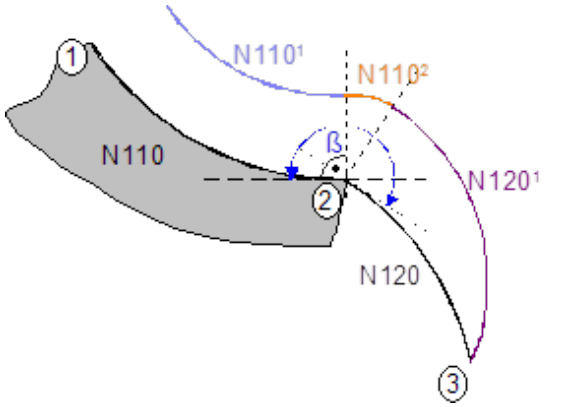
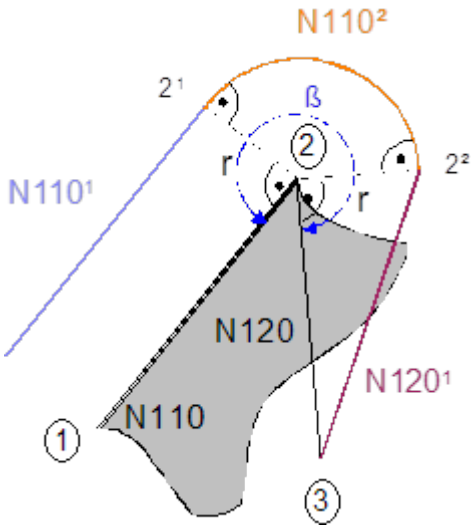
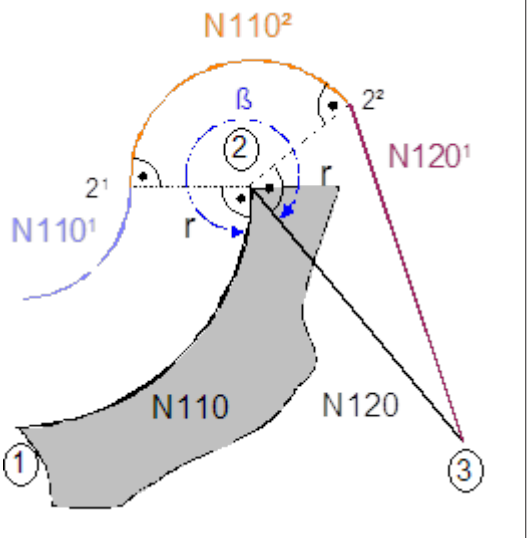
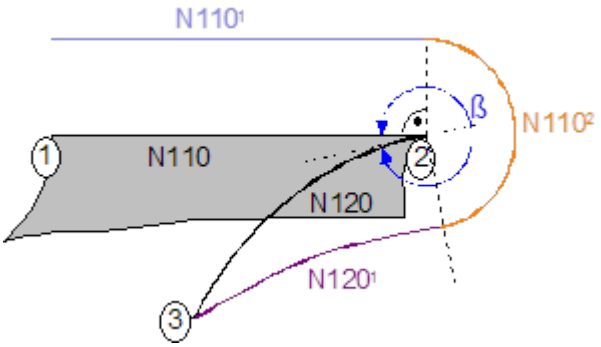
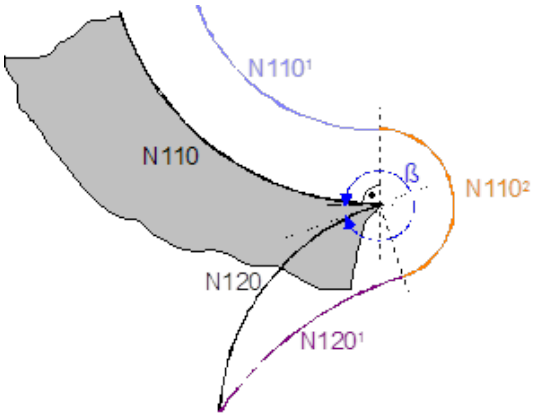
組み合わせタイプ1	組み合わせタイプ2
	
組み合わせタイプ3	**組み合わせタイプ4**
	

表 18: 形状遷移範囲270° < ß ≤ 360°

組み合わせタイプ1	組み合わせタイプ2
	
組み合わせタイプ3	組み合わせタイプ4
	

13.2.3 TRCの垂直選択および解除(G237)

直交モードの工具径補正を使用する場合は、直接または間接モードの場合と同様に、動作ブロックシーケンスに関する制限事項は存在しません。

個々の動作ブロックに関して工具径補正を有効にすることができます。

直交工具径補正を選択すると、プログラムされたパスに直交する直線動作ブロックが追加されます。このブロックは、補正されたパスとプログラムされたパス間の距離を生成します。このブロックの出力は、最初にプログラムされた動作ブロックの前に行われます。

直交工具径補正を選択解除すると、プログラムされたパスに直交する別の直線動作ブロックが追加されます。この動作ブロックは、補正されたパスとプログラムされたパス間の距離を無効にします。

プログラミング例

```
%bsp01.nc
N10 G00 X0 Y0 Z0 G17
N20 X10Y10
N30 V.G.WZ_AKT.R=5      (Tool radius)
N40 G237                (Activation of perpendicular selection)

; Corrected path
N50 G42                 (Selection of TRC on the right)
N60 G03 X60Y60J50F1000
N70 G01 Y100
N80 G03 X10Y150J50
N90 G40                 (Deselection of TRC)

; Presentation of the original contour
N100 G00 X0Y0
N110 X10Y10
N120 G03 X60Y60J50F1000
N130 G01 Y100
N140 G03 X10Y150J50
N150 G00 X0Y0
N999 M30
```

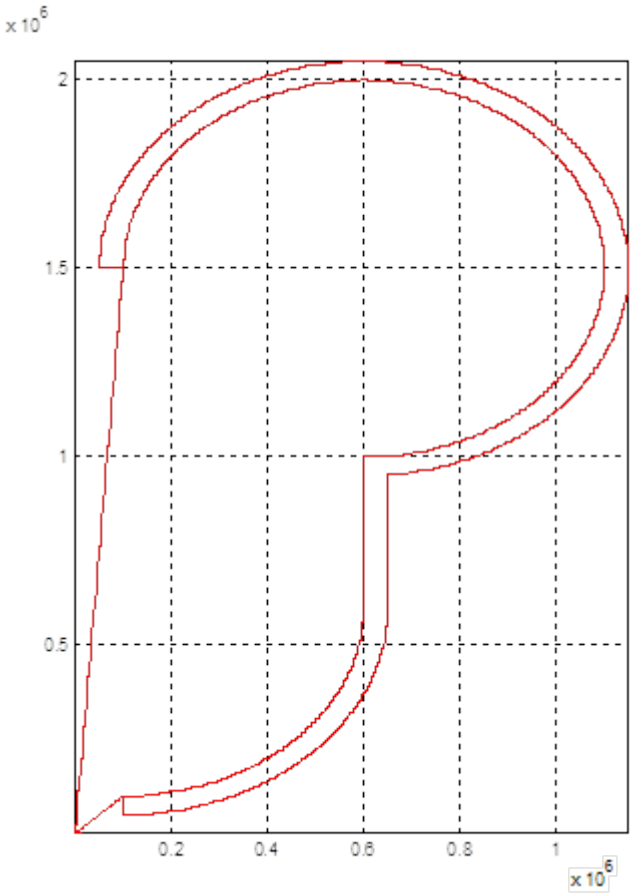


図 88: 図12-3: プログラミング例bsp01.nc

13.2.3.1 直交選択/選択解除での技術ファンクション

直交選択または選択解除では、NCプログラム内での技術ファンクションの配置は、出力時間を決定するため、特別な重要性を持ちます。

直交選択では、選択ブロック、技術ブロック、および最初の動作ブロックの配置は、出力時間の決定的要因です。

直交選択解除では、技術ブロックおよび最後の補正動作ブロックの配置は、出力時間の決定的要因です。

以下の例と対応する図は、この関係を示します。

プログラミング例

```
%bsp02.nc


N10 G00 X0 Y0 Z0 G17
N20 F9000
N30 V.G.WZ_AKT.R=5      (Tool radius)
N40 G237                (Activation of perpendicular selection)
N50 G91                 (Relative programming)
N60 G01 X30 Y10


; Corrected path
N50 G41 M7 X20 Y70      (Selection of TRC on the left)
N55 M8
N60 G03 X60 I30
N70 G01 X30
N80 X25 Y-20
N85 M9
N90 G40                 (Deselection of TRC)
N100 G90 X200 Y0        (Absolute programming)
N110 X0


; Presentation of the original contour
N200 G91                (Relative programming)
N210 G01 X30 Y10
N220 X20 Y70
N230 G03 X60 I30
N240 G01 X30
N250 X25 Y-20
N260 G90 X200 Y0        (Absolute programming)
N270 G00 X0
N999 M30
```

下の図の技術ファンクションM9の出力は、直交選択解除ブロックの出力のすぐ前に行われます。技術ファンクションは、最後の補正動作ブロックと、工具径補正の直交選択解除の間に配置する必要があります。

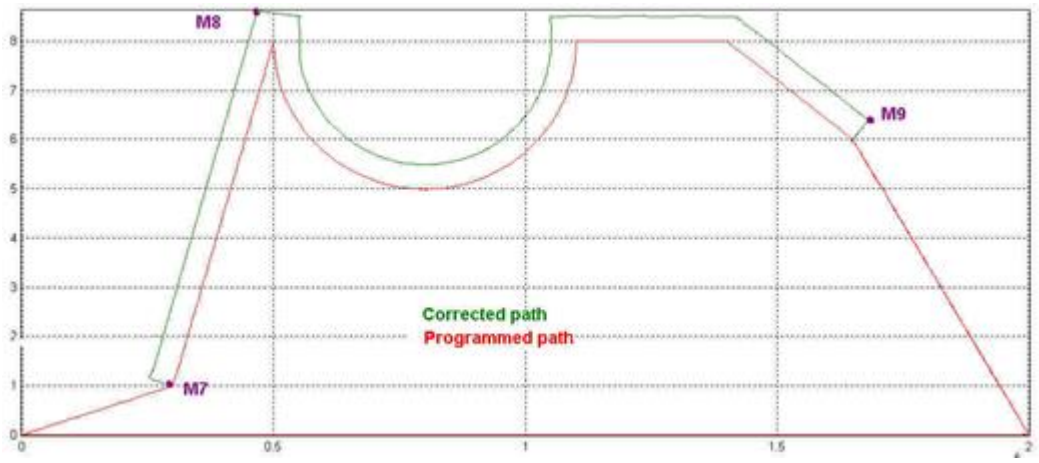


図 89: 図12-4: プログラミング例bsp02.nc

プログラミング例

```
%bsp03.nc

N10 G00 X0 Y0 Z0 G17
N20 F9000
N30 V.G.WZ_AKT.R=5      (Tool radius)
N40 G237                (Activation of perpendicular selection)
N50 G91                 (Relative programming)
N60 G01 X30 Y10

; Corrected path
N50 G41 M7              (Selection of TRC on the left)
N55 M8 X20 Y70
N60 M9 G03 X60 I30
N70 G01 X30
N80 X25 Y-20
N90 G40                 (Deselection of TRC)
N100 G90 X200 Y0        (Absolute programming)
N110 X0

; Presentation of the original contour
N200 G91                (Relative programming)
N210 G01 X30 Y10
N220 X20 Y70
N230 G03 X60 I30
N240 G01 X30
N250 X25 Y-20
N260 G90 X200 Y0        (Absolute programming)
N270 G00 X0
N999 M30
```

技術ファンクションM8は、選択ブロックの後に実行されます。これを達成するには、技術ファンクションを選択ブロックとは別のブロックでプログラムする必要があります。さらに、選択ブロックと技術ファンクションの間に動作ブロックをプログラムすることはできません。

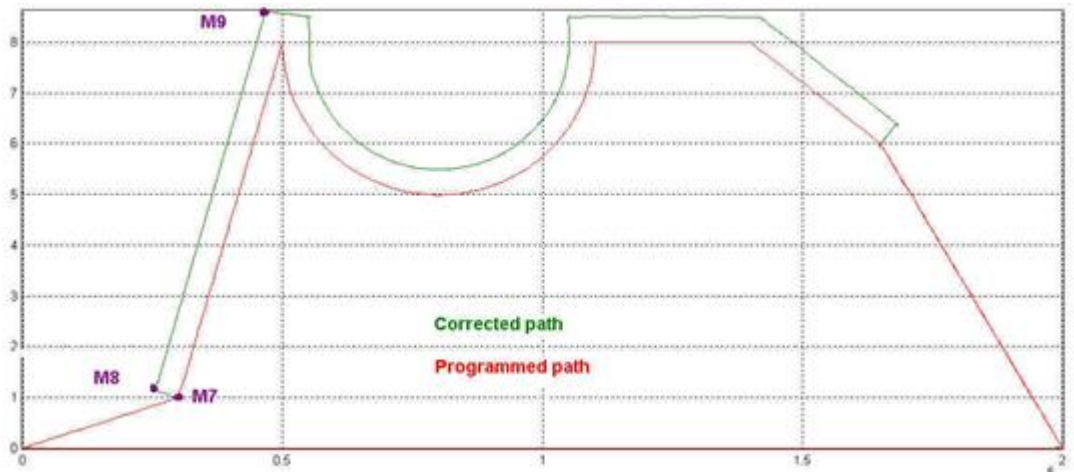


図 90: 図12-5: プログラミング例bsp03.nc

プログラミング例

```
%bsp04.nc

N10 G00 X0 Y0 Z0 G17
N20 F9000
N30 V.G.WZ_AKT.R=5      (Tool radius)
N40 G237                (Activation of perpendicular selection)
N50 G91                 (Relative programming)
N60 G01 X30 Y10

; Corrected path
N50 G41                 (Selection of TRC on the left)
N51 M7
N52 M8
N53 M8
N55 X20 Y70
N60 G03 X60 I30
N70 G01 X30
N80 X25 Y-20
N90 G40                 (Deselection of TRC)
N100 G90 X200 Y0        (Absolute programming)
N110 X0

; Presentation of the original contour
N200 G91                (Relative programming)
N210 G01 X30 Y10
N220 X20 Y70
N230 G03 X60 I30
N240 G01 X30
N250 X25 Y-20
N260 G90 X200 Y0        (Absolute programming)
N270 G00 X0
N999 M30
```

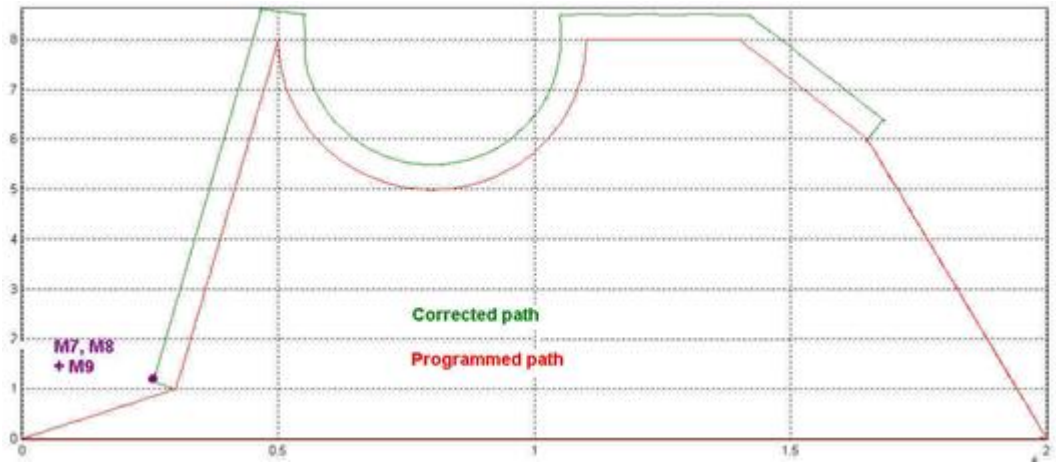


図 91: 図12-6: プログラミング例bsp04.nc

13.2.3.2 単一ブロック内の技術ファンクション

プログラミング例

```
%bsp05.nc

N10 G00 X0 Y0 Z0 G17
N20 F9000
N30 V.G.WZ_AKT.R=5      (Tool radius)
N40 G237                (Activation of perpendicular selection)
N60 G01 X10 Y10

; Corrected path
N50 G41 M7              (Selection of TRC on the left)
N55 M8 X70 Y30
N60 M7
N90 G40                 (Deselection of TRC)
N100 G90 X100 Y0
N110 X0

; Presentation of the original contour
N210 G01 X10 Y10
N220 X70 Y30
N240  X100 Y0
N270 G00 X0
N999 M30
```

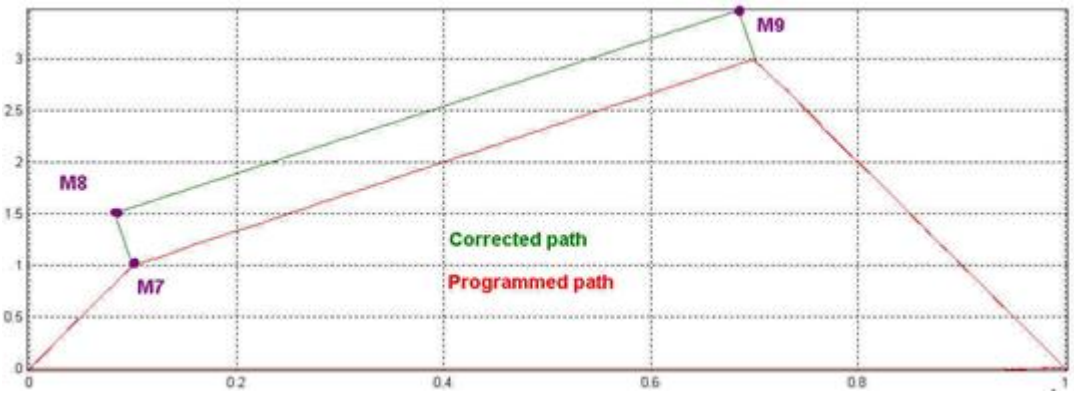


図 92: 図12-7: プログラミング例bsp05.nc

13.2.4 TRCの内角選択(G238)

内角選択を使用することは、閉じた形状の内角でTRCを選択することを意味します。

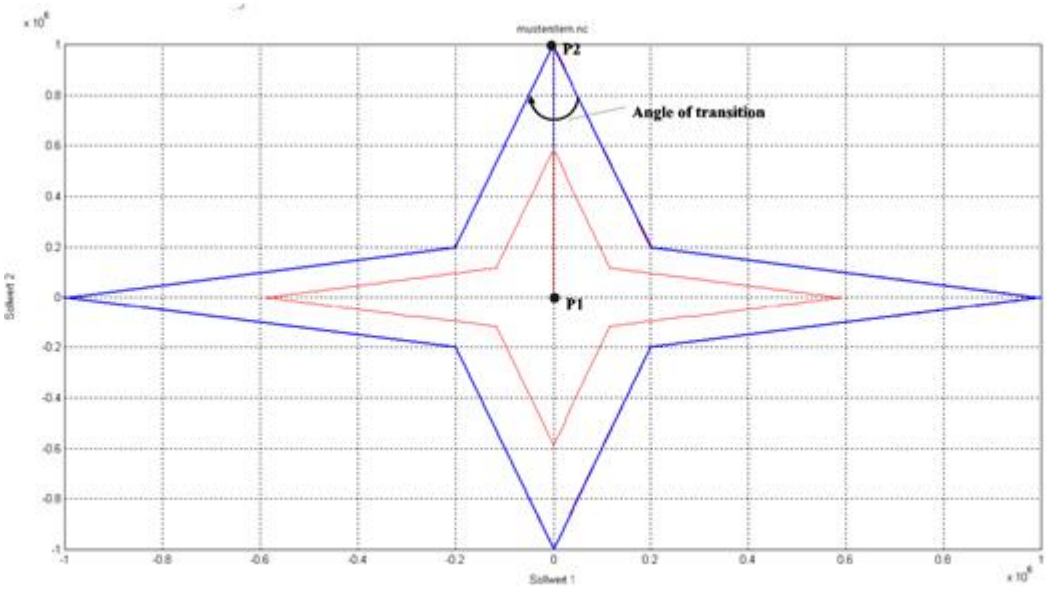
内角選択に関しては、以下の指定が使用されます。

- **選択ブロック**: 選択点で開始される直線動作ブロック
- **最初の動作ブロック**: 選択ブロックの終了で開始される動作ブロック、このブロックはTRC選択後の2番目の動作ブロックでもあります
- **最後の動作ブロック**: 最後の動作ブロックは、TRCの選択解除が現れる前の動作ブロックです

プログラミング例

スター型のような形状では、内角の先の尖った部分でTRCを選択する必要があります。

```
%musterstern.nc

N1 G74 X1Y2Z3
N2 G17 G00 X0Y0Z0 G90
N4 F10000

(Display of the contour)
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

(Display of the path of tool)
N210 G238          (Activation of inner corner selection)
N220 V.G.WZR = 10  (Determine the radius of tool)
N230 G41      (Selection of TRC at the left hand side of the contour)

N240 G01 X0 Y100   (Selection block)

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           (Deselection of TRC)

N400 G00 X0Y0
N999 M30
```



注釈

選択G41またはG42のコマンドは点P1で実行され、点P2は最初の直線ブロックの終点です。この最初の直線ブロックは選択ブロックです。この点は、閉じた形状を得るために、選択解除の前に最後の動作ブロックの終点もシミュレートする必要があります。

外側に(青色)表示されたプロセスはプログラムされた形状で、内側に(赤色)表示されたプロセスは工具のパスです。

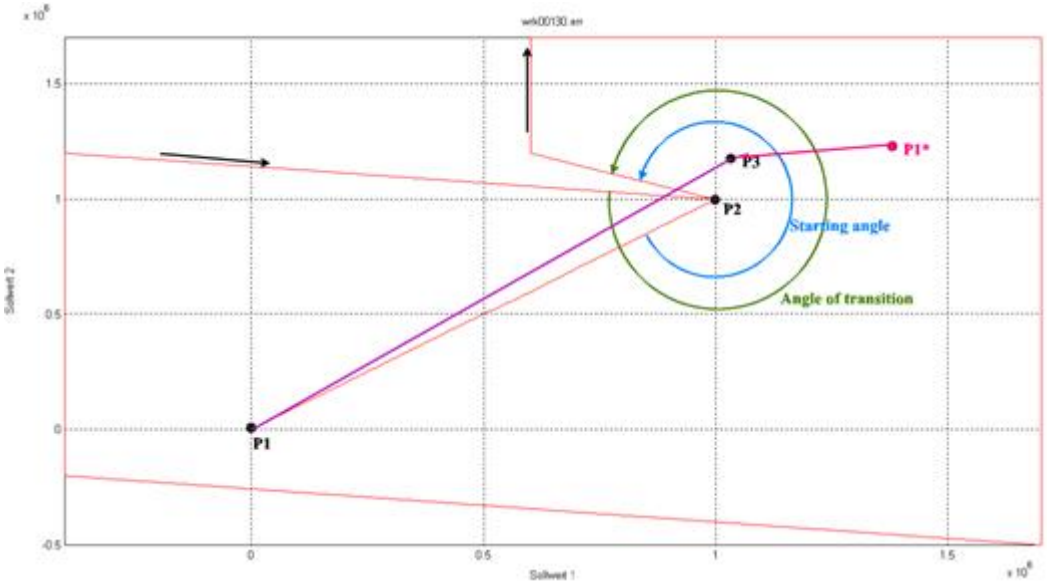
注意:

TRCの選択解除のプログラムされたG40の後、次の動作ブロックによる形状損傷に関するモニタリングは行われません。

13.2.4.1 内角選択の制限事項

- プログラムされた形状を閉じる必要があります。
- 形状の最後の動作ブロックは一周円であってはいけません。
- 形状の最初の動作ブロックは、直線動作ブロックであるとともに、TRCの選択解除後の動作ブロックであることが必要です。
- 点P1は、形状のすべての工レメントに対して、少なくとも工具径の距離を持つ必要があります。
- 選択ブロックは、最初の動作ブロックおよび最後の動作ブロックを除いて、形状工レメントを横断できません。
- TRCが有効な場合にTRCの選択側を変更することは許可されません。
- 内角選択が有効な場合に工具径を変更することは許可されません。
- TRCの選択側は、選択の位置から、工具がプログラムされた形状を横断しないような方法で選択する必要があります。
- TRCの選択後の最初の動作ブロックと2番目の動作ブロックの囲まれた角度は、180°を超えることはできません。

角度の制限の理由:



グラフィックでは、形状の右側でTRCを選択する必要があります。形状の最後の動作ブロックと最初の動作ブロック間の角度が約180°の場合、等距離パスの最初の点はTRCの直接メソッドによって決定されます。この図では、等距離パスの最初の点はP3です。この図は、点P1と点P3間の直線が赤く描かれた形状が損傷していることを示します。

この図で開始角度として指定された角度は、最大180°にすることができます。この範囲の角度では、形状は損なわれません。表示されている角度は180°よりも大きな角度です。代替点(選択P1*)は、形状の損傷を回避します。

注記

制限事項の1つに違反すると、対応するエラーが作成されます。

13.2.5 ブロックを使用しないTRCの選択/解除(G239)

G239では、G138の場合と同様に、TRCスタートアップ動作が直接に実行されます。

TRCの選択解除(G40)では、最後の等距離点が最新の現在位置になります。工具径距離を減らす次の動作ブロックは不要です。

注記

G40後すぐにプログラムされた円弧ブロックは、この場合、実際の選択解除位置と円弧のプログラムされた開始点は同一でないため、エラーメッセージを生成します!

プログラミング例

下の例では、プログラムされたパスは黒で表示され、等距離パスは青で表示されます。

例1:

両方の主軸でTRC選択解除を行った後、位置がプログラムされ、正確に実行されます。

```
%G239_Demo1.nc
N010 G0 X0 Y0 Z0 F1000
N020 G239               (TRC mode)
N030 V.G.WZ_AKT.R = 10
N040 G41 G01 X30 Y20    (TRC selection)
N050 G01 X60
N060 G01 X90
N070 G01 X120
N080 G40                (TRC deselection without motion)

N100 G00 X150 Y0
N110 M30
```

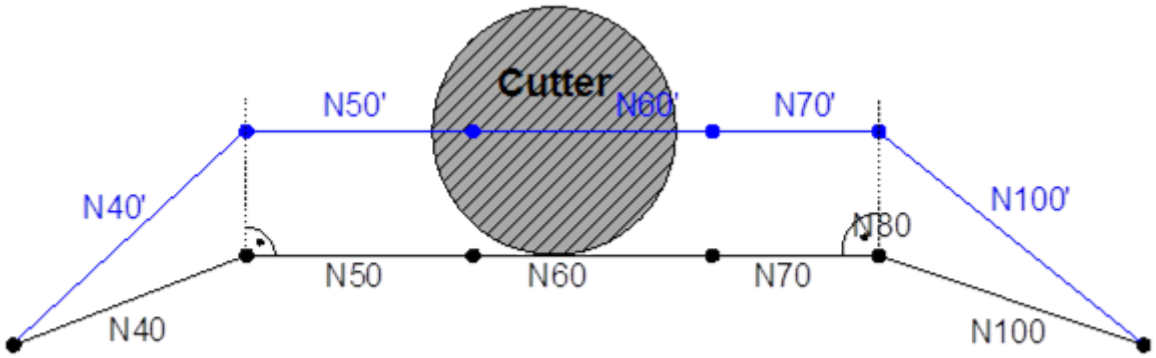


図12-8: G239_Demo1.ncの動作ブロックシーケンス

例2:

ここでは、TRCを選択解除した後、1つの主軸のみがプログラムされます。2番目の座標の位置は同じままです。

```
%G239_Demo2.nc
N010 G0 X0 Y0 Z0 F1000
N020 G239               (TRC mode)
N030 V.G.WZ_AKT.R = 10
N040 G41 G01 X30 Y20    (TRC selection)
N050 G01 X60
N060 G01 X90
N070 G01 X120
N080 G40                (TRC deselection without motion)

N090 G00 X150
N100 G00 X200

N110 M30
```

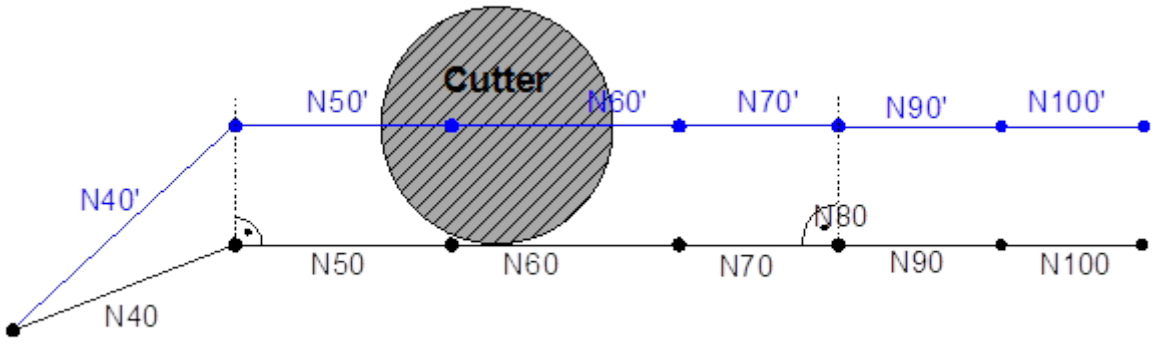


図12-9: G239_Demo2.ncの動作ブロックシーケンス

例3:

G40の後すぐに再度TRCを選択すると、2つの主軸の1つのみがプログラムされます。

```
%G239_Demo3.nc
N010 G0 X0 Y0 Z0 F1000
N020 G239               (TRC mode)
N030 V.G.WZ_AKT.R = 10
N040 G41 G01 X30 Y20    (TRC selection)
N050 G01 X60
N060 G01 X90
N070 G01 X120
N080 G40                (TRC deselection without motion)
( No programming of Y again )
N090 G41 G01 X150       (TRC re-selection)
N100 G00 X200

N110 M30
```

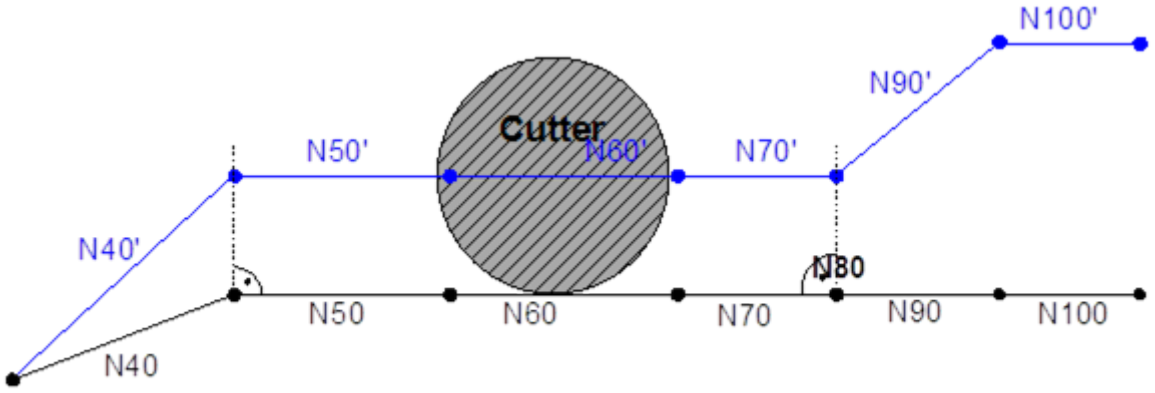


図12-10: G239_Demo3.ncの動作ブロックシーケンス

例4:

ここでは、TRCを再選択した後、2番目の動作ブロックでのみ、2番目の主軸がプログラムされます。

```
%G239_Demo4.nc
N010 G0 X0 Y0 Z0 F1000
N020 G239               (TRC mode)
N030 V.G.WZ_AKT.R = 10
N040 G41 G01 X30 Y20    (TRC selection)
N050 G01 X60
N060 G01 X90
N070 G01 X120
N080 G40                (TRC deselection without motion)

N090 G41 G01 X150       (TRC re-selection)
N100 G00 X200 Y20

N110 M30
```

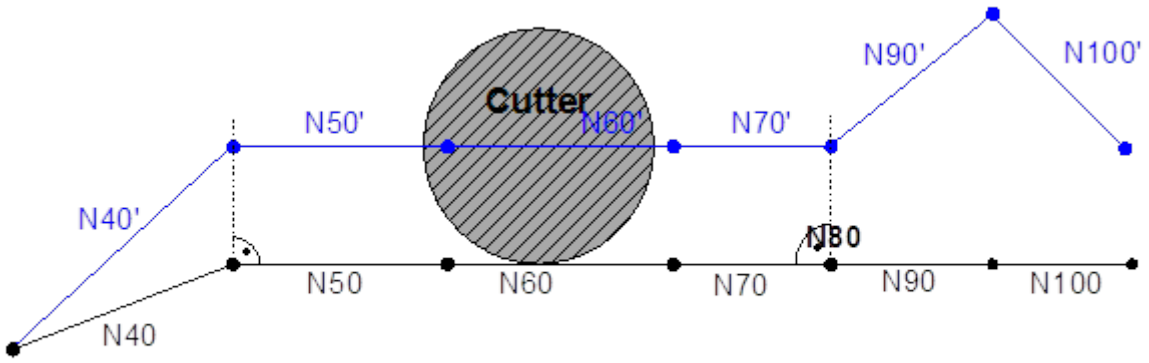


図12-11: G239_Demo4.ncの動作ブロックシーケンス

例5:

この例では、G40後にTRCを再選択した後、最初の選択(G41)の場合と同様に、2番目の主軸の位置がプログラムされます。

```
%G239_Demo5.nc
N010 G0 X0 Y0 Z0 F1000
N020 G239               (TRC mode)
N030 V.G.WZ_AKT.R = 10
N040 G41 G01 X30 Y20    (TRC selection)
N050 G01 X60
N060 G01 X90
N070 G01 X120
N080 G40                (TRC deselection without motion)

N090 G41 G01 X150 Y20   (TRC re-selection)
N100 G00 X200

N110 G40
N120 M30
```

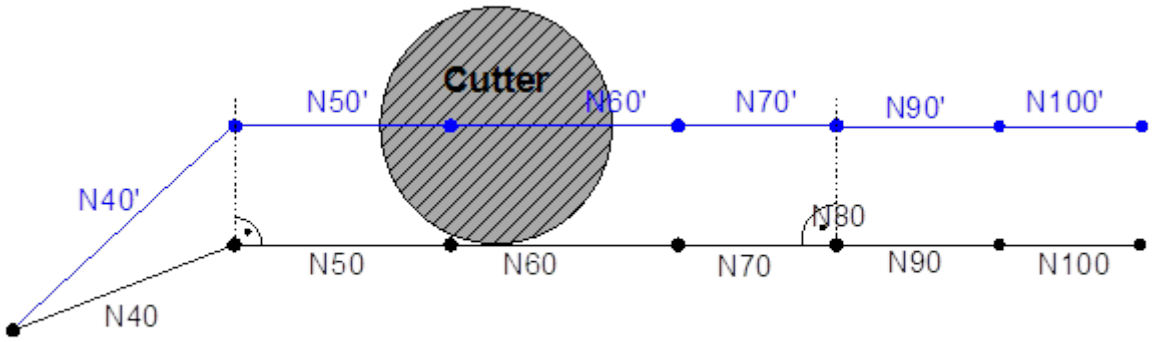


図12-12: G239_Demo5.ncの動作ブロックシーケンス

例6:

TRC選択解除後にプログラムされた円弧ブロックが、指定された中心点(M)が(新しい)起点(S)および(プログラムされた)終点(E)に対応しないため、動作できません。

```
%G239_Demo_Circ.nc
N005 G162
N010 G0 X0 Y0 Z0 F1000
N020 G239                (TRC mode)
N030 V.G.WZ_AKT.R = 10
N040 G41 G01 X30 Y20     (TRC selection)
N050 G01 X60
N060 G01 X90
N070 G40                 (TRC deselection without motion)
N080 G02 X140 Y-30 J-50  (Error, center point not correct)
N100 G00 X200
N110 G40
N120 M30
```

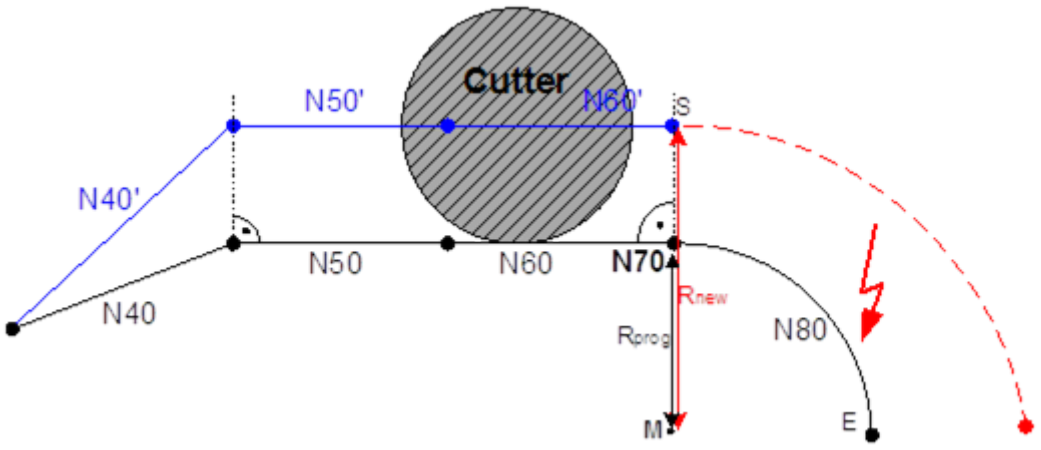


図12-13: G239_Demo_Circ.ncの動作ブロックシーケンス

13.2.6 補正ブロックの生成

補正ブロックの生成は、NCブロックシーケンスに関する制限事項(直接または間接TRC選択に適用される制限事項など)の対象ではありません。現在のブロック後の次のブロックは、プログラムされたNCブロック間の形状遷移の計算に使用されます。

形状遷移は直線上(G25)か(デフォルト)、または円弧上(G26)です(オプション)。送り速度の調整(G10/G11)のオプションもあります。

下のテーブルと補足図に、すべての可能な形状遷移を示します。そこでは、両方の可能な遷移(直線および円弧中間ブロック)が示されます。

TRCによるブロックの挿入

LIN: 直線ブロック、CIR: 円弧ブロック

G25: 直線遷移の挿入

G26: 円弧遷移の挿入

(2 LIN : TRCが、遷移時に、2つの直線ブロックを挿入します)。

	角度範囲					
	0° ~ 180°		180° ~ 270°		270° ~ 360°	
プログラムブロックシーケンス	G25	**G26**	G25	**G26**	G25	**G26**
LIN-LIN	0	**0**	0	**1 CIR**	1 LIN	**1 CIR**
LIN-CIR/CIR-LIN	0	**0**	1 LIN	**1 CIR**	2 LIN	**1 CIR**
CIR-CIR	0	**0**	2 LIN	**1 CIR**	3 LIN	**1 CIR**

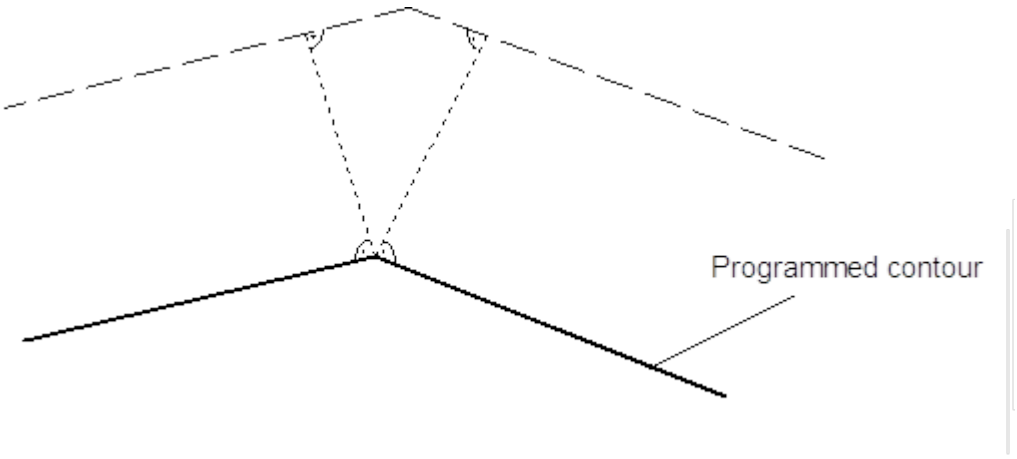


図 93: 図12-14: ブロックシーケンスlinear-linearの場合の、直線への形状遷移の例

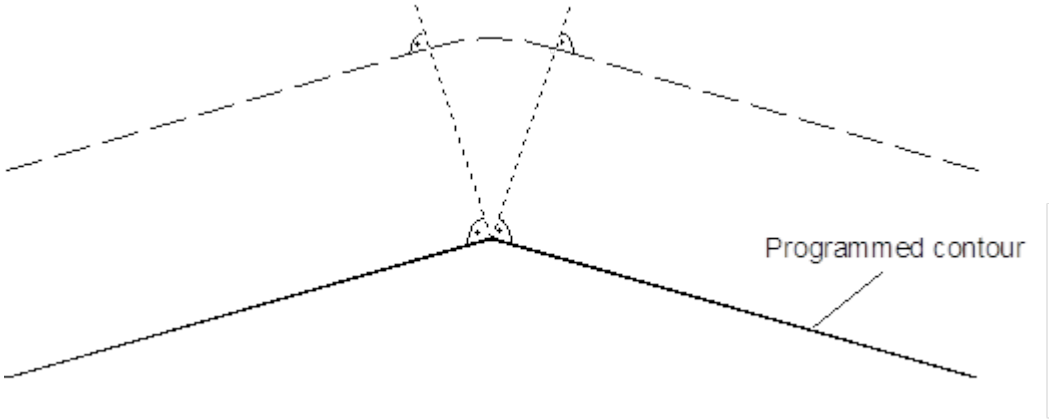


図 94: 図12-15: ブロックシーケンスlinear-linearの場合の、円弧への形状遷移の例

表 19: 図12-16: TRC遷移: 0° < ß ≤ 180 °の場合の組み合わせケース(補正ブロックが追加されないため、G25またはG26には非依存)

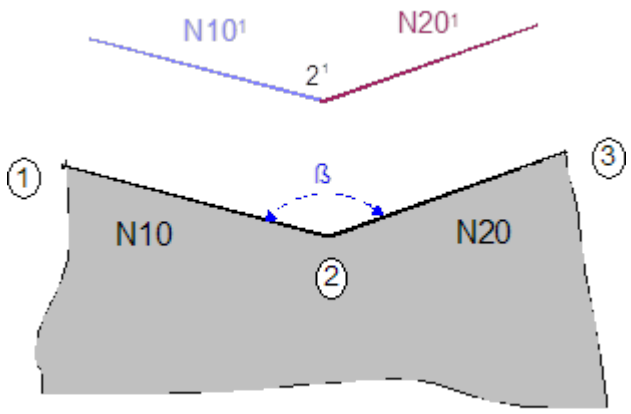
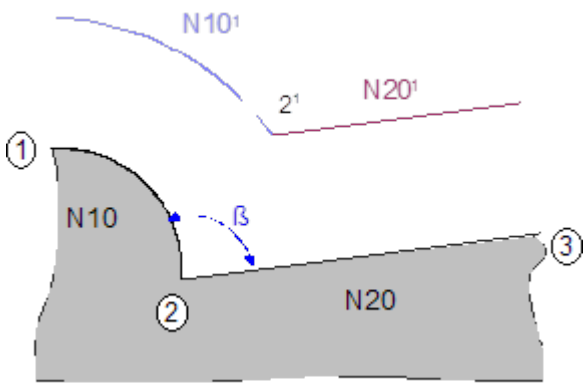
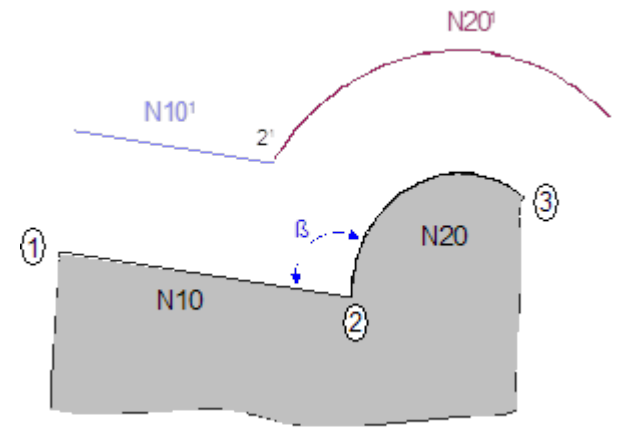
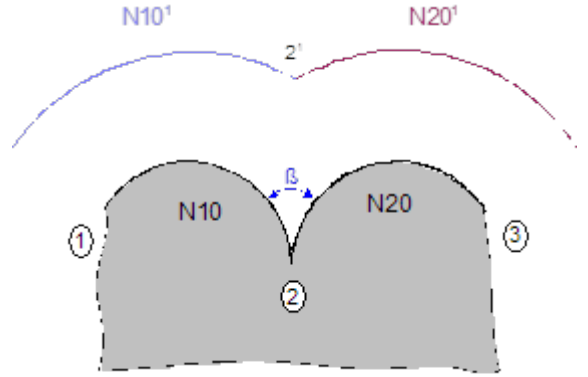
組み合わせタイプ1: 0° < ß ≤ 180°、G41	組み合わせタイプ3: 0° < ß ≤ 180°、G41
	
組み合わせタイプ2: 0° < ß ≤ 180°、G41	組み合わせタイプ4: 0° < ß ≤ 180°、G41
	

表 20: 図12-17: TRC遷移: 180°< ß ≤ 270°の場合の組み合わせケース(直線中間ブロック付き)

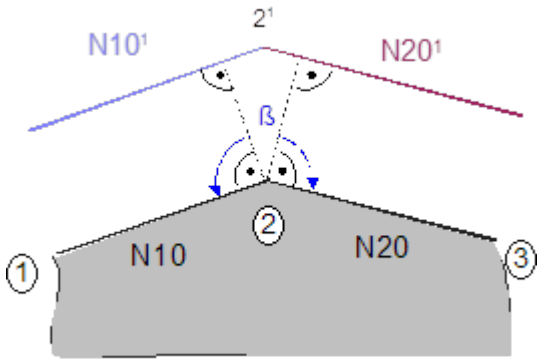
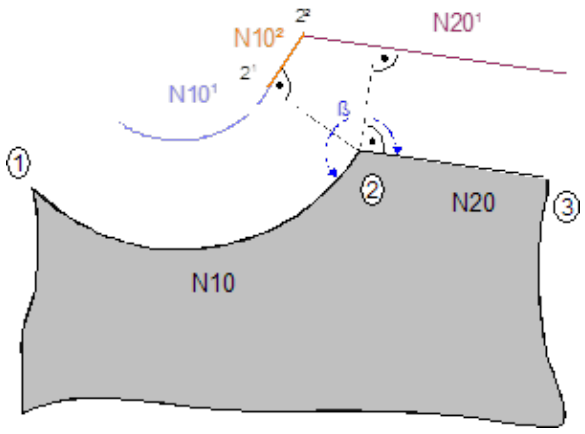
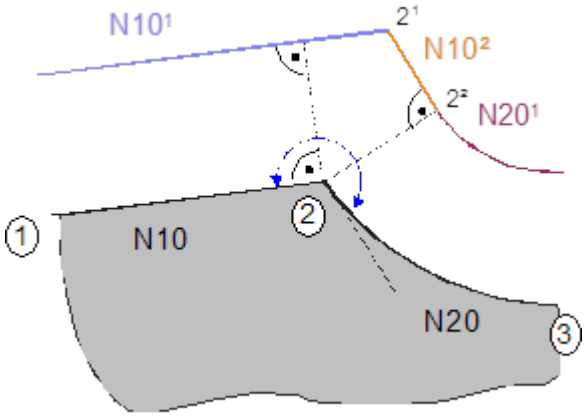
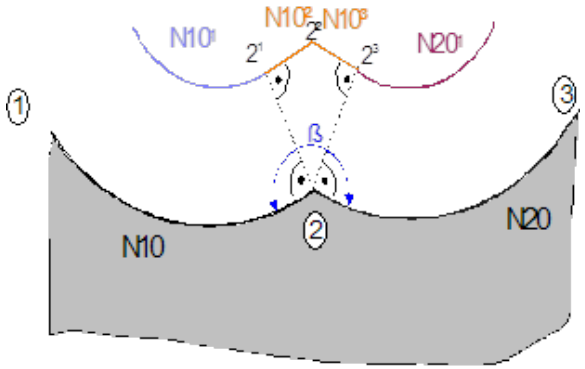
組み合わせタイプ1: 180° < ß ≤ 270°、G41	組み合わせタイプ3: 180° < ß ≤ 270°、G41
	
組み合わせタイプ2: 180° < ß ≤ 270°、G41	組み合わせタイプ4: 180° < ß ≤ 270°、G41
	

表 21: 図12-18: TRC遷移: 270°< ß ≤ 360°の場合の組み合わせケース(直線中間ブロック付き)

