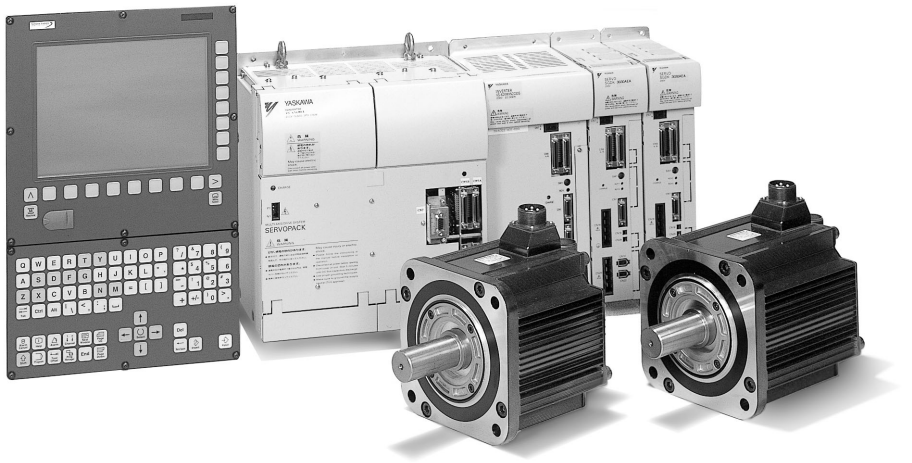
Yaskawa Siemens
numerical controls

Yaskawa Siemens CNC シリーズ

ユーザーズマニュアル
プログラミング編
上級説明書



資料番号　NCSI-SP02-07

安川シーメンス NC 株式会社はシーメンス株式会社に統合の後，2010 年 8 月より
シーメンス・ジャパン株式会社へ社名を変更いたしました。本書に記載の「安川
シーメンス NC 株式会社」などの社名に類する名称は「シーメンス・ジャパン株式
会社」へ読み替えをお願いします。

本マニュアルは Yaskawa Siemens 840DI，Yaskawa Siemens 830DI 両モデル用に
作成されています。本文中の記述では両モデルの機能差は区別されておりませんの
で，それぞれのモデルにどの機能が標準装備されているか，どの機能がオプション
で装備可能かについては別途，機能一覧表をご参照ください。また，本文中に
840DI と言った表現が出て来ますが，830DI も意味していることがあるとご理解
ください。

Yaskawa Siemens
ユーザマニュアル プログラミング編
上級説明書

ユーザ文書

対象制御装置

制御装置
Yaskawa Siemens 840DI

10.00 版

Yaskawa Siemens 文書

版の履歴

今回の版の概略説明および今までに作成された版を下記に示します。

「備考」欄のコードが、各版のステータスを示しています。

「備考」欄のステータスコードの意味は次のとおりです。

A . . . . . 新規作成

B . . . . . 新しいオーダ番号で印刷し直した未改訂の文書

C . . . . . 新しいステータスの改訂版

前回の版以降に実際に変更があったページには、そのページのヘッダ部分に新しい版のコードが示されています。

版	オーダ番号	備考
10.00	NCSI-SP02-07	A

本文書に説明のない他の機能でも制御装置で実行できる場合がありますが、そのような機能は新しい制御装置やサービス時に利用できるとは限りません。

本文書の記述と、対象となるハードウェアおよびソフトウェアとが一致しているかどうかは十分に確認されています。しかし相違点がまったくないとは言えず、完全に一致しているとは保証できません。本文書に記載されている情報は定期的に検討され、必要な変更は次の版に反映されます。さらなる改善のために皆様のご意見をお待ちしています。

本内容は予告なしに変更されることがあります。

はじめに

文書の概要

Yaskawa Siemens 文書は次の 3 つのレベルで構成されています。

- 一般文書
- ユーザ文書
- 製造業者／サービス文書

対象読者

本マニュアルはプログラマ用です。Yaskawa Siemens 840DI（以降 840DI と略す）のプログラミング方法を詳しく説明しています

標準機能の範囲

本プログラミングガイドは標準機能のみを説明しています。拡張機能あるいは機械メーカが行った変更については，関連機械メーカが提供するマニュアルを参照してください。

840DI に関する他の出版物，および関連する制御装置全般に関する（ユニバーサルインタフェース，測定サイクルなどの）出版物など，詳しい内容についてはお近くの当社営業所にお問い合わせください。

本文書に記載がないにも関わらず，制御装置で実行可能なファンクションが存在する場合がありますが，これは，保守時または別の新しい制御装置でもそれらのファンクションが提供されることを意味するものではありません。

適用マシン

本プログラミングガイドは次の制御装置に適用されます：

Yaskawa Siemens 840DI

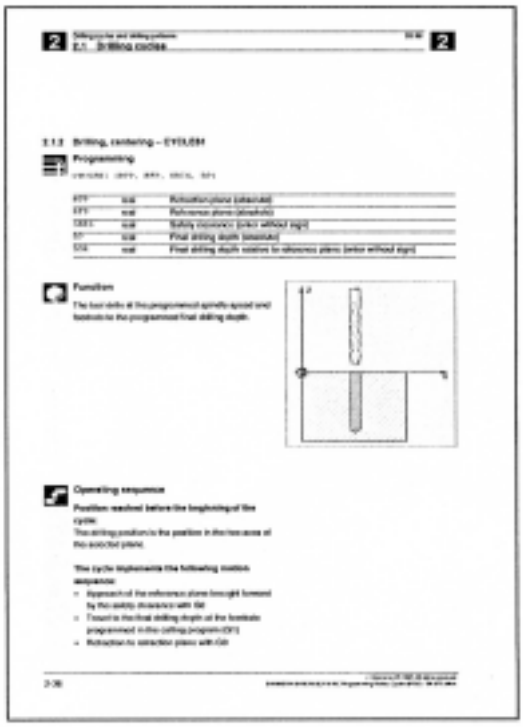
説明の構成

すべてのサイクルとプログラミング法は，できるだけ共通した内部構成に準拠して説明されています。情報をレベルごとに分類してあるので，必要な情報を直接迅速に見つけることができます。

1. 機能の概要

たまにしか使用しないコマンドを調べたいかパラメータの意味を知りたい場合, ファンクションのプログラミング法, およびコマンドおよびパラメータの説明が一目で分かるように工夫されています。

この情報は常に機能説明の先頭に示されています。



(注)

個々のコマンドとパラメータについて, プログラミング言語で使用できるすべての表現方法を示すことは不可能です(もしそうすると膨大なマニュアルになります)。このため，本マニュアルでは現場で最もよく使用されるプログラミング法のみを示しています。

2. 詳細説明

本マニュアルの論理的な説明では次のことを詳しく説明します：

コマンドの目的

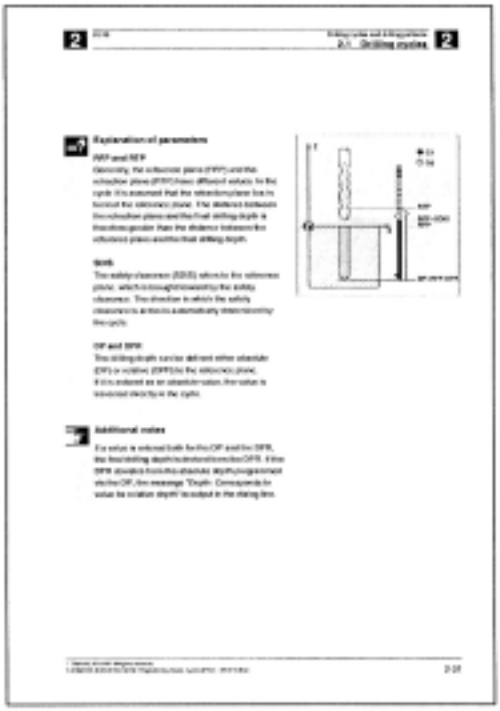
コマンドの作用

操作手順

パラメータの作用

注意事項

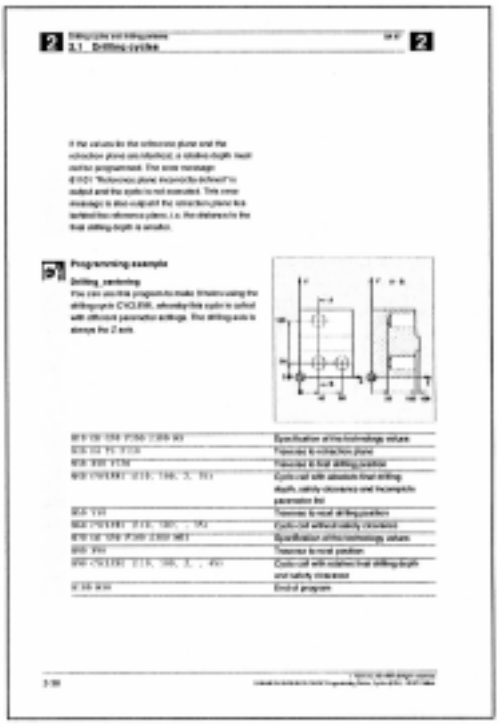
本マニュアルの論理的な部分は，NC 初心者にとって特に有益です。840DI 制御装置が提供する全機能と性能の概要を把握するため，本マニュアルを少なくても 1 回は読み通してください。



3. 論理から実践へ

実際にどのようにコマンドをプログラミングしたらよいかについては，本マニュアルのプログラミング例を参照してください。

ほとんどのコマンドについて，論理的な部分の後で応用例が示されています。



記号の説明


オペレータの入力順序


説明


機能


パラメータ


プログラミング例


プログラミング


追加説明


関連する他の文書およびセクション


注意表示および危険表示

参考

840DI は，最新の技術と，安全規格，慣習および規則に準拠して製造されています。

追加装置

840DI 関連の特殊追加装置および拡張機能を使用することで，840DI 制御装置の応用範囲を用途に合わせて拡張することができます。

作業者

特別に訓練され，認定された，経験豊かな人のみが本制御装置を取扱うことができます。このことは，たとえ短期間であっても，常に当てはまります。

セットアップ，運転，保守ごとに担当者の責任を明確に定義する必要があります。各担当者が責任を正しく果たしているかどうかを監督する必要があります。

アクション

制御装置をインストールしてセットアップする前に，制御装置を取扱う人が指示マニュアルを読み，正しく理解していることを確認する必要があります。制御装置を運転するにあたっては，外から見て分かるような損傷がないか，普段の運転と変わった点がないかどうかなど，全般的な技術上の状態を絶えずチェックする必要があります。

保守

保守担当として特別に訓練された有資格者のみが，メンテナンスガイドに示されている内容に限って修理を行うことができます。修理にあたっては，関連するすべての安全規則が守られなければなりません。

（注）

次に示す使い方は本制御装置の本来の目的から外れており，製造業者の責任の範囲外です：

上記の点に適合しないか，それを超えるようなすべての用法。

制御装置が技術的に完全な状態で運転されたのではない場合，安全注意事項が守られていない場合，あるいは指示マニュアルに示されている指示が守られていない場合。

安全運転に影響するような不良が存在し，それが制御装置のインストールとセットアップの前に是正されていない場合。

制御装置が正しく機能できるため，共通した使い方ができるため，あるいは能動的あるいは受動的な安全が保証されるために必要な制御装置上の装置が変更されたか，ジャンパされたか，シャットダウンされた場合。

十分な危険予知が行われていないと次のことが発生する可能性があります：

- 人身傷害あるいは死亡
- 制御装置，マシンなどの会社およびオペレータの財産の損傷

1 Flexible NC プログラミング

1.1 変数および算術パラメータ

機能

プログラムの柔軟性を向上させるために，固定値に代わって変数を使用することができます。測定値のような信号に応答することや，変数にセットポイントをストアすることによって様々なジオメトリに対して同じプログラムを利用することができます。

上級プログラマならば変数計算とプログラムジャンプを使って，必要とされるプログラミング作業をかなり減らすであろう非常に柔軟なプログラムアーカイブを作成することができます。

変数のタイプ

制御装置は変数を 3 つのタイプに区別します。

ユーザ定義変数	その名前およびタイプをユーザが定義する変数，例えば算術パラメータなど。
算術パラメータ	アドレス R とそれに続く番号を与えられた特別な事前定義算術変数。この事前定義変数は REAL タイプ。
システム変数	制御装置が与える変数で，プログラムで処理（読出し／書込み）可能。システム変数によってゼロオフセットやツールオフセット，実際値，軸に関する測定値，制御ステータスなどへのアクセスが可能となる。（システム変数の意味についてはプログラミング編 基本説明書の付録を参照のこと）

変数のタイプ

タイプ	意味	値の範囲
INT	先頭符号付き整数	± (2^{31} - 1)
REAL	実数（十進小数点付き小数，LONGREAL ～ IEEE）	± (10^{-300} ～ 10^{+300})
BOOL	ブール値 : TRUE (1) および FALSE (0)	1, 0
CHAR	コードで指定した 1 個の ASCII 文字	0 ... 255
STRING	文字列，[...] に文字数， 最大 200 文字	値順序 0 ～ 255
AXIS	軸名（軸アドレス）のみ	チャンネルのすべての軸識別子と主軸
FRAME	並進，回転，スケーリング，ミラーリング用ジオメトリパラメータ，セクション 4 参照。	

算術変数

アドレス R のもとで更なる定義なしに利用できる REAL タイプの算術変数の数は標準で 100 個です。

算術変数の正確な数（最大 1000 個）はマシンデータで定義します。

例 R10=5

システム変数

すべての現行プログラムで利用可能で処理可能なシステム変数を制御装置が提供します。

システム変数はマシンおよび制御ステータスを戻します。

システム変数のなかには値を割り当てることのできないものもあります。

システム変数の名前は常に 文字 "$" とそれに続く特殊な名前で識別されます。

システム変数のタイプ概要

先頭文字	意味
$M	マシンデータ
$S	設定データ
$T	ツール管理データ
$P	プログラム値
$A	現在値
$V	サービスデータ

2 番目文字	意味
N	NCK- 全体
C	チャンネル別
A	軸別

例 $AA_IM

意味：マシン座標系における現在軸別値

1.2 変数の定義

ユーザ定義変数

事前定義変数とは別に，プログラマは自分だけの変数を定義しそれに値を割当てることもできます。

ローカル変数 (LUD) は，それらが定義されているプログラムにおいてのみ有効です。

グローバル変数 (GUD) はすべてのプログラムで適用されます。

メインプログラムで定義されているローカルユーザ変数 (LUD) は，マシンデータでグローバルユーザ変数 (PUD) をプログラムするために定義し直されます。

機械メーカ

機械メーカの仕様を参照のこと

当該変数は，それらがメインプログラムで定義されている場合は，コールされたあらゆるレベルのサブプログラムでも同様に有効です。これらの変数はパートプログラムの開始とともに生成され，パートプログラムの終了あるいは休止とともに削除されます。

例

```
$MN_LUD_EXTENDED_SCOPE=1
PROC MAIN            ; メインプログラム
DEF INT VAR1         ; PUD 定義
...
SUB2                 ; サブプログラムコール
...
M30
PROC SUB2            ; サブプログラムコール SUB2
DEF INT VAR2         ; LUD 定義
...
IF (VAR1==1)         ; PUD 読出し
  VAR1=VAR1+1        ; PUD 読出しと書込み
```

```
  VAR2=1                    ; LUD 書込み ENDIF
SUB3                        ; サブプログラムコール
...
M17
```

マシンデータ $MN_LUD_EXTENDED_SCOPE が設定されている場合，その同じ名前を使ってメインプログラムやサブプログラムで変数を定義することはもうできません。

変数名

変数名は最大 31 個の文字で構成されます。最初の二文字は英字あるいは下線でなければなりません。

文字 "$" はシステム変数用に予約されているので，ユーザ定義変数に使用することはできません。

プログラミング

```
DEF INT name
or DEF INT name=Value
DEF REAL name
or DEF REAL name1,name2=3,name4
or DEF REAL name[array index1,array
index2]
DEF BOOL name
DEF CHAR name
or DEF CHAR name[array index]=("A","B",...)
DEF STRING[string length] name
DEF AXIS name
or DEF AXIS name[array index]
DEF FRAME name
```

変数を定義する際に値が割当てられなければ，システムはその変数をゼロで初期設定します。

変数は使用に先立って，プログラムの開始時に定義しなければなりません。定義は個別のブロックで行います，すなわち 1 ブロック当たり 1 個の変数タイプしか定義できません。

説明

INT	変数タイプ 整数，すなわち整数全体
REAL	変数タイプ 実数，すなわち 10 進小数点付き小数
BOOL	変数タイプ ブール，すなわち 1 または 0 (TRUE または FALSE)
CHAR	変数タイプ 文字，すなわち コード 0 ～ 255 で指定された ASCII 文字
STRING	変数タイプ ストリング，すなわち文字列
AXIS	変数タイプ 軸，すなわち軸アドレスと主軸
FRAME	変数タイプ フレーム，すなわちジオメトリパラメータ
name	変数名

プログラミング例

変数タイプ INT

DEF INT NUMBER	整数タイプの変数が NUMBER の名前で生成される。 システムは当該変数をゼロで初期設定する。
DEF INT NUMBER=7	整数タイプの変数が NUMBER の名前で生成される。 当該変数は値 7 で初期設定される。

変数タイプ REAL

DEF REAL DEPTH	実数タイプの変数が DEPTH の名前で生成される。 システムは当該変数をゼロ (0.0) で初期設定する。
DEF REAL DEPTH=6.25	実数タイプの変数が DEPTH の名前で生成される。 その初期値は 6.25。
DEF REAL DEPTH=3.1,LENGTH=2,QUANTITY	1 行に複数の変数を定義することも可能。

変数タイプ BOOL

DEF BOOL IF_TOOMUCH	ブールタイプの変数が IF_TOOMUCH の名前で生成される。システムは当該変数をゼロ (FALSE) で初期設定する。
DEF BOOL IF_TOOMUCH=1 または DEF BOOL IF_TOOMUCH=TRUE または DEF BOOL IF_TOOMUCH=FALSE	ブールタイプの変数が IF_TOOMUCH の名前で生成される。

変数タイプ CHAR

DEF CHAR GUSTAV_1=65	文字タイプの変数に ASCII 文字用コード割当てることも,
DEF CHAR GUSTAV_1="A"	あるいは直接 ASCII 文字（文字 "A" のコードは 65）を割当てることもできる。

変数タイプ STRING

DEF STRING[6] SAMPLE_1="START"	ストリングタイプの変数は文字列をストアすることができる。最大文字数は変数タイプの後に角括弧で囲まれている。

変数タイプ AXIS

＊ DEF AXIS AXISNAME=(X1)	＊ AXIS タイプの変数は AXISNAME という名前がついていて，チャンネルの軸識別子を持っている，ここでは X1（拡張アドレスを持つ軸名が括弧で囲まれている）。

変数タイプ FRAME

DEF FRAME INCLINE_1	FRAME タイプの変数は INCLINE_1 と呼ばれる。

追加説明

AXIS タイプの変数はチャンネルの軸名と主軸識別子をストアします。

（注）拡張アドレスを持つ軸名は括弧で囲まなければなりません。

ローカル変数に関するプログラミング例

```
DEF INT COUNT
LOOP: G0 X...                          ; ループ
COUNT=COUNT+1
IF COUNTER<50 GOTOB LOOP
M30
```

プログラミング例

現行ジオメトリ軸のスキャン

```
DEF AXIS ABSCISSA;                     ; 第 1 ジオメトリ軸
IF ISAXIS(1)==FALSE GOTOF CONTINUE
ABSCISSA = $P_AXN1
...
CONTINUE:
```

間接主軸プログラミング

```
DEF AXIS SPINDLE
SPINDLE=(S1)
OVRA[SPINDLE]=80                       ; 主軸オーバライド = 80%
SPINDLE=(S3)
....
```

1.3 配列の定義

プログラミング

DEF CHAR NAME[n,m]
DEF INT NAME[n,m]
DEF REAL NAME[n,m]
DEF AXIS NAME[n,m]
DEF FRAME NAME[n,m]
DEF STRING[string length] NAME[m]
DEF BOOL[n,m]

説明

INT NAME[n,m] REAL NAME[n,m]	変数タイプ (CHAR, INTEGER, REAL, AXIS, FRAME, BOOL) n = 第 1 次元の配列サイズ m = 第 2 次元の配列サイズ
DEF STRING[文字列長] NAME[m]	データタイプ STRING は一次元配列でのみ定義できる。
NAME	変数名

CHAR タイプと同様に BOOL タイプにも同じメモリサイズが適用されます。

最大配列サイズはマシンデータで設定されます。

機械メーカ

機械メーカの仕様を参照のこと

タイプ	各配列要素のメモリサイズ
BOOL	1 バイト
CHAR	1 バイト
INT	4 バイト
REAL	8 バイト
STRING	文字列の長さ + 1
FRAME	軸数により最大 400 バイト
AXIS	4 バイト

最大配列サイズによって，変数メモリが管理されるメモリブロックのサイズが決まります。したがって実際に必要とする以上に大きく設定すべきではありません。

標準：812 バイト

大きな配列を定義しない場合は，256 バイトを選択してください。

配列はメモリブロックより大きくすることができます。ブロックサイズの MD 値は，例外的な場合のみ細分化されるような方法で設定すべきです。

標準：256 バイト

要素ごとのメモリ要求：上記参照のこと。

例：

グローバルユーザデータには制御装置のオン／オフ切り替え用の PLC マシンデータを含むべきです（BOOL 配列の定義）。

追加説明

最大 2 次元の配列を定義することができます。

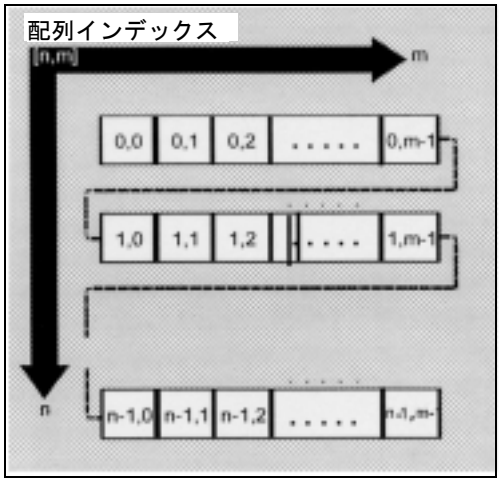
STRING 変数に関する配列は 1 次元のみとしてください。文字列の長さはデータタイプ STRING の後に指定します。

配列インデックス

配列の要素は配列インデックスでアクセスできます。この配列インデックスを使って配列要素を読み出したりあるいは値を割当てたりできます。

最初の配列要素はインデックス [0,0] から始まります。

例えば配列サイズが [3,4] の場合，最大配列インデックスは [2,3] です。



上記の例では，個々の配列要素の順序を図解するために初期設定値は配列要素のインデックスと一致しています。

配列の初期設定

プログラム実行中あるいは配列を定義する際に初期設定値を配列要素に割当てることができます。

2 次元配列に関してはまず右方向の配列インデックスが増加されます。

値リストによる初期設定，SET

1. 配列定義中のオプション

DEF Type VARIABLE = SET(VALUE)

DEF Type ARRAY[n,m] = SET(VALUE, value, ...)

または :

DEF Type VARIABLE = value

DEF Type ARRAY[n,m] = (value, value, ...)

- 割当てられる配列要素の数はプログラムされた初期設定値の数と一致します。
- 値のない配列要素（リストでは空白）には自動的に値 "0" が割当てられます。
- AXIS タイプの変数の値リストには空白があってはなりません。
- 残っている配列要素より多く値がプログラムされると，システムはアラームをトリガします。

例 :

DEF REAL ARRAY[2,3]=(10, 20, 30, 40)

配列を定義する際に SET を自由に指定できます。

2. プログラム実行中のオプション

ARRAY[n,m]= SET(value, value, value,...)

ARRAY[n,m]= SET(expression,

expression, expression,...)

- 配列の定義に際してフィールド要素は先に説明したように初期設定されます。
- ここでは初期設定値として式を用いることもできます。
- 初期設定はプログラムされた配列インデックスから開始されます。選択的に副配列に値を割当てることもできます。

例:

式の割当

DEF INT ARRAY[5, 5]

ARRAY[0,0] = SET(1, 2, 3, 4, 5)

ARRAY[2,3] = SET(VARIABLE, 4*5.6)

軸変数の場合，軸インデックスは処理されません。

例:

1 行での初期設定

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

以下に対応 :

$MA_AX_VELO_LIMIT[1,AX1] = 1.1

$MA_AX_VELO_LIMIT[2,AX1] = 2.2

$MA_AX_VELO_LIMIT[3,AX1] = 3.3.

同じ値での初期設定，REP

1. 配列定義中のオプション

DEF Typ ARRAY[n,m] = REP(value)

配列要素のすべてに同じ値（定数）が割当てられます。

FRAME タイプの変数は初期設定できません。

例：

DEF REAL ARRAY5[10,3] = REP(9.9)

2. プログラム実行中のオプション

ARRAY[n,m] = REP(value)

ARRAY[n,m] = REP(expression)

- ここでは式を初期設定値として用いることもできます。
- 配列要素のすべてが同じ値で初期設定されます。
- 初期設定はプログラムされた配列インデックスから開始されます。選択的に副配列に値を割当てることも可能です。

FRAME タイプの変数にも適用可能で，この方法で非常に簡単に初期設定できます。

例：

1 個の値ですべての要素を初期設定

DEF FRAME FRM[10]

FRM[5] = REP(CTRANS (X,5)).1

プログラミング例

完全な変数配列の初期設定

図は現行割当を示しています。

```
N10 DEF REAL ARRAY1[10,3] = SET(0, 0, 0, 10, 11, 12, 20, 20, 20, 30, 30,
30, 40, 40, 40,)
N20 ARRAY1[0,0] = REP(100)
N30 ARRAY1[5,0] = REP(-100)
N40 ARRAY1[0,0] = SET(0, 1, 2, -10, -11, -12, -20, -20, -20, -30, , , ,
-40, -40, -50, -60, -70)
N50 ARRAY1[8,1] = SET(8.1, 8.2, 9.0, 9.1, 9.2)
```

配列インデックス 2

1

[1,2]	N10: 定義と同時に初期設定				N20/N30: 同じ値で初期設定				N40/N50: 様々な値で初期設定		
	0	1	2		0	1	2		0	1	2
0	0	0	0		100	100	100		0	1	2
1	10	11	12		100	100	100		-10	-11	-12
2	20	20	20		100	100	100		-20	-20	-20
3	30	30	30		100	100	100		-30	0	0
4	40	40	40		100	100	100		0	-40	-40
5	0	0	0		-100	-100	-100		-50	-60	-70
6	0	0	0		-100	-100	-100		-100	-100	-100
7	0	0	0		-100	-100	-100		-100	-100	-100
8	0	0	0		-100	-100	-100		-100	8.1	8.2
9	0	0	0		-100	-100	-100		9.0	9.1	9.2
	配列要素 [5,0] から [9,2] まではデフォルト値 (0.0) で初期設定されている。								配列要素 [3,1] から [4,0] まではデフォルト値 (0.0) で初期設定されている。配列要素 [6,0] から [8,0] までは変更なし。		

1.4 間接プログラミング

間接プログラミングによってプログラムを汎用的に使うことが可能となります。拡張アドレス（インデックス）は適切なタイプの変数で置き換えられます。

以下のものを除き，アドレス構成はすべて可能です :

- N - ブロックナンバ
- G - G コマンド
- L - サブルーチン

設定できるアドレスについては，間接プログラミングはできません。

例 : X[1] を X1 の代用とすることはできません。

プログラミング

ADDRESS[INDEX]

プログラミング例

主軸

S1=300	直接プログラミング
DEF INT SPINU=1 S[SPINU]=300	間接プログラミング : その番号が変数 SPINU （この例では 1）にストアされている主軸の速度 300 rpm

フィード

FA[U]=300	直接プログラミング
DEF AXIS AXVAR2=U FA[AXVAR2]=300	間接プログラミング : そのアドレス名が，変数名 AXVAR2 を持つ AXIS タイプの変数にストアされている位置決め軸のフィード

測定値

$AA_MM[X]	直接プログラミング
DEF AXIS AXVAR3=X $AA_MM[AXVAR3]	間接プログラミング : その名前が変数 AXVAR3 にストアされている軸のマシン座標系での測定値

配列要素

DEF INT ARRAY1[4,5]	直接プログラミング
DEFINE DIM1 AS 4 DEFINE DIM2 AS 5 DEF INT ARRAY[DIM1,DIM2] ARRAY[DIM1-1,DIM2-1]=5	間接プログラミング : フィールドサイズは常に固定値として配列寸法に指定しなければならない。

軸変数を有する軸インストラクション

X1=100 X2=200	直接プログラミング
DEF AXIS AXVAR1 AXVAR2 AXVAR1=(X1) AXVAR2=(X2) AX[AXVAR1]=100 AX[AXVAR2]=2	間接プログラミング : 変数の定義，軸名割当，変数にストアされている軸を 100 および 200 に移動

軸変数を有する補間パラメータ

G2 X100 I20	直接プログラミング
DEF AXIS AXVAR1=X G2 X100 IP[AXVAR1]=20	間接プログラミング : 定義および軸名の割当， 中心点の間接プログラミング

サブプログラムの間接コール

CALL "L" << R10	R10 に入れられている番号でプログラムのコール

追加説明

R パラメータは，省略表記法で 1 次元配列と解釈することもできます（R10 は R[10] に一致）。

1.5 割当

マッチングタイプの値をプログラムで変数／算術パラメータに割当てることができます。

割当は常に個別ブロックで行われます；1 ブロック当たり最大 2 個の割当が可能です。

軸アドレスへの割当（移動動作命令）には必ず変数割当に応じて個別ブロックが必要です。

プログラミング例

R1=10.518 R2=4 VARI1=45 X=47.11 Y=R2	数値の割当
R1=R3 VARI1=R4	マッチングタイプの変数の割当
R4=-R5 R7=-VARI8	反対の先頭符号の割当（INT および REAL タイプに関してのみ可能）

文字列変数への割当

CHAR あるいは STRING の範囲内で，大文字と小文字の区別がなされます。

文字列に 'or" が含まれる場合は，'...' で囲うべきでしょう。

例：

MSG("Viene lavorata l' ''ultima

figura")

テキスト 'Viene lavorata l'ultima figura' を画面に表示します。

表示不能な文字は文字列内に 2 進又は 16 進定数でストアされます。

1.6 算術演算／関数

算術関数は主に R パラメータや REAL タイプの変数（あるいは定数や関数）に用いられます。INT や CHAR タイプにも用いることができます。

算術演算では標準的な表記法が使用されます。実行に関する優先順位は括弧で示されます。三角関数およびその逆関数には角度が指定されます（直角 = 90°）。

演算子／算術関数

+	加算
-	減算
*	乗算
/	除算 注意 : (タイプ INT)/(タイプ INT)=(タイプ REAL); 例 : 3/4 = 0.75
DIV	除算，変数タイプ INT および REAL 用 注意 : (タイプ INT)DIV(タイプ INT)=(タイプ INT); 例 : 3 DIV 4 = 0
MOD	モジュロ除算 (INT または REAL) は INT 除算の余りを生じる， 例えば 3 MOD 4=3
:	チェーン演算子（ FRAME 変数用）
Sin()	サイン
COS()	コサイン
TAN()	タンジェント
ASIN()	アークサイン
ACOS()	アークコサイン
ATAN2(,)	アークタンジェント 2
SQRT()	平方根
ABS()	絶対値
POT()	Z の平方（平方)
TRUNC()	整数に切り捨て
ROUND()	整数に丸め
LN()	自然対数
EXP()	指数関数
CTRANS()	並進
CROT()	回転
CSCALE()	スケール変更
CMIRROR()	ミラーリング

プログラミング例

R1=R1+1	新 R1 = 旧 R1 +1
R1=R2+R3 R4=R5-R6 R7=R8*R9	
R10=R11/R12 R13=SIN(25.3)	
R14=R1*R2+R3	乗算および除算は加算および減算に優先
R14=(R1+R2)*R3	括弧を最初に計算
R15=SQRT(POT(R1)+POT(R2))	内括弧をまず解く R15 = (R1 2 +R2 2) の平方根
RESFRAME= FRAME1:FRAME2 FRAME3=CTRANS(...):CROT(...)	チェーン演算子はフレームを結合して結果のフレームにするか，あるいはフレームコンポーネントに値を割当てる

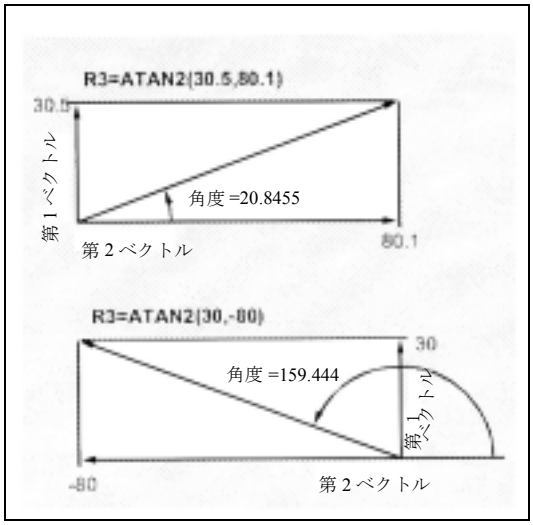
算術関数 ATAN2(,)

当該関数は互いに直角を成す 2 つのベクトルから結果のベクトルの角度を計算します。

結果は四分円の 1 つ (-180 < 0 <+180°) に存在します。角度表示は常に正方向の第 2 の値を基準にしています。

R3=ATAN2(30.5,80.1)

R3=ATAN2(30,-80)



1.7 比較および論理演算子

比較演算子

比較演算子は CHAR，INT，REAL，および BOOL タイプ の変数に用いることができます。コード値は CHAR タイプと比較されます。

次のものは STRING，AXIS，および FRAME タイプに関して可能です : == and <>.

比較演算の結果は常に BOOL タイプです。

比較演算は，例えば，ジャンプ条件を公式化するためなどに用いられます。複雑な式を比較することもできます。

比較演算子の意味

==	等しい
<>	等しくない
>	越える
<	未満
>=	以上
<=	以下
<<	文字列のチェーニング

プログラミング例

IF R10>=100 GOTOF DEST

or

R11=R10>=100

IF R11 GOTOF DEST

比較 R10>=100 の結果はまず R11 にバッファリングされます。

論理演算子

論理演算子は真理値を論理的に結合するのに用いられます。

AND，OR，NOT，および XOR は一般に BOOL タイプの変数に対してのみ用いることができますが，暗黙のタイプ変換によってデータタイプ CHAR，INT，および REAL に対しても用いることができます。

ブールオペランドとブール演算子の間にはスペースを入れなければなりません。

論理（ブール）演算においては，以下のことがデータタイプ BOOL，CHAR，INT および REAL に適用されます :
0 は FALSE に等しい
0 でないということは TRUE に等しい

論理演算子の意味

AND	論理積
OR	論理和
NOT	否定
XOR	排他的論理和

算術式では，すべての演算子に対して実行順序を定義し，その結果として通常の優先順位規則を無効とするために，括弧を使用することができます。

プログラミング例

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

IF NOT R10 GOTOB START

NOT はオペランドにのみ適用されます。

ビット対ビット論理演算子

ビット対ビット論理演算を CHAR および INT タイプの変数に対して行うこともできます。タイプの変換は自動的に行われます。

ビット対ビット論理演算子の意味

B_AND	ビット論理積
B_OR	ビット論理和
B_NOT	ビット否定
B_XOR	ビット排他的論理和

演算子 B_NOT はオペランドにのみ適用されます,
すなわちオペランドはこの演算子の後に来ます。

プログラミング例

IF $MC_RESET_MODE_MASK B_AND ’B10000’ GOTOF ACT_PLANE.1

1.8 演算子の優先順位

演算子の優先順位

各演算子には優先順位が割当てられています。式を評価する際は常に，最も優先順位の高い演算子がまず初めに評価の適用を受けます。演算子の優先順位が同じ場合は，評価は左から右へと行われます。

算術式では，すべての演算子に対して実行順序を定義し，その結果として通常の優先順位規則を無効とするために，括弧を使用することができます。

演算子の順位

（最高優先順位から最低優先順位）

1.	NOT, B_NOT	否定，ビット否定
2.	*, /, DIV, MOD	乗算，除算
3.	+, -	加算，減算
4.	B_AND	ビット論理積
5.	B_XOR	ビット排他的論理和
6.	B_OR	ビット論理和
7.	AND	論理積
8.	XOR	排他的論理和
9.	OR	論理和
10.	<<	文字列のチェーニング，結果タイプ STRING
11.	==, <>, >, <, >=, <=	比較演算子

フレーム用のチェーン演算子 ":" は他の演算子と一緒に式に記述してはなりません。したがってこの演算子には優先順位は必要ありません。

1.9 考えられるタイプ変換

割当時のタイプ変換

ある変数に割当てられている定数値や変数，式などはこの変数のタイプに適合しているはずです。このことが事実とすれば，タイプはその値が割当てられる際に自動的に変換されるということです。

考えられるタイプ変換

へ から	REAL	INT	BOOL	CHAR	STRING	AXIS	FRAME
REAL	可	可 *	可 1)	可 *	-	-	-
INT	可	可	可 1)	可 2)	-	-	-
BOOL	可	可	可	可	可	-	-
CHAR	可	可	可 1)	可	可	-	-
STRING	-	-	可 4)	可 3)	可	-	-
AXIS	-	-	-	-	-	可	-
FRAME	-	-	-	-	-	-	可

* REAL から INT への変換の際，>= 0.5 の小数は切り上げられ，それ以外は切り捨てられる（ROUND 関数と同じ結果）

1) 値が <> 0 であれば TRUE，値が == 0 であれば FALSE

2) 値が許容される値の範囲内にある場合

3) 文字が 1 個のみの場合

4) 文字列の長さ 0 = FALSE，それ以外は TRUE

変換の際に値が最大範囲を超えている場合は，エラーメッセージが出力されます。

追加説明

式のなかに混合タイプが存在する場合は，タイプ変換が自動的に実行されます。

1.10 文字列演算

概要

「割当」や「比較」などの標準的な演算以外に，さらに次のような文字列処理も可能です :

説明

STRING へのタイプ変換 :

STRING_ERG = <<bel._Typ [1)]	結果タイプ : STRING
STRING_ERG = AXSTRING (AXIS)	結果タイプ : STRING

STRING からのタイプ変換 :

BOOL_ERG = ISNUMBER (STRING)	結果タイプ : BOOL
REAL_ERG = NUMBER (STRING)	結果タイプ : REAL
AXIS_ERG = AXNAME (STRING)	結果タイプ : AXIS

文字列のチェーニング :

bel._Typ [1)] << bel. Typ [1)]	結果タイプ : STRING

小文字／大文字への変換 :

STRING_ERG = TOUPPER (STRING)	結果タイプ : STRING
STRING_ERG = TOLOWER (STRING)	結果タイプ : STRING

文字列の長さ :

INT_ERG = STRLEN (STRING)	結果タイプ : INT

文字列内の文字／文字列のサーチ :

INT_ERG = INDEX (STRING, CHAR)	結果タイプ : INT
INT_ERG = RINDEX (STRING, CHAR)	結果タイプ : INT
INT_ERG = MINDEX (STRING, STRING)	結果タイプ : INT
INT_ERG = MATCH (STRING, STRING)	結果タイプ : INT

部分列の選択 :

STRING_ERG = SUBSTR (STRING, INT)	結果タイプ : INT
STRING_ERG = SUBSTR (STRING, INT, INT)	結果タイプ : INT

一文字の選択 :

CHAR_ERG = STRINGVAR [IDX]	結果タイプ : CHAR
CHAR_ERG = STRINGARRAY [IDX_ARRAY, IDX_CHAR]	結果タイプ : CHAR

[1)] "bel._Typ" は変数タイプ INT，REAL，CHAR，STRING および BOOL を表している。

0 文字の特別な意味

0 文字は文字列終了識別子として内部的に解釈されます。

ある文字が 0 文字で置き換えられると，そこでその文字列は短縮されます。

例 :

```
DEF STRING[20] STRG = "Axis .
stationary"
STRG[6] = "X"                          ; メッセージ "Axis Xstationary" を供給

MSG(STRG)
STRG[6] = 0
MSG(STRG)                              ; メッセージ "Axis" を供給
```

1.10.1 タイプ変換

タイプ変換によって，様々なタイプの変数をメッセージ の不可欠な要素として使うことが可能となります (Msg)。

STRING への変換

が結果として行われます，ただし演算子 << がデータタイプ INT，REAL，CHAR および BOOL に対して暗黙の内に使用されることが条件です（「文字列のチェーニング」参照)。

INT 値は通常の読み取り可能な形式に変換されます。REAL 値の場合小数点以下最大 10 位まで指定されます。

AXIS タイプの変数は AXSTRING 関数によって STRING に変換することができます。

FRAME 変数は変換できません。

例 :

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

STRING からの変換

NUMBER 関数が STRING から REAL への変換を行います。

ISNUMBER が値 FALSE を戻した場合は，同じパラメータで NUMBER がコールされた時にアラームが出力されます。

文字列は AXNAME 関数でデータタイプ AXIS に変換することができます。文字列を構成軸識別子に割当てることができない場合はアラームが出力されます。

構文

BOOL_ERG = ISNUMBER (STRING)	結果タイプ : BOOL
REAL_ERG = NUMBER (STRING)	結果タイプ : REAL
STRING_ERG = AXSTRING (AXIS)	結果タイプ : STRING
AXIS_ERG = AXNAME (STRING)	結果タイプ : AXIS

意味 :

ISNUMBER (STRING) は，文字列が意味上有効な REAL 数を表していれば TRUE を戻します。したがって，文字列を有効な数に変換できるかどうかをチェックすることが可能です。