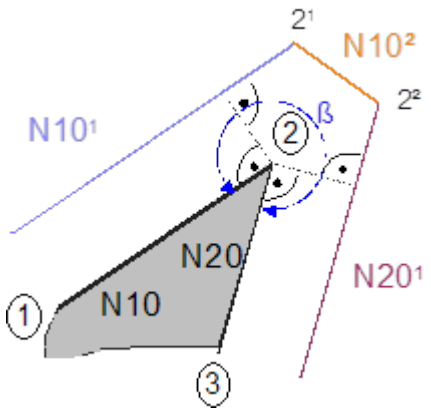
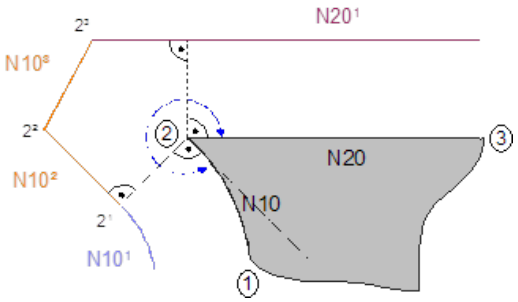
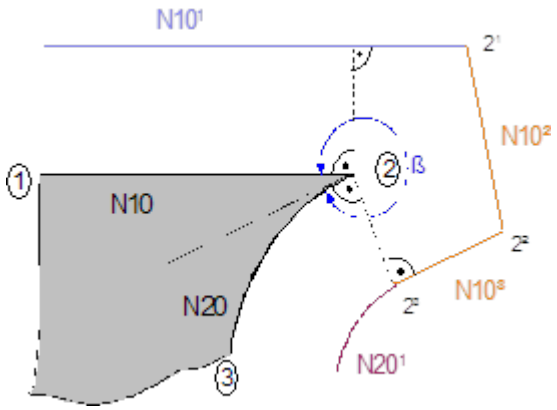
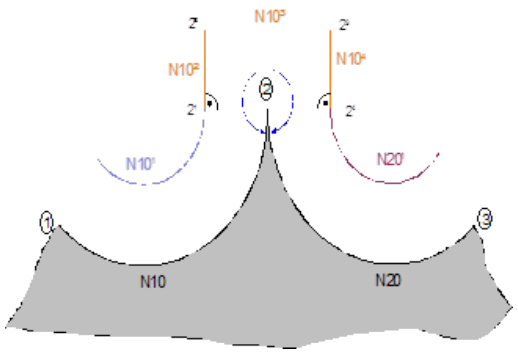
組み合わせタイプ1: 270° < ß ≤ 360°、G41	組み合わせタイプ3: 270° < ß ≤ 360°、G41
	
組み合わせタイプ2: 270° < ß ≤ 360°、G41	組み合わせタイプ4: 270° < ß ≤ 360°、G41
	

表 22: 図12-19: TRC遷移: 180°< ß ≤ 270°の場合の組み合わせケース(円弧中間ブロック付き)

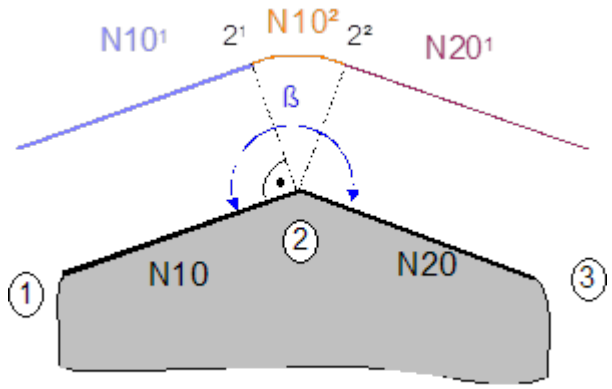
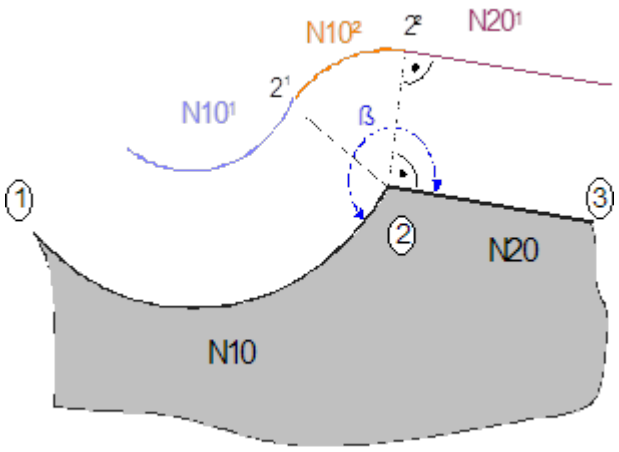
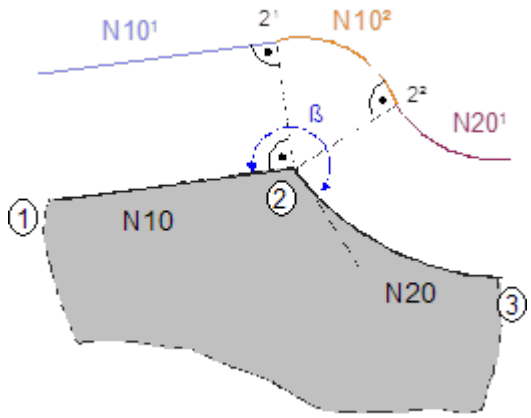
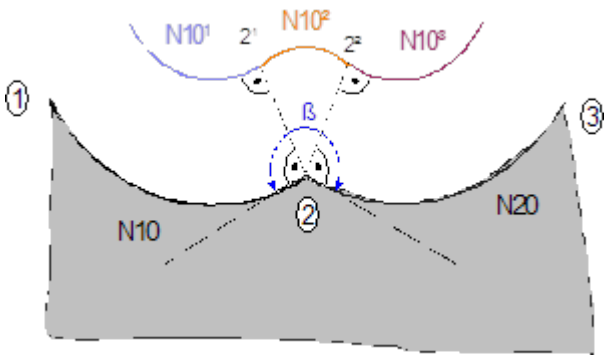
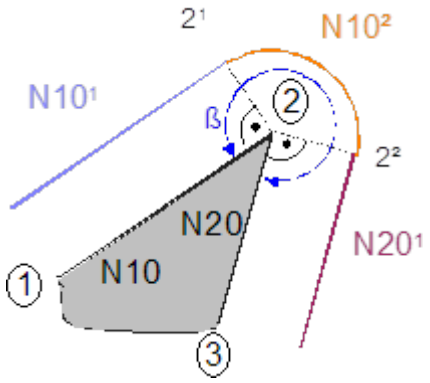
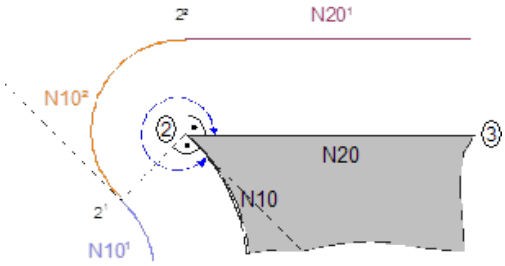
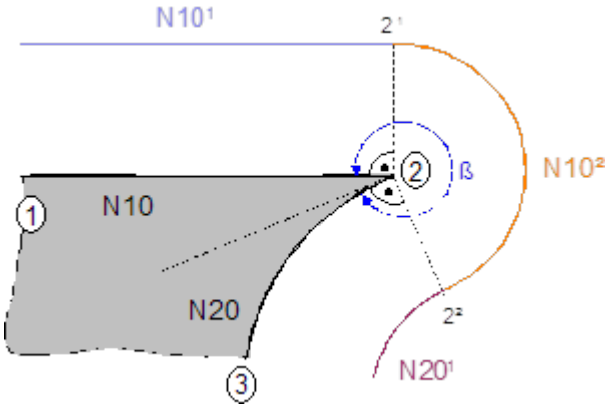
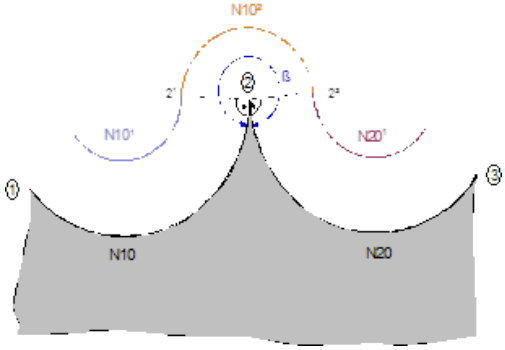
組み合わせタイプ1: 180° < β ≤ 270°、G41、G26	組み合わせタイプ3: 180° < β ≤ 270°、G41、G26
	
組み合わせタイプ2: 180° < β ≤ 270°、G41、G26	組み合わせタイプ4: 180° < β ≤ 270°、G41、G26
	

表 23: 図12-20: TRC遷移: 270°< ß ≤ 360°の場合の組み合わせケース(円弧中間ブロック付き)

組み合わせタイプ1: 270° < ß ≤ 360°、G41、G26	組み合わせタイプ3: 270° < ß ≤ 360°、G41、G26
	
組み合わせタイプ2: 270° < ß ≤ 360°、G41、G26	組み合わせタイプ4: 270° < ß ≤ 360°、G41、G26
	

13.2.7 形状変更中の動作

例えば、G41からG42への直接形状変更は、TRCの明示的な選択解除(G40)せずに可能です。この形状変更は、TRC選択解除と選択に対応します。形状変更は、分離およびアプローチブロックの直線NCブロックにのみ実行する必要があります。形状変更の直前/直後の円弧ブロックも可能です。

形状変更中に円弧ブロックがプログラムされると、エラーメッセージが出力されます。

プログラミング例

```
N05 G01 Y10 F100 G41 V.G.WZR=2  (Linear block with select. of TRC left)
                                (of the contour)
N10 X20 Y25
N20 X40 G42                     (Linear block with select. of TRC right)
                                (of the contour)
N30 X60 Y10
N40 X0 Y0 G40                   (Deselection of TRC)
N50 M30
```

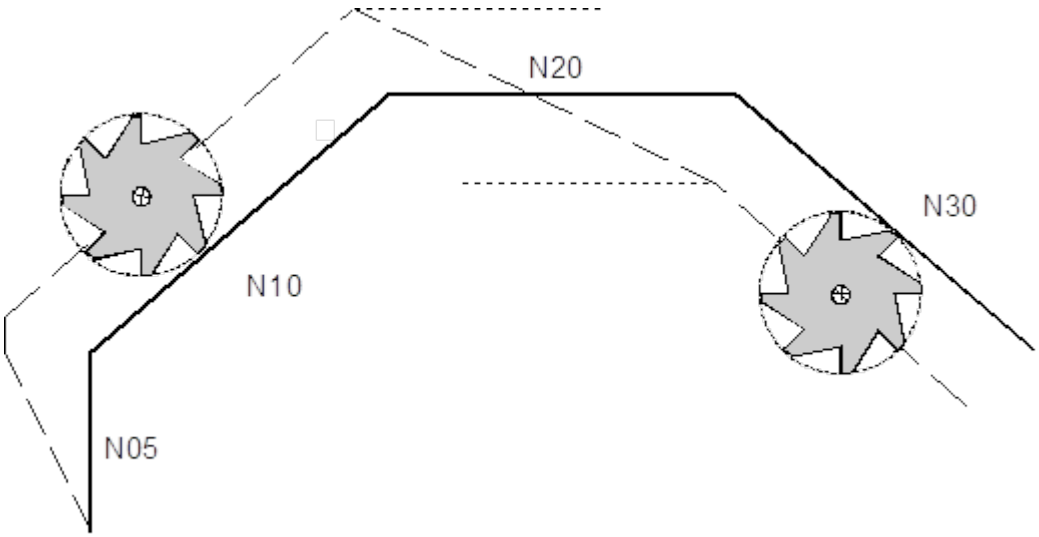


図 95: 図12-21: 選択解除なしの選択変更の例

13.2.8 工具半径の変更中の動作

工具半径の変更は、直線ブロック内と円弧ブロック内の両方で可能です。

プログラミング例1: 直線ブロック内の変更

```
%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 G02 X200
N50 X240 Y100
...
N200 G40 X500
N999 M30
```

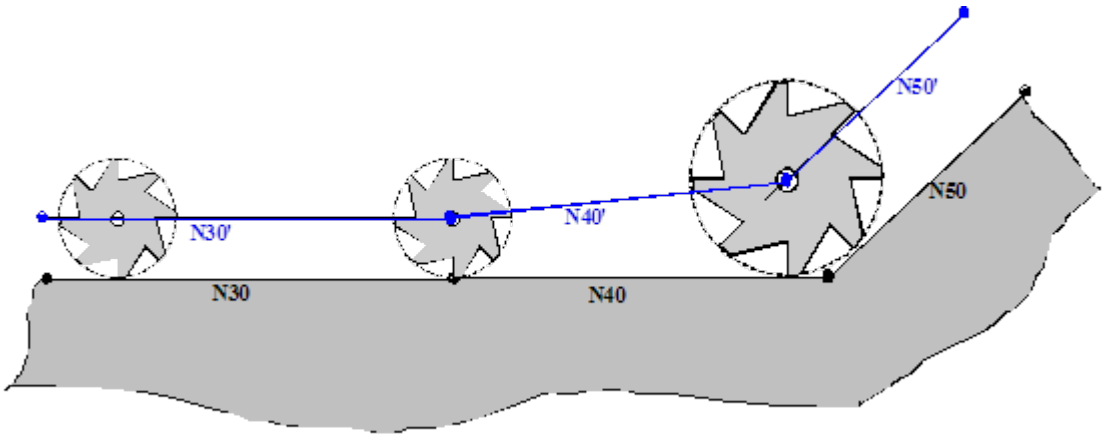


図 96: 図12-22: 直線ブロック内の工具径変更

プログラミング例2: 円弧ブロック内の変更

```
%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
N50 X240 Y150
...
N200 G40 X500
N999 M30
```

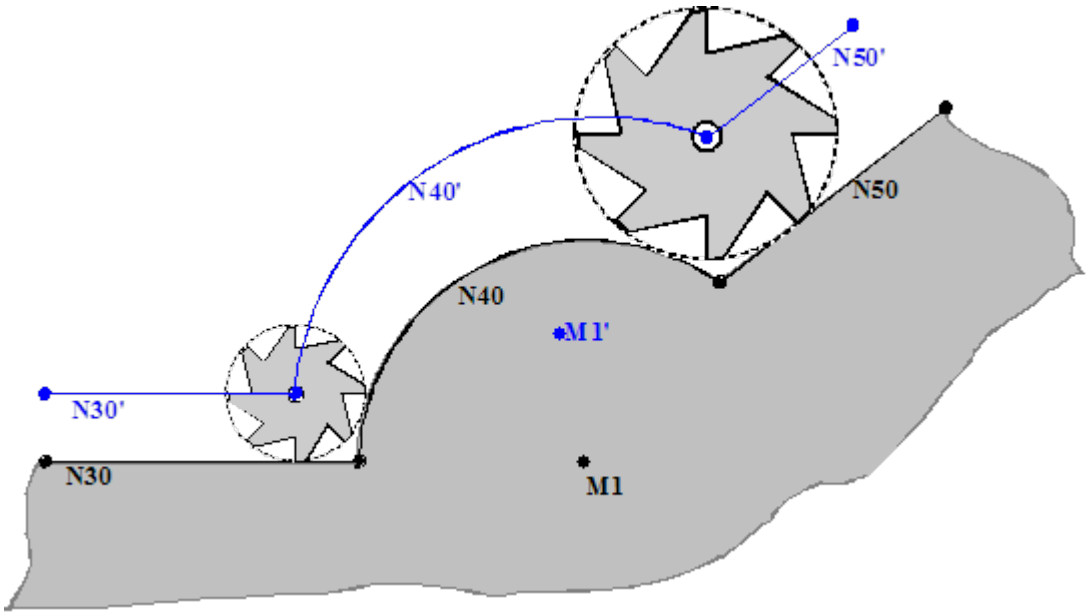


図 97: 図12-23: 円弧ブロック内の工具径変更

13.2.9 接線の変化が連続的なモードでのTRCの選択および解除

TRCの直接選択/選択解除モード中は、通常、加工開始時にパス形状にねじれが生じます。角度が180°よりも大きい場合、さらに、選択されたブロックおよび選択解除の前のブロックに続くブロックの形状が損傷します。

直接選択/選択解除モード中のこの形状損傷を避けるだけでなく、パス上のれじれ点で発生するジャークを最小化するために、タンジェンシャルエントリおよびタンジェンシャルイグジットが役に立ちます。

同一のブロック内で、G40、G41、G42と組み合わせてG05をプログラムする必要があります。これから、形状の開始時または終了時に接線遷移を行うかどうかが導き出されます。

その時点の位置から、次の形状エレメントG01、G02...が円接線方向にプログラムされた送り速度でアプローチされます; 必要に応じて、G10/G11を使用して送り調整を行います。

G41またはG42と組み合わせたG05は形状開始時にタンジェンシャルエントリを発生させ、有効なG41/G42の下でのG05とG40は形状終了時にタンジェンシャルイグジットを発生させます。その結果、選択/選択解除ブロックは円弧ブロックに変換されます。

TRCの選択および選択解除中にG05ファンクションによってトリガされる動作ブロックについては、図12-17および図12-18に示します。異なる開始点(P1、P1')による類似のプログラミングで現れる合計で4つのコースを示します。

接線モードでの選択

選択点(AnP)は、従来の直接呼出しの場合に適用されるものとして計算されます。最初に選択されたブロックの方向と開始点の位置に基づいて、円弧選択ブロックの回転方向が指定されます。円弧中心点(MP)は、開始点(P1またはP1')からの中心垂線と選択点(AnP)とアプローチブロックの直線開始点(P2)と選択点(AnP)の交点です。

接線モードでの選択解除

最後に補正された終点(AbP)(この場合、最後の選択点と呼ばれた後)は、従来の直接選択解除モードの場合と同様に計算されます。最後に選択されたブロックの方向と選択解除点(P4またはP4')の位置を考慮して、円弧選択解除ブロックの回転方向が指定されます。円弧中心点は、最後の選択点(AbP)の接続線に対する中心垂線と選択解除点と直線の交点です。最後の選択点(AbP) - 選択解除の前に最後にプログラムされた点(P3)。

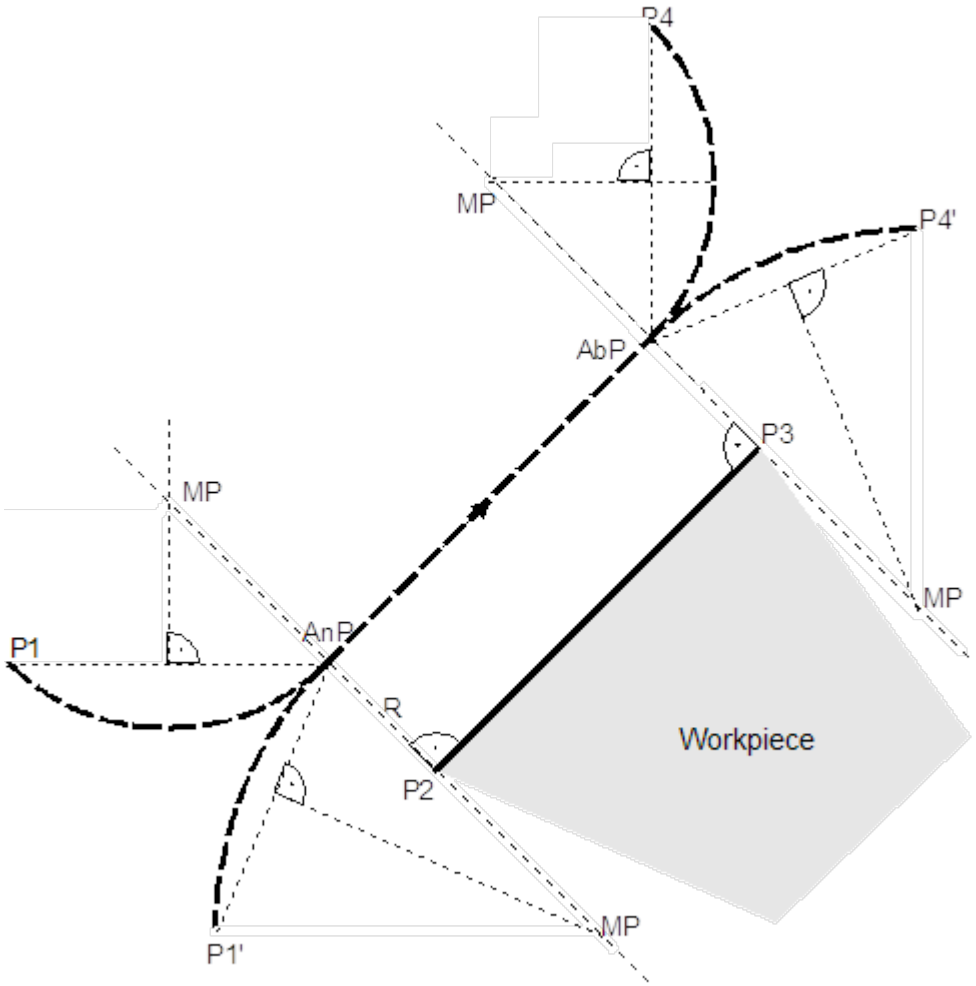


図 98: 図12-24: 接線モードでのTRCの選択および選択解除

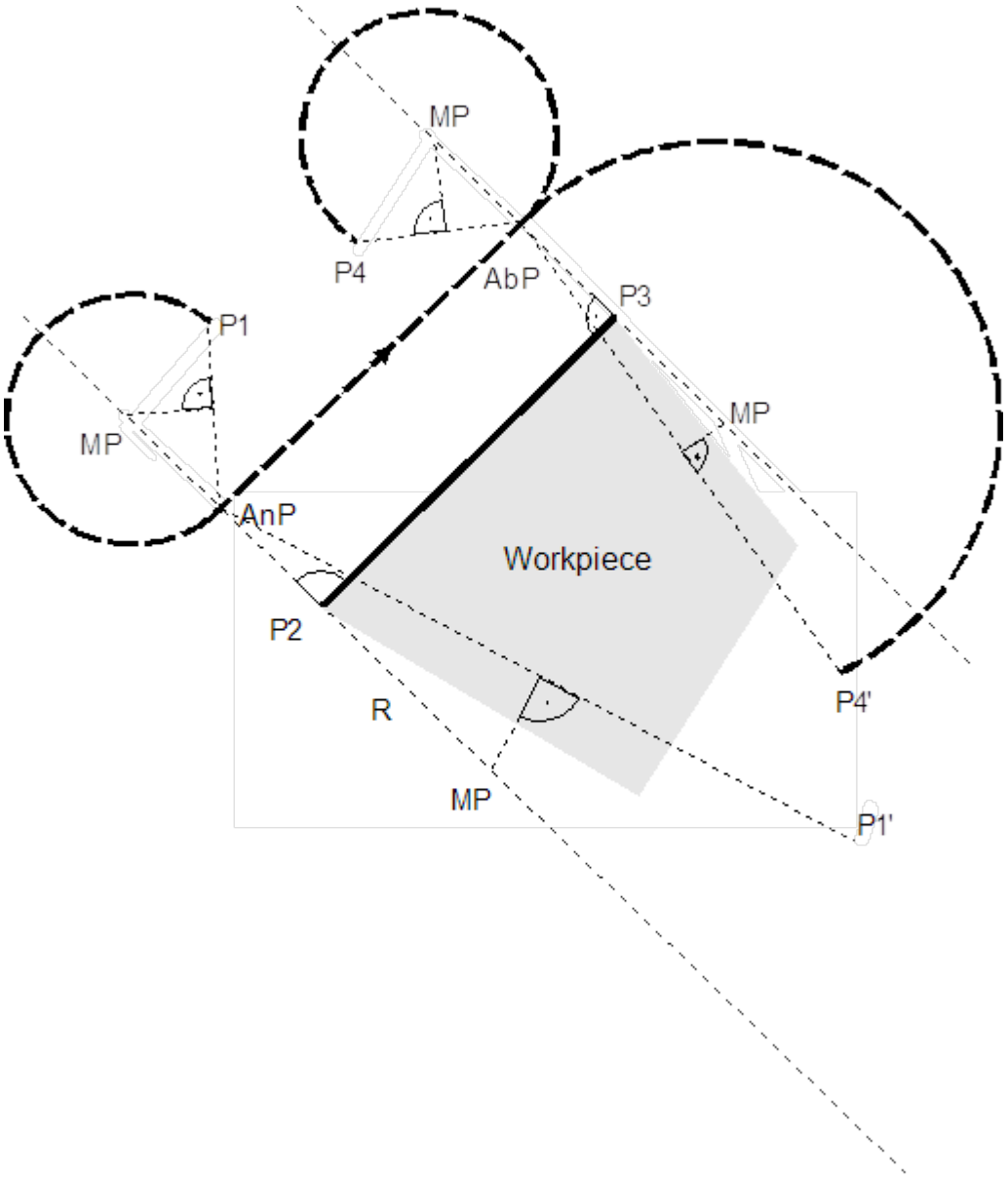


図 99: 図12-25: 接線モードでのTRCの選択および選択解除

13.2.10 TRCの制限

中間ブロックの計算では、TRCは、実際のブロックと2つのその他の関連するNCブロックを考慮します。「関連」は、工具径補正が行われる選択された主面内の移動ブロックを意味します。「関連なし」は、例えば技術ブロック、時間遅延(G04)、または主面に直交する1つの軸による移動動作です。これらの3つのブロック内で、プログラムされたプロファイルと逆方向に動作する中間ブロックを計算すると、形状違反が認識されます。図12-26では、このコンテキストでの例が示されます。

実際のブロックから3ブロックより遠く離れた1つ以上のブロックによってボトルネック位置が発生した場合は、この形状違反はTRCでは認識されません。このケースのための説明は、図12-27に提供されます。

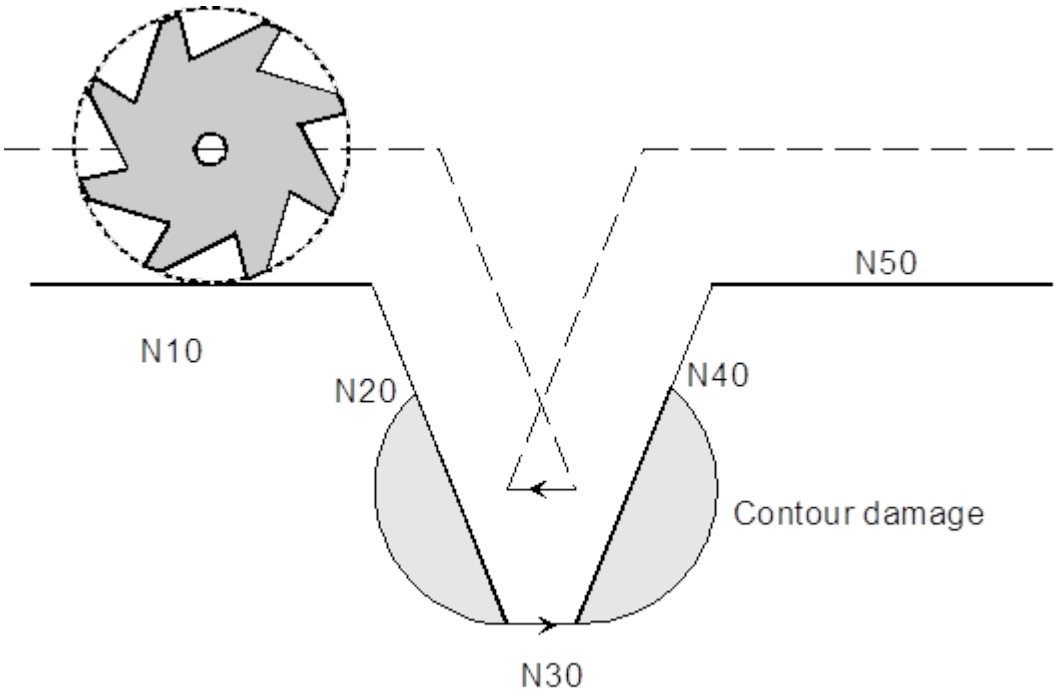


図 100: 図12-26: 検出された形状損傷の例

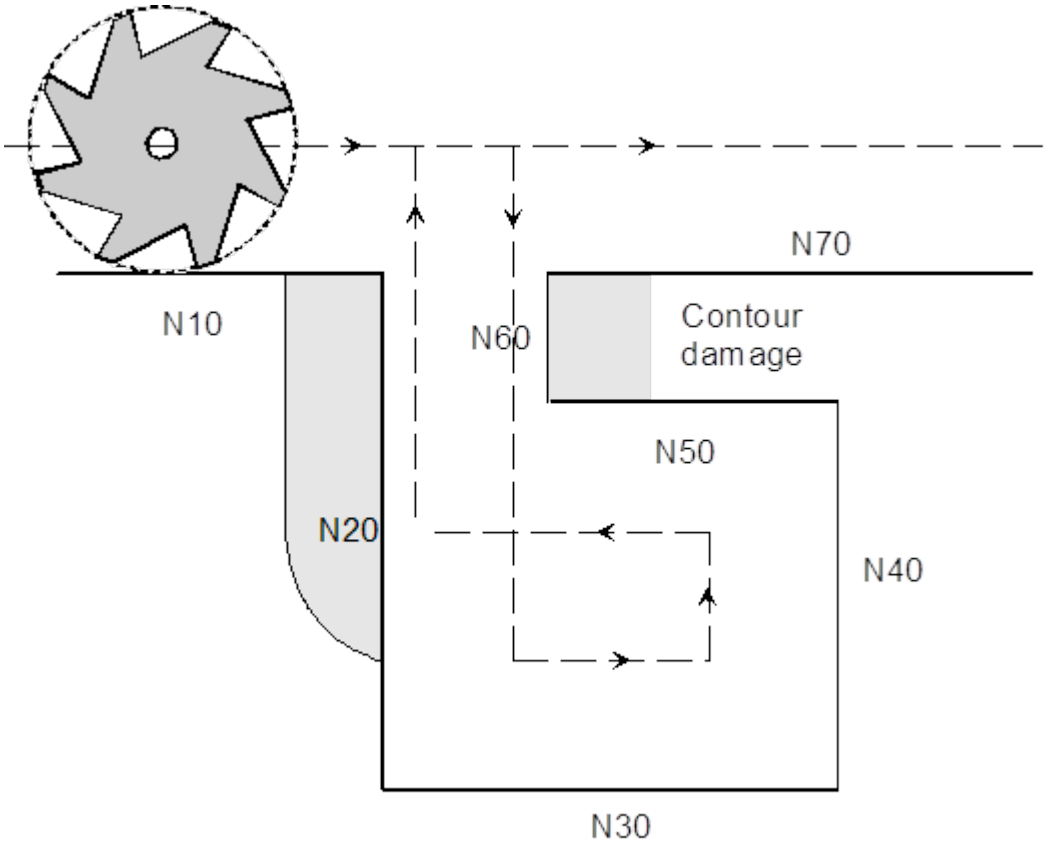


図 101: 図12-27: 検出されない形状損傷の例

13.2.11 プログラム可能なその他のオプション

次のコマンドは、オプションのTRCファンクションのプログラミングを可能にします。

選択後、このオプションTRCファンクションは、メインプログラム終了またはRESETまで有効ですが、NCプログラムの実行中にいつでも選択解除することもできます。

```
#TRC [ [ CONV_CIR_TO_LIN<expr> ] [ KERF_MASKING<expr> ] [ REVERSE<expr> ]
       [ IGNORE_CONT_DAMAGE<expr>] [ REMOVE_MASKED_BLOCKS <expr> ] ]
       [ EXT_ANGLE_BLOCK_INTERSECTION<expr>] ]
```

CONV_CIR_TO_LIN<*expr*> このパラメータは、プログラムされた半径が工具半径未満である円弧ブロックを直線ブロックに直接に変換することを制御します。有効性のための前提条件は、有効な形状マスキングプロセス(G141)です。

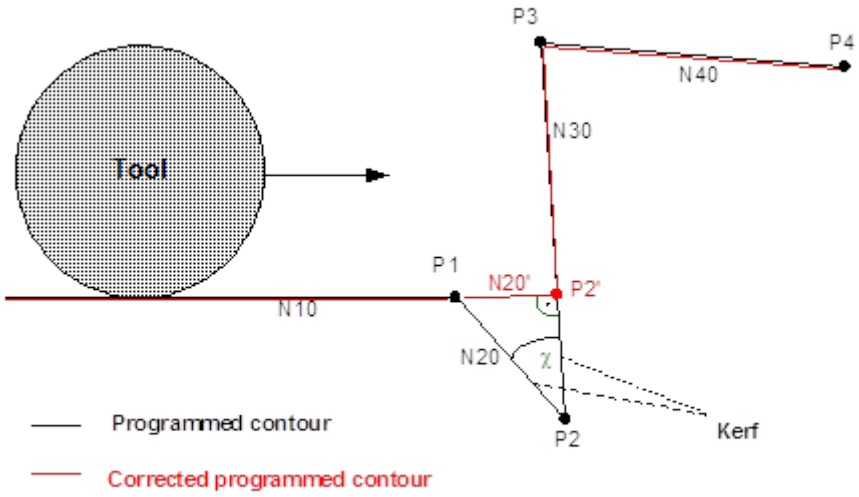
値	優先度
0	円弧ブロックの変換なし(デフォルト)。
1	直線ブロックへの円弧ブロックの変換。

KERF_MASKING<*expr*> このパラメータは、切り口のマスキングを明示的に有効にします。

値	優先度
0	切り口マスキングなし(デフォルト)。
1	切り口マスキングが有効。

形状マスキングを使用して、このTRCオプションは暗黙的に有効/無効になります。

切り口が工具で処理できないとき、切り口マスキングの機能は、プログラムされた点のシフトに基づいて行われます。



REVERSE<expr> このパラメータは、TRCが有効な場合に、選択解除(G41<->G42)せずに選択側の直接変更を可能にします。

値	優先度
0	選択側の直接変更なし(デフォルト)。
1	選択側の直接変更が有効。

選択側の変更は、常に反転する点で行われます。このための条件は、直線動作ブロックによって正確に動作が反転することです。

円弧動作ブロックを使用して、反転する点で両方の円弧の接線を同一にする必要があり、両方の円弧の方向は異なる方向にする必要があります。

IGNORE_CONT_DAMAGE<*expr*> このパラメータによって、形状損傷は明示的に無視されます。

値	優先度
0	形状損傷を無視しない(デフォルト)。
1	形状損傷を無視する。

REMOVE_MASKED_BLOCKS<*expr*> このパラメータでは、形状内の検出済みループを形状マスキングによって検出できます。閉ループの識別のための両方の動作ブロックも検出されます。

従軸の純粋な動作内容は保存されます。TRCの観点から言えば、3番目の主軸の動作はこれらの純粋な動作に属します。

このパラメータは、非常に短い動作ブロックによる形状の場合に特に適しています。

有効な形状マスキング(G141)は、このパラメータの有効性の条件です。

値	優先度
0	閉じた形状ループは検出されません(デフォルト)。
1	閉じた形状ループが検出されます。

注記

形状ループ内の動作ブロックが完全に削除されます。これらの動作ブロックでの従軸の既存の追加動作に関する検査は行われません。

EXT_ANGLE_BLOCK_INTERSECTION<expr>　このパラメータでは、2つの動作ブロック間の遷移角度のリミットを180°から181°に切り替えることができます。これによって、この角度範囲の追加TRC遷移ブロックの作成が回避されます。

リミット自体の値を変更することはできません。

値	優先度
0	遷移角度のリミットは180°です(デフォルト)。
1	遷移角度のリミットは181°です。

プログラミング例

円弧ブロックの 変換:

```
N1000 V.G.WZ_AKT.R=1              (Tool radius)
N2300 G140                        (Deactivate contour masking)
....
N2450 #TRC[CONV_CIR_TO_LIN=1]     (Activate parameter)
N2500 G41                         (Selection of TRC left of the contour)
...
(Circular element with radius less than the tool radius)
N2550 G03 X3557.83 Y-577.61 I0.00 J0.60
(no direct conversion from circular to a linear block)
...
N3000 G141                        (Activate contour masking)
...
(Circular element with radius less than the tool radius)
N3550 G03 X3557.83 Y-577.61 I0.00 J0.60
(direct conversion from circular to a linear block)

...
N3600 G40                         (Deselection of TRC)
```

選択側の直接変更:

```
...
N090 G90
N100 #TRC[REVERSE=1]
N110 G00 X0 Y0
N120 G01 X100 Y100
N130 G41 G01 X150          (Selection of TRC left)
```

```
N140 G01 X250
N150 G01 Y200               (Point of reversal)
N160 G42                    (Change side of selection of TRC to right)
N170 G01 X100
N180 G01 X100Y150
N160 G40 G01 X0
N170 G01 Y0
...
```

14 変数と変数の計算

変数としては、一方で固有名が割り当てられたデコーダの内部データが存在し、他方で基本的に名前を自由に選択できるユーザ定義変数が存在します。外部変数(V.E...)を除いて、変数の有効性と使用は特定のNCチャンネルのみに限定されています。

通常の構文を次に示します。

V.<NAME_1>.<NAME_2>.<NAME_3>.{<*NAME_n*>.}

- V. 変数へのアクセスを示します
- *<NAME_1>*. グローバルデータ指定:
  - 「A.」 軸特有の変数を示します、
  - 「SPDL.」 スピンドル固有の変数を示します、
  - 「G.」 軸間変数、グローバルに有効な変数を示します、
  - 「E.」 外部変数を示します、
  - 「P.」 ユーザ定義、非プログラム間、非グローバル変数を示します、
  - 「L.」 ユーザ定義、非プログラム間、ローカル変数を示します、
  - 「S.」 ユーザ定義、プログラム間、グローバル変数を示します。
- *<NAME_2>*. データ名を指定します
- *<NAME_3>*. 例えば、同じ種類の複数のデータを区別する必要がある場合のインデックス。

軸識別のプログラミング

最後の識別コードは、軸特有の変数およびいくつかのグループ固有変数の軸コードを示します。その結果、以下の指定

「.X」　または　「[0]」、

「.Y」　または　「[1]」、

「.Z」　または　「[2]」

は、チャンネルパラメータ[1]-5で、名前「X」がインデックス0の軸に割り当てられ、名前「Y」がインデックス1の軸に割り当てられ、および名前「Z」がインデックス2の軸に割り当てられている場合に、オプションで使用する必要があります。

例: X軸の絶対座標:

V.A.ABS.**X**　または　V.A.ABS**[0]**

スピンドル固有の変数の場合は同様に、スピンドル名、またはチャンネルパラメータデータセットに従った対応するインデックスを使用する必要があります。

「.S」　　または　　「[0]」、

「.S2」　　または　　「[1]」、 …

例: Sスピンドルの論理軸番号:

V.SPDL.LOG_AX_NR.**S**　　または　　V.SPDL.LOG_AX_NR**[0]**

プログラミング例

両方の行は、変数V.A.BZP.Y (Y軸方向の座標プリセット)の値の近くでX軸方向の直線補間を生成します。

```
N10  G92 X0 Y40 Z0                    (Position preset)
N20  G91 G01 F1000 XV.A.BZP.Y
N30  XV.A.BZP[1]                      (here: Axis index 1 == Y )
N40  M30
```

すべての変数からその内容を読み取ることができ、いくつかの変数に値を割り当てることもできます。アクセスタイプは変数ごとに厳密に指定されていますが、一般的には読み取りアクセスのみが許可されます。これは、ほとんどの変数にとって、書き込みアクセスは事実上役に立たないためです。

プログラミング例

ここでは、インデックス1の軸の2.ゼロオフセットベクトルに値100が割り当てられています。

```
N10  V.G.NP[2].V[1] = 100
…
```

EXISTファンクション(「演算式<expr> [▶ 25]」の章を参照)は、変数の一般的な可用性をチェックします。

プログラミング例

軸特有の変数に対するEXIST要求によって、NCチャンネルでの特別な軸の一般的な可用性がチェックされます。

```
...
N10 G90 Y0
N20 $IF EXIST[V.A.LOG_AX_NR.X] == TRUE
N30   X-10              (X-axis is in channel, move to position -10)
N40 $ELSE
N50   #CALL AX [X,1,0]  (X-axis is not in channel, first request axis)
N60 $ENDIF
...
M30
```

1つの外部変数に対するアクセスの前の最初のステップで、そのアクセスが一般的に可能であるかどうかがチェックされます。

```
...
N10 G90 Y0
N20 $IF EXIST[V.E.POS_1] == TRUE
N30   XV.E.POS_1         (Move X-axis on position POS_1)
N40 $ELSE
N50   #MSG ["V.E.POS_1 not available!"]     (Create message and stop)
N55   M0
N60 $ENDIF
...
M30
```

14.1 軸特有の変数(V.A.)

軸特有の変数のコードは「V.A...」です。

注記

V.A.変数は直線および回転軸に対してのみプログラムでき、スピンドルに対してはプログラムできません。

V.A.<var_name>	優先度	データタイプ	入出力の単位	有効なアクセス Read (読み取り)/ Write (書き込み)
MENT.X	直前のNCブロックのメンタル座標 (「ミラーリング G20-G23 [▶ 78]」の章を参照)	Real	[mm]、[inch]	R
PROG.X	直前のNCブロックのプログラム座標 有効な形状回転(#ROTATION)中に、その変数は機械軸上にマッピングされた座標値を提供します。	Real	[mm]、[inch]	R
ABS.X	直前のNCブロックの絶対座標(現在有効な各座標系のNCコマンド#CHANNEL INIT [▶ 160]後の現在の各絶対座標)。	Real	[mm]、[inch]	R
ACS.ABS.X	現在の絶対位置、有効な変換中に機械軸上にマッピング済み。	Real	[mm]、[inch]	R
ACT_POS.X	現在の座標系での実際の現在位置(オフセットなし)	Real	[mm]、[inch]	R
-SWE.X	現在の有効なマイナス側のソフトウェアリミットスイッチ	Real	[mm]、[inch]	R
+SWE.X	現在の有効なプラス側のソフトウェアリミットスイッチ	Real	[mm]、[inch]	R
-SWE_MDS.X	設定されたマイナス側のソフトウェアリミットスイッチ(パラメータP-AXIS-00177に従って設定)。	Real	[mm]、[inch]	R
+SWE_MDS.X	設定されたプラス側のソフトウェアリミットスイッチ(パラメータP-AXIS-00178に従って設定)。	Real	[mm]、[inch]	R
REF.X	機械基準点(機械基準点検索が成功した後にのみ設定されます)	Real	[mm]、[inch]	R
BZP.X	座標プリセット	Real	[mm]、[inch]	R
PZV.X	機械テーブルオフセット	Real	[mm]、[inch]	R
MESS.X	計測動作後に座標系の軸特有の計測値が収納されます。計測中に有効です。その値には、常にすべてのオフセットが含まれます。 2.5Dの場合: ACS値、および、CS/TRAFOの場合: PCS値	Real	[mm]、[inch]	R

バージョンV2.11.2020.07以降では、軸特有の変数V.A.MEAS.ACS.VALUEおよびV.A.MEAS.PCS.VALUEを変数V.A.MESSの代わりに使用します。追加されたこれらの変数は、すべてのオフセットが含まれた軸座標系でも、オフセットを含まないプログラミング座標系でも計測値を提供します

MEAS.ACS.VALUE.X	計測動作後に軸座標系(ACS)の軸特有の計測値が収納されます。値には、すべてのオフセットが含まれます。	Real	[mm]、[inch]	RL
MEAS.PCS.VALUE.X	計測動作後にプログラミング座標系(PCS)の軸特有の計測値が収納されます。値には、オフセットは含まれ**ません**。	Real	[mm]、[inch]	R
MOFFS.X	計測オフセット	Real	[mm]、[inch]	R
MERF.X	計測動作が完了しているか? yesの場合、1	Boolean	0 , 1	R
MEIN.X	含まれる計測オフセット	Real	[mm]、[inch]	R
RERF.X	機械基準点検索は完了しましたか? yesの場合、1	Boolean	0 , 1	R
MANUAL_OFFSETS.X またはSOFFS.X	手動操作中の動作パス。NCコマンド#GET MANUAL OFFSETS [▶ 159]と組み合わせてのみ有効	Real	[mm]、[inch]	R
MODE.X	現在の軸モード	Integer	-	R
MODULO_VALUE.X	モジュロ範囲	Real	[°]	R
LOG_AX_NR.X	軸の論理軸番号	Integer	-	R
AX_LIST_NAME.X	設定された軸名(P-AXIS-00297に従って設定)	String	-	R
MIRROR.X	軸のミラーモード: (1: ミラーリングなし -1: 軸がミラーリングされます)	Integer	-	R
WCS.X MCS.X	機械座標(MCS)と加工物座標(WCS)の間の変換。 NCコマンド#WCS TO MCS [▶ 530]および#MCS TO WCS [▶ 530]と組み合わせてのみ有効	Real	[mm]、[inch]	R/W

軸名文字列(例: X_SCHLITTEN、「3.1: 軸コマンド」の章も参照)の使用に関しては、これらの軸名を変数を識別するために使用します。

例: V.A.MENT.X_SCHLITTEN

プログラミング例

```
N10 G90 G92 X50
N20 G100 X100                ;Meas. travel, interrupt 2mm before target
N30 G90 G92 X0
N40 XV.A.MESS.X YV.A.MOFFS.X    ;X on 148 (98+50), Y on 2
```

```
or

N40 XV.A.MEAS.ACS.VALUE.X YV.A.MOFFS.X  ;X on 148 (98+50), Y on 2

N50 XV.A.MEAS.PCS.VALUE.X               ;X on 98
```

14.2 スピンドル固有の変数(V.SPDL.)

スピンドル固有の変数のコードは、「V.SPDL. ...」です。

V.SPDL.<var_name>	優先度	データタイプ	入出力の単位	有効なアクセス Read (読み取り)/ Write (書き込み)
LOG_AX_NR.S	スピンドルの論理軸番号	Integer	-	R
PLC_CONTROL.S	スピンドルはPLCスピンドルか? yesの場合、1	Boolean	0 , 1	R
NBR_IN_CHANNEL	現在のNCチャンネルで使用可能なスピンドルの合計数	Integer	-	R
M_FCT_FREE	MファンクションM3、M4、M5、M19の分類方法? スピンドルMファンクションとして明示的に定義: 0 他の技術ファンクションに自由に使用可能: 1 **警告 書き込みアクセスにより、内部チャンネルパラメータ(P-CHAN-00098)が常に実行されます!**	Boolean	0 , 1	R/W

プログラミング例

スピンドルをプログラムする前に、最初にそのスピンドルがチャンネルで既知であるかどうかがチェックされます。

```
...
N10 G90 Y0
N20 $IF EXIST[V.SPDL.LOG_AX_NR.S] == TRUE
N30   M3 S1000          (Spindle S with speed 1000 rpm)
N40 $ELSE
N50   #MSG ["Spindle S is not available!"]  (Message output and stop)
N55   M0
N60 $ENDIF
...
M30
```

14.3 グローバル変数(V.G.)

チャンネル内の軸間でグローバルに有効な変数は、「V.G. ...".

いくつかのV.G.変数でプログラム可能な軸特有のコードに関しては、軸インデックスまたは対応する軸名を使用する必要があります。

V.G.<var_name>	優先度	データタイプ	入出力の単位	有効なアクセス Read (読み取り)/ Write (書き込み)
BLOCK_NR	最後にプログラムされたNCブロック番号	Integer	-	R
PROG_SA	NCブロックの変更がこれ以上行われない場合、ローカルブロックタイプ	Integer	-	R
MASS_MM	計測単位が[mm]の場合、0	Boolean	0 , 1	R
MASS_360	計測単位が[°]の場合、0	Boolean	0 , 1	R
I	円弧プログラミングのI軸座標。中心点補正(G165)が有効な場合の、補正値へのアクセス	Real	[mm]、[inch]	R
J	円弧プログラミングのJ軸座標。中心点補正(G165)が有効な場合の、補正値へのアクセス	Real	[mm]、[inch]	R
K	円弧プログラミングのK軸座標。中心点補正(G165)が有効な場合の、補正値へのアクセス	Real	[mm]、[inch]	R
R	円弧補間中に移動された半径	Real	[mm]、[inch]	R
FEEDRATE	最後にプログラムされた送り速度(Fワード)	Real	[mm/min]、[m/min]、[inch/min]	R
FEEDRATE_SCALE	mm/min単位でプログラムされた送り速度を「prog_start.feedrate_factor」(P-CHAN-00108)で設定された単位に調整することができます。次の値を提供します。 「prog_start.feedrate_factor」の1000 = 0.1（m/min) 「prog_start.feedrate_factor」の1 = 100 (mm/min) プログラミング例: F2000/V.G.FEEDRATE_SCALEの結果 1000: F2 (m/min) 1: F2000 (mm/min) **[V2.11.2024.00から]**	Integer	-	R
MERR[i]	現在の平面での主軸の円弧中心点の補正オフセット(<i> = 0、1)	Real	[mm]、[inch]	R

「WZ[j]」変数によって、任意の工具のデータに対する読み取りアクセスが可能になります。これらの変数は、外部工具管理(トランスペアレントアクセス)および内部工具テーブルの使用の両方で利用可能です(この場合、jはそれぞれ工具リストの工具のインデックスが示す工具番号に対応します)。

WZ[j].R	工具の半径	Real	[mm]、[inch]	R
WZ[j].L	工具の長さ	Real	[mm]、[inch]	R
WZ[j].P[i]	工具のパラメータ	Real	-	R
WZ[j].V[i]または WZ[j].V.X	工具の軸<i>または「X」のオフセット	Real	[mm]、[inch]	R
WZ[j].ME	工具リストを使用するとき、半径、長さ、および軸オフセットの計測単位は常に0 ([mm]として)を提供します、その他の場合、その変数は意味を持ちません	Boolean	0 , 1	R
WZ[j].OK	工具の有効フラグ; 有効な場合、1	Boolean	0 , 1	R
WZ[j].SPDL_AX_NR	割り当てられたスピンドルの論理軸番号	Integer	-	R
WZ[j].KIN_PARAM[i]	工具のキネマティックパラメータ	Real	[0.1 µm]、[10^{-4}°]	R
WZ[j].KIN_ID	工具のキネマティックID	Integer	-	R
WZ[j].TYPE	工具タイプ(0: フライス削り 1: 旋削 2: 研削)	Integer	-	R
WZ[j].SRK_ID	旋削工具のカッター方向	Integer	-	R
WZ[j].S_MIN_SPEED	最小速度(工具の動的データ)	Real	[U/min]	R
WZ[j].S_MAX_SPEED	最大速度(工具の動的データ)	Real	[U/min]	R
WZ[j].S_MAX_ACC	最大加速度(工具の動的データ)	Real	[°/sec^2]	R
WZ[j].SISTER_VALID	姉妹工具(TOOL-ID)の有効フラグ	Boolean	0 , 1	R
WZ[j].SISTER	有効な姉妹工具の番号	Integer	-	R
WZ[j].VARIANT_VALID	各種工具(TOOL-ID)の有効フラグ	Boolean	0 , 1	R
WZ[j].VARIANT	有効な各種工具の番号	Integer	-	R

「WZ_AKT」変数、「T_AKT」、および「D_AKT」によって、現在の選択された工具のデータに対するアクセスが可能になります。これらの変数は、外部工具管理と内部工具テーブルの使用の両方で使用可能です。

T_AKT	選択された工具の番号	Integer	-	R
D_AKT	選択された工具補正データセットの番号	Integer	-	R

書き込みアクセスでは、選択の時間の間に工具データの一時的な変更が常に発生します。選択解除(D0)、または新しい工具(Dxx)の選択の後に、変更されたデータが失われます。変更されたデータは保存されません!

例外:

工具の選択解除中または新しい工具の選択中に外部の工具管理を使用している場合は、いわゆる追加工具データ(V.G.WZ_AKT.P[i])を採用してP-CHAN-00103を保存します。

WZ_AKT.R	現在の工具の半径	Real	[mm]、[inch]	R/W
WZ_AKT.L	現在の工具の長さ	Real	[mm]、[inch]	R/W
WZ_AKT.P[i]	現在の工具のパラメータ	Real	-	R/W
WZ_AKT.V[i]または WZ_AKT.V.X	現在の工具の軸<i>または「X」のオフセット	Real	[mm]、[inch]	R/W
WZ_AKT.ME	工具リストを使用するとき、現在の工具の半径、長さ、および軸オフセットの計測単位は常に0 ([mm]として)が提供されます。その他の場合、その変数は意味を持ちません。	Boolean	0 , 1	R
WZ_AKT.OK	現在の工具の工具有効フラグ; 有効な場合、1	Boolean	0 , 1	R
WZ_AKT.SPDL_AX_NR	割り当てられたスピンドルの論理軸番号	Integer	-	R
WZ_AKT.KIN_PARAM[i]	現在の工具のキネマティックパラメータ。 **警告 書き込みアクセスでは、値を内部単位でプログラムする必要があります!**	Real	[0.1 µm]、[10^{-4}°]	R/W
WZ_AKT.KIN_ID	現在の工具のキネマティックID	Integer	-	R
WZ_AKT.TYPE	選択された工具の工具タイプ (0: フライス削り 1: 旋削 2: 研削)	Integer	-	R
WZ_AKT.TOOL_FIXED	工具が配置可能または位置が固定	Boolean	0 , 1	R/W
WZ_AKT.SRK_ID	選択された旋削工具のカッター方向	Integer	-	R
WZ_AKT.S_MIN_SPEED	最小速度(工具の動的データ)	Real	[rpm]	R
WZ_AKT.S_MAX_SPEED	最大速度(工具の動的データ)	Real	[rpm]	R
WZ_AKT.S_MAX_ACC	最大加速度(工具の動的データ)	Real	[°/sec^2]	R
WZ_AKT.SISTER_VALID	その姉妹工具(TOOL-ID)の有効フラグ	Boolean	0 , 1	R
WZ_AKT.SISTER	その有効な姉妹工具の番号	Integer	-	R
WZ_AKT.VARIANT_VALID	各種工具(TOOL-ID)の有効なフラグ	Boolean	0 , 1	R
WZ_AKT.VARIANT	各種工具の有効な番号	Integer	-	R
WZ_AKT.WEAR_RADIUS	径補正(OTC)での合計の径摩耗(不連続摩耗と連続摩耗の合計)	Real	[mm]、[inch]	R
WZ_AKT.WEAR_RADIUS_CONT	径補正(OTC)での連続径摩耗	Real	[mm]、[inch]	R
WZ_AKT.WEAR[i]または WZ_AKT.WEAR.X	長さ補正(OTC)での軸<i>または「X」の摩耗	Real	[mm]、[inch]	R
WZ_AKT.WEAR_CONST	摩耗定数(OTC)	Real	[0.1 µm/m]	R/W

V.G.WZ_AKT.P[i]の代わりにV.G.WZ_INFO[i]を使用した追加工具パラメータに対するアクセスも可能です。それは、外部工具管理でのみ使用できます。

WZ_INFO[i]	現在の工具のパラメータ	Real	-	R
WZR	現在の工具径(旧構文)	Real	[mm]、[inch]	R/W
WZL	現在の工具の長さ(旧構文)	Real	[mm]、[inch]	R/W

NP[j].V[i]または NP[j].V.X	1つの軸<i>または「X」のゼロオフセット。 **警告 書き込みアクセスにより、内部ゼロオフセットデータが永続的に実行されます!**	Real	[mm]、[inch]	R/W
NP[j].ALL	ゼログループのすべての軸のアドレス指定。 **警告 書き込みアクセスにより、内部ゼロオフセットデータが永続的に実行されます!**	Real	[mm]、[inch]	R/W
NP_AKT.V[i]または NP_AKT.V.X	軸<i>または「X」の現在有効なゼロオフセット。	Real	[mm]、[inch]	R/W
NP_AKT.ALL	すべての軸の現在有効なゼロオフセットのアドレス指定	Real	[mm]、[inch]	R/W
NP_AKT.IDX	現在有効なゼロオフセットグループのインデックス (G53の0、G54の1など)	Integer	-	R
NP_DEFAULT	スタートアップ後のゼロオフセットグループのデフォルトインデックス	Integer	-	R/W

CNC_CHANNEL	現在のチャンネル番号	Integer	-	R
IPO_COUNT	システム時間カウンタ	Integer	-	R
CS_ACTIVE	CS (#CS…)は有効か? 有効な場合、1	Boolean	0 , 1	R
CS_COUNT	有効な(リンクされた)CSの番号	Integer	-	R
ACS_ACTIVE	ACS (#ACS…)は有効か? 有効な場合、1	Boolean	0 , 1	R
ACS_COUNT	有効な(リンクされた)ACSの番号	Integer	-	R
CS	加工座標系(CS)オン/オフ	Integer	-	R
ACS	フィクスチャ適応座標系(ACS)オン/オフ	Integer	-	R
TOOL_COMP	有効な工具補正(1: RTCP 2: TLC 3: 2.5D)			R
AKT_PLATZ	現在のクランプ位置オフセットインデックス			R

AX_LINK.NR	現在または最後の有効な結合グループの番号	Integer	-	R
AX_LINK.ACTIVE	軸結合(#AX LINK ON)は有効か? 有効な場合、1	Boolean	0 , 1	R
AX_LINK_GROUP[i].ACTIVE	番号[i]の特定の結合グループは有効か? 有効な場合、1	Boolean	0 , 1	R

ROT_ACTIVE	形状回転(#ROTATION...)は有効か? 有効な場合、1	Boolean	0 , 1	R
ROT_ANGLE	形状回転の角度	Real	[°]	R
ROT_CENTER1	形状回転の中心点に対する最初の主軸のオフセット	Real	[mm]、[inch]	R
ROT_CENTER2	形状回転の中心点に対する2番目の主軸のオフセット	Real	[mm]、[inch]	R
TIMER[i]	番号<i>のタイマのカウンタ値	Integer	[ms]	R
RANDOM	範囲0.0 ~ 1.0のランダム値を生成します **警告 TwinCATでは使用できません!**	Real	-	R
PROG_ABS	サイズ決め、0: G91が有効 1: G90が有効	Boolean	0 , 1	R
CNC_RELEASE	CNCバージョンのビルド番号	Integer	-	R
WCS_POSLIMIT_1 WCS_POSLIMIT_2 WCS_POSLIMIT_3	WCS内の主軸の動作リミット。 NCコマンド#GET WCS POSLIMIT [▶ 532]と組み合わせてのみ有効	Real	[mm]、[inch]	R

以下の変数によって、チャンネルパラメータリストで定義されたキネマティックの設定に対するアクセスが可能になります。

V2.11.28*.*以前のバージョンで (TwinCAT 2) V3.00からのシングルステップ変換用。* (TwinCAT 3) アクセスはV.G.KIN[j]を使用して行われます。* (ここで、j:= キネマティックID)。

マルチステップ変換(TwinCAT 3からの)では、アクセスはV.G.KIN_STEP[i].ID[j]を使用して行われます。*

(ここで、i:= 変換ステップ0または1、j:= キネマティックID)。

注記

書き込みアクセスによって、内部チャンネルパラメータデータの常時動作が行われます! 値を内部単位でプログラムする必要があります!

変更は、次のスタートアップまたはチャンネルパラメータリストの再解釈まで有効なままです!

TwinCAT 2およびTwinCAT 3 (シングルステップ): **V.G.KIN[j]。***	**優先度**	**データタイプ**	入出力の単位	有効なアクセス Read (読み取り)/ Write (書き込み)
TwinCAT 3 (マルチステップ): **V.G.KIN_STEP[i].ID[j]。***				
PARAM[k]	キネマティックパラメータ ここで、<k>:= 0 ... パラメータの最大数 - 1	Real	[0.1 µm]、[10^{-4}°]	R/W

「**汎用キネマティック**」(ID 91、[FCT-C27])の設定に対するアクセス用追加変数。以下に記載された変数を使用できます。

ZERO_ORIENTATION[k]	工具の初期方向 ここで、<k>:= 0、1、2 (X、Y、Z)	Real	-	R/W
ZERO_POSITION[k]	工具の初期位置 ここで、<k>:= 0、1、2 (X、Y、Z)	Real	[0.1 µm]、[10^{-4}°]	R/W
PROGRAMMING_MODE	プログラミングモード、P-CHAN-00112を参照	Integer	-	R/W
RTCP	角度変換	Boolean	-	R/W
NUMBER_OF_AXES	キネマティックチェーン内の軸の数	Integer	-	R/W
AXIS[k]。*	キネマティックチェーンの軸 ここで、<k>:= 0 … *.NUMBER_OF_AXES - 1	-	-	-
AXIS[k].TYPE	軸タイプ (1: 直進、2: 回転)	Integer	-	R/W
AXIS[k].ORIENTATION[l]	軸の方向ベクトル(方向) ここで、<l>:= 0、1、2 (X、Y、Z)	Real	-	R/W
AXIS[k].POINT[l]	軸上の補間点 ここで、<l>:= 0、1、2 (X、Y、Z)	Real	[0.1 µm]、[10^{-4}°]	R/W
CHAIN[k]	キネマティックチェーン内の軸順序の説明 ここで、<k>:= 0 … *.NUMBER_OF_AXES - 1	Integer	-	R/W

以下の3つの変数によって、現在選択されているキネマティックIDに対する読み取りアクセスが可能になります。

V.G.KIN_ID	シングルステップ変換の現在選択されているキネマティックID	Integer	-	R
V.G.KIN_ID_STEP[0]	マルチステップ変換の最初のステップの現在選択されているキネマティックID (TwinCat 3 以降)	Integer	-	R
V.G.KIN_ID_STEP[1]	マルチステップ変換の2番目のステップの現在選択されているキネマティックID (TwinCat 3 以降)	Integer	-	R
V.G.TOTAL_KIN_OFFSET[i]	選択されたキネマティックの有効な合計オフセット(有効な工具のキネマティックオフセットとチャンネルパラメータリストからのキネマティックオフセットの合計を含めて)。 **[V2.11.2024.00以降]**	Real	0.1 µm], [10^{-4}°]	R

MAIN_FILE_NAME	NCメインプログラムのファイル名	String	-	R
MAIN_PROG_NAME	NCメインプログラムの名前(%...)	String	-	R
MAIN_PROG_NR	プログラム名が1桁である場合、NCメインプログラムの番号	Integer	-	R
FILE_NAME	現在有効なNCプログラムのファイル名	String	-	R
PROG_NAME	現在有効なNCプログラムの名前(%...)	String	-	R
PROG_NR	NCプログラム名が1桁である場合、現在有効なNCプログラムの番号	Integer	-	R
PROG_LEVEL	現在有効なNCプログラムのプログラムレベル: 1: メインプログラム ≥ 2: ローカルプログラムまたはグローバルサブプログラム	Integer	-	R
PATH_NR	現在有効なNCメインプログラムまたはグローバルサブプログラムの論理パス番号(スタートアップリスト(P-STUP-00019)に従った)	Integer	-	R
FILE_OFFSET	V.G.FILE_OFFSET変数がプログラムされたNCブロックの開始のファイル位置	Integer	-	R
MIRROR_ACTIVE	ミラーリング(G351 X... oder G21、G22、G23)が有効か? 有効な場合、1	Boolean	0 , 1	R
BLOCK_SEARCH_ACTIVE	ブロック検索が有効か? 有効な場合、1	Boolean	0 , 1	R
TRAFO_ACTIVE	キネマティックトランスフォーメーション(#TRAFO...)が有効か? 有効な場合、1	Boolean	0 , 1	R
MCS_ACTIVE	機械軸座標系(#MCS...)への一時的な遷移が有効か? 有効な場合、1	Boolean	0 , 1	R
HSC_ACTIVE	自由な形の表面処理(#HSC...)またはスプライン軸補間(#SET SPLINE... oder G151)が有効か? 有効な場合、1	Boolean	0 , 1	R
HSC_SURFACE_ACTIVE	自由な形の表面処理(#HSC [SURFACE...])が有効か? 有効な場合、1	Boolean	0 , 1	R
CONT_MODE_ACTIVE	多項式輪郭加工(G261)が有効か? 有効な場合、この変数は、現在の輪郭加工タイプの値を提供します。 輪郭加工タイプ1の場合1 (AUTO_DEV [▶ 99]) 輪郭加工タイプ2の場合2 (AUTO_VEL [▶ 101]) 輪郭加工タイプ3の場合3 (DIST [▶ 103]) 輪郭加工タイプ4の場合4 (DEV [▶ 105]) 輪郭加工タイプ5の場合5 (POS [▶ 107]) 輪郭加工タイプ6の場合6 (DIST_SOFT [▶ 109]) G260の場合0	Integer	-	R
TRC_ACTIVE	工具径補正(G41、G42)が有効か? 有効な場合、1	Boolean	0 , 1	R

OTC_ACTIVE	オンライン工具補正(#OTC…)が有効か? 有効な場合、1	Boolean	0 , 1	R
CYCLE_ACTIVE	現在のプログラムレベルはサイクルか? 現在のプログラムがL / LL CYCLE...または伝送パラメータ付きのG8xxで呼び出された場合、1	Boolean	0 , 1	R
@P[i].VALID	サイクル呼び出し [▶ 181]、またはサブプログラム呼び出しG8xx [▶ 85] で特定の伝送パラメータがプログラムされているか? 0: パラメータがプログラムされていない 1: パラメータがプログラムされている Index <i>には、妥当性(例えば、@P1の場合1、またはG8xxで呼び出される場合の**最初の**パラメータ)をチェックする必要があるパラメータの番号が格納されます。 **警告 読み取りアクセスの場合の注意事項: この変数は、サイクル内、または伝送パラメータ付きのグローバルサブプログラムG8xx内でのみ使用できます!**	Boolean	0 , 1	R
TIME_STAMP	現在のタイムスタンプ<date time> (以下を使用した形式 <*DD.MM.YYYY hh:mm:ss:zzz*>)。 D: 日、2桁 M: 月、2桁 Y: 年、4桁 h: 時、2桁 m: 分、2桁 s: 秒、2桁 z: ミリ秒、3桁 (例: 16.06.2015 14:08:10:123) **[V2.10.1507.02 以降]**	String	-	R

TIME_STAMP_FILE_NAME	現在のタイムスタンプ<date time> (例えば、レコードファイルの名前で使用する以下を使用したISO 6801:2004形式 <*YYYYMMDDThhmmsszzz*>、「.」 および「:」なし)。 Y: 年、4桁 M: 月、2桁 D: 日、2桁 h: 時、2桁 m: 分、2桁 s: 秒、2桁 z: ミリ秒、3桁 (例: 20150616T140810123) **[V2.11.2024.03以降]**	String	-	R
LIFT_ACTIVE	軸のリフト/シンク(X[LIFT...])は有効か? 有効な場合、1	Boolean	0 , 1	R

以下の変数によって、チャンネルパラメータで定義されたM/Hファンクションのデータに対するアクセスが可能になります。

プリ出力機能(MEP_SVS, MET_SVS)を備えたM/Hコードファンクションの定義では、以下を考慮に入れる必要があります。
M/Hコードファンクションのプリ出力値と同期タイプの書き込みアクセス(W)が適合していない場合、同期タイプが暗黙的に適合します。

M/Hコードファンクションのプリ出力値と同期タイプの読み取りアクセス(R)が適合しない場合、エラーメッセージが作成されます。

注記

変更は、次のスタートアップまたはチャンネルパラメータリストの再解釈まで有効なままです!

M_FCT[i].SYNCH	チャンネルパラメータP-CHAN-00027で定義されたMファンクション<i>の同期タイプ	Integer	-	R/W
M_FCT[i].PRE_OUTP_PATH	Mファンクション<i>の同期タイプMEP_SVSからのプリ出力距離P-CHAN-00070、チャンネルパラメータで定義済み。	Real	[mm]、[inch]	R/W
M_FCT[i].PRE_OUTP_TIME	Mファンクション<i>の同期タイプMET_SVSからのプリ出力時間P-CHAN-00070、チャンネルパラメータで定義済み。	Real	[sec]	R/W
H_FCT[i].SYNCH	チャンネルパラメータP-CHAN-00041で定義されたHファンクション<i>の同期タイプ	Integer	-	R/W
H_FCT[i].PRE_OUTP_PATH	Hファンクション<i>の同期タイプMEP_SVSからのプリ出力距離P-CHAN-000107、チャンネルパラメータで定義済み。	Real	[mm]、[inch]	R/W
H_FCT[i].PRE_OUTP_TIME	Hファンクション<i>の同期タイプMET_SVSからのプリ出力時間P-CHAN-000107、チャンネルパラメータで定義済み。	Real	[sec]	R/W

V.G.SPEED_LIMIT"変数によって、チャンネルパラメータで定義された速度リミットLook Aheadパラメータ[1]-6のデータに対するアクセスが可能になります。

書き込みアクセス中に、変更されたパラメータ設定はチャンネルへ出力されます。有効である可能性がある動作は中断されます!

距離パラメータの値と単位の書き込みアクセス(W)が適合しない場合、単位が暗黙的に適合します。

距離パラメータの値と単位の読み取りアクセス(R)が適合しない場合、エラーメッセージが作成されます。

注記

変更は、プログラムエンドまたはRESETまで有効なままです! プログラム開始後、チャンネルパラメータの設定が再度有効になります。

SPEED_LIMIT.ENABLE	速度リミットLook Aheadの選択/選択解除(0: 選択解除 1: 選択)	Boolean	0 , 1	R/W
SPEED_LIMIT.VEL_LIMIT	速度リミット値の定義	Integer	[%]	R/W
SPEED_LIMIT.TIME	「コーナー」までの/からの距離の単位定義(0: パス 1: 時間)	Boolean	0 , 1	R/W
SPEED_LIMIT.DIST_TO_CORNER	「コーナー」までのパス距離	Real	[mm]、[inch]	R/W
SPEED_LIMIT.DIST_FROM_CORNER	「コーナー」からのパス距離	Real	[mm]、[inch]	R/W
SPEED_LIMIT.TIME_TO_CORNER	「コーナー」までの時間距離	Real	[sec]	R/W
SPEED_LIMIT.TIME_FROM_CORNER	「コーナー」からの時間距離	Real	[sec]	R/W
SPEED_LIMIT.OVERRIDE_WEIGHT	速度リミット値の重み付け(0: 重み付けなし 1: オーバーライドによる重み付け)	Boolean	0 , 1	R/W

MAX_NC_BLOCKS_AHEAD	ブロック関連のデコーダリミット前処理: すべてのNCブロック	Integer	-	R/W
MAX_MOTION_BLOCKS_AHEAD	ブロック関連のデコーダリミット前処理: 動作NCブロックのみ	Integer	-	R/W
MAX_TIME_AHEAD	時間関連のデコーダリミット前処理	Real	[s]	R/W

CIRCLE_CP_ABS	円弧中心点の定義: G162の場合0 G161の場合1	Boolean	0 , 1	R
CIRCLE_CPC_ACTIVE	円弧中心点補正(G165)は有効か? 有効な場合、1 G164の場合0	Boolean	0 , 1	R

SEGMENTATION_ACTIVE	セグメント化(#SEGMENTATION...)は有効か? 有効な場合、1	Boolean	0 , 1	R
STROKE_DEF_ACTIVE	ストローク動作の定義(#STROKE DEF…)は有効か? 有効な場合、1	Boolean	0 , 1	R
PUNCH_ACTIVE	パンチング(#PUNCH...)は有効か? 有効な場合、1	Boolean	0 , 1	R
NIBBLE_ACTIVE	ニブリング(#NIBBLE...)は有効か? 有効な場合、1	Boolean	0 , 1	R

FRICTION_ACTIVE	摩擦補正値の記録(#FRICTION ON […]は有効か? 有効な場合、1 **[V2.11.2022.05以降]**	Boolean	0 , 1	R

CROSS_COMP_INIT[i] or CROSS_COMP_INIT.X	軸<i>または「X」のクロス補正は初期化されているか? yesの場合、1	Boolean	0 , 1	R
PLANE_COMP_INIT[i]または PLANE_COMP_INIT.X	軸<i>または「X」の平面補正は初期化されているか? yesの場合、1	Boolean	0 , 1	R
LEAD_COMP_INIT[i]または LEAD_COMP_INIT.X	軸<i>または「X」の送りねじエラー補正は初期化されているか? yesの場合、1	Boolean	0 , 1	R
TEMP_COMP_INIT[i]または TEMP_COMP_INIT.X	軸<i>または「X」の温度補正は初期化されているか? yesの場合、1	Boolean	0 , 1	R
CROSS_COMP_ACTIVE[i]または CROSS_COMP_ACTIVE.X	軸<i>または「X」のクロス補正は有効か? yesの場合、1	Boolean	0 , 1	R
PLANE_COMP_ACTIVE[i]または PLANE_COMP_ACTIVE.X	軸<i>または「X」の平面補正は有効か? yesの場合、1	Boolean	0 , 1	R
LEAD_COMP_ACTIVE[i]または LEAD_COMP_ACTIVE.X	軸<i>または「X」の送りねじエラー補正は有効か? yesの場合、1	Boolean	0 , 1	R
TEMP_COMP_ACTIVE[i]または TEMP_COMP_ACTIVE.X	軸<i>または「X」の温度補正は有効か? yesの場合、1	Boolean	0 , 1	R
FILE_EXIST	#FILE EXIST [▶ 332]によるチェックの結果 ファイルのチェックが正の場合、1	Boolean	0 , 1	R
MEAS_TYPE	現在有効な計測タイプ [▶ 59] **[V2.11.2022.03以降]**	Integer	-	L

14.4 ユーザ定義変数

ユーザ定義変数(V.P.、V.S.、V.L.)は、**#VAR**で開始され、**#ENDVAR**によって閉じられる宣言ブロック内のプログラム名の**後の**NCメインプログラムまたはサブプログラムで作成されます(最後に初期化されます)。

ユーザ定義変数V.P.、V.S.、およびV.L.にはREAL値を割り当てることができます。

旧バージョンとの互換性のために、V.PおよびV.S変数は、宣言ブロックの外部でも作成できます。この場合、この作成は、その変数への最初の書き込みアクセス中に、暗黙的に行われます。ただし、V.L変数と配列変数は、常に宣言ブロックで作成する必要があります。読みやすくするために、パラメータ配列の初期化は文字「\」を使用して複数のNCブロックに渡って記述することができます。作成には以下の構文を使用します。

#VAR	宣言ブロックの開始
:	
:	宣言および初期化の部分
:	
#ENDVAR	宣言ブロックの終了

プログラミング例

```
%prog_name
:
#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]
  V.P.VAR_1
  V.L.VAR_1
  V.S.VAR_1
#ENDVAR
```

配列変数に対するアクセスは、インデックス0から開始されます! 上記の例に従えば、アクセスV.L.MY_ARRAY[0][5]は値15を与えます。

ユーザ定義変数と配列変数は、NCプログラムで削除することもできます。そのために、次の構文の#DELETEコマンドが提供されています。

#DELETE V. *<name>* {,**V.***<name>*}

プログラミング例

```
#DELETE V.P.ARRAY_1, V.L.MY_ARRAY, V.P.VAR_1, V.L.VAR_1, V.S.VAR_1
```

さらに、SIZEOFおよびEXISTファンクション(「演算式<expr> [▶ 25]」の章を参照)は、配列サイズ変数の寸法の決定と、ユーザ定義変数の存在のチェックのために使用されます。

プログラミング例

ユーザ定義されたV.S.配列変数(任意の有効なインデックス)に対するEXIST要求によって、この変数が以前のパートプログラムで既に定義されているのか、あるいはこの変数を定義する必要があるのかどうかがチェックされます。

```
...
N10 $IF EXIST[V.S.EXAMPLE[0]] == TRUE
N20  V.S.EXAMPLE[2] = 10 (enter value for V.S-variable[2])
N30 $ELSE
N40   #VAR
N50     V.S.EXAMPLE[5] = [1,2,3,4,5 ]
N60   #ENDVAR
N60 $ENDIF
...
M30
```

14.4.1 グローバル、パートプログラム終了後に有効でない(V.P.)

コード「V.P. ...」を使用して、個々の変数を定義し、値をそれらの変数に割り当てることが可能です。変数は現在のパートプログラムでグローバル(つまり、作成後はパートプログラムとそのすべてのサブプログラムの任意のポイントで既知)ですが、プログラム終了後は有効ではありません。V.P.変数にREALで値を割り当てることができます。

構文:

V.P. < *FREE_DEF*>	グローバル変数、プログラム終了後に有効でない

<FREE_DEF> ここで*FREE_DEF*は任意に選ばれる名前であり、一定の最大数の文字(ブランク、タブ、注釈、比較演算子、数学演算子、括弧を除く)で構成することができます。

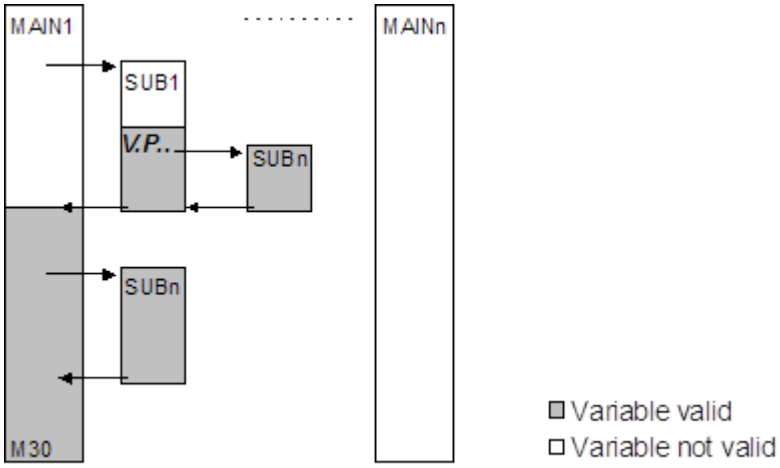
プログラミング例

V.P.VAR_5の作成と値20による初期化。その後、X軸方向の直線補間がこの変数(X20)の値を用いて実行されます。

```
:
#VAR
  V.P.VAR_5
#ENDVAR
:
N10 V.P.VAR_5 = 20
N20 XV.P.VAR_5
:
or
:
#VAR
  V.P.VAR_5 = 20
#ENDVAR
```

```
:
N20 XV.P.VAR_5
:
```

プログラム終了後に有効でない、ユーザ定義された変数の最大数は一定です[6]-6-21。プログラムの開始時に、これらのV.P.変数の名前と値がすべて削除されます。



14.4.2 グローバル、パートプログラム終了後に有効(V.S.)

コード「V.S. ...」を使用して、個々の変数を定義し、値をそれらの変数に割り当てることが可能です。これらの変数には、すべてのプログラムレベルおよびすべての(メイン)プログラムで、同じ名前でアクセスすることができます。リセット後も、これらの変数は有効なままです。変数の値は、上書きによってのみ変更することができます。変数自体の削除は、NCコマンド#DELETEまたは制御システムの再起動/起動によってのみ可能です。V.S.変数にREALで値を割り当てることができます。

構文:

V.S. < *FREE_DEF*>	グローバル変数、プログラム終了後に有効

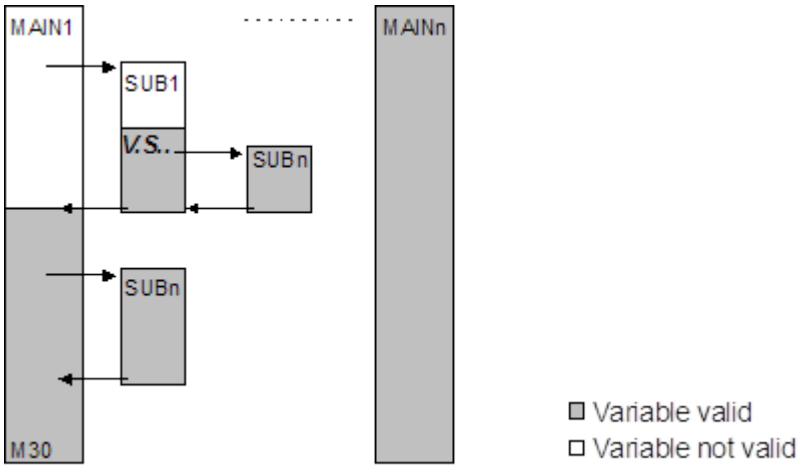
<FREE_DEF> ここで*FREE_DEF*は任意に選ばれる名前であり、文字(ブランク、タブ、注釈、比較演算子、数学演算子、括弧を除く)で構成されます。

プログラミング例

配列変数V.S.VAR[5]の作成と初期化。その後、X軸方向の直線補間が変数V.S.VAR[4] (X20)を用いて実行されます。

```
:
#VAR
  V.S.VAR[5] = [5,10,10,15,20]
#ENDVAR
:
N20 XV.S.VAR[4]
:
```

ユーザ定義されるグローバル変数の最大数は一定です[6]-6.22。



14.4.3 ローカル、パートプログラム終了後に有効でない(V.L.)

コード「V.L. ...」を使用して、個々の変数を定義し、値をそれらの変数に割り当てることが可能です。これらの変数は、現在のサブプログラムとすべての呼び出されるサブプログラムでローカルです。これらの変数は、定義するサブプログラムレベルから離れるときに削除されます。V.S.変数にREALで値を割り当てることができます。

構文:

V.L. < *FREE_DEF*>	ローカル変数、プログラム終了後に有効でない

<FREE_DEF> ここで*FREE_DEF*は任意に選ばれる名前であり、文字(ブランク、タブ、注釈、比較演算子、数学演算子、括弧を除く)で構成されます。

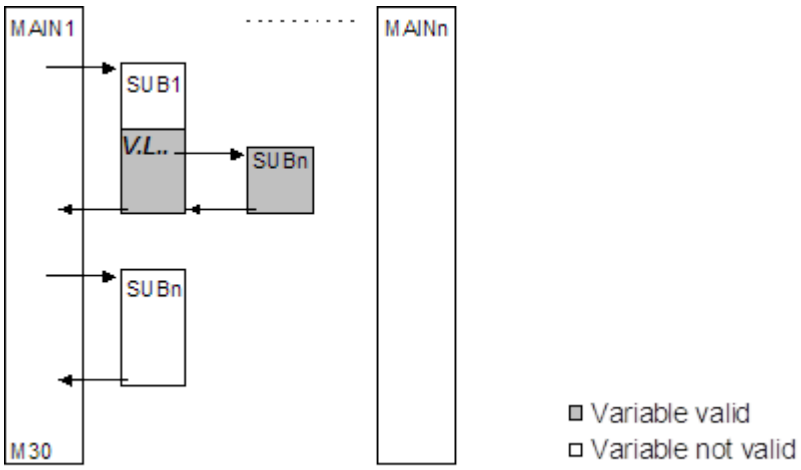
プログラミング例

V.L.LOC_VARの作成と値10による初期化。その後、X軸方向の直線補間がこの変数(X10)の値を用いて実行されます。

```
:
#VAR
  V.L.LOC_VAR
#ENDVAR
:
N10 V.L.LOC_VAR = 10
N20 XV.L.LOC_VAR
:
or
:
#VAR
  V.L.LOC_VAR = 10
#ENDVAR
:
```

```
N20 XV.L.LOC_VAR
:
```

ユーザ定義されるローカル変数の最大数は一定です[6]-6.23。プログラムの開始時にこれらのV.L.変数の名前と値がすべて削除されます。



14.5 外部変数(V.E.)

コマンド「V.E. ...」を使用して、NCプログラムで外部アドレスへの書き込みまたは外部アドレスからの読み取りを行うことができます。並行プロセスからのNCプログラムの解読処理を制御できるように、この変数タイプへの外部アクセスを特殊なインターフェイス経由で実行することができます。

NCチャンネルからの外部変数へのアクセス(インターポレータによる同期アクセスまたは解読処理による非同期アクセス)が可能です。

設定と初期化の詳細については、外部変数の取扱説明書を参照してください[8]。

プログラミング例

```
N100 $IF V.E.EXT1 >= 100          (Corresponding to the value of V.E.EXT1)
                                  (branching off is done into various cases)
N110 G01 X100 Y100 F1000

N120 $ELSE
N130 G01 X100 YV.E.EXT1 F1000     (Linear interpolation in Y-direction)
                                  (with the value from V.E.EXT1)
N140 ENDIF

N150 V.E.EXT1 = V.A.ABS.X         (To the external variables the absolute)
                                  (X-coordinates are assigned)
N160 G01 XV.E.EXT2                (Linear interpolation in X-direction)
                                  (with the value from V.E.EXT2)
```

制御システムの起動後、設定されたV.E.変数がゼロで初期化されます。

その後、必要に応じてNCプログラムのV.E.変数を#INITコマンドによって再び初期化することができます。このコマンドの後に複数のV.E.変数を続けることができます。これらの変数は完全に初期化されます。個々のV.E.変数の他に、完全なV.E:配列とV.E.構造体およびV.E.構造体の下位要素を初期化することもできます。

#INIT V.E. *<name>* {,**V.E.***<name>*}

注記

アクセス権

変数が読み取りアクセス権しか持っていない場合、#INITコマンドを用いて変数を初期化することはできません。読み取りのみが可能な少なくとも1つの下位要素を含むV.E.構造体にも、同じことが当てはまります。

注記

同期V.E.変数

As V.E.構造体に**1つ**の同期変数が含まれている場合、#INITを用いた初期化操作全体が同期します(つまり、インターポレータコンテキストでのみ実行されます)。そのため、可能な下位要素がこの操作によっても影響を受けます。理由は、以降の読み取りアクセス操作でまだ再初期化されていない可能性があるためです!
この場合に完全な同期を得るには、ユーザが#INITコマンドの前に手動でプログラム#FLUSH WAITコマンドを使用する必要があります。
ヒント: #INITコマンドを使用する際は、すべての要素が完全に同期的または非同期的であるようにV.E.構造体変数を作成することをお勧めします。

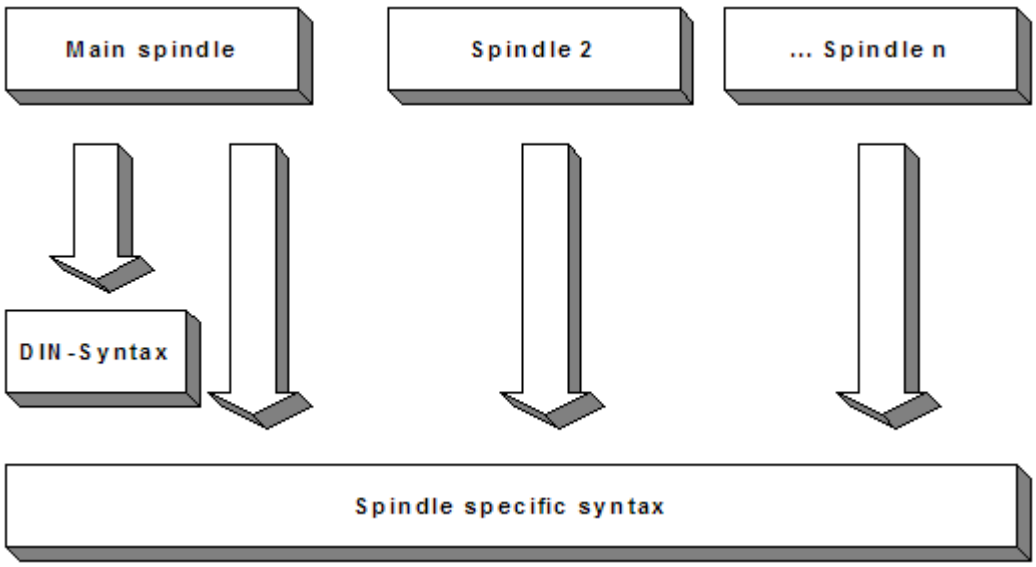
プログラミング例

```
Initialization of single V.E. variables:
Nxx #INIT V.E.EXT1, V.E.EXT2, V.E.EXT3


Initialization of V.E. array variable:
Nxx #INIT V.E.ARRAY1


Initialization of specific V.E. array variables:
Nxx #INIT V.E.ARRAY1[5], V.E.ARRAY1[8], V.E.ARRAY1[20]


Initialization of a V.E. structure variable:
Nxx #INIT V.E.STRUCT1


Initialization of specific elements of a V.E. structure variable:
Nxx #INIT V.E.STRUCT1.NBR_POINTS, V.E.STRUCT1.POINTS


Combined initialization of V.E. variables:
Nxx #INIT V.E.EXT2, V.E.ARRAY1[5], V.E.STRUCT1.POINTS
...
```

15 スピンドルプログラミング

スピンドルプログラミングは、ISOまたはISO+拡張で定義されている従来型の標準的な構文に従って実行されます。これは、特に互換性および一部の標準機能(旋回、タッピング、ギヤ切替など)に関連する理由で必要とされます。

柔軟なスピンドルプログラミングにおける新しい機械コンセプトと製造技術の要件に適合するために、チャンネル上の各スピンドルに対応したプログラミングのオプションを備えています。この構文により、1つのNCブロックで同時にマルチスピンドルシステムで互いに独立して複数のスピンドルを指定することが可能ですP-CHAN-00082、[6]-8.8。この場合、標準構文とスピンドル特有の構文の両方で、一度にプログラムできるスピンドルは1つ(いわゆる「メインスピンドル」)のみであることに注意する必要があります。その他のスピンドルは、すべてスピンドル特有の構文によってのみ指定することができます(「スピンドル特有の構文でのプログラミング [▶ 476]」の章)。



スピンドルとメインスピンドルは、チャンネルパラメータリスト[1]-3で定義されています。この設定は制御システムの起動後に利用可能です。メインスピンドルは、NCコマンド(#MAIN SPINDLE、0章)を用いてNCプログラムで変更することができます。

下記の表は、どのNCコマンドをDIN規格でのみスピンドルプログラミングと組み合わせて使用する必要があるのか、またどのNCコマンドが拡張されたスピンドル特有の構文内でも許可されるのかを示しています。

表 24: 表14-1: すべてのスピンドル関連コマンドの概要

説明	DIN規格	スピンドル特有の構文
スピンドルMファンクション	M3、M4、M5、M19	M3、M4、M5、M19
速度	S	REV
位置	S.POS	POS
ユーザ定義の M/Hファンクション	Mxx/Hxx	Mxx/Hxx
ギヤ切替(機械的)	M40-M45	
ねじ切り	G33	
タッピング	G63	
旋回	G95、G96、G97、G196	
C軸	#CAX	
ギヤ切替(新しいデータ)	G112	
原点復帰	G74	G74
オーバーライド100%	G167	G167
軸の 要求		CALLAX
スピンドル軸の解放		PUTAX
新しいツール動的データの 引き継ぎ		GET_DYNAMIC_DATA
フィードフォワード制御		G135、G136、G137
送りリンク		FEED_LINK

15.1 スピンドルの設定

15.1.1 軸パラメータ

スピンドル特有のパラメータを入力するために、軸チャンネルパラメータリスト[2]を閉ループ制御対象スピンドルと開ループ制御対象(PLC)スピンドルの両方に定義する必要があります。

15.1.2 チャンネルパラメータ

パートプログラムでスピンドルをプログラムするために、チャンネルパラメータ[1]で行われるその他の入力が必要です。

この場合、このチャンネルによって指定しなければならない各スピンドルを宣言する必要があります。そのために、文字列(軸名)および対応する論理軸番号を各スピンドルに定義します。スピンドルの軸名は自由に選ぶことができますが、「S」で始まる必要があります(例えば、S、S_MAIN、S1、SPINDEL_1)。

また、スピンドルMファンクション(M3/M4/M5/M19)とSワードについて、同期をスピンドル特有に定義する必要があります。そのために、M3/M4/M5/M19の適切な意味を切り替える必要があります(P-CHAN-00098)。

スピンドルMファンクションがNCブロックでプログラムされている場合、Sファンクションの同期方法は効果がありません。同期は、スピンドルMファンクションの設定内容に従ってのみ行われます。適用される優先順位:

M19 > M3/M4/M5 > S

シミュレーションモード「製造時間計算」でスピンドルを許可する必要がある場合、そのために必要なデータをスピンドル特有に前もって割り当てることもできます。

標準プログラミングとの適合性を維持するために、スピンドルをいわゆるメインスピンドルとして宣言する必要があります(P-CHAN-00051とP-CHAN-00053)。次に、このメインスピンドルを特定の標準機能(タッピングやギヤ切替など)とともに、従来型のDIN規格に沿ってプログラムすることができます。システムに存在するスピンドルが1台のみの場合であっても、メインスピンドルとして設定する必要があります。

フラグを設定すると、オプションのギヤ切替をメインスピンドルに対して有効にすることもできます(P-CHAN-00052)。

チャンネルパラメータ[1]で定義された設定が、制御システムの起動後に利用できる初期設定の割り当てです。

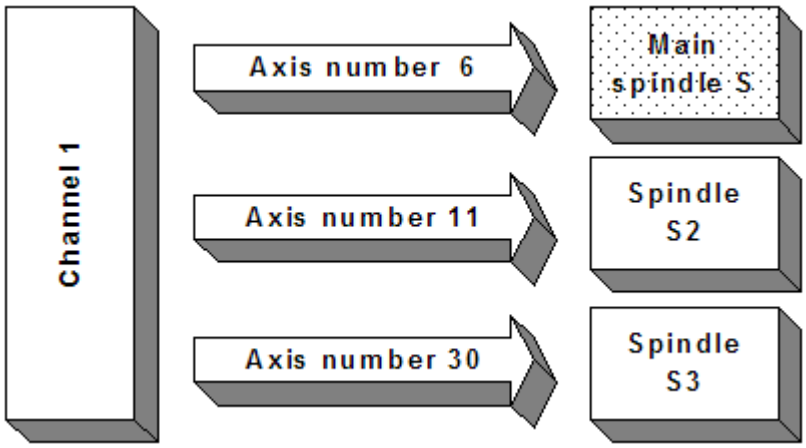
例1:

3台のスピンドルを備えるシングルチャンネルシステムの設定。軸番号6のスピンドルがメインスピンドルであること。このスピンドルのギヤ切替は無効です。

チャンネルパラメータリスト[1]からの抽出:

```
:
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
:
```

- 起動後は、論理軸番号6のスピンドルがメインスピンドルです。このスピンドルはスピンドル名「S」によって指定され、従来型のDIN規格に沿って、またはスピンドル特有の構文でプログラムすることができます。スピンドル「S2」と「S3」はスピンドル特有の構文でのみプログラムすることができます。



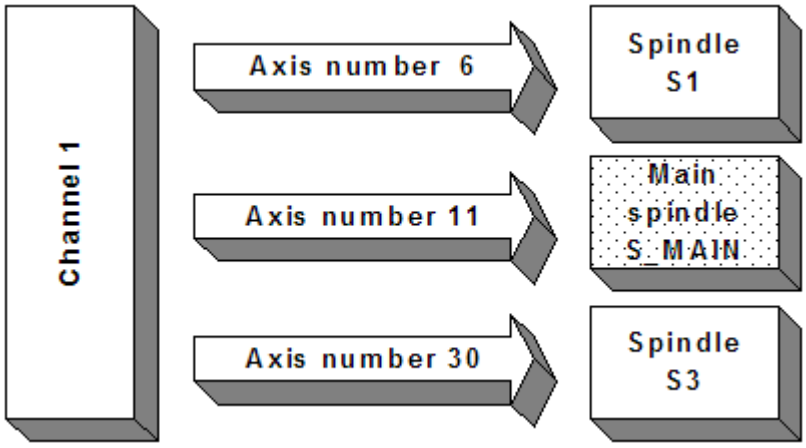
例2:

3台のスピンドルを備えるシングルチャンネルシステムの設定。軸番号11のスピンドルがメインスピンドルであること。このスピンドルのギヤ切替は無効です。

チャンネルパラメータリスト[1]からの抽出:

```
:
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
:
```

- 起動後は、論理軸番号11のスピンドルがメインスピンドルです。このスピンドルはスピンドル名「S_MAIN」によって指定され、従来型のDIN規格に沿って、またはスピンドル特有の構文でプログラムすることができます。スピンドル「S1」と「S3」はスピンドル特有の構文でのみプログラムすることができます。



例3:

合計3台のスピンドルを備える2チャンネルシステムの設定:

チャンネル1: 3台のスピンドル。軸番号11のスピンドルがメインスピンドルであること。

チャンネルパラメータリスト[1]からの抽出:

```
:
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
:
```

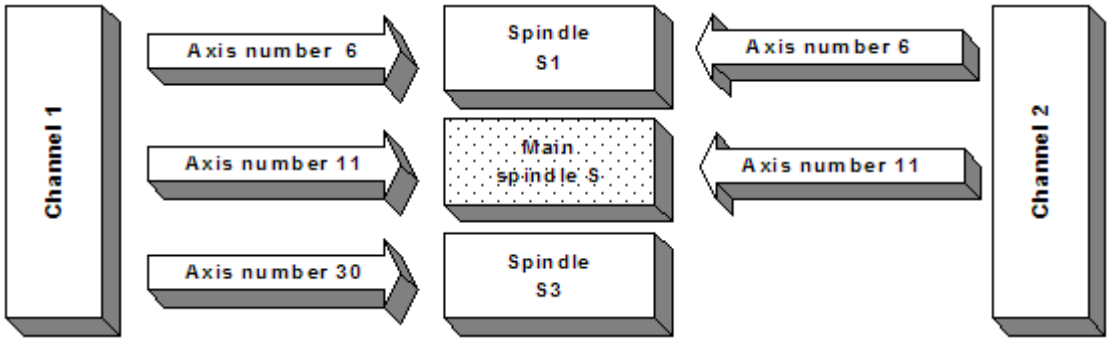
チャンネル2: 2台のスピンドル。軸番号11のスピンドルがメインスピンドルであること。

チャンネルパラメータリスト[1]からの抽出:

```
:
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
```

- 起動後は、両方のチャンネルによってスピンドル名「S」を使用して、論理軸番号11のスピンドルをメインスピンドルとして指定することができます。このスピンドルは、従来型のDIN規格に沿って、またはスピンドル特有の構文でプログラムすることができます。スピンドル「S1」は、スピンドル特有の構文で両方のチャンネルからプログラムすることもできます。スピンドル「S3」はチャンネル1でのみ利用できます。このスピンドルはチャンネル2では知られていません。



15.2 DIN規格に沿ったプログラミング

15.2.1 スピンドルMファンクション

15.2.1.1 スピンドル移動(M3、M4、M5)

M03	スピンドル回転時計回り(cw)	(モーダル)
M04	スピンドル回転反時計回り(ccw)	(モーダル)
M05	スピンドル停止	(モーダル)

スピンドルMファンクションM03/M04/M05はスピンドルの動作モードを定義します。これらのファンクションはSワードと組み合わせて使用する必要があります(「スピンドル速度(Sワード) [▶ 452]」の章)。これらのファンクションには停止操作があり、NCブロックでそれぞれ単独でのみプログラムすることができます。