NUMBER (STRING) は文字列で表された値を REAL 値として戻します。

AXSTRING (AXIS) は指定軸識別子を文字列で与えます。

AXNAME (STRING) は指定文字列を軸識別子に変換します。

例

DEF BOOL BOOL_ERG	
DEF REAL REAL_ERG	
DEF AXIS AXIS_ERG	
DEF STRING[32] STRING_ERG	
BOOL_ERG = ISNUMBER ("1234.9876Ex-7")	; ここで : BOOL_ERG == TRUE
BOOL_ERG = ISNUMBER ("1234XYZ")	; ここで : BOOL_ERG == FALSE
REAL_ERG = NUMBER ("1234.9876Ex-7")	; ここで : REAL_ERG == 1234.9876Ex-7
STRING_ERG = AXSTRING(X)	; ここで : STRING_ERG == "X"
AXIS_ERG = AXNAME("X")	; ここで : AXIS_ERG == X

1.10.2 文字列のチェーニング

この機能性によって，文字列を個々の構成要素とともにコンパイルすることが可能となります。チェーニング関数は演算子 : << で実現されます。この演算子は，基本タイプ CHAR，BOOL，INT，REAL および STRING の組合わせのすべてに対して STRING を目標タイプとしています。必要な変換はすべて現行の規則にしたがって実行されます。

FRAME および AXIS タイプはこの演算子では使用できません。

構文 :

bel._Typ << bel._Typ	結果タイプ : STRING

意味 :

指定文字列（場合によっては暗黙の内に変換された他のタイプ）が連鎖されます。

この演算子は単項のバリアントとしても利用することができます。したがって，STRING （ FRAME や AXIS ではなく）への明示的なタイプ変換を実行することが可能です。

構文 :

<< bel._Typ	結果タイプ : STRING

意味 :

指定タイプが暗黙の内に STRING タイプに変換されます。

例えば，この関数はテキストリストからメッセージやコマンドをコンパイルしてパラメータ（モジュール名など）を挿入するために使用することができます :

MSG(STRG_TAB[LOAD_IDX]<<MODULE_NAME)

文字列チェーニングの中間結果は最大文字列の長さを越えてはなりません。

プログラミング例

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

1.10.3 小文字／大文字への変換

この機能性を使って文字列内のすべての文字を同じ型の活字に変換することができます。

構文 :

STRING_ERG = TOUPPER	(STRING)	結果タイプ : STRING
STRING_ERG = TOLOWER	(STRING)	結果タイプ : STRING

意味 :

小文字はすべて大文字かあるいは小文字のいずれかに変換されます。

例 :

ユーザ入力も同様に MMC で起動できるため，テキストを同型表示することが可能です（すなわち大文字表示あるいは小文字表示）:

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

1.10.4 文字列の長さ

この機能性により文字列の長さを指定することができます。

構文 :

INT_ERG = STRLEN (STRING)	結果タイプ : INT

意味 :

0 文字ではない文字の数 ー 文字列の先頭から数える ー が戻されます。

例 :

この関数は，例えば下記のような一文字アクセスと組合わせて，文字列の終わりを決定するのに使うことができます :

IF(STRLEN (MODULE_NAME) > 10) GOTOF ERROR

1.10.5 文字列内の文字／文字列のサーチ

単一の文字や別の文字列内にある完全な文字列をサーチするのにこの機能性を使用することができます。関数の結果はその文字／文字列がサーチした文字列内のどこに位置しているかを示します。

INT_ERG = INDEX	(STRING,CHAR)	結果タイプ : INT
INT_ERG = RINDEX	(STRING,CHAR)	結果タイプ : INT
INT_ERG = MINDEX	(STRING,STRING)	結果タイプ : INT
INT_ERG = MATCH	(STRING,STRING)	結果タイプ : INT

意味 :

サーチ関数 : サーチが成功した場合は，その文字列（最初のパラメータ）内の位置を戻します。文字／文字列を見つけることができなければ，値 "-1" を戻します。この場合，最初の文字は位置 0 にあります。

INDEX	最初のパラメータの先頭からサーチして二番目のパラメータで指定されている文字を見つけ出す。
RINDEX	最初のパラメータの最後からサーチして二番目のパラメータで指定されている文字を見つけ出す。
MINDEX	文字リストが（文字列として）転送されるという点を除けば INDEX 関数と同じ。このリストで発見された最初の文字のインデックスが戻される。
MATCH	文字列内の文字列を捜し出す。

文字列はしたがって一定の基準に従って，すなわち
ブランクあるいはパス分離記号（スラッシュ）("/")
のある位置で分割されます。

プログラミング例

入力をパスとモジュール名に分割する方法 :

DEF INT PFADIDX, PROGIDX	
DEF STRING[26] INPUT	
DEF INT LISTIDX	
INPUT = "/_N_MPF_DIR/_N_EXECUTE_MPF"	
LISTIDX = MINDEX (INPUT, "M,N,O,P") + 1	LISTIDX の値として 3 が戻される : "N" がパラメータ INPUT 内の最初の文字のため（選択リストの先頭から開始)。
PFADIDX = INDEX (INPUT, "/") +1	; PFADIDX はしたがって 1
PROGIDX = RINDEX (INPUT, "/") +1	; PROGIDX = はしたがって 12
	; 次のセクションで紹介する SUBSTR 関数を使うことによって，変数 INPUT を「パス」と「モジュール」に分割できる :
VARIABLE = SUBSTR (INPUT, PFADIDX, PROGIDX-PFADIDX-1)	そこで "_N_MPF_DIR" を供給
VARIABLE = SUBSTR (INPUT, PROGIDX)	そこで "_N_EXECUTE_MPF" を供給

1.10.6 部分列の選択

当該機能性により，文字列から部分列を分離することができます。このために，先頭文字のインデックスと望ましい文字列の長さ（もし必要なら）を指定します。長さに関する情報が指定されなければ，その文字列データは残りの文字列に適用されます。

STRING_ERG = SUBSTR	(STRING,INT)	結果タイプ : INT
STRING_ERG = SUBSTR	(STRING,INT, INT)	結果タイプ : INT

意味 :

第一のケースでは，最初のパラメータで定義される位置からその文字列の終わりまでの部分列が戻されます。

第二のケースでは，結果文字列は 3 番目のパラメータで定義されている最大の長さに限定されます。

開始位置が文字列の終わりより後ろの場合は，空の文字列 ("") が戻されます。

開始位置あるいは長さが負の場合は，アラームが出力されます。

例 :

```
DEF STRING [29] ERG
ERG = SUBSTR ("ACKNOWLEDGMENT: 10 to 99", 10, 2)                ; ERG はしたがって == "10"
10, 2)
```

1.10.7 一文字選択

この関数を使って文字列の個々の文字を選択することができます。この関数は読取りアクセスと書込みアクセスの両方の操作に適用されます。

構文 :

CHAR_ERG = STRINGVAR [IDX]	結果タイプ : CHAR
CHAR_ERG = STRINGARRAY [IDX_ARRAY, IDX_CHAR]	結果タイプ : CHAR

意味 :

文字列内の指定位置にある文字が読取り／書込みされます。位置パラメータが負あるいは文字列より大きい場合はアラームが出力されます。

メッセージの例 :

事前アセンブルされている文字列に軸識別子を挿入

```
DEF STRING [50] MESSAGE = "Axis n has
reached position"
MESSAGE [6] = "X"
MSG (MESSAGE) ;                    メッセージ "Axis X has reached position" を戻す
```

ユーザが定義をした変数（ LUD，GUD および PUD データ）内の一文字にアクセスすることができるだけです。また，サブルーチンコールでデータにアクセスするこの方式は "Call-by-value" （値呼び）タイプのパラメータに対してのみ使用できます。

例 :

システムデータ内の一文字にアクセスする方法，

マシンデータ，...:

```
DEF STRING [50] STRG
DEF CHAR ACKNOWLEDGMENT
...
STRG = $P_MMCA
ACKNOWLEDGMENT = STRG [0]                    ; 肯定応答構成要素の評価
```

参照呼び (call-by-reference) で一文字にアクセスする方法

パラメータ :

```
DEF STRING [50] STRG
DEF CHAR CHR1
EXTERN UP_CALL (VAR CHAR1)                   ; 参照呼びパラメータ！
...
CHR = STRG [5]
UP_CALL (CHR1)                               ; 参照呼び
STRG [5] = CHR1
```

1.11 CASE インストラクション

プログラミング

CASE (expression) OF constant1 GOTOF LABEL1 ... DEFAULT GOTOF LABELn
CASE (expression) OF constant1 GOTOB LABEL1 ... DEFAULT GOTOB LABELn

コマンドの説明

CASE	ジャンプ命令のためのボキャブラリワード
GOTOF	宛先付きジャンプ命令，前方向（プログラムの終わり方向へ）
GOTOB	宛先付きジャンプ命令，後ろ方向（プログラムの開始方向へ）
LABEL	宛先（プログラム内のラベル）;
LABEL:	ジャンプ宛先名の後にコロンを入れる
Expression	算術式
Constant	INT タイプの定数
DEFAULT	プログラムパス，ただし前もって名前を付けられた定数がなにも適用されていない場合

機能

この CASE ステートメントによって様々なブランチを INT タイプの値にしたがって実行することが可能となります。

動作

プログラムは，CASE ステートメントで評価される定数の値に応じて，宛先で指定された点までジャンプします。

この定数が事前定義された値のどれとも一致しない場合は，DEFAULT 命令を使ってジャンプの宛先を決めることができます。

DEFAULT 命令がプログラムされていない場合は，CASE ステートメントに続くブロックがジャンプの宛先です。

プログラミング例

例 1

```
CASE（式）OF 1 GOTOF LABEL1 2 GOTOF LABEL2 ... DEFAULT GOTOF
LABELn
"1" および "2" が可能性のある定数です。
式の値 = 1 （INT 定数）ならば，LABEL1 のブロックへジャンプ
式の値 = 2 （INT 定数）ならば，LABEL2 のブロックへジャンプ
...
それ以外は LABELn のブロックへジャンプ
```

例 2

```
DEF INT VAR1 VAR2 VAR3
CASE(VAR1+VAR2-VAR3) OF 7 GOTOF LABEL1 9 GOTOF LABEL2 DEFAULT GOTOF LABEL3
LABEL1: G0 X1 Y1
LABEL2: G0 X2 Y2
LABEL3: G0 X3 Y3.1
```

1.12 制御構造

説明

IF-ELSE-ENDIF	2 つの選択肢からいずれかを選択
LOOP-ENDLOOP	エンドレスループ
FOR-ENDFOR	カウントループ
WHILE-ENDWHILE	ループの開始点で条件付きループ
REPEAT-UNTIL	ループの終了点で条件付きループ

機能

制御装置は標準としてプログラム順に NC ブロックを処理します。

本セクションで説明するプログラムブランチの他に，これらのコマンドを使って追加の選択肢やプログラムループを定義することができます。

これらのコマンドによってユーザは構成の優れた読みやすいプログラムを作成することが可能となります。

動作

1. IF-ELSE-ENDIF

IF-ELSE-ENDIF ブロックを用いて二つの選択肢の内の一つを選択します :

IF （式）

NC ブロック

ELSE

NC ブロック

ENDIF

この式の値が TRUE，すなわち条件が満たされると，次のプログラムブロックが実行されます。条件が満たされなければ，ELSE プログラムブランチが実行されます。この ELSE ブランチは省略可能です。

2. エンドレスプログラムループ LOOP

エンドレスループはエンドレスプログラムで使用されます。ループの終わりには必ず開始に戻るブランチがあります。

LOOP

NC ブロック

ENDLOOP

3. カウントループ FOR

FOR ループは決められた実行数だけ操作を繰り返す必要がある場合に使用します。この場合，カウント変数は開始値から終了値まで増加します。開始値は終了値より小さくなければなりません。この変数は INT タイプです。

FOR 変数 = 開始値 TO 終了値

NC ブロック

ENDFOR

4. ループ開始点で条件付きプログラムループ WHILE

WHILE プログラムループは条件が満たされている限り実行されます。

WHILE 式

NC ブロック

ENDWHILE

5. ループ終了点で条件付きプログラムループ
REPEAT

REPEAT ループは一回実行された後，条件が満たされるまで連続して繰り返されます。

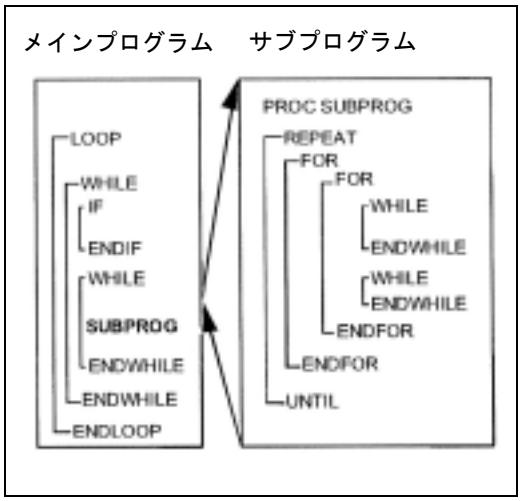
REPEAT

NC ブロック

UNTIL （式）

ネスティングの深さ

チェック構造体はプログラム内でローカルに適用されます。最大 8 個のチェック構造体からなるネスティングの深さを各サブプログラムレベルでセットアップすることができます。



ランタイム応答

インタプリタモード（標準として有効）の場合，プログラムブランチを使って，チェック構造体で達成できるよりももっと効果的にプログラム処理時間を短縮することが可能です。

プリコンパイルサイクルに関してはプログラムブランチとチェック構造体に差はありません。

補足条件

チェック構造体要素を有するブロックは抑制することができません。このタイプのブロックではラベルを使用してはいけません。

チェック構造体はインタプリティブに処理されます。ループの終わりが検出されると，このプロセスでチェック構造体が見出されることを見込んで，ループの開始を見つけ出すためにサーチが行われます。

このためインタプリタモードでは，プログラムのブロック構造は完全にはチェックされません。

チェック構造体とプログラムブランチを混ぜて使うのは，一般的に賢明とは言えません。

サイクルを事前処理する際に，チェック構造体が正しくネストされていることを確認するためにチェックを行うことができます。

チェック構造体はプログラムのステートメントセクション内にのみ挿入できます。プログラムヘッダ内の定義は条件付きで実行されないことや，たびたび実行されないことがあります。

チェック構造体のボキャブラリワードやブランチ宛先にマクロを重ねることは許されません。マクロが定義されると当該チェックは実行されません。

プログラミング例

1. エンドレスプログラム

```
%_N_LOOP_MPF
LOOP
    IF NOT $P_SEARCH                    ; ブロックサーチなし
        G01 G90 X0 Z10 F1000
        WHILE $AA_IM[X] <= 100
            G1 G91 X10 F500             ; 穴あけ加工パターン
            Z-5 F100
            Z5
        ENDWHILE
```

```
            Z10
    ELSE                                    ; ブロックサーチ
MSG("No drilling during block search")
    ENDIF
    $A_OUT[1] = 1                           ; 次の穴あけ加工プレート
    G4 F2
ENDLOOP
M30
```

2. 固定量のパーツの製造

```
%_N_WKPCCOUNT_MPF
DEF INT WKPCCOUNT
FOR WKPCCOUNT = 0 TO 100
G01 ...
ENDFOR
M30
```

1.13 プログラムの協調

チャンネル

チャンネルは他のチャンネルとは独立して自らのプログラムを処理することができます。チャンネルはプログラムによって自分に一時的に割当てられた軸や主軸を制御することができます。

スタートアップ中に制御装置に対して 2 つ以上のチャネルをセットアップすることができます。

プログラムの協調

ワークピースの加工に複数のチャンネルが関与している場合は，プログラムを同期させる必要があるかもしれません。

プログラムの協調には特別な命令（コマンド）があります。各命令はブロックに個別にプログラムします。

プログラム協調用命令

• 絶対パスでの指定	
INIT (n,"_HUGO_DIR_N_ name_MPF") または INIT (n,"_N_MPF_DIR_N_ name_MPF")	絶対パスは以下の規則に従ってプログラムされる : • Current directory_N_ name_MPF "current directory" （現在ディレクトリ）は選択されたワークピースディレクトリ，もしくは標準ディレクトリ _N_MPF_DIR を表す。特定のチャンネルで実行される • 特定のプログラムを選択 : n: そのチャンネルの番号，制御構成に応じた値 • 完全なプログラム名
• 相対パス指定 例 : INIT(2,"DRESS")	プログラムコールに関しては同じ規則が相対パス定義に適用される。
INIT(3,"SUB_1_SPF")	サブプログラムコールで "_SPF" をプログラム名に付加しなければならない。
START (n,n)	その他のチャンネルで選択されたプログラムを開始。 n,n,: そのチャンネルの番号：値は制御構成による
WAITM (Marker No.,n,n,...)	同じチャンネルにマーカ "Marker No.（マーカ番号）" を設定する。正確なストップで前のブロックを終了。指定されたチャンネル "n" で同じ "Marker no." を持つマーカを待つ（現在チャンネルを指定する必要はない）。同期後マーカを削除。 チャンネル当たり 10 個のマーカを同時に設定可能。

WAITMC(Marker No., n, n, ...)	同じチャンネルにマーカ "Marker No.（マーカ番号）" を設定。その他のチャンネルがまだマーカに達していない場合のみ正確なストップが起動される。指定されたチャンネル "n" で同じ "Marker no." を持つマーカを待つ（現在チャンネルを指定する必要はない）。指定されたチャンネルでマーカ "Marker N" に達すると，正確なストップを終了することなしに続行。
WAITE (n,n)	指定されたチャンネルのプログラムの終わりを待つ（現在チャンネル指定せず）
SETM(Marker No., Marker No., ...)	現在の処理に影響を及ぼすことなく同じチャンネルにマーカ "Marker No." を設定。SETM() は RESET および NC START 後も有効状態維持。SETM() はまたシンクロナイズドアクションと関係なくプログラム可能。
CLEARM(Marker No., Marker No., ...)	現在の処理に影響を及ぼすことなく同じチャンネル内のマーカ "Marker No." を削除。マーカはすべて CLEARM() で削除可能。CLEARM (0) はマーカ "0" を削除。CLEARM() は RESET および NC START 後も有効状態維持。CLEARM() はまたシンクロナイズドアクションと関係なくプログラム可能。

（注）上記のコマンドはすべて個別のブロックにプログラムしなければなりません。

チャンネル名

チャンネル名は変数で番号に変換しなければなりません（セクション 1.1 「変数および算術パラメータ」を参照のこと）。

この番号割当は不用意に変更されないように保護してください。

例 :

"MACHINE" と呼ばれるチャンネルにはチャンネル番号 1 が入ることになっています,

"LOADER" と呼ばれるチャンネルにはチャンネル番号 2 が入ることになっています,

DEF INT MACHINE=1, LOADER=2

これらの変数にはチャンネルと同じ名前が当てられます。

命令 START はしたがって :

START(MACHINE)

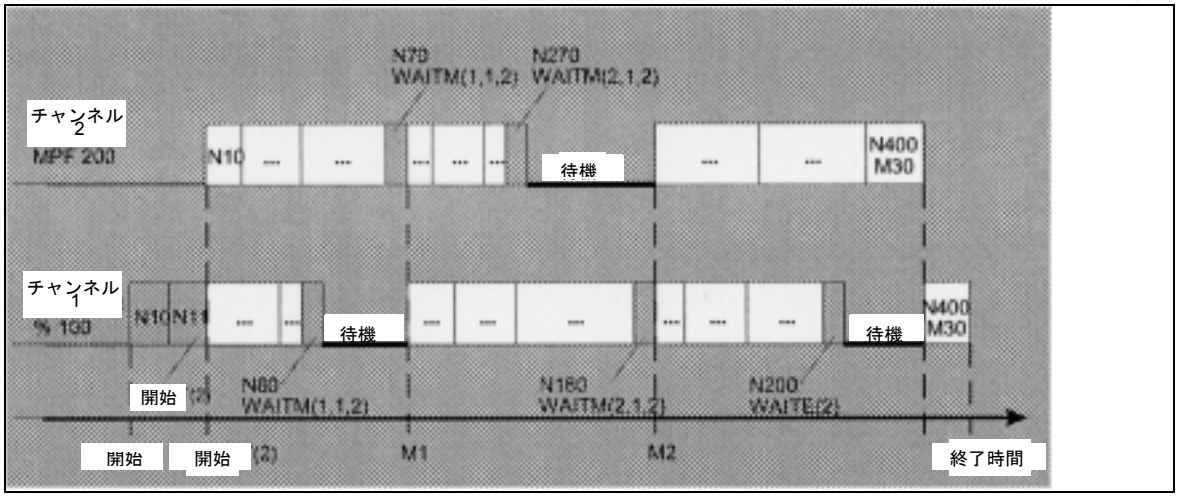
プログラム協調の例

チャンネル 1:

%_N_MPF100_MPF	
N10 INIT(2,"MPF200")	
N11 START (2) .	チャンネル 2 でプログラム実行
N80 WAITM(1,1,2)	チャンネル 1 およびチャンネル 2 で WAIT マーク 1 を待ち，チャンネル 1 で実行継続
N180 WAITM(2,1,2) .	チャンネル 1 およびチャンネル 2 で WAIT マーク 2 を待ち，チャンネル 1 で実行継続
N200 WAITE(2)	チャンネル 2 でプログラムの終わりを待つ
N201 M30 ...	チャンネル 1 でプログラム終了，すべてを終了

Channel 2:

%_N_MPF200_MPF	
;$PATH=/_N_MPF_DIR	
N70 WAITM(1,1,2) .	チャンネル 2 でプログラム実行
チャンネル 1 およびチャンネル 2 で WAIT マーク 1 を待ち，チャンネル 1 で実行継続	
N270 WAITM(2,1,2) .	チャンネル 1 およびチャンネル 2 で WAIT マーク 2 を待ち，チャンネル 2 で実行継続
N400 M30	チャンネル 2 でプログラムの終了



ワークピースからのプログラム例

```
N10 INIT(2,"/_N_WKS_DIR/_N_SHAFT1_WPD/_N_STOKREM1_MPF")
```

相対パス定義を使った Init コマンドの例

```
; プログラム /_N_MPF_DIR/_N_MAIN_MPF がチャンネル 1 で選択される
N10 INIT(2,"MYPROG") ; プログラム /_N_MPF_DIR/_N_MYPROG_MPF をチャンネル 2 で選択。
```

追加説明

すべてのチャンネルがアクセスできる変数（NCK 固有グローバル変数）をプログラム間のデータ交換に使うことができます。それ以外はチャンネルごとに個別のプログラムを書かねばなりません。

1.14 割込みルーチン

プログラミング

SETINT(3) PRIO=1 NAME

SETINT(3) PRIO=1 LIFTFAST

SETINT(3) PRIO=1 NAME LIFTFAST

G... X... Y... ALF=...

DISABLE(3)

ENABLE(3)

CLRINT(3)

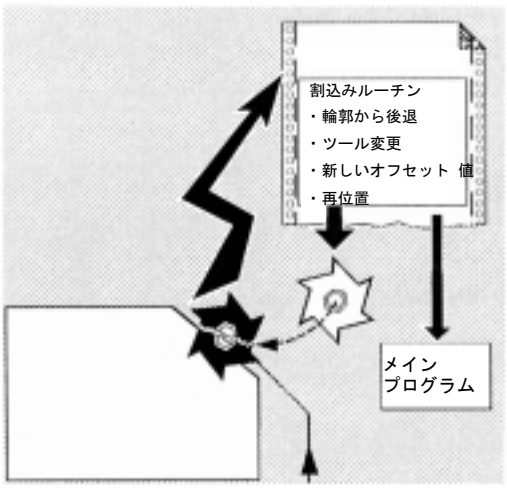
コマンドの説明

SETINT(n)	入力 n がイネーブルされると割込みルーチンを開始，n (1 〜 8) は 7 個の入力の内の 1 つの番号
PRIO=1	優先順位 1 から 128 までを定義（1 が最高優先順位）
LIFTFAST	輪郭からの急上昇
NAME	実行すべきサブプログラムの名前
ALF=...	プログラム可能な移動方向（モーションブロックで）
DISABLE(n)	割込みルーチン番号 n を停止
ENABLE(n)	割込みルーチン番号 n を再起動
CLRINT(n)	割込みルーチン番号 n の割込み割当てクリア

機能

例 : ツールが加工中に故障します。これによって現在の加工プロセスを停止させる信号がトリガされ，同時にサブプログラムが開始されます - このサブプログラムが割込みルーチンと呼ばれます。当該割込みルーチンにはこの場合に実行されるべきすべてのインストラクションが含まれます。

割込みルーチンを実行し終えて，マシンが運転を続行する用意ができると，制御はメインプログラムまでジャンプして戻り，REPOS コマンドに依存しながら，中断点から加工を続行します。



REPOS の詳細については，セクション 9，パストラベル挙動，再位置決めを参照してください。

動作

サブプログラムとして割込みルーチンを作成

割込みルーチンは定義の際にサブプログラムとして識別します。

例 :

PROC ABHEB_Z

N10...

N50 M17

プログラム名 ABHEB_Z の後に NC ブロックが続き，最後にエンドオブプログラム M17 が来てメインプログラムに戻ります。

（注）SETINT インストラクションは割込みルーチン内にプログラムすることができ，追加の割込みルーチンを起動するのに利用できます。追加割込みルーチンはこの入力でトリガされます。

サブプログラム作成方法の詳細についてはセクション 2 に記載しています。

中断点の保存，SAVE

割込みルーチンは定義の際に SAVE で識別することができます。

例 :

PROC ABHEB_Z SAVE

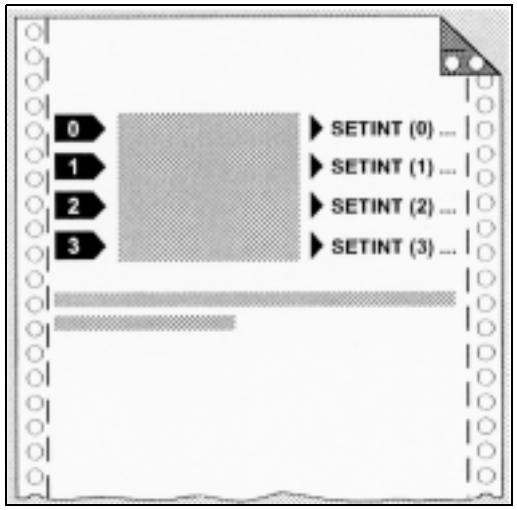
N10...

N50 M17

割込みルーチンの終わりでモダル G 関数は，SAVE の属性によって，それらが割込みルーチンの開始時に与えられていた値に設定されます。設定可能なゼロオフセット（モダル G 関数グループ 8）に加えてプログラム可能なゼロオフセットと基本オフセットも回復されます。G 関数グループ 15 （フィードタイプ）が，例えば，G94 から G95 に変更された場合は，適切な F 値も同様に回復されます。

それゆえ加工動作は後で中断点から再開できるのです。

割込みルーチンの割当と開始，SETINT



制御装置はプログラムの実行を中断し対応する割込みルーチンを開始するための信号を 8 個（入力 1 ～ 8）用意しています。

プログラムへの入力の割当はメインプログラムで行われます。

例：

N10 SETINT(3) PRIO=1 ABHEB_Z

入力 3 がイネーブルされると，ルーチン ABHEB_Z が直ちに開始されます。

複数の割込みルーチンの開始と優先順位の定義，PRIO=

複数の SETINT 命令を自分の NC プログラムにプログラムする場合，したがって同時に複数の信号が発生する可能性がある場合は，割込みルーチンの実行順序を決めるために，それらの優先順位を割当てなければなりません : 優先順位 PRIO 1 から 128 までが有効で，1 が最高優先順位です。

例：

N10 SETINT(3) PRIO=1 ABHEB_Z

N20 SETINT(2) PRIO=2 ABHEB_X

当該ルーチンは，入力が同時にイネーブルされると優先順位順に連続して実行されます。まず SETINT(3)，次に SETINT(2)。

割込みルーチン実行中に新たな信号が受信された場合は，現在の割込みルーチンはより優先順位の高いルーチンによって割込まれます。

割込みルーチンの停止／再起動,
DISABLE, ENABLE

NC プログラムの割込みルーチンを DISABLE(n) を使って停止させたり，ENABLE(n) を使って再起動させたりできます（n は入力番号を表す）。

当該入力／ルーチン割当ては DISABLE と同時に保存され，ENABLE と同時に再起動されます。

割込みルーチンの再割当て

すでに割当て済みの入力に新たなルーチンが割当てられると，前の割当ては自動的に取り消されます。

例:

```
N20 SETINT(3) PRIO=2 ABHEB_Z
...
...
N120 SETINT(3) PRIO=1 ABHEB_X
```

割当てのクリア，CLRINT

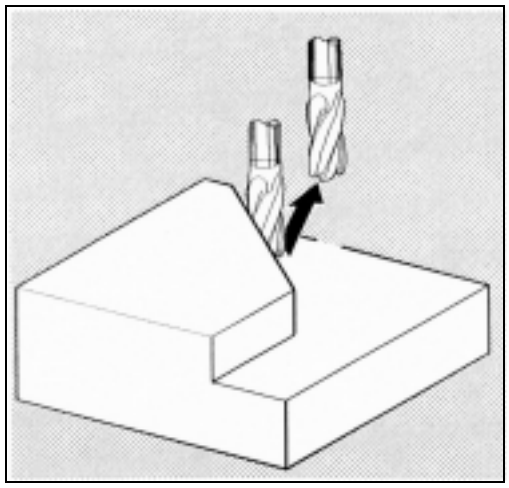
割当ては CLRINT(n) でクリアできます。

例:

```
N20 SETINT(3) PRIO=2 ABHEB_Z
N50 CLRINT(3)
```

入力 3 とルーチン ABHEB_Z の割当てはクリアされます。

輪郭からの急上昇，LIFTFAST



入力が切り換わると，LIFTFAST はワークピースの輪郭からツールを急いで後退させます。

SETINT 命令に LIFTFAST だけでなく割込みルーチンも含まれている場合は，割込みルーチンの前に liftfast が実行されます。

例：

N10 SETINT(2) PRIO=1 LIFTFAST

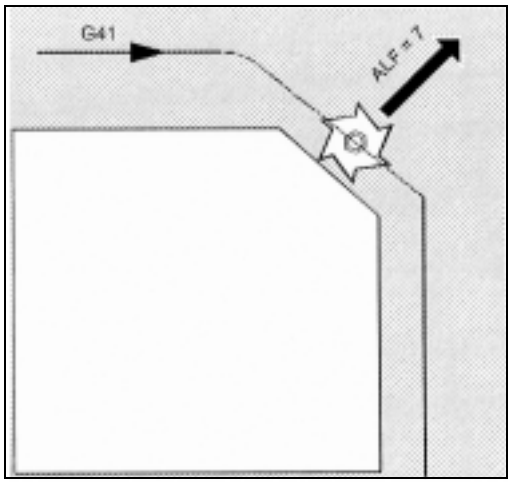
あるいは

N30 SETINT(2) PRIO=1 ABHEB_Z LIFTFAST

いずれの場合も，最高優先順位の入力 2 がイネーブルされると，liftfast が実行されます。

- N10 の場合，アラーム 16010 と同時に実行が停止されます（非同期サブプログラム ASUP が何も指定されていないため）。
- N30 の場合，非同期サブプログラム "ABHEB-Z" が実行されます。

急上昇での動作順序



liftfast と同時に輪郭からジオメトリ軸が後退するその距離をマシンデータで定義することができます。

プログラム可能な移動動作方向，ALF=...

liftfast と同時にツールが移動する方向を NC プログラムに入力します。

可能な移動動作方向は特別なコード番号で制御装置にストアされ，この番号で呼び出すことができます。

例：

N10 SETINT(2) PRIO=1 ABHEB_Z LIFTFAST

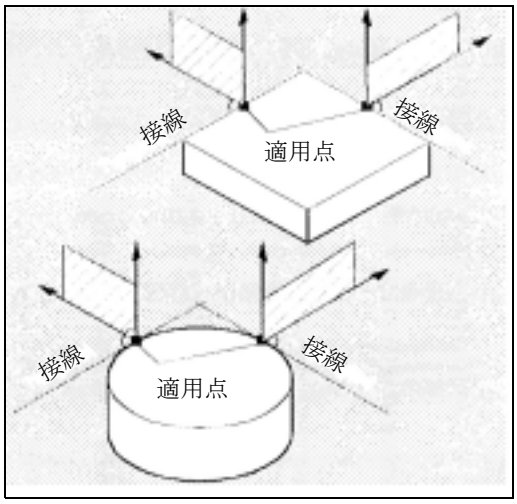
ALF=7

ツールは - G41 起動状態で（輪郭の左側へ加工方向）- 上から見て垂直に輪郭から離れていきます。

移動動作方向を説明する際の基準面

プログラムされた輪郭に対するツール適用点で，対応するコード番号で lift-off 動作を指定するために基準として用いられる面にツールがクランプされます。

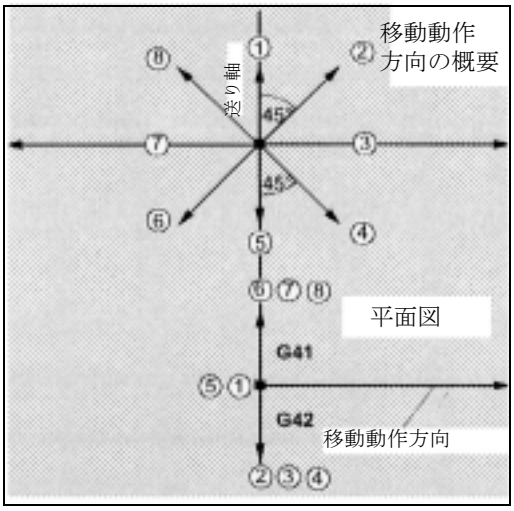
この基準面はツールの適用点において，縦のツール軸（インフィード方向）とこの軸に対して直角に位置し，さらにはその接線に対して直角に位置するベクトルから導き出されます。



移動動作方向に関するコード番号，概要

コード番号と基準面に関する移動動作方向を右の図に示します。

ALF=0 で liftfast 機機能を起動します。



（注）ツール径補正が有効の場合，以下のコードを使うべきではありません：
G41 と同時にコード 2，3，4
G42 と同時にコード 6，7，8

これらの場合，ツールは輪郭に接近してワークピースと衝突するかもしれません。

後退動作の方向は，変数 ALF を有する G コード LFTXT または LFWP でプログラムされます。

- LFTXT

後退動作の面はパスタンジェントとツール方向から決定されます。この G コード（デフォルト設定）は目下のところ急上昇動作の場合の挙動をプログラムするために使用されます。

- LFWP

後退動作の面は，G コード G17，G18，あるいは G19 で選択されるアクティブな作業面です。後退動作の方向はパスタンジェントには左右されません。したがって軸に平行な急上昇動作をプログラムすることが可能です。

後退動作面に関し，以前はあった 45 度の個別ステップにおける方向をプログラムするために ALF が用いられます。LFTXT の場合，ツール方向の後退は ALF=1 として定義されます。LFWP の場合，作業面における方向は以下に従っています :

- G17: X/Y 面　　ALF=1 X 方向に後退
  ALF=3 Y 方向に後退
- G18: Z/X 面　　ALF=1 Z 方向に後退
  ALF=3 X 方向に後退
- G19: Y/Z 面　　ALF=1 Y 方向に後退
  ALF=3 Z 方向に後退

プログラミング例

この例では，壊れたツールは自動的に代わりのツールと交換されることになっています。加工は新しいツールで続行することになっています。したがって加工は交換後新しいツールで続行されます。

メインプログラム

N10 SETINT(1) PRIO=1 W_WECHS -> -> LIFTFAST	入力 1 がイネーブルされると，ツールは自動的に liftfast （ツール半径補正 G41 の場合コード番号 7）で輪郭から後退する。割込みルーチン W_WECHS が次に実行される。
N20 G0 Z100 G17 T1 ALF=7 D1	
N30 G0 X-5 Y-22 Z2 M3 S300	
N40 Z-7	
N50 G41 G1 X16 Y16 F200	
N60 Y35	
N70 X53 Y65	
N90 X71.5 Y16	
N100 X16	
N110 G40 G0 Z100 M30	

サブプログラム

PROC W_WECHS SAVE	現在の動作状態の保存に関するサブプログラム
N10 G0 Z100 M5	ツール交換位置，主軸停止
N20 T11 M6 D1 G41	ツールの交換
N30 REPOSL RMB M3	再位置決めとメインプログラムへの復帰

-> 単一ブロックでプログラム化

REPOS コマンドをサブプログラムにプログラムしなければ，軸は割込まれたブロックの後に続くブロックの終わりの方へと移動します。

1.15 軸転送，主軸転送

コマンドの説明

RELEASE（axis name, axis name, ...）	軸をイネーブル
GET（axis name, axis name, ...）	軸受取り
GETD（axis name, axis name, ...）	軸の直接受取り
Axis name	システムに軸割当て : AX1，AX2，... または マシン軸名を指定
RELEASE(S1)	主軸 S1，S2，... をイネーブル
GET(S2)	主軸 S1，S2，... を受取り
GETD(S3)	主軸 S1，S2，... の直接受取り

機能

1 つ以上の軸あるいは主軸は常に 1 つのチャンネルでのみ使用することができます。1 つの軸が 2 つの異なるチャンネル間で交互しなければならない場合（例えばパレットチェンジャなど）は，まず現在のチャンネルでイネーブルして，それからもう一方のチャンネルに転送しなければなりません :

軸はチャンネルからチャンネルへと転送されます。

動作

軸転送に関する前提条件

- 当該軸はマシンデータですべてのチャンネルに定義しなければなりません。
- POWER ON の後で軸が割当てられるチャンネルは軸別マシンデータで定義されます。

軸の開放 : RELEASE

軸をイネーブルする際は，以下のことに注意してください :

1. 当該軸は変換に関与してはならない。
2. 軸リンク（タンジェンシャル制御，カップリングされた動作）に関与している軸はすべてイネーブルされねばならない。
3. 現在の位置決め軸は転送してはならない。
4. ガントリマスタ軸の従属軸はすべてマスタによって転送される。

軸の転送：GET

このコマンドで実際の軸転送が実行されます。このコマンドがプログラムされているチャンネルは軸に対して全責任を負います。

GET の結果：

同期での軸転送：

軸は，GET の前に別のチャンネルに割当てられているかあるいはとりあえず PLC に割当てられていて，なお且つ "WAITP"，G74 あるいは移動距離削除で再同期が取られていない場合は，必ず同期をとらなければなりません。

- 前処理停止が次に引き起こされる（STOPRE に関し）
- 転送が完了するまで実行が中断される。

同期なしで軸転送：

軸が同期を取る必要がない場合は，GET によって前処理停止が引き起こされることはありません。

例：

```
N01 G0 X0
N02 RELEASE(AX5)
N03 G64 X10
N04 X20
N05 GET(AX5)
N06 G01 F5000
N07 X20
N08 X30
N09 ...
```

同期を取る必要がない場合，これは実行可能なブロックではない。

実行可能なブロックではない。

N04 に関して X 位置のため実行可能なブロックではない。

N05 の後初めての実行可能なブロック。

自動 "GET"

軸が，原則としてはチャンネルで利用することができても，現在「チャンネル軸」として定義されていない場合は，GET が自動的に実行されます。その／それらの軸がすでに同期を取られている場合は，前処理停止は引き起こされません。

GET で受け取られた軸はキーリセットやプログラムリセットの後でもこのチャンネルに割当てられたままです。プログラムの開始時に，この転送された軸あるいは主軸がその元のチャンネルで必要な場合は，そのログラムで再割当てしなければなりません。

この軸は POWER ON と同時にマシンデータで定義されたそのチャンネルに割当てられます。

直接軸転送 : GETD

軸は GETD (GET Directly) で別のチャンネルから直接受け取られます。これは，この GETD の場合マッチング用 RELEASE が別のチャンネルで全くプログラムされていないということを意味しています。さらには他のチャンネル通信が確立されているということも意味しています（例えば待機マーカなど)。

プログラミング例

6 本の軸の内，次のものがチャンネル 1 での加工に使用されます : 第 1，第 2，第 3 および第 4。

チャンネル 2 の第 5 と第 6 軸はワークピースの交換に使用されます。

軸 2 はこの 2 つのチャンネル間で転送され，そのから POWER ON 後にチャンネル 1 に割当てられることになっています。

チャンネル 1 のプログラム "MAIN"

%_N_MAIN_MPF	
INIT (2,"TRANSFER2")	チャンネル 2 のプログラム TRANSFER2 を選択
N... START (2)	チャンネル 2 のプログラム開始
N... GET (AX2)	軸 AX2 の受取り
N... RELEASE (AX2)	軸 AX2 をイネーブル
N... WAITM (1,1,2)	チャンネル 1 および チャンネル 2 の同期化のため両チャンネルの待機マーカを待つ
N... N... M30	軸転送後プログラムの残り

チャンネル 2 のプログラム "Transfer2"

%_N_TRANSFER2_MPF	
N... RELEASE (AX2)	
N160 WAITM (1,1,2)	チャンネル 1 および チャンネル 2 の同期化ため両チャンネルの待機マーカを待つ
N150 GET (AX2)	軸 AX2 の受取り
N... N...M30	軸転送後プログラムの残り

1.16 NEWCONF: マシンデータを有効に設定

機能

有効性レベル "NEW_CONFIG" のマシンデータは
すべて NEWCONF 言語コマンドで有効に設定されます。
この機能はソフトキー "Set MD active" を起動することに
相当します。
この NEWCONF 機能が実行される時には暗黙の前処理
停止が存在します，つまりパス移動が中断されます。

説明

NEWCONF	"NEW_CONFIG" 有効性レベルのマシンデータはすべて有効に設定される

プログラミング例

フライス加工操作：様々なテクノロジで穴あけ位置の加工

N10 $MA_CONTOUR_TOL[AX]=1.0	; マシンデータ変更
N20 NEWCONF	; マシンデータを有効に設定

1.17 WRITE：ファイルの書込み

プログラミング

WRITE(var int error, char[160] filename, char[200] string)

WRITE コマンドは指定されたファイルの終わりにブロックを付け加えます。

パラメータの説明

error	リターン用エラー変数 0 エラーなし 1 パス禁止 2 パスが見つからない 3 ファイルが見つからない 4 ファイルタイプ誤り 10 ファイルはいっぱい 11 ファイル使用中 12 フリーリソースなし 13 アクセス権なし 20 その他のエラー
filename	文字列が書き込まれるファイルの名前。 ファイル名はパスおよびファイルの識別子で指定できる。パス名は絶対名，すなわち "/" で始まらねばならない。ファイル名にドメイン識別子 (_N_) がない場合は，相応に付け加えられる。識別子（_MPF，_SPF または _CYC）が何も指定されていなければ，_MPF が自動的に付加される。パスが何も指定されていなければ，そのファイルはカレントディレクトリ（選択されたプログラムのディレクトリ）に保存される。ファイル名の長さは最大 32 バイト，パスの長さは最大 128 バイトまで可能。 例： PROTFILE _N_PROTFILE _N_PROTFILE_MPF _N_MPF_DIR__N_PROTFILE_MPF\
string	書き込まれるテキスト。内部的に LF がそこで付加される；これはテキストが一文字だけ長くなるということを意味する。

機能

WRITE コマンドを使って，データ（例えばサイクル測定の結果など）を指定ファイルの終わりに付け加えることができます。

このログファイルの KB 単位での最大長さを MD 11420 LEN_PROTOCOL_FILE で設定します。この長さは WRITE コマンドで作成されたファイルのすべてに適用できます。

ファイルが一度この指定長さに達すると，エラーメッセージが出力され文字列は保存されません。空きメモリが十分にある場合は，新しいファイルを作成することができます。

作成されたファイルは

- すべてのユーザが読み，編集し，削除することができ，
- 現在実行中のパートプログラムに書き込むことができます。

ブロックは M30 後，ファイルの終わりで挿入されます。

プログラミング例

N10 DEF INT ERROR	;
N20 WRITE(ERROR,"TEST1","LOG FROM 7.2.97")	; LOG FROM 7.2.97 からのテキストをファイル TEST1 に書き込む
N30 IF ERROR	;
N40 MSG ("Error with WRITE command: <<ERROR)	;
N50 M0	;
N60 ENDIF	;
...	
WRITE(ERROR, "/_N_WKS_DIR/_N_PROT_WPD/ _N_PROT_MPF", "LOG FROM 7.2.97")	; 絶対パス

追加説明

- 当該ファイルが NC に存在しない場合は，新たに作成し WRITE コマンドで書き加えることができます。

- 同じ名前のファイルがハードディスク上に存在する場合は，このファイルが閉じられた後重ね書きされます（NC 内）。
  改善法：NC 内のその同じ名前をサービス操作エリアの下で " 属性 " ソフトキーを使って変更します。

機械メーカ

パートプログラムからのブロックは WRITE コマンドでファイルにストアすることができます。ログファイル用のファイルサイズ (KB) はマシンデータで指定します。

1.18 DELETE：ファイルの削除

プログラミング

DELETE(var int error, char[160] filename)

DELETE コマンドは指定のファイルを削除します。

パラメータの説明

error	リターン用エラー変数 0　エラーなし 1　パス禁止 2　パスが見つからない 3　ファイルが見つからない 4　ファイルタイプ誤り 11　ファイル使用中 12　フリーリソースなし 20　その他のエラー
filename	削除されるファイルの名前。 ファイル名はパスおよびファイルの識別子で指定できる。パス名は絶対名，すなわち "/" で始まらねばならない。ファイル名にドメイン識別子 (_N_) がない場合は，相応に付け加えられる。識別子（_MPF，_SPF または _CYC）が何も指定されていなければ，_MPF が自動的に付加される。パスが何も指定されていなければ，そのファイルはカレントディレクトリ（選択されたプログラムのディレクトリ）に保存される。ファイル名の長さは最大 32 バイト，パスの長さは最大 128 バイトまで可能。 例 : PROTFILE _N_PROTFILE _N_PROTFILE_MPF _N_MPF_DIR__N_PROTFILE_MPF\

機能

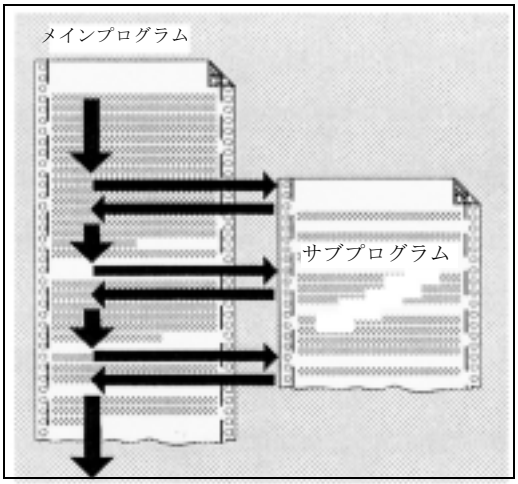
ファイルはすべて，それらが WRITE コマンドで作成されたものかどうかにかかわらず，DELETE コマンドで削除することができます。より高いアクセス権を使って作成されたファイルも同様に DELETE で削除できます。

プログラミング例

N10 DEF INT ERROR	;
N20 DELETE(ERROR,"TEST1")	; ファイル TEST1 を削除
N30 IF ERROR	;
N40 MSG ("Error with DELETE command:" <<ERROR)	;
N50 M0	;
N60 ENDIF	;
...	