M03またはM04がプログラムされており、有効な速度が設定されている場合に、スピンドル回転が有効です。

M05はスピンドル回転を停止します。このスピンドルMファンクションは、制御システム起動と最初のプログラム開始後の初期設定のスピンドルモードです。

プログラム終了時にM05が設定されていない場合、スピンドルは回転し続けます。

プログラミング例

```
N10             S1000  (Speed 1000 U/min. is stored, no spindle rotation)
                       (because M05 is default)
N20      M03           (Spindle rotation cw with 1000 U/min.)
N30      M04           (Spindle rotation ccw with 1000 U/min.)
N40             S500    (Spindle rotation ccw with 500 U/min.)
N50      M05    S300    (Spindle rotation, speed 300 U/min. is stored)
N60      M04           (Spindle rotation ccw with 300 U/min.)
N70      M05           (Spindle stops)
N80      M03    S1000   (Spindle rotation cw with 1000 U/min.)
N90      M30           (Program end
```

チャンネルパラメータリスト[1]からの抽出:

M3/M4/M5に関して同期モードをスピンドル特有に定義する必要があります。同期モード「0」(NO_SYNCH)の場合、Mファンクションは実行されません。

```
:
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 スピンドルの位置決め(M19、*.POS)

M19	スピンドルの位置決め	(ノンモーダル)
*.POS	スピンドル位置	(モーダル)

スピンドルの位置決めを以下の構文で表すことができます。

M19 [*spindle_name*.**POS**[=] <*position*>] [**M03** | **M04**] [*spindle_name* <*speed*>]

M19 スピンドル位置決め

spindle_name [1]-3に準拠するスピンドル名

<*position*> スピンドル位置を指定するための数式

<speed> スピンドル速度を指定するための数式

同じNCブロックにおけるM03/M04またはスピンドル速度はオプションです。ただし、有効なスピンドル速度(> 0)を設定する必要があります。

スピンドル位置はモーダルであり、M19を再びプログラムする場合に再指定する必要はありません。以前にスピンドル位置がプログラムされていない場合、スピンドルは初期設定位置「ゼロ」に移動します。

スピンドルが回転していない場合、最短の移動経路で位置決めが実行されます。

M19によるスピンドルの位置決めは、**閉ループ位置制御対象**スピンドルについてのみ可能です。

プログラミング例

以下のそれぞれの例でスピンドルが位置180に位置決めされます。「=」文字はオプションです。

```
M19 S.POS180

M19 S.POS 180

M19 S.POS=180

M19 SPINDEL.POS=180

M19 S1.POS=180
```

位置決め中はスピンドルが回転しません。最短の移動経路が計算されます。

```
N10 M05 S100            (Spindle stops, speed 100 U/min. is stored)
N20 M19 S.POS180        (Positioning with 100 U/min to position 180)
                        (The direction of rotation results from the)
                        (shortest movement distance)
N30 M19 S200 S.POS90    (The direction of rotation results from the)
                        (shortest movement distance)
                        (Positioning with 200 U/min ccw to position 90)
```

チャンネルパラメータリスト[1]からの抽出:

M19では、同期モードをスピンドル特有に定義する必要があります。同期モード「0」(NO_SYNCH)の場合、Mファンクションは実行されません。

```
:
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 スピンドル速度(Sワード)

<*spindle_name*> [=] <*speed*> (モーダル)

<*spindle_name*> [1]-3に準拠するスピンドル名

<*speed*> スピンドル速度を指定するための数式

設定中にメインスピンドルにチャンネルパラメータリストの文字列を割り当てることができます(P-CHAN-00053)。曖昧さを避けるために、複数の文字を含むすべてのスピンドル名と速度入力の間に等号(=)がなければなりません。

値をSワードに直接、またはパラメータを用いて割り当てることができます。また、10進数を使用することもできます(REAL型)。

Sワードの場合、スピンドルMファンクションと組み合わせた以下の種類の使用を区別する必要があります。

1. M03、M04、M19と組み合わせたSワード:
   Sワードまたはそれに使用される文字列をM03/M04またはM19と組み合わせてプログラムすると、Sワードに続く値がスピンドル速度として解釈され、スピンドルに出力されます。
2. M05と組み合わせたSワード:
   Sワードに続く値が、M05と組み合わせて作業データでスピンドル速度として受け入れられますが、スピンドルには出力されません。

Sワード単独では、NCプログラムでの動作を生成しません。スピンドルモードM03/M04/M19は、このために既知であることが必要です。そのため、Sワードが設定される(> 0、送り速度および動作すべき軸が指定されている場合にのみ動作が生じるG01/G02/G03と類似)まで、M03/M04/M19のプログラミングによって動作が起こることはありません。

S値が負であると、エラーメッセージが出力されます。

ネジ穴から引き抜くときに回転方向の反転をトリガするため、負のS値はG63 (タッピング)と連動してのみ可能です。

プログラミング例

スピンドルSを含むプログラミング:

```
N10  S300          (Speed 300 U/min. is stored)
N20  M04           (Spindle rotation ccw with 300 U/min.)
N30  M03 S1000     (Spindle rotation cw with 1000 U/min.)
```

```
N40  S500          (M03 active, Spindle rotation cw with 500 U/min.)
N50  M05 S100      (Spindle stops, speed 100 U/min. is stored)
N60  M04           (Spindle rotation ccw mit 100 U/min.)
N70  M05           (Spindle stops)
N80  M30           (Program end)
```

チャンネルパラメータリスト[1]からの抽出:

Sワードでは、同期モードをスピンドル特有に定義する必要があります。Sワードを無視することはできないため、同期モードが「0」(NO_SYNCH)の場合はエラーメッセージが出力されます。

```
:
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
:
```

プログラミング例

```
N10  M03 S100                (Spindle rotation cw with 100 U/min.)

N20  M19 S.POS90             (With 100 U/min cw to position 90)

N30  M04                     (Spindle rotation ccw with 100 U/min.)

N40  M19 S200 S.POS 180      (With 200 U/min ccw to position 180)

N50  M05 S150                (Spindle stops, speed 150 U/min. is stored)

N60  M19 S.POS=135           (Positioning with 150 U/min. with shortest)
                             (way to position 135.)

N70  M03 S300                (Spindle rotation cw with 300 U/min.)

N80  M19 S200 S.POS270       (With 200 U/min cw to position 270)

N90  M03 S400 S.POS45        (Spindle rotation cw with 400 U/min., )
                             (position 45 is stored)

N100 M19                     (With 400 U/min. cw to position 45)

N110 M04 S800                (Spindle rotation ccw with 800 U/min.)

N120 S1200                   (Spindle rotation ccw with 1200 U/min.)

N130 M5                      (Spindle stops)

N140 M03                     (Spindle rotation cw with 1200 U/min.)

N150 M19                     (With 1200 U/min. cw to position 45)
```

15.2.3 主軸のギヤ切替(M40 - M45)

主軸伝達の変更はM40...M45を用いてプログラムします。これらのMファンクションは最大6つのギヤ段数を定義します。

Mファンクションを同じNCブロックでスピンドル速度および回転方向のMファンクションとともにプログラムすることができます。ファンクションM40～M45は、ギヤ段数の選択と機械的ギヤ切替の始動に使用します。

構文:

M40 | **M41** | **M42** | **M43** | **M44** | **M45** [<*spindle_name*><*expr*>] [**M03** | **M04**] (モーダル)

M40 ~ M45 ギヤ段数1 ~ 6

<spindle_name><expr> [1]に準拠するスピンドル名と数式で構成されるスピンドル速度

プログラミング例

```
S800 M41 M03    (Speed 800 U/min, gear range 2, rotation cw)
```

- 解読ファンクションM40 ~ M45はモーダルであり、ブロックの開始時に有効になります。M40 ~ M45は互いに打ち消し合います。
- このシステムでは、最大6つのギヤ段数(M40 ~ M45)の定義が可能です。各ギヤ段数について[1]-4の「速度レンジの表」で最小速度と最大速度が定義されています(単位 = rpm)。
- 閉ループ位置制御対象主軸の10 V 出力の最大速度が、マルチゲイン係数P-AXIS-00128およびP-AXIS-00129によってアナログドライブで定義されます。
- ギヤ段数が自動選択されるシステムでは、速度Sをプログラムすることによってのみこれが決められます。その後、M40 ~ M45をプログラムする必要はありません。
- M40 ~ M45は、閉ループ位置制御対象スピンドルの場合にのみプログラムすることができます。
- NCカーネルは常にギヤ切替を最小限に抑えようと試みます(例えば、新しい速度を現在のギヤ段数で使用できる場合、ギヤ切替がM40 ~ M45によって明示的にプログラムされていても、ギヤ切替は停止します)。

チャンネルパラメータリスト[1]からの抽出:

- MファンクションM40 ~ M45の定義と同期モードの定義。

```
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
:
```

- ギヤ切替の有効化:

```
:
main_spindle_gear_change                1   0: OFF   1: ON
:
```

- スピンドル伝達の設定(検索方向、速度レンジ):

```
:
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)
#
:
```

プログラミング例

- 自動ギヤ段数決定: ON

```
:
spindel[0].autom_range                              1
:
```

```
NC-program:
S650 M03              OK, M41□ SPS
S750                  OK, no changing, M41 already selected
S950                  OK, automatic changing, M42 □  SPS
S1050                 OK, no changing, M42 already selected
S750                  OK, automatic changing, M41 □  SPS
S500                  OK, no changing, M41 already selected
S350                  OK, no changing, M41 already selected

S8000                 Error, speed too high

A programmed gear stage is always checked:
M41 S750              OK, " automatic" changing, M41 □  SPS

..but

M40 S750              Error, wrong gear stage
```

プログラミング例

- 自動ギヤ段数決定: OFF

```
:
spindle[0].autom_range                              0
:

NC-program:
M41 S650 M03          OK, M41□ SPS
M41 S750              OK, no changing, M41 already selected
M42 S950              OK, changing, M42 □  SPS
```

```
M42 S1050              OK, no changing, M42 already selected
M41 S750               OK, changing, M41 □  SPS
M41 S500               OK, no changing, M41 already selected
M41 S350               OK, no changing, M41 already selected

M41 S200               Error, program different gear stage (M40)
S950                   Error, no gear stage (M42) programmed
```

15.2.4 旋回ファンクション

15.2.4.1 カッター径補正(G40/G41/G42)

G40	SRK選択解除	(モーダル、初期設定)
G41	輪郭の左のSRK	(モーダル)
G42	輪郭の右のSRK	(モーダル)

工具先端径補正(SRK)は、G17/G18/G19を使用して選択される加工平面で動作します。この平面では、一方の軸をモード「面旋回」で、他方の軸をモード「縦旋回」で操作する必要があります。

工具補正値として、Dワードで保存されたデータセットが使用されます。旋回工具の場合、加工平面(面の軸、縦軸)に対するカッター刃の向きを追加識別コード1～9 (図14-1を参照)によって指定する必要があります。

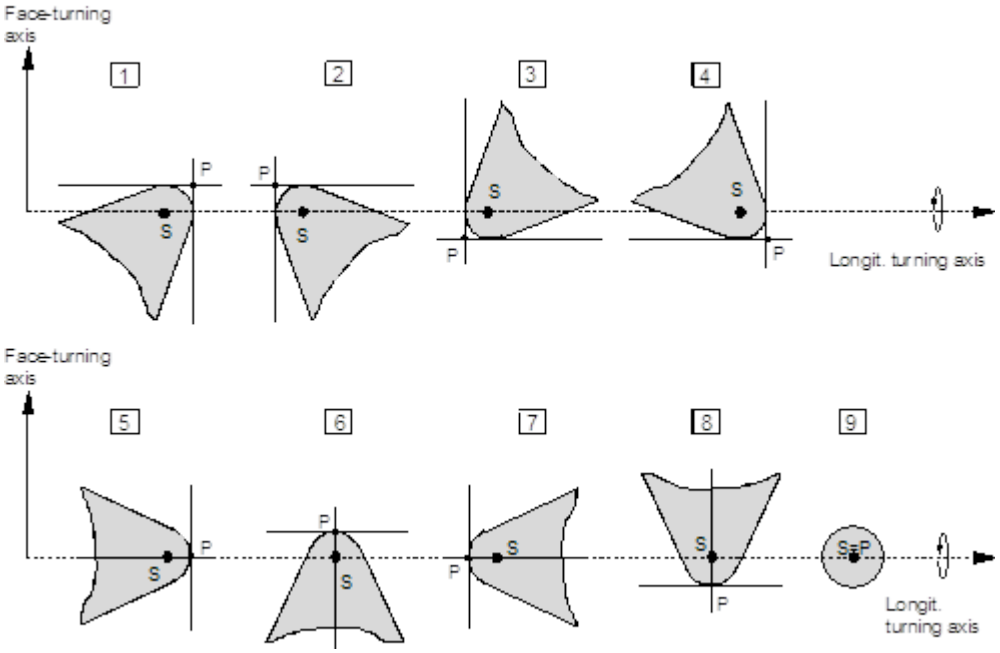


図 102: 図14-1: 加工平面に対するカッター刃の向き。

次に、一般的な旋回工具が以下の値/パラメータによって特性化されます。

- Tool type 1 (旋回工具)
- SRK-orientation 1 ~ 9
- Tool radius 工具先端半径
- Tool length --
- Tool offset (図14-2を参照)

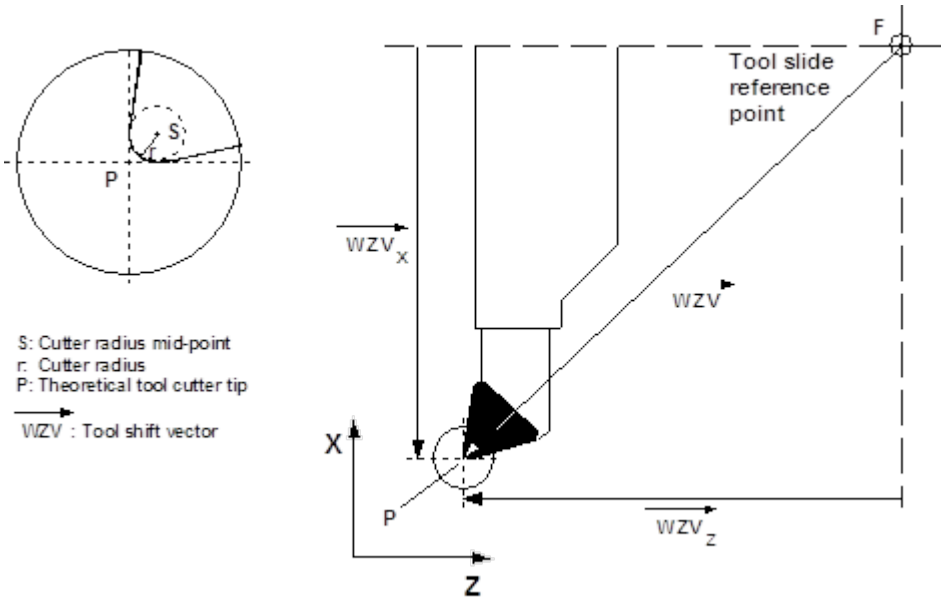


図 103: 図14-2: 工具補正のための工具計測。

工具軸オフセットを指定する際は、加工平面における工具シフトベクトルの成分が関わるため、オフセットの数学記号に注意する必要があります。図14-2で示されている旋回工具の例では、X軸とY軸両方の方向のオフセットが負の(数学) 記号を持っています。

旋回工具とフライス工具の交換は、G41またはG42が選択されている場合に許可されます。絶対プログラミングG90の場合、新しい工具の現在の軸オフセット値が、工具のタイプに従って次の移動ブロックで考慮に入れられます。

15.2.4.2 直径プログラミング(G51/G52)

G51	直径プログラミングの選択	(モーダル)
G52	直径プログラミングの選択解除	(モーダル、初期設定)

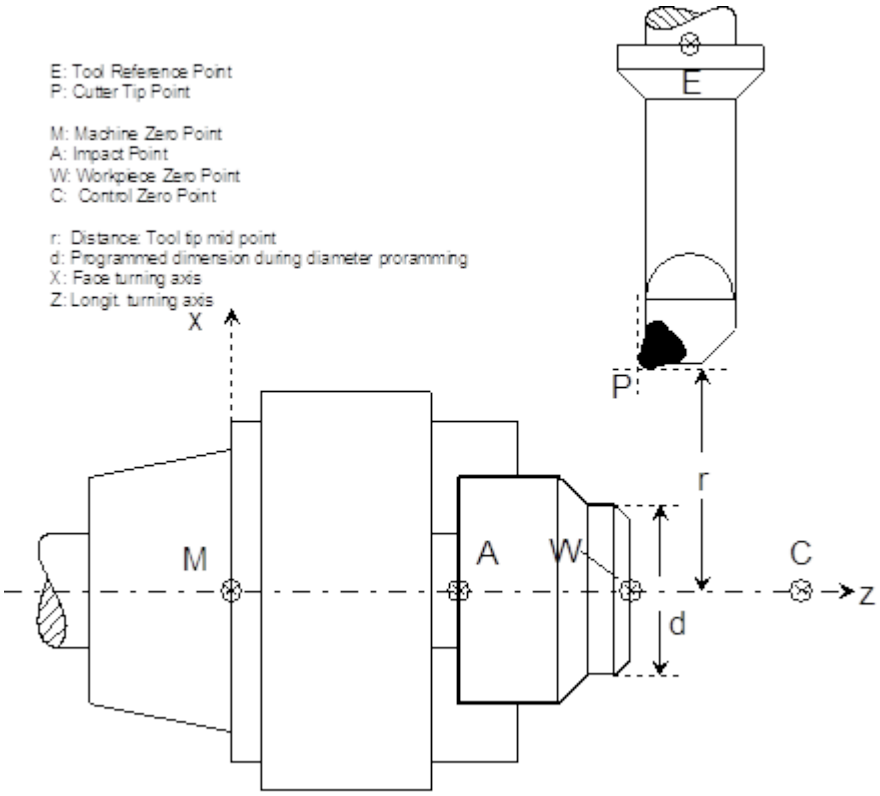


図 104: 図14-3: 旋回の基準点および直径プログラミング

直径プログラミングが選択されている場合、面旋回軸の移動ブロックにおける位置の値が、旋回の中心点に対する直径値として解釈されます。

面旋回軸のプログラム座標が実際にワーク直径と一致するのは、面旋回軸の原点が旋回の中心点に位置する場合に限られます(オフセットが直径として有効であるかどうかという事実に関係なく。後述の例を参照)。

機械データを通じて、モード「面旋回」で軸に対してパラメータ設定を行うことができます。

- 絶対プログラミング(G90)による直径プログラミング(P-AXIS-00058)。
- 相対プログラミング(G91)による直径プログラミング(P-AXIS-00059)。
- G92と直径の原点シフト(P-CHAN-00091)

直径座標を負の値でプログラムしたり、ミラーリングしたりすることはできません。

G51は、モード「面旋回」で操作されるすべての軸に作用します。ただし、選択時に正確に1つの面旋回軸が加工平面に存在している必要があります(G17、G18、G19)。

中心点の座標は直径でプログラムされません。

直径プログラミングの選択解除はファンクションG52を使用して行います。

プログラミング例

```
Conditions:     (Facing turning axis, translatory, P-AXIS-00015)
                (Diameter prog. with G90, P-AXIS-00058)
```

```
                (G92, G54.. take effect as a diameter, P-CHAN-00091)

N10 G90 G01 F1000

N20 G51 X80     (Diameter 80mm)

N30 G92 X10     (BZV with 10mm as a diameter)

N40 X0          (prog. position 0 + BPV => Diameter 10mm)

N50 G91 X50     (relative 50mm, not in the diameter)

N60 G90 X-20    (Error: negative diameter)

N70 G21 X30     (Error: During diameter prog. the coordinates of the)
                (face-turning axis must not be mirrored)
N75 G22 X30     (Mirroring on face-turning axis permissible)

N80 G52         (Deselection of diameter programming)

N90 M30
```

15.2.4.3 1回転当たりの送り速度(G95)

G95	送り速度(mm/回転)	(モーダル)

有効なG95による旋回中に、スピンドル速度(rpm)に関係なくFワード(mm/rev)を使用して一定の切りくず厚さを調整することができます。

これにより、軸の送りが位置制御対象スピンドルの回転速度(rpm)に関連付けられます。これは、プログラムされたGファンクションと組み合わせた場合にのみ有効です。そのため、G95からG94またはG93への変更の場合、G95で有効なFワードは受け入れられません。

プログラミング例

```
N10 F1000 X100 M3 S1200   (Feed 1000 mm/min (G94))

N20 G95 F1.5              (Feed 1.5 mm/rev., spindle speed 1200 1/min)

N30 G94 X50               (Feed 1000 mm/min from N10 is valid)

N40 G93 F20 X20           (Machining duration 20 s)

N50 G95 Y200 S2000        (Feed 1.5 mm/rev. from N20 valid,)
                          (spindle speed 2000 1/min)

N60 M30
```

15.2.4.4 一定切削速度(G96/G97/G196)

G96	一定切削速度の選択	(モーダル)
G97	一定切削速度の選択解除、スピンドル速度の選択	(モーダル、初期設定)
G196	G96における最大スピンドル速度	(G196はモーダルでない、最大速度モーダル)

GファンクションG96/G97/G196を使用して、Sワード(またはS文字列)の解釈を任意に変更することができます。

G96 S (m/min) (切削速度)

G97 S (1/min) (スピンドル速度)

G196 S (1/min) (G96実行中の最大スピンドル速度)

G96の選択時、スピンドルの開始回転数(rpm)が、プログラムされた切削速度および工具先端から旋回の中心点までの距離によって与えられます。この距離は、最後の(現在のNCブロックにおいてではなく)プログラムされた位置および面旋回軸の基準点オフセットから得ることができます。現在の加工平面(G17、G18、G19)では、正確に1つの面旋回軸が存在する必要があります。

Sワードを使用してG96向けにプログラムされた切削速度は、G97による選択解除まで有効です。G96では、一定切削速度が最初のSワードのプログラミングによって有効になります。

G196による最大スピンドル速度の指定はオプションであり、G96実行中にのみ有効です。

旋回の中心点の近くで、プログラムされた最大スピンドル速度(rpm) (G196)または割り当てられた軸パラメータP-AXIS-00212で指定された最大スピンドル速度によって、一定切削速度の限度が決まります。

G97による選択解除時、最後に設定されたスピンドル速度が維持されます。

早送りモード(G00)での面旋回軸の移動ブロックはG96の中断につながるため、工具の位置決め中の不要なrpm値変更を防止することができます。G01、G02、またはG03による次の移動ブロックによってG96の停止が中止されます。

プログラミング例

```
(X ... Face-turning axis)

N10 M03 S1000 G01 F1500 X100

N20 G196 S6000  (max. rpm. 6000 1/min)

N30 G96 S63     (Selection of const. cutting speed 63m/min,)
                (Workpiece radius 100mm according to X-coordinates)
N40 X80

N50 S4 X50      (New cutting speed 4m/min; workpiece radius 80mm, at) (block end 50mm)

N60 G97         (Deselection const.cutting speed)

N70 S8000       (max. rpm. 6000 1/min here not effective!)

N80 G92 X-10    (Reference point offset in X by -10mm)
```

```
N90 G96 X60      (Cutting speed from N50 not valid: const.
                 (cutting speed not active, rpm-value 8000 1/min)

N100 S25 X70     (Cutting speed 25m/min, Workpiece radius 50mm (=60mm+BPV)
                 (Const. cuttting speed, active)
N110 G00 X450    (Rapid mode: rpm-value remains constant)
N115 X70

N120 G01 X40     (Suppression of G96 cancelled)

N110 M30
```

15.2.4.5 エンドレスで回転するスピンドルによるねじ切り(G33)

ZX平面(縦軸Z、横軸X)の構文例:

G33 Z<*expr*> K<*expr*> [<*spindle_name*>.OFFSET[=]<*expr*>] (モーダル)

G33 エンドレスで回転するスピンドルによるねじ切り。G33ファンクションはモーダルです。停止ブロックタイプ(G00、G01、G02、G03、スプライン、多項式)を含む次の移動ブロックによってねじ切りが選択解除されます。

Z<*expr*> 目標位置(「ねじ長」)

K<expr> ねじリードは有効なねじ切りの下で、文字I、J、Kを使用した数学記号なしで、寸法(mm/回転)でプログラムされます。これらはX軸/Y軸/Z軸に割り当てられます。

ねじリードはプログラム終了までモーダルであり、G33選択時にゼロであってはなりません。送りはFワードを使用してプログラムされるのではなく、スピンドル速度とねじリードから得られます。

Z軸が縦旋回軸である場合、傾斜角が45°未満である縦ねじまたはテーパねじのリードはアドレス文字Kによって指定されます。傾斜角が45°以上である横ねじまたはテーパねじの場合、この指定はIによって(X軸が面旋回軸として使用される場合)、またはJによって(Y軸が面旋回軸として使用される場合)それぞれ行われます。

図14-4で、Z-X平面での指定文字を使用したねじリードの指定の例が示されています。

<*spindle_name*>.OFFSET=<*expr*> スピンドルのモジュロレンジにおけるねじオフセット角度。オプションは、多条ねじでのみ必要です。オフセット角度はプログラム終了までモーダルです。P-CHAN-00053に準拠するスピンドル名。「=」文字はオプションです。

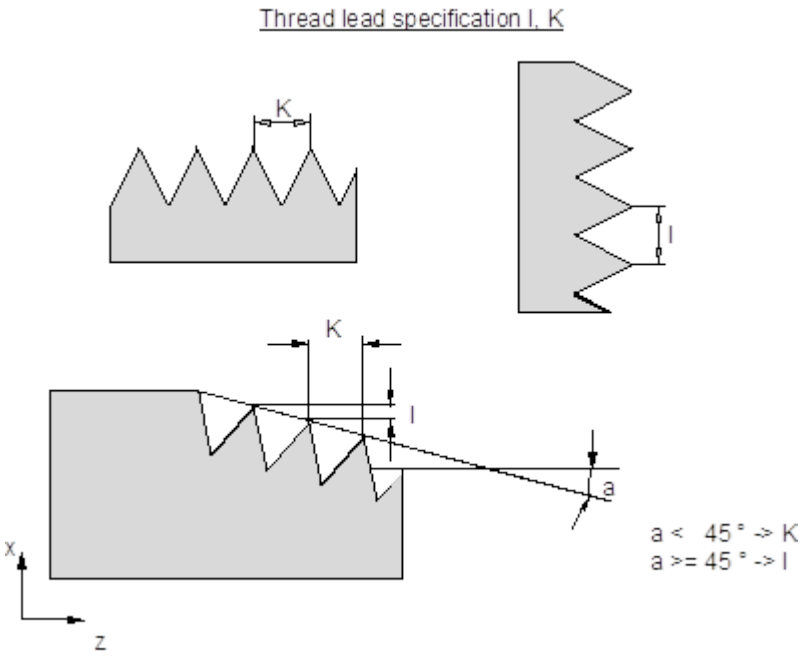


図 105: 図14-4: ねじリードの指定

プログラミング例

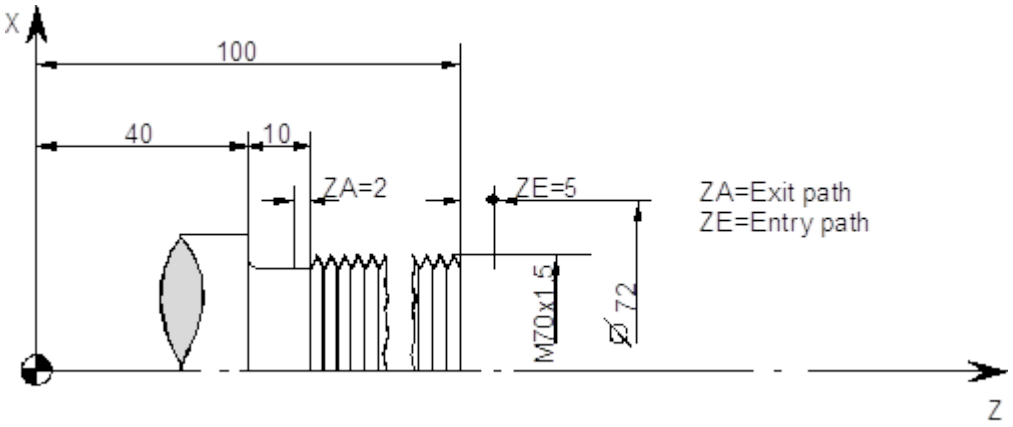


図 106: 図14-5: 形状例の説明

複数のカットを含む縦ねじ(M70x1.5)の切削:

```
%L Longit_Thread
N100 G33 Z48 K1.5                    Cut thread turn
N110 G00 X72                         Retraction and move
N120     Z105                        to start position
N130 M29                             Sub program end

%G33  (thread depth 0.92 mm)
N10 G51                              Selection of diameter programming
N15 T1 D1 M03 S400                   Tool selection, start spindle
N20 G00 X72 Z105                     Approach to starting point

N25 G01 X69.54 F1000                 Positioning on 1. cutting depth
N30 LL Longit_Thread                 1. cut

N35 G01 X69.08                       Positioning on 2. cutting depth
N30 LL Longit_Thread                 2. cut

N35 G01 X68.62                       Positioning on 3. cutting depth
N30 LL Longit_Thread                 3. cut

N35 G01 X68.16                       Positioning on end depth
```

```
N30 LL Longit_Thread                  4. cut

N35 G01 X68.16                        Positioning on end depth again
N30 LL Longit_Thread                  Empty cut

N60 M05 X150 Z200                     Moving to end position
N65 M30                               Program end
```

2条縦ねじ(M70x1.5)の切削:

```
%G33_2  (2-start thread, thread depth 0.92 mm)
N10 G51                               Selection of diameter programming
N15 T1 D1 M03 S400                    Tool selection, start spindle
N20 G00 X72 Z105                      Approach to starting point
N25 G01 X68.16 F1000                  Positioning on thread depth
N30 G33 Z48 K1.5                      Cut 1. thread turn
N35 G00 X72                           Retraction and move
N40     Z105                          to next
N45 G01 X68.16                        start position
N50 G33 Z48 K1.5 S.OFFSET=180         Cut 2. thread turn at 180°
N55 G00 X72                           Retraction and move
N60 M05 X150 Z200                     to end position
N65 M30                               Program end
```

15.2.5 タッピング(G63)

XY平面の構文例:

G63 **Z**<*expr*> **F**<*expr*> <*spindle_name*><*expr*> (モーダル)

G63 タッピング

Z<*expr*> 目標位置

F<*expr*> 送り速度

<*spindle_name*><*expr*> P-CHAN-00053に準拠するスピンドル名と数式で構成されるスピンドル速度

この種類のタッピング(G63)では、スピンドルがCNCと移動経路の組み合わせによって操作されます。この場合、スピンドルと軸の送り込み動作の正確な動的適合が行われます。そのため補正付きチャックは不要です。プログラムされた送り速度は、プログラムされたスピンドル速度およびねじリードに適合する必要があり、以下のように計算されます。

送り速度F [mm/min] = スピンドル速度S [rpm] * ねじリード[mm/rev]

G63は、異なるブロックタイプ(例えば、直線動作G01)の選択によって選択解除されます。ノンモーダルブロックタイプ(例えば、G04によるドウェル時間)はG63を無効にしません。

経路送り速度(Fワード)とスピンドル速度(Sワード)をG63と同じNCブロックで指定する必要はありません。プログラムされた最後の値は、送り速度の計算時に常に使用する必要があります。

経路送り速度またはスピンドル速度がG63が選択されていてゼロに等しい場合、エラーメッセージが出力されます。

スピンドルに対するG63の実行中にスピンドルMファンクション(M03、M04、M05、M19)をプログラムすることはできません。

注記

G63を選択する前にスピンドルが停止していることを確認する必要があります。これは、事前にM05 (スピンドル停止)またはM19 (スピンドル位置決め)をプログラムすることによって達成できます。

左ねじの切削またはねじ穴からの後退は**負のS値**でプログラムします。

プログラミング例1

リードが1.25 mmで奥行きが50 mmのネジがドリル加工されます。プログラムされたスピンドル速度Sが200 rpmの場合、計算される送りFは以下のようになります。

F = 200*1.25 = 250 mm/min

```
:
G63 Z-50 F250 S200     (Tapping)
    Z50 S-200          (Retraction from threaded bore)
G01 F100 X100 ...      (Deselection of tapping, linear interpolation)
:
```

プログラミング例2

```
%Tapping

N05 X0 Y0 Z0

N10 G91

N20 M03

S100 X100 M19 S.POS180 Z100    (Positioning tool and spindle)

N30 G63 Z-100 F300 S200        (Tapping)
N40     Z100       S-200       (Retraction from threaded bore 1)

N50 G01 X200 F3000 M5          (Positioning tool and spindle)

N60 G63 Z-70 F300 S200         (Tapping)
N70     Z70        S-200       (Retraction from threaded bore 2)

N80 M05 G01 X300 F1000         (Deselection of tapping, linear interpol.)

N90 M30
```

15.2.6 C軸加工

この機能は既存の旋回ファンクションを補完し、旋回台を備える旋盤およびフライス盤における円筒ワークの面加工と外側面加工を実現します。このプロセスでは、ワークが回転軸またはスピンドル(C軸)によって動かされ、駆動される工具(カッターなど)が2つの並進軸X (またはY)およびZによって動かされます。パラメータP-CHAN-00008でのC軸加工に固有の設定では、P-AXIS-00015が必要です。

面加工と外面加工は直交座標で表すことができます。

すべての補間タイプ(直線補間、円弧補間、スプライン軸補間など)が端面と外側面でサポートされます。この機能によって、旋回中心を横切る経路輪郭も実現します。旋盤の場合、C軸の自動調整によって加工されます。

2.5-D工具径補正は、よく知られているGコマンドで使用することができます。

旋回中心の近くの輪郭では、特に拡張された動的監視ファンクションを使用して、動的軸特性を超えることを防止することができます。

15.2.6.1 加工モード

すべての加工モードの主軸は、XまたはY (機械タイプに応じて)、Z、およびCです。

15.2.6.1.1 モード1: 経路成分への主軸の包含

この「基本モード」は原則的に旋盤でのC軸加工に必要です。その理由は、この場合、位置制御対象主軸を回転軸(例えば「C」)に変換する必要があるためです。

回転ワーク固定具(例えば、旋回台)を有するフライス盤とマシニングセンターでは、C軸加工も可能です。この場合、モード1 (#CAX)の選択は不要です。

3つの物理軸X(またはY)、Z、およびCを直接プログラムすることができます。直線軸(直交座標)とC軸(角度単位)。

半径/直径プログラミングはG52/G51に依存します。

2つの直線軸が主面を定義します。

ZX (G18)またはYZ (G19)です。

#CAX [[[<*ain_spindle_name*>,] <*C-axis_name*>]]

<main_spindle_name> メインスピンドル名のみP-CHAN-00053に準拠してプログラムすることができます。C軸以外のスピンドルを使用する場合、これをメインスピンドルで最初に宣言する必要があります(**プログラミング例**または0章を参照)。これを行わないと、エラーメッセージが出力されます。

<C-axis_name> NCプログラムで自由に定義できるC軸名。C軸名をプログラムしないと、P-CHAN-00010からの初期設定名が使用されます。

主面(円弧補間、工具径補正など)は、C軸の有効化の前と全く同じです。

変換を取り消す前にこのスピンドルのコマンド(M3、M4、M5など)をプログラムすると、エラーが生成されます。

C軸が以下によってスピンドルインターポレータに再び変換されます。

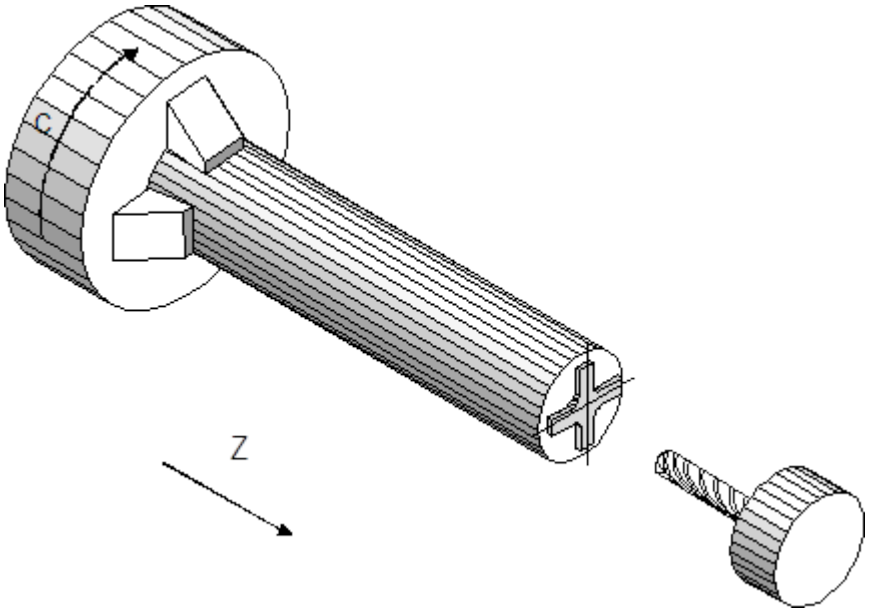
#CAX OFF

プログラミング例

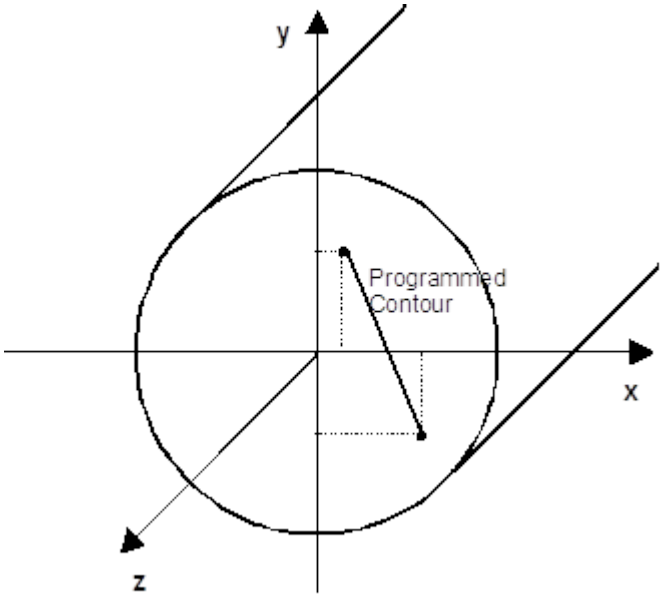
```
:
#CAX                              (Assumption: default C axis is "C")
G01 G90 X50 Z10 C90 F200
#CAX OFF                          (Deselection C-axis mode)
...
#CAX[S, C] or #CAX[C]             (Assumption: main spindle is "S")
G01 G90 X50 Z10 C90 F200
#CAX OFF
...
#MAIN SPINDLE [S2]                 ("S2" becomes the new main spindle "S")
#CAX[S, C]                        (Selection C-axis mode)
G01 G90 X50 Z10 C90 F200
...
#CAX OFF                          (Deselection C-axis mode)
...
#CAX[S3, C]                       (Error, "S3" is not a main spindle)
```

15.2.6.1.2 モード2: 面加工

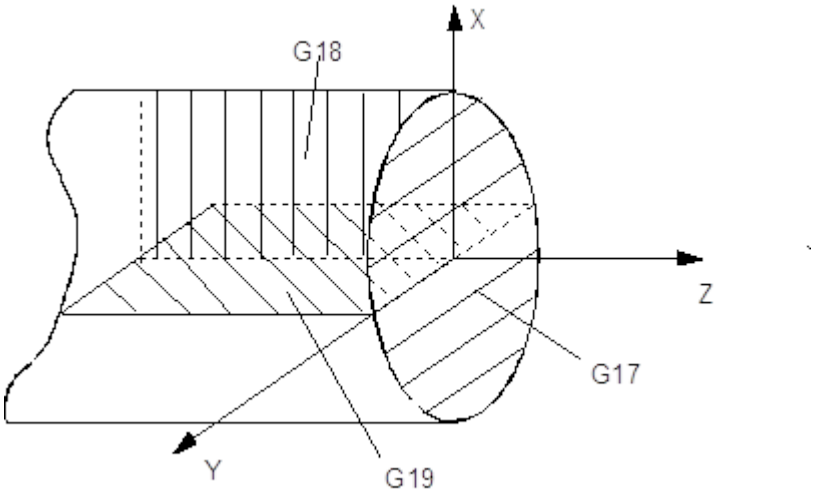
このモードは旋盤とマシニングセンターで便利です。面上の必要な輪郭は、仮想直交座標系を使用してmm(またはinch)単位でプログラムします。



面上の輪郭をプログラムする場合、3つの論理軸X、Y (またはC)、およびZを使用できます。輪郭は直交座標でプログラムされます。



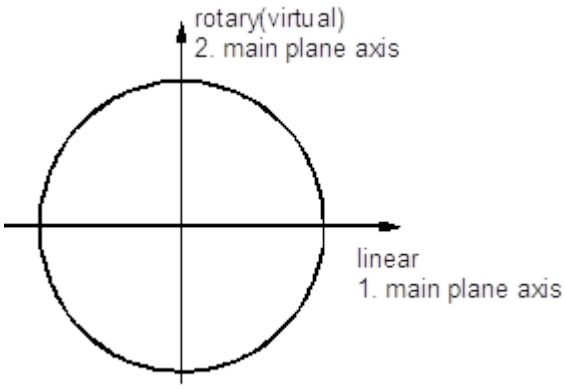
次の図は、面加工で使用できる主面を示しています。G17平面のみが技術的に重要です。



#FACE [<*1.main_axis_name*>, <*2.main_axis_name*> **]**

<1.main_axis_name> 現在の主面に従う第1の主軸の軸名。

<2.main_axis_name> 現在の主面に従う第2の主軸の軸名(仮想直交軸)。



選択時、主面(円弧補間、工具径補正など)が常に[第1の]主面軸と[第2の]主面軸によって定義されます。有効な面加工中にG18、G19によって主面を変更することは許されません。

補助軸をプログラムすることができます。補助軸は変換の影響を受けません。

このモードは以下によって取り消されます。

#FACE OFF

上記のコマンドによって、最後の有効なモード(例えば、モード1)に戻ります。つまり、最後の有効な主面が自動的に選択され、最後の有効な軸オフセットが復元されます。

旋盤のプログラミング例

第2の主軸の軸名「C」の例。

```
:
#CAX[S, C]                              (Assumption: main spindle is "S")
#FACE[X, C]                             (Selection of facing)
:
G01 X40 C-30 Z50 F1000                  (Pre positioning)
G01 Z30                                 (Infeed)
G01 X10 C40                             (Move contour)
G01 Z50                                 (Retraction)
:
#FACE OFF
#CAX OFF
```

第2の主軸の軸名「Y」の例。

ヒント:「Y」という同じ名前の軸がNCチャンネルに存在することはできません。

```
:
#CAX[S, Y]                              (Assumption: main spindle is "S")
#FACE[X, Y]                             (Selection of facing)
:
G01 X40 Y-30 Z50 F1000                  (Pre positioning)
G01 Z30                                 (Infeed)
G01 X10 Y40                             (Move contour)
G01 Z50                                 (Retraction)
:
#FACE OFF
#CAX OFF
```

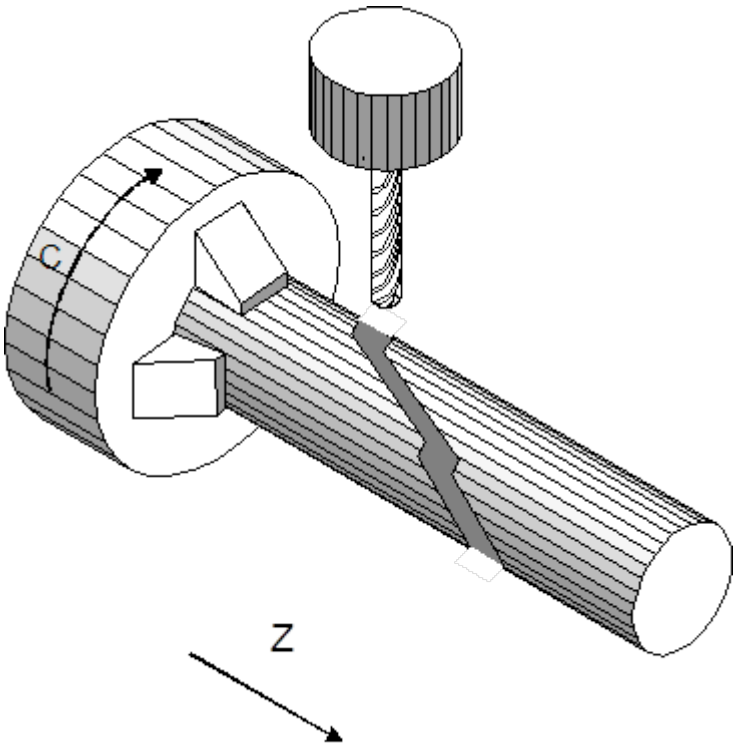
マシニングセンターのプログラミング例

例: チャンネルの回転軸(工具軸)が「C2」です。コマンド#CAXのプログラミングは不要です。

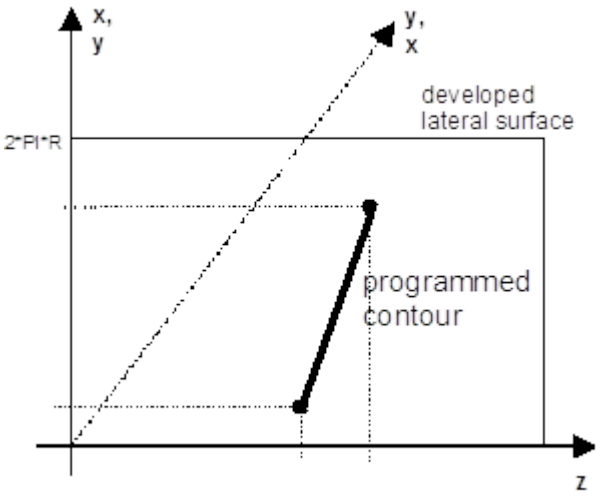
```
:
#FACE[X, C2]                            (Selection of facing)
:
G01 X40 C2=-30 Z50 F1000                (Pre positioning)
G01 Z30                                 (Infeed)
G01 X10 C2=40                           (Move contour)
G01 Z50                                 (Retraction)
:
#FACE OFF
```

15.2.6.1.3 モード3: 外側面

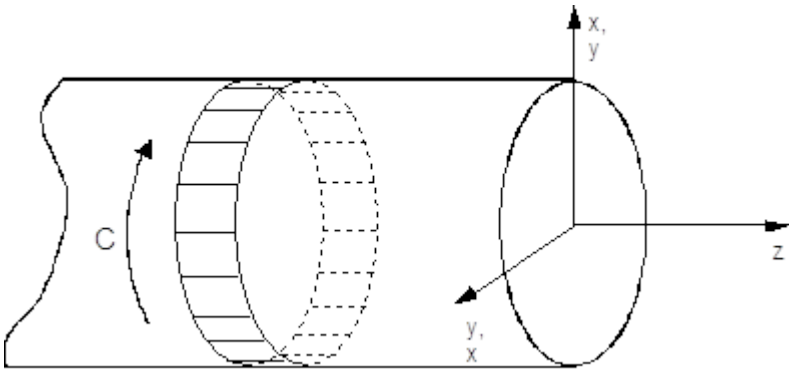
このモードは旋盤とマシニングセンターで便利です。円柱面上の必要な輪郭は、仮想座標系を使用してmm(またはinch)単位でプログラムします。



外側面上の輪郭をプログラムする場合、3つの論理軸X/Y/Zを使用できます。輪郭は、直交座標系で外側面に対してプログラムされます。このモードでは、ワークの基準半径Rを追加でプログラムする必要があります。



外側面加工では、主面がZ-Cによって形成されます。

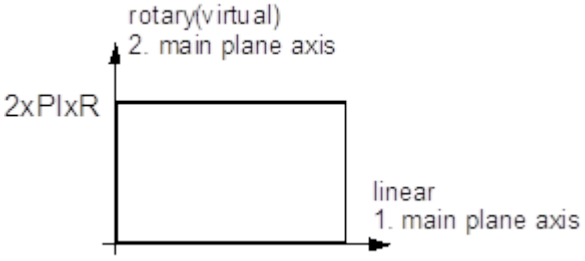


#CYL [<*1.main_axis_name*>, <*2.main_axis_name*>, <*3.main_axis_name*><*expr*> **]**

<1.main_axis_name> 現在の主面に従う第1の主軸の軸名。

<2.main_axis_name> 現在の主面に従う第2の主軸の軸名(仮想直線軸、扁平突出)。

<3.main_axis_name><expr> 基準半径が指定されている現在の主面に従う第3の主軸の軸名。



選択時、主面(円弧補間、工具径補正など)が常に[第1の]主面軸と[第2の]主面軸によって定義されます。有効な外側面加工中にG18、G19によって主面を変更することは許可されません。

補助軸をプログラムすることができます。補助軸は変換の影響を受けません。

このモードは以下によって取り消されます。

#CYL OFF

上記のコマンドによって、最後の有効なモード(例えば、モード1)に戻ります。つまり、最後の有効な主面が自動的に選択され、最後の有効な軸オフセットが復元されます。

旋盤のプログラミング例

第2の主軸の軸名「C」の例。

```
:
#CAX [S, C]            (Selection of lateral surface machining)
G01 X60 C45            (Setting- and positioning; X:60mm  C:45°)
#CYL [Z, C, X60]
G00 G90 Z0 C0          (Z: 0mm  C:0mm!)
G01 C100 F500
G02 Z100 R50
G01 C0
Z0
...
...
...
#CYL OFF
#CAX OFF
```

15.2.6.2 C軸モード間の切り替え

すべてのC軸モードの選択解除が、選択解除コマンド(例えば、コマンド#CYL OFF)によって定期的に行われます。現在の有効なモードを選択解除せずに異なるモードに直接遷移することは、モード2と3の間で可能です。そのため、例えばC軸の遷移のための代表的なNCシーケンスは、以下のようにプログラムします。

プログラミング例

```
N10 #CAX [..]    (Inclusion of the LR-spindle into the path compound)
..............
N120 #FACE [..]  (Selection of facing)
```

```
.............
N230 #CYL [..]    (Direct transition to lateral surface machining)
.........
N300 #CYL OFF     (Deselection of lateral surface machining and)
                  (transition to conventional machining with physical )       (C-axis)
N400 #CAX OFF     (Return of the C-axis to the position controlled)
                  (spindle)
```

15.2.6.3 面加工/外側面加工中の工具オフセット

コマンド#FACEと#CYLによって、キネマティックが暗黙的に選択されます。そのため、NCプログラムで#KIN IDによるキネマティックIDも[..] #TRAFO ONによる変換も有効にする必要はありません。

面加工では、2つの機械種類(旋盤とフライス盤)がサポートされています。対応する工具オフセットをチャンネルパラメータまたは工具パラメータで、以下の図に従ってキネマティックID13、14、および15の割り当てられたオフセットデータに入力する必要があります。

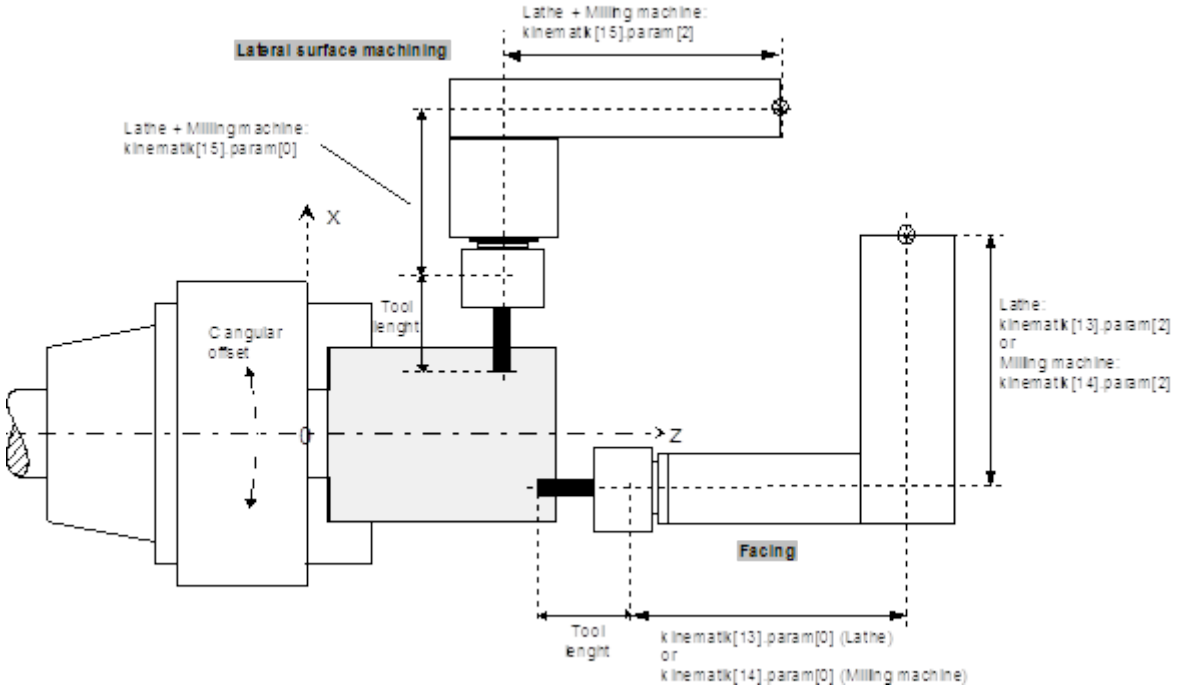


図14-6: CAX加工のためのキネマティックオフセットの説明

チャンネルパラメータにおける設定内容の例:

旋盤での面加工(KIN-ID 13):

```
kinematik[13].param[0]                1080000   Z offset [0.1µm]
kinematik[13].param[1]                      0   C angular offset [10-4°]
kinematik[13].param[2]                 900000   X offset [0.1µm]
```

フライス盤での面加工(KIN-ID 14):

```
kinematik[14].param[0]                1080000   Z offset [0.1µm]
kinematik[14].param[1]                      0   C angular offset [10-4°]
kinematik[14].param[2]                 900000   X offset [0.1µm]
```

旋盤またはフライス盤での外側面加工(KIN-ID 15):

```
kinematik[15].param[0]                  700000   X offset [0.1µm]
kinematik[15].param[1]                       0   C angular offset [10^-4°]
kinematik[15].param[2]                 1200000   Z offset [0.1µm]
```

15.2.7 ギヤ切替(G112)

G112 *<spindle_name><expr>*	(ノンモーダル)

G112 ギヤ切替

<spindle_name><expr> デコーダパラメータリストに従うスピンドル名と数式で構成されるギヤデータ記録

機械的ギヤ切替操作も暗黙的に実行される、M40-45によるギヤ切替とは違って、Sワードを伴うG112のプログラミングは、1段のスピンドルギヤデータ(動的値)を更新させるだけです。

正しいギヤ段数の機械的変更をユーザが明示的にプログラムする必要があります(例えば、ユーザ定義されたMファンクションによって、または上述のようにM40-45によって)。

注記

ギヤデータ記録を変更する前に、スピンドルを停止状態にしておく必要があります。これは、スピンドル停止(M5)またはスピンドル位置決め(M19…)をプログラムすることによって、以前のNCブロックで達成できます。

プログラミング例

```
%Test_G112

N010 G112 S2   (Load dynamic data of gear range 2 for spindle „S)
N020 M3 S9000
N030 M5        (Stop spindle)
N040 G112 S1   (Load dynamic data of gear range 1 for spindle „S)
N050 M4 S8000
N060 M30
```

15.2.8 原点復帰(G74)

G74 *<spindle_name><dummy_expr>* (ノンモーダル)

G74 原点復帰

<spindle_name><dummy_expr> P-CHAN-00053に従うスピンドル名と数式の指定。

閉ループ位置制御対象スピンドルについて参照検索を行うことができます。直線軸の参照とは違って、スピンドル名を使用してプログラムされた値は、参照の順序に関して意味がなく、複雑な構文を表すためだけに必要です。

スピンドルMファンクションのプログラミングは、G74と同じNCブロックでは許可されません。

スピンドルを参照するときは、以下の事例を区別する必要があります。

プログラミング例

```
1. Referencing of the spindle starts simultaneously with Y-axis referencing:
   Nxx   G74  X2  Y1  S1

2. As for 1.! The system continues with the next NC-block without waiting until the spindle has been
referenced so that the X-axis is references quasi-simultaneously:
   Nxx   G74  S1
   Nyy   G74  X1  Y2

3. Axes X and Y are referenced first. Spindle referencing then commences:
   Nxx   G74  X1  Y2
   Nyy   G74  S1
```

15.2.9 オーバーライド(G167)

G167 *<spindle_name><expr>]* (ノンモーダル)

G167 スピンドルオーバーライドを100 %に設定

<spindle_name><expr> P-CHAN-00053に従うスピンドル名と数式の指定で構成されるスピンドル速度。

ファンクションG167はスピンドルオーバーライドの外部影響を無効にし、実際にプログラムされた速度を実装します。

プログラミング例

```
:
N10 M3 S1000     (With override 50% speed is 500 U/min.)
N20 G167 S1000   (Override influencing off, speed is 1000 U/min.)
N30 S3000        (Override influencing active again, speed is 1500 U/min.)
Nxx
```

15.3 スピンドル特有の構文でのプログラミング

スピンドル特有の構文には、**複数の**スピンドルを**同じ** NCブロックで互いに独立してプログラムできるという利点があります。

これは、スピンドル名に付随している、括弧で囲まれた数式内で実行します。この括弧で囲まれた数式では、特定のコマンドのみ許可されます。これらのコマンドは、常にスピンドル特有に処理・実行されます。メインスピンドルは、スピンドル名P-CHAN-00053によってプログラムする必要があります。

<*spindle_name*> [[**M3** | **M4** | **M5** | **M19**] [**REV***<expr>*] [**POS***<expr>*] { **M***<expr>* } { **H***<expr>* }
[**G74**] [**G167**] [**CALLAX** | **PUTAX**] [**GET_DYNAMIC_DATA**] [**G130**]
[[**G135** | **G137**] [**G136***<expr>*]] [**FEED_LINK…**] [**OVERRIDE…**] { \ } **]**
{ <*spindle_name*> **[** .... **]** }

<spindle_name>	[1]-3およびP-CHAN-00053に従うスピンドル名称
M3、M4、M5、M19	スピンドルMファンクション
REV*<expr>*	スピンドル速度
POS*<expr>*	スピンドル位置
M*<expr>*	ユーザ定義のMファンクション
H*<expr>*	ユーザ定義のHファンクション
G74	原点復帰
G167	スピンドルオーバーライド100%
CALLAX	軸の要求
PUTAX	軸の解放
GET_DYNAMIK_DATA	動的工具データの継承
G135, G136*<expr>*, G137	フィードフォワード制御
FEED_LINK…	スピンドル送りリンク
OVERRIDE…	スピンドルオーバーライド
\	複数行にわたるコマンドの明確なプログラミングのための区切り記号(「バックスラッシュ」)。

G130 加速度の重み付け

プログラミング例

```
:
N10 S[M3 REV500 M19 POS45 M18 M15 H20 ...]      S2[M4 REV5000]
Nxx
:
```

15.3.1 スピンドルMファンクション

15.3.1.1 スピンドル移動(M3/M4/M5)

M03	スピンドル回転時計回り(cw)	(モーダル)
M04	スピンドル回転反時計回り(ccw)	(モーダル)
M05	スピンドル停止	(モーダル)

スピンドルMファンクションM03およびM04はスピンドルの回転方向を定義し、スピンドル速度(REVワード)と組み合わせて使用する必要があります。M05はスピンドル回転を停止します。これらのスピンドルMファンクションは、制御システムの起動後とプログラムが初めて開始されるときの、初期設定のスピンドルモードです。これらのMファンクションは停止作用があり、それぞれ、括弧で囲まれた数式内で単独でのみプログラムすることができます。

M03またはM04がプログラムされており、有効な速度(REV)が設定されている場合に、スピンドル回転が有効です。

プログラム終了時にM05が設定されていない場合、スピンドルは回転し続けます。

プログラミング例

1台のスピンドル「S」のプログラミング:

```
N10 S[REV1000]      (Speed 1000 U/min. is stored, no spindle rotation)
                    (because M05 is default)
N20 S[M03]          (Spindle rotation cw with 1000 U/min.)
N30 S[M04]          (Spindle rotation ccw with 1000 U/min.)
N40 S[REV500]       (Spindle rotation ccw with 500 U/min.)
N50 S[M05 REV300]   (Spindle rotation, speed 300 U/min. is stored)
N60 S[M04]          (Spindle rotation ccw with 300 U/min.)
N70 S[M05]          (Spindle stops)
N80 S[M03 REV1000]  (Spindle rotation cw with 1000 U/min.)
N90 M30             (Program end)
```

プログラミング例

2台のスピンドル「S」および「S2」のプログラミング:

```
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 with 1500 U/min.)
N30 S[M04]                          (S ccw 1000 U/min.)
N40                 S2[M05]         (S2 stop)
N50 S[M05 REV300]                   (S stop, speed 300 U/min. is stored)
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                             (Program end)
```

チャンネルパラメータリスト[1]からの抽出:

M3/M4/M5に対する同期モードをスピンドル特有に定義する必要があります。同期モード「0」(NO_SYNCH)の場合、Mファンクションは実行されません。

```
:
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 スピンドルの位置決め(M19、POS)

M19	スピンドルの位置決め	(ノンモーダル)
POS	スピンドル位置	(モーダル)

スピンドルMファンクションM19はスピンドルの位置決めを行います。M19はPOSワードと組み合わせて使用する必要があります。同じNCブロックにおけるM03/M04またはスピンドル速度(REV)はオプションです。ただし、有効なスピンドル速度(> 0)を設定する必要があります。M19をM5(スピンドル停止)とともに使用することはできません。

スピンドル位置POSはモーダルであり、M19が再びプログラムされる場合に再指定する必要はありません。以前にスピンドル位置がプログラムされていない場合、スピンドルは初期設定位置「ゼロ」に移動します。

スピンドルが回転していない場合、最短の移動経路で位置決めが実行されます。

M19によるスピンドルの位置決めは、位置制御対象スピンドル(閉ループ)についてのみ可能です。

プログラミング例

1台のスピンドル「S」のプログラミング:

```
N10 S[REV100 POS45 M19 M3]  (Spindle rot. cw with 100 U/min. to pos. 45)
N20 S[M03 REV1000]          (Spindle rot. cw with 1000 U/min.)
N30 S[M19 M4 REV150]        (Spindle rot. ccw with 150 U/min. to pos. 45)
                            (->POS is modal!)
N40 S[M05]                  (Spindle stop)
N50 S[M19]                  (Move at shortest distance with 150 U/min. to)
                            (position 45)
N60 S[REV200 M19 POS90]     (Move at shortest distance with 200 U/min. to)
                            (position 90)
N70 S[M03 REV1000]          (Spindle rotation cw with 1000 U/min.)
N80 S[M05]                  (Spindle stop)
N90 M30                     (Program end)
```

プログラミング例

2台のスピンドル「S」および「S2」のプログラミング:

```
(For N10: Move both spindles cw with 1000 U/min. to position 45)

N10 S[M03 REV200 M19 POS45] S2[M04 REV200 POS45 M19]

N20 S[REV1000] S2[REV1500]          (S cw with 1000 U/min.,
                                    (S2 ccw with 1500 U/min.)
N30 S[M04]                          (S ccw 1000 U/min.)
N40 S2[M19 POS0]                    (Move S2 ccw to position 0)
N50 S[M05 REV300]                   (S stop, 300 U/min. is stored)

(For N60: Move S cw with 300 U/min. to position 90),
(For N60: Move S2 cw with 200 U/min. to position 45)

N60 S[M03 M19 POS90] S2[M03 REV200 POS45 M19]
N80 M30                             (Program end)
```

チャンネルパラメータリスト[1]からの抽出:

M19の場合、同期モードをスピンドル特有に定義する必要があります。同期モード「0」(NO_SYNCH)の場合、Mファンクションは実行されません。

```
:
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 スピンドル速度(REV)

値をREVワードに直接、またはパラメータを用いて割り当てることができます。また、10進数を使用することもできます(REAL型)。

REVワードにおけるスピンドルMファンクションと組み合わせた、以下の種類の使用を区別する必要があります。

- M03/M04/M19と組み合わせたREVワード:

REVワードをM03/M04/M19と組み合わせてプログラムすると、REVワードに続く値がスピンドル速度として解釈され、スピンドルに出力されます。

- M05と組み合わせたREVワード:

M05とともに、REVワードに続く値がスピンドル速度の作業データに伝送されますが、スピンドルには出力されません。

REVワード単独では、NCプログラムでの動作を生成しません。スピンドルモードM03/M04/M19は、このために既知であることが必要です。そのため、M03/M04/M19のプログラミングは、REVワードが設定されている場合にのみ動作可能です。

REV値が負であると、エラーメッセージが出力されます。

プログラミング例

スピンドルS1を含むプログラミング:

```
N10 S1[REV300]         (Speed 300 U/min. is stored)
N20 S1[M04]            (Spindle rotation ccw with 300 U/min.)
N30 S1[M03 REV]1000    (Spindle rotation cw with 1000 U/min.)
N40 S1[REV500]         (M03 active, spindle rotation cw with 500 U/min.)
N50 S1[M05 REV100]     (Spindle stop, speed 100 U/min. is stored)
N60 S1[M04]            (Spindle rotation ccw with 100 U/min.)
N70 S1[M05]            (Spindel stop)
N80 M30                (Program end)
```

チャンネルパラメータリスト[1]からの抽出:

Sワードでは、同期モードをスピンドル特有に定義する必要があります。Sワードを無視することはできないため、同期モードが「0」(NO_SYNCH)の場合には、エラーメッセージが出力されます。

```
:
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 ユーザ定義のM/Hファンクション

括弧で囲まれた数式内でプログラムされたユーザ定義のすべてのM/Hファンクション(技術項目)は、**常に**スピンドル特有に処理・出力されます。

プログラミング例

2つのユーザ定義のMファンクションによる1台のスピンドル「S2」のプログラミング:

```
N10 S2[M3 REV300 M10 M11]  (S2 cw with 300 U/min.and executes M10/ M11)
N20 M30                    (Program end)
```

2台のスピンドル「S」および「S2」のプログラミング:

```
N10 S[M10 M11] S2[REV1000 M3 M11]  (S executes M10 and M11, S2 rotates cw)
                                   (with 1000 U/min.and executes M11)
N20 M30                            (Program end)
```

チャンネルパラメータリスト[1]からの抽出:

ユーザ定義のM/Hファンクションは、解読処理パラメータリスト[1]で定義されています。同期モード「0」(NO_SYNCH)の場合、M/Hファンクションは実行されません。

```
:
# Definition of the M-functions and types of synchronisations
# ======================================================
m_synch[1]                                          0x00000001     MOS
m_synch[2]                                          0x00000002     MVS_SVS
:
m_synch[10]                                         0x00000002     MVS_SVS
m_synch[11                                          0x00000008     MNS_SNS
m_synch[12]                                         0x00000004     MVS_SNS

m_synch[48]                                         0x00000008     MNS_SNS
m_synch[49]                                         0x00000002     MVS_SVS
:
:
# Definition of the H-functions and types of synchronisations
# ======================================================
h_synch[1]                                          0x00000001     MOS
h_synch[2]                                          0x00000001     MOS
:
```

15.3.4 原点復帰(G74)

参照検索をスピンドル特有に実行することができます。スピンドルMファンクションのプログラミングをG74とともに行うことはできません。

プログラミング例

2台のスピンドル「S」および「S2」のプログラミング:

```
N10 S[G74] S2[G74]             (Homing of S and S2)
N20 M30                        (Program end)
```

15.3.5 オーバーライド(G167)

ファンクションG167はスピンドルオーバーライドの外部影響をスピンドル特有に無効にし、実際にプログラムされた回転速度を実装します。1台のスピンドル [▶ 487]のプログラムされたオーバーライド値の設定は有効なままです。

プログラミング例

1台のスピンドル「S2」のプログラミング:

```
N10 S2[M3 REV1000]      (With override 50% speed is 500 U/min.)
N20 S2[G167 REV1000]    (Override influencing off, speed 1000 U/min.)
N30 S2[REV3000]         (Override influencing active again,)
                        (speed 1500 U/min.)
N40 M30                 (Program end)
```

15.3.6 スピンドル軸の解放/要求(PUTAX/CALLAX)

このファンクションを使用できるかどうかは、設定やバージョンの具体的な範囲によって異なります。

これらのコマンドにより、スピンドル軸をスピンドル特有に解放または取得することができます。これらのコマンドを他のスピンドル特有のコマンドと同時に使用することはできません。

プログラミング例1

```
%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
```

プログラミング例2

複数のスピンドルのプログラミング:

```
:
N10 S[CALLAX] S2[CALLAX] S3[PUTAX]  (S, S2 request axes,)
                                    (S3 releases their axis)
N20 S[M3 REV200] S2[G74)  (S rotates cw with 200 U/min.), S2 is homing)
N30 S3[ M4 REV3000]       (Error, S3 has no axis)
N40 S[PUTAX REV400)       (Error, PUTAX is not programmed exclusively)
:
```

15.3.7 工具動的データの受け入れ(GET_DYNAMIC_DATA/ DEFAULT_DYNAMIC_DATA)

このファンクションを使用できるかどうかは、設定やバージョンの具体的な範囲によって異なります。

工具動的データ(最小/最大速度、最大加速度)は、新しい工具のプログラミング後に停止状態から補間にスピンドルが遷移すると自動的に有効になります(Dワード、#TOOL DATA)。変更された工具動的データがスピンドル特有のコマンド「GET_DYNAMIC_DATA」によって、回転しているスピンドルに対して受け入れられ、許可されます。

停止したスピンドルについては、コマンド「DEFAULT_DYNAMIC_DATA」によって、工具に依存しない初期設定の動的データに戻ることができます。現在の有効なステップ(P-TOOL-00016/P-TOOL-00017)は変更されません。

これらのコマンドを他のスピンドル特有のコマンドと同時に(または組み合わせて)使用することはできません。

プログラミング例

```
N10 T1 D1                   ;Support of tool dynamic data inside the
                            ;tool data
N15 M6                      ;Tool change of tool 1
N20 S[M3 REV2002]           ;S rotates 2000 U/min. with tool dynamic data
                            ;of D1
N25 X100 Y100               ;Movement block with tool dynamic data of D1
N30 T99 D99                 ;Support of new tool dynamic data inside the
                            ;tool data during rotating spindle (N20)
N35 X200 Y150               ;Movement block with tool dynamic data of D1
N40 S[GET_DYNAMIC_DATA]     ;Taking over tool dynamic data D99
```

```
N45 X150 Y200                ;Movement block with tool dynamic data of D99
N50 X50 Y50                  ;Movement block with tool dynamic data of D99
N60 S[M05] Z100              ;Spindel stop
N70 S[DEFAULT_DYNAMIC_DATA] ;Change back on default dynamic data
N99 M30                      ;Program end
```

15.3.8 スピンドル特有のフィードフォワード制御(G135、G136、G137)

このファンクションを使用できるかどうかは、設定やバージョンの具体的な範囲によって異なります。

このコマンドによって、フィードフォワード制御のスピンドル特有のプログラミングができます。これらのコマンドを他のスピンドル特有のコマンドと同時に使用することはできません。

有効化をG135によってプログラムします。スピンドル特有の計算されたフィードフォワード制御変数のパーセンテージ重み付けが、G136によって行われます。これは100%に限定されます。G137はフィードフォワード制御を無効にします。フィードフォワード制御の選択と重み付けを同じコマンドシーケンスに入力することも可能です。

NCプログラム内で正方向送り制御のオン/オフを切り替えると、加重係数はG136で設定された値のまま維持されます。G136がプログラムされていない場合には100%となります。

プログラミング例

```
S[G135]               Activation of feedforward control for S
S[G136=80]            Definition of weighting in percent
S[G137]               Deactivation of feedforward control
S2[G135 G136=90]      Activation and weighting 90% for S2
S2[G136=0]            Change of weighting on 0%
S1[G135]              Activation and default weighting 100% for S1
```

15.3.9 スピンドルフィードリンク(FEED_LINK)

通常、スピンドル速度はプログラムによってのみ制御されます。他のパラメータに頼ってスピンドル速度を制御することはできません。一部の特別な技術(例えば、HSCフライス加工、木材加工における工ッジベンディング)と高品質ワーク表面の作成の場合も、外部変数によるスピンドル速度の制御が必要です。

以下のコマンドを使用して、スピンドル特有の速度を経路送りに対して動的に適合させることができます。特に、誤ったまたは不適切な切削パラメータによって損傷を受けることがある一部の材料(例えば、木材やプラスチック)にとって、これが必要です。

注記

スピンドルの動的データは考慮されません。有効なフィードリンク中に、命令されたリンク係数と実際のリンク係数の間の大きすぎる偏差を避けるために、要件に応じてスピンドルダイナミクスを十分高くするか、または経路動作を小さくする必要があります。

スピンドルフィードリンクのプログラミング構文:

```
<spindle_name> [ FEED_LINK [ON | OFF]  [ [ [FACT<expr>]  [CORR<expr>] ] | AUTO]  [MAIN] ]
```

<spindle_name> フィードリンクスピンドルの名前

FEED_LINK 「スピンドルフィードリンク」の識別。必ず**1番目に**プログラムされるキーワードであること。

ON スピンドルフィードリンクの選択

OFF スピンドル送りリンクの選択解除

FACT*<expr>* **同期**リンク: 経路上の送り(Fワード)と基本的なスピンドル速度の間の、あらかじめ定義されたリンク係数(単位[0.1 %])。リンク係数は、有効なリンク中にいつでも変更できます。

CORR*<expr>* リンク係数を変更するための補正係数(単位[0.1 %])。変更後のリンク係数は、FACTとCORRの組み合わせです(C_{RES} = FACT*CORR)。

AUTO リンク**オンザフライ**: このリンク係数はスピンドルの現在の速度と経路上の現在の送りから自動的に計算され、リンクの選択解除まで一定のままです。

MAIN スピンドルフィードリンクは、少なくとも1台のスピンドルが動作に関与している場合にのみ有効です。従軸のみが動く場合、スピンドル速度は影響を受けません。

- キーワードAUTOをプログラムできるのは、ONと組み合わせた場合に限られます。FACTとCORRは、既に有効なフィードリンク中に再びプログラムすることができます。
- キーワードMAINをプログラムできるのは、ONと組み合わせた場合に限られます。
- リンク選択時に、スピンドルは「速度 != 0」で工ンドレス回転状態であること。また、経路上の送りはリンク係数の 自動決定中に「!=0」であること。
- フィードリンクの選択解除時に有効なスピンドル速度が、命令されたスピンドル速度として使用され続けます。必要に応じて、フィードリンクの選択解除後にSワードによってスピンドル速度を再定義する必要があります。

注記

Sワードによるスピンドルジョブは、有効なフィードリンク中には無効です。

動的パラメータによって、工具を定義して有効にする必要があります。一般に、この工具P-TOOL-00013の最小速度によって、有効なフィードリンクと経路上の送り0でのフライス加工中のスピンドルが、工具の停止と故障を引き起こすことを回避できます。

プログラミング例

例1: (手動でプログラムされるリンク係数、有効なリンク係数が補正係数のプログラミングによって変更されます)。

```
%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
```

例2: (リンク係数が現在のスピンドル速度と軌道上の現在の送りから自動的に生成されます)。

```
%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
```

例3: (リンク係数が自動的に生成され、フィードリンクが主軸動作に対してのみ有効です)。

```
%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       ;Spindle speed remains constant
N70 G01 Z40 F15
N80 G01 Z60
N90 G01 C90       ;Spindle speed remains constant
N100 G01 C120     ;Spindle speed remains constant
N110 G01 X50 Y50 Z50
N120 S[FEED_LINK OFF]
N130 M03 S200
N140 G01 X10
N150 M05 S0
N160 M30
```

15.3.10 プログラム可能なスピンドルオーバーライド(OVERRIDE)

このコマンドは、NCプログラムでのスピンドル速度の制御を可能にします。割り当てられたスピンドルが動いている場合、スピンドルに対応してプログラムされたオーバーライドが有効です。

追加で外部オーバーライドも定義した場合、両方のオーバーライド値の乗算によって、有効なオーバーライドが得られます。

ファンクションG167 [▶ 482]は、外部オーバーライド値のみの影響を抑えます。

<spindle_name> **[OVERRIDE SPEED_FACT** *<expr>* **]**

<spindle_name>	スピンドルの名前
OVERRIDE	スピンドル特有のオーバーライドプログラミングの識別。必ず**1番目に**プログラムされるキーワードであること。
SPEED_FACT *<expr>*	スピンドル速度ブロックのオーバーライド係数[0.1% - 200%]

プログラミング例

```
%spdl_override

N10  G01 X100 Y100 Z100 F1000
N20  S[M3 REV1000]
N40  S[OVERRIDE SPEED_FACT=20]        ;Spindle override 20%
N50  X0                               ;G01 movement with S200
N60  Y0                               ;G01 movement with S200
N70  Z0                               ;G01 movement with S200
N60  S[OVERRIDE SPEED_FACT=100]       ;Spindle override 100%
N90  Y100                             ;G01 movement with S1000
N100 Z100                             ;G01 movement with S1000
N110 X200 Y200                        ;G01 movement with S1000
N120 X300 Y300 Z200                   ;G01 movement with S1000
N130 M5
M30
```

15.4 メインスピンドルの変更

#MAIN SPINDLE [[<*spindle_name*> | <*spindle_number*>]]

<*spindle_name*> [1]-3に従う初期設定スピンドル名。

<*spindle_number*> [1]-3に従うスピンドルの論理軸番号。

コマンド#MAIN SPINDLEを使用して、NCプログラムでのメインスピンドルの定義を変更することができます。新しいメインスピンドルを選択するには、初期設定の名前[1]-3**または**対応する論理軸番号を指定します。

スピンドル名をプログラムせずに、基本状態(開始後の)を復元できます。つまり、チャンネルパラメータリストP-CHAN-00051でプリセットされているスピンドルが再び元のメインスピンドルになります。

チャンネルパラメータリスト[1]からの抽出:

3台のスピンドルを備えるシングルチャンネルシステムの設定例。軸番号6のスピンドルがメインスピンドルです。

```
:
# Spindle data
# ============
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
:
```

開始後の設定:

S1は名前が「S」のメインスピンドルです。

「S2」と「S3」はその他のスピンドルです。

プログラミング例

```
%
N10 S100 M3 S2[REV200 M3] S3[REV300 M4]
N20 #MAIN SPINDLE [S2]          (S2 is new main spindle "S")
N30 S110 M3 S1[REV210 M3] S3[REV310 M4]
N40 #MAIN SPINDLE [S3]          (S3 is new main spindle "S")
N50 S120 M3 S1[REV220 M3] S2[REV320 M4]
N60 #MAIN SPINDLE               (Back to default state S1 -> "S")
N70 S150 M3 S2[REV250 M3] S3[REV350 M4]
N80 M5 S2[M5] S3[M5]            (All spindles STOP)
N99 M30
```

スピンドルがメインスピンドルであれば、そのスピンドルをメインスピンドル名P-CHAN-00053または初期設定の名前[1]-3でプログラムすることができます。#MAIN SPINDLE[ ]による別のメインスピンドルの選択後は、初期設定の名前でのみ再び指定することができます。

上記の例では、以下のスピンドル名が有効です。

許可される名前	Spindle 1	Spindle 2	Spindle 3
...開始後	**SまたはS1**	S2	S3
...#MAIN SPINDLE [S2]の後	S1	**SまたはS2**	S3
...#MAIN SPINDLE [S3]の後	S1	S2	**SまたはS3**
...#MAIN SPINDLEの後	**SまたはS1**	S2	S3

既に述べたように、メインスピンドルを従来のDIN規格に沿ってプログラムできます。この場合、表14-1に従うすべてのコマンドを使用できます。ただし、メインスピンドルはスピンドル特有の構文でプログラムすることもできます。しかしこの場合、限られたコマンドセットしか使用できません(表14-1も参照)。

プログラミング例

メインスピンドルの場合、以下のパートプログラムブロックが同等に許可されます。

```
:
N10 S1=1000 M3   or
N20 S1000 M3   or
N30 S1[REV1000 M3]   or
N40 S[REV1000 M3]
:
```

15.5 同期スピンドル操作

補間軸の同期操作(定義、有効化、無効化)に加え、LINKコマンドを使用してスピンドル軸のマスタ/スレーブ関係を定義することもできます。

#SET AX LINK [*<coupling_group>*, *<slave>* = *<master>* {, *<slave>* = *<master>*} **]**

または

#AX LINK [NBR] [*<coupling_group>*, *<slave>* = *<master>*{, *<slave>* = *<master>*} **]**

<coupling_group> 結合グループ数[1]

<slave> 結合ペア i のスレーブスピンドルの名称または論理軸番号[2]

<master> 結合ペア i のマスタスピンドルの名称または論理軸番号[2]

NBR 論理スイッチNBRを使用して、評価を軸名から論理軸番号に変更できます。変更後は、軸結合(および結合ペア)を論理軸番号で定義する必要があります。

この場合、「同期操作」の章に加えて以下のルールが適用されます。

- 結合ペアは、チャンネル上の知られている**スピンドル名**または**論理軸番号**によって定義します。つまり、チャンネル上で知られているスピンドルのみを結合することができます。
- チャンネルパラメータリスト[1]-2では、結合グループ0を初期設定グループとしてプリセットすることができます。これは、開始後に#ENABLE AX LINKまたは#AX LINK ONを使用して直接指定することができます。NCプログラムで再定義することは**できません**。

プログラミング例

チャンネルパラメータリスト[1]の情報: S (メインスピンドル名S1)、S1、S2、S3

->結合ペアをスピンドル名S、S2、およびS3を使用して形成することができます。

スピンドル結合のプログラミングと選択/選択解除:

```
N10 #SET AX LINK[1, S2=S, S3=S]   (Main spindle is master for S2 and S3)
N20 #ENABLE AX LINK[1]  (Activation of spindle couplings)
N30 S1000 M3            (Main spindle S+S2+S3 are rotating cw 1000 U/min)
N40 #DISABLE AX LINK    (Deactivation of spindle couplings)

or alternative

N10 #AX LINK[1, S2=S, S3=S]
N20 #AX LINK ON[1]
N30 S1000 M3
N40 #AX LINK OFF

or alternative

N10 #AX LINK NBR[1, 11=6, 17=6]       Coupling via log. axis numbers
N20 #AX LINK ON[1]
N30 S1000 M3
```

```
N40 #AX LINK OFF
N50 M30                 (Program end)
```

(1) 1 ... [結合グループの最大数–1]、[6]-2.11を参照

(2) 結合ペアの最大数、[6]-2.12を参照

- 異なる軸タイプを結合ペア**内で**#SET AX LINKまたは#AX LINKに定義することはできません。ただし、結合ペアが、別の軸タイプの結合ペアと結合グループを形成することができます。

例:

#SET AX LINK [1, B=S, S2=X] **FALSE**

#SET AX LINK [1, B=X, S2=S] **ALLOWED**

- 結合グループが有効である間は、このグループに存在するスピンドルを#MAIN SPINDLE [..]でメインスピンドルとして宣言することはできません。宣言すると、デコーダとインターポレータの間に矛盾が生じることがあるためです。
- さらに、プログラマは次のことを認識している必要があります。#MAIN SPINDLE [..]を使用すると、既に定義された結合グループが有効でなくなることがある(スピンドル名に一貫性がないため)ということです。
- M03/M04/M05の技術項目は、マスタスピンドルとスレーブスピンドルの両方についてNCカーネルによって同期されます。
- FEEDHOLDとOVERRIDEによるマスタスピンドルの外部動作制御はスレーブスピンドルに対して作用**しません**。結合が有効である間、スレーブスピンドルは自身のOVERRIDEインターフェイスとFEEDHOLDインターフェイスの評価を続けます。
- M19では、マスタスピンドルとスレーブスピンドルは同じ絶対位置に移動します。スピンドルの開始位置または動的データが異なる場合、この絶対位置に同時に達しないことがあります。
- プログラムが中止されると、結合が取り消されます。
- スピンドルがマスタまたはスレーブとして有効である場合、リンクを有効にしたチャンネルによってのみスピンドルを処理することができます。

チャンネルパラメータリスト[1]からの抽出:

```
:
# Predefintion of possible axis links for synchronous operation
# ================================================
#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 ブロック間の同期(Late Sync)

15.6.1 暗黙の同期

スピンドルの加速と減速の操作によって、プログラムの実行で大幅なむだ時間が生じることがあります。理由は、機械が停止状態の場合に頻繁に、最初に(タイプMVS_SVSのM03など)またはG00によるブロックの位置決め中に(タイプMVS_SNSのM3)、スピンドルを必要な速度に設定する必要があるためです。

Mファンクションでは、暗黙の同期が、G01/G02/G03/G151などによる加工操作への切り替えの場合に承認をチェックするだけのオプションです。この動作は同期モードMVS_SLMによって達成されます。他の同期モードP-CHAN-00027に関してのみ識別子を使用できます。

プログラミング例

```
:
N10 G00 M03 S1000 Z600          (M03: Synchronization mode MVS_SLM)
N20 X100 Y100
N30 Z400
N40 G01 Z200                    (Check, if M03 is acknowledged)
:
```

N40では、インターポレータが、即時減速の開始時におけるMファンクションの承認をチェックします。承認が出力されている場合、ブロックの終了時に中断はありません。承認が出力されていない場合、減速が起こり、ブロックの終了まで承認が出力されていない場合、システムが目標点で停止します。

動作ブロックによる同期まで、その他のチャンネル固有のMファンクションをプログラムすることが可能です。チャンネル固有のMファンクションの同期は、軸特有の同期と完全に並行して処理されます。

15.6.2 明示的な同期

位置決めにG01が使用されている場合、章暗黙の同期 [▶ 491]に従う暗黙の同期を行うことはできません。

ブロック間の同期の場合、コマンド#EXPL SYNをこのために使用することができます。これにより、Mファンクションの明示的な同期が可能になります。

#EXPL SYN	(ノンモーダル)

この追加コマンドを使用して同期する必要があるMファンクションは、チャンネルパラメータリストP-CHAN-00027の同期モードMVS_SLPで定義します。他の同期モード(P-CHAN-00027)に関してのみ識別子を使用することができます。

プログラミング例

```
:
N10 G01 M03 S1000 Z200 F5000    (M03: Synchronization mode MVS_SLP)
N20 X100 Y100
N30 Z400
N40 #EXPL SYN                    (Check, if M03 is acknowledged)
:
```

即時減速時に、経路が命令「#EXPL SYN」に基づいて、承認が届いているかどうかをチェックします。届いていない場合、ランプダウンが発生します。

その他のチャンネル特有/軸特有のMファンクションは、同期コマンドに先立って処理することができます。

15.7 スピンドルMファンクションの同期

インターポレータと関連するスピンドルの間の同期は直接行われます。つまり、M03、M04 (速度到達)、およびM05 (速度ゼロ)の承認がスピンドル自体で行われます。既存の同期モードP-CHAN-00027に加えて設定できるビットPLC_INFOによって、PLCも承認のために通知する必要があるかどうかが決められます。この場合、以下のことに注意してください。

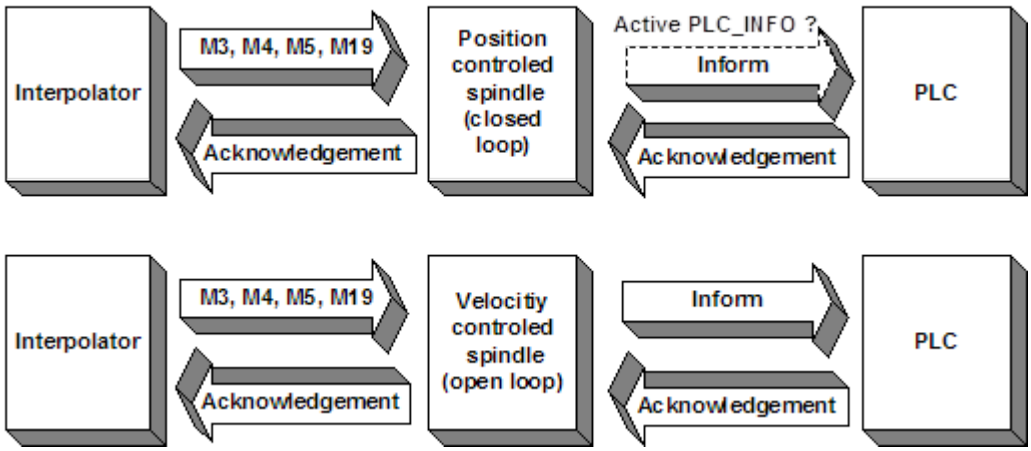
速度制御対象スピンドルの場合、各スピンドルMファンクションでは一般的にPLCも承認のために自動的に通知されます。そのため、PLC_INFOビットも設定する必要はありません。

位置制御対象スピンドルの場合、PLC_INFOビットを使用するのが実用的です。この場合、同期モードに加え、各スピンドルMファンクションについてPLC_INFOビットを設定することができます。これにより、PLCが承認のために通知されるようになります。

チャンネルパラメータリスト[1]からの抽出:

スピンドルS1が位置制御対象スピンドルであること。

```
:
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
:
```



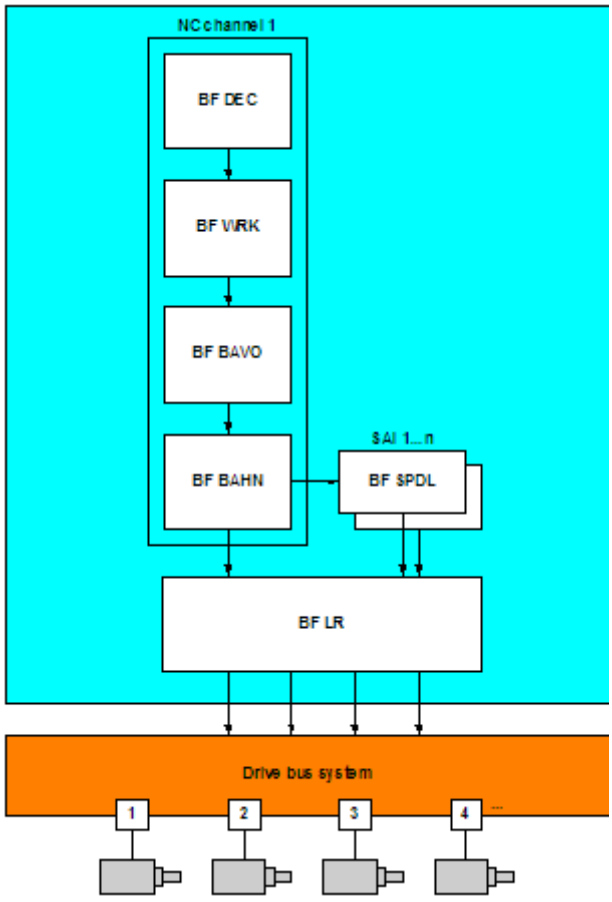
15.8 PLCopenプログラミング

Within the scope of the Motion Control Platform (MCP) different function blocks (FB) for the motion tasks are made available. This FB's act on a single axis and are operated via the PLC. Here each axis in the system is configured as a so-called **S**ingle **A**xis **I**nterpolator (SAI).

Alternatively such axes also can be controlled via the NC program, because in the system a SAI always is configured like a normal spindle. For that purpose for the following FB's special NC commands are available. These commands enable a PLCopen conformable programming in NC syntax.

- MC_Home Reference point travel
- MC_MoveAbsolute Movement of the axis to an absolute position
- MC_MoveAdditive Relative motion in addition to the commanded position
- MC_MoveRelative Relative motion in addition to the current position
- MC_MoveSuperImposed Relative motion in addition to a motion already active
- MC_MoveVelocity Endless motion at the specified speed
- MC_Stop Stopping of an axis motion
- MC_GearIn Gear coupling with a gear ratio
- MC_GearOut Releasing of a gear coupling
- MC_Phasing Phase offset of couplings

The following topology displays the basic integration of the SAI (spindle) axis in the overall system:



In part program a SAI axis is programmed in spindle specific syntax. For that reason it has to be configured by its name and further data analogous to a normal spindle in the channel parameter list. The most important settings in the channel parameters especially are:

- spdl_anzahl (P-CHAN-00082) – Total number of (SAI) spindles
- bezeichnung (P-CHAN-00007) – Name of the (SAI) spindle
- log_achs_nr (P-CHAN-00036) – Logical axis number of the (SAI) spindle

For more information see the documentation [1]-Chapter: Configuration of spindles- and chapter Configuring spindles [▶ 446] .

The PLCopen functions

- MC_MoveSuperImposed
- MC_GearIn
- MC_GearOut
- MC_Phasing

require additional specific SAI characteristics of the spindle axis, which are configured in the axes parameters. The necessary settings are described in the documentation [2]-Chapter: Settings for SAI- .

In the following for each NC command the correspondig FB is displayed. The syntax of these commands and the units of the programmed values are based on the corresponding input pins (VAR_INPUT) of the assigned FB's.

General syntax of a (SAI-)NC command:

```
<spindle_name>[ <FB name> <Input_pin1> < Input_pin2> < Input_pin n...> { \ } ]
```

The axis name on the beginning of the NC command addresses the (SAI) spindle axis, which can be activated via the NC channel.

The description of the input pins and the units and value ranges can also be taken from the documentation [9].

The input pin "Execute" always is set implicitly by the programming of the NC command. That's why for this pin no specific keyword is available.

Within the brackets [...] the line separator '\' can be used for a clear programming of the command over multiple lines.