2 サブプログラム，マクロ

2.1 サブプログラムの使用



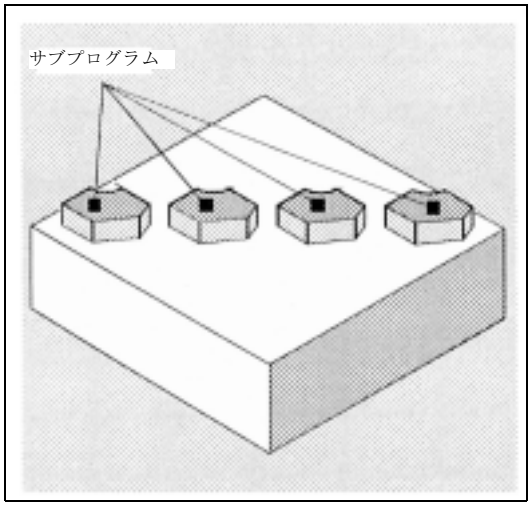
サブプログラムとは？

原則として，サブプログラムはパートプログラムと同じ構造をしています。サブプログラムは移動コマンドとスイッチングコマンドを有する NC ブロックで構成されます。

原則として，メインプログラムとサブプログラムとの違いは何もありません。サブプログラムには 2 回以上実行しなければならない加工サイクルあるいは加工セクションのいずれかが含まれます。

サブプログラムの使用

繰り返される加工順序は 1 度サブプログラムにプログラムするだけです。例えば 2 度以上現れる特定の輪郭形状や加工サイクルなど。



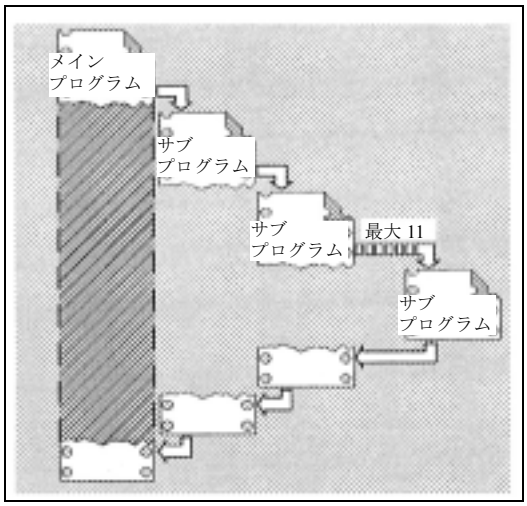
このサブプログラムはどんなメインプログラムでもその中でコールし，実行することができます。

サブプログラムの構造

サブプログラムの構造はメインプログラムの構造と同じです。

サブプログラムでは，パラメータ定義でプログラムヘッダをプログラムすることも可能です。

ネスティングの深さ



サブプログラムのネスティング

サブプログラムはそれ自身がサブプログラムコールを含むことができます。

そして，コールされたそのサブプログラムがさらにサブプログラムコールを含むといったことができます。

サブプログラムレベルの最大数，すなわちネスティングの深さは 12 です。

これは次のことを意味します :

メインプログラムには 11 個のネストされたサブプログラムコールを含むことができる。

制限事項

割込みルーチンでサブプログラムをコールすることも可能です。サブプログラムを使っての作業に関しては，4 レベルをフリーの状態に保たねばなりません，つまり 7 個のサブプログラムコールしかネストしてはなりません。

当社供給の加工および測定サイクルの場合，3 レベルが必要です。サブプログラムからサイクルをコールする場合は，レベル 5 の深さまでしかコールしてはなりません（4 レベルが割込みルーチン用にリザーブされている場合)。

2.2 SAVE メカニズムを有する サブプログラム

このためには，PROC の定義ステートメントで 追加コマンド SAVE を指定してください。

割込みルーチンの終わりで，モーダル G 関数は SAVE の属性によりそれらが割込みルーチンの開始時に所持していた値に設定されます。設定可能なゼロオフセット（モーダル G 関数グループ 8）に加えて，プログラム可能なゼロオフセットと基本オフセットが回復されます。G 関数グループ 15 （フィードタイプ）が例えば G94 から G95 に変更された場合は，適切な F 値が同様に回復されます。

例 :

サブプログラムの定義

PROC CONTOUR SAVE

N10 G91 ...

N100 M17

メインプログラム

%123

N10 G0 X... Y... G90

N20...

N50 CONTOUR

N60 X... Y...

CONTOUR サブプログラム G91 ではインクリメンタル指令が適用されます。メインプログラムに戻った後は，メインプログラムのモーダル関数が SAVE でストアされているため，再びアブソリュート指令が適用されます。

2.3 パラメータ転送を伴うサブプログラム

プログラム開始，PROC

プログラム実行時に呼出しプログラムからパラメータを引き継がねばならないサブプログラムは，ボキャブラリワード PROC で指定します。

プログラム終了 M17, RET

コマンド M17 は，サブプログラムの終わりを指定しますが，呼出しメインプログラムへ戻るための命令でもあります。

M17 に代わるものとして：ボキャブラリワード RET は，連続パスモードの中断や PLC への関数出力なしで，サブプログラムの終わりを表します。.

RET は個別 NC ブロックでプログラムしなければなりません。

例 :

PROC CONTOUR

N10...

...

N100 M17

メインプログラムとサブプログラム間でのパラメータの受渡し

メインプログラムでパラメータを用いて作業を行っている場合，その計算値や割当て値をサブプログラムでも使うことができます。このために，メインプログラムの現パラメータが，サブプログラムのコール時にその仮パラメータに渡され，それからサブプログラム実行中に処理されます。

例 :

N10 DEF REAL LENGTH,WIDTH

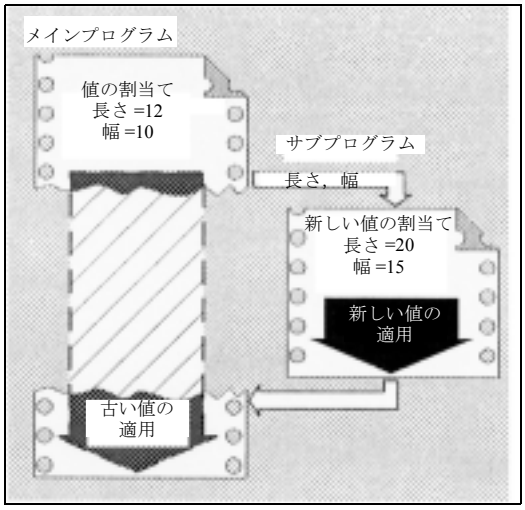
N20 LENGTH=12 WIDTH=10

N30 BORDER (LENGTH,WIDTH)

メインプログラムの N20 で割当てられている値が，サブプログラムがコールされると N30 に入れられます。

パラメータは指定順に渡されます。

パラメータ名はメインプログラムとサブプログラムで同じである必要はありません。



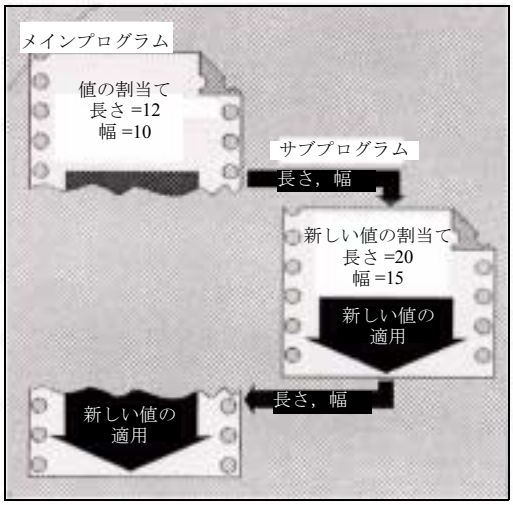
2 通りのパラメータ受渡し

値が渡されるだけ（値呼び）

渡されたパラメータがサブプログラムの実行につれて変化しても，そのことがメインプログラムに影響を及ぼすことはありません。当該パラメータはメインプログラムではもとのままでです（図参照）。

データ交換を伴うパラメータ転送（参照呼び）

サブプログラムでパラメータに変更があった場合，その変更のせいでメインプログラムでもパラメータが変更します（図参照）。



プログラミング

パラメータ受渡しに関係するパラメータは，サブプログラムの始めでそのタイプおよび名前とともにリストアップしなければなりません。

パラメータの受渡し 値呼び

PROC PROGRAM_NAME(VARIABLE_TYPE1 VARIABLE1,VARIABLE_TYPE2 VARIABLE2,...)

例 :

PROC CONTOUR(REAL LENGTH, REAL WIDTH)

パラメータ受渡し 参照呼び，
ボキャブラリワード VAR で識別

PROC PROGRAM_NAME(VAR VARIABLE_TYPE1 VARIABLE1,VAR VARIABLE_TYPE2 ...,)

例 :

PROC CONTOUR(VAR REAL LENGTH, VAR REAL WIDTH)

参照呼びでの配列の受渡し，
ボキャブラリワード VAR で識別

PROC PROGRAM_NAME(VAR VARIABLE_TYPE1 ARRAY_NAME1[array size],

VAR VARIABLE_TYPE2 ARRAY_NAME2[array size], VAR VARIABLE_TYPE3

ARRAY_NAME3[array size1, array size2], VAR VARIABLE_TYPE4 ARRAY_NAME4[

],VAR VARIABLE_TYPE5 ARRAY_NAME5 [,array size])

例 :

PROC PALLET (VAR INT ARRAY[,10])

追加説明

PROC の定義ステートメントは個別の NC ブロックでプログラムしなければなりません。パラメータ転送に関しては最大 127 個のパラメータを宣言することができます。

配列の定義

仮パラメータの定義には次のことが適用されます :

2 次元配列の場合，第 1 次元のフィールド数を指定する必要はありませんが，コンマは必要です。

例 :

VAR REAL ARRAY[,5]

特定の配列次元に関しては，可変長の配列を持つサブプログラムを処理するすることが可能です。ただし，その変数を定義する際は，要素をいくつ含む予定かを定義しなければなりません。

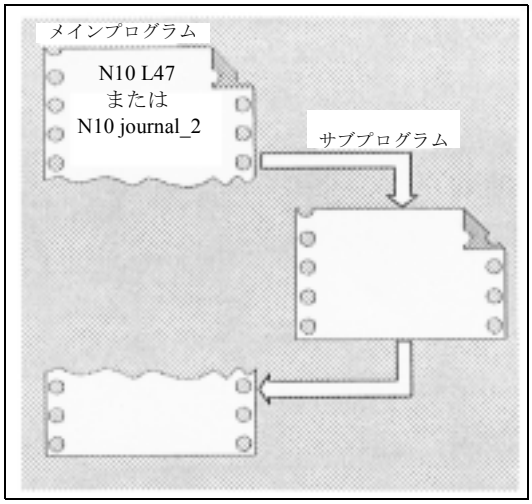
配列の定義の説明については，プログラミング 編「上級説明書」を参照してください。

プログラミング例

可変配列寸法に関するプログラミング	
%_N_DRILLING_PLATE_MPF	メインプログラム
DEF REAL TABLE[100,2]	位置テーブルを定義
EXTERN DRILLING_PATTERN (VAR REAL[,2],INT)	
TABLE[0.0]=-17.5	位置を定義
...	
TABLE[99.1]=45	
DRILLING_PATTERN(TABLE,100)	サブプログラムコール
M30	

渡された可変寸法の位置テーブルを使って穴あけ加工パターンを作成	
%_N_DRILLING_PATTERN_SPF	サブプログラム
PROC DRILLING_PATTERN(VAR REAL ARRAY[,2],-> -> INT NUMBER)	パラメータ引き渡し
DEF INT COUNT	
STEP: G1 X=ARRAY[COUNT,0]-> -> Y=ARRAY[COUNT,1] F100	加工順序
Z=IC(-5)	
Z=IC(5)	
COUNT=COUNT+1	
IF COUNT<NUMBER GOTOB STEP	
RET	サブプログラムの終了

2.4 サブプログラムのコール



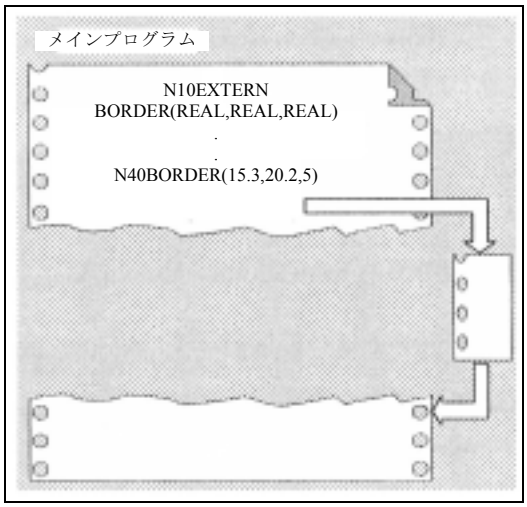
パラメータ転送のないサブプログラムコール

メインプログラムにおいて，アドレス L とサブプログラム番号，あるいはルーチン名のいずれかを使ってサブプログラムをコールすることができます。

例 :

N10 L47 または

N10 JOURNAL_2



パラメータ転送を伴うサブプログラム，EXTERN で宣言

パラメータ転送を伴うサブプログラムについては，それらをコールする前に，例えばメインプログラムの始めなどで EXTERN を使ってリストアップしなければなりません。

サブプログラムの名前と変数タイプは転送順に宣言します。

サブプログラムがワークあるいはグローバルサブプログラムディレクトリ内にある場合は，EXTERN を明記さえすればいいです。

EXTERN として，サイクルを宣言をする必要はありません。

EXTERN ステートメント

EXTERN NAME(TYPE1, TYPE2, TYPE3, ...) または

EXTERN NAME(VAR TYPE1, VAR TYPE2, ...)

例 :

N10 EXTERN BORDER(REAL, REAL, REAL)

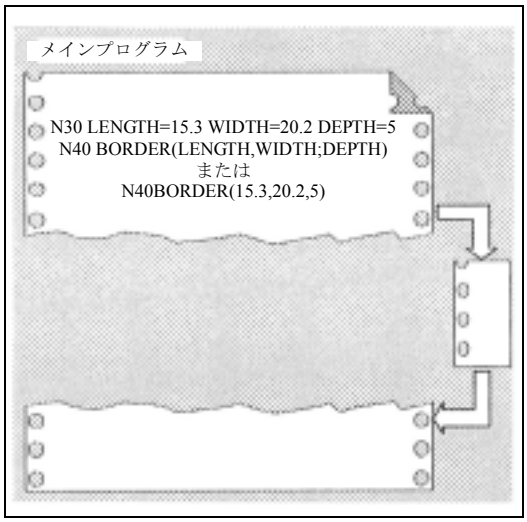
...

N40 BORDER(15.3,20.2,5)

N10 サブプログラムの宣言，N40 パラメータ転送を伴うサブプログラムコール

パラメータ転送を伴うサブプログラムコール

メインプログラムにおいて，プログラム名とパラメータ転送を指定することによってサブプログラムをコールします。パラメータを転送する際は，変数あるいは直接値（VAR パラメータ用ではない）を転送することができます。



例 :

N10 DEF REAL LENGTH,WIDTH,DEPTH

N20...

N30 LENGTH=15.3 WIDTH=20.2 DEPTH=5

N40 BORDER(LENGTH,WIDTH,DEPTH)

または

N40 BORDER(15.3,20.2,5)

サブプログラムの定義とサブプログラムのコールは整合しなければならない

変数タイプと受け渡しの順序は，PROC でサブプログラム名に宣言されている定義と一致しなければなりません。パラメータ名はメインプログラムとサブプログラムで異なっていてもかまいません。

例 :

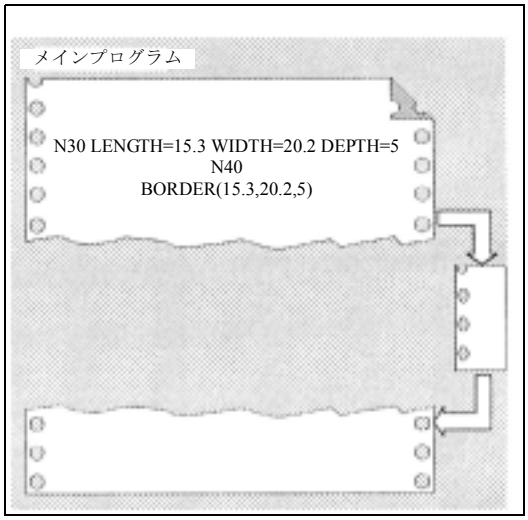
サブプログラムにおける定義 :

PROC BORDER(REAL LENGTH, REAL WIDTH,
REAL DEPTH)

メインプログラムにおけるコール :

N30 BORDER(LENGTH, WIDTH, DEPTH)

不完全なパラメータ転送



サブプログラムコールの際は，指定値と指定パラメータのみ省略することができます。この場合，問題のパラメータにはサブプログラムで値ゼロが割当てられます。

順序を示すためにコンマは必ず書かねばなりません。当該パラメータが順序の終わりに来る場合は，その対応コンマも省略することができます。

すぐ前の例に戻って :

N40 BORDER(15.3, ,5)

平均値 20.2 がここでは省略されています。

（注）AXIS タイプの現パラメータは省略してはなりません。
VAR パラメータは完全に渡さねばなりません。

不完全なパラメータ転送に関しては，システム変数 $P_SUBPAR[i] で，当該転送パラメータがサブプログラムに対してプログラムされているかどうかを示すことができます。

このシステム変数には引数 (i) として当該転送パラメータの番号が含まれます。

このシステム変数 $P_SUBPAR は以下の値を戻します :

- TRUE，ただし当該転送パラメータ がプログラムされている場合
- FALSE，ただし転送パラメータとして値が何も設定されていない場合。

許されないパラメータ番号が指定された場合は，パートプログラム処理はアラーム出力と同時に打ちきられます。

例 :

サブプログラム

PROC SUB1 (INT VAR1, DOUBLE VAR2)

IF $P_SUBPAR[1]==TRUE

; パラメータ VAR1 はサブプログラムコールでプログラムされている

ELSE

; パラメータ VAR1 はサブプログラムコールでプログラムされておらず，したがってシステムによってデフォルト値 0 で事前設定されている

ENDIF

IF $P_SUBPAR[2]==TRUE

; パラメータ VAR2 はサブプログラムコールでプログラムされている

ELSE

; パラメータ VAR2 はサブプログラムコールでプログラムされておらず，したがってシステムによってデフォルト値 0.0 で事前設定されている

ENDIF

; パラメータ 3 は定義されていない

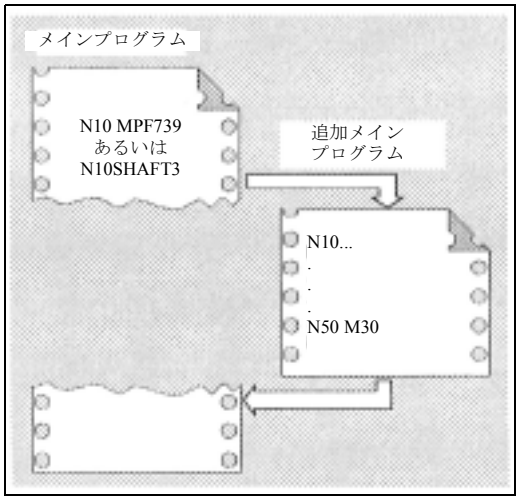
IF $P_SUBPAR[3]==TRUE -> Alarm 17020

M17

メインプログラムをサブプログラムとしてコール

メインプログラムはサブプログラムとしてコールすることもできます。

メインプログラムに設定されているプログラムの終了 M2 または M30 がこの場合は M17 として評価されます（呼出しプログラムへの復帰を伴うプログラムの終了)。



プログラム名を指定することによってコールをプログラムしてください。

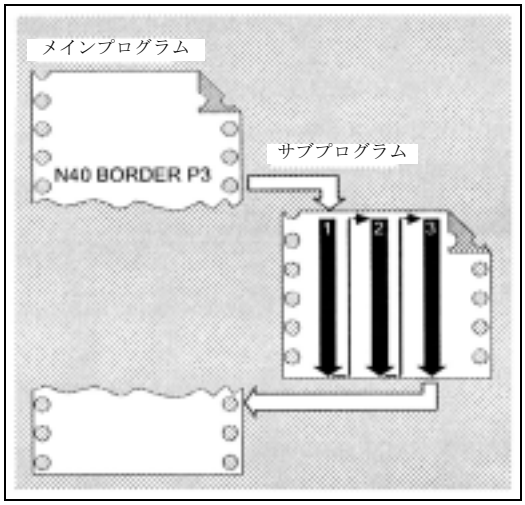
例 :

N10 MPF739 あるいは

N10 SHAFT3

サブプログラムもまたメインプログラムとして開始することができます。

2.5 プログラム反復を伴うサブプログラム



プログラム反復，P

サブプログラムを連続して複数回実行したい場合は，サブプログラムコールのブロックにアドレス P で必要とするプログラム反復回数をプログラムすることができます。

例 :

N40 BORDER P3

サブプログラム Border は連続して 3 回実行されねばなりません。

値の範囲 :

P: 1 ... 9999

すべてのサブプログラムコールに対して次のことが適用されます :

サブプログラムコールは必ず個別の NC ブロックにプログラムしなければなりません。

プログラム反復とパラメータ転送を伴うサブプログラムコール

パラメータはプログラムコール時，すなわち最初のパスで転送されるだけです。パラメータはプログラムが反復されている間不変です。

プログラム反復中にパラメータを変更したい場合は，サブプログラムで宣言文を定義しなければなりません。

2.6 モーダルサブプログラム，MCALL

モーダルサブプログラムコール，MCALL

この機能があると，サブプログラムがパスモーションを有するブロックのたびに自動的にコールされ実行されます。

このようにして，ワーク上の様々な位置で実行しなければならないサブプログラムのコールを自動化することができます。例えば，パターンの穴あけ加工の場合など。

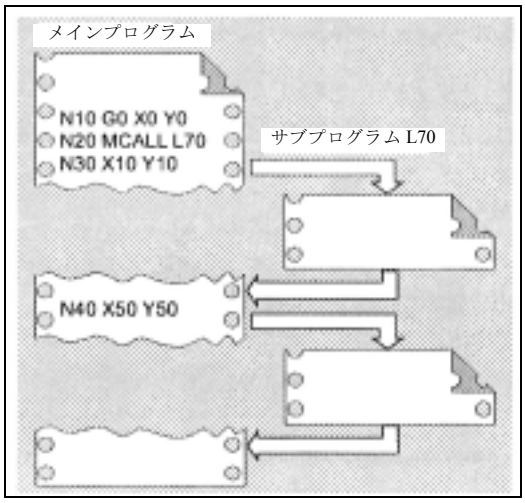
例：

N10 G0 X0 Y0

N20 MCALL L70

N30 X10 Y10

N40 X50 Y50



ブロック N30 と N40 では，プログラム位置がアプローチされ，それからサブプログラム L70 が実行されます。

N10 G0 X0 Y0

N20 MCALL L70

N30 L80

この例では，プログラムされたパス軸を有する，次の NC ブロックがサブプログラム L80 にストアされます。そして L70 が L80 にコールされます。

⚠ プログラム実行の際は，常に 1 度に MCALL コール 1 回しか適用で来ません。パラメータは MCALL で 1 度渡されるだけです。.

モーダルサブプログラムコールの停止

サブプログラムコールのない MCALL で，あるいは新しいサブプログラムに新しいモーダルサブプログラムコールをプログラムすることによって停止されます。

2.7 サブプログラムの間接コール

間接サブプログラムコール，CALL

プログラムの特定の個所における現行の条件次第で，様々なサブプログラムをコールすることができます。

サブプログラムの名前は STRING タイプの変数にストアします。そのサブプログラムコールは CALL とその変数名で発行します。

間接サブプログラムコールはパラメータ転送を伴わないサブプログラムに対してのみ可能です。

サブプログラムを直接コールする場合は，その名前を文字列定数にストアしてください。

例 :

文字列定数で直接コール :

```
CALL "/_N_WCS_DIR/_N_SUBPROG_WPD/
_N_PART1_SPF"
```

変数で間接コール :

```
DEF STRING[100] PROGNAME
PROGNAME="/_N_WCS_DIR/_N_SUBPROG_WPD/
_N_PART1_SPF"
CALL PROGNAME
```

サブプログラム PART1 が変数 PROGNAME に代入されています。CALL とパス名でそのサブプログラムを間接的にコールすることができます。

2.8 パス指定でパラメータを伴うサブプログラムコール，PCALL

PCALL を使って，パラメータ転送を伴うサブプログラムを絶対パスでコールすることができます :

PCALL path/program name (parameter 1, ..., parameter n)

説明

PCALL	絶対パス名によるサブプログラムコールのためのボキャブラリワード
Path name	“/” で始まり，サブプログラム名が含まれる絶対パス名 絶対パス名が指定されていない場合は，PCALL はプログラム識別子による標準的なサブプログラムコールのように機能する。プログラム識別子は先頭文字 _N_ や拡張子なしで書く。 プログラム名を先頭文字 _N_ と拡張子を付けてプログラムしたい場合は，そのことを Extern として，先頭文字 _N_ と 拡張子で明示的に宣言しなければなりません。
Parameters 1 to n	サブプログラムの PROC ステートメントと一致する現パラメータ

例 :

PCALL/_N_WCS_DIR/_N_SHAFT_WPD/
SHAFT(parameter1, parameter2, ...)

2.9 現在ブロック表示の抑止，DISPLOF

プログラミング

PROC ... DISPLOF

機能

DISPLOF があると，現在ブロック表示がサブプログラムのために抑止されます。DISPLOF は PROC ステートメントの最後に置かれます。

現在ブロックの代わりに，サイクルコールあるいはサブプログラムコールが表示されます。

デフォルトで，ブロック表示が起動されます。DISPLOF によるブロック表示の停止は，サブプログラムからの復帰あるいはプログラムの終了まで適用されます。この DISPLOF 属性を有するサブプログラムからさらに追加サブプログラムがコールされた場合は同様にそこで現在ブロック表示は抑止されます。ブロック表示を抑止した状態でサブプログラムが非同期のサブプログラムに割込まれた場合は，現在のサブプログラムのブロックが表示されます。

プログラミング例

サイクルの間現在ブロック表示の抑止

%_N_CYCLE_SPF	
;$PATH=/_N_CUS_DIR	
PROC CYCLE (AXIS TOMOV, REAL POSITION) SAVE DISPLOF	
	; 現在ブロック表示抑止
	; ここで現在ブロックとしてサイクルコールを表示
	; 例えば : CYCLE(X, 100.0)
DEF REAL DIFF	; サイクル内容
G01 ...	;
...	
RET	; サブプログラムが戻り，呼出しプログラムの次のブロックから再び表示

2.10 単一ブロックの抑止，SBLOF, SBLON

プログラミング

```
PROC ... SBLOF      ; このコマンドは PROC ブロックあるいは個別ブロックにプログラムできる
SBLON               ; このコマンドは個別ブロックにプログラムしなければならない
```

説明

SBLOF	単一ブロックの停止
SBLON	単一ブロックの再起動

機能

プログラム別単一ブロック抑止

単一ブロックタイプに関してはすべて，SBLOF で指定されたプログラムは，1 個のブロックのように全体として実行されます。SBLOF は，PROC 行に書き入れると，サブプログラムが終了あるいは打ち切られるまで有効です。

SBLOF はコールされるサブプログラムでも有効です。

例 :

```
PROC EXAMPLE SBLOF
G1 X10
RET
```

プログラムにおいて単一ブロック抑止

SBLOF はブロックに単独で存在することができます。このブロックより先は,

- 次の SBLON まで，あるいは
- 有効なサブプログラムレベルの終了まで，単一ブロックモードは停止されます。

例：

```
N10 G1 X100 F1000
N20 SBLOF                                   単一ブロック停止
N30 Y20
N40 M100
N50 R10=90
N60 SBLON                                   単一ブロック再起動
N70 M110
N80 ...
```

N20 から N60 まではワンステップとして単一ブロックモードで実行されます。

非同期サブプログラムの単一ブロックディスエーブル

REPOS を使ってシステム内部で実行される ASUP1.SYF および ASUP2.SYF サブプログラムはステップバイステップで実行しなければなりません。システム ASUP は，SBLOF をプログラミングすることによって 1 ステップで実行することができます。

例 :

```
N10 SBLOF
N20 IF $AC_ASUP==’H200’
N30 RET                                  モード変更で REPOS なし
N40 ELSE
N50 REPOSA                               その他の場合の REPOS
N60 ENDIF
N70 RET
```

補足条件

- 現在ブロックの表示は DISPLOF によってサイクルの間抑止することができます。
- DISPLOF が SBLOF と一緒にプログラムされている場合は，サイクル内の単一ブロック停止でサイクルコールが依然として表示されたままです。
- MD 20117 内のデフォルト設定 :
  単一ブロックモードにおける非同期サブプログラムの挙動用 IGNORE_SINGLEBLOCK_ASUP は，BLOF をプログラムすることによってプログラム別に重ね書きすることができます。.
- 試験目的のために，OPI 変数で SBLOF の有効性を抑止することが可能です（OEM ドキュメンテーション参照）。

プログラミング例 1

サイクルがプログラマに対するコマンドのように機能する

メインプログラム

N10 G1 X10 G90 F200
N20 X-4 Y6
N30 CYCLE1
N40 G1 X0
N50 M30

プログラムサイクル : 1

N100 PROC CYCLE1 DISPLOF SBLOF	単一ブロック抑止
N110 R10=3*SIN(R20)+5	
N120 IF (R11 <= 0)	
N130 SETAL(61000)	
N140 ENDIF	
N150 G1 G91 Z=R10 F=R11	
N160 RET	

サイクル CYCLE1 は，単一ブロックが有効の場合，ワンステップとして実行される。

プログラミング例 2

変更ゼロオフセットとツールオフセットを起動するために PLC から実行される ASP は多分目で確認できないでしょう。

```
N100 PROC NV SBLOF DISPLOF
N110 CASE $P_UIFRNUM OF 0 GOTOF _G500
     -->1 GOTOF _G54 2 GOTOF _G55 3
     -->GOTOF _G56 4 GOTOF _G57
     -->DEFAULT GOTOF END
N120 _G54: G54 D=$P_TOOL T=$P_TOOLNO
N130 RET
N140 _G54: G55 D=$P_TOOL T=$P_TOOLNO
N150 RET
N160 _G56: G56 D=$P_TOOL T=$P_TOOLNO
N170 RET
N180 _G57: G57 D=$P_TOOL T=$P_TOOLNO
N190 RET
N200 END: D=$P_TOOL T=$P_TOOLNO
N210 RET
```

2.11 マクロ

マクロとは？

マクロとは，全体として一つの名前が割当てられている一連の命令です。G，M および H 機能あるいは L サブプログラム名もマクロとして使うことができます。

プログラム実行中にマクロがコールされると，そのプログラム名のもとでプログラムされている命令が順々に実行されます。

マクロの使用

繰り返し実行される一連の命令は，独立したマクロブロックにマクロとして 1 度，それもプログラムの始めに 1 度プログラムするだけです。それで，マクロはどんなメインプログラムやサブプログラムでもコールし実行することができます。

プログラミング

マクロはボキャブラリワード DEFINE...AS で識別します。

マクロの定義は以下の通りです :

DEFINE NAME AS <instruction>

例 :

マクロの定義 :

DEFINE LINE AS G1 G94 F300

NC プログラムでコール :

N20 LINIE X10 Y20

マクロの起動

- マクロは，NC にロードされると有効になります (" ロード " ソフトキー)。

3 桁の M/G 機能

- 3 桁の M および G 機能のプログラミングが可能です。
  例 :
  N20 DEFINE M100 AS M6
  N80 DEFINE M999 AS M6

追加説明

マクロのネスティングはできません。

2 桁の H および L 機能はプログラムできます。

プログラミング例

マクロの定義例

DEFINE M6 AS L6	ツール変更があると，必要なデータ転送を実行するサブプログラムがまずコールされる。実際のツール変更 M 機能がそのサブプログラムに出力される（例えば，M106)。
DEFINE G81 AS DRILL(81)	DIN G 機能のエミュレーション
DEFINE G33 AS M333 G333	スレッド切削加工中，PLC との同期が要求される。もとの G 機能 G33 は，ユーザにとってプログラミングに変更がないように，G333 に改名されている。

グローバルマクロファイルの定義例 :

マクロファイルを制御装置に読み込んだら，当該マクロを起動させます（上記参照)。これらのマクロはこれでパートプログラムで使用することができます。

%_N_UMAC_DEF	
;$PATH=/_N_DEF_DIR; カスタマ別マクロ	
DEFINE PI AS 3.14	
DEFINE TC1 AS M3 S1000	
DEFINE M13 AS M3 M7	; 主軸右，クーラントオン
DEFINE M14 AS M4 M7	; 主軸左，クーラントオン
DEFINE M15 AS M5 M9	; 主軸停止，クーラントオフ
DEFINE M6 AS L6	; ツール変更プログラムコール
DEFINE G80 AS MCALL	; 穴あけ加工サイクル解除
M30	;

- ボキャブラリワードおよび予約名はマクロで再定義してはなりません。
- マクロの使用によって制御装置のプログラミング言語が著しく変わる可能性があります！
  従って，マクロを使用する際は注意してください。
- マクロはNCプログラムでも宣言できます。
  マクロ名として識別子のみ許されます。
  G機能マクロは，制御装置全体に対しグローバルにマクロブロックで定義することしかできません。
- マクロに関しては，識別子，G，M，H機能，およびLプログラム名を定義することができます。

2.12 外部サブプログラムの実行

プログラムは，EXTCALL を使って "Execution from external" モードで ハードディスクから再ロードすることができます。

EXTCALL path/program name

説明

EXTCALL	サブプログラムコールのためのキーワード
Path name	オプションであって，不可欠ではない STRING タイプの定数／変数 "/" で始まる絶対パス
Program name	プログラム識別子には先頭文字 _N_ がある／ない，そして拡張子はない。<"> 文字を使ってプログラム名に拡張子を付けることができる。

例 :

EXTCALL "SHAFT" または EXTCALL"/_N_WCS_DIR/_N_SHAFT_WPD/SHAFT"

機能

複雑なワークの加工中に，メモリスペース要求の関係でメインメモリにストアすることのできない独立した加工ステージ用のプログラム順序が作成される場合があります。

EXTCALL を使って，"Execution from external" モードで ハードディスク からプログラムを再ロードすることができます。

ハードディスクのディレクトリ構造体経由でアクセスできるプログラムはすべて再ロードできます。

SD 42700 EXT_PROG_PATH

チャンネル別設定データ

コールパスの設定オプションがフレキシブルなので，SD 42700 EXT_PROG_PATH が利用できます。

SD 42700 は，識別子と結合してコールされるべきである「プログラム絶対パス名」を形成するパス定義を含んでいます。

絶対パス名なしにその外部サブプログラムがコールされた場合は，ユーザメモリからのサブプログラムコールに関しては同じサーチパスが MMC に対して実行されます。

調整可能なロードメモリ（FIFO バッファ）

"Execution from external" モードでプログラム（メインプログラムまたはサブプログラム）を処理するには NCK にロードメモリが必要です。ロードメモリサイズのデフォルト設定値は 30 キロバイトです。

このメモリサイズは MD18360 EXT_PROG_BUFFER_SIZE で調整することができます。

POWER ON, RESET

リセットや POWER ON が実行されると，外部サブプログラムコールが中断され，その対応ロードメモリが消去されます。

追加説明

外部サブプログラムは，GOTOF，GOTOB，CASE，IF - ELSE，FOR，LOOP，WHILE，REPEAT などのジャンプコマンドを含むことは許されません。

サブプログラムコールは許されます。

プログラミング例

設定データ $SC_EXT_PROG_PATH は以下のパスを含みます : "_N_WCS_DIR/_N_WST1"

ユーザメモリ内にある名プログラム _N_MAIN_MPF が選択されます。

```
%_N_MACHINE1_MPF
N10 PROC MAIN
N20 ...
N30 EXTCALL ROUGHING_SPF                ; 外部サブプログラム ROUGHING_SPF のコール
N40 ...
N50 M30
```

サブプログラム ROUGHING_SPF（workpieces->WST1 の項目で MMC ディレクトリ構造体内にある）
N10 PROC ROUGHING
N20 G1 F1000 N30 X=... Y=... Z=... N40 ...
N90 M17

2.13 サイクル : ユーザサイクルのパラメータ設定

ファイルおよびパス

説明

ov.com	サイクルの概要
uc.com	サイクルコール記述

機能

カスタマイズされたサイクルはこれらのファイルを使ってパラメータ表示することができます。

動作

cov.com ファイルは，受渡し時にその標準サイクルに添付され，しかるべく拡張されることになっています。uc.com ファイルはユーザが作成することになっています。