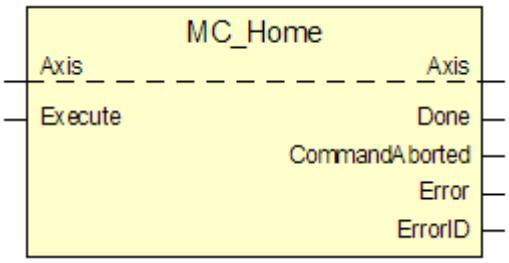
15.8.1 コマンドMC_Home

軸の基準点移動をMC_Homeで命令します。軸がこのコマンドに対応する方法は、基本的に参照操作の種類によって異なります。

NCコマンド:

<*axis_name*>[MC_Home]

PLCopenでのブロック図:



プログラミング例

```
S[MC_Home]
```

15.8.2 コマンドMC_MoveAbsolute

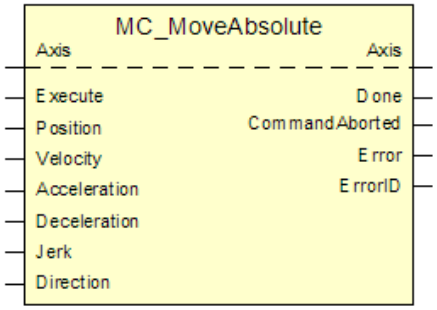
絶対位置への軸の移動をMC_MoveAbsoluteで命令します。この動作は、常に一定の設定入力「Jerk」によって加々速度制限されます。この値は「加速」と「減速」の両方に有効です。

NCコマンド:

<*axis_name*>[MC_MoveAbsolute Position=<*expr*> Velocity=<*expr*> Acceleration=<*expr*> Deceleration=<*expr*> Jerk=<*expr*> Direction=<*expr*>]

PLCopenでのブロック図:

入力ピン	単位
位置	[0.1µmまたは10^{-4}°]
速度	[1µm/sまたは10^{-3}°/s]
加速	[1mm/s^2または1°/s^2]
減速	[1mm/s^2または1°/s^2]
加々速度	[1mm/s^3または1°/s^3]
方向	1 正方向 2 最短距離 3 負方向 4 現在の方向



プログラミング例

```
S[MC_MoveAbsolute  Position=133  Velocity=1000  Acceleration=500 \
  Deceleration=600 Jerk=20000  Direction=2]
```

15.8.3 コマンドMC_MoveAdditive

軸が「Discrete Motion」状態である場合、MC_MoveAdditiveを使用して、命令された位置に加えて相対動作を命令します。この動作は、常に一定の設定入力「Jerk」によって加々速度制限されます。この値は「加速」と「減速」の両方に有効です。

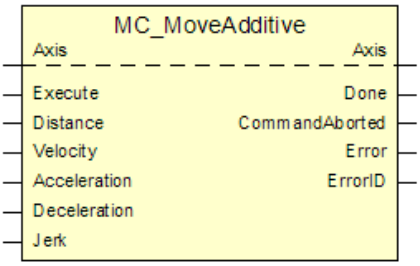
軸が「Continuous Motion」状態であり、このFBからコマンドを受け取る場合、命令時の現在の位置からの相対距離が追加されます。

NCコマンド:

<*axis_name*>[**MC_MoveAdditive Distance=**<*expr*> **Velocity=**<*expr*> **Acceleration=**<*expr*>
Deceleration=<*expr*> **Jerk=**<*expr*>]

PLCopenでのブロック図:

入力ピン	単位
距離	[0.1µmまたは10^{-4}°]
速度	[1µm/sまたは10^{-3}°/s]
加速	[1mm/s^2または1°/s^2]
減速	[1mm/s^2または1°/s^2]
加々速度	[1mm/s^3または1°/s^3]



プログラミング例

```
S[MC_MoveAdditive  Distance=277  Velocity=1100  Acceleration=550 \
  Deceleration=660  Jerk=22000]
```

15.8.4 コマンドMC_MoveRelative

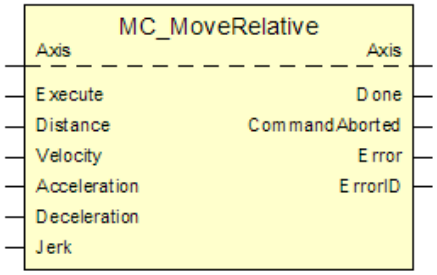
現在の位置に加え、MC_MoveRelativeを使用して相対動作を命令します。これは、軸が「Discrete Motion」状態または「Continuous Motion」状態のいずれであっても関係ありません。この動作は、常に一定の設定入力「Jerk」によって加々速度制限されます。この値は「加速」と「減速」の両方に有効です。

NCコマンド:

<*axis_name*>**[MC_MoveRelative Distance=**<*expr*>** Velocity=**<*expr*>** Acceleration=**<*expr*>
Deceleration=<*expr*>** Jerk=**<*expr*>**]**

PLCopenでのブロック図:

入力ピン	単位
距離	[0.1µmまたは10^{-4}°]
速度	[1µm/sまたは10^{-3}°/s]
加速	[1mm/s^2または1°/s^2]
減速	[1mm/s^2または1°/s^2]
加々速度	[1mm/s^3または1°/s^3]



プログラミング例

```
S[MC_MoveRelative  Distance=321  Velocity=1200  Acceleration=555 \
  Deceleration=666  Jerk=22000]
```

15.8.5 コマンドMC_MoveSuperImposed

既に有効な動作に加え、MC_MoveSuperImposedを使用して相対動作を命令します。有効な動作は中断されず、命令された動作と重ね合わせられます。この動作は、常に一定の設定入力「Jerk」によって加々速度制限されます。この値は「加速」と「減速」の両方に有効です。

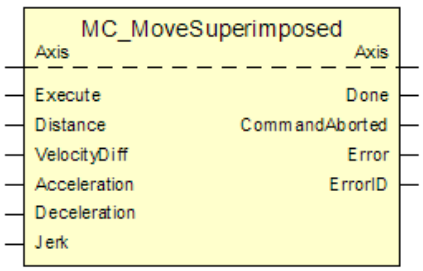
重ね合わせられる補間の場合に「加速度」値も重ね合わせられるため、軸に動的に過大な負荷がかからないように、対応する軸パラメータを定義する必要があります。

NCコマンド:

<*axis_name*>[MC_MoveSuperImposed Distance=<*expr*> VelocityDiff=<*expr*>
Acceleration=<*expr*> Deceleration=<*expr*> Jerk=<*expr*>]

PLCopenでのブロック図:

入力ピン	単位
距離	[0.1µmまたは10^{-4}°]
VelocityDiff	[1µm/sまたは10^{-3}°/s]
加速	[1mm/s^2または1°/s^2]
減速	[1mm/s^2または1°/s^2]
加々速度	[1mm/s^3または1°/s^3]



プログラミング例

```
S[MC_MoveSuperImposed  Distance=321  VelocityDiff=783  Acceleration=811 \
  Deceleration=922 Jerk=45000]
```

15.8.6 コマンドMC_MoveVelocity

MC_MoveVelocityを使用して、指定された速度でのエンドレス動作を命令します。この動作は、常に一定の設定入力「Jerk」によって加々速度制限されます。この値は「加速」と「減速」の両方に有効です。

動作を停止するには、新しいコマンドを軸に送る別のコマンドによって、このコマンドを中断する必要があります。

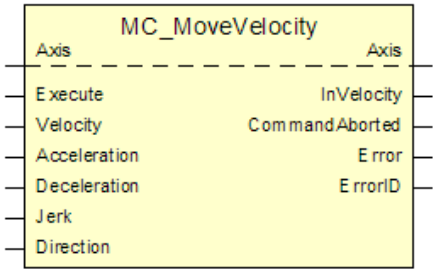
MC_MoveSuperImposedと組み合わせた場合、出力「in_velocity」はTRUEのままです。

NCコマンド:

<*axis_name*>[**MC_MoveVelocity Velocity=**<*expr*> **Acceleration=**<*expr*> **Deceleration=**<*expr*> Jerk=<*expr*> **Direction=**<*expr*>]

PLCopenでのブロック図:

入力ピン	単位
速度	[1µm/sまたは10^{-3} °/s]
加速	[1mm/s^2または1°/s^2]
減速	[1mm/s^2または1°/s^2]
加々速度	[1mm/s^3または1°/s^3]
方向	1 正方向 3 負方向 4 現在の方向



プログラミング例

```
S[MC_MoveVelocity  Velocity=1333  Acceleration=770  Deceleration=880 \
  Jerk=10000  Direction=1]
```

15.8.7 コマンドMC_Stop

MC_Stopは動作を停止させ、軸を「Stopping」状態にします。この動作停止は、常に減速率の一定の設定入力「jerk」によって加々速度制限されます。

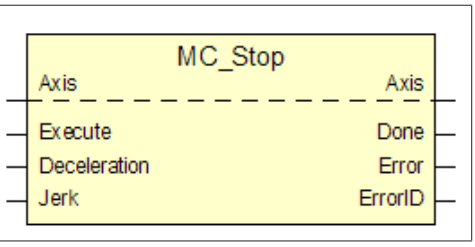
このコマンドは、他の(SAI)動作コマンドによって、実行中のコマンドをすべて中止します。

NCコマンド:

<*axis_name*>[MC_Stop Deceleration=<*expr*> Jerk=<*expr*>]

PLCopenでのブロック図:

入力ピン	単位
減速	[1mm/s^2または1°/s^2]
加々速度	[1mm/s^3または1°/s^3]



プログラミング例

```
S[MC_Stop  Deceleration=999  Jerk=25000]
```

15.8.8 コマンドMC_GearIn

MC_GearInはギヤ比とのギヤ結合を命令します。ギヤ比は主軸と従軸の間の速度比を定義します。速度の同期が加々速度制限されます。「Jerk」がNCコマンドで設定されます。

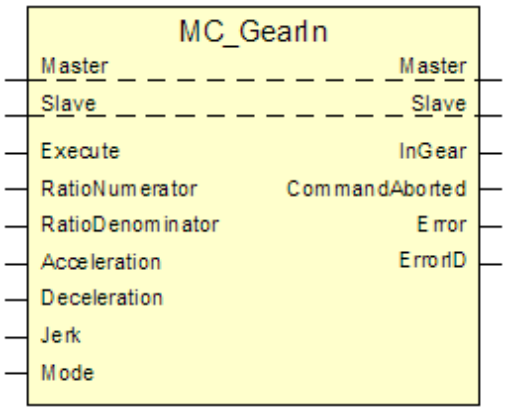
従軸をマスタ設定点または実際のマスタ値に結合することができます。この選択は「Mode」パラメータで行います。

NCコマンド:

<*axis_name*>[**MC_GearIn** **Master**=<*expr*> **RatioNumerator**=<*expr*> **RatioDenominator**=<*expr*> Acceleration=<*expr*> **Deceleration**=<*expr*> **Jerk**=<*expr*> **Mode**=<*expr*>]

PLCopenでのブロック図:

入力ピン	単位	
マスタ	主軸の論理軸番号	
RatioNumerator	ギヤ比の分子	
RatioDenominator	ギヤ比の分母	
加速	[1mm/s^{2}または1°/s^2]	
減速	[1mm/s^{2}または1°/s^2]	
加々速度	[1mm/s^{3}または1°/s^3]	
モード	0	主軸と従軸の間の結合のタイプ: コマンド値に基づく結合。 自動位相補償: オフ。
	128	主軸と従軸の間の結合のタイプ: 現在値に基づく結合。 自動位相補償: オフ。
	256	自動位相補償: オン。



プログラミング例

```
S[MC_GearIn  Master=11  RatioNumerator=2 RatioDenominator=3 \
  Acceleration=500  Deceleration=600  Jerk=20000  Mode=384]
```

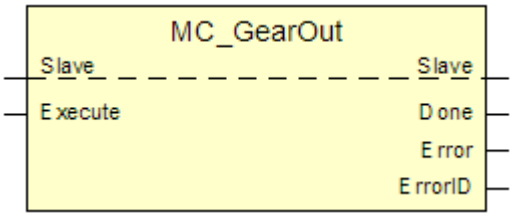
15.8.9 コマンドMC_GearOut

MC_GearOutは、速度比によって指定されている主軸への従軸の結合を解放します。スレーブの現在の速度が保たれます(エンドレス動作)。

NCコマンド:

<*axis_name*>[MC_GearOut]

PLCopenでのブロック図:



プログラミング例

```
S[MC_GearOut]
```

15.8.10 コマンドMC_Phasing

MC_Phasingを使用すると、主軸に対する従軸のオフセットが実現します。そのために、主軸の位相差が主軸の視点から指定され、従軸が加速または減速によってこの位相差をなくそうと試みます。この動作は、常に一定の設定入力「Jerk」によって加々速度制限されます。この値は「加速」と「減速」の両方に有効です。

この力学的相似により、限定期間で主軸と従軸の結合が解放されます。

カミングの場合、このコマンドによって、スレーブの視点からの「見かけの」マスタ位置が変更されます。**ギヤリング**の場合、スレーブにおける重ね合わされた動作を命令することによって、マスタとスレーブの間に位相差が生じます。そのため、ギヤリングの場合、MC_Phasingには MC_MoveSuperImposed(MC_Phasingが制御システムで実際に変換される)と同じ効果があります。

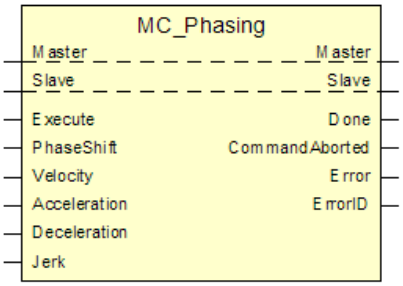
カミングの場合、動的値「Velocity」、「Acceleration」、および「Deceleration」はスレーブの視点からの「見かけの」マスタ位置の変更を指します。これに対してギヤリングでは、これらの動的値は従軸自体の重ね合わされた動作を指します。

NCコマンド:

```
<axis_name>[ MC_Phasing  Master=<expr>  PhaseShift=<expr>  Velocity=<expr>
                Acceleration=<expr>  Deceleration=<expr>  Jerk=<expr> ]
```

PLCopenでのブロック図:

入力ピン	単位
マスタ	主軸の論理軸番号
PhaseShift	[0.1μmまたは10^{-4} °]
速度	[1μm/sまたは10^{-3} °/s]
加速	[1mm/s^2または1°/s^2]
減速	[1mm/s^2または1°/s^2]
加々速度	[1mm/s^3または1°/s^3]



プログラミング例

```
S[MC_Phasing Master=11  PhaseShift=25  Velocity=1000  Acceleration=500 \
  Deceleration=600  Jerk=20000]
```

16 記号文字列(マクロ)

「C」プログラミング言語でのマクロ生成と同様に、「記号文字列」機能は、記号名(マクロ名)を実行可能なNCコードに割り当てることを可能にします。演算式と複数の複雑なNCコマンドを割り当てることができます。割り当てられたNCコードは、マクロ名が指定されているときにいつでも実行されます。この機能により、プログラミングがコンパクトで明確かつ有意義なものになります。

16.1 マクロの定義と初期化

記号文字列は、チャンネルパラメータリスト[1]-1で、またはNCプログラミングによって定義することができます。NCプログラムでは、以下の構文が必要です。

```
"<Macro_name>" = "<NC_Code>"
```

<Macro_name> *Macro_name*

<NC_Code> *置き換える必要がある文字列*

マクロ名(代入の左側l)と、置き換える必要があるテキスト(右側)は、両方とも引用符で囲む必要があります。解読処理では、大文字と小文字が区別されます。以前に定義されたマクロを再定義できるかどうかが確定されています[6]-6.39。マクロ定義はプログラム全体で有効であり、次の開始まで保持されます(ただし、マクロテーブルの初期化が明示的にプログラムされないことが条件です)。マクロ定義の最大数[6]-6.25と長さ[6]-6.37/-6.38は確定されています。

マクロ定義と他のコマンドが同じブロックで共存することができます。

バージョンV2.11.2010.02から、コマンド#INIT MACRO TABがコマンド#INIT MAKRO TABに取って代わります。互換性の理由から、このコマンドは当分の間使用できますが、新しいNCプログラムでは使用しないでください。

以下のコマンドをマクロテーブルの初期化に使用する必要があります。

```
#INIT MACRO TAB
```

このコマンドは、以前に定義されたマクロテーブルのすべてのマクロ定義を消去します。チャンネルパラメータリスト[1]-1によって事前に割り当てられたマクロは保持されます。

16.2 マクロのネスティング

マクロを代入の右側のNCコードと組み合わせて使用すること(ネスティング)が許可されています。必要に応じて、ネスティングの最大深さを設定できます[6]-6.40。ネスティングは、区切り引用符に先行する「\」文字によって示されます。マクロは、常にNCブロックの複雑な式を表します(NCコマンド、演算式、項)。これにより、関連する演算式なしでNCコマンドのアドレス文字のみをマクロが表すことが不可能になります。これに関しては以降の章で説明します。

"*<MACRO_NAME>*" = "*<NC_CODE>* \"*<MACRO_NAME_i>*\" *<NC_CODE>*"

注記

マクロには、マクロ自体の入れ子構造になったマクロ名を含めることはできません。他のマクロ名のネスティングのみ許可されています。

プログラミング例

```
N10 "POS_1" = "X500 Y200"           (Macro definition)
N20 "MOVE1" = "G01 \"POS1\" F1000"  (Macro definition with nesting)
N30 "MOVE1"                         (Macro call
M30
```

16.3 演算式の利用

マクロ名を演算式およびその部分に割り当てることができます。演算式内で再帰処理(ネスティング)も可能です。文字列で常に完全なレベル(つまり、先頭に「[」、末尾に「]」を挿入することによって、その結果が影響を受けないような項)が組み合わされていることを確認する必要があります。

プログラミング例

```
Correct:
N10 "STRING1" = "0.5"
N20 "STRING2" = "5 * 12"
N30 "STRING3" = "SIN[89.5 + \"STRING1\"]"

N40 X[-2 * "STRING1" + "STRING2" + "STRING3"]     (Moving to X60)

M30

Incorrect:
Only complete arithmetic expressions may be combined in a string
```

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

16.4 NCブロックレベルでの利用

NCプログラムでのマクロの定義とその呼び出しは、以下のとおりです。

プログラミング例

```
:
N10 "POSITION_1" = " X200 Y200 Z300 "      (Macro definition)
N20 "POSITION_2" = " X300 Y100 Z50 "       (Macro definition)
:
N200 "POSITION_1"                          (Macro call)
:
N500 "POSITION_2"                          (Macro call)
:
```

以下の例は、入れ子構造にされたマクロ定義のプログラミングを示しています。

プログラミング例

```
:
N10 " STRING_1 " = " 5*12 "                        (Macro definition)
N20 " STRING_2 " = " G \"STRING_1\" + 5 "
N30 " STRING_3 " = " M \" STRING_1\ " \" STRING_2 \" "
:
N200 " STRING_3 "                        (Call the nested macro)
:                                        (Corresponds to: N200 M60 G65)
```

16.5 アドレス文字と演算式の分離

マクロにアドレス文字のみを割り当て、演算式を2番目のマクロでプログラムまたは定義することが可能です。

これにより、主軸を参照するマクロを上位のNCプログラムで定義することが可能になります。次にこのマクロをサブプログラムまたはサイクルで使用できるため、選ばれた加工レベルからの独立性が確保されます。

プログラミング例

```
Only the address letter is contained in the alternate text. The arithmetic expression is programmed
or defined in a separate alternate text:

"1.HA" = "X"    "2.HA" = "Y" (HA: primery axis)
```

```
"Ziel_1.HA" = "COS[ P3 * 100 ] / ABS[ V.G.NP_AKT.V[ 0 ] - P12 ]"

"1.HA" "Ziel_1.HA"     "2.HA" 100
```

16.5.1 制限事項

置き換える必要があるテキスト内で行終了/文字列終了(「\0」)文字を読み取ることはできません。そのため、マクロ定義は1行に限定されます。

マクロNCコード文字列には、制御ブロック命令($)を含めることはできません。

17 5軸機能

17.1 回転工具中心点(RTCP)

この機能を使用するには、拡張パッケージ「Transformations」のライセンスが必要です。このライセンスは、標準ライセンスの範囲に含まれていません。

#TRAFO ON RTCPの選択(モーダル)		
#TRAFO OFF	RTCPの選択解除	(モーダル)

ファンクションRTCPは空間内の工具補正を表します。

RTCPを選択していると、工具の向きを変更する際に現在の工具の接触点がワークに対して静止した状態で保持されます(キネマティックトランスフォーメーションのパラメータ値を割り当てる場合は、工具先端オフセットパラメータP-CHAN-00094、P-TOOL-00009の割り当てに注意してください)。

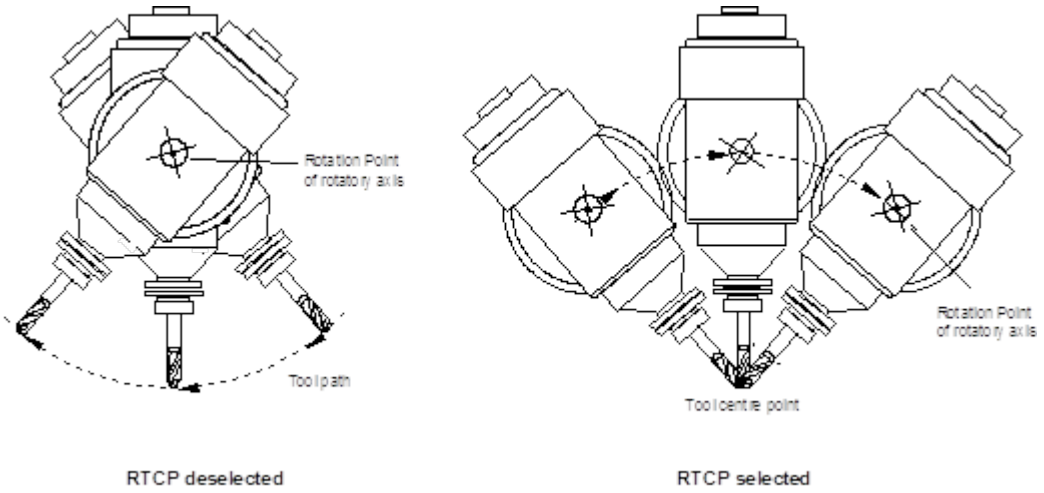


図 107: 図16-1: RTCP動作制御

図16-1の左側の図は、回転機械軸が移動した場合の動作を示しています。

RTCPによって回転の中心点が工具先端に移動します(工具の回転の中心点)。工具の動作に基づくX、Y、Z軸のオフセットが、対応する逆動作によって各サイクルで補正されます。

X、Y、Z軸のみがキネマテックトランスフォーメーションの出力となり、機械の回転軸は通常通りにプログラムされます。

TLCが有効な場合は、RTCPファンクションを選択できません(16.2章を参照してください)。

注記

有効なキネマティックトランスフォーメーションでは、ax_versatz[<*ax_index*>] (P-TOOL-00006)の軸特有の工具オフセットのみが考慮されます(軸が変換ファンクションの影響を受けない場合)。RTCPの場合には変換タイプに依存するため、 これらの軸はインデックス > 2 を有する軸です。

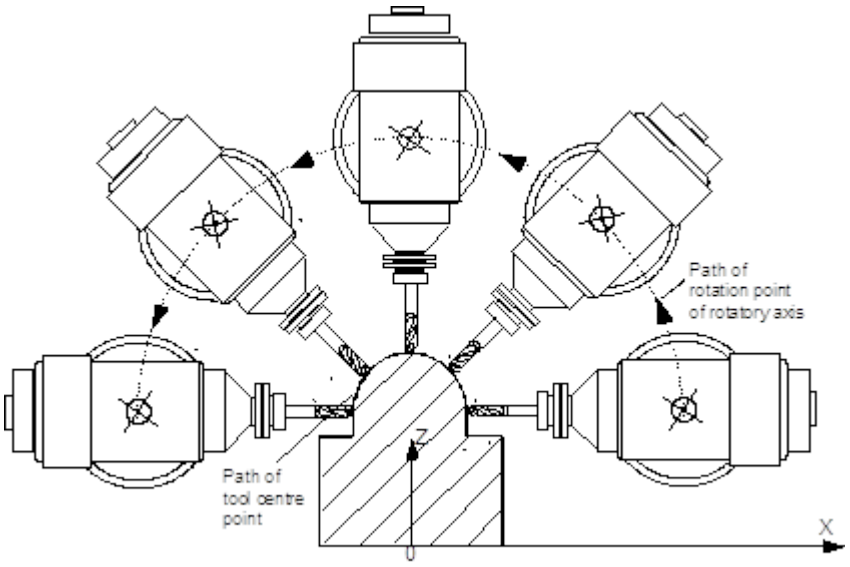
最初の3軸(インデックス0、1、2)の軸特有の工具オフセットは、**有効**な変換時には考慮されません。有効な変換中にこれらの軸に対しても工具オフセットが効果的でなければならない場合、上述の工具のキネマティック固有のオフセット(P-TOOL-00009)で値を入力する必要があります。

プログラミング例1

```
N10 T1  D1                  (Selection of the 2½-D tool correction)
N20 #TRAFO ON               (Selection of RTCP)

N30 G01  F100  B45  C30     (Programming the rotary axes modifies tool)
                            (orientation. Tool contact point remains sta-)
                            (tionary)
N40...
 .
 .

N100 #TRAFO OFF             (Deselection of RTCP)

N200 M30
```

プログラミング例2

```
:
N10 #KIN ID [1]                     (Selection of machine kinematic)
N20 #TRAFO ON                       (Selection of RTCP)
N30 G01 G18 X20 Y0 Z25 B90 F500     (Move workpiece from the right side)
N40 G91 X-8                         (Move to start position)
N50 G90 G02 X-12 I-12 B-90 F2000    (Processing)
N60...
:
```



17.2 工具長補正(TLC)

この機能を使用するには、拡張パッケージ「Transformations」のライセンスが必要です。このライセンスは、標準ライセンスの範囲には含まれていません。

#TLC ON [*<delta_tool_length>* **]**	TLCの選択	(モーダル)
#TLC OFF	TLCの選択解除	(モーダル)

<delta_tool_length> 工具長の差[mm]。

TLCは、プログラミングシステムによって作成されたNCプログラムを有効にし、工具長が変更された場合であっても特定の長さを機械で使用するように考慮します。工具の新しいオフセットや半径は補正できないことに注意してください。

RTCPが有効である場合にTLCファンクションを選択することはできません。

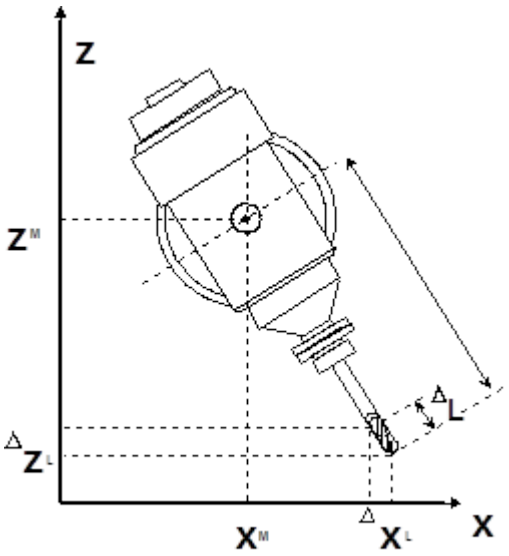


図 108: 図16-2: 工具長の変更時に各サイクルのTLCがΔLを変換します。

プログラミング例

```
N10 #TLC ON [15]       (Selection of TLC with extended tool, WZL =15 mm)
N20 .
.N100 #TLC OFF         (De-selection of TLC)

N200 #TLC ON [-20]     (Selection of TLC with shortened tool, WZL = -20 mm)
N210 .
.
N300 #TLC OFF          (De-selection of TLC)

N200 M30
```

17.3 工具の向き(TOOL ORI CS)

#TOOL ORI CS	工具アライメントの選択	(モーダル)

#TOOL ORI CSに続く最初の移動ブロックが、現在のBKSの第3の主軸と平行になるように工具を位置合わせします(W_0またはMKSも可能です)。このブロックで回転軸をプログラムすると、向きに有効な位置が無効になります。

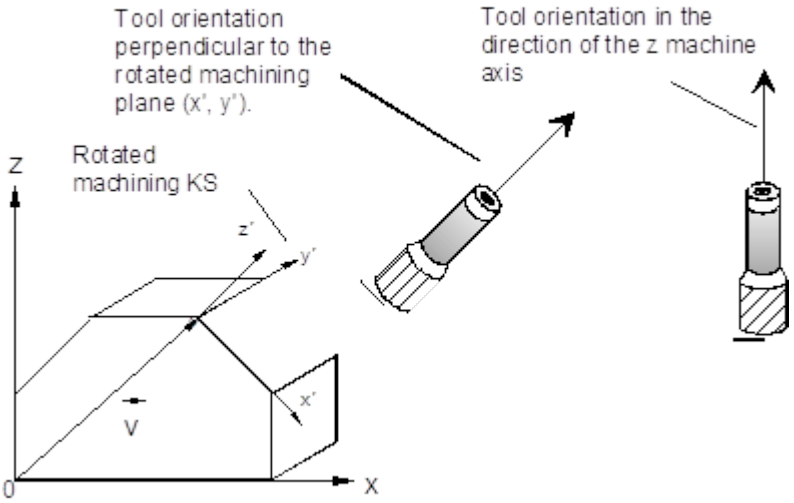


図 109: 図16-3: 加工平面(X-Y)に対して垂直になるように工具を位置合わせ

プログラミング例

```
N10 B10  C20           (Slanting the tool)
N20 #TOOL ORI CS       (In next motion block, align tool in parallel to)
                       (the Z axis of the current BKS, here MKS)
N30 X0  Y0             (Motion block in the MKS, tool is aligned B=0,C=0)

N40 B25  C-80          (Slanting the tool)
N50 #TOOL ORI CS       (Align tool in next motion block)
N60 #TRAFO ON          (RTCP selection)

N70 #CS ON[0,0,0,-80,-30,45]

                       (Transition to a rotated BKS)
N80 X100               (Motion block in BKS. Tool alignm. from N50 refers)
                       (to the MKS, thus B=0,C=0)
N90 #TOOL ORI CS       (Align tool in rotated BKS)
N100 Y150              (Tool is aligned in parallel to the Z axis of BKS)

N110 #TOOL ORI CS
N120 Z100  B45 C10     (#TOOL ORI CS is without effect in programming of)
                       (rotary axes)
N130 G18               (Change to the Z-X-interpolation plane)
N140 #TOOL ORI CS      (Align tool in parallel to the Y axis)
N150 X0                (Tool aligned perpendicularly to the X-Z machining)
                       (plane)

M30
```

17.4 機械のキネマティクス(KIN ID)

この機能を使用するには、拡張パッケージ「Transformations」のライセンスが必要です。このライセンスは、標準ライセンスの範囲には含まれていません。

以下のコマンドを機械/工具ヘッドのキネマティクスの定義に使用します。

#KIN ID [[*<expr>*]] 機械/工具ヘッドのキネマティックの定義 (モーダル)

<expr> キネマティクスIDは、機械および/または工具ヘッド関連のキネマティクスのタイプを識別するために使用します。システム起動後の初期設定は、チャンネルパラメータリストP-CHAN-00032のkinematik_id要素の割り当てから生じます。

パラメータなしで#KIN IDをプログラムすると、初期設定のキネマティクスIDが設定されます。

また、工具データでのkin_id要素の割り当てによって、キネマティクス変更を工具交換とともに自動的に行うことができます。

キネマティクスIDが不明であると、エラーメッセージが出力され、RTCP TLCまたはTOOL ORI CSの選択時に解読処理が停止します。

注記

RTCPまたはTLCが有効な場合に、#KIN ID...を使用してキネマティクスを変更することはできません。

プログラミング例

```
N10 #TOOL ORI CS    (Aligning tool; default kin. from P-CHAN-00032 valid)
N20 T1 D1           (Tool selection)
N30 #TRAFO ON       (RTCP selection, default kin. from P-CHAN-00032 valid)
.
.
N40 #TRAFO OFF      (RTCP de-selection)
N50 #KIN ID[2]      (Selection of the kinematics that is stored in the
                    (internal library under the ID '2').
N60 #TRAFO ON       (RTCP selection, kinematics 2)
.
.
N70 #TRAFO OFF      (RTCP de-selection)

N80 M30
```

17.5 補正移動なしの位置決め(PTP)

この機能を使用するには、拡張パッケージ「Transformations」のライセンスが必要です。このライセンスは、標準ライセンスの範囲には含まれていません。

#PTP ON	有効な変換中のPTPモーション制御の選択	(モーダル)
#PTP OFF	有効な変換中のPTPモーション制御の選択解除	

キネマティックトランスフォーメーションの選択後に工具の位置決めと配向を行うと、工具中心点(TCP)が経路上を移動するため、機械軸で補正移動が生じます。この補正移動が望ましくない場合、上記のコマンドを使用して、最適化されたタイミングの移動を実行できます。

5軸加工機では、移動は工具ヘッドの基準点に基づいています。TCPとは対照的に、この基準点は、プログラムされた送り(Fワード)または早送りによって直線上を移動します(図16-4を参照)。

非直交機械構造物(例えば、ロボットや三脚)では、TCPも基準点も直線上を移動しません。プログラムされた送り(Fワード)または早送りは、機械軸に対して作用します。しかし、移動の終了時に、プログラムされたPCS目標位置にTCPがあることが保証されます。

移動のプログラミングはPCSプログラミングと同じであり、MCS座標へのPCS座標の変換が制御システムによって実行されます。オフセットと工具データの包含が、有効なキネマティックトランスフォーメーション中と同じ方法で実行されます。これが、コマンド#WCS TO MCSの使用と大きく異なる点です。

注記

CNCのリアルタイム部分での有効なPTPモーション制御中は、PCS座標がACS値として表示されます。

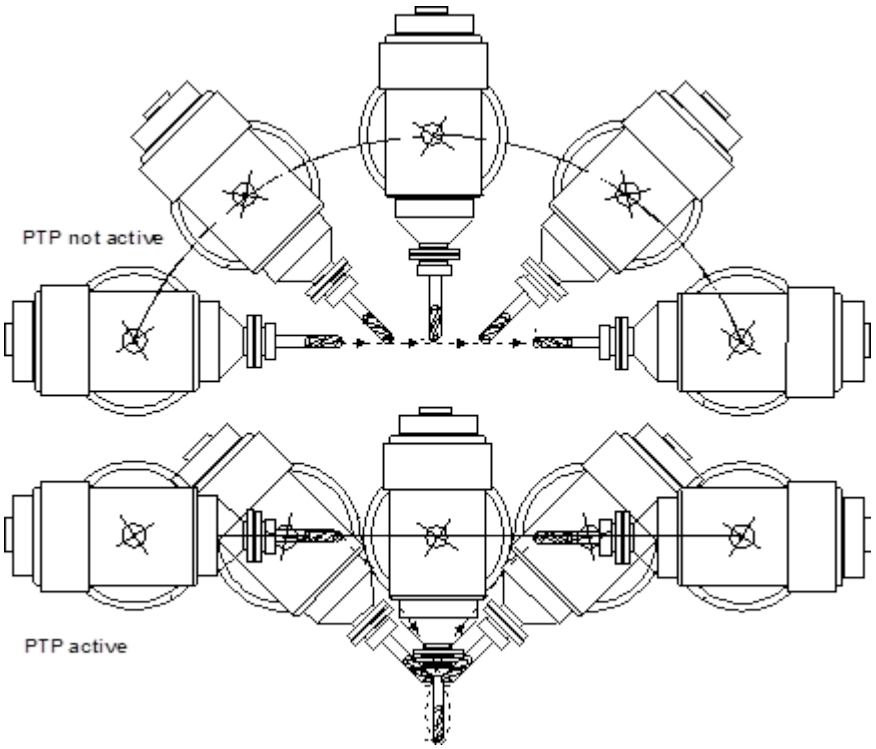


図 110: 図16-4: 変換PTP動作制御

有効なPTPモーション制御中には、追加の座標変換の選択/選択解除(#(A)CS ON/OFF、#MCS ON/OFFなど)はできません。

プログラミング例

```
Example to Fig. 16-4:

N10 T1 D1                   (Tool selection)
N20 G00 X-200 Y0 Z0 A90     (MCS start position)
N30 #TRAFO ON               (RTCP selection, default kinematic)
                            (from P-CHAN-00032 is valid)
N40 #PTP ON                 (Trafo PTP motion ON)
N50 G00 X200 Y0 Z0 A-90     (PCS target position)
...
N180 #PTP OFF               (Trafo PTP motion OFF)
N185 G01 X100 Y150 F5000
...
N500 #TRAFO OFF             (RTCP deselection)
N999 M30


Example with automatic tool orientation:

N10 T1 D1                   (Tool selection)
N20 #TRAFO ON               (RTCP selection, default kinematic)
                            (from P-CHAN-00032 is valid)
N30 #TOOL ORI CS            (Orientate tool, default kinematic)
                            (from P-CHAN-00032 is valid)
N40 #PTP ON                 (Trafo PTP motion ON)
N50.G00 X0 Y0 Z100 G90      (PCS start position)
...
N180 #PTP OFF               (Trafo PTP motion OFF)
N185 G01 X100 Y150 F5000
...
N500 #TRAFO OFF             (RTCP deselection)
N999 M30
```

特定の軸に対するPTPモーション制御の組み合わせでは、場合によって、技術的な理由から(例えば、ソフトウェア制限内での軸の位置決めを保証するために)動作を停止する必要があります。

この目的のために、PTPプログラミングが有効であるときに単軸の移動を#AX LOCK/UNLOCKコマンドで一時的に停止することができます。

#AX LOCK [{AX *<axis_name>*]	ロックする必要がある軸の選択
#AX UNLOCK [ALL]	すべてのロックされた軸の解放

AX*<axis_name>*	ロックする必要がある軸の名前
ALL	ロックされた軸の解放。複数の軸がロックされている場合、それらをまとめて解放することのみ可能です。1つの特定の軸のみを解放することはできません。 単一の軸のみロックされている場合も、ALLで解放する必要があります。

#AX LOCK、#AX UNLOCKのプログラミングは、有効な#PTP中にのみ可能です。

キネマティックトランスフォーメーションのロックされたACS出力軸は、G00およびG01で移動しません。

プログラミング例

```
Correct:

N10 #CYL[EDGES=4 ROUNDING=5 LENGTH1=40 LENGTH2=40]
N20 #PTP ON
N30 #AX LOCK[AX=Z]
N40 G00 G90 U30
……
N60 AX UNLOCK [ALL]
N70 #PTP OFF
N80 G01 G90 Z0 F3000
N90 G01 U40 F2000
…….

Wrong:

N10 #AX LOCK[AX=Z]
N20 #CYL[EDGES=4 ROUNDING=5 LENGTH1=40 LENGTH2=40]
N30 #PTP ON
N40 G00 G90 U30
……..
N60 #AX UNLOCK [ALL] (position request)
N70 #PTP OFF
N80 G01

…….
N200 #CYL OFF
```

17.6 座標系

17.6.1 加工座標系(CS)の定義

CSの定義と保存:	
#CS DEF [[*<CS-ID>*]] [<v1>,<v2>,<v3>,<*φ*1>,<*φ*2>,<*φ3*>]	(モーダル)

同時有効化の定義と保存:	
#CS ON [[*<CS-ID>*]] [<v1>,<v2>,<v3>,<*φ*1>,<*φ*2>,<*φ3*>]	(モーダル)

#CS ON [*<CS-ID>*]	保存されたCSの選択	(モーダル)

#CS ON	最後に定義されたCSの選択	(モーダル)

#CS OFF	最後に有効になったCSの選択解除	(モーダル)
最後に有効になったCSの選択解除のみ許可されるため、パラメータCS-IDをここでプログラムすることはできません。		

<CS-ID> 座標系ID。プログラム開始時にCS-IDに初期値1が割り当てられます。#CS DEFまたは#CS ONでCS-IDがプログラムされない場合、次の空きCS-IDが自動的に計算されます。この方法でプログラムされたCSは、#CS OFFで選択解除された後で再び使用することができません。

<vi> 並進シフトベクトルの成分。(これらは、G17の場合のシーケンスにおける主軸を指します)。

<φi> 回転の角度

CSは、相対オフセット(図16-5のV_2)と、現在のワーク座標系(**WKS**)に対する回転によって特徴付けられます。現在のゼロオフセット、クランプ位置オフセット、および座標プリセット(図16-5のV_1)によって、機械座標系(**MKS**)に対するBKSの位置が決まります。

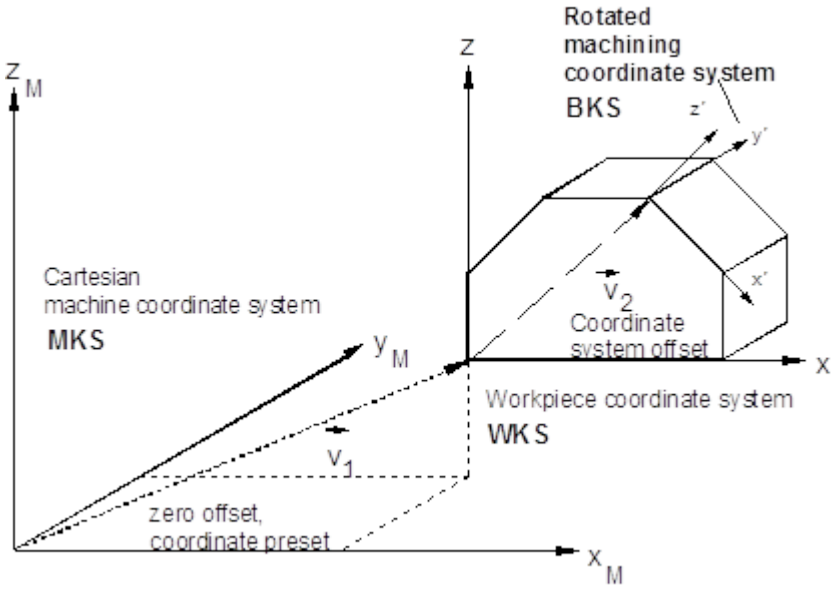


図 111: 図16-5: 傾斜面での加工

回転φ1/φ2/φ3は、数学的に正の方向(図16-6)で以下の順序で行われます。

第3軸(例えば、z)の周りの角度φ3の第1回転

新しい第2軸(例えば、y´)の周りの角度φ2の第2回転

新しい第1軸(例えば、x´´)の周りの角度φ1の第3回転

(これらの軸の順序は常にG17での主軸の順序に対応し、現在選択されているG17/G18/G19から独立しています)。

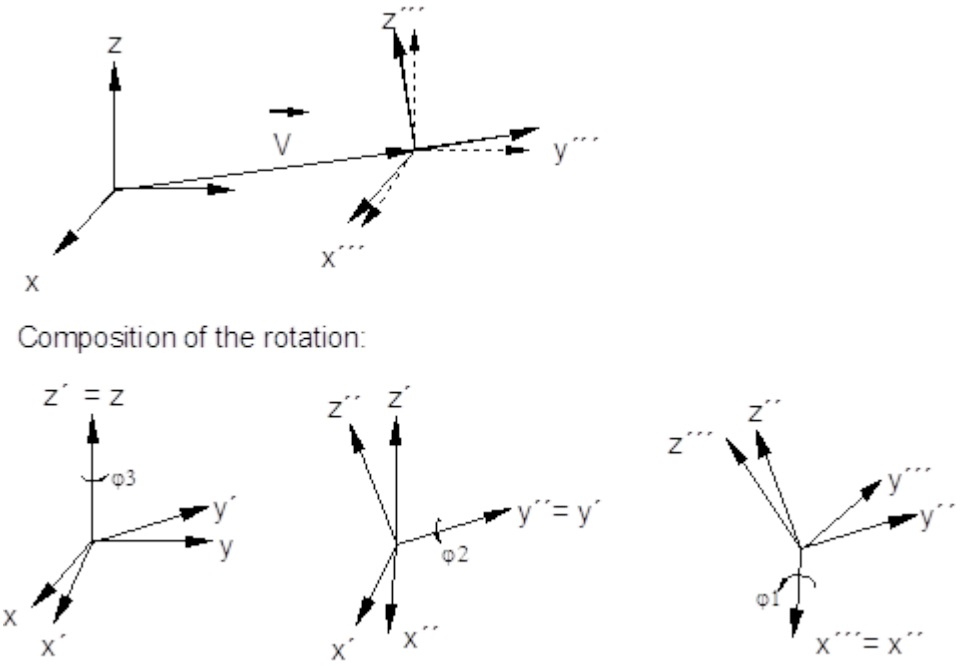


図 112: 図16-6: 1つのシフトと3つの回転による加工座標系(CS)の定義

#CS DEF <[CS-ID]> [...]またはCS ON <[CS-ID]> [...]によって定義されているCSの位置(現在のWKSに対する)が保存され、パラメータを指定することなく#CS ONを使用して再選択することができます。MKSの全体的シフトがそのうちに変更されている場合、CSはMKSに対する新しい位置を持つことになります。

ゼロオフセットと座標プリセットをCSで加工中にプログラムすることができます。これらの値はCSが選択解除されるまで有効なだけで、保存されることはありません。

軸名称がCSで保持されます。

プログラミング例1

```
N005 P1 = 2
N010 #CS DEF [1][P1,15,5,20,30,45]  (Definition and activation of a CS)
                                    (under ID 1:)
                                    (Relative shifts: X2, Y15, Z5)
                                  (Rotations: 45°around Z, 30°around Y')
                                  (20°around X'')
N020 #CS ON[1]                      (Activation of CS with ID 1)
:
:
N100 #CS OFF                        (De-activation of CS with ID 1)
:
:
N200 P1=10
N210 #CS ON [P1,15,5,2,3,60]     (Definition and activation of a CS)
                                 (under the automatically defined ID 2)
:
:
N300 #CS OFF     (Deselection of the latest activated CS (ID2))
                 (After that the CS with ID2 is deleted!)
:
N400 M30
```

プログラミング例2

```
N5 P1 = 2
N10 #CS DEF [3][P1,15,5,2,3,4.5]    (Definition and storage of a CS (ID 3))
N20 #CS DEF [2][P1,15,5,2,3,4.5]    (Definition and storage of a CS (ID 2))
N30 #CS DEF [5][0,1,2,0,30,30]      (Definition and storage of a CS (ID 5))
N30 #CS ON                          (Activation of a CS with the lastly)     (programmed ID 5)
.
N50 #CS OFF
N60 #CS ON [3]                      (Activation of the CS with ID 3)
.
N80 #CS OFF
N90 #CS DEF [3][1,1.2,1.3,0,0,33]   (Redefinition of the CS with ID 3)
.
M30
```

プログラミング例3

例えばCS ON [...]を使用して(*CS_ID*なしで)複数の座標系を次々に選択した場合、それらが、連鎖する新しい全体的CSを形成します。これは、対応する#CS OFFによって段階的に選択解除する必要があります。

CS IDを使用した/使用しない組み合わせCS選択ができますが、NCプログラムの明確さの理由から推奨されません。

CSの複数プログラミングの例(*CS_ID*を使用しない):

```
N010 #CS ON [0,0,0,0,0,20]  (Definition and activation of a CS under)
                            (the automatically defined ID 1)
                            (No shifts, only rotation 20°around Z)
:
:
N050 #CS ON [0,0,0,0,0,30]  (Definition and activation of a CS under)
                            (the automatically defined ID 2)
                            (No shifts, only rotation 30° around Z)

->(The result is a CS with a total rotation of 50° around Z)
:
N100 #CS OFF    (Deselection of CS with ID 2, after that the CS)
                (with ID 2 is deleted!)

->(CS with ID 1 with a rotation of 20° around Z remains active)

:
:
N200 #CS OFF    (Deselection of CS with ID 1, after that the CS)
                (with ID 1 is deleted and all CS are deselected!)
:
:
N400 M30
```

17.6.2 固定具適合のための座標系(ACS)の定義

固定具適合座標系(**ACS**)は、ワークまたはワークパレットの傾斜位置を補正する役割を果たします。その定義、選択、および選択解除は加工CSと同様に行われます。

ACSの定義と保存:

#ACS DEF [**[***<ACS-ID>***]**] **[<v1>,<v2>,<v3>,<*φ*1>,<*φ*2>,<*φ3*>]** (モーダル)

同時有効化の定義と保存:

#ACS ON [**[***<ACS-ID>***]**] **[** <v1>,<v2>,<v3>,<φ1>,<φ2>,<φ3> **]** (モーダル)

#ACS ON [*<ACS-ID>***]**	保存されたACSの選択	(モーダル)

#ACS ON	最後に定義されたACSの選択	(モーダル)

#ACS OFF	最後に有効になったACSの選択解除	(モーダル)

最後に有効になったACSの選択解除のみ許可されるため、パラメータACS-IDをここでプログラムすることはできません。

<CS-ID> 座標系ID。プログラム開始時にACS-IDに初期値1が割り当てられます。#ACS DEFまたは#ACS ONでACS-IDをプログラムしないと、次の空きACS-IDが自動的に計算されます。この方法でプログラムされたACSは、#ACS OFFで選択解除された後で再び使用することができません。

<vi> 並進シフトベクトルの成分。(これらは、G17の場合のシーケンスにおける主軸を指します)。

<φi> 回転の角度

ACSはモーダルであり、CSから独立して有効/無効にすることができます。

ACSでは、ゼロオフセットと座標プリセットをプログラムすることができます。ただし、これらはACSの無効化まで有効なだけで、保存されることはありません。

プログラミング例1

```
N005 P1 = 2
N010 #ACS DEF [1][P1,15,5,20,30,45] (Definition and activation of an ACS)
                                    (under ID 1:)
                                    (Relative shifts: X2, Y15, Z5)
                                  (Rotations: 45°around Z, 30°around Y')
                                  (20°around X'')
N020 #ACS ON[1]                     (Activation of ACS with ID 1)
:
:
N100 #ACS OFF                       (De-activation of ACS with ID 1)
```

```
:
:
N200 P1=10
N210 #ACS ON [P1,15,5,2,3,60]    (Definition and activation of a ACS)
                                 (under the automatic defined ID 2)
:
:
N300 #ACS OFF    (Deselection of the latest activated ACS (ID2))
                 (After that the ACS with ID2 is deleted!)
:
N400 M30
```

プログラミング例2

```
N10 #ACS DEF [1][10,15,5,2,3,4.5]  (Definition and storage of an ACS (ID 1))
N20 #ACS DEF [3][0,15,5,2,3,4.5]   (Definition and storage of an ACS (ID 3))
N30 #ACS DEF [P1+3][2*P1,1,2,0,30,30] (Definition and storage of ACS (ID 5))
N30 #ACS ON                        (Activation of the ACS with the lastly)
                                   (programmed ID 5)
.
N50 #ACS OFF
N60 #ACS ON [3]                    (Activation of the ACS with ID 3)
.
N80 #ACS OFF
N90 #ACS DEF [3][0,1.2,1.3,0,0,3]  (Redefinition of the ACS with ID 3)
.
M30
```

プログラミング例3

例えばACS ON [...]を使用して(*ACS_ID*なしで)複数の座標系を次々に選択した場合、それらが、連鎖する新しい全体のACSを形成します。これは、対応する#ACS OFFによって段階的に選択解除する必要があります。

ACS IDを使用した/使用しない組み合わせACS選択ができますが、NCプログラムの明瞭さの理由から推奨されません。

ACSの複数プログラミングの例(*ACS_ID*を使用しない):

```
N010 #ACS ON [0,0,0,0,0,20] (Definition and activation of a ACS under)
                            (the automatically defined ID 1)
                            (No shifts, only rotation 20°around Z)
:
:
N050 #ACS ON [0,0,0,0,0,30] (Definition and activation of a ACS under)
                            (the automatically defined ID 2)
                            (No shifts, only rotation 30° around Z)

->(The result is a ACS with a total rotation of 50° around Z)
:
N100 #ACS OFF    (Deselection of ACS with ID 2, after that the ACS)
                 (with ID 2 is deleted!)

->(ACS with ID 1 with a rotation of 20° around Z remains active)

:
:
N200 #ACS OFF    (Deselection of ACS with ID 1, after that the ACS)
                 (with ID 1 is deleted and all ACS are deselected!)
:
:
N400 M30
```

17.6.3 座標系の結合

ACSとCSを組み合わせることによって、新しい座標変換を構築することができます。

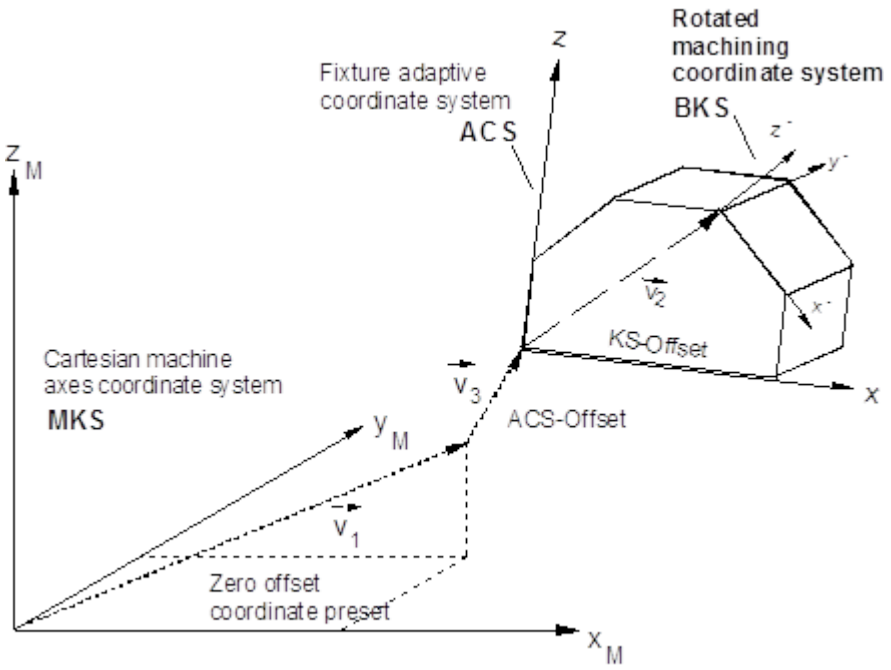


図 113: 図16-7: ACSとCSを組み合わせると、ワークが傾いてクランプされる傾斜面での加工が可能になります。

複数のACSとCSが、それぞれの有効化の順序で個別に結合されます。結果的にACSがCSと結合されて、全体的な変換になります。この結合はプログラミングとは無関係に行われ、常にACSから始まります(結合できるACSとCSの最大数はアプリケーションに固有です[6]-5.7)。

単一のACSの無効化は、有効化とは逆に行われます。同じことがCSに当てはまります。これを簡素化するには、IDパラメータを使用せずに#(A)CS OFFをプログラムします(図16-5と図16-6を参照)。

プログラミング例

```
%Linkage
.
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
```

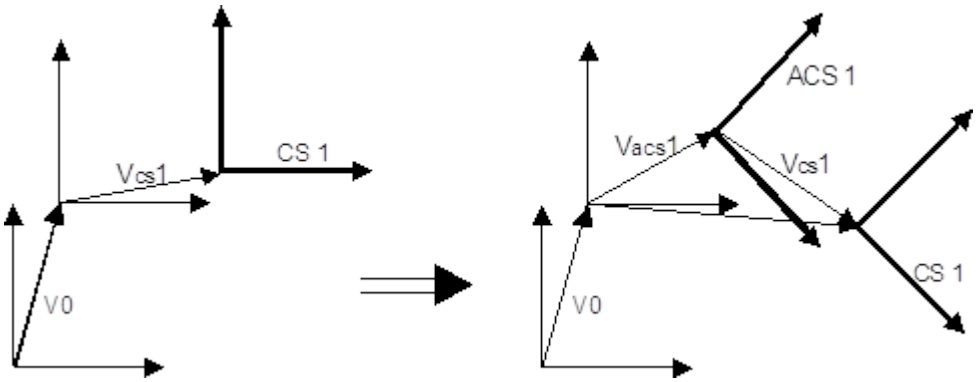


図 114: 図16-8: 既に有効であるCSの無効化を伴わないACSの有効化または変更

ACSまたはCSの相対的結合により、有効化の順序に応じて異なる結果が生じることがあります(図16-9を参照)。

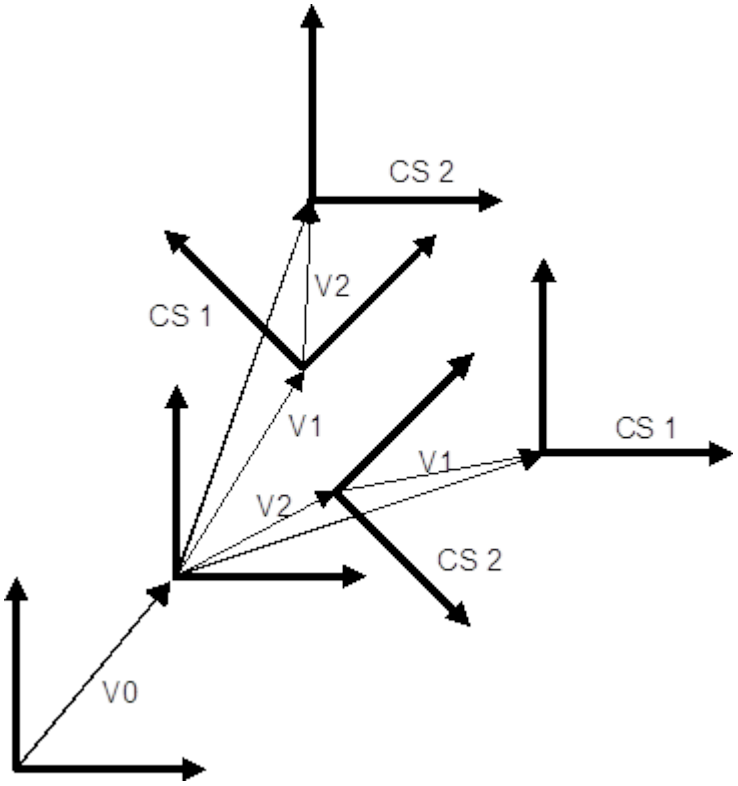
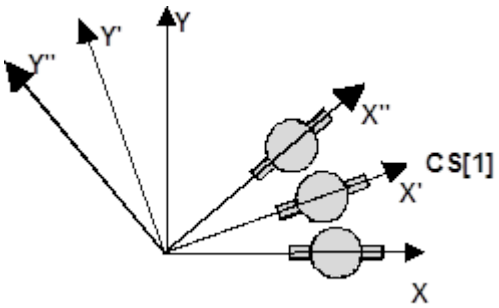


図 115: 図16-9: 有効化の順序に応じた異なるCS結合の結果(CS[1] - CS[2]またはCS[2] - CS[1])。

同じIDのCS (またはACS)も複数で有効になり、それ自体と結合することができます。

プログラミング例

```
N10 #CS DEF[1][0,0,0,0,0,20]
N20 LL TEILEPRG             (contour in 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
```



以下のNCコマンドによって、現在有効な変換全体を保存することができます。

#CS DEF ACT [<*CS_ID*>]
#ACS DEF ACT [<*ACS_ID*>]

(A)CS OFFによる(A)CSの順次無効化とは対照的に、以下のNCコマンドを使用して、CSまたはACSの結合から降順に部分的変換を直接無効にすることができます。

#CS OFF ALL	すべてのCSの選択解除
#ACS OFF ALL	すべてのACSの選択解除

プログラミング例

```
N10 #CS ON[3]
N20 #CS ON[4]
N30 #CS DEF ACT[5]      (Storage of CS[3] o CS[4] under CS[5])
N31 #CS OFF ALL         (Deactivation of all CS)
N32 #ACS ON[3]
N33 #ACS ON[4]
N34 #ACS DEF ACT[5]     (Storage of ACS[3] o ACS[4] under CS[5])
N35 #ACS OFF ALL        (Deactivation of all ACS)
N36 X0 Y0 Z0

N360 #CS ON [5]
N370 #ACS ON[5]
N380 #CS DEF ACT[1]     (Storage of ACS[5] o CS[5] under CS[1])
N390 #ACS OFF ALL
N400 #CS OFF ALL
N500 #CS ON             (Activation of CS[1])
```

```
N510 #CS OFF

M30
```

17.6.4 エフェクタ座標系(ECS)

キネマティックトランスフォーメーションと組み合わせて、この機能を使用するには、拡張パッケージ「Transformations」のライセンスが必要です。

エフェクタCSの主な用途は、工具が任意に配向された状態での加工中に、工具破壊、NC-Reset、またはNCプログラムの停止が生じた場合の取り消しを行うことです。ECSの特定はコマンドTOOL ORI CS (「工具の向き(TOOL ORI CS) [▶ 512]」の章)の逆です。つまり、工具を加工平面に対して位置合わせするのではなく、現在の工具軸に直交する加工平面が特定されます。

#ECS ON	ECSの選択	(モーダル)
#ECS OFF	ECSの選択解除	(モーダル)

ECSを有効にするときは、他のCSが有効であってはなりません。

配向軸の位置から、ECSが特定されます(そのz軸が現在の工具軸に平行であるように)。これにより、X軸とY軸の位置は定義されない(任意である)ため、内部で事前に定義する必要があります。ECSの原点は通常、工具先端または工具軸の外側に位置します。つまり、無衝突分離(collision free withdrawal)は、エフェクタZ軸方向に沿ったインクリメンタル移動によってのみ保証されます。

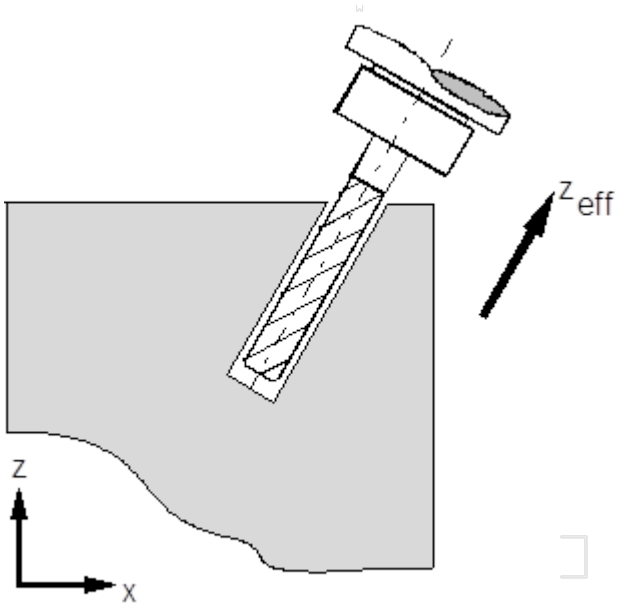


図 116: 図16-10: 傾斜ボーリングにおける加工

プログラミング例

```
N05 #CS ON[1.5,0,32,14.5,0,45]  (Activation of a CS)
N10 #TOOL ORI CS
N15 X0 Y0 Z0
N20 LL TEILEPRG          (Subroutine call for machining)
...
                         (Tool fracture, NC-reset)
                         (Withdrawal strategy)
N05 #ECS ON              (Determination of the cartesian transformation)
                         (according to the positions of the orientation)
                         (axes)
N10 G91 G01 F200
N20 Z62           (Withdrawal movement along the tool- or ECS-Z-axis res.)
...
N400 M30
```

17.6.5 機械軸座標系(MCS)への一時的遷移

キネマティックトランスフォーメーションと組み合わせて、この機能を使用するには、拡張パッケージ「Transformations」のライセンスが必要です。

この機能により、MCS、有効なキネマティクスおよび/または直交変換と、軸に含まれるすべてのオフセットを一時的に無効にすることができるため、機械軸を直接位置決めすることができます。MCSから離れた後に、有効化の前の状態が復元されます。

例えば、工具交換には、機械原点に対して十分に知られている座標を通常用いた、確定された工具交換位置のアプローチが必要です。CSでは、このような機械軸位置のアプローチは複雑になることがあります。その理由は、NCプログラムでCS軸が位置決めされるためです。

以下の命令で、機械軸座標系への一時的な遷移が有効/無効になります。

#MCS ON [EX TOOL]	MCSへの一時的な遷移の有効化	(モーダル)
#MCS OFF	MCSへの一時的遷移の無効化	(モーダル)

EX TOOL　　MCSでの工具交換時に工具オフセットが含まれません。これは#MCS OFFによって行われます。

MCSでは、NC機能の使用に関する制限がありませんが、RTCP/TLC、CS、ACS、およびECSの有効化はできません。

また、MCSでのプログラムされたオフセットはその無効化まで有効なだけで、保存されることはありません。

MCSでの外部軸交換(例えば、#CALL AX..)による軸設定の変更中に次のことに注意する必要があります。それは、キネマティクスおよび/または直交変換を再び有効にするには、確定した軸設定が絶対に必要であるということです。

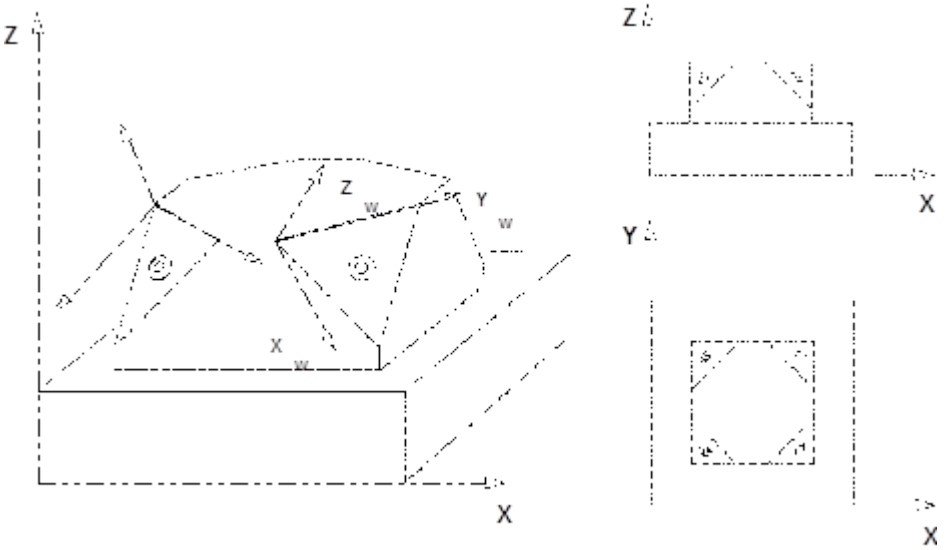
プログラミング例

```
N10 #RTCP ON
N20 #CS ON[1.5,0,32,14.5,0,45]    (Activation of a CS)
N30 G01 G90 F5000
N40 X0 Y0 Z0
N50 #MCS ON EX TOOL   (Transition to the machine-CS with the option)
                      (EX TOOL' (Tool will be included with MCS OFF)
N60 LL TOOLCHANGE     (Subroutine call for tool change)
...
N70 #MCS OFF          (Deactivation of the MCS, RTCP and CS will be)
                      (reactivated again)
N100 #RTCP OFF
N110 #CS OFF

N400     M30
```

プログラミング例

```
Work piece: Cube 100x100x50

UNTERPRG1 (parts program 1)     (Face milling of the current machining)
                                (plane)
N10 G01 G90 X0 Y-10 Z5 F250
N20 Z0 F2000                  (Start-up motion )
N30 G91
N40 $FOR P1=0, 5, 1           (P1: Number of infeed motions)
N50 $FOR P2=0, 7, 1
N60 Y90
N70 X3
N70 Y-90
N80 X3
N90 Y90
N100 X-42 Y-80
N110 $ENDFOR
N120 Z-2                      (Approach)
N130 $ENDFOR
N140 G90 Z50                  (Return motion )
N150 M29


%L UNTERPRG2 (parts program 2)     (Circle with centre boring)
N10 X0 Y0 Z10 F1000           (Linear and absolute approach motion)
N20 X0 Y0 Z5
N30 Z0
N40 LL UNTERPRG3              (Circular motion)
N50 Y35.34                    (Approach bore location)
N60 X15
N70 Z0
N80 LL UNTERPRG4              (Bore cycle)
N90 M29


%L UNTERPRG3 (parts program 3)     (Circular motion)
N10 X7.5 Y35.34               (Approach circle start point)
N20 G91 Z-4
N30 G90 G02 X7.5 Y35.34 I7.5 J0
N40 G01 Z20
N50 M29


%L UNTERPRG4  (parts program 4)     (Bore cycle)
N10 G91 Z-5                   (Relative bore motion)
```

```
N20 Z2
N30 Z-5
N40 Z2
N50 Z-5
N60 G90 Z20                    (Lift-off motion)
N70 M29

N55 #TOOL ORI CS               (Tool orientation)
N60 LL UNTERPRG1               (Contour machining)
N70 LL UNTERPRG2               (Contour machining)
N80 #CS OFF                    (De-selection machining coordinate system)
% Main program
N05 #KIN ID[1]                 (Machine kinematics 1)
N10 G74 Z1 X2 Y3 B4 C5         (Reference point return)
N20 T1 D1                      (Tool selection)
N40 #RTCP ON                   (RTCP selection)

N50 #CS ON[0,-50,51,0,30,-45]  (Selection of rot., shifted coord. syst.)
N52 G00 X0 Y0 Z5               (Positioning in new origin)
N55 #TOOL ORI CS               (Tool orientation)
N60 LL UNTERPRG1               (Contour machining)
N70 LL UNTERPRG2               (Contour machining)
N80 #CS OFF                    (De-selection machining coordinate system)

N90 #CS ON[-50,0,51,0,30,-135]  (Selection of rot., shifted coord. syst.)
N92 G00 X0 Y0 Z5               (Positioning in new origin)
N95 #TOOL ORI CS               (Tool orientation)
N100 LL UNTERPRG1              (Contour machining)
N110 LL UNTERPRG2              (Contour machining)
N120 #CS OFF                   (De-selection machining coordinate system)

N130 #CS ON[0,50,51,0,30,-225]  (Selection of rot., shifted coord. sys.)
N132 G00 X0 Y0 Z5              (Positioning in new origin)
N135 #TOOL ORI CS              (Tool orientation)
N140 LL UNTERPRG1              (Contour machining)
N150 LL UNTERPRG2              (Contour machining)
N160 #CS OFF                   (De-selection machining coordinate system)

N170 #CS ON[50,0,51,0,30,-315]      (Selection of rot., shifted coord. syst.)
N172 G00 X0 Y0 Z5              (Positioning in new origin)
N175 #TOOL ORI CS              (Tool orientation)
N180 LL UNTERPRG1              (Contour machining)
N190 LL UNTERPRG2              (Contour machining)
N200 #CS OFF                   (De-selection machining coordinate system)

N210 #RTCP OFF                 (RTCP de-selection )
N220 Z120                      (Return motion)
N230 #TOOL ORI CS              (Tool orientation)
N240 X0 Y0                     (Motion block because of tool orientation)
N250 X0 Y0 Z120 B0 C0          (Motion in machine co-ordinate system)
N260 M30                       (End of program)
```



17.7 座標変換のための補助ファンクション

この機能を使用するには、拡張パッケージ「Transformations」のライセンスが必要です。このライセンスは、標準ライセンスの範囲には含まれていません。

5軸加工機と直交構造(例えば、ヘキサポッド)を備える加工機では、動作を定義するための2つの代表的なバージョンを使用します。

第1のバージョンでは、円弧、直線、または多項式を用いてユーザが空間における輪郭をプログラムし、工具先端(TCP)がプログラムされた輪郭に沿って経路上を移動します。

第2のバージョンでは、ユーザが空間またはワーク座標(WCS)(機械座標(MCS)にマッピングされる)における目標点をプログラムします。軸における可能な最大速度に対応して、TCPが予測不可能な曲線(PTP)上を移動します。空間における曲線に沿ったTCP動作がないため、PTP動作は一般にTCPパス移動よりも速くなります。

NCプログラムでは、プログラムされたWCS目標点からMCS目標点への上記のマッピング(逆変換)を以下のNCコマンドによって実行することができます。計算されたMCS目標点への移動ブロックは、アブソリュート指令でユーザが明示的にプログラムする必要があります。

この方法は、例えば衝突領域外に各単一軸を順次移動するために使用することができます。

注記

このコマンドは、無効な変換および無効なユーザ定義座標系とともに使用することだけができます。

ワーク座標(WCS)からの機械座標(MCS)の計算:

#WCS TO MCS [[CS<*ID_expr*>] [KIN]]	(ノンモーダル)

CS<*ID_expr*> 特定の有効なIDを用いた直交変換を考慮した目標点の計算。

KIN 現在有効なキネマティックトランスフォーメーションを考慮した目標点の計算。

WCS目標点の宣言と計算されたMCS目標点の保存のために、以下の軸特有の変数を利用できます。これらの変数により、書き込みアクセスと読み取りアクセスができます。

V.A. WCS.* 軸特有の補助変数「work piece coordinate」(WCS)に対するアクセス。**有効な**変換中、この値は、通常プログラムされたワーク座標に対応しません。

V.A. MCS.*	軸特有の補助変数「machine coordinate」(MCS)に対するアクセス。**無効な**変換中、この値はプログラムされた機械座標に対応しません。

プログラミング例

一部の特定WCS点からNCが、直交/キネマティック逆変換によってMCS目標点を計算します。前述の特別な変数をNCプログラムで使用して、ユーザがこの計算されたMCS位置でアクセスできます。プログラムされたキーワードおよび関連するCS-IDに応じて、ユーザが実行される変換を定義します。

```
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[CS 1, KIN]   (Transformation of WCS into MCS)
N50 G00 Z=V.A.MCS.Z   (Moving of single axes on MCS-target points..)
N60 G00 X=V.A.MCS.X
N70 G00 Y=V.A.MCS.Y
N…
:
```

間違った使い方:

```
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 [CS 1] <- not allowed because a CS is active


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] <- not allowed because a trafo is active
```

機械座標(MCS)からのワーク座標(WCS)の計算:

WCS目標点へのMCS目標点の逆マッピング(順変換)は、以下のコマンドを使用して実行します。この方法は、例えば計測された値をWCSにマッピングするために使用することができます。

#MCS TO WCS [[CS<*ID_expr*>] [KIN]]	(ノンモーダル)

CS<*ID_expr*>	特定の有効なIDを用いた直交変換を考慮した目標点の計算。
KIN	現在有効なキネマティックトランスフォーメーションを考慮した目標点の計算。

17.8 ワーク座標系内の移動制限の計算のための補助ファンクション

現在のワーク座標系(WCS)における移動制限は、以下のコマンドを使用して成分VC1/VC2/VC3を含むプログラムされた移動ベクトルの方向で計算します。これらのベクトル成分は、標準化された形式でプログラムされていません。軸特有のソフトウェアリミットスイッチが移動制限の基礎です。この値に基づいて、制御システムが現在の座標系の移動制限を計算します。

#GET WCS POSLIMIT [VC1<*expr*> VC2<*expr*> VC3<*expr*>]

VC1<*expr*>,
VC2<*expr*>,
VC3<*expr*>

方向ベクトルの成分、REAL

以下のグローバル変数を使用して、座標系の最初の3つの軸に関する計算結果を読み出すことができます。

V.G.WCS_POSLIMIT_1	WCSの第1の主軸における移動制限
V.G.WCS_POSLIMIT_2	WCSの第2の主軸における移動制限
V.G.WCS_POSLIMIT_3	WCSの第3の主軸における移動制限

注記

プログラムされたモーション方向は、実際に実行されるモーション中に絶対に維持する必要があります。これを行わないと、計算されるモーション制限が誤ったものになります!

キネマティックトランスフォーメーション(#RTCP)が有効な場合、回転軸がプログラムされない可能性があります!

プログラミング例

```
N05...
N10 G98 X-100 Y-100 Z-100     (Shifting of negative software limits)
N20 G99 X200  Y200  Z300      (Shifting of positive software limits)
N30 #ECS ON                   (Selection of effector coordinate system)
N40 #GET WCS POSLIMIT [VC1=0 VC2=0 VC3=1]  (Calculation of WCS limit)
N50 G01 G90 Z=V.G.WCS_POSLIMIT_3 F2000 (Move to WCS limit in Z)
N60 #ECS OFF                  (Deselection of effector coordinate system)
N70...
```

コマンドの正しい使い方の例

```
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...
```

間違った使い方をすると、移動方向が設定に対応しないようになります。

```
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...
```

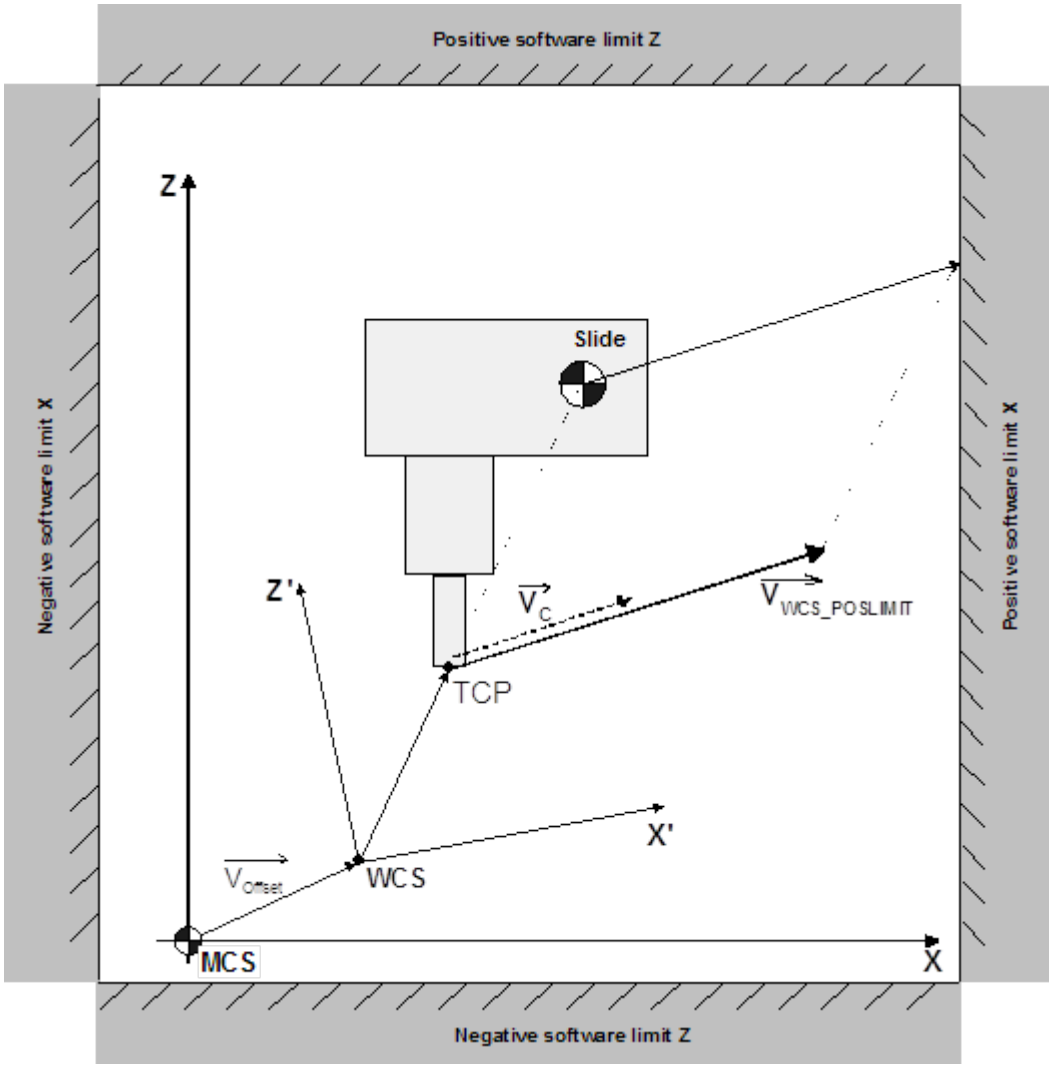


図 117: 図16-11: 現在のWCSのZX平面における移動制限の例

18 モジュロ軸の変更されたプログラミング

このプログラミングマニュアルで、拡張されたモジュロプログラミングが説明されています。この追加構文は、回転方向のプログラミング、アブソリュート指令における最大1つの回転への制限、および実際のモジュロ調整をチェックするための新しい変数をサポートします(0章)。

<axis_name> [[+] | -] *<expr>*

<axis_name> モジュロ軸の名称**。

<expr> 軸名の後の最初の記号は**常に**回転方向を示します。

-は回転が時計回り(cc)であることを意味します。
+は回転が反時計回り(ccw)であることを意味します。
記号なしは回転が最短距離の方向であることを意味します。

軸位置の単位は[度]または[mm]です。

**

注記

長い軸名称はサポートされません(例えば、「C_MODULO」)。

アブソリュート指令におけるプログラミング(G90):

- 軸に割り当てられた値(目標位置)がモジュロレンジにシフトされます。そのため、最大1つの回転を移動することができます。
- この値は[3*2+5]、P1、[P1+P2-3]、[-30]のような数式で表すことができます。
- 軸名の後の最初の記号は常に回転方向を示します。その他の記号は、すべて(絶対)位置定義の一部として評価されます。
- 例(仮定: 360度モジュロ):

```
G90 G1 C+560 ⇔ G90 G1 C+200  (Movement to position 200 in + direction)

G90 G1 C-P1     (Go to position P1 (with implicit modulo) in - direction)
```

- プログラミングされた位置が現在の位置である場合、動作はありません。
- モジュロ軸の移動はソフトウェアリミットでは停止しません。

インクリメンタル指令におけるプログラミング(G91)

- 軸に割り当てられた値は、以前の位置に対する軸の回転の大きさを示します。軸名の後の最初の記号は常に回転方向を示します。複数の回転の動作ができます。インクリメンタルプログラミングでは、その他の記号は許可されません。

- 例(仮定: 360度モジュロ):
- G91 G1 C+560 (「現在位置に560°」+方向で移動)
- この値がモジュロ値を上回る場合、回転の数が考慮されます。複数の回転の動作ができます。

18.1 変数

モジュロ情報を読み取るためのNCプログラム変数:

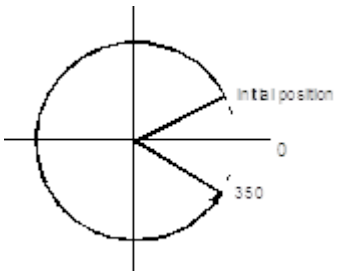
V.A.MODE[i] 軸テーブルに従って軸モードを提供します。
例えば、軸がモジュロタイプである場合は4です。
->は、モジュロとして宣言されている軸を読み取るために使用します。

V.A.MODULO_VALUE[i] モジュロレンジの値を読み取るために使用します。
例えば、モジュロレンジが0～360°の場合は360です。
(軸がモジュロ軸でない場合、この値は意味がありません)。

18.2 絶対座標におけるモジュロプログラミング

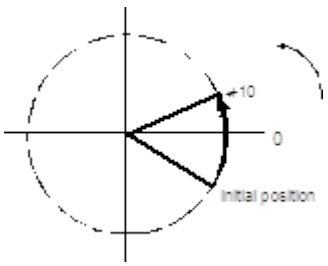
プログラミング例

G90 G1 C+350 ⇔ 位置に350° +方向で移動



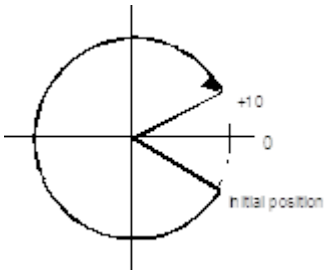
P1 =+10

G90 G1 C+P1 ⇔ G1 C+10 ⇔ 位置に10° +方向で移動

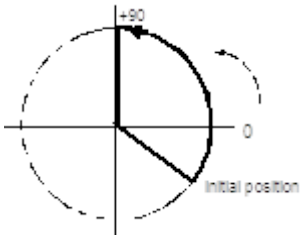


P1= -350

G90 G1 C-P1 ⇔ G1 C-(-350) ⇔ G1 C-10 ⇔ 位置に10° -方向で移動



G90 G1 C+450 ⇔ G1 C+(450 mod 360) ⇔ 位置に90° +方向で移動



正しいプログラミングの例:

```
C+200           Rotate in positive direction to position 200
C-200           Rotate in negative direction to position 200
C+-200          Rotate in positive direction to position -200 (= +160)
C+[-200]        Rotate in positive direction to position -200 (= +160)
C--200          Rotate in negative direction to position -200 (= +160)
C-[-200]        Rotate in negative direction to position -200 (= +160)
C200            Rotate in shortest way to position 200
C[+200]         Rotate in shortest way to position 200
C[-200]         Rotate in shortest way to position -200
```

間違ったプログラミングの例:

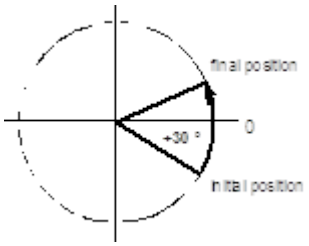
ありません。その理由は、軸名の後の最初の記号が回転方向を決定し、追加の記号はすべて位置式に属するからです。

注記

2つの記号(方向と位置)のプログラミングが許可されるのは、軸のモードが「モジュロ」である場合に限られます(P-AXIS-00015)。軸名の直後に記号がプログラムされていない場合、最短距離での位置決めが行われます。

18.3 相対座標におけるモジュロプログラミング

G91 G1 C+30



G91 G1 C-30

正しいプログラミングの例:

```
C+200    Rotate in positive direction to current position plus 200

C-200    Rotate in negative direction to current position minus 200.

C200     Rotate in positive direction to current position plus 200
```

間違ったプログラミングの例:

```
C+-200   Error: Negative motion path during rel. programming not allowed.

C--200   Error: Negative motion path during rel. programming not allowed.

P1=-1
C-[P1]   Error: Negative motion path during rel. programming not allowed.
```

19 拡張工具プログラミング

19.1 ファンクションの説明

19.1.1 工具ID

拡張工具プログラミングの範囲内で、CNCは新しい工具関連通信オブジェクトを工具管理領域(WZM)に提供します。これらは強化された工具記述(WZ)および工具サービス寿命変数です。

現在の標準プログラミングでは、NCプログラムで工具を識別するのに単一要素番号を使用します。DIN 66025に準拠して、この数値IDは、新しいデータ(計算工具交換)を含めるために使用されるDワードとともにプログラムします。T言語コマンドと組み合わせて、このIDは、物理的に変更しなければならない次の工具を定義します。

新しいデータを含めるために、このデータを現在の外部工具管理ファンクションから要求する必要があります。この工具管理ファンクションは、特定の製造メーカ関連のアルゴリズム(工具識別に基づいて変更する必要がある工具を決めるために使用される)を処理します。次の2つのことを考慮する必要があります。1) 伝送された工具IDは工具タイプを定義するだけです。2) 工具マガジンに、すぐに使用できる同じタイプの複数の工具(代替工具)を格納することができます。そのため、特定の工具をNCプログラムで明確に識別することはできません。

最初の例では、T番号が技術項目として使用されています。これは、工具がNCチャンネル経由でPLCに達することを意味します。DIN 66025に準拠して、動作中のスピンドルに新しい工具を物理的に挿入するためにM06を使用します。Tコマンドの後、M06によって動作中のスピンドルに工具が実際に挿入される前に、準備手段(例えば、工具マガジンにおいて)を講じるために、2つのコマンド「T with tool number」とM06を個別に指定することができます。

拡張された工具管理ファンクションでは、工具データを3桁の数字によって定義します。以下、この数字を「工具ID」と呼びます。

工具ID = 基本工具番号 + 代替工具番号 + 変更番号

基本工具番号は工具タイプを表し、代替工具番号はこのタイプの工具ユニットを表します。変更番号には、純粋にデータに関連する意義があります。1つの工具にさまざまなデータ記録を使用することができます。

19.1.2 工具寿命監視

工具寿命値を取得するには、工具の接触時間(サービス寿命)と接触中にカバーされる距離(サービス距離)を計算する必要があります。特定された値が工具管理ファンクションに送られます。

ここでは、移動ブロックの動作のみを考慮する必要があります。早送り位置決めが寿命カウント変数に影響を及ぼしてはなりません。

工具が交換された後、インターポレータにデータ(工具ID、接触時間、接触距離)が表示されます。

サービス寿命はmsで記録され、サービス距離はmmで記録されます。

NCプログラムでは、工具寿命変数の重み付けが可能であることが必要です。導入する必要がある係数が、工具寿命変数の取得を工具負荷率に適合させるために使用されます。

例:

常に接触している工具は、100%の係数で重み付けする必要があります。ただし、移動距離のわずか半分で材料が取り外される場合、0.5の加重係数を計算で使用できます。

サービス時間とサービス距離の加重係数の初期値は1.0です。

19.2 コマンドと変数のプログラミング

工具IDはプレーンテキストコマンドでプログラムします。#TOOL DATAによる新しい工具データの要求はDコマンドに対応します。#TOOL PREPによる物理的工具交換のための準備技術コマンドはTコマンドに対応します。

#TOOL DATA [*<basic>* [, *<sister>* [, *<variant>*]]]	(新しい工具データの要求)
#TOOL PREP [*<basic>* [, *<sister>* [, *<variant>*]]]	(工具交換の通知)

basic、**sister**、および**variant**の各パラメータの数を設定できます[6]-9.18。適切な値は1～3です。2つのパラメータが期待される場合、それらは基本工具番号と代替工具番号になります。

基本工具番号(**basic**)の指定は必須です。パラメータの数が3の場合、代替工具番号に関する**sister**と変更に関する**variant**をオプションでプログラムすることができます。**sister**または**variant**がプログラムされていない(コンマの後にコンマが続くか、または右角括弧の後に右角括弧が続く)場合、ゼロが挿入されます。

プログラミング例

```
#TOOL DATA [ P10, "SISTER", 0 ]     ⇔     #TOOL DATA [ P10, "SISTER", ]
and
#TOOL PREP [P10, 0, "VARIANT"]      ⇔     #TOOL PREP [P10, , "VARIANT"]
```

DコマンドとTコマンドの指定における演算式は、基本工具番号として解釈する必要があります。そのため、工具管理ファンクションで、工具データ記録の選択のために以前と同じ自由度が維持されます。

T *<basic>*	または	D<basic>

デコーダ変数は、工具識別の要素へのアクセスを実装するために使用します。現在のソフトウェアバージョンでは、既にV.G.T_AKTによって外部工具管理ファンクションから、最後に要求された工具の番号を問い合わせることが可能です。新たに導入された構文と並行して、このパラメータは常に基本工具番号を表します。

V.TOOL.BASIC	(最後にプログラムされた基本工具番号への読み取りアクセス)
V.TOOL.SISTER	(最後にプログラムされた代替工具番号への読み取りアクセス)
V.TOOL.VARIANT	(最後にプログラムされた変更番号への読み取りアクセス)

プログラミング例

```
#TOOL DATA [ P10, "SISTER", 3 ]
.....
#TOOL PREP [V.TOOL.BASIC,V.TOOL.SISTER, V.TOOL.VARIANT]
```

19.2.1 サービス寿命とサービス距離の加重係数

Tコマンドによって通知されるすべての工具交換において、完全な工具ID、サービス寿命、およびサービス距離が工具管理ファンクションに送られます。次に、すべての数量がゼロに設定され、新たに置き換えられた工具のサービス変数検出が有効になります。

サービス寿命とサービス距離の加重係数をプログラムするために、2つの解読処理変数が導入されています(アクセスはリアルタイムと同期しません)。

V.TLM.TIME_FACT	(サービス寿命の重み付け)
V.TLM.DIST_FACT	(サービス距離の重み付け)

これらの変数の読み取り/書き込みができます。プログラム開始時は両方の係数が100%です。両方の変数をNCブロックで書き込むことができます。

取得条件:

- 早送りブロックは工具寿命監視に含まれていません。
- 工具寿命監視が、ゼロの送り速度で停止します。
- 早送り補間を除いて、すべての移動タイプが取得に含まれています。例えば、G01、G02、G03、スプライン軸補間、およびG63が含まれています。
- 加重係数が計算に含まれています。

- 動作に関与する軸について、主軸と従軸は区別されません。距離を合計するために、経路送り速度が常に使用されます。ブロックで独自にプログラムされる従軸の場合、従軸によってカバーされる距離がサービス距離に加算されます。これが望ましくない場合、加重係数V.TLM.TIME_FACT/DIST_FACT = 0..を指定することによって、プログラマがこれを訂正することができます。
- 有効なマスタ/スレーブ配列は考慮されません。
- リセットまたはプログラム中止の場合、最後の現在値も工具管理ファンクションのデータベースに保存されます。
- 置き換えのみが行われる場合、つまり工具がワークスピンドルにまだセットされずに現在のT番号がゼロである場合、データは送られません。
- 工具データが送られるのは、工具管理ファンクションP-CHAN-00016が実際に存在する場合に限られます。

19.2.2 工具寿命データの読み取りと削除

次の工具交換の前に工具寿命データにアクセスするために、特別なコマンドを使用できます。これらのコマンドは、例えば同期操作中に従軸の工具の寿命データを特定するために必要です。その理由は、「サービス寿命とサービス距離の加重係数 [▶ 540]」の章で述べたように、これらのデータが記録されないためです。以下のコマンドを使用して、現在のチャンネルにおける有効な工具の寿命データが読み取られ、プログラムされた工具(例えば、従軸の工具)について工具管理ファンクション領域(WZM)に送られます。

#TOOL LIFE READ [*<basic>* [, *<sister>* [, *<variant>*]]]	(現在の工具寿命データの読み取りとツールへの割り当て) (任意の工具に)

工具IDのプログラミングでは、例えばコマンド#TOOL PREPの場合と同じ規則が有効です(「コマンドと変数のプログラミング [▶ 539]」の章)。HÜEMNOS協定に準拠して、工具IDに1つ、2つ、または3つの工具パラメータを含めることができます。1つのパラメータの指定が必須です。このパラメータは、常に基本工具番号(**basic**)として解釈されます。工具IDのパラメータは任意の数式で表すことができます(V.G.T_AKT、V.TOOL.BASIC、Pxx、V.L.xxxなど)。

現在の工具寿命データがWZMに送られた後、内部工具寿命記録はリセットされずに存続します。

以下のコマンドが、データをWZMに送ることなく、現在のチャンネルの有効な工具の内部工具寿命記録を削除します。

#TOOL LIFE REMOVE	(現在の工具寿命データの削除)

プログラミング例

```
....
:
....    Standard operation
:
#TOOL LIFE READ [V.G.T_AKT]      (Read tool life data of current tool)
#TOOL LIFE REMOVE               ( Reset tool life data for separate)
                                (recording of tool life data during syn-)
                                (chronous operation)
```

```
Selection of synchronous operation
:
....    Synchronous operation
:
De-selection of synchronous operation

#TOOL LIFE READ [T10]      (Hypothesis: T10 is active tool of a slave axis)
                           (during synchronous operation --> Send tool)
                           (life data of current "master axis"-tool to WZM )
                           (for T10)
:
....    Standard operation
:
```

19.2.3 工具データの更新

以下のコマンドは、有効な工具の工具データを例えば軸交換命令の後で、追加の新しいチャンネル軸に繰り返し含めるために便利です。

#TOOL REFRESH	(有効な工具のデータの更新)

このコマンドには、有効な工具の既に入手可能なデータが直ちに含まれます。工具データは、内部工具データリスト[5]から引き継がれることもなく、外部工具管理ファンクションから要求されることもありません。Dワードの反復プログラミングを取り消すことができます。

20 位置決め軸

位置決め軸は並進軸または回転軸です。位置決め軸は、同じNCチャンネルの経路軸成分とは関係なく補間することができます。位置決め軸にはそれぞれ独自の軸インターポレータがあり、独自の送り速度で移動することができます。モジュロモードの回転位置決め軸は、常に最短距離を移動します。

制限事項:

以下の場合、位置決め軸移動のプログラミングはできません。

- この軸が同期操作で現在有効である場合
- キネマティック/直交トランスフォーメーションが有効であり、一部の特定条件が有効でない場合(「直交/キネマティックトランスフォーメーションと位置決め軸 [▶ 550]」の章を参照)
- ブロック検索が有効である場合
- シミュレーションモード(オンラインシミュレーション、輪郭の可視化、生産時間の計算)が有効である場合
- スプライン軸補間が有効である場合
- 多項式輪郭加工が有効である場合
- 旋回ファンクションが有効である場合

20.1 独立軸

独立軸のプログラミングでは、2つの異なる操作モードが利用可能です。

- ブロック終了時の補間軸と独立軸のコマンド値ベースの同期。
- 複数のブロックにまたがる補間軸と独立軸のコマンド値ベースの同期。

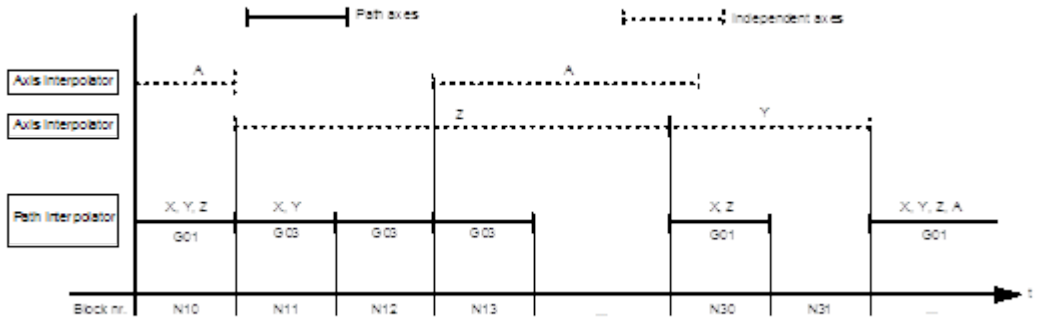


図 118: 図19-1: 補間軸/独立軸の動作図

独立軸の場合、オフセットは考慮されません。

独立軸を含む追加の手動モード(G201/G202)が可能です。

独立軸のプログラミング構文:

```
<axis_name> [ INDP_SYN | INDP_ASYN  G90 | G91  G00 | [G01 | G100 FEED<expr>] |
            TIME<expr> | FEED_MAX_WEIGHT<expr>  POS<expr>  [SLOPE_TYPE<ident>]
            {M<expr>}  {H<expr>} [DRY_RUN] { \ } ]
```

<axis_name>	独立軸の軸名称
INDP_SYN	同期(ブロック単位)独立軸動作の識別。すべての軸が終了位置に達している場合、次のブロックへの遷移のみが行われます。必ず**1番目に**プログラムされるキーワードであること。
INDP_ASYN	非同期(複数ブロック)独立軸動作の識別。終了位置には同期がありません。特別なNCコマンド(#WAIT INDP)によって、または従来型の補間軸のように独立軸のプログラミングによって、コマンド値ベースの同期が行われます。必ず**1番目に**プログラムされるキーワードであること。
G90/G91	絶対/相対値
G00/G01	早送り/直線補間
FEED*<expr>*	軸特有の送り(経路ユニット/min)
TIME*<expr>*	軸特有の動作時間(秒)
FEED_MAX_WEIGHT*<expr>*	軸特有の最大送りP-AXIS-00212に関連する加重係数(%)。100%未満の加重係数のみ許可されます(G194に従う、章4.26)。
POS*<expr>*	軸位置
SLOPE_TYPE*<ident>*	#SLOPE [TYPE=...]に従う傾斜のプロファイルタイプ(章操作モードの選択 [▶ 261])。傾斜タイプがプログラムされていない場合、初期設定としてチャンネルパラメータのタイプP-CHAN-00071が使用されます。
古い構文: *SLOPE_PROFIL<expr>*	*傾斜のプロファイルタイプ(0、1、2、3)。傾斜タイプがプログラムされていない場合、初期設定としてチャンネルパラメータのタイプ P-CHAN-00071 が使用されます。*
G100 **[V2.11.2801.05から]**	計測タイプ1、2、または7 (章計測ファンクション [▶ 59])と組み合わせて、独立軸を含む計測動作をプログラムすることもできます。計測位置のラッチが、関与する各軸について個別に実行されます。経路移動またはG100計測と並行して、独立した計測動作も可能です。詳細については、[FCT-C4//計測]を参照してください。
DRY_RUN	軸動作のドライラン。移動はNCチャンネルでのみ実行されますが、軸は実際には動きません。結果は、物理軸に対するチャンネル内の軸座標のオフセットです。このオフセットはプログラム開始時に自動的に削除されるか、明示的にプログラムされた#CHANNEL INIT [CMD_POS]によって削除されます(プログラミング例3を参照)。
M*<expr>*	軸特有のMファンクション
H*<expr>*	軸特有のHファンクション
\	複数行にわたるコマンドの明確なプログラミングのための区切り記号(「バックスラッシュ」)。

軸特有のM/Hファンクションは、動作をプログラムせずに独立軸に出力することもできます。識別INDP_SYNまたはINDP_ASYNのみを追加する必要があります。

```
<axis_name> [ INDP_SYN | INDP_ASYN  M<expr> {M<expr>}  H<expr> {H<expr>} { \ } ]
```

<axis_name>	独立軸の軸名称
INDP_SYN/INDP_ASYN	独立軸の識別
M*<expr>*	軸特有のMファンクション
H*<expr>*	軸特有のHファンクション
\	複数行にわたるコマンドの明確なプログラミングのための区切り記号(「バックスラッシュ」)。

特別な非同期軸動作(INDP_ASYN)のコマンド値ベースの同期をNCコマンドを使用して実行することができます。

```
#WAIT INDP [  <axis_name> { ,<axis_name> }  ]
```

<axis_name>　非同期軸の軸名称

対応する#WAIT INDP [ ]の前で(またはこれを使用しないで)非同期軸を再びプログラムする場合、コマンド値ベースの同期がインターポレータにおいて暗黙的に行われます。

NCコマンドを使用して、すべての存在する有効な非同期軸動作のコマンド値ベースの同期(INDP_ASYN)を実行することができます。

```
#WAIT INDP ALL
```

同じブロックにおいて、独立軸を同じ軸の軸特有のM/HファンクションP-CHAN-00039およびP-CHAN-00025とともにプログラムすると、エラーメッセージが出力されます。
例: M10はX軸特有のMファンクション(m_default_outp_ax_name[10] X):
N10　**M10**　**X** [INDP_SYN G01 G90 POS10 FEED1000 M7]
　　|_____| ← **Error!**

プログラミング例

例1:

```
:
N10 X10 Y11 Z[INDP_SYN POS50 G01 FEED100 G90]  (N10 is finished, if X,)
                                               (Y and the independent)
                                               (synchr. Z-axis have)
                                               (finished their movements)

N20 X20 Y22            (N20 is executed after all movements of N10 are)
                       (finished)

N30 X5 Y10 Z[INDP_ASYN POS500 G01 FEED200 G90]   (N30 is finished, if X)
                                                 (and Y have finished)
                                                 (their movements; the)
                                                 (independent asynchr.)
                                                 (Z-axis goes on with)
                                                 (its movement)

N40 X20 Y30            (N40 is interpolated; the asynchronous independent)
                       (Z-axis is still moving)

N50 #WAIT INDP[Z]      (Synchronization of Z-movement of N30)

N60 X30 Y40 Z60        (X, Y, Z are interpolated together in the path)
                       (compound, after Z-axis was synchronized in N50)

N70 Z[INDP_SYN M50 ]   (Output of M50 for the independent Z-axis)
N80 …
```

例2:

```
N10 X10 Y11 Z[INDP_ASYN POS500 G01 FEED200 G90]   (N10 is interpolated,)
                                                  (the independent asyn-)
                                                  (chronous Z-axis still)
                                                  (moves)
N20 X20 Y22      (N20 is interpolated, the independent asynchronous)
                 (Z-axis still moves)
N30 Z550         (Implicit synchronization of Z-movement of N10, before)
                 (movement Z550 is started)
N40 X20 Y30 Z60  (N40 is interpolated)
N50 …
```