両ファイルとも 受動ファイル方式で "User cycles" ディレクトリにロードすることになっています（そうでなければプログラムで適切なパス指定がなされなければなりません）：

;$PATH=/_N_CST_DIR。

cov.com の適合 - サイクルの概要

標準サイクルと共に提供される cov.com ファイルは
以下の構成をしています :

%_N_COV_COM	ファイル名
;$PATH=/_N_CUS_DIR	パス名
;Vxxx 11.12.95 Sca cycle overview	コメント行
C1(CYCLE81) drilling, centering	1 番目のサイクルコール
C2(CYCLE82) drilling, counterboring	2 番目のサイクルコール
...	
C24(CYCLE98) chaining of threads	最後のサイクルコール
M17	ファイルの終了

新しく加えられた各サイクルには以下の構文を有す
る行を付加しなければなりません :

C<Number> (<Cycle name>) comment text

Number: 整数，ただし前にそのファイルで使用され
ていてはなりません ;

Cycle name: 入れられるサイクルのプログラム名

Comment text: 任意にそのサイクルのコメントテキス
ト

例 :

C25 (MY_CYCLE_1) usercycle_1

C26 (SPECIALCYCLE)

uc.com ファイルの例

ユーザサイクル記述

この説明はプログラミング例が続いているということを基本としています :

例 :

以下の 2 つのサイクルに対して，サイクルパラメータ表示が新たになされなければなりません :

PROC MY_CYCLE_1 (REAL PAR1, INT PAR2, CHAR PAR3, STRING[10] PAR4)	
; このサイクルは以下の転送パラメータを有する :	
;	
;PAR1:	-1000.001 <= PAR2 <= 123.456 の範囲で実数値，100 でデフォルト
;PAR2:	0 <= PAR3 <= 999999 の範囲で正の整数，0 でデフォルト
;PAR3:	1 個の ASCII 文字
;PAR4:	サブプログラム名として長さ 10 の文字列
;	
...	
M17	

PROC SPECIALCYCLE (REAL VALUE1, INT VALUE2)	
; このサイクルは以下の転送パラメータを有する :	
;	
;VALUE1:	値範囲制限およびデフォルト値のない実数値
;VALUE2:	値範囲制限およびデフォルト値のない整数
...	
M17	

対応ファイル uc.com

%_N_UC_COM
;$PATH=/_N_CUS_DIR
//C25(MY_CYCLE_1) usercycle_1
(R/-1000.001 123.456 / 100 /Parameter_2 of cycle)
(I/0 999999 / 1 / integer value)
(C//"A" / Character parameter)
(S///Subprogram name)
//C26(SPECIALCYCLE)
(R///Entire length)
(I/*123456/3/Machining type)
M17

uc.com ファイルの構文記述

ユーザサイクル記述

各サイクルのヘッダ行 :

"//" が先頭で，その後は cov.com ファイルと同様

//C<Number> (<Cycle name>) comment text

例 :

//C25(MY_CYCLE_1) usercycle_

各パラメータの記述行 :

(<Data type identifier> / <Minimum value> <Maximum value> (<Default value> /
<Comment>

データタイプ識別子 :

R	**実数用**
I	数用
C	**文字用（1 文字）**
S	**文字列用**

最小値，最大値（省略可）

入力時にチェックされる入力値の制限 ; この範囲外の値は入力できません。

トグルキーで演算可能な値の列挙を記入することができます ; 値は "*" の後にリストアップされ，その場合他の値は許されません。

例 :

(I/*123456/1/Machining type)

文字列タイプおよび文字タイプには制限はありません ;

デフォルト値 （省略可）

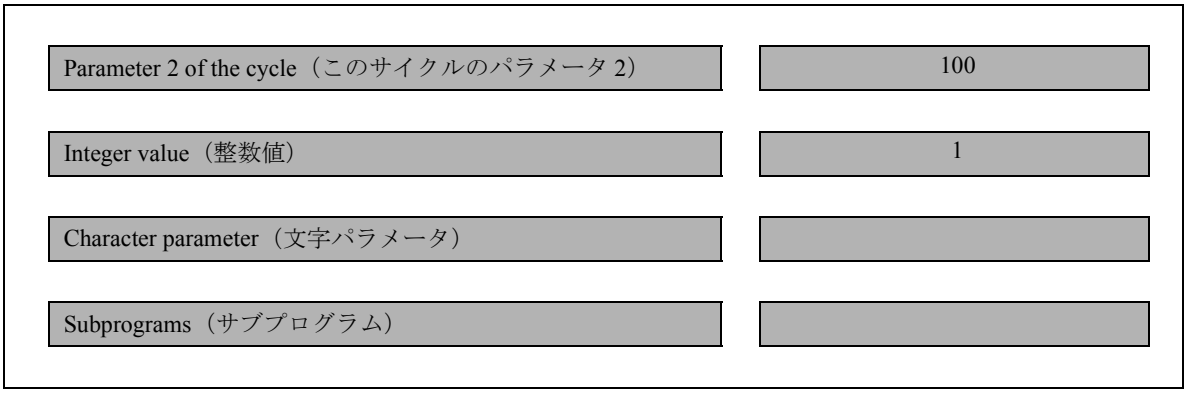
当該サイクルがコールされた時，対応画面でデフォルト値である値 ; オペレータ入力で変更できます。

コメント

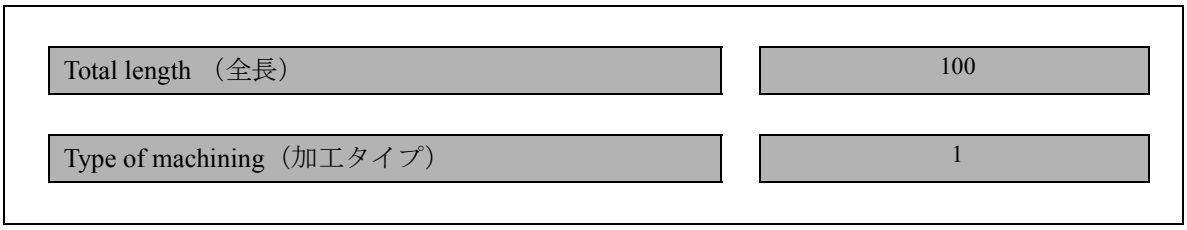
当該サイクルに関し，コール画面のパラメータ入力フィールドの前に表示される最大 50 文字のテキスト

両サイクルの表示例

サイクル MY_CYCLE_1 の表示画面



サイクル SPECIALCYCLE の表示画面



3 ファイル及びプログラムマネージメント

3.1 概要

メモリ構成

ユーザが利用できるメモリは 2 つのエリアに構成されています。

1. ユーザメモリ

ユーザメモリには現在のシステムとユーザデータが入っており，制御装置はそれらを使って動作します（有効ファイルシステム)。

例 :

有効マシンデータ，有効ツールオフセットデータ，有効ゼロオフセット。

2. プログラムメモリ

プログラムメモリにはファイルとプログラムがストアされます，したがって永久的なストアです（受動ファイルシステム)。

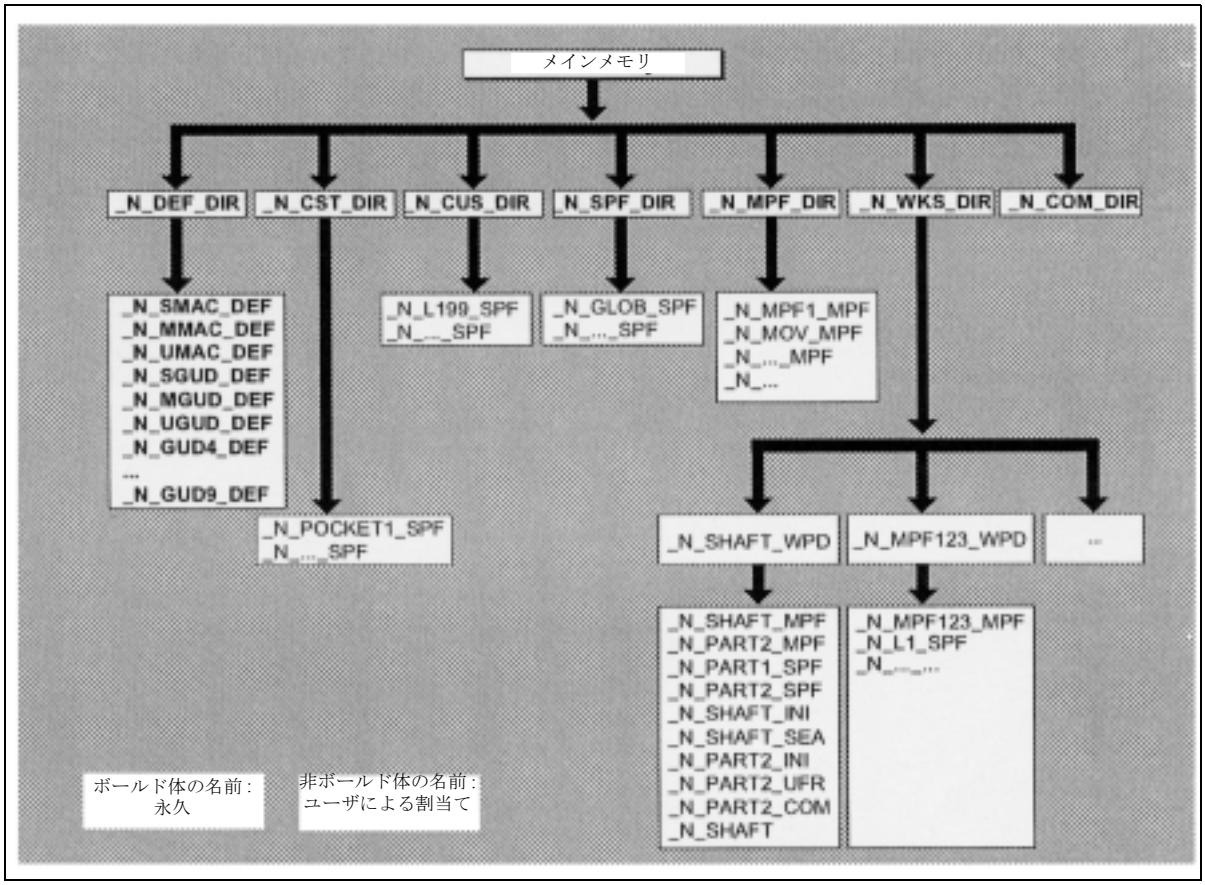
例 :

メインプログラムおよびサブプログラム，マクロ定義。

3.2 プログラムメモリ

概要

メインメモリにはメインプログラムとサブプログラムがストアされます。多数のファイルタイプもまた一時的にここにストアされるので，必要に応じて（例えば，特定のワークの加工に関する初期設定のために）作業エリアに転送することができます。



ディレクトリ

ディスプレイおよびオペレータユニットが接続されると，以下のディレクトリが標準として用意されます:

1. _N_DEF_DIR	データモジュールおよびマクロモジュール（スタートアップと同時に割当て）
2. _N_CST_DIR	標準サイクル（スタートアップと同時に割当て）
3. _N_CUS_DIR	ユーザサイクル（スタートアップと同時に割当て）
4. _N_WCS_DIR	ワーク
5. _N_SPF_DIR	グローバルサブプログラム
6. _N_MPF_DIR	メインプログラム用標準ディレクトリ
7. _N_COM_DIR	コメント用標準ディレクトリ

ファイルタイプ

メインメモリには以下のファイルタイプをストアすることができます:

name_MPF	メインプログラム
name_SPF	サブプログラム

name_TEA	マシンデータ
name_SEA	設定データ
name_TOA	ツールオフセット
name_UFR	ゼロオフセット／フレーム
name_INI	初期化ファイル
name_GUD	グローバルユーザデータ
name_RPA	R パラメータ
name_COM	コメント
name_DEF	グローバルユーザデータおよびマクロの定義

ワークディレクトリ，_N_WCS_DIR

ワークディレクトリは，プログラムディレクトリの標準構成中に _N_WCS_DIR の名前で存在します。

このワークディレクトリには，ユーザがプログラムしたワークのためのワークディレクトリがすべて含まれます。

ワークディレクトリ，識別子 WPD

データおよびプログラム処理をよりフレキシブルにするために，特定のデータやプログラムをグループ化する，つまり個々のワークディレクトリにストアすることができます。

ワークディレクトリには，ワークを加工するのに必要なファイルがすべて含まれます。

これらはメインプログラムであり，サブプログラムであり，初期化プログラムであり，コメントファイルでありえます。

例 :

ワーク SHAFT 用に作成されたワークディレクトリ _N_SHAFT_WPD には以下のファイルが含まれます :

_N_SHAFT_MPF	メインプログラム
_N_PART2_MPF	メインプログラム
_N_PART1_SPF	サブプログラム
_N_PART2_SPF	サブプログラム
_N_SHAFT_INI	ワークデータの一般的な初期化プログラム
_N_SHAFT_SEA	設定データ初期化プログラム
_N_PART2_INI	Part 2 プログラム用データの一般的な初期化プログラム
_N_PART2_UFR	Part 2 プログラム用フレームデータの初期化プログラム
_N_SHAFT_COM	コメントファイル

外部 PC でワークディレクトリの作成

下記のステップは外部データステーションで実行します。

ファイルおよびプログラムの管理（PC から制御装置まで）に関しては，ユーザーズマニュアル 操作編を参照してください。

;$PATH インストラクション

宛先パス $PATH=... はファイルの 2 行目に指定します。

例 :

;$PATH=/_N_WCS_DIR/_N_SHAFT_WPD

当該ファイルは指定パスにストアされます。

重要

このパスがない場合は，SPF タイプのファイルは /_N_SPF_DIR に，拡張子 _INI のついたファイルは作業メモリに，その他のファイルはすべて /_N_MPF_DIR にそれぞれストアされます。

前の例 SHAFT の場合のパスに関する例 :

%_N_SHAFT_MPF

;$PATH=/_N_WCS_DIR/_N_SHAFT_WPD

N10 G0 X... Z...

・

M2

・

・

%_N_SHAFT_SPF

加工用ワークの選択

チャンネルでの実行のためワークディレクトリを選択することができます。

同じ名前を持つメインプログラムがこのディレクトリにストアされている場合は，これが自動的に選択され実行されます。

例 :

ワークディレクトリ /_N_WCS_DIR/_N_SHAFT_WPD にはファイル _N_SHAFT_SPF および _N_SHAFT_MPF が含まれます。

ジョブリストおよび実行用プログラムの選択に関しては，ユーザーズマニュアル 操作編を参照してください。

サブプログラムコールとサーチパス

サブプログラム（あるいは初期化ファイル）をコールする際にパートプログラムでサーチパスが明確に指定されていない場合は，その呼出しプログラムが固定サーチパスをサーチします。

絶対パス指定でのサブプログラムコールの例 :

CALL"/_N_CST_DIR/_N_CYCLE1_SPF"

プログラムは通常パスを指定しないでコールされます :

例 :

CYCLE1

サーチパス順序

1. Current directory / name	ワークディレクトリまたは 標準ディレクトリ _N_MPF_DIR
2. Current directory / name_SPF	
3. Current directory / name_MPF	
4. /_N_SPF_DIR / name_SPF	グローバルサブプログラム
5. /_N_CUS_DIR / name_SPF	ユーザサイクル
6. /_N_CST_DIR / name_SPF	標準サイクル

3.3 ユーザメモリ

初期化プログラム

作業メモリデータを初期化するたに使用されるプログラムがあります。

この目的のために，以下のファイルタイプを使うことができます :

name_TEA	マシンデータ
name_SEA	設定データ
name_TOA	ツールオフセット
name_UFR	ゼロオフセット／フレーム
name_INI	初期化ファイル
name_GUD	グローバルユーザデータ
name_RPA	R パラメータ

データエリア

データは，それらが適合する様々なエリアに編成することができます。例えば，1 つの制御装置で複数のチャンネルを使用することができ，さらに通常は複数の軸を使用できます。従って以下のエリアが利用できます :

識別子	データエリア
NCK	NCK 別データ
CH<n>	チャンネル別データ（n はチャンネル番号）
AX<n>	軸別データ（n はマシン軸の番号）
TO	ツールデータ
COMPLETE	全データ

外部 PC で初期化プログラムの作成

データを保存する際にユニットとして取扱われるべきエリアを決めるためにデータエリア識別子とデータタイプ識別子を用いることができます。

例 :

_N_AX5_TEA_INI	軸 5 のマシンデータ
_N_CH2_UFR_INI	チャンネル 2 のフレーム
_N_COMPLETE_TEA_INI	全マシンデータ

制御装置を始動させると，一組のデータが自動的にロードされ制御装置の適切な操作が保証されます。

初期化プログラムの保存

作業メモリ内のファイルは外部 PC で保存することができるうえ，そこから再び作業メモリに読み込むことができます。

- 当該ファイルは COMPLETE で保存します。
- INI ファイル :INITIAL を使って _N_INITIAL_INI を全エリアにわたって作ることができます。

初期化プログラムのロード

INI プログラムは，それが単一チャンネルのデータしか使わないのであれば，パートプログラムとして選択およびコールすることもできます。したがって，プログラム制御データを初期化することも可能です。

ファイルタイプに関する情報はユーザーズマニュアル 操作編 に記載されています。

多重チャンネル制御の手順

複数チャンネル用の CHANDATA（チャンネル番号）はファイル N_INITIAL_INI でのみ許されます。

N_INITIAL_INI はインストレーションファイルで，このファイルで制御装置の全データが初期化されます。

例 :

%_N_INITIAL_INI

CHANDATA(1)

; マシン軸割当て チャンネル 1

$MC_AXCONF_MACHAX_USED[0]=1

$MC_AXCONF_MACHAX_USED[1]=2

$MC_AXCONF_MACHAX_USED[2]=3

CHANDATA(2)

; マシン軸割当て チャンネル 2

$MC_AXCONF_MACHAX_USED[0]=4

$MC_AXCONF_MACHAX_USED[1]=5

CHANDATA(1)

; 軸のマシンデータ

; 正確停止 ウィンドウ コアース（粗）:

$MA_STOP_LIMIT_COARSE[AX1]=0.2 ; 軸 1

$MA_STOP_LIMIT_COARSE[AX2]=0.2 ; 軸 2

; 正確停止 ウィンドウ ファイン（微）:

$MA_STOP_LIMIT_COARSE[AX1]=0.01 ; 軸 1

$MA_STOP_LIMIT_COARSE[AX1]=0.01 ; 軸 2

パートプログラムでは，CHANDATA 命令は NC プログラムを実行するチャンネルに対してしか使わないかもしれません，つまり，この命令は NC プログラムが別のチャンネルで誤って実行されるのを防ぐために使うことができるということです。

エラーが発生すると，プログラム処理は打ち切られます。

3.4 ユーザデータの定義

機能

ユーザデータ (GUD) はスタートアップ手順の一部として定義されます。

必要なマシンデータはそれに応じて初期設定しなければなりません。

ユーザメモリを構成しなければなりませんが，必要なメモリ構成は定義ファイルの後にロードされるファイル %_N_INITIAL_INI で定義しなければなりません。関係するマシンデータはすべてその名前の構成要素として GUD を有しています。

- ユーザデータ (GUD) はサービスオペレーティングエリアで定義することができます。これは，データバックアップ (%_N_INITIAL_INI) の長たらしい再インポートは必要ないということを意味しています。
  以下の事項が適用されます :
  - ハードディスク上にある定義ファイルは有効ではない。
  - NC 上にある定義ファイルは常に有効である。

予約ブロック名

以下のモジュールはディレクトリ /_N_DEF_DIR にストアすることができます :

_N_SMAC_DEF	マクロ定義（当社，保護レベル 0）を含む
_N_MMAC_DEF	マクロ定義（機械メーカー，保護レベル 2）を含む
_N_UMAC_DEF	マクロ定義（ユーザ，保護レベル 3）を含む
_N_SGUD_DEF	グローバルデータの定義（当社，保護レベル 0）を含む
_N_MGUD_DEF	グローバルデータの定義（機械メーカー，保護レベル 2）を含む
_N_UGUD_DEF	グローバルデータの定義（ユーザ，保護レベル 3）を含む
_N_GUD4_DEF	自由に定義可能
_N_GUD5_DEF	測定サイクルの定義（当社，保護レベル 0）を含む
_N_GUD6_DEF	測定サイクルの定義（当社，保護レベル 0）を含む
_N_GUD7_DEF	標準サイクルの定義（当社，保護レベル 0）を含むか，あるいは標準サイクルがなく自由に定義可能
_N_GUD8_DEF	自由に定義可能
_N_GUD9_DEF	自由に定義可能

アクセス許可は，APR あるいは APW コマンドを使って定義ファイルで割当てます。

GUD 定義ファイルをまず起動すると，そこに含まれている定義済みアクセス許可がすべて評価され，それから自動的に 元の GUD 定義ファイルの読取り／書込みアクセス許可に再転送されます。

（注）GUD 定義ファイルのアクセス許可入力値によって，GUD 定義ファイルにとって必要なアクセス許可を制限することはできますが，拡大することはできません。

例

定義ファイル _N_GUD7_DEF が次のものを含む :
APW2

a) 当該ファイル _N_GUD7_DEF は書込み保護として値 3 を有する。その値 3 がその後値 2 で重ね書きされる。

b) 当該ファイル _N_GUD7_DEF は書込み保護として値 0 を有する。この値に対する変更はない。

APW 命令で，このファイルの書込みアクセス権に遡及的な変更がなされます。

APR 命令で，このファイルの読取りアクセス権に遡及的な変更がなされます。

（注）この GUD 定義ファイルに，許可で認められているより高いアクセスレベルを誤って入力した場合は，そのアクティブファイルを再インポートしなければなりません。

ユーザデータ (GUD) の定義

1. ブロック _N_INITIAL_INI を保存。
2. ユーザデータの定義ファイルの作成
   - サービスオペレーティングエリアで
     事前定義ファイル名が用意されています（前のページ参照）：
     _N_SGUD_DEF
     _N_MGUD_DEF
     _N_UGUD_DEF
     _N_GUD4_DEF ... _N_GUD9_DEF
     これらの名前の付いたファイルには GUD 変数の定義を入れることができます。

   その変数を GUD 変数として識別し，その定義が有効となるべきエリアを定義するために追加属性が必要です：
     NCK, CHAN.

   暗黙の前処理停止を定義することができます：
     SYNR: 読取り中前処理停止
     SYNRW: 読取り／書込み中前処理停止
3. この定義ファイルを制御装置のプログラムメモリにロード。

   制御装置は必ずデフォルトディレクトリ _N_DEF_DIR を作成します。
   この名前は GUD 定義ファイルのヘッダにパスとして入力され，RS-232C インタフェースで読み込まれた時に評価されます。

定義ファイルの例，グローバルデータ（当社）：

%_N_SGUD_DEF	
;$PATH=/_N_DEF_DIR	
DEF NCK REAL RTP	; 後退面
DEF CHAN INT SDIS	; セーフティクリアランス
M30	

4. 定義ファイルの起動

GUD 定義ファイルは，NC（" ロード " ソフトキー）にロードされると有効になります。

スタティックメモリがフォーマットされるので，_N_INITIAL_INI を読み込む前にすべてのプログラム，フレームおよびマシンデータを保存してください。

5. データの保管

ファイル _N_COMPLETE_GUD を作業メモリからアーカイブすると，そのファイルに含まれているデータだけが保存されます。グローバルユーザ変数用に作成された定義ファイルは別にアーカイブしなければなりません。

グローバルユーザデータに対する変数割当ても _N_INITIAL_INI にストアされますが，その名前は定義ファイルの名前と同じでなければなりません。

グローバルデータ用定義ファイルの例
（機械メーカー）：

%_N_MGUD_DEF	
;$PATH=/_N_DEF_DIR	
; 機械メーカーのグローバルデータ定義	
DEF NCK SYNRW INT QUANTITY	; 読取り／書込み中暗黙の前処理停止 ; 制御装置で利用できる特定のデータ ; すべてのチャンネルからアクセス
DEF CHAN INT TOOLTABLE[100]	; マガジン位置にあるツール番号のチャンネル別イメージ用ツールテーブル
M30	; 各チャンネル用に作成された個別テーブル

3.5 ユーザデータ (GUD) に対する保護レベルの定義

説明

APR n	読取りアクセス保護
APW n	書込みアクセス保護
n	0/10（最高レベル）から 7/17（最低レベル）までの保護レベル n

APW 0 ～ 7, APR 0 ～ 7:

当該モジュール変数は，NC プログラムや MDA モードで書込みや読取りすることはできません。

APW 10 ～ 17, APR 10 ～ 17:

当該モジュール変数は，NC プログラムや MDA モードで今なお書込みや読取りすることができます。

保護レベル

0/10 = 当社

1/11 = OEM_ HIGH

2/12 = OEM _LOW

3/13 = エンドユーザ

4/14 to 7/17 = キースイッチ位置 3 から 0

（注）コマンド入力順序は以下の通りです :
APR.. APW..
その他の順序はいずれも構文エラーを意味します。完全ファイルを保護するために，コマンドはそのファイルの最初の行に入力しなければなりません！

機能

GUD モジュールの改ざんを防ぐために，このモジュールに対してアクセス基準を定義することができます。このような基準を導入することによって，例えば，機械メーカーが GUD モジュールとして設定してあるサイクルへの変更を禁止することが可能となります。

アクセス保護はこのモジュールで定義されたすべての変数に適用されます。

保護データにアクセスしようとすると，制御装置は適切なアラームを出力します。

動作

アクセス保護レベルは，どんな変数であれ、それを定義する前に，関係モジュールにその望ましい保護レベルでプログラムします。

両ボキャブラリワードは個別のブロックにプログラムしなければなりません。

読取り／書込みアクセス保護に関する定義ファイルの例（機械メーカー）：

```
%_N_GUD6_DEF
;$PATH=/_N_DEF_DIR
APR 5 APW 2                ; 保護レベル キースイッチ位置 2 で読取り／表示
                           ; 保護レベル OEM_LOW で書込み
                           ; 注意！この入力によって，ファイル自身のアクセス権が変更になる可能性がある（前記参照）
DEF CHAN REAL_CORRVAL
...
M30                        ;
```

3.6 GUD および MAC の自動起動

機能

GUD の定義ファイルおよびマクロ定義ファイルを編集します

NC で定義ファイルを編集した場合，Editor を出る時に，定義を有効に設定すべきかどうか催促してきます。

例 :

"Do you want to activate the definitions from file GUD7.DEF?"（「ファイル GUD7.DEF からその定義を起動したいですか？」）

"OK" → 次に，現在有効なデータを保存するかどうかプロンプトが表示されます。
"Do you want to save the previous data in the definitions?"（「定義内の前のデータを保存したいですか？」）

"OK" → 編集すべき定義ファイルの GUD ブロックが保存され，新しい定義が起動されると，保存されたデータが再びインポートされます。

"Cancel" → 新しい定義が起動され，古いデータはクリアされます。

"Cancel" → 定義ファイル内の変更は放棄され，その対応データブロックは変更されません。

アンロード

定義ファイルをアンロードすると，問い合わせを表示後，対応データブロックは削除されます。

ロード

定義ファイルをロードする場合，プロンプトが表示され，そのファイルを起動するかあるいはデータを保持するか聞いてきます。

起動しなければ，ファイルはロードされません。

カーソルがロードされた定義ファイル上にある場合は，定義を起動するためにソフトキーのラベル表示が " ロード " から "Activate" に変わります。

"Activate" を選択すると，もう一つのプロンプトが表示され，データを保持したいかどうか聞いてきます。

データは変数定義ファイルの場合のみ保存され，マクロの場合は保存されません。

追加説明

定義ファイルを起動するに十分なメモリー容量がない場合は，そのメモリサイズが一度変更されているので，そのファイルを NC から MMC へ転送して，その後，起動するためにもう一度 NC に逆転送しなければなりません。

4 保護ゾーン

4.1 保護ゾーン CPROTDEF，NPROTDEF の定義

プログラミング

DEF INT NOT_USED

CPROTDEF(n,t,applim,applus,appminus)

NPROTDEF(n,t,applim,applus,appminus)

EXECUTE (NOT_USED)

コマンドの説明

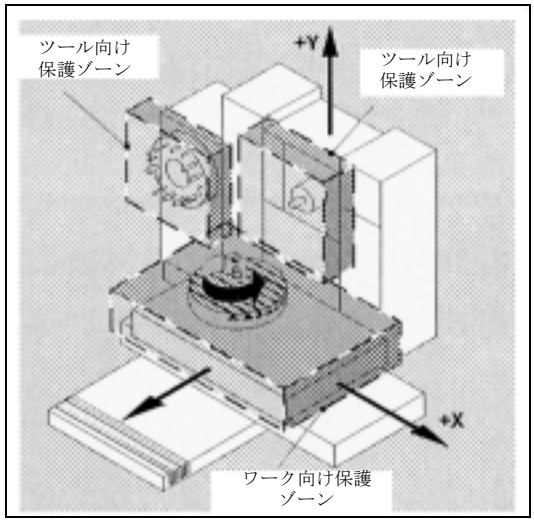
DEF INT NOT_USED	ローカル変数，データタイプ整数を定義 (cf. セクション 10)
CPROTDEF	チャンネル別保護ゾーン（NCU 572/573 の場合のみ）
NPROTDEF	マシン別保護ゾーン
EXECUTE	エンド定義

パラメータの説明

n	定義済み保護ゾーンの番号
t	TRUE = ツール向け保護ゾーン FALSE = ワーク向け保護ゾーン
applim	第 3 次元リミットのタイプ 0 = リミットなし 1 = 正方向 2 = 負方向 3 = 正および負方向リミット
applus	第 3 次元の正方向におけるリミット値
appminus	第 3 次元の負方向におけるリミット値
NOT_USED	EXECUTE についてはエラー変数による保護ゾーンへの影響なし

機能

マシン上の様々な要素やその構成部品およびワークの不正な動作を防ぐために，保護ゾーンを活用することができます。



ツール向け保護ゾーン :

ツールに付属する部品（例えば，ツール，ツールキャリアなど）用。

ワーク向け保護ゾーン :

ワークに付属する部品（例えば，ワークの部品，クランプテーブル，クランプ，主軸チャック，心押し台など）用。

動作

保護ゾーンの定義

保護ゾーンの定義には以下のものが含まれます :

- チャンネル別保護ゾーン用 CPROTDEF
- マシン別保護ゾーン用 NPROTDEF
- 保護ゾーンについての輪郭記述
- EXECUTE で定義の終了

NC パートプログラムで保護ゾーンを起動する際に，保護ゾーンの基準点に対する相対オフセットを指定することができます。

輪郭記述のための基準点

ワーク向け保護ゾーンは基本座標系で定義します。ツール向け保護ゾーンはツールキャリア基準点 F を基準にして定義します。

保護ゾーンの輪郭定義

保護ゾーンの輪郭は，選択平面内での最大 11 個の移動動作で指定します。

最初の移動動作は輪郭に向かう動作です。CPROTDEF あるいは NPROTDEF と EXECUTE との間にプログラムされる移動動作は実行されず，保護ゾーンを定義するだけのものです。

作業平面

必要な平面は，CPROTDEF および NPROTDEF の前に，G17，G18，G19 で選択し，EXECUTE の前に変更してはなりません。このアプリケーションは CPROTDEF あるいは NPROTDEF と EXECUTE との間にプログラムしてはなりません。

輪郭要素

下記のものが許されます :

- 直線輪郭要素として G0，G1
- 時計回りの円セグメントとして G2（ツール向け保護ゾーンに関してのみ）
- 反時計回り円セグメントとして G3

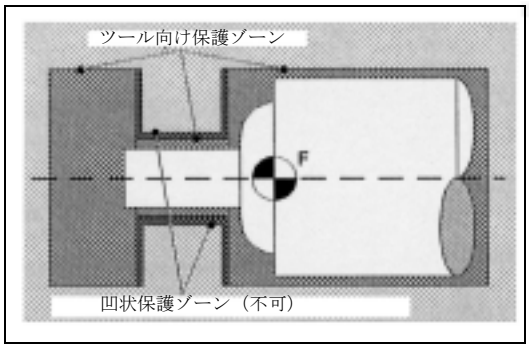
完全な円で保護ゾーンが描かれる場合は，その円を 2 つの半円に分割しなければなりません。配列順序 G2，G3 あるいは G3，G2 は許されません。必要なら短い G1 ブロックを挿入しなければなりません。

輪郭記述における最後の点は最初の点と一致しなければなりません。

外部保護ゾーン（ワーク向け保護ゾーンに対してのみ可能）は時計回り方向に定義します。

力学的にバランスの取れた保護ゾーン（例えば主軸チャックなど）の場合は，完全な輪郭（ただ単に回転の中心に至るまでだけでなく！）を記述しなければなりません。

ツール向け保護ゾーンは必ず凸状でなければなりません。凹状の保護ゾーンを望む場合は，分割して複数の凸状保護ゾーンにすべきです。



保護ゾーン定義中は，下記のものが有効であってはなりません :

- 切削半径あるいはツールノーズ径補正
- 変換
- フレーム .

基準点アプローチ (G74)，固定点アプローチ (G75)，ブロックサーチ停止，プログラムエンドのいずれもプログラムしてはなりません。

4.2 保護ゾーンの起動／停止 : CPROT, NPROT

プログラミング

CPROT (n,state,xMov,yMov,zMov)

NPROT (n,state,xMov,yMov,zMov)

コマンドとパラメータの説明

CPROT	**チャンネル別保護ゾーンのコール**
NPROT	マシン別保護ゾーンのコール
n	保護ゾーン番号
state	ステータスパラメータ 0 = 保護ゾーン停止 1 = 保護ゾーン事前起動 2 = 保護ゾーン起動
xMov,yMov,zMov	ジオメトリ軸上の定義済み保護ゾーンの移動

機能

衝突監視のため，前もって定義されている保護ゾーンを起動あるいは有効な保護ゾーンを停止します。

同じチャンネルで同時に有効となりえる保護ゾーンの最大数はマシンデータで定義します。

ツール向け保護ゾーンが有効でない場合は，ツールパスをワーク向け保護ゾーンと照合します。

ワーク向け保護ゾーンが有効でない場合は，保護ゾーンの監視は行われません。

動作

起動ステータス

保護ゾーンは通常パートプログラムにおいてステータス = 2 で起動します。

マシン向け保護ゾーンの場合でも，ステータスは常にチャンネル別です。

保護ゾーンを PLC ユーザプログラムで起動する場合は，ステータス = 1 で事前起動します。

保護ゾーンはステータス = 0 で停止，つまりディスエーブルします。オフセットは必要ありません。

（事前）起動時の保護ゾーンの変位

1，2，あるいは 3 次元で変位が生じる可能性があります。

オフセットが適用されるのは :

- ワーク別保護ゾーンのマシンゼロ点，
- ツール別保護ゾーンのツールキャリア基準点 F。

追加説明

保護ゾーンは，立ち上げとそれに続く基準点アプローチ実行後直ちに起動することができます。

そのためには，システム変数 $SN_PA_ACTIV_IMMED [n] または $SN_PA_ACTIV_IMMED[n] を TRUE に設定しなければなりません。保護ゾーンは必ずステータス = 2 で起動し，オフセットはありません。

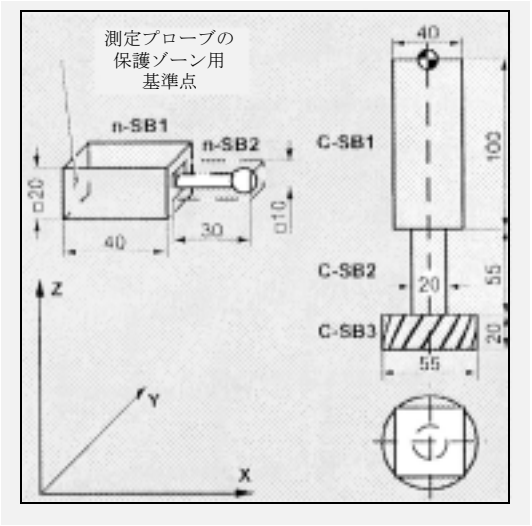
保護ゾーンの多重起動

保護ゾーンは，複数のチャンネルで同時に有効となりえます（例えば，2 つの向かい合った側面がある心押し台)。

その保護ゾーンは，すべてのジオメトリ軸が参照されている場合のみ監視されます。以下の規則が適用されます :

- 当該保護ゾーンは単一チャンネルの種々のオフセットと同時に起動することはできません。
- マシン向け保護ゾーンは双方のチャンネルで同じ方向になっていなければなりません。

プログラミング例



フライス盤に関しては，フライスと測定プローブが衝突する可能性があるため，それを監視しなければなりません。測定プローブの位置は，当該機能起動時にオフセットで定義しなければなりません。

このため，以下の保護ゾーンを定義します :

- 測定プローブホルダ (n-SB1) と測定プローブそのもの (n-SB1) の両方に対するマシン別およびワーク向け保護ゾーン。
- フライスホルダ (c-SB1)，カッタシャンク (c-SB2) およびフライスそのもの (c-SB3) に対するチャンネル別およびツール向け保護ゾーン。

保護ゾーンの向きはすべて Z 方向です。

当該機能起動時における測定プローブの基準点の位置は，X = -120，Y = 60 および Z = 80 でなければなりません。

DEF INT PROTECTB	補助変数の定義
保護ゾーンの定義 G17	方向設定
NPROTDEF(1,FALSE,3,10,-10) G01 X0 Y-10 X40 Y10 X0 Y-10 EXECUTE(PROTECTB)	保護ゾーン n-SB1
NPROTDEF(2,FALSE,3,5,-5) G01 X40 Y-5 X70 Y5 X40 Y-5 EXECUTE(PROTECTB)	保護ゾーン n-SB2
CPROTDEF(1,TRUE,3,0,-100) G01 X-20 Y-20 X20 Y20 X-20 Y-20 EXECUTE(PROTECTB)	保護ゾーン c-SB1
CPROTDEF(2,TRUE,3,-100,-150) G01 X0 Y-10 G03 X0 Y10 J10 X0 Y-10 J-10 EXECUTE(PROTECTB)	保護ゾーン c-SB2
CPROTDEF(3,TRUE,3,-150,-170) G01 X0 Y-27,5 G03 X0 Y27,5 J27,5 X0 Y27,5 J-27,5 EXECUTE(PROTECTB)	保護ゾーン c-SB3

保護ゾーンの起動：

NPROT(1,2,-120,60,80)	オフセットと同時に保護ゾーン n-SB1 起動
NPROT(2,2,-120,60,80)	オフセットと同時に保護ゾーン n-SB2 起動
CPROT(1,2,0,0,0)	オフセットと同時に保護ゾーン c-SB1 起動
CPROT(2,2,0,0,0)	オフセットと同時に保護ゾーン c-SB2 起動
CPROT(3,2,0,0,0)	オフセットと同時に保護ゾーン c-SB3 起動

5 特殊移動コマンド

5.1 コード化された位置 CAC, CIC, CDC, CACP, CACN へのアプローチ方法

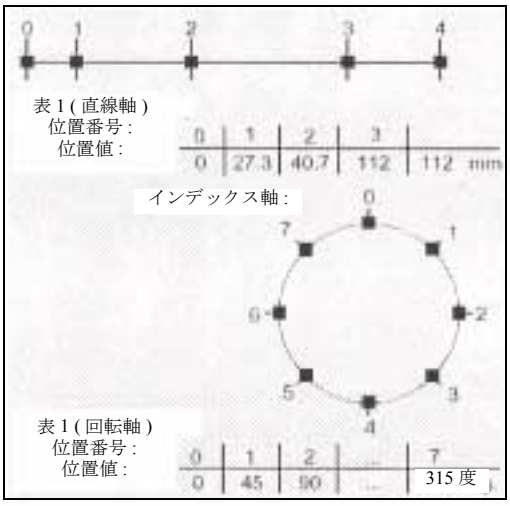
コマンドの説明

CAC(n)	コード化された位置に絶対的にアプローチする
CIC(n)	正の方向 (+) または負の方向 (-) に n スペース増分して，コード化された位置にアプローチする
CDC(n)	可能なかぎり最短ルートを介して，コード化された位置にアプローチする (回転軸のみ)
CACP(n)	正の方向に，コード化された位置に絶対的にアプローチする (回転軸のみ)
CACN(n)	負の方向に，コード化された位置に絶対的にアプローチする (回転軸のみ)
(n)	各軸についての，位置 No 1, 2, ... 最大 60 個所

動作

マシンデータの中にある 2 軸専用位置テーブルに，最大 60 (0 ～ 59) 個所の位置を入力することができます。

代表的な位置テーブルの例は，図を参照してください。



追加説明

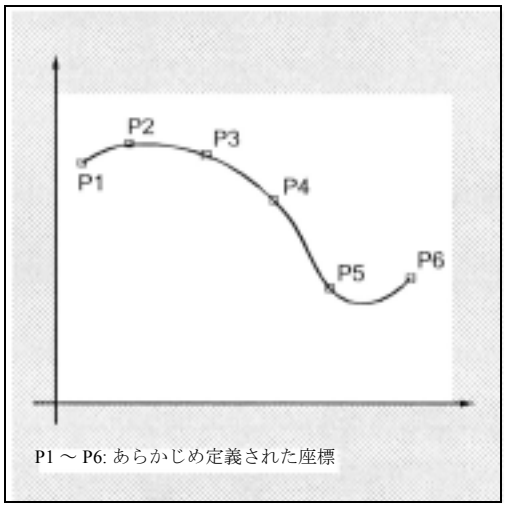
軸が 2 点間にある場合，CIC (...) で，相対位置コマンドに応答して，移動しません。

絶対位置値を用いて最初の移動コマンドをプログラムすることを常にお薦めします。

プログラミング例

N10 FA[B]= 300	位置決め軸 B 用フィード
N20 POS[B]= CAC (10)	コード化された位置 10 (絶対位置) にアプローチ
N30 POS[B]= CIC (-4)	現在の位置から 4 スペース戻る

5.2 スプライン補間



P1 ～ P6: あらかじめ定義された座標

概要

スプライン補間機能を使って，滑らかなカーブに沿って，一連の点を関連づけることができます。たとえば，スプラインは，デジタル化された一続きの点を使ってカーブを作るために用いることができます。

異なる補間効果を示す特性の異なる数タイプのスプラインがあります。ユーザは，スプラインのタイプを選ぶだけでなく，異なるパラメータの範囲を操作することもできます。希望するパターンを得るには，通常，幾度か試みてみることが必要です。

プログラミング

ASPLINE X Y Z A B C 、

BSPLINE X Y Z A B C 、

CSPLINE X Y Z A B C

機能

スプラインのプログラミング時，カーブに沿った一続きの点を結びます。

3 タイプのスプラインから 1 タイプを選ぶことができます :

-A スプライン (アキマスプライン)

-B スプライン (均一でない , 合理的な基準スプライン ,NURBS)

-C スプライン (立方スプライン)

追加説明

A, B および C のスプラインは，モーダル的に有効で，モーションコマンドのグループに所属しています。ツール径補正が適用できます。衝突監視は，平面投影に実装されます。

スプラインのグルーピングで補間すべき軸は，コマンドの SPLINEPATH を使って，選択されます (詳しくは，下記のとおりです)。

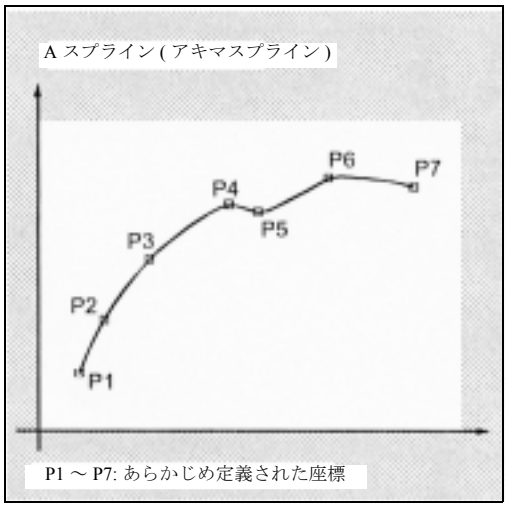
動作

A スプライン

A スプライン （アキマスプライン）は，中間点を正確に通ります。事実上は，不適切な揺動はありませんが，補間点での連続カーブが作られません。

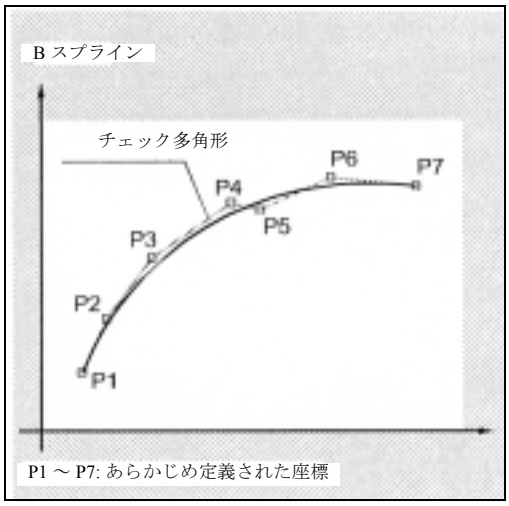
アキマスプラインはローカルです。すなわち，補間点を変更すると最大 6 近接点に限り影響します。

したがって，このタイプのスプラインの主なアプリケーションは，デジタル化された点の補間です。追加条件は，アキマスプラインについてプログラムできます（詳しくは下記を参照のこと）。度数 3 の多項式は，補間用に使用します。



B スプライン

B スプラインの場合，プログラムされた位置は，補間点ではなく，スプラインの単なるチェック点です。すなわち，カーブはこれらの点に向かって描かれますが，直接それらの点は通りません。



これらの点をつなぐ線は，スプラインのチェック多角形を形成します。B スプラインは，彫りのある表面でツールパスを定義するためには最適な手段です。B スプラインの主な目的は，CAD システムのインタフェースとして使用することです。度数 3 の B スプラインは，連続的にカーブして移行するにもかかわらず，揺動はまったくありません。

プログラムされた補足条件 (詳しくは，下記を参照のこと) は，B スプラインに影響しません。B スプラインは , 始点および終点でチェック多角形に対する接線です。

重み :

重みは，すべての補間点についてプログラムできます。

プログラミング :

PW = n

値の範囲 :

0 <= n <= 3; 0.0001 の段階で

影響 :

n > 1　　チェックポイントは，カーブに，さらに大きな「力」を加える。

n < 1　　チェックポイントは，さらに小さい「力」を加える。

スプライン度数 :

度数 3 の多角形を標準として使用しますが，度数 2 の多角形も可能です。

プログラミング :

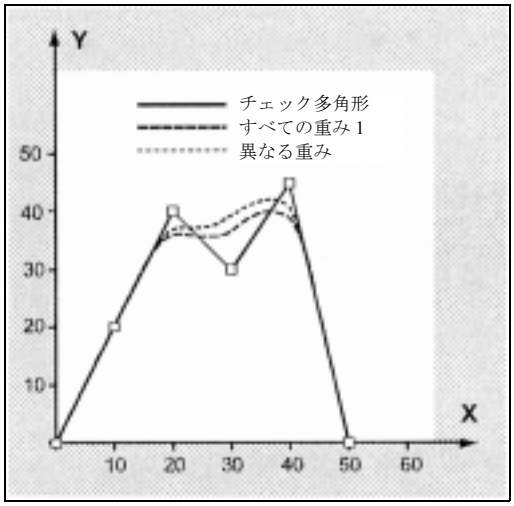
SD = 2

ノード間距離 :

ノード間距離は，制御装置内部で正しく計算されますが，システムでは，ユーザプログラムされたノード間距離を処理することもできます。

プログラミング :

PL = パス寸法に関して，値の範囲



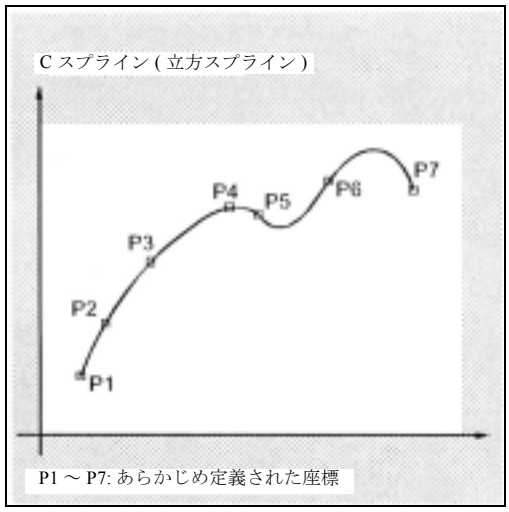
B スプラインの例 :

すべての重み 1	異なる重み	チェック多角形
N10 G1 X0 Y0 F300 G64	N10 G1 X0 Y0 F300 G64	N10 G1 X0 Y0 F300 G64
N20 BSPLINE	N20 BSPLINE PW=0.3	N20 ; 省略
N30 X10 Y20	N30 X10 Y20 PW=2	N30 X10 Y20
N40 X20 Y40	N40 X20 Y40	N40 X20 Y40
N50 X30 Y30	N50 X30 Y30 PW=0.5	N50 X30 Y30
N60 X40 Y45	N60 X40 Y45	N60 X40 Y45
N70 X50 Y0	N70 X50 Y0	N70 X50 Y0

C スプライン

アキマスプラインと対照的に，立方スプライン (C スプライン) が中間点で連続的にカーブします。ただし，予期せず発振する傾向があります。分析的に計算されたカーブに沿って補間点がある場合には，C スプラインが使用できます。度数 3 の多角形を使用します。

このスプラインはローカルではありません。すなわち，補間点を変更すると，多数のブロックに影響します（効果が次第に減少するにつれて）。



補足条件

次の補足条件は，アキマスプラインと立方スプラインのみに適用します (A スプライン および C スプライン)。

上記のスプラインカーブの遷移応答 (開始と終了) は，それぞれ 3 つのコマンドから成る 2 グループの命令を介してセットできます。

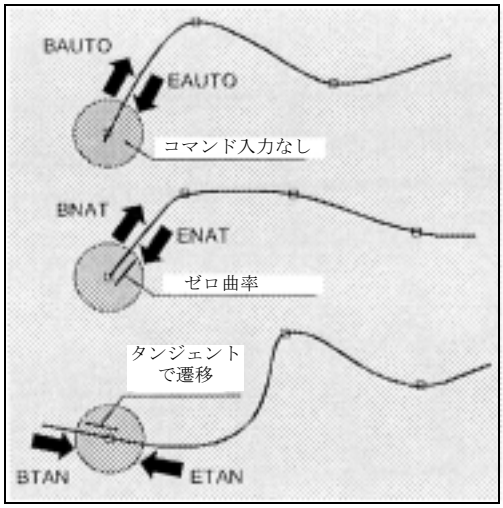
コマンドの説明

スプラインカーブの開始 :

BAUTO	コマンド入力なし ; 最初の点の位置によって開始が決まる
BNAT	ゼロ曲率
BTAN	前のブロックにタンジェント遷移 (初期設定)

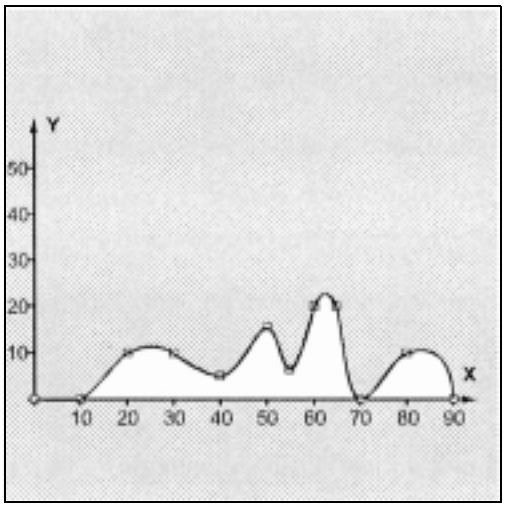
スプラインカーブの終了 :

EAUTO	コマンド入力なし ; 最後の点の位置によって終了が決まる
ENAT	ゼロ曲率
ETAN	次のブロックにタンジェント遷移 (初期設定)



例

C スプライン , 開始および終了時のゼロ曲率



N10 G1 X0 Y0 F300	
N15 X10	
N20 BNAT ENAT	C スプライン , 開始および終了時にゼロ曲率
N30 CSPLINE X20 Y10	
N40 X30	
N50 X40 Y5	
N60 X50 Y15	
N70 X55 Y7	
N80 X60 Y20	
N90 X65 Y20	
N100 X70 Y0	
N110 X80 Y10	
N120 X90 Y0	
N130 M30	

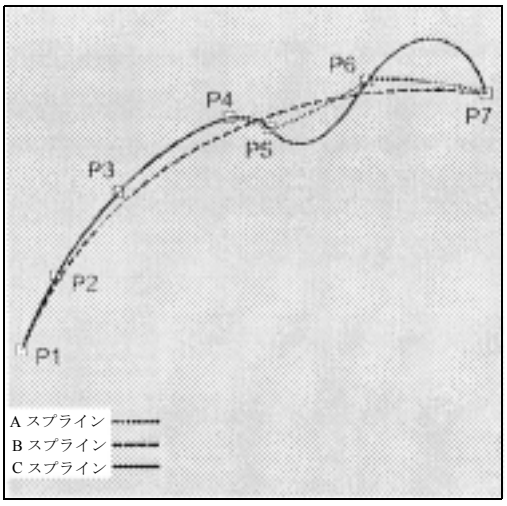
どのスプラインがどんな働きをするのか？

同じ補間点を持つ 3 タイプのスプラインの比較 :

A スプライン (アキマスプライン)

B スプライン (ベジェスプライン)

C スプライン (立方スプライン)



スプライングループ

最大 8 つのパス軸が，スプライン補間グルーピングに関連付けることができます。SPLINEPATH 命令は，どの軸をスプラインに関連付けるべきかを定義します。この命令は，独立したブロックにプログラムされます。SPLINEPATH が明示的にプログラムされていない場合には，チャンネル内の始めから 3 つまでの軸が，スプライングルーピングとして移動します。

プログラミング

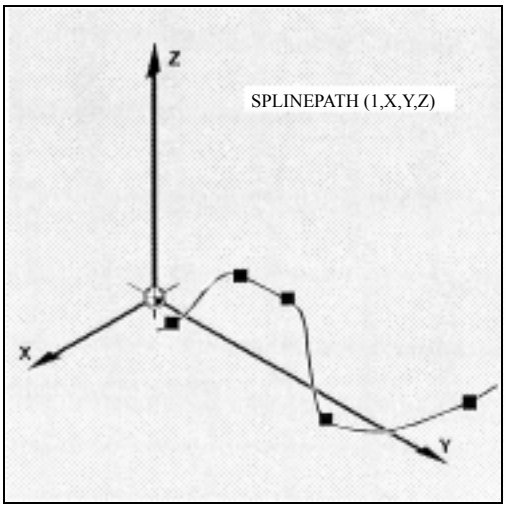
SPLINEPATH(n,X,Y,Z,...)

説明

SPLINEPATH(n,X,Y,Z,...)	n = 1, 固定値 X,Y,Z,... パス軸名称

例

3 つのパス軸を持つスプライングルーピング



N10 G1 X10 Y20 Z30 A40 B50 F350	
N11 SPLINEPATH(1,X,Y,Z)	スプライングルーピング
N13 CSPLINE BAUTO EAUTO X20 Y30 Z40 A50 B60	C スプライン
N14 X30 Y40 Z50 A60 B70	
...	補間点
N100 G1 X... Y...	スプライン補間の選択解除

スプラインのための設定

G コードの ASPLINE，BSPLINE および CSPLINE は，スプラインでブロック終りを結びます。

このためには，一連のブロック (終点) が同時に計算されなければなりません。

計算用のバッファサイズは，10 ブロックが標準です。

すべてのブロック情報が，スプライン終点であるとは限りません。

しかし，制御装置には，10 ブロックから一定数のスプライン終点ブロックが必要です。

各スプラインについては下記のとおりです :

A スプライン :	各 10 ブロックから少なくとも 4 ブロックは，スプライン用ブロックでなければならない。この中には，コメント用ブロックやパラメータ計算は含まない。
B スプライン :	各 10 ブロックから少なくとも 6 ブロックは，スプライン用ブロックでなければならない。この中には，コメント用ブロックやパラメータ計算は含まない。
C スプライン :	少なくとも 10 ブロックから，マシンデータ $MC_CUBIC_SPLINE_BLOCKS+1 の内容は，スプライン用ブロックでなければならない (標準例では 9 もある)。点の数は，必ず，スプラインセグメントを計算するのに使われるマシンデータ $MC_CUBIC_SPLINE_BLOCKS (標準値 8) に入力しなければならない。

許容値を超える場合やスプライン関連軸の 1 つが，位置決め軸としてプログラムされると，アラームが出力されます。

5.3 コンプレッサ COMPON/ COMPCURVE

原則として，CAD/CAM システムは，プログラムされた精度を満たす直線ブロックを規定します。複雑な輪郭の場合，かなりの量のデータになり，短いパスセクションになります。コンプレッサの場合，これらの一定数 (最大 10) 短いパスセクションがまとまって，１つのパスセクションを形成することができます。

モーダル G コードの COMPON あるいは COMPCURV は "NC block compressor" を起動します。直線補間の場合，この機能は，多数の直線ブロックを一まとめにして（数は 10 に制限される），度数 3 の多項式 (COMPON)　または 度数 5 の多項式 (COMPCURV) を使って，マシンデータを介して指定されたエラー許容範囲内で，それらの直線ブロックにアプローチします。このように，NC は，多数の小さいモーションブロックではなく，１つの大きいモーションブロックを処理します。

この圧縮操作は，直線ブロック (G1) においてのみ実行できます。これは他の任意のタイプの NC インストラクション，たとえば，パラメータ計算によって，補助機能出力によって，中断されます。

圧縮されるべきブロックには，ブロック番号，G1，軸アドレス，フィードおよびコメントだけを入れることができます。この順番は強制的です。変数は使用できません。

G コード COMPON の場合，ブロック遷移は速度だけが一定ですが，関連軸の加速は，ブロックの遷移時に飛び越しができます。こうすると，マシンの揺動を増やすことができます。

G コード COMPCURV の場合，ブロック遷移は一定に加速されます。これによって，ブロック遷移時にすべての軸の速度と加速が滑らかになります。

プログラミング

COMPON/COMPCURV	コンプレッサを起動する
COMPOF	コンプレッサを停止する

機械メーカー

3 つのマシンデータが，コンプレッサ機能に用意されます :

- $MC_COMPRESS_BLOCK_PATH_LIMIT
  最大パス長がセットされます。このパスに沿ったすべてのブロックが圧縮に適しています。
  あまり大きなブロックは圧縮されません。
- $MA_COMPRESS_POS_TOL
  公差は軸ごとにセットできます。生成されたスプラインカーブは，プログラムされた終点から，設定した値以上はずれません。この値を高く設定すればするほど，多くのブロックを圧縮できます。
- $MC_COMPRESS_VELO_TOL
  アクティブなコンプレッサでの最大許容パスフィードずれは，FLIN および FCUB に関連づけてプリセットできます。

例

N10 COMPON	あるいは COMPCURV, コンプレッサがオン
N11 G1 X0.37 Y2.9 F600	G1 は，終点とフィードの前にプログラムしなくてはならない。
N12 X16.87 Y-4.698 N13 X16.865 Y-4.72 N14 X16.91 Y-4.799 ...	
N1037 COMPOF ...	コンプレッサオフ

すべてのブロックは，簡単な構文を満たす場合に圧縮されます。

e.g.

N19 X0.103 Y0. Z0.

N20 X0.102 Y-0.018

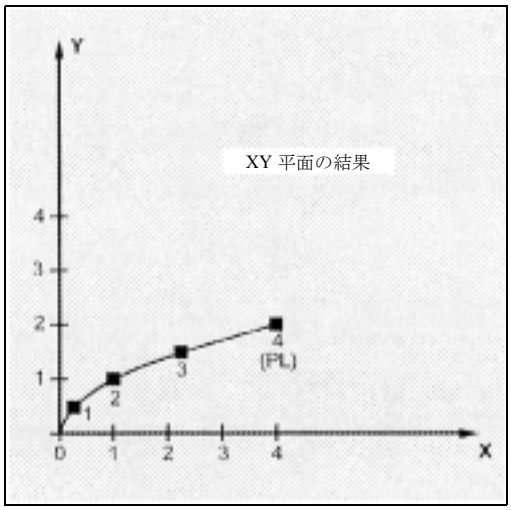
N21 X0.097 Y-0.036

N22 X0.089 Y-0.052

N23 X0.078 Y-0.067

たとえば，C=100 や A=ACNC のような拡張アドレスは圧縮されません。.

5.4 多項式補間 - POLY



制御装置は，すべての選択されたパス軸が機能に従って作動しているカーブ（パス）を移動することが可能です。(最大度数 3 の多項式)

多項式機能を表すのに用いる方程式は，通例，次のとおりです :

$f(p)= a_0 + a_1 p + a_2 p^2 + a_3 p^3$

これらの文字には次のような意味があります :

a_n: 一定の係数

p: パラメータ

これらの係数に具体的な値を割当てると，たとえば，線，放物線，冪関数のような多様なカーブ形状を生成することができます。

係数を $a_2 = a_3 = 0$ として設定すると，たとえば，$f(p) = a_0 + a_1 p$ では直線を作ることができます。

意味 :

a_0 = 前のブロックの終りにある軸位置

a_1 = 定義エリア (PL) の終りにある軸位置

定義

多項式補間 (POLY) は，Real タイプのスプライン補間の一つではありません。スプラインセクションが直接プログラムできる外部生成スプラインカーブをプログラムするためのインタフェースとして機能することを主な目的とします。

補間のこのモードによって，多項式の係数を計算するタスクを NC は行なわなくて済みます。係数が CAD システムやポストプロセッサによって直接与えられる場合には，最適です。

多項式補間は，G0，G1，G2，G3，A スプライン，B スプラインおよび C スプラインと共に，最初の G グループに所属します。つまり，これがアクティブの場合は，多項式の構文をプログラムする必要はありません : つまり，これらの名称と終点でプログラムされている軸のみが，その終点まで直線で移動します。すべての軸がこのようにしてプログラムされている場合，制御装置は，G1 がプログラムされているかのように応答します。

多項式補間は，G グループ (たとえば G0, G1) のほかのコマンドで停止します。

多項式の係数

PO 値 (PO[]=) は，1 つの軸についてすべての多項式の係数を指定します。値は，コンマで区切られており，多項式の度数に従って指定されます。ブロック内の異なる軸については，異なる多項式の度数がプログラムできます。

プログラミング

POLY PO[X]=(xe1,a2,a3) PO[Y]=(ye1,b2,b3) PO[Z]=(ze1,c2,c3) PL=n

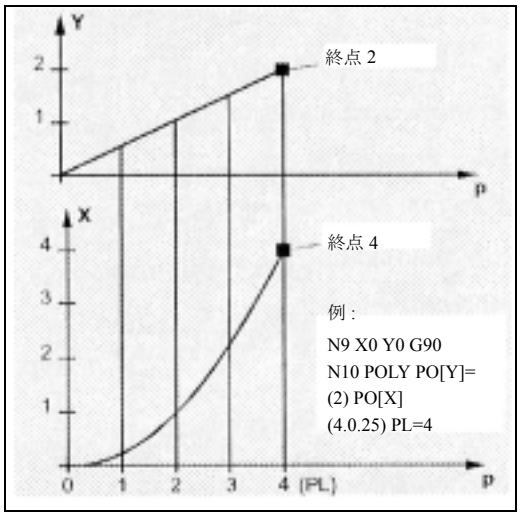
説明

POLY	POLY を含むブロックで多項式の補間を起動
PO[ ]=(...,...,...)	終点と多項式の係数
xe, ye, ze	関連軸用終点の指定 ; パス寸法に関する値の範囲
a2, a3	係数 a_2 および a_3 の値をプログラムする ; パス寸法に関する値の範囲。いずれの場合も最新の係数は，ゼロに等しい場合は省略できる。
PL	多項式が定義されるパラメータインターバルの長さ (機能 f(p) の定義範囲)。インターバルは, 常に，0 で始まる。p は，0 と PL の間の値にセットできる。PL の理論値の範囲は，PL: 0.0001 ... 99 999.9999 である。PL 値は，それがプログラムされるブロックに適用される。PL 値がプログラムされていない場合は，PL=1 が適用される。

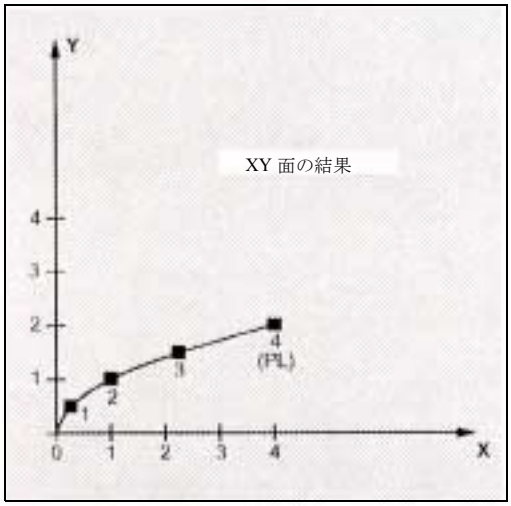
例

N10 G1 X... Y... Z... F600	
N11 POLY PO[X]=(1,2.5,0.7) -> -> PO[Y]=(0.3,1,3.2) PL=1.5	多項式補間がオン
N12 PO[X]=(0,2.5,1.7) PO[Y]=(2.3,1.7) PL=3 ... N20 M8 H126 ...	
N25 X70 PO[Y]=(9.3,1,7.67) PL=5	軸の混合設定
N27 PO[X]=(10.2.5) PO[Y]=(2.3)	PL 値がプログラムされていない ; PL=1 が適用
N30 G1 X... Y... Z.	多項式補間がオフ
...	

X/Y 面のカーブの例



N9 X0 Y0 G90 F100
N10 POLY PO[Y]=(2) PO[X]=(4,0.25) PL=4



例外的な分母の多項式

コマンド PO[]=(...) を使って，ジオメトリ軸に共通の分母多項式（軸名称の指定なしに）をプログラムすることができます。つまり，ジオメトリ軸のモーションは，2 つの多項式の商として補間されます。

このプログラミングのオプションを用いると，たとえば円錐などの形（円，楕円，放物線，双曲線）を正確に表すことができます。

例

POLY G90 X10 Y0 F100	ジオメトリ軸が，X10, Y0 の位置まで，直線で移動する
PO[X]=(0,-10) PO[Y]=(10) PO[]=(2,1)	ジオメトリ軸が，X0, Y10 まで四分円で移動する

分母の多項式の一定の係数 (a_0) は，常に 1 であると仮定します。指定された終点は，G90/G91 には依存しません。

上記の例から得られた結果は，下記のとおりです :

$X(p)=10(1-p^2)/(1+p^2)$ および $Y(p)=20p/(1+p^2)$

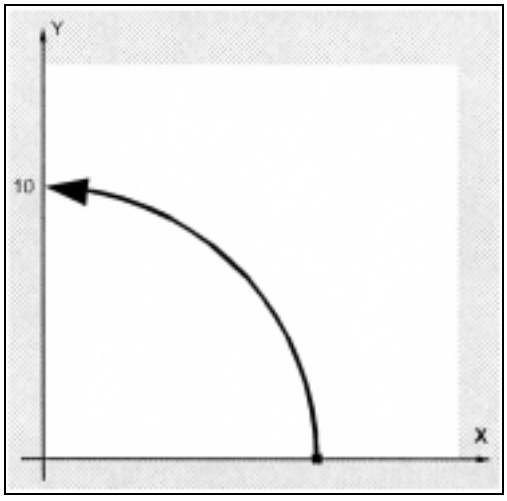
ここでは $0<=p<=1$

プログラムされた始点の結果として，終点，係数 a2 および PL=1, 中間値は次のとおりです :

分子 $(X)=10+0*p-10p^2$

分子 $(Y)=0+20*p+0*p^2$

分母 $= 1+2*p+1*p^2$



ゼロのある分母多項式が，多項式補間のアクティブ時に，インターバル [0,PL] 内にプログラムされると，アラームが出力されます。分母多項式は，スペシャル Axis の動作には影響を与えません。

追加説明

ツール径補正は，多項式補間と関連付け，G41, G42 で起動できます。また，直線補間モードあるいは円形補間モードと同じ方法で使用できます。

5.5 タッチトリガプローブ MEAS, MEAW での測定

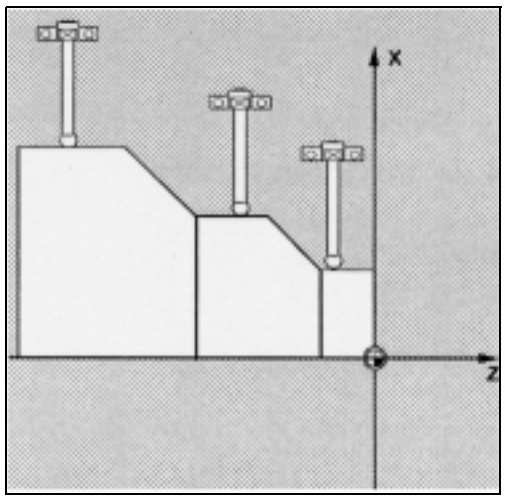
プログラミング

MEAS= ± 1	G... X... Y... Z...	(残移動量削除で上昇エッジの場合の +1/+2 測定)
MEAS= ± 2	G... X... Y... Z...	(残移動量削除で下降エッジの場合の -1/-2 測定)
MEAW= ± 1	G... X... Y... Z...	(残移動量削除で上昇エッジの場合の +1/+2 測定)
MEAW= ± 2	G... X... Y... Z...	(残移動量削除で下降エッジの場合の -1/-2 測定)

コマンドの説明

MEAS= ± 1	測定入力 1 のとき，プローブ 1 で測定
MEAS= ± 2*	測定入力 2 のとき，プローブ 2 で測定
MEAW= ± 1	測定入力 1 のとき，プローブ 1 で測定
MEAW= ± 2*	測定入力 2 のとき，プローブ 2 で測定

* 最大値 2 の入力は，構成レベルに依存する



動作

NC ブロックにプログラムされたすべての軸について，プローブの切換えエッジに一致する位置が得られ，各軸専用のメモリに書き込まれます。最大 2 つのプローブがインストールできます。

測定結果

これらの軸について，下記の変数に従って，測定結果が得られます :

- マシン座標系 $AA_MM[axis]
- ワーク座標系 $AA_MW[axis]

これらの変数の読取り時，内部前処理停止が，内部的に生成されます。前処理停止は，プログラム中の適切な位置で STOPRE を用いてプログラムされなければなりません。これ以外の場合，システムは，誤って，値を読み取ります。

測定状態

ステータス変数 $AC_MEA[n] (n= プローブ番号) は，タッチトリガプローブの切換えステータスがプログラム内で評価される必要がある場合に，調べることができます :

0	測定ジョブは実行されない
1	測定ジョブが順調に完了した (プローブは切換えステータスを持つ)

プローブがプログラム実行中にずれると，この変数は 1 にセットされます。測定ブロックの最初に，変数はプローブの開始ステータスに対応するように自動的にセットされます。

測定ブロック MEAS, MEAW のプログラミング

コマンド MEAS は，補間モードと関連付けしてプログラムされるとき，ワークの実際位置にアプローチし，同時に測定値を記録します。実際位置とセットポイント位置の移動距離が削除されます。

プログラムされた位置に常にアプローチしなければならない専用測定タスクの場合には，MEAW 機能を使用します。

MEAS および MEAW は，モーションコマンドで，１つのブロック中にプログラムします。フィードと補間のタイプ (G0, G1, ...) を，用意された測定タスクに合うように選択しなければなりません ; これは，軸の数にも適用されます。

例 :

N10 MEAS=1 G1 F1000 X100 Y730 Z40

最初の測定入力および直線補間時に，プローブを用いた測定ブロック。自動的に先読み停止となります。

測定値の記録

ブロック中の移動したすべてのパス軸と位置決め軸の位置（軸の最大数は，制御の構成によって決まります）を記録します。MEAS の場合，モーションは，プローブの切換え後，定義された方法でブレーキがかかります。

コメント

GEO 軸が，測定ブロックにプログラムされると，そのときのすべての GEO 軸についての測定値が記録されます。

ポジション値が変化したすべての軸の測定値が記録されます。

追加説明

MEAS および MEAW の機能は，モーダルでなくアクティブです。

5.6 拡張された測定機能 MEASA, MEAWA, MEAC

プログラミング

MEASA[axis]=(mode, TE1,..., TE 4)	残移動量削除して測定
MEAWA[axis]=(mode, TE 1,..., TE 4)	残移動量削除せずに測定
MEAC[axis]=(mode, measurement memory, TE 1,...TE4)	残移動量削除せずに連続測定

説明

Axis	測定に使用されるチャンネル軸名称
Mode	以下のモードと測定モードからなる運転モードのための 2 桁の設定 (ones decade) and ...0 測定ジョブを強制終了 ...1 Mode 1: 同時に起動できる最大 4 トリガイベント ...2 Mode 2: 連続して起動できる最大 4 トリガイベント ...3 Mode 3: 連続して起動できる最大 4 トリガイベント ただし，START 点では，トリガイベント 1 の監視はしない 測定系 (10 の位) 0... または設定なし : アクティブな測定系 1... 測定系 1 2... 測定系 2 3... 両方の測定系
TE 1...4	トリガイベント : 1 上昇エッジ , プローブ 1 -1 下降エッジ , プローブ 1 2 上昇エッジ , プローブ 2 -2 下降エッジ , プローブ 2
測定メモリ	FIFO の数（循環ストレージ）

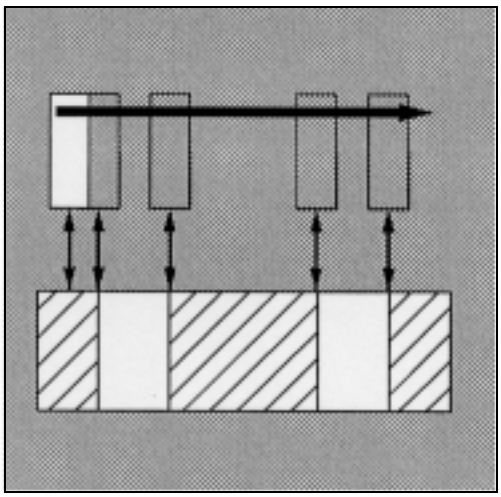
機能

この測定系では，いくつかのプローブといくつかの測定系で，軸方向に測定を行なうことができます。

MEASA, MEAWA のプログラム時，測定ランごとに，プログラムされた軸について最大 4 測定値が得られ，トリガイベントにしたがってシステム変数に保存されます。

MEASA と MEAWA は，非モーダルコマンドです。

連続測定運転は，MEAC で実行できます。この場合，測定結果は，FIFO 変数に保存されます。測定ランごとの測定値最大数は，MEAC も 4 です。



動作

パートプログラムかまたはシンクロナイズドアクションから測定値をプログラムすることができます (セクション 10 参照)。

一定時間有効であるのは，各軸につき 1 つの測定ジョブに限られることに注意してください。

追加説明

- フィードは，用意された測定タスクに合うように調整しなければなりません。
- MEASA および MEAWA の場合，結果の正確さは，同じタイプのトリガイベントはただ 1 つで，また，位置コントローラサイクルごとに起こるトリガイベントが 4 つ以下である送り速度でのみ保証できます。
- MEAC で連続測定する場合，補間サイクルと位置制御サイクルとの比が，8 : 1 を超えてはなりません。

トリガイベント

トリガイベントは，プローブの番号と測定信号のトリガ基準（上昇エッジあるいは下降エッジ）から成ります。

各測定につき，アドレスで呼び出されたプローブの最大 4 つのトリガイベントが処理できます。すなわち，2 つの測定信号エッジで最大 2 プローブを処理できます。
トリガイベントの最大数と処理順番とは，選択されたモードによって決まります。

同じトリガイベントは，測定ジョブで一度だけプログラムすることができます (モード 1 にのみ適用されます)!

運転モード

モード設定の 1 番目の数字は，必要な測定系を選択します。1 つの測定系だけがインストールされると，もう一つ別の測定系がプログラムされても，インストールされたシステムが自動的に選択されます。

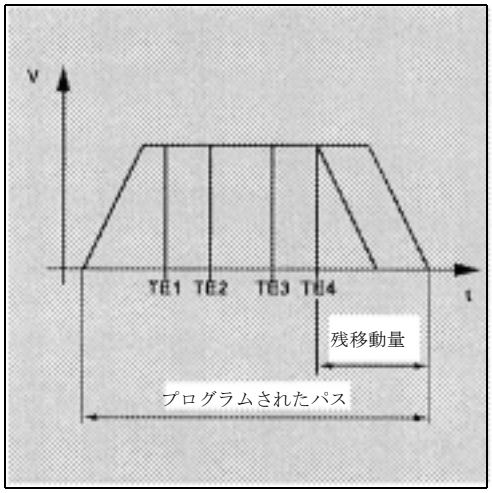
2 番目の数字，すなわち測定モードでは，測定プロセスが，関連のある制御システムの性能に適用されます :

- モード 1: トリガイベントは，それらが起こる時間順に評価されます。このモードの選択時，1 つのトリガイベントだけが 6 軸のモジュールについてプログラムできます。2 つ以上のトリガイベントが指定されると，モードの選択は，モード 2 に自動的に切替わります（メッセージなし）。
- モード 2: トリガイベントは，プログラムされた順に評価されます。
- モード 3: トリガイベントは，プログラムされた順に評価されますが，START 時にはトリガイベントの監視はありません。

追加説明

2 つの測定系が使用される場合は，2 つ以下のトリガイベントがプログラムできます。

残移動量削除を用いた測定と用いない測定



コマンド MEASA のプログラム時，すべての必要な測定値が記録されるまで，残移動量は削除されません。

プログラム位置に常にアプローチしなければならない専用測定タスクの場合は，MEAWA 機能を使用します。

MEASA および MEAWA は，同一ブロック中にプログラムすることができます。

MEASA/MEAWA が同一ブロックに MEAS/MEAW でプログラムされると，エラーメッセージが出力されます。

- MEASA は，シンクロナイズドアクションにはプログラムできません。
  他の方法として，MEAWA に残移動量削除を加えて，シンクロナイズドアクションとしてプログラムできます。
- MEAWA を使って，測定ジョブをシンクロナイズドアクションから開始する場合，測定値はマシン座標でのみ使用できることになります。

MEASA, MEAWA の測定結果

測定結果は，次のシステム変数に従って使用できます :

- マシン座標システムにおいて :

$AA_MM1[axis]	トリガイベント 1 でのプログラムされた測定系の測定値
...	...
$AA_MM4[axis]	トリガイベント 4 でのプログラムされた測定系の測定値

- ワーク座標系 :

$AA_WM1[axis]	トリガイベント 1 でのプログラムされた測定系の測定値
...	...
$AA_WM4[axis]	トリガイベント 4 でのプログラムされた測定系の測定値

追加説明

前述の変数を読み取るとき，内部的な前処理停止は生成されません。

前処理停止は，適切な位置で，STOPRE (セクション 9.7) を使ってプログラムしなければなりません。

そうでなければ，不正な値が読み込まれることになります。

ジオメトリ軸について，軸測定を開始しようとする場合，残りすべてのジオメトリ軸について明示的にプログラムしなければなりません。

これは，関連する軸についても適用されます。

例 : N10 MEASA[X]=(1,-1) MEASA[Y]=(1,-1)

MEASA[Z]=(1,-1) G01 X100 F100;

2 つのエンコーダでの測定ジョブ

測定ジョブを，2 つの測定系で実行すると，関連軸の両方の測定系可能な 2 つのトリガイベントがそれぞれ得られます。したがって，確保された変数の割当ては，次のとおり，プリセットされます :

$AA_MM1[axis]	または	$AA_MW1[axis]	トリガイベント 1 での，エンコーダ 1 の測定値
$AA_MM2[axis]	または	$AA_MW2[axis]	トリガイベント 1 での，エンコーダ 2 の測定値
$AA_MM3[axis]	または	$AA_MW3[axis]	トリガイベント 2 での，エンコーダ 1 の測定値
$AA_MM4[axis]	または	$AA_MW4[axis]	トリガイベント 2 での，エンコーダ 2 の測定値

測定プローブステータスは，$A_PROBE[n] を介して，読み取ることができます。

n= プローブ

1== ずれのあるプローブ

0== ずれのないプローブ

MEASA, MEAWA のための測定ジョブステータス

プローブ切換えステータスが，プログラムで評価される必要がある場合は，n = プローブ番号を使って，$AC_MEA[n] を介して，測定ジョブステータスの信号を送ることができます。

ブロック中にプログラムされているプローブ "n" のすべてのトリガイベントがいったん起こると，この変数は，"1" のステージに切り換ります。そうでない場合は，この値は 0 になります。

測定が，シンクロナイズドアクションから開始される場合，$AC_MEA は更新されません。この場合，新しい PLC ステータス信号 DB(31-48) DBB62 ビット 3 または同等の変数 $AA_MEAACT["Axis"] の信号が送られます。

意味 : $AA_MEAACT==1: 測定有効

$AA_MEAACT==0: 測定無効

参照： 計測機能 (M5)/FB/

連続測定 MEAC

MEAC についての測定値は，マシン座標系で使用でき，プログラムされている FIFO[n] メモリ (循環メモリ) に保存されます。

2 つのプローブを測定用に構成する場合，2 つ目のプローブの測定値は，この目的専用に構成された FIFO[n+1] メモリに別に保存されます（マシンデータに定義される）。FIFO メモリは，循環メモリで，ここで，循環の原則にしたがって $AC_FIFO 変数 に測定値が書き込まれます。

参照： /PGA/ Section 10,
Synchronized actions

追加説明

- FIFO の内容は循環メモリから一度だけ読み取ることができます。この測定データを多様に使用する場合には，ユーザデータにバッファされなければなりません。
- FIFO メモリについての測定値の数が，マシンデータに定義された最高値を超える場合，測定は自動的に終了されます。

- エンドレスの測定プロセスは，周期的に測定値を読み出すことによって，実装できます。この場合，データは，新しい測定値の読み込みと同じ周波数で読み出されなければなりません。

プログラミング例

モード 1 で，残移動量削除を用いて測定
(時間順に評価)

a) 1 つのエンコーダを使って

...	
N100 MEASA[X] = (1,1,-1) G01 X100 F100	アクティブなエンコーダを使ってモード 1 で測定。トラベルパスのプローブ 1 から X = 100 まで，上昇／下降エッジを用いて測定信号を待つ。
N110 STOPRE	前処理停止
N120 IF $AC_MEA[1] == FALSE gotof END	測定の結果をチェックする。
N130 R10 = $AA_MM1[X]	1 番目にプログラムされたトリガイベント (上昇エッジ) で得られた測定値を保存する
N140 R11 = $AA_MM2[X]	2 番目にプログラムされたトリガイベントで（下降エッジ）得られた測定値を保存する

プログラミング例

b) 2 つのエンコーダを使って

...	
N200 MEASA[X] = (31,1-1) G01 X100 F100	エンコーダを 2 つとも使用して，モード 1 で測定。トラベルパスのプローブ 1 から X = 100 まで，上昇／下降エッジで測定信号を待つ。
N210 STOPRE	前処理停止
N220 IF $AC_MEA[1] == FALSE gotof END	測定の結果をチェックする。
N230 R10 = $AA_MM1[X]	上昇エッジで測定系 1 の測定値を保存する
N240 R11 = $AA_MM2[X]	上昇エッジで測定系 2 の測定値を保存する
N250 R12 = $AA_MM3[X]	下降エッジで測定系 1 の測定値を保存する
N260 R13 = $AA_MM4[X]	下降エッジで測定系 2 の測定値を保存する
N270 END:	

モード 2 で，残移動量削除を使った測定 (プログラムされた順に評価)

...	
N100 MEASA[X] = (2,1,-1,2,-2) G01 X100 F100	アクティブなエンコーダを使ってモード 2 で測定。次の順番で，測定信号を待つ : X = 100 までのトラベルパスにおいて，プローブ 1 の下降エッジ , プローブ 1 の上昇エッジ , プローブ 2 の上昇エッジ , プローブ 2 の下降エッジ
N110 STOPRE	前処理停止
N120 IF $AC_MEA[1] == FALSE gotof	プローブ 1 を使った測定の結果をチェックする
PROBE2	
N130 R10 = $AA_MM1[X]	1 番目にプログラムされたトリガイベントで得られた測定値を保存する (上昇エッジプローブ 1)
N140 R11 = $AA_MM2[X]	2 番目にプログラムされたトリガイベントで得られた測定値を保存する (下降エッジプローブ 1)
N150 PROBE2:	
N160 IF $AC_MEA[2] == FALSE gotof END	プローブ 2 を使った測定の結果をチェックする
N170 R12 = $AA_MM3[X]	3 番目にプログラムされたトリガイベントで得られた測定値を保存する (上昇エッジプローブ 2)
N180 R13 = $AA_MM4[X]	4 番目にプログラムされたトリガイベントで得られた測定値を保存する (下降エッジプローブ 2)
N190 END:	

プログラミング例

モード 1 での連続測定 (時間順に評価)

最大 100 測定値までの測定

...	
N110 DEF REAL MEASVALUE[100]	
N120 DEF INT INDEX = 0	
N130 MEAC[X] = (1,1,-1) G01 X1000 F100	アクティブなエンコーダを使って，モード 1 で測定し，$AC_FIFO1 の下で，測定値を保存し，X = 1000 までのトラベルパス上の，プローブ 1 から，下降エッジを用いて，測定信号を待つ。
N135 STOPRE	
N140 MEAC[X] = (0)	軸位置への到達時，測定を終了する。
N150 R1 = $AC_FIFO1[4]	パラメータ R1 の累積測定値の数を保存する。
N160 FOR INDEX = 0 TO R1-1	
N170 MEASVALUE[INDEX] = $AC_FIFO1[0]	$AC_FIFO1 から測定値を読み出し，保存する。
N180 ENDFOR	

10 測定値の後，残移動量削除をして測定

...	
N10 WHEN $AC_FIFO1[4]>=10 DO MEAC[x]=(0) DELDTG(x)	残移動量削除
N20 MEAC[x]=(1,1,1,-1) G01 X100 F500	
N30 MEAC[X]=(0)	
N40 R1 = $AC_FIFO1[4]	測定値の数
...	