例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

; Removing of DRY_RUN offset
N001 #TIME 2
N111 #CHANNEL INIT[CMDPOS]                 ; IPO=6,  LR=6, offset=0
N222 #TIME 2
N400 Y10 Z5
M30
```

20.2 振動軸

このファンクションを使用できるかどうかは、設定やバージョンの具体的な範囲によって異なります。

特定の加工技術(研削など)では、輪郭加工動作とは関係なく実行される振動軸動作が必要です。

後で「振動動作」と呼ばれるこの動作は、ワークに対して定期的に反転する工具によって実行されます。

研削中の振動軸の例を後で示します。ワークの加工は、振動X動作をY軸とZ軸での位置決め動作と重ね合わせることによって行われます。

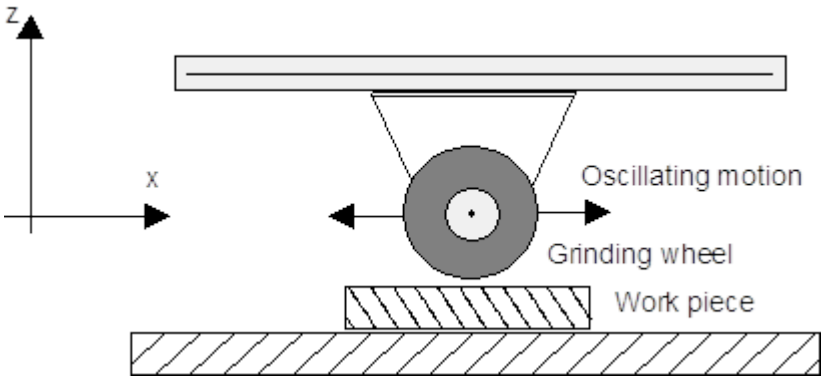


図 119: 図19-2: 振動軸を伴う研削

振動動作の重要な特性は、2つの絶対位置間の振動動作と送り速度によって生じます。

NCプログラムによって振動動作が開始/停止し、そのパラメータが定義されます。

設定された軸の範囲内での振動軸の定義ができます。振動動作は輪郭加工動作と同期しません。

振動動作は以下のいずれかの仕方で行われます。

- NCコマンドによって直接行われる
- プログラムされた振動軸の輪郭加工動作への遷移時に暗黙的に行われる
- 解読処理と補間の同期のための軸位置が要求されたときに暗黙的に行われる
- NCプログラムの最後に暗黙的に行われる

振動動作のチャンネルパラメータP-CHAN-00071で傾斜タイプ(直線/非直線傾斜)を設定することによって、動的位相における速度プロファイルの特性を定義することができます。

プログラミング構文は、独立軸の各軸に対応したプログラミングに基づいています。軸識別子の後で、振動動作のパラメータをキーワード(および該当する場合は関連値)によって定義します。

```
<axis_name> [ OSC  ON | OFF   FEED<expr> | FREQ<expr> | TIME<expr>
              [1ST_POS<expr>  2ND_POS<expr>] | [ ZERO_POS<expr>  EXCUR<expr>]
              [1ST_DELT<expr>  2ND_DELT<expr>]  [NBR_OSC<expr>] { \ } ]
```

<axis_name>	振動軸の名前
OSC	「振動」の識別。必ず**1番目に**プログラムされるキーワードであること。
ON	振動オン。振動動作が始まる前に、まず有効な輪郭加工動作がブロック終了時に停止します。
OFF	振動オフ。その後、振動軸を通常の輪郭加工動作で動かすことができます。以前の取り消しなしで新しいコマンドが実行されたときの振動動作の暗黙の無効化。
FEED*<expr>*	振動動作の送り速度(経路ユニット/min)
FREQ*<expr>*	振動動作の周波数(Hz)
TIME*<expr>*	振動動作の時間(秒)
1ST_POS*<expr>*	第1反転位置(mm)
2ND_POS*<expr>*	第2反転位置(mm)
ZERO_POS*<expr>*	振動動作の原点(mm)
EXCUR*<expr>*	偏位(mm)
1ST_DELT*<expr>*	第1反転位置での待機時間(秒)
2ND_DELT*<expr>*	第2反転位置での待機時間(秒)
NBR_OSC*<expr>*	振動の数
\	複数行にわたるコマンドの明確なプログラミングのための区切り記号(「バックスラッシュ」)。

振動動作の特性は反転位置と軸送り速度によって決まります。反転位置は直接指定することができますが、別の方法として、ゼロ位置と偏位によって自動的に決めることもできます。

- 振動位置は、常に**絶対位置**です。
- 振動動作の無効化の後、停止が常に**振動位置2**で起こります!

別の方法として、振動速度を送り速度、周波数、または時間によって決めることもできます。

動的特性により制限が生じない場合、直線傾斜が使用されるときには、軸の数量、周波数、および期間が正確に(そして非直線傾斜が使用されるときはおおよそ)維持されます。

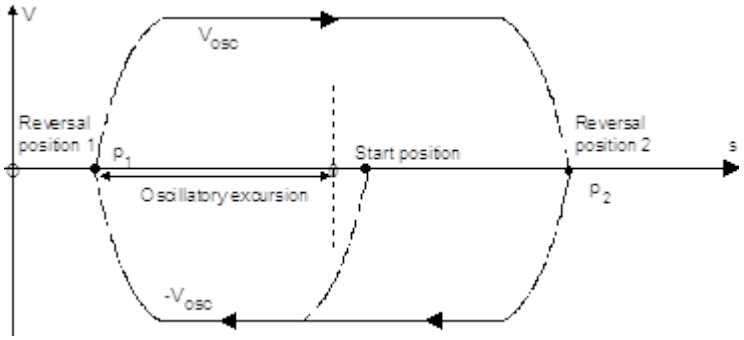


図 120: 図19-3: 振動動作中の位置決め

プログラミング例

```
Specifying the oscillation travel distance via reversal positions
N10 X[OSC  ON  1ST_POS=-100  2ND_POS=100  FEED=1000]

Specifying the oscillation travel distance via the zero position and the excursion, 10 oscillations
N20 X[OSC  ON  ZERO_POS=0  EXCUR=100  FEED=1000  NBR_OSC=10]

Specifying 1 Hertz oscillation frequency
N30 X[OSC  ON  ZERO_POS=0  EXCUR=100  FREQ=1]

Specifying a 4s period
N40 X[OSC  ON  ZERO_POS=0  EXCUR=100  TIME=4]

Oscillating motion with feed motion of a contouring axis
N50 X[OSC  ON 1ST_POS=111 2ND_POS=222 FREQ=1]
N60 G01 G90 Y500 F200

Oscillating with waiting times of 0.5 s each at the reversal positions
N70 X[OSC ON 1ST_POS-100 1ST_DELT0.5 2ND_POS200 2ND_DELT0.5 FEED1000]

Cancelling oscillation
Oscillation is stoped with reaching of reversal position 2:
N80 X[OSC  OFF]

Fast stop of oscillation
If an OFF is programmed in combination with FEED, the oscillation motion is stopped at once
(feedhold for oscillation axis) and the reversal position 2 is directly moved with the new feed.
N90 X[OSC  OFF  FEED=5000]
```

キーワードと値の間の等号の指定はオプションです。

20.3 直交/キネマティックトランスフォーメーションと位置決め軸

20.3.1 位置決めとシフト

有効な独立軸または振動軸を含む、従来型の操作とCSモード(#CS、#ACS)では、この軸の絶対位置をプログラムする必要があります。これは、工具交換が実行されている場合、この軸のプログラミング中に工具長を考慮する必要があることが意味します。ゼロオフセット(G54..G59)または以前にプログラム座標プリセット(G92)は有効ではありません。

有効なキネマティックトランスフォーメーション(#TRAFO)中に、工具オフセットがトランスフォーメーションで直接考慮されます。これは、有効な独立軸または振動軸についても工具オフセットが考慮されることを意味します。この場合も、ゼロオフセット(G54..G59)または以前にプログラムされた座標プリセット(G92)は有効ではありません。

20.3.2 制限事項

直交/キネマティックトランスフォーメーションを有効にする前に、振動動作または独立動作を選択解除する必要があります。

位置決め軸をプログラムできるのは、以下に対してのみです。

- 直交キネマティクス
- XY機械台上の垂直な工具の第3の主軸(通常はZ軸) (例えば、CAヘッドキネマティックにおける0°のA軸)

プログラミング例

独立軸のプログラミング:

```
N10 #KIN ID[9]
N20 #TRAFO ON
N30  Z[INDP_ASYN G01 G90 POS20 F0.01 SLOPE_PROFIL 0]
N40  Z[INDP_ASYN G01 G90 POS-20 F0.01 SLOPE_PROFIL 0]
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_PROFIL 0]
N40  Z[INDP_ASYN G01 G90 POS-20 F0.01 SLOPE_PROFIL 0]
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_PROFIL 0]
N50  Z[INDP_ASYN G01 G90 POS-20 F0.01 SLOPE_PROFIL 0]
N60  G01 G90 X100 F0.1
```

```
N70 #CS OFF
N80 #TRAFO OFF
N100 M30
```

プログラミング例

振動軸のプログラミング:

```
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
```

プログラミング例

以下のパートプログラムは、CS、キネマティックトランスフォーメーション、および振動の無効なネスティングを示しています。

```
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 各軸に対応したプログラミング

プログラミング構文は、位置決め軸の各軸に対応したプログラミングに基づいています。軸識別子の後で、パラメータをキーワード(および該当する場合は関連値)によって定義します。

21.1 NCプログラムでの軸補正の選択/選択解除

TwinCat CNC: V2.10.1501.00以降

その他: V263以降

複数の軸補正[FCT-C5]をNCプログラムでコマンドによって直接選択/選択解除することもできます。複数の軸に対する特に1つのNCブロックの軸の場合、異なる軸補正を同時に有効/無効にすることができます。

軸補正(COMPコマンドによってオフにされる)は、NCプログラムの終了後も無効なままです。つまり、プログラム終了時、補正が自動的に再び有効になることは**ありません**。以下のNCプログラムでは、COMPコマンドによって補正を明示的に再びオンにする必要があります。

```
<Achsname> [ COMP [ [ ON | OFF [ CROSS PLANE LEAD TEMP FRICT ] ] | OFF_ALL ]
           [ NO_MOVE ] { \ } ]
```

<axis_name>	軸の名前
COMP	軸特有の補正の選択/選択解除の識別。必ず**1番目に**プログラムされるキーワードであること。
ON	プログラムされた補正をオンにする
OFF	プログラムされた補正をオフにする
CROSS	クロス補正のキーワード
PLANE	平面補正のキーワード
LEAD	スピンドルリードスクリュの誤差補正のキーワード
TEMP	温度補正のキーワード
FRICT	摩擦補正のキーワード**[V2.11.2022.05から]**
OFF_ALL	すべての有効な補正をオフにします。キーワードの後で、その他の補正キーワードをプログラムすることはできません。
NO_MOVE	軸補正をオン/オフにすると、位置オフセットが行われます。通常は、NCプログラムの処理が継続される前に、位置オフセットが破棄されます。キーワードNO_MOVEによって、この動作が停止します。チャンネルは変更された位置で初期化されます。次に、位置オフセットがNCプログラムの最初の軸動作によって破棄されます。
\	複数行にわたるコマンドの明確なプログラミングのための区切り記号(「バックスラッシュ」)。

プログラミング例

```
N10 X[COMP OFF CROSS PLANE]   Switch off cross-and plane compensation in X axis

N50 X[COMP OFF CROSS] Y[COMP ON LEAD TEMP]    Compensation programming of multiple axes in one
common NC block

N100 Z[COMP OFF_ALL]   Switch off all compensations in Z axis

N200 Y[COMP OFF_ALL NO_MOVE] Switch off all compensations on the Y axis
       without axis movement.
```

21.2 間隔制御(間隔タッチプローブを備えるスピンドル)

このファンクションを使用できるかどうかは、設定やバージョンの具体的な範囲によって異なります。

必要なハードウェアが工具支持軸(スピンドル軸)に装着されている場合、このファンクションによって、平らでない平面までの工具の間隔を事前に定義・制御することができます。この間隔は数値単位系によって把握され、NCによって平らでない平面まで追従されます。

タッチプローブを備えるスピンドルの間隔制御は、パラメータP-AXIS-00328で有効になります。有効化は以下のNCコマンドで行われます。

詳細については、機能説明「間隔制御」[FCT-M3]を参照してください。

```
<Achsname> [ DIST_CTRL  [ON [ CONST_DIST ] | OFF | CHECK_POS | FREEZE | REF]
                SET_POS<expr>  SET_DIST<expr>  [ NO_MOVE ]  VAL1<expr> - VAL5<expr>  { \ } ]
```

<axis_name>	工具を搬送する軸の名前。
DIST_CTRL	「間隔制御対象スピンドル」の識別。必ず**1番目に**プログラムされるキーワードであること。
ON	事前に定義されたワーク表面による間隔制御オン。有効化の際に設定位置(SET_POS)を設定する必要があります。
ON CONST_DIST **[V2.11.2804.03から]**	事前に定義されたワーク表面までの間隔による間隔制御オン。有効化の際に間隔(SET_DIST)を設定する必要があります。
OFF	間隔制御オフ。
CHECK_POS	位置が許容差範囲にあるかどうかをチェックします。
FREEZE	ワーク上の制御間隔をフリーズします。軸位置または出力補正値が維持されます。つまり、ワーク表面に応じた軸の再調整が中断されます。
REF	数値単位系(センサ)参照(絶対単位系が存在しない場合)
SET_POS*<expr>*	ワーク表面の指定[mm] (絶対位置)。リセットの場合、設定位置がリセットされます。つまり、間隔制御を再び有効にする前に、新しい設定位置を指定する必要があります。
SET_DIST*<expr>* **[V2.11.2804.03から]**	ワーク表面に対する一定間隔の指定[mm]。リセットの場合、間隔がリセットされます。つまり、間隔制御を再び有効にする前に、新しい間隔を指定する必要があります。
NO_MOVE	通常は、間隔制御がオフになると、位置補正オフセットが(間隔規制により)破棄されます。NO_MOVEをOFFと組み合わせてプログラムすることによって、この動作が停止します。変更された位置でチャンネルが初期化されます。次に、位置オフセットがNCプログラムの最初の軸動作によって破棄されます。
VAL1*<expr>*-VAL5*<expr>*	自由に割り当てできる値(5)
\	複数行にわたるコマンドの明確なプログラミングのための区切り記号(「バックスラッシュ」)。

キーワードON/OFF、FREEZE、CHECK_POS、およびREFは、同じコマンドシーケンスにおける排他的キーワードです。キーワードSET_POS、SET_DIST、およびVAL1-VAL5は常に他のキーワードと組み合わせてプログラムすることができます。

プログラム終了時の距離制御がまだ有効である場合、自動的に選択解除されません。
リセットのイベントでは、有効な距離制御が常に自動的に選択解除されます。

プログラミング例

```
%DIST_1
N10 Z[DIST_CTRL  SET_POS=30]       Set expected position of
                                   work piece surface
N20 Z[DIST_CTRL  ON]               Selection
:
Nxx Z[DIST_CTRL OFF]               Deselection
N999 M30

%DIST_2
N10 Z[DIST_CTRL ON  SET_POS=30]    Selection + set expected position
                                   of work piece surface
:
Nxx Z[DIST_CTRL FREEZE]            Hold position
:
Nxx Z[DIST_CTRL OFF]               Deselection
N999 M30

%DIST_3
N10 Z[DIST_CTRL ON  SET_POS=50]    Selection + set expected position
                                   of work piece surface
:
Nxx Z[DIST_CTRL OFF NO_MOVE]       Turning distance control off,
                                   Z axis is not moving
Nxx G0 Z100                        While moving on target position 100
                                   the generated correction offset is
                                   considered
N999 M30

%DIST_4
N10 Z[DIST_CTRL SET_POS=30 SET_DIST=10]    Setting of distance parameters
N20 Z[DIST_CTRL ON]                Selection with predefined workpiece
                                   surface (SET_POS)
:
Nxx Z[DIST_CTRL OFF]               Deselection
:
Nxx Z[DIST_CTRL ON CONST_DIST]     Selection with predefined distance to
                                   the workpiece surface (SET_DIST)
:
Nxx Z[DIST_CTRL OFF]               Deselection
N999 M30
```

21.3 プログラム可能な軸オーバーライド

このコマンドは軸送りを可能にします。必要に応じて、 NCプログラムの送りと早送りブロックの異なる影響を可能にします。割り当てられた軸が動いている場合、経路移動中の各軸に対応したプログラムされたオーバーライドが有効です。この機能によって、PLCによる送りのリアルタイム効果は影響を受けません。

第2のファンクションとしてプログラム可能なパスオーバーライド [▶ 344]も利用できます。

軸特有のオーバーライド値が異なる、同じNCブロックの複数の動く軸では、常に最小のオーバーライドが行われます。さらにパスオーバーライドも定義されている場合、有効なオーバーライドは、両方のオーバーライド値を乗算して得られます。

ファンクションG166 [▶ 166]は、プログラムされたオーバーライド値の効果を抑止します。

<*axis_name*> [OVERRIDE FEED_FACT<*expr*> RAPID_FACT<*expr*> { \ }]

<*axis_name*>	軸の名前
OVERRIDE	軸特有のオーバーライドプログラミングの識別。必ず**1番目に**プログラムされるキーワードであること。
FEED_FACT<*expr*>	送りブロックのオーバーライド係数[0.1% - 200%]
RAPID_FACT<*expr*>	早送りブロックのオーバーライド係数[0.1% - 200%]
\	複数行にわたるコマンドの明確なプログラミングのための区切り記号(「バックスラッシュ」)。

プログラミング例

```
%ax_override

N10 G01 X100 Y100 Z100 F1000
N40 X[OVERRIDE FEED_FACT=20 RAPID_FACT=60] Axis override X G01 20%, G00 60%
N50 Y[OVERRIDE FEED_FACT=30 RAPID_FACT=70] Axis override Y G01 30%, G00 70%
N60 Z[OVERRIDE FEED_FACT=40 RAPID_FACT=80] Axis override Z G01 40%, G00 80%
N50 G00 X0                                 G00 motion with 60% override
N60     Y0                                 G00 motion with 70% override
N70     Z0                                 G00 motion with 80% override
N80 G01 X100 F2000                         G01 motion with 20% override
N90     Y100                               G01 motion with 30% override
N100    Z100                               G01 motion with 40% override
N110    X200 Y200                          G01 motion with 20% override
N120    X300 Y300 Z200                     G01 motion with 20% override
M30
```

21.4 プログラム可能な加減速過負荷

プロセスに影響する輪郭に関連する技術的理由から、例えば、多項式輪郭での一定の経路速度を保証するために、ドライブの所定の動的リミット値を超える必要がある場合があります。

従軸特有のコマンドは、割り当てられたパラメータP-AXIS-00394と組み合わせて使用すると、軸の動的重み付け(%)が許容最大加速度P-AXIS-00008を超えることができます。P-AXIS-00394は、軸の加速度加重係数(パーミル)の許容上限を表します。この加重係数は、対応する有効な傾斜プロファイルの送り動的リミット値を指しています。

現在、加速度の重み付けのファンクションを輪郭加工モード6 [▶ 109]と組み合わせて使用することができます。

<*axis_name*> [DYNAMIC DIST_SOFT | ACC_FACT<*expr*> { \ }]

<axis_name>	軸の名前
DYNAMIC	軸の動的重み付けの識別。必ず**1番目に**プログラムされるキーワードであること。
DIST_SOFT	多項式輪郭加工モード6の識別
ACC_FACT*<expr>*	軸特有の加重係数(%)
\	複数行にわたるコマンドの明確なプログラミングのための区切り記号(「バックスラッシュ」)。

最小の重み値は100%です!

最大の重み値はP-AXIS-00394に制限されています。

プログラミング例

```
%dynamic
N10 #SLOPE[TYPE=STEP]
N20 #CONTOUR MODE[DIST_SOFT PATH_DIST=35 ACC_MAX=100 ]
N30 C[DYNAMIC DIST_SOFT ACC_FACT=200]
   (* Acceleration overload factor for C axis 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 座標移動での軸の同期

TwinCat CNC: V2.11.2013.22以降

その他: V263以降

一部の特定プロセスでは、座標移動と組み合わせた単一軸の同期移動が必要です。特定のプログラムされた位置では、従軸が事前に定義された位置にあって、事前に定義された速度で動くことが必要です。従軸は、同期が取り消されるまで同期速度で動きます。

代表的な用途の例は、エンドレス材料が連続的に動くような機械で見ることができます。この場合、連続的な座標移動中に材料を切削する必要があります。特定のマスタ位置(ワーク長さ)で、回転ナイフが切削位置にあることが必要です。次に、切削が終わるまでナイフが同期速度で動きます。

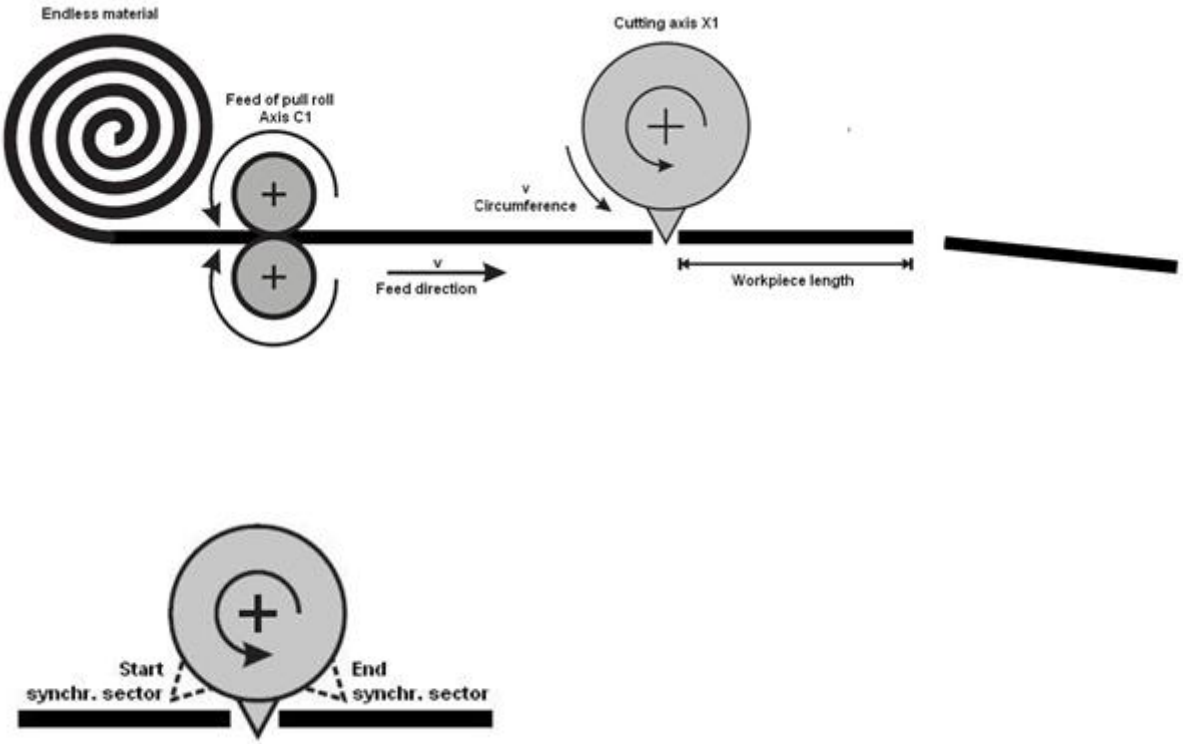


図 121: 図20-1: 同期切削

制限事項:

次の場合に軸を同期することはできません。

- 今のところ、この軸が座標移動の一部である。

設定:

機能を使用するには、スタートアップリスト([STUP])で以下の設定を行う必要があります。

configuration.channel[0].path_preparation.function FCT_DEFAULT | **FCT_SYNC**

configuration.channel[0].interpolator.function FCT_IPO_DEFAULT | **FCT_SYNC**

同期移動のプログラミング構文:

```
<axis_name> [ SYNC IN| OUT G90 | G91 G00 | G01 FEED<expr>
              FEED_MAX_WEIGHT<expr> POS<expr> DIST<expr> { \ } ]
```

<axis_name>	同期する必要がある軸の名前
SYNC	軸の同期移動の識別。必ず**1番目に**プログラムされるキーワードであること。
IN	同期移動の開始をマークするための識別。
OUT	同期移動の終了をマークするための識別。
G90/G91	アブソリュート/インクリメンタル指令
G00/G01	早送り/直線補間
FEED*<expr>*	軸特有の送り[経路ユニット/min]
FEED_MAX_WEIGHT*<expr>*	加重係数[%]、軸特有の最大送りP-AXIS-00212。100%未満の重み付け値のみ許可されます(G194を参照 [▶ 144])。
POS*<expr>*	同期速度に達する軸位置[mm、inch]
DIST<expr>	同期速度で動いた間隔[mm、inch]
\	複数行にわたるコマンドの明確なプログラミングのための区切り記号(「バックスラッシュ」)。

プログラミング例

```
%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 axis reaches maximum velocity on axis position 130,
             ;while X axis is reaching position 67.913 at this point

N040 X=1.5   ;A and X axes move synchronously, in doing so the velocity
             ;of X axis is adjusted in a way, that it moves a distance
             ;of 3mm, while A axis is moving 70° with maximum velocity
N050 X=1.5

N060 X=14.541 A[SYNC OUT G91 G0 POS160] G261
             ;At beginning of this block synchronous motion is cancelled
             ;Path axes are moving with programmed feed, A axis moves
             ;independently on the stated 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 軸多項式のプログラミング

TwinCAT CNC: V2.11.2016.08から

その他: V263から

軸特有の多項式の特性

軸のモーションを多項式の軸特有の定義によってプログラムすることができます。

この軸特有の多項式モーションは直線移動(G00、G01)についてプログラム可能です。そのため、対応する有効なGファンクション(G00またはG01)の動的パラメータを使用します。

各軸多項式について、多項式パラメータの補間のためのアーク長を設定します。アーク長をプログラムしないと、値1.0が割り当てられます。

多項式係数は、軸名の後の括弧で**昇**順に定義します。識別しやすいように、まずキーワードPOLYをプログラムする必要があります。不要な高次多項式を省略することができます。係数をプログラムしないと、値0が割り当てられます。最初の係数「A0」のみを設定する必要があります。

多項式の可能な最大次数は5です。

評価

多項式係数は**絶対**軸位置に関連しており、5次多項式に入力されます。

$$p(s) = A0 + A1 * s + A2 * s^2 + A3 * s^3 + A4 * s^4 + A5 * s^5$$

アーク長は、既に実行された経路長と同時に、多項式についてゼロから、プログラムされたアーク長まで補間されます。

これは、以下の時点の多項式軸の絶対位置(mmまたは度)に当てはまります。

1. 移動開始時 $(s = 0)$:

$$p(0) = A0$$

2. 移動終了時($s = L$) :

$$p(L) = A0 + A1 * L + A2 * L^2 + A3 * L^3 + A4 * L^4 + A5 * L^5$$

軸特有の多項式プログラミングは現在のNCブロックでのみ有効です(ノンモーダル)。必要に応じて、対応する軸の次の移動ブロックで再びプログラムする必要があります。

プログラミング

スキーム:

Axis [POLY L<arc_length> A0 A1 A2 A3 A4 A5]

例:

X [POLY L=1.0 A0=0.1 A1=0.2 A2=0.3 A3=0.4 A4=0.5 A5=0.6]

同じNCブロックで、直線移動と**単一の**軸特有の多項式の混合プログラミングが可能です。有効なシフト(G54、G92、#PSET...)がプログラムされた多項式位置に含まれています。

プログラムされた多項式軸では、動作の計算も監視も実行されません。また、ソフトウェアリミットのコマンド値固有の監視も実行されません(リミットの現在値固有の監視のみ実行されます)。

軸特有の多項式のプログラミング構文:

<axis_name> **[POLY [L***<arc_length>* **]** **A0***<a0>* **[A1***<a1>* **A2***<a2>* **A3***<a3>* **A4***<a4>* **A5***<a5>* **]]**

<axis_name>	多項式軸の名前
POLY	軸の多項式プログラミングの識別。必ず**1番目に**プログラムされるキーワードであること。
L*<arc_length>*	動かす必要がある軸のアーク長(オプション。プログラムされない場合、Lの値は1.0になります。プログラムする値は > 0であること)
A0*<a0>*	第1の多項式係数、必須(多項式の開始値)
A1*<a1>* - A5*<a5>*	第2～第6の多項式係数、x、x^2、x^3、x^4、x^5 (オプション、プログラムされていない係数の初期値は0)

プログラミング例

```
;C-Axis, arc length L = 0.7  A0 = 0.1, A1 = 0.3, A2 = 0.5
Nxx C[POLY L=0.7 A0=0.1 A1=0.3 A2=0.5]


;X-Axis, arc length L = 0.3, A0 = 0.2, A1= 0.5
Nxx X[POLY L=0.3 A0=0.2 A1=0.5]


;Simple programming with arc length (Default L1), only A0-coefficient
Nxx X[POLY A0=0.2]


;Mixed programming of linear motion and axis polynomial
Nxx G01 F1000 X100 Y150 C[POLY L=0.7 A0=0.1 A1=0.3 A2=0.5]
```

キーワードと値の間の等号の指定はオプションです。

22 付録

22.1 コマンドの概要

22.1.1 Gファンクション(G..)

G00	早送り
G01	直線補間
G02	円弧補間、時計回り(cw.)
G03	円弧補間、反時計回り(ccw.)
G02/G03	ヘリカル補間
G04	ドウェル
G05	工具径補正(TRC)の直接接線選択/選択解除
G08	ブロック開始時の加速
G09	ブロック終了時の減速
G10	送り速度(一定)
G11	送り速度(適合)
G12	コーナー送り低減の選択解除
G13	コーナー送り低減の選択
G17	XY平面
G18	ZX平面
G19	YZ平面
G20	ミラーリングの選択解除
G21	Y軸上のプログラムされた経路のミラーリング
G22	X軸上のプログラムされた経路のミラーリング
G23	G21とG22の重ね合わせ
G25	工具径補正(TRC)の直線遷移
G26	工具径補正(TRC)の円弧遷移
G33	ねじ切り、均一リード
G40	工具径補正(TRC)/SRK選択解除
G41	工具径補正(TRC)/SRK(輪郭の左)
G42	工具径補正(TRC)/SRK(輪郭の右)
G51	直径プログラミングの選択
G52	直径プログラミングの選択解除
G53	ゼロオフセットの選択解除
G54 ~ G59	ゼロオフセットの選択
G60	正確な停止

G61	多項式輪郭加工
G63	タッピング
G70	インチデータ入力
G71	メートルデータ入力
G74	原点復帰
G80～G89	暗黙的なサブプログラム呼び出し
G90	アブソリュート指令
G91	インクリメンタル指令(相対)
G92	座標プリセット
G93	加工時間を指定するためのFワード
G94	送り速度を指定するためのFワード
G95	送り速度(1回転当たりのmm)
G96	一定の切削速度
G97	スピンドル速度の指定(1/min)
G98	マイナス側のソフトウェアリミットスイッチの設定
G99	プラス側のソフトウェアリミットスイッチの設定
G100	計測ファンクション(タイプ1～7)
G101	計測オフセットをシフトに含める
G102	計測オフセットをシフトから抽出する
G106	目標点までの移動を伴う計測
G107	ブロックグローバルエッジバンディングの選択解除
G108	エッジバンディング
G112	ギヤ切替
G115	先読みの一般的影響
G116	先読みを操作する
G117	先読みをオンにする(初期設定)
G128	最大速度の重み付け
G129	早送り速度の重み付け
G130	加速度の軸特有の重み付け
G131	加速度の軸グループ特有の重み付け(G01/G02/G03用)

G132	ランプ時間の軸特有の重み付け
G133	ランプ時間の軸グループ特有の重み付け(G01/G02/G03用)
G134	幾何学的ランプ時間の軸グループ特有の重み付け
G135	フィードフォワード制御の選択
G136	フィードフォワード制御の重み付けの指定
G137	フィードフォワード制御の選択解除
G138	工具径補正(TRC)の直接選択/選択解除
G139	工具径補正(TRC)の間接選択/選択解除
G140	輪郭マスキングの選択解除
G141	輪郭マスキングの選択
G150	スプライン軸補間の解除
G151	スプライン軸補間の選択
G159	拡張されたゼロオフセット
G160	ゼロオフセットの軸特有の有効化
G161	円弧中心点、絶対指定
G162	円弧中心点、相対指定
G163	半径プログラミング
G164	円弧中心点オフセットオフ
G165	円弧中心点オフセットオン
G166	オーバーライド100%
G167	スピンドルオーバーライド100%
G193	経路関連送り適合
G194	最大送り速度の重み付け
G196	最大スピンドル速度(G96用)
G200	並行補間を伴わない手動モードの選択
G201	並行補間を伴う手動モードの選択
G202	並行補間を伴う手動モードの選択解除
G231	加速度の軸グループ固有の重み付け(G00用)
G233	ランプ時間の軸グループ固有の重み付け(G00用)
G237	工具径補正(TRC)の垂直選択/選択解除
G238	工具径補正(TRC)の内部コーナー選択
G239	ブロックを使用しない工具径補正(TRC)の直接選択/選択解除
G260	多項式輪郭加工の選択解除
G261	多項式輪郭加工の選択
G293	時間関連送り適合

G301	面取りの挿入
G302	ラウンディングの挿入
G310	中断可能なブロック(計測タイプ5～6)
G351	軸情報によるミラーリング
G359	正確な停止の選択解除
G360	正確な停止の選択
G800～G819	追加の暗黙サブプログラム呼び出し
G900	ブロック終了後の減速への切り替え
G901	ブロック終了後の減速

22.1.2 Mファンクション(M..)

M00	プログラム停止
M01	オプショナルストップ
M02	プログラム終了
M03	スピンドル回転cw
M04	スピンドル回転ccw
M05	スピンドル停止
M17	サブプログラムの終了
M19	スピンドルの位置決め
M29	サブプログラムの終了
M30	プログラム終了
M40～45	スピンドルギヤレンジの選択

22.1.3 DINおよびISG拡張に準拠する予約済みファンクション

D	工具ジオメトリ補正
E*	ブロック終了時の送り速度
F	ブロック内の送り速度
H	PLCに対する技術項目
L	サブプログラム呼び出し(グローバル)
LL	サブプログラム呼び出し(ローカル)
L、LL CYCLE	サイクル呼び出し
L SEQUENCE	ブロックシーケンスの呼び出し
N	ブロック番号
P	パラメータ計算
R	半径プログラミング
S	スピンドル速度
S.POS	スピンドル位置
S.OFFSET	多条ねじの角度オフセット
T	工具選択
/	NCブロックの省略
****	NCブロックにおける改行

22.1.4 制御ブロックステートメント($..)

BREAKステートメント($BREAK)
CONTINUEステートメント($CONTINUE)
DOループ($DO, $ENDDO)
REPEATループ($REPEAT, $UNTIL)
FORループ($FOR, $ENDFOR)

GOTOステートメント($GOTO)

IF- ELSE分岐($IF, $ELSE, $ELSEIF, $ENDIF)
SWITCH分岐($SWITCH, $CASE, $DEFAULT, $ENDSWITCH)
WHILEループ($WHILE, $ENDWHILE)

22.1.5 追加ファンクション(#..)

#ACS ON/OFF	固定具適合座標系の定義/有効化
#ADD	ブロック終了時の追加情報
#AKIMA STARTVECTOR	開始接線の定義
#AKIMA ENDVECTOR	終了接線の定義
#AKIMA TRANS	遷移タイプの指定
#AX DEF	軸設定の定義(拡張構文)
#AX DEF DEFAULT	初期軸設定の読み込み(拡張構文)
#AX LINK ON/OFF/ALL	プログラミング結合命令(拡張構文)
#AX LOCK / UNLOCK	PTP中の軸動作のロック
#AX REQUEST	軸の要求(拡張構文)
#AX RELEASE	軸の解放(拡張構文)
#AX RELEASE ALL	すべての軸の解放(拡張構文)
#BACKWARD STORAGE CLEAR	バックワードストレージのクリア
#BLOCKSEARCH LOCKED/ RELEASED	ブロック検索用のプログラム領域のロック
#CACHE LOAD/CLEAR/ALL	**ローカルキャッシュへのNCプログラムの読み込み[FCT-C23]**
#CALL AX	軸の要求
#CAX	C軸操作のためのスピンドル軸の要求
#CAX OFF	スピンドルへのC軸の変換
#CAXTRACK ON/OFF	自動軸追従
#CHANNEL INIT	すべてまたは一部の特定軸の現在のコマンドまたは現在位置の受け入れ
#CLEAR CONFIG	保存済み設定の削除
#COMMAND WR/SYN	SERCOSコマンドの書き込み
#COMMAND WAIT/SYN	SERCOSコマンドの書き込み
#COMMENT BEGIN/END	内部ブロックが有効なコメント
#CONTOUR MODE	多項式輪郭加工のパラメータ設定
#CONTROL AREA BEGIN/END	制御領域の定義
#CONTROL AREA ON/OFF	制御領域の選択/選択解除
#CONTROL AREA CLEAR	制御領域の削除
#CORNER PARAM	コーナー送り低減のパラメータ設定
#CS ON/OFF	加工座標系の定義/有効化
#CYL	外側面加工の選択
#CYL OFF	外側面加工の選択解除
#CYL ORI LATERAL	**5/6軸管加工[FCT-M5]**
#DELETE	ユーザ定義された変数またはパラメータの削除
#DISTANCE PROG START ON / OFF / CLEAR	プログラム開始からの移動距離の記録[FCT-C6]
#DRIVE WR SYN #DRIVE WAIT SYN	ドライブファンクションの切り替え
#DYNAMIC WEIGHT ON / OFF	動的な重み付けの切り替え
#ECS ON/OFF	エフェクタ座標系の選択/選択解除
#EDGE MACHINING	エッジ加工の制御

#ENABLE AX LINK	結合命令の選択
#DISABLE AX LINK	
#ERROR	ユーザ定義エラーの出力
#EXPL SYN	明示的な同期
#EXPORT VE	構造体へのV.E.変数のエクスポート[FCT-C22]
#FACE	面加工の選択
#FACE OFF	面加工の選択解除
#FGROUP	送りグループの定義
#FGROUP ROT	回転ワークのための送り軸
#FGROUP WAXIS	最も弱い軸が送り軸になる
#FILE NAME	ファイル名の定義
#FILE RENAME	ファイル名の変更
#FILE DELETE	ファイルの削除
#FILE EXIST	ファイルの存在のチェック
#FILTER	フィルタプログラミング
#FLUSH	動作の中断を伴うCNCチャンネルのフラッシング
#FLUSH CONTINUE	動作の停止を伴わないCNCチャンネルのフラッシング
#FLUSH WAIT	解読処理と補間の同期
#FRICTION ON / OFF	摩擦補償のためのデータ記録[FCT-C25]
#FREE TOOL CHANGE ON/OFF	有効な同期動作中の工具交換
#GET CMDPOS	1つの軸の現在のコマンド位置の受け入れ
#GET ACTPOS	1つの軸の現在の実位置の受け入れ
#GET MANUAL OFFSETS	現在の手動オフセットの受け入れ
#GET WCS POSLIMIT	ワーク座標系内の移動制限
#HANDWHEEL	ハンドホイール操作モードのパラメータ設定
#HSC ON/OFF	自由形状の表面プログラミング
#IDENT WR/RD/SYN	SERCOSパラメータの書き込み/読み取り
#IF/ELSE/ENDIF	条件解釈
#INIT MACRO TAB	マクロテーブルの初期化
#INIT V.E.xx	V.E.変数の初期化
#JOG CONT	連続ジョグモードのパラメータ設定
#JOG INCR	インクリメンタルジョグモードのパラメータ設定
#KIN ID	機械のキネマティクス
#LOAD CONFIG	保存された設定の読み込み
#MACHINE DATA	機械データの書き込み
#MAIN SPINDLE	メインスピンドルの切り替え
#MANUAL LIMITS	手動オフセットリミット値のプリセット
#MCS ON/OFF	機械軸座標系への一時的遷移
#MCS TO WCS	ワーク座標への機械座標の変換
#MEAS MODE	計測タイプの切り替え
#MEAS	拡張された計測オプション
#MSG	NCプログラムからのメッセージ
#MSG INFO	メッセージ情報

#MSG SAVE	ファイルへのメッセージの書き込み
#NIBBLE ON / OFF	ニブリングの選択/選択解除
#OPTIONAL EXECUTION ON/OFF	経路上の前進/後退中のプログラム部分の省略
#OTC ON / OFF	オンライン工具補正[FCT-C20]
#OVERRIDE	プログラム可能なパスオーバーライド
#PSET	プログラミング位置プリセットの選択/選択解除
#PRESET	
#PTP ON/OFF	補正移動なしの位置決め
#PUNCH ON / OFF	パンチングの選択/選択解除
#PUT AX	軸の解放
#PUT AX ALL	すべての軸の解放
#ROTATION ON/OFF	形状回転
#RTCP ON/OFF	回転工具中心点の選択/選択解除(古い構文)
#SAVE CONFIG	設定の保存
#SCALE ON / OFF	輪郭のスケーリング
#SEGMENTATION ON/OFF	直線ブロックおよび円弧ブロックの区分の選択/選択解除
#SET AX	軸設定の定義
#SET AX LINK	プログラミング結合命令
#SINGLE STEP	シングルステップモード用のプログラム領域のロック
#SPLINE ON/OFF	スプライン軸補間の選択/選択解除
#SPLINE TYPE AKIMA	AKIMAスプラインの選択
#SPLINE TYPE BSPLINE	Bスプラインの選択
#SIGNAL	信号の送信
#SIGNAL REMOVE	ブロードキャスト信号の削除
#SLOPE	加速度プロファイルのパラメータ設定
#SLOPE DEFAULT	加速度プロファイルのパラメータ設定
#SUPPRESS OFFSETS	オフセットの停止
#TANGFEED	正接送り速度適用のための最小半径
#STROKE DEF BEGIN / END	パンチング/ニブリングのストロークシーケンスの定義
#TIME	ドウェル時間
#TIMER	時間計測
#TLC ON/OFF	工具長補正の選択/選択解除
#TOOL DATA	工具データの需要
#TOOL DEF	複数の工具を単一ユニットに分ける[FCT-C18]
#TOOL LIFE READ/REMOVE	工具寿命データの読み取り/削除
#TOOL ORI CS	工具アライメント
#TOOL PREP	工具交換の準備
#TOOL REFRESH	工具データの更新
#TRAFO ON/OFF	回転工具中心点の選択/選択解除
#TRANSVELMIN ON/OFF	最小ブロック遷移速度
#TRC	工具径補正の追加オプション
#VAR...#ENDVAR	ユーザ定義されたパラメータまたは変数の宣言
#VECTOR LIMIT ON/OFF	動的なパスリミットの調整

#VOLCOMP ON / OFF	体積補正[FCT-C26]
#WAIT	信号の待機
#WAIT FOR	イベントを待つ
#WAIT INDP	非同期独立軸を待つ
#WAIT INDP ALL	すべての非同期独立軸を待つ
#WCS TO MCS	機械座標へのワーク座標の変換

22.1.6 追加の軸特有のファンクション(<X>[..])

INDP_SYN	同期(ブロック単位)独立軸移動
INDP_ASYN	非同期(複数ブロック)独立軸移動
OSC	振動軸
COMP	軸補正の選択/選択解除
DIST_CTRL	間隔制御された軸
OVERRIDE	プログラム可能な軸オーバーライド
DYNAMIC	プログラム可能な加減速過負荷
LIFT	軸の上昇/下降[FCT-A11]
LIFT_START/LIFT_END	
SYNC IN/OUT	座標移動での軸の同期
POLY	軸多項式のプログラミング

22.1.7 PLC-Openファンクション(<X>[MC_..])

MC_Home	基準点移動
MC_MoveAbsolute	絶対位置への軸の移動
MC_MoveAdditive	命令された位置に加えた相対動作
MC_MoveRelative	現在の位置に加えた相対移動
MC_MoveSuperImposed	既に有効な移動に加えた相対動作
MC_MoveVelocity	指定された速度でのエンドレス動作
MC_Stop	軸動作の停止
MC_GearIn	ギヤ比によるギヤ結合
MC_GearOut	ギヤ結合の解放
MC_Phasing	結合の位相差

22.1.8 変数プログラミング(V.)

V.A. ...	軸特有の変数
V.SPDL. ...	スピンドル特有の変数
V.G. ...	グローバル変数
V.P. ...	ユーザ定義された変数、プログラムグローバル
V.S. ...	ユーザ定義された変数、グローバル
V.L. ...	ユーザ定義された変数、プログラムローカル
V.E. ...	外部変数
V.TOOL. ...	工具識別変数
V.TLM. ...	工具寿命変数

22.1.9 その他のファンクション

数式 [▶ 24]と文字列処理 [▶ 30]のプログラミングでは、複数の演算操作/ファンクションを利用できます。

22.1.10 移行されたNCコマンド

以下の表は、機能や構文に関する理由から新しい構文に移行されたNCコマンドの一覧です。互換性の理由で、以前のコマンドも利用できますが、新しいNCプログラミングでは使用しないでください。

古い構文:	新しい構文	以下のバージョンから
#SET DEC LR SOLL	#CHANNEL INIT [...] [▶ 160]	V2.10.1504.00
#VECTORVEL ON / OFF	#VECTOR LIMIT ON / OFF [...] [▶ 322]	V2.10.1507.02
#VECTORACC ON / OFF	#VECTOR LIMIT ON / OFF [...] [▶ 322]	V2.10.1507.02
#INIT MAKRO TAB	#INIT MACRO TAB [▶ 505]	V2.11.2010.02
G200 #ACHSE [...]	G200 X.. X..Y.. [▶ 152]	V2.11.2010.02
G201 #ACHSE [...]	G201 X.. X..Y.. [▶ 151]	V2.11.2010.02
G202 #ACHSE [...]	G202 X.. X..Y.. [▶ 151]	V2.11.2010.02
#GET IPO OFFSET	#GET MANUAL OFFSETS [▶ 159]	V2.11.2010.02
#SET OFFSET [...] X	#MANUAL LIMITS [...] [▶ 156]	V2.11.2010.02
#SET HR [...] X	#HANDWHEEL [...] [▶ 154]	V2.11.2010.02
#SET TIP [...] X	#JOG CONT [...] [▶ 154]	V2.11.2010.02
#SET JOG [...] X	#JOG INCR [...] [▶ 155]	V2.11.2010.02
#SET IPO SOLLPOS [...]	#GET CMDPOS [...] [▶ 162]	V2.11.2010.02
#SET SLOPE PROFIL [...]	#SLOPE ... [...] [▶ 261]	V2.11.2010.02
#SET ASPLINE STARTTANG X.. X..Y..	#AKIMA STARTVECTOR X.. X..Y.. [▶ 265]	V2.11.2010.02
#SET ASPLINE ZIELTANG X.. X..Y..	#AKIMA ENDVECTOR X.. X..Y.. [▶ 266]	V2.11.2010.02
#SET ASPLINE MODE [...]	#AKIMA TRANS [...] [▶ 264]	V2.11.2010.02
#SET CORNER PARAM [...]	#CORNER PARAM [...] [▶ 119]	V2.11.2010.02
#SET TANGFEED RMIN [...]	#TANGFEED [...] [▶ 237]	V2.11.2010.02
#SET SPLINE ON / OFF	#SPLINE ON [▶ 263] / OFF [▶ 263]	V2.11.2010.02
#SET SPLINETYPE AKIMA	#SPLINE TYPE AKIMA [▶ 262]	V2.11.2010.02
#SET SPLINETYPE BSPLINE	#SPLINE TYPE BSPLINE [▶ 268]	V2.11.2010.02

23 参照先

[1] 取扱説明書/「チャンネルパラメータ」の一般的説明[CHAN]

番号	優先度
1	構造体makro_def[i]の要素。*
2	構造体synchro_data.koppel_gruppe[0]の要素。*
3	構造体spindel[i]の要素。*
4	構造体spindel[i].range_table[i]の要素。*
5	構造体gruppe[j].achse[i]の要素。*
6	構造体speed_limit_look_aheadの要素。*
7	構造体dynamic_weighting[i]の要素。*

[2] 取扱説明書/「軸パラメータ」の一般的説明[AXIS]

番号	優先度
1	構造体getriebe[i].slope_profilの要素。*
2	構造体getriebe[i].lslope_profilの要素。*
3	構造体filter[i]の要素。*

[3] 取扱説明書/「ゼロオフセットデータ」の一般的説明[ZERO]

[4] 仕様SERCOSインターフェイス、IEC 61491 [SERC]

[5] 取扱説明書/「工具データ」の一般的説明[TOOL]

[6] システムパラメータの制御/メーカ固有の設定の取扱説明書[SYSP]

[7] 取扱説明書/「開始リスト」の一般的説明[STUP]

[8] 取扱説明書/「外部変数」の一般的説明[EXTV]

[9] PLCopen用のモーション制御プラットフォーム[MCP-P1]

詳細はこちら:
www.beckhoff.com/TF5200

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Germany
+49 5246 9630
info@beckhoff.com
www.beckhoff.com