下記のプログラミングのエラーが検出され，適切に
指示されます :

- MEASA/MEAWA is programmed with MEAS/

  MEAW in the same block (MEASA/MEAWA が，同一ブロック中に，MEAS/MEAW と共にプログラムされている)

  例：
  N01 MEAS=1 MEASA[X]=(1,1) G01 F100
  POS[X]=100

- MEASA/MEAWA with number of parameters <2 or >5 (パラメータ <2 または >5 の数を用いて MEASA/MEAWA)

  例 :
  N01 MEAWA[X]=(1) G01 F100 POS[X]=100

- MEASA/MEAWA with trigger event not equal to 1/ -1/ 2/ -2 (1/ -1/ 2/ -2 に等しくないトリガイベントを用いて MEASA/MEAWA)

  例 :
  N01 MEASA[B]=(1,1,3) B100

- MEASA/MEAWA with invalid mode (無効なモードを用いて MEASA/MEAWA)

  例 :
  N01 MEAWA[B]=(4,1) B100

- MEASA/MEAWA with trigger event programmed twice (2 度プログラムされているトリガイベントで MEASA/MEAWA)

  例 :
  N01 MEASA[B]=(1,1,-1,2,-1) B100

- MEASA/MEAWA and missing GEO axis (MEASA/MEAWA および GEO 軸が不明)

  例 :
  N01 MEASA[X]=(1,1) MEASA[Y]=(1,1) G01 X50
  Y50 Z50 F100

- Unsuitable measuring job with GEO axes (GEO 軸で不適切な測定ジョブ)

  例 :
  N01 MEASA[X]=(1,1) MEASA[Y]=(1,1)
  MEASA[Z]=(1,1,2) G01 X50 Y50 Z50 F100 GEO
  axis X/Y/Z

5.7 OEM ユーザ専用機能

OEM アドレス

OEM アドレスの意味は，OEM ユーザが決定します。

可能な OEM アドレスの機能は，コンパイルサイクルによって組み込まれています。5 つの OEM アドレスは確保されます。

アドレス識別子は，セット可能です。

OEM アドレスは，任意のブロックにプログラムできます。

OEM 補間

OEM ユーザは，2 つの追加補間を定義することができます。これらの機能は，コンパイルサイクルによって組み込まれます。

G 機能 (OEMIPO1, OEMIPO2) の名称は，OEM ユーザがセットします。

OEM アドレスは，(上記参照) OEM 補間専用に使用できます。

確保された G グループ G800 - 819

それぞれ 10 個の OEM G 機能をもった 2 つの G グループが，OEM ユーザのために確保されます。

これらによって，OEM ユーザによって組み込まれた機能は，外部アプリケーションのためにアクセスできます。

機能およびサブプログラム

OEM ユーザは，また，あらかじめ定義された機能とサブプログラムを，パラメータ転送によって，セットアップすることもできます。

5.8 設定可能なパス基準 , SPATH, UPATH

プログラミング

SPATH	FGROUP 軸用パス基準は，アーク長さである
UPATH	カーブパラメータは，FGROUP 軸用パス基準である

概要

多項式補間中，ユーザは，速度を決定する FGROUP 軸および他のパス軸との 2 種類の関係を必要とすることがあります :

後者は，次のとおり制御されます。

- FGROUP 軸のパスに同期する，あるいは
- カーブパラメータに同期する

希望した応答を選択して，プログラムするための G code (SPATH, UPATH) が使用できます。

機能

多項式補間中，（ここでは，より厳格な意味での多項式補間 (POLY)，すべてのスプライン補間タイプ (ASPLINE, BSPLINE, CSPLINE) およびコンプレッサ (COMPON, COMPCURV) を用いた直線補間について述べます）すべてのパス軸の位置 i が，多項式 p i (U) によってプリセットされます。カーブパラメータ U は，0 から 1 まで，NC ブロック内を移動するので，それが標準になります。

プログラムされたパスフィードが関連する軸は，FGROUP を使って，パス軸から選択できます。ただし，多項式補間中は，これらの軸がパス S 上を一定速度で補間するということは，通常，カーブパラメータ U の変更が一定でないことを意味します。

したがって，FGROUP に含まれていない軸については，パスに従う 2 通りの方法があります :

1. それらの軸がパス S (SPATH) と同期して移動するか，あるいは
2. FGROUP 軸 (UPATH) のカーブパラメータ U と同期して移動する

両タイプのパス補間は，異なるアプリケーションで使用し，G コードの SPATH および UPATH を介して切替えることができます。

UPATH と SPATH も，パス動作で，F ワード多項式 (FPOLY, FCUB, FLIN) の関係を決定します。

丸めの拡張

パス軸のすべてが FGROUP に含まれない場合は，含まれない軸がブロック遷移時に突然の速度変更を起こすことがしばしばあります。ブロックチェンジ時，減速によって，制御装置はこのブロックチェンジの範囲を，MD 32300:MAX_AX_ACCEL および MD 32310:_MAX_ACCEL_OVL_FACTOR にセットされた許容値に制限することができます。この減速は，パス軸の指定された位置関係を丸めることによって防ぐことができます。

- G641 での丸め
  丸めは，G641 によってモーダルで起動され，丸め半径 ADIS (もしくは，急速移動での ADISPOS) をパス機能について指定します。
  ブロック変更点周辺のこの半径内のパス関係を解除し，力学的に最適なパスがそれに取って代わることを制御装置が任意に行ないます。
  制限事項 : すべての軸について使用できる ADIS 値は，1 つだけです。

参照： プログラミング 編基本説明書 ,
セクション 5 軌跡移動動作

- G642 での丸め
  軸公差の丸めは，G642 によってモーダル的に起動します。丸めは，定義された ADIS エリア内では起こりません。その代わりに，MD 33100: COMPRESS_POS_TOL にセットされた軸公差に必ず従います。
  残りの可能な機能は，G641 と同じです。

参照： 連続送り，イグザクトストップモードと先読み /FB/, (B1)

補足条件

セットされたパス基準は，次の場合は重要ではありません。

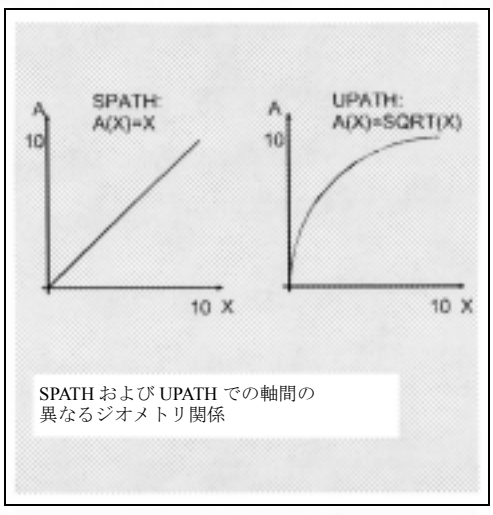
- 直線および円補間の場合
- ねじ切り中の場合
- すべての軸が FGROUP に含まれる場合

起動

FGROUP に含まれない軸のパス基準は，45 番目の G コードグループに含まれた 2 つのコマンド SPATH および UPATH を介してセットされます。コマンドはモーダルです。SPATH がアクティブの場合，軸はパスに同期して移動します ; UPATH がアクティブの場合，移動はカーブパラメータに同期します。

プログラミング例

次のプログラム例は，両方のタイプのモーション制御間の差を示しています。いずれの場合も初期設定 FGROUP(X,Y,Z) がアクティブです。



SPATH および UPATH での軸間の異なるジオメトリ関係

N10 G1 X0 A0 F1000 SPATH
N20 POLY PO[X]=(10, 10) A10
あるいは
N10 G1 X0 F1000 UPATH
N20 POLY PO[X]=(10, 10) A10

ブロック N20 中で，FGROUP 軸のパス S は，カーブパラメータ U の二乗によって決まります。

したがって，SPATH あるいは UPATH が有効であるかどうかにしたがって，X のパスに沿って同期した軸 A について異なる位置が発生します。

電源オン，モード変更，リセット，ブロックサーチ，Repos のときの制御応答

リセット後，MD 20150:GCODE_RESET_VALUES [44] を介して定義された G コードが有効です (45 番目の G コードグループ)。

丸めのタイプについての基本設定値は，MD 20150: GCODE_RESET_VALUES [9] (10 番目の G コードグループ) にセットされます。

マシン／オプションデータ

リセット後に有効な G コードグループ値は，マシンデータ MD 20150:GCODE_RESET_VALUES [44] を介して決定します。

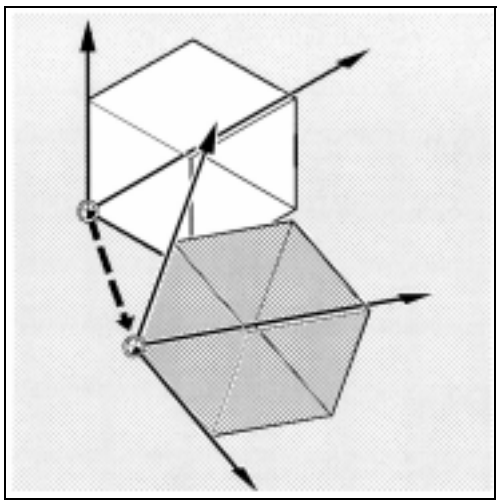
現在インストールされているものとの互換性を保つために，SPATH を初期値としてセットします。

丸めのタイプについての基本設定値は，MD 20150: GCODE_RESET_VALUES [9] (10 番目の G コードグループ) にセットします。

軸のマシンデータ MD 33100:COMPRESS_POS_TOL は，コンプレッサ機能および G642 を用いた丸めについての公差を含んでいます。

6 フレーム

6.1 フレーム変数を使用する座標変換



フレーム変数を使用する座標変換の定義

プログラミング編基本説明書ですでに説明したプログラミングオプションの他に、定義済みのフレーム変数を使用して座標系を定義することもできます。

座標系

次の座標系が定義されます：

MCS: マシン座標系

BCS 基本座標系

BOS: 基本原点系

SZS: 設定可能なゼロ系

WCS: ワーク座標系

定義済みのフレーム変数とは？

定義済みのフレーム変数とは、制御言語にその使用および影響がすでに定義されていて、NC プログラムで処理することができる用語機構をいいます。

フレーム変数の例：

- 基本フレーム（基本オフセット）
- 設定可能なフレーム
- プログラム可能なフレーム

フレーム変数／フレーム関係

フレームの値をフレーム変数に割当てることによって、座標変換を起動することができます。

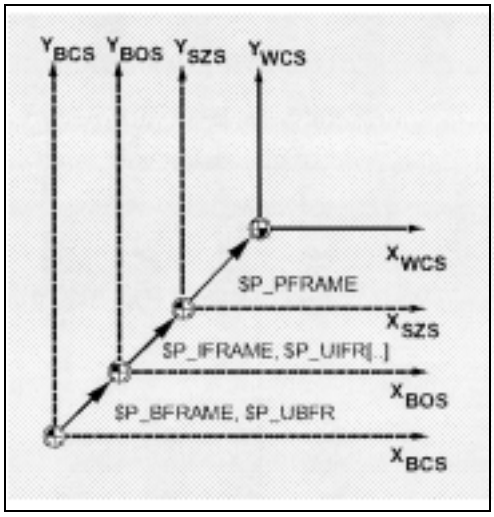
例 : $P_PFRAME=CTRANS(X,10)

フレーム変数 :

$P_PFRAME の意味 : 現在プログラム可能なフレーム

フレーム :

CTRANS(X,10) の意味 : 10 mm の単位でプログラム可能な X 軸のゼロオフセット¥



現在値の読み出し

パートプログラムに定義済みの変数を使用して、現在の実際値を読み出すことができます :

$AA_IM[axis]	MCS の現在値を読む
$AA_IB[axis]	BCS の現在値を読む
$AA_IBN[axis]	BOS の現在値を読む
$AA_IEN[axis]	SZS の現在値を読む
$AA_IW[axis]	WCS の現在値を読む

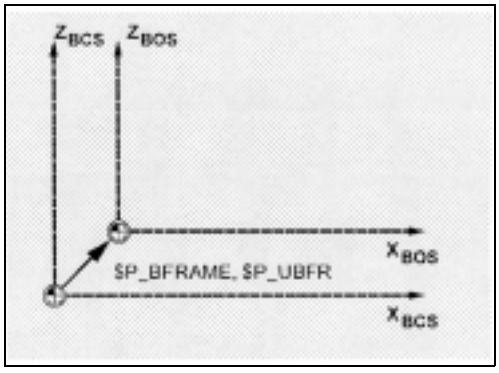
定義済みのフレーム変数の概要

$P_BFRAME

基本座標系 (BCS) と 基本原点系 (BOS) 間の基準を設定する現在の基本フレーム変数

$P_UBFR を使用して描かれた基本フレームがプログラム内で即時に有効になるためには、次のいずれかが必要となります

- G500, G54...G599 をプログラミングすること
- $P_UBFR を使用して $P_BFRAME を描くこと



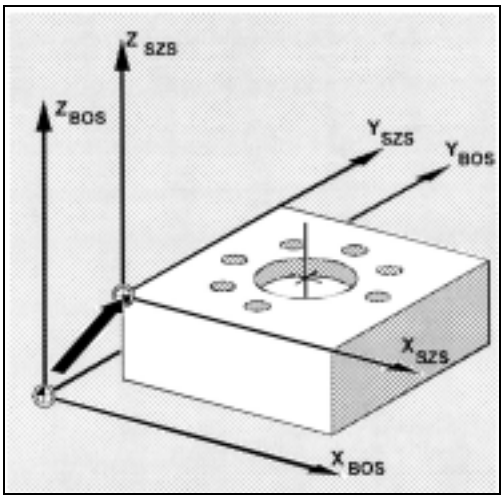
$P_IFRAME

基本原点系 (BOS) と設定可能なゼロ系 (SZS) 間の基準を設定する現在の設定可能なフレーム変数

$P_IFRAME は $P_UIFR[$P_IFRNUM] に対応します

例えば、After G54 が設定されると、

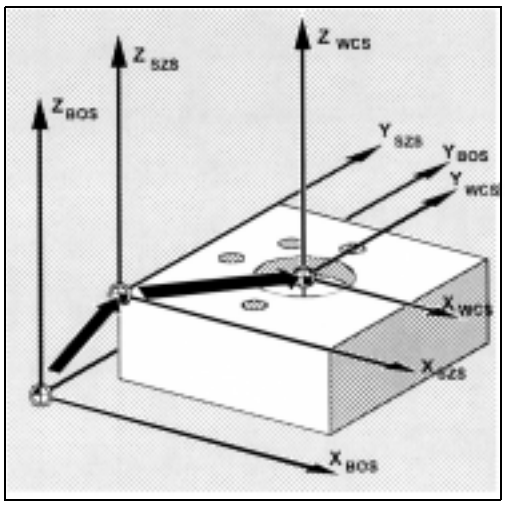
$P_IFRAME には、G54 によって定義された移動、回転、スケーリング、およびミラーリングが含まれます。



$P_PFRAME

設定可能なゼロ系 (SZS) とワーク座標系 (WCS) 間の基準を設定する現在の設定可能なフレーム変数

$P_PFRAME は、TRANS/ATRANS, ROT/AROT, SCALE/ASCALE, MIRROR/AMIRROR の指令結集からなるフレームか, CTRANS, CROT, CMRROR, CSCALE をプログラム可能なフレームに割当てることにより得られたフレームからなります。



$P_ACTFRAME

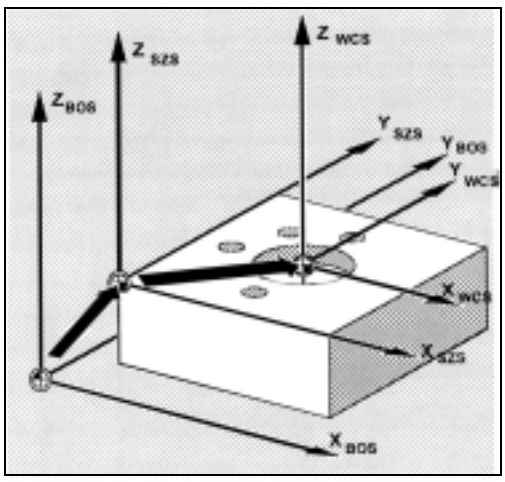
CSCALE をプログラム可能な FRAME に割当てることにより得られたフレームが含まれます。現在の基本フレーム変数 $P_BFRAME , 現在の設定可能なフレーム変数

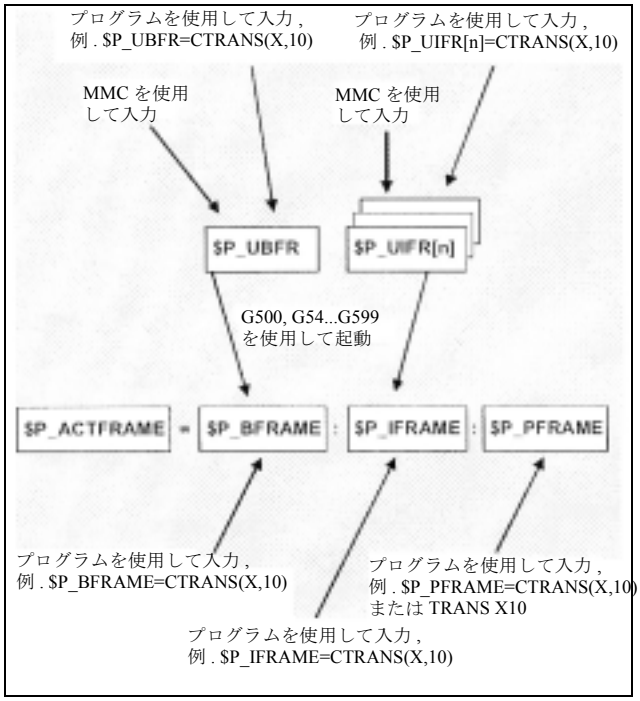
$P_IFRAME および現在のプログラム可能なフレーム変数 $P_PFRAME をチェーニングした結果得られた現在の総フレーム

$P_ACTFRAME は、現在有効なワークゼロを描きます。

$P_IFRAME, $P_BFRAME または $P_PFRAME が変更されると、$P_ACTFRAME が計算し直されます。

$P_ACTFRAME は、
$P_BFRAME:$P_IFRAME:$P_PFRAME に対応します。





MD 20110 RESET_MODE_MASK が次のように設定されている場合、基本フレームと設定可能なフレームは Reset （リセット）すると有効になります :

ビット 0=1, ビット 14=1 --> $P_UBFR
（基本フレーム）有効

ビット 0=1, ビット 5=1 --> $P_UIFR [$P_UIFRNUM]
（設定可能なフレーム）有効

事前定義された設定可能なフレーム $P_UBFR

基本フレームは $P_UBFR を使用してプログラミングされますが、それと同時にパートプログラムで有効にはなりません。

次のような場合、$P_UBFR を使用してプログラミングされた基本フレームが計算に含まれています

- Reset が起動されていて、ビット 0 と 14 が MD RESET_MODE_MASK に設定され、さらに
- 命令 G500, G54...G599 がすでに実行されている。

事前定義された設定可能なフレーム $P_UIFR[n] を使用すると、パートプログラムより設定可能なゼロオフセット G54 ～ G599 を読み出したり書き込むことができます。

これらの変数は、$P_UIFR[n] と呼ばれる一次元配列のタイプ FRAME を作成します。

G コマンドへの割当て

標準 $P_UIFR[0]...$P_UIFR[4] あるいは同じ意味を持つ 5 G コマンド（G500 と G54 ～ G57）として、5 つの設定可能なフレームをセットし、そのアドレスに値を保存することができます。

$P_IFRAME=$P_UIFR[0]　　G500 に対応する

$P_IFRAME=$P_UIFR[1]　　G54 に対応する

$P_IFRAME=$P_UIFR[2]　　G55 に対応する

$P_IFRAME=$P_UIFR[3]　　G56 に対応する

$P_IFRAME=$P_UIFR[4]　　G57 に対応する

マシンデータを使用して、フレーム数を変更することができます :

$P_IFRAME=$P_UIFR[5] G505 に対応する

... ...

$P_IFRAME=$P_UIFR[99] G599 に対応する

これにより、最大 100 の座標系を作成でき、さまざまなプログラムで大域的にコールすることができます（例えば、種々の治具用のゼロ点としてなど)。

フレーム変数は、必ず NC プログラム内の個別の NC ブロックに設定してください。

例外 : G54, G55, ... を使用する場合の設定可能なフレームのプログラミング

6.2 フレーム変数／フレームへの割当て値

値を直接割り当てる、フレームをチェーニングする、あるいはフレームを NC プログラムの他のフレームに割り当てることができます。

直接値の割当て

プログラミング

$P_PFRAME=CTRANS (X, 軸値 , Y, 軸値 , Z, 軸値 , ...)

$P_PFRAME=CROT (X, 角度 , Y, 角度 , Z, 角度 , ...)

$P_PFRAME=CSCALE (X, スケール , Y, スケール , Z, スケール , ...)

$P_PFRAME=CMIRROR (X, Y, Z)

プログラミング $P_BFRAME は、$P_PFRAME と同様に実行される。

コマンドの説明

CTRANS	指定された軸の移動
CROT	指定された軸の周りの回転
CSCALE	指定された軸のスケールの変更
CMIRROR	指定された軸の方向転換

機能

これらの機能を使用すると、NC プログラムにフレームやフレーム変数を直接割り当てることができます。

動作

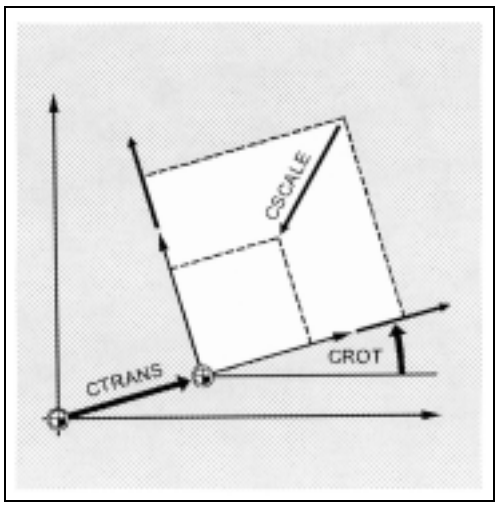
順番にいくつかの計算法則をプログラミングすることができます。

例 :

$P_PFRAME=CTRANS(...):CROT(...):CSCALE...

この場合、必ず、コマンドをコロンチェーン演算子 (...):(...) で区切って入力してください。

そうすることで、コマンドはまずリンクされ、その後プログラムされた順番に加法的に実行されます。



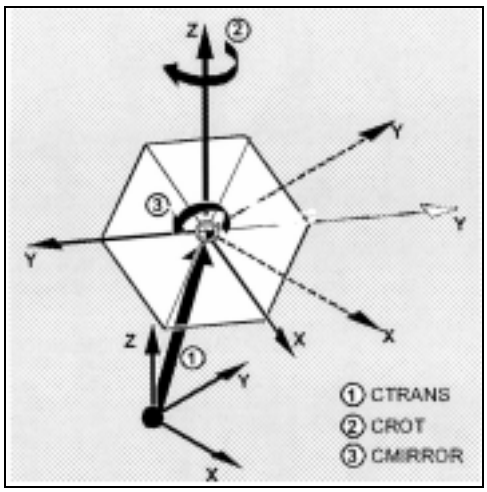
追加説明

上記に述べたコマンドを使用してプログラムされた値は、フレームに割当てられ、保存されます。

この値は、有効フレーム変数 $P_BFRAME または $P_PFRAME のフレームに割当てられて初めて、アクティブになります。

プログラミング例

現在プログラム可能なフレームに値を割当てると、移動、回転、およびミラーリングが起動されます。




```
N10 $P_PFRAME=CTRANS(X,10,Y,20,Z,5):CROT(Z,45):CMIRROR(Y)
```

フレーム要素の読取りおよび変更
プログラミング (例)

R10=$PUIFR[$P_UIFRNUM, X, RT]	X 軸回りの回転角度 RT を、現在設定可能なゼロオフセット $P_UIFRNUM から変数 R10 に割り当てます。
R12=$P_UIFR[25, Z, TR]	Z のオフセット値 TR を、セットフレーム No. 25 のデータ記録から変数 R 12 に割り当てます。
R15=$P_PFRAME[Y, TR]	現在プログラム可能なフレームの Y のオフセット値 TR を、変数 R15 に割り当てます。
$P_PFRAME[X, TR]=25	現在プログラム可能なフレームの X のオフセット値 TR を変更します。X25 が即時に適用されます。

コマンドの説明

$P_UIFRNUM	このコマンドで、現在有効な設定可能なゼロオフセットに対して自動的に基準が設定されます。
P_UIFR[n, ..., ...]	フレーム数 n を指定し、設定可能なフレーム数 n にアクセスします。
TR FI RT SC MI	読み取りされるまたは変更される要素を指定します：TR 変換、F1 変換微、RT 回転、SC スケール、MI ミラーリング。対応する軸も指定されます（例を参照してください）。

機能

この機能を使用すると、フレームの個々のデータ、例えば特定のオフセットデータや回転角度などにアクセスすることができます。

これらの値を修正し、別の変数に割当てることもできます。

動作

フレームの呼び出し

システム変数 $P_UIFRNUM を指定して、$P_UIFR または G54, G55, ... を使用するカレントのゼロオフセットにアクセスすることができます ($P_UIFRNUM には現在セットされているフレームの番号が含まれます)。

該当する番号 $P_UIFR[n] を指定すると、その他の保存されている全ての設定可能な $P_UIFR フレームがコールされます。

事前定義されたフレーム変数およびユーザ定義のフレームでは、$P_IFRAME のように名前を指定します。

データの呼び出し

軸名、ならびにアクセスしたりや変更したい値のフレーム要素は、角括弧で囲んで、例えば、[X, RT] や [Z, MI] のように書き込まれます。

完全フレームのリンク

完全フレームは、別のフレームに割当てることができます。

プログラミング (例)

DEF FRAME SETTING1 SETTING1=CTRANS(X,10)
$P_PFRAME=SETTING1

ユーザフレーム SETTING1 の値を現在プログラム可能なフレームに割当てます。

DEF FRAME SETTING4 SETTING4=$P_PFRAME
$P_PFRAME=SETTING4

現在プログラム可能なフレームは、一時的に保存されるので再コールできます。

追加説明

回転 RT の範囲

第 1 ジオメトリ軸の回転 : -180° から +180°

第 2 ジオメトリ軸の回転 : -89.999° から +90°

第 3 ジオメトリ軸の回転 : -180° から +180°

フレームチェーニング

プログラミング（例）

$P_IFRAME=$P_UIFR[15]:$P_UIFR[16]

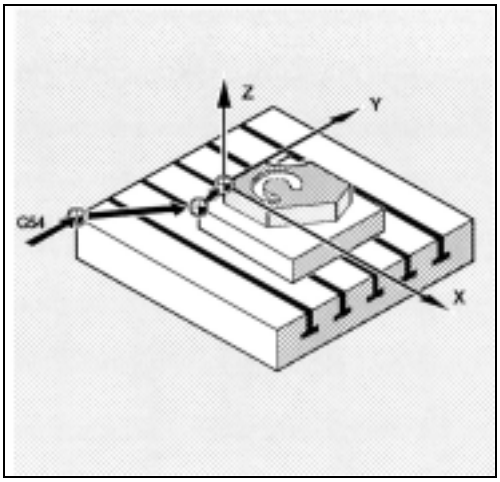
$P_UIFR[15] には、ゼロオフセット用のデータなどが含まれています。
$P_UIFR[16] のデータ、例えば回転用のデータなどは、順番に、加法的に処理されます。

$P_UIFR[3]=$P_UIFR[4]:$P_UIFR[5]

設定可能なフレーム 3 は、設定可能なフレーム 4 と 5 をチェーニングして作成されます。

機能

フレームチェーニングは、パレットに配置され、同一プロセスで加工されるいくつかのワークの記述に非常に適しています。



動作

フレームは、プログラムされた順序にチェーニングされます。フレーム要素（変換、回転など）は、順番に、加法的に実行されます。

フレーム要素には、パレットタスクの記述用の中間値しか含まれません。これらの値は、チェーニングされ、さまざまなワークゼロを作成します。

ただし、フレームは必ず、コロンチェーン演算子 (:) で相互にリンクされていなければなりません。

新規フレームの定義

プログラミング

DEF FRAME PALLET1

PALLET1=CTRANS(...):CROT(...)...

機能

上記の事前定義された設定可能なフレームの他に、新規フレームを作成することもできます。

新規フレームを作成するには、FRAME タイプ の変数を作成し、そこに選択した名前を割当てます。

動作

機能 CTRANS, CROT, CSCALE および CMIRROR を使用して、NC プログラムの使用するフレームに値を割当てます。

詳細については、前ページを参照してください。

6.3 粗／ファインオフセット

機能

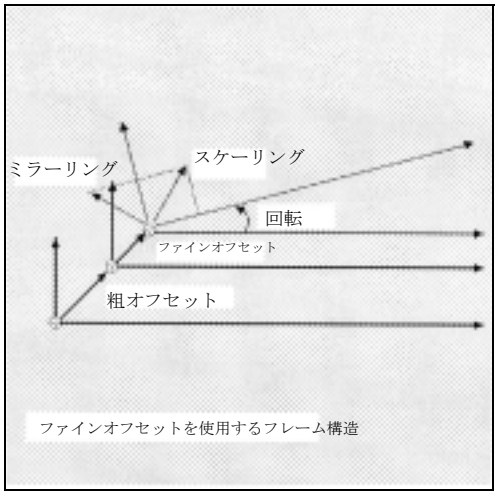
ファインオフセット

基本フレームならびに全ての設定可能なフレームのファインオフセットは、CFINE(X, ..,Y, ...) コマンドによってプログラミングすることができます。

粗オフセット

粗オフセットは、CTRANS(...) を使用して定義されます。

粗とファインのオフセットを合計すると、総オフセットになります。



ファインオフセットを使用するフレーム構造

プログラミング

```
$P_UBFR=CTRANS(x, 10) : CFINE(x, 0.1) : CROT(x, 45) ; チェーニングオフセット , ファインオフセット、および回転
$P_UIFR[1]=CFINE(x, 0.5, y, 1.0, z, 0.1)            ; 総オフセットは CFINE に上書きされる（粗オフセットを含む)。
```

要素仕様 F1 を通じて、ファインオフセットの個々の要素へアクセスできます。

プログラミング

```
DEF REAL FINEX                                      ; 変数 FINEX の定義
FINEX=$P_UIFR[$P_UIFRNUM, x, FI]                    ; 変数 FINEX を介してファインオフセットを読み出す
FINEX=$P_UIFR[3, X, FI]                             ; 変数 FINEX を介して第 3 フレームの X 軸のファインオフセットを読み出す
```

MD18600:

MM_FRAME_FINE_TRANS=1 の場合にのみ、ファインオフセットが可能です。

オペレータ入力によって変更されたファインオフセットは、対応するフレームが起動されて初めて有効になります、すなわち起動は G500, G54...G599 を介して行われます。フレームの起動されたファインオフセットは、そのフレームが有効である限りは有効になっています。

プログラム可能なフレームには、ファインオフセットはありません。ただし、プログラム可能なフレームがファインオフセットを持つフレームに割当てられている場合、総オフセットは、粗オフセットとファインオフセットを加算して設定します。プログラム可能なフレームを読み出すと、ファインオフセットは常にゼロとなります。

機械メーカ

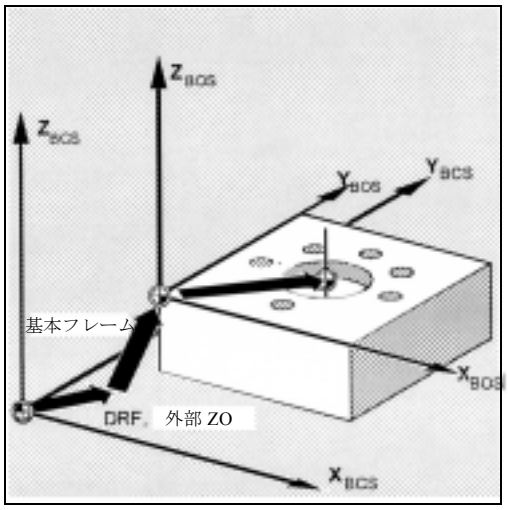
MD18600 MM_FRAME_FINE_TRANS を使用して、次の変数にファインオフセットを設定することができます:

0: ファインオフセットの入力もプログラミングもできない。

G58 と G59 は有効でない。

1: 設定可能なフレーム用のファインオフセット、基本フレーム、プログラム可能なフレーム、G58 および G59 が、入力／プログラミング可能である。

6.4 DRF オフセット



ハンドルを使用したオフセット , DRF

本セクションで述べた移動の他に、ハンドル（ DRF オフセット）を使用して、ゼロオフセットを定義することができます。

DRF オフセットは、基本座標系に影響を与えます。

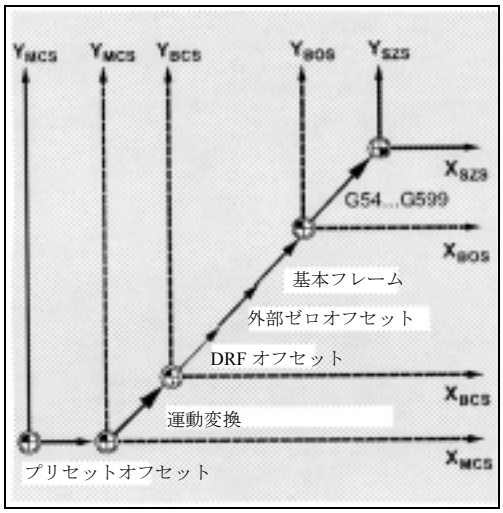
関連性については、図を参照してください。

詳細については、結合説明書 機能編を参照してください。

DRF オフセットのクリア , DRFOF

DRFOF は、チャンネルに割当てられた全ての軸のハンドルオフセットをクリアにします。DRFOF は、個別の NC ブロックにプログラムされます。

6.5 外部ゼロオフセット



外部ゼロオフセット

この方法を使っても、基本座標系とワーク座標系間でゼロポイントを移動させることができます。

外部ゼロオフセットを使用する場合、直線変換しか設定できません。

プログラミングオフセット値 , $AA_ETRANS

軸別システム変数を割当てると、オフセット値が設定されます。

オフセット値の割当て

$AA_ETRANS[axis]=$R_I$

R_I はタイプ REAL の算術変数で、新規の値を含んでいます。

通常、外部オフセットは PLC によってセットされますが、パートプログラムに指定することはできません。

パートプログラムに入力された値は、対応する信号が VDI インタフェース（ NCU-PLC インタフェース）でイネーブルになっていないと、有効になりません。

6.6 プログラミング PRESET オフセット、PRESETON

プログラミング

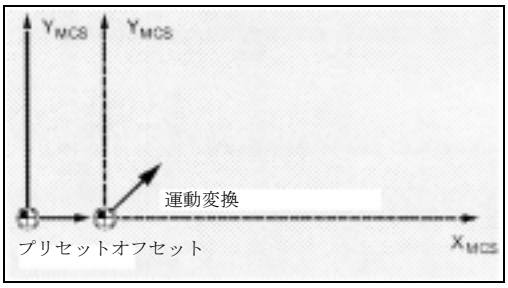
PRESETON(AXIS,VALUE,...)

コマンドの説明

PRESETON	実際値のセット
Axis	マシン軸パラメータ
Value	指定された軸に適用する新規の実際値

機能

例外として、新規にプログラムされた実際値を現在の位置（固定）の 1 つ以上の軸に割当てる必要がある場合もあります。



動作

実際値は、マシン座標系に割当てられます（この値がマシン軸を示します)。

例 :

N10 G0 A760

N20 PRESETON(A1,60)

軸 A は、位置 760 に移動します。位置 760 で、

マシン軸 A1 は新規の実際値 60 を割当てられます。

この点を基準に、新規の現在値系によって位置決めが行われます。

機能 PRESETON を使用すると、基準点が無効になります。したがって、この機能は、基準点を必要としない軸に対してのみ使用してください。

元のシステムが保存されている場合、G74 を使用して基準点にアプローチしてください（参照、セクション 2.3.1)。

6.7 フレームの停止

コマンドの説明

DRFOF	ハンドルオフセット (DRF) の停止（クリア）
G53	プログラム可能なフレームおよび全ての設定可能なフレームのノンモーダル停止
G153	プログラム可能なフレーム、基本フレームおよび全ての設定可能なフレームのノンモーダル停止
SUPA	全てのプログラム可能なフレーム、基本フレーム、全ての設定可能なフレーム、およびハンドルオフセット (DRF) のノンモーダル停止

追加説明

"zero frame"「ゼロフレーム」(軸指定なし) をプログラム可能なフレームに割当てると、プログラム可能なフレームはクリアになります。

例 :

$P_PFRAME=TRANS( )

$P_PFRAME=ROT( )

$P_PFRAME=SCALE( )

$P_PFRAME=MIRROR( )

6.8 空間上の 3 つの測定点に基づくフレーム計算 , MEAFRAME

MEAFRAME は、計測サイクルのサポートに使用される 840DI 言語の拡張子です。

機能

ワークが加工のために位置決めされると、デカルトマシン座標系に比例するその位置は通常、その目標位置を基準にシフトと回転が共に実行されます。

正確な加工や測定を行うために、部品の調整を費用をかけて行うか、あるいはパートプログラムに定義された動作を変更するか、そのいずれかを行う必要があります。

目標位置がわかっている空間の 3 点を測定して、フレームを確定することができます。測定は、背板に正確に固定されている特別な穴または球に接触している触角センサまたは光学センサを使用して行われます。

MEAFRAME 機能が、3 つの理想点とそれに対応する測定点を基にフレームを計算します。

測定座標を回転や変換を使って理想座標上にマップするには、測定点で作成した三角形を理想三角形と合同にする必要があります。これには、測定三角形を理想三角形に再形成するのに必要となる偏差平方和を最小にする補正アルゴリズムが使用されます。

実行ひずみを使用して測定の質を判定するため、MEAFRAME は実行ひずみを追加変数として戻します。

プログラミング

MEAFRAME(IDEAL_POINT,MEAS_POINT,FIT_QUALITY)

コマンドの説明

MEAFRAME	空間の 3 測定点のフレーム計算
IDEAL_POINT	理想点の 3 座標を含むリアルデータの 2 次元配列
MEAS_POINT	測定点の 3 座標を含むリアルデータの 2 次元配列
FIT_QUALITY	次の情報に戻る実変数の変数： -1: 理想点はほぼ直線に位置する : フレームの計算はできない。戻りフレーム変数にはニュートラルフレームが含まれる。 -2: 測定点はほぼ直線に位置する : フレームの計算はできない。戻りフレーム変数にはニュートラルフレームが含まれる。 -4: 回転マトリックスの計算はさまざまな理由で正の値にならない : 偏差（点間距離）の合計を使用して、測定した三角形を理想三角形と合同の三角形に再形成する。

アプリケーション例

```
; パートプログラム 1
;
DEF FRAME CORR_FRAME
;
; 測定点の設定
DEF REAL IDEAL_POINT[3,3] = SET(10.0,0.0,0.0, 0.0,10.0,0.0, 0.0,0.0,10.0)
DEF REAL MEAS_POINT[3,3] = SET(10.1,0.2,-0.2, -0.2,10.2,0.1, -0.2,0.2, 9.8); 試験用
DEF REAL FIT_QUALITY = 0
;
DEF REAL ROT_FRAME_LIMIT = 5; パート位置では最大 5° 回転を許容
DEF REAL FIT_QUALITY_LIMIT = 3; 理想三角形および ; 測定三角形には、最大 3 mm の偏差を許容
DEF REAL SHOW_MCS_POS1[3]
DEF REAL SHOW_MCS_POS2[3]
DEF REAL SHOW_MCS_POS3[3]
; =======================================================
;
N100 G01 G90 F5000
N110 X0 Y0 Z0
;
N200 CORR_FRAME=MEAFRAME(IDEAL_POINT,MEAS_POINT,FIT_QUALITY)
;
N230 IF FIT_QUALITY < 0
SETAL(65000)
GOTOF NO_FRAME
ENDIF
,
```

```
N240 IF FIT_QUALITY > FIT_QUALITY_LIMIT
SETAL(65010)
GOTOF NO_FRAME
ENDIF
;
N250 IF CORR_FRAME[X,RT] > ROT_FRAME_LIMIT; 第 1 RPY 角度の制限
SETAL(65020)
GOTOF NO_FRAME
ENDIF
;
N260 IF CORR_FRAME[Y,RT] > ROT_FRAME_LIMIT; 第 2 RPY 角度の制限
SETAL(65021)
GOTOF NO_FRAME
ENDIF
;
N270 IF CORR_FRAME[Z,RT] > ROT_FRAME_LIMIT; 第 3 RPY 角度の制限
SETAL(65022)
GOTOF NO_FRAME
ENDIF
;
N300 $P_IFRAME=CORR_FRAME; 設定可能なフレームを使用するプローブフレームの起動
;
; 理想位置にジオメトリ軸を位置決めしてフレームをチェックする
;
N400 X=IDEAL_POINT[0,0] Y=IDEAL_POINT[0,1] Z=IDEAL_POINT[0,2]
N410 SHOW_MCS_POS1[0]=$AA_IM[X]
N420 SHOW_MCS_POS1[1]=$AA_IM[Y]
N430 SHOW_MCS_POS1[2]=$AA_IM[Z]
;
N500 X=IDEAL_POINT[1,0] Y=IDEAL_POINT[1,1] Z=IDEAL_POINT[1,2]
N510 SHOW_MCS_POS2[0]=$AA_IM[X]
N520 SHOW_MCS_POS2[1]=$AA_IM[Y]
N530 SHOW_MCS_POS2[2]=$AA_IM[Z]
;
N600 X=IDEAL_POINT[2,0] Y=IDEAL_POINT[2,1] Z=IDEAL_POINT[2,2]
N610 SHOW_MCS_POS3[0]=$AA_IM[X]
N620 SHOW_MCS_POS3[1]=$AA_IM[Y]
N630 SHOW_MCS_POS3[2]=$AA_IM[Z]
;
N700 G500; 設定可能なフレームの停止、ゼロフレームを使用するプリセット ( 値はセットされない
)

6-204
;
NO_FRAME:
M0
M30
```

6.9 NCU グローバルフレーム

機能

NCU グローバルフレームは、各 NCU のチャンネルすべてに対して同時に使用します。NCU グローバルフレームは全てのチャンネルから読み出せて書き込みができます。

グローバルフレームを使用すると、オフセット、スケーリングおよびミラーリングをチャンネル軸およびマシン軸に適用することができます。

グローバルフレームを使用する場合、軸同士の幾何学上の関係は存在しません。したがって、回転の実行やジオメトリ軸識別子のプログラミングはできません。

- グローバルフレームを回転に使用することはできません。
  回転をプログラミングした場合、そのプログラミングは拒否され、アラーム "18310 Channel %1 block %2 frame: rotation not allowed" が出力されます。
- グローバルフレームとチャンネル別フレームをチェーニングすることができます。チェーニングの結果生じたフレームには、全ての軸の回転などのフレーム要素が全て含まれます。回転要素を持つフレームがグローバルフレームに割当てられると、その設定は拒否され、アラーム "Frame: rotation not allowed" が出力されます。

NCU グローバル基本フレーム : $P_NCBFR[n]

最大 8 までの NCU グローバル基本フレームを設定することができます。

機械メーカ

グローバル基本フレームの数は、マシンデータを使用して設定されます。(/FB/ K2 参照 , 軸、座標系、フレーム)

チャンネル別基本フレームは、同時に表示することができます。

グローバルフレームは、NCU の全てのチャンネルから読み出せて書き込みができます。グローバルフレームに書き込む際には、例えば Wait marks （待機マーク）(WAITMC) を使用するなど、チャンネル座標に注意を払ってください。

NCU グローバル設定可能なフレーム : $P_UIFR[n]

設定可能なフレーム G500,

G54...G599 の設定は全て、NCU グローバル設定か、またはチャンネル別設定のいずれかになります。

機械メーカ

MD 18601

MM_NUM_GLOBAL_USER_FRAMES を使用すると、全ての設定可能なフレームをグローバルフレームとして設定することができます。

参照： 軸構成及び座標系 (K2)

チャンネル軸識別子およびマシン軸識別子を、フレームプログラムコマンドの軸識別子として使用することができます。ジオメトリ識別子のプログラミングは拒否され、アラームが出力されます。

6.9.1 チャンネル別フレーム

機能

MD28081 MM_NUM_BASE_FRAMES を使用すると、チャンネルの基本フレーム数を設定できます。

標準設定では、チャンネルにつき最低 1 つの基本フレームが設定されます。1 つのチャンネルを最大 8 つの基本フレームがサポートします。8 つの基本フレーム以外に、そのチャンネルで 8 つの NCU グローバル基本フレームを使用することもできます。

設定可能なフレーム／基本フレームは、パートプログラムを介してならびに OPI を介して PLC から読み出すことや書き込むことができます。

グローバルフレームにはファインオフセットも使用できます。また、G53, G153,SUPA および G500 を介して、チャンネル別フレームを使用する場合と同様のグローバルフレームの抑止が起こります。

$P_CHBFR[n]

基本フレームは、システム変数 $P_CHBFR[n] を使用して読み出しと書き込みができます。基本フレームを書き込む場合、チェーニングされた総基本フレームは起動されません。（これは G500, G54..G599 命令が実行されなければ起動しません。）主として変数は、MMC と PLC 基本フレームへの書きこみプロセス用のメモリとして働きます。これらのフレーム変数は、データをバックアップすると保存されます。

チャンネルの第 1 基本フレーム

事前定義されたフレーム変数 $P_UBFR に書き込みをしても、配列インデックス 0 を持つフレームを同時に起動することはできず、起動されるのは命令 G500, G54..G599 が実行された場合にのみとなります。プログラムでも、この変数の書き込みや読み出しができます。

$P_UBFR

$P_UBFR は、$P_CHBFR[0] と同一です。

標準として、チャンネルには基本フレームが 1 つついているので、このシステム変数は旧バージョンと互換性があります。チャンネル別基本フレームがない場合、読み出し／書き込み時に、アラーム "Frame: instruction not permissible" が出力されます。

6.9.2 チャンネルで有効なフレーム

機能

$P_NCBFRAME[n]

カレント NCU グローバル基本フレーム

現在のグローバル基本フレーム配列要素は、システム変数 $P_NCBFRAME[n] を介して読み出しと書き込みができます。その結果生じた総基本フレームは、チャンネルの書き込みプロセスによって計算されます。

変更済みフレームは、フレームが設定されているチャンネルでのみ有効になります。フレームが NCU の全てのチャンネルで変更されなければならないような場合、[n] および $P_NCBFRAME[n] の両方を設定する必要があります。その場合、その他のチャンネルは、例えば G54 を使用するなどして、そのフレームを起動できなければなりません。基本フレームを書き込むと、総基本フレームは計算し直されます。

$P_CHBFRAME[n]

カレントチャンネル基本フレーム

現在のチャンネル基本フレーム配列要素は、システム変数 $P_CHBFRAME[n] を使用して書き込みと読み出しをすることができます。結果として生じた総基本フレームは、チャンネルの書き込みプロセスによって計算されます。基本フレームを書き込むと、総基本フレームは計算し直されます。

$P_BFRAME

チャンネルのカレント第 1 基本フレーム

現在の基本フレームは、パートプログラムで有効な配列インデックス 0 を持つ事前定義されたフレーム変数 $P_BFRAME を介して、パートプログラム内で読み出しと書き込みができます。書きこまれた基本フレームは、即座に計算に含まれます。

$P_BFRAME は $P_CHBFRAME[0] に対応します。デフォルトにより、このシステム変数は常に有効値です。チャンネル別基本フレームがない場合、書き込み／読み出し時に、アラーム "Frame: instruction not permissible" が出力されます。

$P_ACTBFRAME

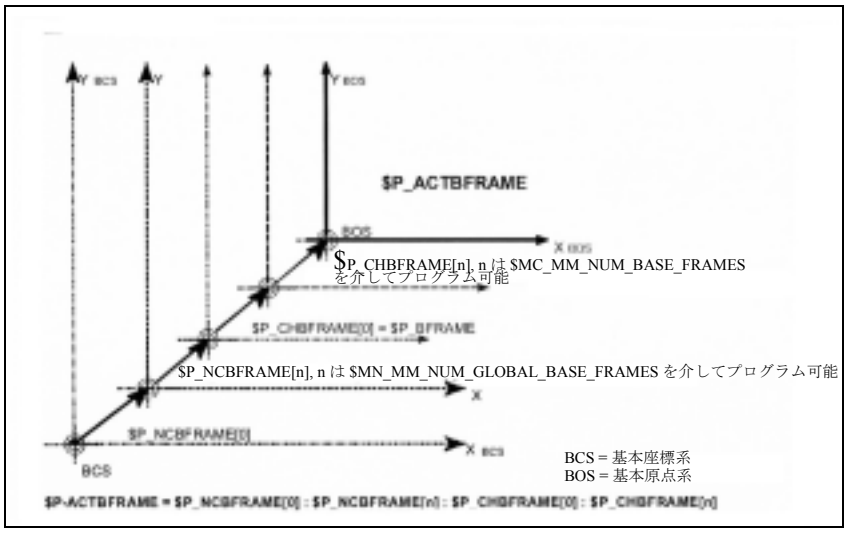
総基本フレーム

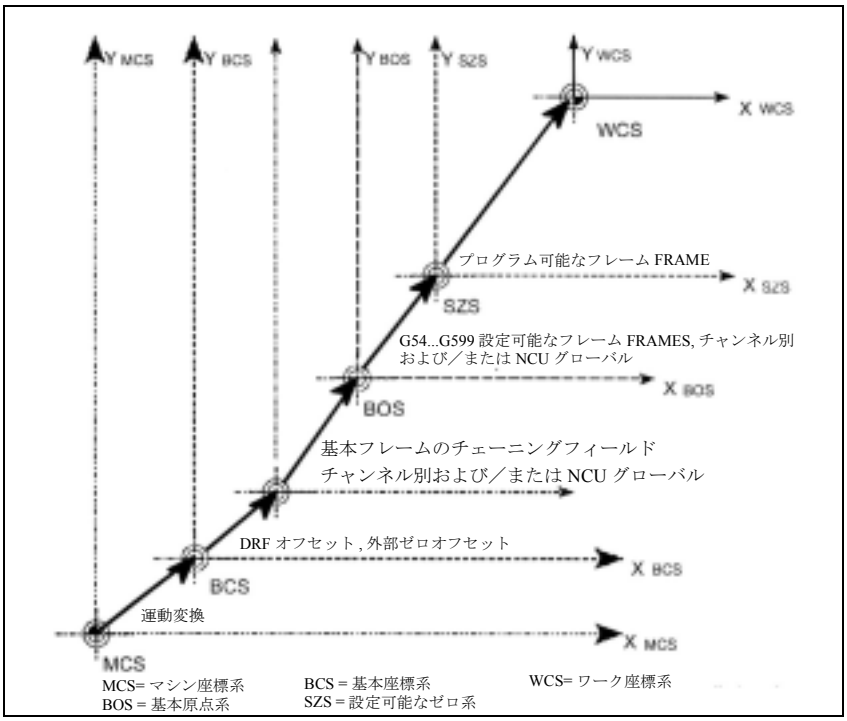
変数 $P_ACTBFRAME がチェーニングされた総基本フレームを確定します。この変数は読み出しのみです。

$P_ACTBFRAME は、次に対応します

$P_NCBFRAME[0] : ... : $P_NCBFRAME[n] :

$P_CHBFRAME[0] : ... : $P_CHBFRAME[n].





総基本フレームのプログラミング

システム変数 $P_CHBFRMASK および $P_NCBFRMASK を介して、

どの基本フレームを“総“基本フレームの計算に含めるか決めることができます。この変数は、OPI を介してのみ、プログラムでの設定や読み出しができます。

この変数はビットマスクとして翻訳され、$P_ACTBFRAME のどの基本フレーム配列要素を計算に含めるべきか指定します。

$P_CHBFRMASK を使用して計算に含めるべきチャンネル別基本フレームを、また $P_NCBFRMASK を使用して計算に含めるべき NCU グローバル基本フレームを指定することができます。

変数のプログラミングにより、総基本フレームおよび総フレームが再度計算されます。Reset（リセット）が実行された後、基本設定値は、

$P_CHBFRMASK = $MC_CHBFRAME_RESET_MASK および

$P_NCBFRMASK = $MN_NCBFRAME_RESET_MASK となります。

例

$P_NCBFRMASK = ’H81’ ; $P_NCBFRAME[0] : $P_NCBFRAME[7]

$P_CHBFRMASK = ’H11’ ; $P_CHBFRAME[0] : $P_CHBFRAME[4]

$P_IFRAME

現在の設定可能なフレーム

チャンネルで有効な現在設定可能なフレームは、事前定義された設定可能なフレーム変数 $P_IFRAME を使用して、パートプログラムで読み出しと書き込みができます。

書きこまれた設定可能なフレームは、即座に計算に含まれます。NCU グローバルフレームを使用すると、修正されたフレームは、そのフレームが設定されたチャンネルでしか有効になりません。NCU の全てのチャンネルに対してそのフレームが変更される場合、both $P_UIFR[n] および $P_IFRAME がプログラミングされる必要があります。その場合、その他のチャンネルは、例えば G53 を使用するなどして個々のフレームを起動している必要があります。

$P_PFRAME

現在プログラム可能なフレーム

$P_PFRAME は、TRANS/ATRANS,

G58/G59, ROT/AROT, SCALE/ASCALE,

MIRROR/AMIRROR のプログラミングの結果、あるいはプログラム可能な FRAME に CTRANS,

CROT, CMIRROR, CSCALE を割当てた結果生じたプログラム可能なフレームです。

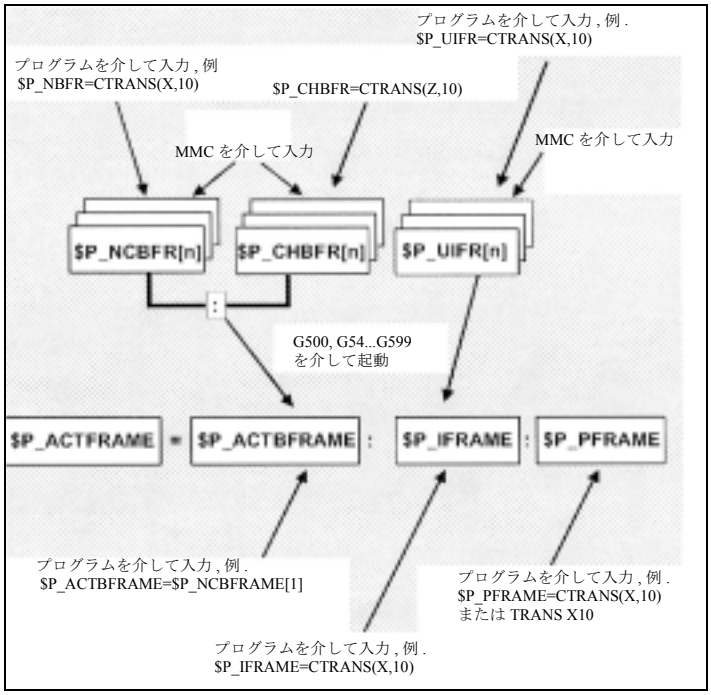
現在の、プログラム可能なフレーム変数は、設定可能なゼロ系 (SZS) とワーク座標系 (WCS) 間の基準を設定します。

$P_ACTFRAME

現在の演算結果の総フレーム $P_ACTFRAME は、全基本フレーム、現在の設定可能なフレーム、およびプログラム可能なフレームのチェーニングから得られます。

この現在のフレームは、フレーム要素が変更される度にアップデートされます。

$P_ACTFRAME は、$P_ACTBFRAME : $P_IFRAME : $P_PFRAME に対応します。

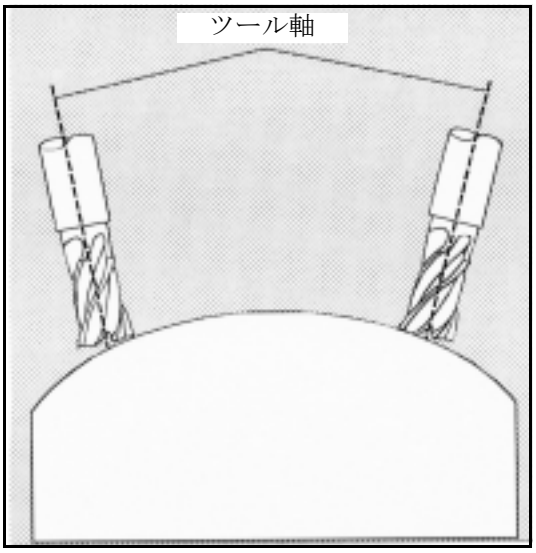


7 変換

7.1 3 軸、4 軸、および 5 軸変換 : TRAORI

曲面を工作する際に最高の切削状態を得るためには、ツールのアプローチ角度が変更されなければなりません。

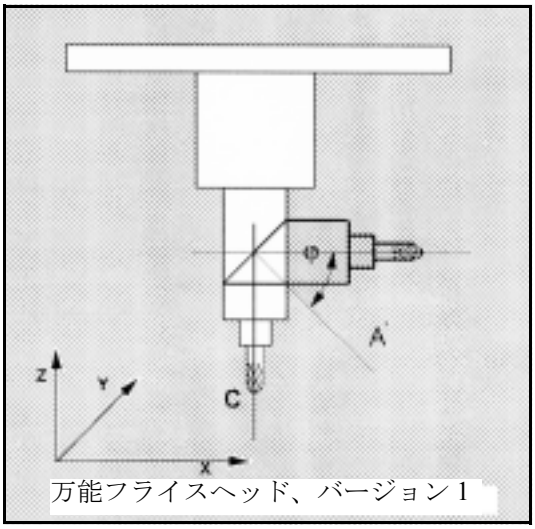
アプローチ角度変更に必要なマシンデータは、軸データに格納されています。



万能フライスヘッド

ここでは、3 つの直線軸 (X, Y, X) ならびに 2 つの向き調整軸が、ツールの設定角度と切削点を定義しています。2 つの向き調整軸の一方が、通常の場合、45° の角度で位置決めされた傾斜軸として使用されています（図例の A’ ）。

回転軸の軸順序ならびにツールの向き調整方向は、マシンキネマティックスの機能を使用して設定されます。右の例では、調整はマシンキネマティックス CA で表わされています。



万能フライスヘッド、バージョン 1

次のような相互関係が成り立ちます :

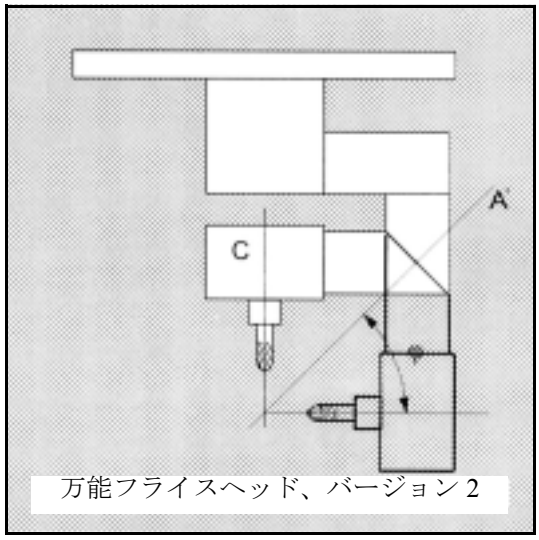
A' は、X 軸に対して角度 φの位置にある。

B' は、Y 軸に対して角度 φの位置にある。

C' は、Z 軸に対して角度 φの位置にある。

角度 φは、マシンデータを介して、0° から +89° の範囲に設定することができます。

ツール向き調整の方向決定によっては、ツールの長さ補正がツール向き調整方向に作用するには、NC プログラムに有効な切削面（ G17, G18, G19 ）が設定されなければなりません。



万能フライスヘッド、バージョン 2

直線旋回軸を使用する変換

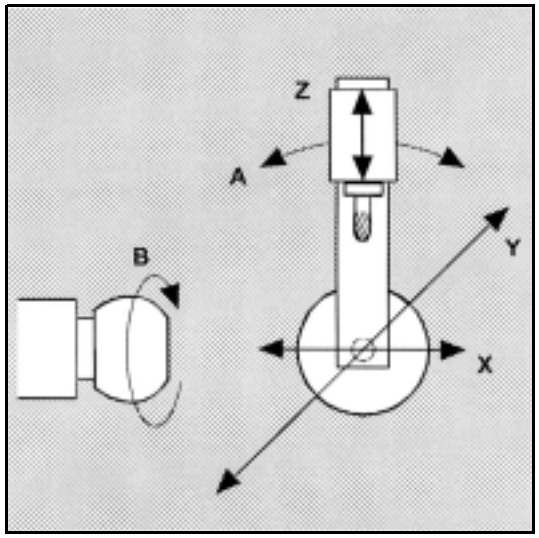
これは、移動中のワークと移動中のツールの配置です。

このキネティックは、3 つの直線軸（ X, Y, X ）とそれに対して直角を成す 2 つの回転軸で構成されています。例としては、第 1 の回転軸が 2 つの直線軸のクロススライドを介して移動すると、ツールは第 3 の軸に並行して位置決めされます。

第 2 の回転軸がワークを回転させます。

第 3 の直線軸（旋回軸）が、クロススライドの平面内に存在します。

回転軸の軸順序ならびにツールの向き調整の方向は、マシンキネティックスの機能であるマシンデータを介して設定されます。



次のような補間相互関係が成り立ちます :

軸 :	軸順序 :
第 1 回転軸	A A B B C C
第 2 回転軸	B C A C A B
直線旋回軸	Z Y Z X Y X

3 軸 および 4 軸変換

3 軸 および 4 軸変換は、5 軸変換の変形です。

2 つまたは 3 つの直線軸と 1 つの回転軸を設定することができます。変換は、回転軸が向き調整面に互いに直交するように位置決めされていると仮定して行われます。

ツールは、回転軸に垂直な平面内でのみ向き調整が可能です。変換は、可動ツールと移動ワークを使用するタイプのマシンで使用できます。

3 軸 および 4 軸変換は、5 軸変換と同様に設定およびプログラミングすることができます。

プログラミング

TRAORI(n)
TRAFOOF

コマンドの説明

TRAORI	最初に決定された向き調整変換を起動する
TRAORI(n)	n が割当てられている向き調整変換を起動する
n	変換数 (n = 1 または 2), TRAORI(1) は TRAORI に対応している
TRAFOOF	変換を停止する

追加説明

変換が起動されている場合、位置のパラメータ（X, Y, Z）は常に、ツールの先端を指します。

変換に含まれる回転軸の位置に変更がある場合、ツール先端の位置が変わらないように、その他のマシン軸で補正動作が行われます。

7.1.1 ツール向き調整のプログラミング

5 軸プログラムは通常、CAD/CAM システムで作成され、制御装置において入力されることはありません。したがって、次の説明は主に、後処理プログラムのプログラマを対象としています。

ツールの向き調整のプログラミングには 3 つの方法があります :

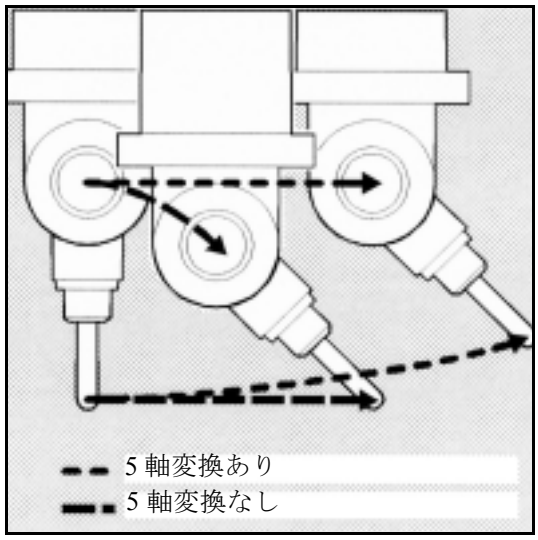
1. 回転軸動作のプログラミング。向き調整の変更は必ず、基本座標系か、またはマシン座標系で行ってください。向き調整軸は同期軸として移動されます。

2. A2, B2, C2 を使用する Euler 角度または RPY 角度での プログラミング

   あるいは A3, B3, C3 を使用する方向ベクトルのプログラミング。

   方向ベクトルは、ツール先端からツールホルダの方向に向かっている。

3. リード角 LEAD およびサイド角 TILT (正面フライス加工) のプログラミング。



いずれの場合も、向き調整変換がアクティブになっていなければ、向き調整プログラミングは使用できません。

所 : これらのプログラムは、どのマシンキネマティックスへでも転送することができます。

プログラミング

G1 X Y Z A B C	回転事項動作のプログラミング
G1 X Y Z A2= B2= C2=	Euler 角度のプログラミング
G1 X Y Z A3= B3= C3=	方向ベクトルのプログラミング
G1 X Y Z A4= B4= C4=	ブロックスタートの表面ノーマルベクトルのプログラミング
G1 X Y Z A5= B5= C5=	ブロックエンドの表面ノーマルベクトルのプログラミング
LEAD	ツール向き調整のプログラミング用のリード角度
TILT	ツール向き調整のプログラミング用のサイド角度

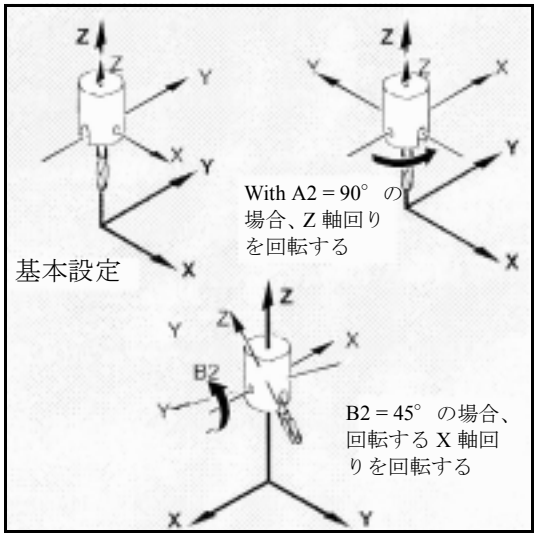
マシンデータを使用して、Euler 角度と RPY 角度を切り替えることができる。

Euler 角度のプログラミング

A2, B2, C2 を使用して向き調整用にプログラムされた値は、Euler 角度（度）として翻訳されます。

まず Z 軸回りの A2 、次に新規の X 軸回りの B2 、最後に 新規の Z 軸回りの C2 を使用して、Z 方向にベクトルを回転させることによって、方向ベクトルが作成されます。

この場合、C2 （新規 Z 軸回りの回転）の値は関係がないので、プログラムする必要はありません。

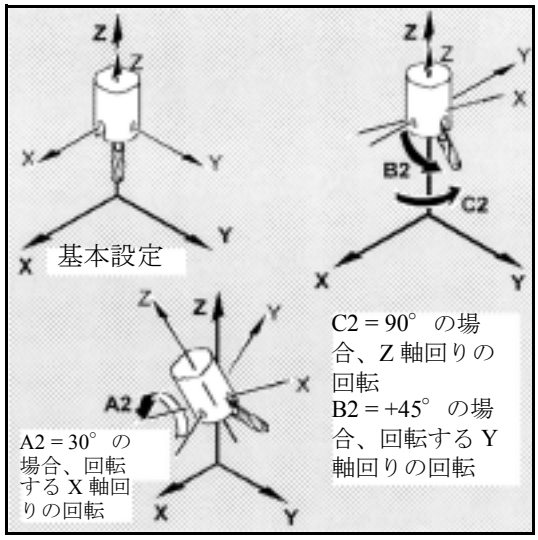


RPY 角度でのプログラミング

A2, B2, C2 を使用する向き調整用にプログラムされた値は、RPY 角度（度）として翻訳されます。

まず Z 軸回りの C2 、次に新規の Y 軸回りの B2 、最後に 新規の X 軸回りの A2 を使用して、Z 方向にベクトルを回転させることによって、方向ベクトルが作成されます。

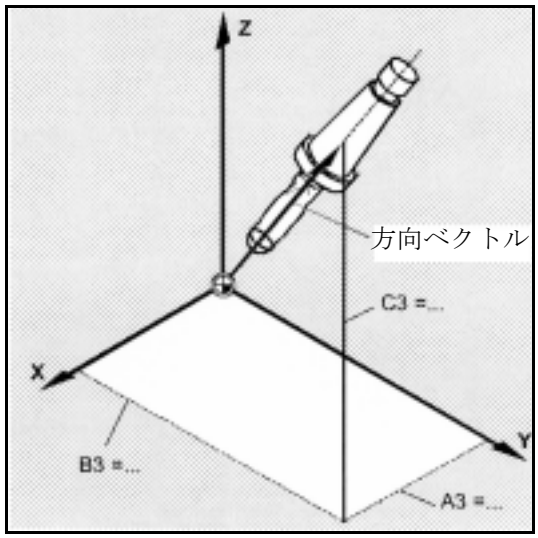
Euler 角度のプログラミングの場合とは異なり、この 3 つの値で向き調整ベクトルが確定されます。



方向ベクトルのプログラミング

方向ベクトルの構成要素は、A3, B3, C3 を使用してプログラムされます。このベクトルがツール装置の方向を指す場合、ベクトルの長さは関係ありません。

プログラムされていないベクトル構成要素は、ゼロにセットされます。



正面フライス加工

正面フライス加工モードを使用すると、どんな曲率を持つ表面でも加工できます。

この種類の 3D フライス加工では、ワーク表面上の 3D パスのラインバイライン（1 列 1 列ごと）の定義が必要です。

通常 CAM で行われる計算には、ツールの形と寸法が含まれます。

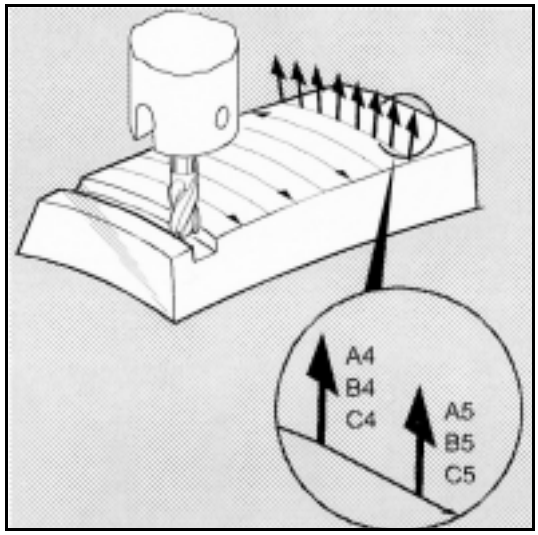
これらの計算後、NC ブロックは後処理プログラムを介して制御装置に読み取られます。

表面の定義

パス曲率は、次の要素を持つ表面ノーマルベクトルを介して定義されます :

A4, B4, C4 ブロック開始のスタートベクトル

A5, B5, C5 ブロック終了のエンドベクトル



ブロックがスタートベクトルしか持っていない場合、ノーマルベクトルはブロック全体で一定となります。

ブロックがエンドベクトルしか持っていない場合、大円補間を使用して、先行するブロックのエンドベクトルとプログラムされているエンド値の間で補間が行われます。

スタートベクトルとエンドベクトルの両方がプログラムされている場合、大円補間を使用してその 2 方向間で補間が行われ、連続した平滑な移動パスが産出されます。すなわち、連続平滑パス（滑らかな輪郭）を創ることができます。

基本設定では、ノーマル表面ベクトルは、有効面 G17 から G19 とは関係なく、Z 方向に向かいます。

ベクトルの長さには意味はありません。

プログラムされていないベクトル要素は、ゼロにセットされます。

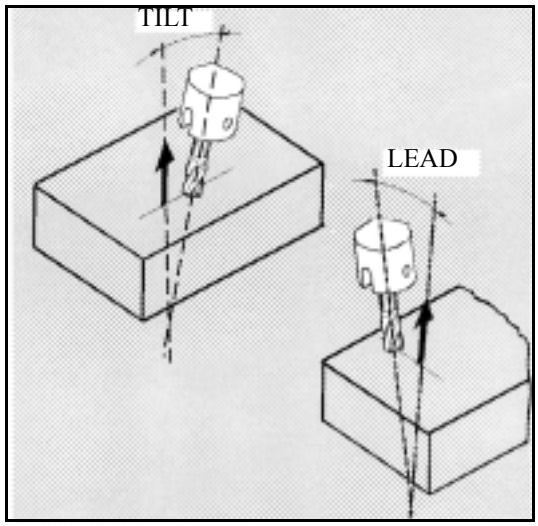
ORIWKS が有効になっている場合（次ページを参照のこと）、ノーマル表面ベクトルは有効フレームと関連し、フレームと一緒に回転されます。

表面ノーマルベクトルは、マシンデータを介してセットされたリミット値を持つパスタンジェントに対して必ず垂直でなければならず、垂直でない場合にはアラームが出力されます。

LEAD および TILT を使用するツール向き調整のプログラミング

演算結果ツール向き調整は次の要素を計算して得られます：

- パスタンジェント
- 表面ノーマルベクトル
- リード角度 LEAD
- ブロックエンドの傾斜角度 TILT



コマンドの説明

LEAD	パスタンジェントと表面ノーマルベクトルで作成された平面の表面ノーマルベクトルを基準にした角度
TILT	表面ノーマルベクトルを基準にしたパスタンジェントに垂直である、平面内の角度

内側コーナーでの挙動（ 3D ツールオフセットの場合）

ブロックが内側コーナーで削られた場合、ブロックエンドで演算結果ツール向き修正もまた行われる。

7.1.2 向き修正軸リファレンス - ORIWKS, ORIMKS

プログラミング

N.. ORIMKS
または
N.. ORIWKS

コマンドの説明

ORIMKS	マシン座標系における回転
ORIWKS	ワーク座標系における回転

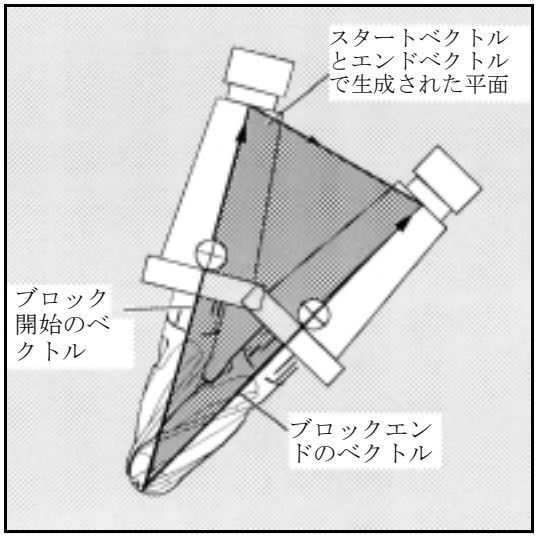
機能

Euler 角または RPY 角、あるいは向き調整ベクトルを使用して、ワーク座標系における回転をプログラミングする場合、ORIMKS/ORIWKS を介して回転動作をセットすることができます。

動作

ORIMKS を使用する場合、ツールの動作はマシンキネマティックスによって異なります。ツール先端が空間に固定されている「向き修正」が変更される場合、回転軸位置同士で直線補間が行われます。

ORIWKS を使用する場合、ツール動作はマシンキネマティックスとは別に実行されます。ツールチップが空間に固定されている向き修正が変更される場合、ツールはスタートベクトルとエンドベクトルで生成された平面内を移動します。



追加説明

ORIWKS はデフォルト設定です。どのマシンで 5 軸プログラムを走らせるか最初から明確でない場合、ORIWKS が選択されることになります。マシンが実際にとる動作は、マシンキネマティックスによって異なります。

ORIMKS を使用すると、例えば装置同士の衝突を避けるなどの、実際のマシン動作をプログラムすることができます。

どの補間タイプを有効にするかは、マシンデータ $MC_ORI_IPO_WITH_G_CODE:

ORIMKS/ORIWKS または ORIMACHAX/ORIVIRTAX に定義されます (7.1.4 向き調整軸を参照)。

7.1.3 特異点とその取り扱い

ORIWKS に関する説明 :

5 軸マシンの特地点近傍の向き調整動作は、マシン軸に大きな動作を要求します。(例えば、C が回転軸 A が旋回軸の場合の回転旋回ヘッドに対して、A = 0 の場合の全ての位置が特異となります)。

マシンへの過負荷を防止するために、特異点近傍のツールパス速度は速度制御によって顕著に低減されます。

マシンデータ

$MC_TRAFO5_NON_POLE_LIMIT

$MC_TRAFO5_POLE_LIMIT を使用すると、

極点近傍の向き調整動作が極点を通過できるように変換をパラメータ化することができるので、マシニングをスピードアップすることができます。

7.1.4 向き調整軸

プログラミング

N.. ORIEULER または ORIRPY

あるいは

N.. ORIVIRT1 または ORIVIRT2

N.. G1 X Y Z A2= B2= C2=

コマンドの説明

ORIEULER	Euler 角を使用する向き調整プログラミング
ORIRPY	RPY 角を使用する向き調整プログラミング
ORIVIRT1	仮想向き調整角度を使用する向き調整プログラミング （定義 1)、MD $MC_ORIAX_TURN_TAB_1 に準じる定義
ORIVIRT2	仮想向き調整角度を使用する向き調整プログラミング（定義 2)、MD $MC_ORIAX_TURN_TAB_2 に準じる定義
G1 X Y Z A2= B2= C2=	仮想軸の角度プログラミング

プログラミング

N.. ORIMACHAX または ORIVIRTAX

N.. G1 X Y Z A B C

コマンドの説明

ORIMACHAX	向き調整軸の線補間
ORIVIRTAX	大円補間
ORIMKS	マシン座標系の回転　説明はセクション 7.1.2 を参照のこと
ORIWKS	ワーク座標系の回転　説明はセクション 7.1.2 を参照のこと
G1 X Y Z A B C	マシン位置のプログラミング

機能

向き調整軸機能は、ツールの向き調整を空間に描きます。
それにより、それ自体の回りの回転を描く第 3 等級の自由
度（自由度 3）が生じます、これは 6 軸変換に必要です。

MD $MC_ORI_DEF_WITH_G_CODE は、プログラムされている角度 A2, B2 および C2 が定義される方法を指定します :

MD $MC_ORIENTATION_IS_EULER に準じる定義 (標準) または

G_group 50 に準じる定義 (ORIEULER, ORIRPY, ORIVIRT1, ORIVIRT2).

MD $MC_ORI_IPO_WITH_G_CODE は、有効になる補間タイプを定義します :

ORIWKS/ORIMKS または ORIMACHAX/ ORIVIRTAX.

JOG モード

このモードでは、向き調整角度の補間は常に線形です。移動キーを介する連続増分移動を使用すると、向き調整軸を 1 つだけ移動することができます。ハンドルを使用すると、両方の向き調整軸が同時に移動することができます。

向き調整軸の手動移動では、ラピッド移動オーバーライドを使用すると、チャンネル別フィードレートオーバーライドスイッチまたはラピッド移動オーバーライドスイッチが有効になります。

次のマシンデータを使用して、個別の速度を指定することができます :

$MC_JOG_VELO_RAPID_GEO

$MC_JOG_VELO_GEO

$MC_JOG_VELO_RAPID_ORI

$MC_JOG_VELO_ORI

フィードのプログラミング

FORI1	大円上の向き調整ベクトルを旋回するためのフィード
FORI2	旋回された向き調整ベクトル回りのオーバーレイ回転用のフィード

向き調整動作を使用する場合、プログラムされたフィードは角速度 [度 / 分] に対応します。

G コード上でのフィードの効果：

ORIMACHAX をプログラミングする場合、向き調整軸用のフィードは、FL[ ] 命令（フィードリミット）によって制限することができます。

ORIVIRTAX をプログラミングする場合、フィードは必ず FORI1 または FORI2 を使用して設定してください。FORI1 および FORI2 は、一旦 NC ブロックにプログラムすることもできます。この方法でプログラミングする場合、パスは常に取り得る最短距離を移動します。

オーバーレイされた回転および旋回動作を使用する場合、最小フィードレートは必ず使用されたフィードレートとなります。向き調整動作を使用する場合、プログラムされてフィードレートは、角速度 [度 / 分] に対応します。

ジオメトリ軸および向き調整軸が共に同じパスを移動中である場合、移動動作は最小フィードレートから求められます。

7.1.5 デカルト座標 PTP 移動

プログラミング

N.. TRAORI

N.. STAT=‘B10‘ TU=‘B100‘ PTP

N.. CP

コマンドの説明

PTP	Point to Point この動作は、同期動作として実行される ; 動作に拘わっている最低速軸が、速度の優勢軸となる。
CP	Continuous Path （連続パス）動作は、デカルト座標パス動作として実行される。
STAT=	継ぎ目位置 ; 値は、変換によって異なる
TU=	URN 情報　軸角度のアプローチを、-360 度から +360 度に設定することができる。

機能

この機能を使用すると、位置をデカルト座標系にプログラムすることができます ; ただし、マシン動作はマシン座標で行われます。

継ぎ目位置を変更する（例えば変更中に動作が特異点を通過する）場合などに、この機能を使用することができます。

（注）この機能は、変換と共に使用して初めて有効となります。また、"PTP travel" （ PTP 移動）は、G0 および G1 と一緒でなければ使用できません。

動作

デカルト座標移動とマシン軸移動との切換えは、コマンド PTP および CP を介して行われます。CP はデフォルト設定です。

位置のプログラミング (STAT=)

ただ単にデカルト座標およびツール向き調整を使用して位置を指定するだけでは、マシン位置を独自に定義することはできません。どのキネマティックスが使用されているかによって、最高 8 もの異なる継ぎ目位置があります。したがって、変換別となります。明確にデカルト座標を軸角度に変換するには、STAT= コマンドを使用して継ぎ目の位置を指定する必要があります。"STAT" コマンドには、可能性として考えられる位置それぞれに対するバイナリ値として 1 ビットが含まれています。

参照 :

次の文書に、さまざまな変換の詳細説明があります :

Description of Functions（機能の説明）（パート 3）

(Part 3), "Transformation Package Handling"（変換パッケージの取り扱い）

"STAT" のプログラムに必要な位置ビットの説明は、次を参照してください :

Description of Functions（機能の説明）（パート 3）

(Part 3), "3-Axis to 5-Axis Transformation".（3 軸から 5 軸への変換）

軸角度のプログラミング (TU=)

軸角度（< ± 360 ）の決定アプローチには、"TU=" コマンドを使用して、このデータをプログラムする必要があります。

このコマンドは、ノンモーダルです。

軸は最短距離を通って移動します :

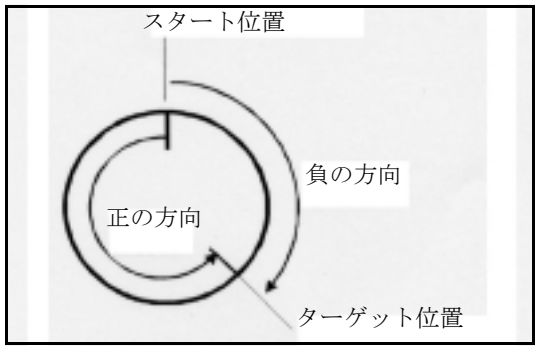
- 位置を使用して TU がプログラミングされていない場合
- 移動範囲 > ± 360 度を持つ軸を使用する場合

例：

図に示したターゲット位置には、正の方向からでも負の方向からでもアプローチすることができます。その方向は、アドレス A1 の下で設定されます。

A1=22?°, TU=bit 0, ・正の方向

A1=-13?°, TU=bit 1, ・負の方向



追加説明

モードの変更

機能 " Cartesian PTP travel"（デカルト座標 PTP 移動）は、AUTO モードおよび MDA モードでのみ有効です。モードが JOC に変更された場合、移動は CP 設定を使用して実行されます。AUTO または MDA に変更が戻された場合、この運転モードでの移動モードの最終設定はリセットされます。

Power On / reset

Power ON またはリセットした後は、設定はマシンデータ

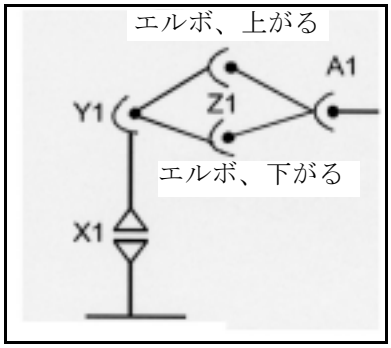
$MC_GCODE_RESET_VALUES[49] を基準にして行われます。移動モード

"CP" はデフォルト設定となります。

Repos

機能　"Cartesian PTP travel" がブロック中断中に設定された場合、PTP を使用して再位置決めが行われます。

プログラミング例



N10 G0 X0 Y-30 Z60 A-30 F10000	スタート位置・エルボ、上がる
N20 TRAORI(1)	変換 ON
N30 X1000 Y0 Z400 A0	
N40 X1000 Z500 A0 STAT=´ B10 ´ TU=´ B100 ´ PTP	変換せずに再び向き調整 ・エルボ、下がる
N50 X1200 Z400 CP	CP 変換が再び有効
N60 X1000 Z500 A20	
N70 M30	

7.2 回転パート上のフライス加工 : TRANSMIT

プログラミング

TRANSMIT または TRANSMIT(n)

TRAFOOF

コマンドの説明

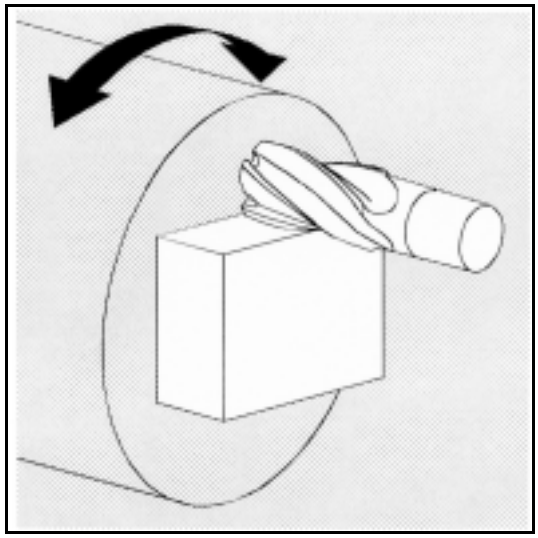
TRANSMIT	最初の宣言 TRANSMIT 機能を起動する
TRANSMIT(n)n	n 番目の宣言 TRANSMIT 機能を起動する ; n は 2 より大きくなってはならない (TRANSMIT (1) は、TRANSMIT に対応する)。
TRAFOOF	アクティブ変換を停止する

他の変換のうちのいずれかがそれぞれのチャンネルで起動されている場合、アクティブ TRANSMIT 変換もまた停止される。(例、TRACYL, TRAANG, TRAORI).

TRANSMIT 機能を使用すると、次を行うことができます :

- 回転クランプの回転パートのエンド正面加工 (ドリル穴、輪郭)
- デカルト座標系を使用してこのようなマシニングオペレーションをプログラムすることができます
- 制御は、デカルト座標系のプログラムされている移動動作を実際のマシン軸上の移動動作に変換することができます（標準設定）:
  - 回転軸
  - 回転軸に垂直なインフィード軸
  - 回転軸に平行な縦軸

  直線軸は、互いに垂直に位置決めされている。
- 回転センタを基準にしたツールセンタオフセットが可能である。
- 速度制御により、回転動作用に定義された限度が加えられる。



回転軸

回転軸はジオメトリ軸に割当てられるため、回転軸をプログラムすることはできません。したがって、回転軸を直接チャンネル軸としてプログラムすることもできません。

極点

極点を移動する方法が 2 通りあります :

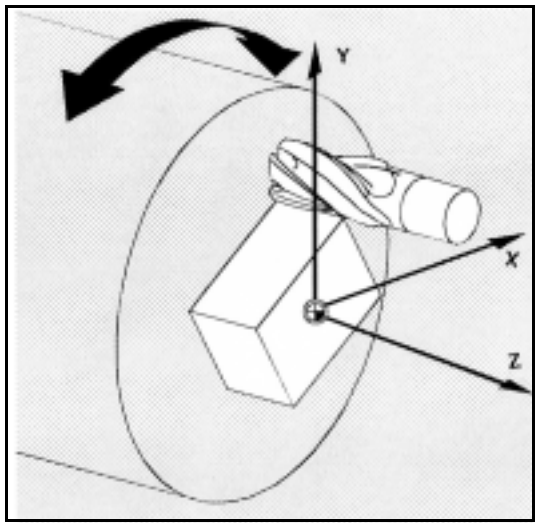
1. 直線軸だけを移動する
2. 極点内の回転軸の回転を使用して極点内を移動して、極点から出る

これは、MD 24911 および 24951 で設定できます。

参照 :

/FB/ M1 Kinematic Transformation（キネマティック変換）

プログラミング例



N10 T1 D1 G54 G17 G90 F5000 G94	ツール選択
N20 G0 X20 Z10 SPOS=45	アプローチ基準点
N30 TRANSMIT	TRANSMIT 機能の起動
N40 ROT RPL=-45 N50 ATRANS X-2 Y10	フレームのセット
N60 G1 X10 Y-10 G41 OFFN=1 N70 X-10 N80 Y10 N90 X10 N100 Y-10	正方形の粗加工；公差 1 mm
N110 G0 Z20 G40 OFFN=0 N120 T2 D1 X15 Y-15 N130 Z10 G41	ツールチェンジ
N140 G1 X10 Y-10 N150 X-10 N160 Y10 N170 X10 N180 Y-10	正方形の仕上げ加工
N190 Z20 G40 N200 TRANS N210 TRAFOOF	フレームの選択解除
N220 G0 X20 Z10 SPOS=45	基準点にアプローチする
N230 M30	

7.3 円筒補間 : TRACYL

プログラミング

TRACYL(d) または TRACYL(d,t)

TRAFOOF

コマンドの説明

TRACYL(d)	最初の宣言 TRACYL(d) 機能を起動する
TRACYL(d,n)	n 番目の宣言 TRACYL(d,n) 機能を起動する ; n は 2 より大きくなってはならない。 TRACYL(d,1) は TRACYL(d) に対応する。
d	加工される円柱の現在の直径を示す値
TRAFOOF	変換オフ
OFFN	輪郭オフセット - ノーマル : プログラムされている基準輪郭からみぞの端までの距離

他の変換のいずれかが同一チャンネル上でアクティブになっている場合、アクティブ TRACYL 変換は停止される。

(例、TRANSMIT, TRAANG, TRAORI).

機能

円筒補間 TRACYL

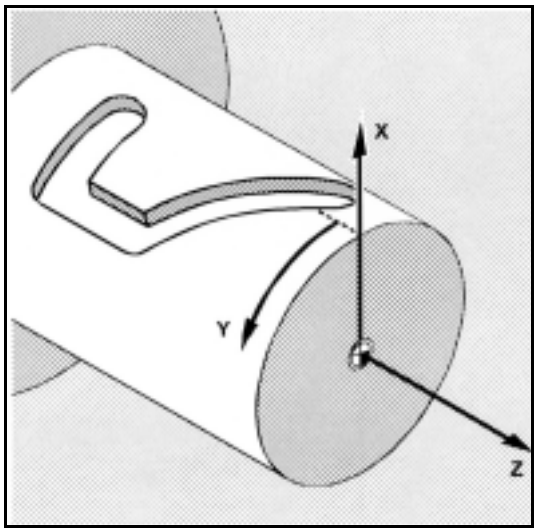
円筒補間 TRACYL

を使用すると、次を行うことができます :

加工

- 円柱上の縦方向のみぞ
- 円柱上の横方向のみぞ
- 円柱上のその他の形態のみぞ

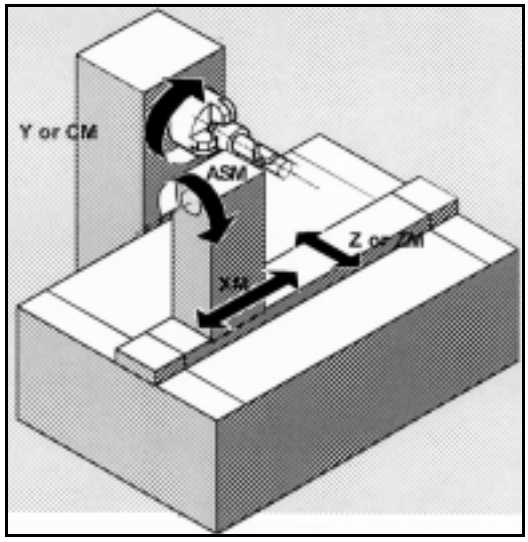
みぞの形は、処理された円柱水準面エリアを基準にしてプログラムされます。



ワーク座標系

円筒補間のタイプは、2 種類あります :

- みぞサイド補正なし
- みぞサイド補正あり



マシン座標系

みぞサイド補正なしの場合 ;

制御により、円柱座標系のプログラムされた移動動作が実際のマシン軸の動作に変換されます :

- 回転軸
- 回転軸に垂直なインフィード軸
- 回転軸に平行な縦軸

直線軸は互いに垂直になるように位置決めされています。インフィード軸は回転軸に交差しています。

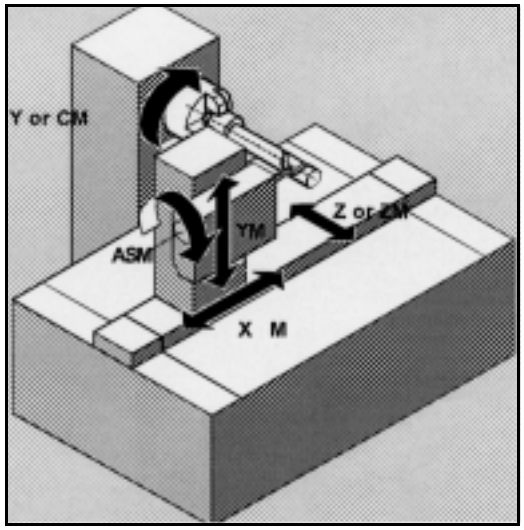
みぞサイド補正あり :

上記と同じキネマティックスに加えて :

- 周囲の方向に平行な縦軸

直線軸は互いに垂直に位置決めされています。

速度制御により回転動作に定義された制御が加えられる。

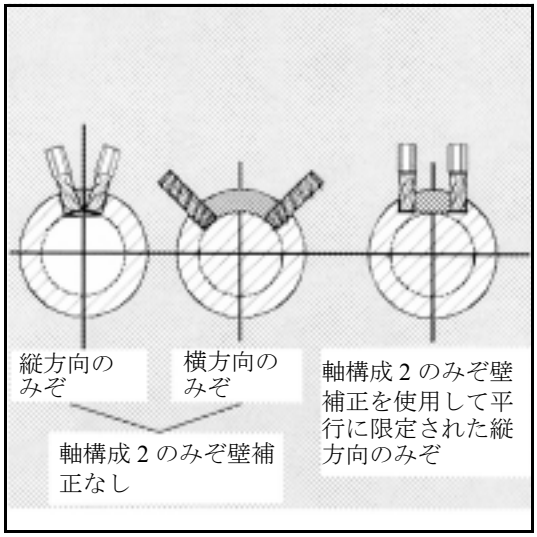


マシン座標系

みぞの断面

軸構成 1 を使用すると、みぞの幅がツールの半径に一致する場合、回転軸沿いのみぞは必ず並行になります。

周囲（横方向のみぞ）に並行するみぞは、スタート時と終了時は平行ではありません。



オフセット輪郭ノーマル OFFN

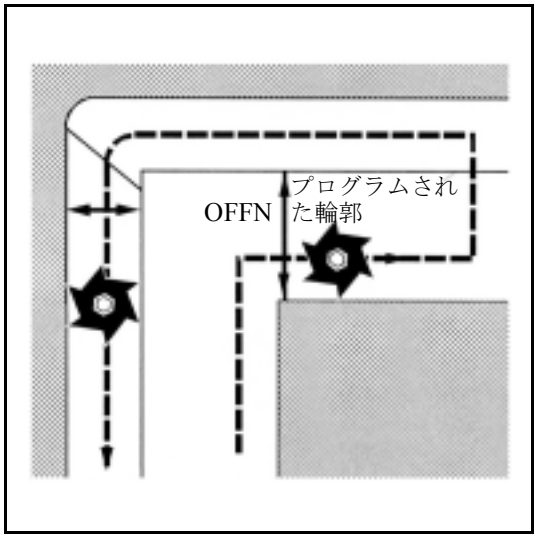
TRACYL を使用するフライスみぞでは、

- みぞの中心線は、パートプログラムに設定され
- みぞの幅は、OFFN を介して設定されます。

ツール半径が選択されていると、みぞの損傷を防止するために OFFN のみが有効になります。また、みぞの反対側に損傷を与えないように、必ず OFFN>= ツール半径を実行してください。

みぞのフライス加工用のパートプログラムには通常、次のようなステップがあります：

1. ツールの選択
2. TRACYL の選択
3. 適切な座標オフセット (FRAME) の選択
4. 位置決め
5. OFFN のプログラミング
6. TRC の選択
7. ブロックにアプローチ (TRC を入力し、みぞの側壁にアプローチする)
8. みぞ中心線の輪郭
9. TRC の選択解除
10. ブロック後退 (TRC を出て、みぞの側壁から後退する)
11. 位置決め



12. TRAFOOF
13. 元の座標オフセット (FRAME) をもう一度選択する

例外：

- TRC 選択 :
  TRC は、TRC に基づいてプログラムされるのではなく、プログラムされたみぞ中心線を基準にしてプログラムされます。G42 を設定すると、ツールはみぞ側壁の左側へ移動します（ G41 の替わり)。
  負のリーディングサインを使用してみぞ幅が OFFN に指定されていない場合、これを行う必要はありません。
- OFFN は、TRACYL を使用する場合と使用しない場合では、効果が異なります。また、TRC が有効になっていると OFFN は TRCYL を使用しない計算に含まれるため、TRAFOOF 後に設定がゼロに戻ってしまいます。
- パートプログラム内で OFFN を変更することができます。すなわち、みぞ中心線をセンタから移動させることができます（図を参照してください)。
- 制御みぞ :
  TRACYL を使用すると、制御みぞ用として全く同じみぞが作成されることはありません（それでは、まるでみぞ幅と同じ直径を持つツールを使用して製造するのと同じではありませんか)。
  原則として、より小さいの円柱形ツールを使用する場合、より大きい円柱形ツールを使用して作成するのと同じみぞサイドジオメトリを作成することはできません。
  TRACYL は誤差を最小限にします。誤差が発生しないように、ツール半径はみぞ幅の半分よりわずかに小さくしてください。

みぞサイド補正を使用して円柱表面カーブ変換を行う場合、みぞのマシニングがプログラムされているみぞ中心線に合わせて調整されるため、補正に使用する軸はゼロ (y = 0) に設定されている必要があります。

回転軸

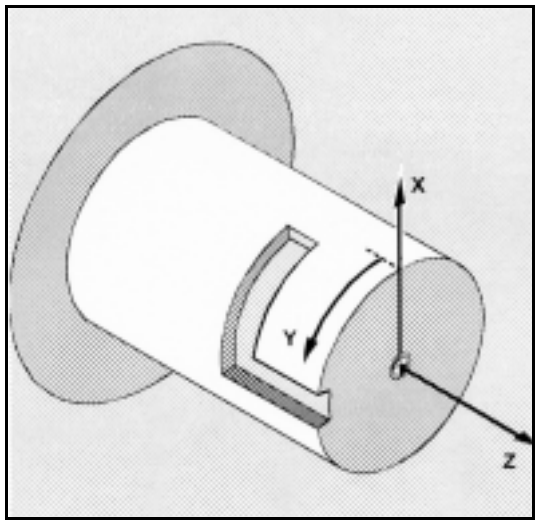
回転軸はジオメトリ軸に割当てられているのでプログラムすることはできません。したがって、直接チャンネル軸として設定することもできません。

軸使用

次の軸は、位置決め軸または往復軸として使用することはできません：

- 円柱表面カーブの周囲方向のジオメトリ軸 (Y 軸)
- みぞサイド補正に使用される追加直線軸 (Z 軸)

プログラミング例



10 T1 D1 G54 G90 F5000 G94	ツールの選択、クランピング補正
N20 SPOS=0 N30 G0 X25 Y0 Z105 CC=200	基準点へのアプローチ
N40 TRACYL (40)	円柱表面カーブ変換のスイッチオン
N50 G19	平面の選択

鉤型みぞの作成

N60 G1 X20	みぞの底面までツールをインフィードする
N70 OFFN=12	溝中心線を基準に、みぞ側壁距離を 12 mm に設定する
N80 G1 Z100 G42	みぞ右側壁にアプローチする
N90 G1 Z50	円柱軸に平行なみぞセクション
N100 G1 Y10	周辺に平行なみぞセクション
N110 OFFN=4 G42	みぞ左側壁にアプローチする ; みぞ側壁距離をみぞ中心線から 4 mm に設定する
N120 G1 Y70	周辺に平行なみぞセクション
N130 G1 Z100	円柱軸に平行なみぞセクション
N140 G1 Z105	G40 みぞ側壁から後退する
N150 G1 X25	後退
N160 TRAFOOF	
N170 G0 X25 Y0 Z105 CC=200	基準点にアプローチする
N180 M30	

7.4 傾斜軸 : TRAANG

プログラミング

TRAANG(a) または TRAANG(a,n)
TRAFOOF

コマンドの説明

TRAANG(a)	最初の宣言傾斜変換軸を起動する
TRAANG(a,n)	n 番目の宣言傾斜変換軸を起動する。n は、2 より大きくなってはならない。TRAANG(a,1) は、TRAANG(a) に対応する。
α	傾斜軸の角度
TRAFOOF	変換オフ

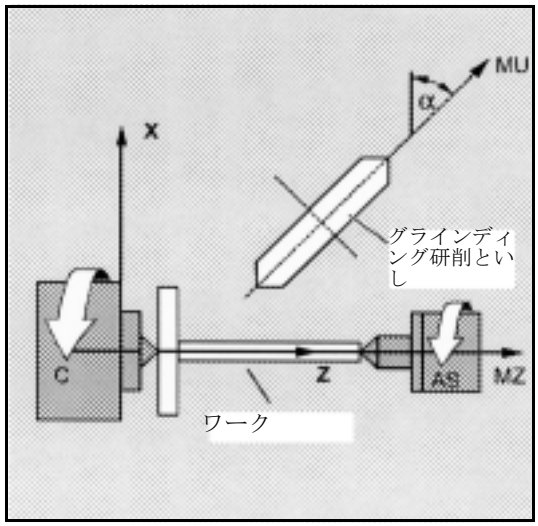
α (角度) が、省略されている、あるいはゼロ値が入力されている場合、変換は以前に選択されたパラメータ設定を使用して起動されます。最初の選択時には、マシンデータのデフォルト値が使用されます。

同じチャンネルで他の変換 (例、TRACYL, TRANSMIT, TRAORI) のうちのいずれかが起動されている場合、アクティブ TRAANG 変換は停止されます。

機能

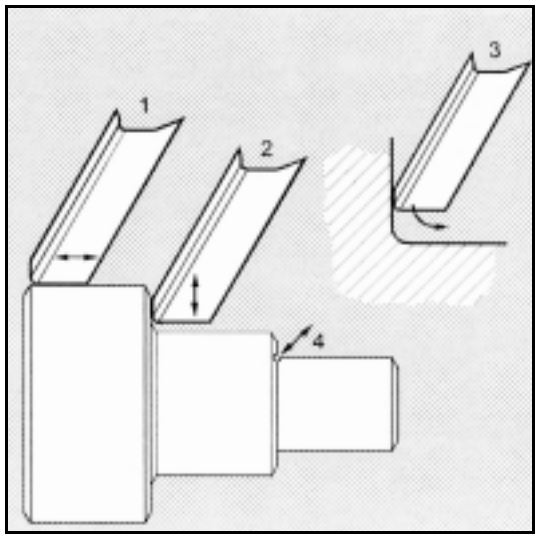
傾斜軸機能は、グラインディング用に作成されたものです。この機能を使用すると、次のことができます：

- 傾斜インフィード軸を使用するマシニング
- デカルト座標系をプログラミングに使用することができます。
- 制御により、デカルト座標系のプログラムされた移動動作が実際のマシン軸の動作に変換されます（標準設定）：傾斜インフィード軸



次のようなマシニングオペレーションができます：

1. 縦方向のグラインディング
2. 横方向のグラインディング
3. 特異輪郭のグラインディング
4. 傾斜みぞのグラインディング



次の設定は、マシンデータに定義されます。

- マシン軸と傾斜軸の間の角度
- "inclined axis" （傾斜軸）機能に宣言された座標系の原点を基準にしたツールゼロの位置
- 補正動作用の平行軸上で利用できる速度予約
- 補正動作用の平行軸上で利用できる軸加速予約

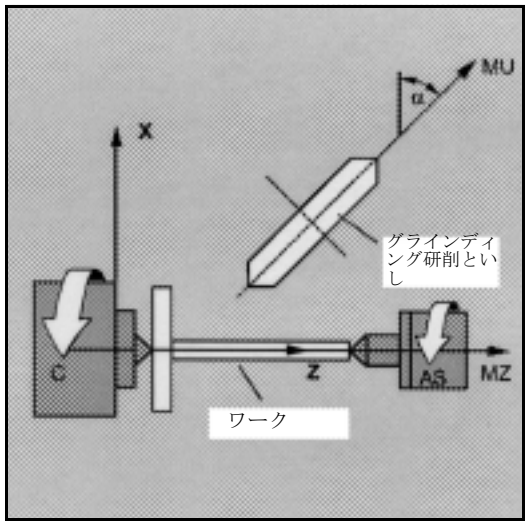
軸構成

デカルト座標系にプログラムするために、デカルト座標系と実際のマシン軸 (MU, MZ) の関係を制御装置に宣言する必要があります：

- ジオメトリ軸の名前
- ジオメトリ軸のチャンネル軸への割当て
  - 一般の場合
    （傾斜軸が有効でない場合）
  - 傾斜軸が有効な場合
- チャンネル軸のマシン軸番号への割当て
- 主軸の同定
- マシン軸名の割当て

この手順は、"Inclined axis active" （傾斜軸有効）を除き、通常の軸構成の場合と同様です。

プログラミング例



N10 G0 G90 Z0 MU=10 G54 F5000 -> -> G18 G64 T1 D1	ツールの選択、クランピング補正、平面の選択
N20 TRAANG(45)	傾斜インフィード変換軸を起動する
N30 G0 Z10 X5	基準点へアプローチする
N40 WAITP(Z)	揺動用の軸をイネーブルにする
N50 OSP[Z]=10 OSP2[Z]=5 OST1[Z]=-2 -> -> OST2[Z]=-2 FA[Z]=5000 N60 OS[Z]=1 N70 POS[X]=4.5 FA[X]=50 N80 OS[Z]=0	寸法に対して行われた揺動 (振幅に関しては、セクション 11 を参照のこと)
N90 WAITP(Z)	振幅軸を位置決め軸としてイネーブルにする
N100 TRAFOOF	スイッチオフ変換
N110 G0 Z10 MU=10	後退
N120 M30	

-> 1 つのブロックにプログラミングします

7.5 変換選択の追加条件

変換の選択は、パートプログラムまたは MDA を使用して行うことができます。注記：

- 中間モーションブロックが挿入されていないこと（面取り／半径）
- スプラインブロック順序が完了していること。完了していない場合、アラームが出力されます。
- ツール微オフセットが停止していること（ FTOCF)。停止していない場合、アラームが出力されます。
- ツール径補正が停止していること（ G40)。停止していない場合、アラームが出力されます。
- 制御装置が、変換に起動されたツールの長さ補正を含んでいること。
- 制御装置によって、変換の前に有効になっていた現在のフレームを停止されていること。
- 制御装置によって、変換に含まれる軸に関する有効作業エリア制限が選択解除されていること(WALIMOF に対応する)。
- 保護ゾーンモニタリングが停止されていること。
- 連続パスモードおよび近似位置決めが中断されていること。
- マシンデータに指定されている全ての軸が、ブロック同期化されていること。
- 交換された軸が元に変更し戻されていること、元に戻されていない場合にはアラームが出力されます。
- 依存軸を使用して、メッセージが出力されます。

ツール変更

ツール変更は、ツール径補正が選択解除されていなければ実行することができません。

ツールの長さ補正の変更とツール径補正の選択／選択解除の変更は、必ず別のブロックにプログラミングしてください。

フレーム変更

どの命令も、基本座標系に適用することができます（ FRAME 、ツール径補正）。G91 を使用するフレーム変更（寸法増分）は、有効になっていない場合の変更と同様、個別に扱うことはできません。移動される増分は、新規フレームのワーク座標系で計算されます --- 以前のブロックで有効なフレームとは無関係です。

例外

変換に関係する軸は、以下の用途に使用できません

- プリセット軸（アラーム）
- 固定点へのアプローチ（アラーム）
- 基準化（アラーム）

7.6 変換の停止 : TRAFOOF

プログラミング

TRAFOOF

コマンドの説明

TRAFOOF	全てのアクティブ変換／フレームを停止する

機能

TRAFOOF が発行されると、アクティブな変換およびフレームが全て再び停止されます。

その後に要求されるフレームは、プログラムし直さなければ起動できません。

（注）変換の非選択の場合にも起動の場合と同様の条件が適用されます（7.5 変換選択の追加条件のセクションを参照してください）。

7.7 チェーニングされた変換

2 つの変換が順次サポートされるので、最初の変換から生じた軸の動作要素はチェーニングされた第 2 の変換用の入力データとなります。第 2 の変換から生じた動作要素は、マシン軸に対して有効となります。

- チェーニングには 2 つの変換を含めることができます。
- 第 2 の変換は常に "Inclined axis" （傾斜軸） (TRAANG) となります。
- 最初の変換は、次のいずれかとなります :
  - 向き調整変換 (TRAORI), 万能フライスヘッドも含む
  - TRANSMIT
  - TRACYL
  - TRAANG

アプリケーション

- 傾斜グラインディングハンドル（例えば、ツールグラインディング）を使用して、円柱オペレーション（TRACYL）の側壁ライン側線としてプログラムされた輪郭のグラインディング

- 傾斜研削といしを使用して TRANSMIT によって作成された非円形輪郭の仕上げ切削

チェーニングされた変換用の起動コマンドを使用するための前提条件は、個々の変換が相互にチェーニングされていて、起動するそのチェーニングされた変換がマシンデータによって定義されている、ということです。

変換のそれぞれの説明に述べられている追加条件および例外は、チェーニングされた変換にも適用されます。

追加説明

変換用のマシンデータの構成方法については、
Description of Functions（機能の説明）: M1 および F2
を参照してください。

機械メーカー (MH7.1)

マシンデータによって事前定義されている変換については、機械メーカーの仕様書をお読みください。

変換および変換のチェーニングはオプションです。さまざまな制御装置でチェーニングして使用できる変換については、カタログを参照してください。

チェーニングされた変換には次のコマンドが使用されます：

起動する場合、TRACON

停止する場合、TRAFOOF

起動

プログラミング

TRACON(trf, par) チェーニングされた変換が起動されます。

パラメータの説明

trf	チェーニングされた変換数： 0 または 1 （最初または唯一のチェーニングされた変換の場合） 何もプログラミングされていない場合も、0 または 1 がプログラミングされている場合と同じ値になります。すなわち、最初／唯一の変換が起動されます。 2 （2 番目のチェーニングされた変換の場合） （0 ～ 2 以外の値が入力されると、アラームが出力されます）

par	1 つまたは複数のパラメータが、チェーニング内の変換のためにコンマで区切られている場合、そのパラメータには、例えば傾斜軸の角度などを入力する必要があります。パラメータがセットされていない場合、デフォルト設定または最後に使用されたパラメータが有効になります。 万が一デフォルト設定を先のパラメータに対して有効にする場合、指定されたパラメータが要求される順序で確実に評価されるように、必ずコンマを挿入してください。特に、ただ 1 つのパラメータを指定する場合には、たとえ trf を指定する必要がない場合でも、例えば、TRACON(, 3.7) のように、必ずコンマを先につけてください。

機能

チェーニングされた変換が起動されます。他の以前に起動されていた変換は、TRACON() を使用して必然的に停止されます。

ツールは常にチェーニングの最初の変換に割当てられます。その後、次の変換がアクティブツール長がゼロであるかのような挙動をします。マシンデータにセットされたツールの基本長さ (_BASE_TOOL_) のみが、チェーニングの最初の変換に対して有効となります。

プログラミング

TRAFOOF

機能

このコマンドは、最後に起動された（チェーニングされた）変換を停止します。

7.8 切換え可能なジオメトリ軸 , GEOAX

プログラミング

GEOAX(n, チャンネル軸 ,n, チャンネル軸 ,...)

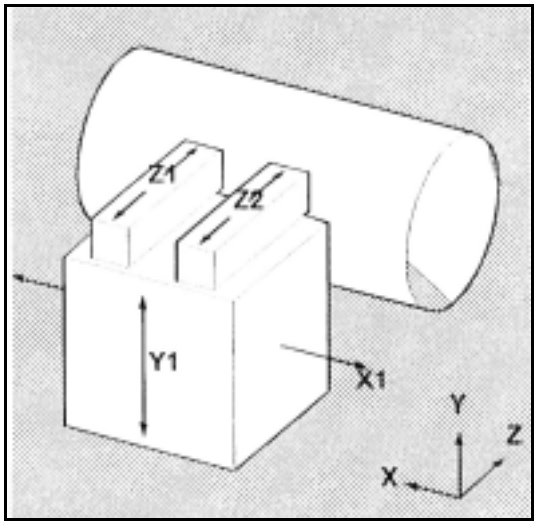
GEOAX()

パラメータの説明

GEOAX(n, チャンネル軸 ,n, チャンネル軸 ,...)	ジオメトリ軸を切替える。
GEOAX()	基本ジオメトリ軸構成をコールする
n	他のチャンネル軸に割り当てるジオメトリ軸の数 (n=1, 2 または 3) n=0: ジオメトリ軸グループから指定されたチャンネル軸を置換せずに削除する。
チャンネル軸	ガントリ軸グルーピングに含めるチャンネル軸の名前

機能

機能 "Switchable geometry axes" （切換え可能なジオメトリ軸）を使用すると、マシンデータに設定されたジオメトリ軸グループをパートプログラムから変更することができます。同期補助軸として定義されているチャンネル軸は、どのジオメトリ軸にでも置換することができます。



例:

ツールスライドをチャンネル軸 X1,

Y1, Z1, Z2 を介して移動させることができます。また同様に、軸 Z1 および Z2 を、パートプログラム内のジオメトリ軸 z として使用することができます。パートプログラムに設定された

GEOAX が軸同士を切替えます。

起動後、接続

X1, Y1, Z1 が有効になります（マシンデータに定義できます)。

N100 GEOAX (3,Z2)

N110 G1 .....

N120 GEOAX (3,Z1)

チャンネル軸 Z2 は、Z 軸として機能します。

チャンネル軸 Z1 は、Z 軸として機能します。

動作

ジオメトリ軸数

コマンド GEOAX(n, チャンネル軸 ...) での、数 n は、結果的に指定されたチャンネル軸が割当てられるジオメトリ軸を示します。

ジオメトリ軸数 1 から 3 (X 軸 , Y 軸および Z 軸) を使用し、

チャンネル軸へ切替えることができます。

n = 0 の場合、ジオメトリ軸グルーピングから割当てられたチャンネル軸が削除されます。ジオメトリ軸の再割当てはありません。

ジオメトリ軸グループ内の切替えの結果置換された軸は、そのチャンネル名を介して行われた切替えの後、補助軸としてプログラムすることができます。

ジオメトリ軸が切替えられると、フレーム、保護ゾーンおよび作業エリア制限は全て削除されます。

ハンドルオフセット (DRF) または外部ゼロオフセットは、切替え後も継続して有効のままです。

軸位置の変換

すでに割当てられているチャンネル軸に新規に軸数を割当てることにより、ジオメトリ軸グループ内に位置変更をプログラムすることができます。

N... GEOAX (1, XX, 2, YY, 3, ZZ)

N... GEOAX (1, U, 2, V, 3, W)

チャンネル軸 XX は最初のジオメトリ軸で , YY は 2 番目、ZZ は 3 番目のジオメトリ軸です。チャンネル軸 U は最初のジオメトリ軸で、V は 2 番目、W は 3 番目のジオメトリ軸です。

位置決めおよび制限

1. 次の場合、ジオメトリ軸切り替えはできません :
   - 有効な変換 ,
   - 有効なスプライン補間 , (プログラミング編 基本説明書 : セクション 8)
   - 有効なツール半径補正 , (プログラミング編 基本説明書 : セクション 8)
   - 有効なツール微補正
2. ジオメトリ軸とチャンネル軸に同じ名前がつけられている場合、ジオメトリ軸を切り替えることはできません。
3. 切替えに関係する軸はいずれもブロックリミットを越えて継続するような動作に関わるべきではないが、これはタイプ A の位置決め軸または後続の軸を使用する場合には有効です。
4. GEOAX コマンドは、すでにこのコマンドが実行されたジオメトリ軸を置換する場合にのみ使用することができます (すなわち、新規軸の定義には使用できません)。

切替えの停止

コマンド GEOAX() は、ジオメトリ軸グループの基本構成をコールします。

POWER ON して、運転モード基準点アプローチに切替えると、基本構成は自動的に有効になります。

説明

切替え手順およびツール長補正

有効なツール長補正は、切替え後も有効のままです。新規に割当てられた軸や位置が切替えられたジオメトリ軸に影響を与えることになります。これらのジオメトリ軸に対して最初の移動コマンドが出されます。したがって、ツール長補正とプログラムされた移動パスの合計が、移動されるパスの演算結果となります。

切替え後も軸グループにその位置を維持するジオメトリ軸は、ツールの長さ補正に関してもそのステータスを維持します。

ジオメトリ軸構成および変換変更

有効な変換に適応されるジオメトリ軸構成（マシンデータに定義されています）は、機能 "Switchable geometry axes" （切替え可能なジオメトリ軸）を使用して変更することはできません。

変換を使用してジオメトリ軸構成を変更する必要がある場合、別の変換をプログラミングしなければなりません。

GEOAX を使用して変更されたジオメトリ軸構成は、変換を起動すると削除されます。

変換用のマシンデータ設定とジオメトリ軸切替え用のマシンデータ設定が折衝する場合、変換用の設定が優先されます。

例 :

変換が有効です。マシンデータの設定にしたがって、その変換は RESET 後も有効ですが、同時に RESET はジオメトリ軸の基本構成を再設定することになります。この場合、変換を使用して定義されたジオメトリ軸構成は変更されません。

プログラミング例

マシンには、XX, YY, ZZ, U, V, W の 6 つのチャンネル軸があります。このマシンデータのジオメトリ軸構成の基本設定は次のようになります：

チャンネル軸 XX = 最初のジオメトリ軸 (X 軸)

チャンネル軸 YY= 2 番目のジオメトリ軸 (Y 軸)

チャンネル軸 ZZ = 3 番目のジオメトリ軸 (Z 軸)

N10 GEOAX()	ジオメトリ軸の基本構成が有効。
N20 G0 X0 Y0 Z0 U0 V0 W0	全ての軸がポジション 0 へ迅速に移動中
N30 GEOAX(1,U,2,V,3,W)	チャンネル軸 U は最初のジオメトリ軸 (X), V は 2 番目のジオメトリ軸 (Y), W が 3 番目のジオメトリ軸 (Z).
N40 GEOAX(1,XX,3,ZZ)	チャンネル軸 XX は最初のジオメトリ軸 (X), ZZ は 3 番目のジオメトリ軸 (Z). チャンネル軸 V は 2 番目のジオメトリ軸 (Y).
N50 G17 G2 X20 I10 F1000	X, Y 平面内の一周。チャンネル軸 XX および Y が移動する。
N60 GEOAX(2,W)	チャンネル軸 W が 2 番目の ジオメトリ軸 (Y) となる。
N80 G17 G2 X20 I10 F1000	X, Y 平面内の一周。チャンネル軸 XX および W が移動する。
N90 GEOAX()	初期設定にリセット
N100 GEOAX(1,U,2,V,3,W)	チャンネル軸 U は最初のジオメトリ軸 (X), V は 2 番目のジオメトリ軸 (Y), W は 3 番目のジオメトリ軸 (Z).
N110 G1 X10 Y10 Z10 XX=25	チャンネル軸 U, V, W のそれぞれがポジション 10 に移動する , XX は補助軸としてポジション 25 に移動する。
N120 GEOAX(0,V)	V はジオメトリ軸グループから削除される。U および W は、依然最初のジオメトリ軸 (X) と 3 番目のジオメトリ軸 (Z). 2 番目のジオメトリ軸 (Y) はまだ再割り当てされていない。
N130 GEOAX(1,U,2,V,3,W)	チャンネル軸 U は依然最初のジオメトリ軸 (X), V が 2 番目のジオメトリ軸となり (Y) そして W が 3 番目のジオメトリ軸 (Z) となる。
N140 GEOAX(3,V)	V が 3 番目のジオメトリ軸 (Z) となり , W が上書きされる。したがってジオメトリ軸グループから削除される。2 番目のジオメトリ軸 (Y) は依然再割り当てされていない。

8 ツールオフセット

8.1 オフセットメモリ

オフセットメモリの構造

データフィールドはすべて、T 番号および D 番号を使って呼び出すことができます（例外、"Flat D No." （フラット D 番号）)。また、この番号にはツールのジオメトリデータの他に、ツールタイプなどのデータも含まれます。

ツール管理が NCK の外部で行われている場合、"Flat D No. structure" （フラット D 番号構成）が使用されます。その場合、D 番号はツールに割当てられることなく関連のツールオフセットブロックを使用して作成されます。

T を使用して、パートプログラムにプログラミングすることもできます。

ただしその場合、この T はプログラムされた D 番号には関係しません。

ジオメトリ変数の入力がいくつかあります（例えば、長さ 1 や半径など)。これらを合計して計算に使用する値（例えば、総長さ 1 、総半径など）が作成されます。

要求されないオフセット値には、ゼロ値が割当てられます。

オフセットメモリの P1 から P25 までのそれぞれの値は、システム変数を介してプログラムから読み出したりまた書き込むことができます。

ツールパラメータ番号 (DP)	意味	コメント
$TC_DP 1	ツールタイプ	概要については、リストを参照のこと
$TC_DP 2	ツールポイント方向	旋盤加工ツール用のみ
ジオメトリ	ツール長さ補正	
$TC_DP 3	長さ 1	タイプと平面によっての計算
$TC_DP 4	長さ 2	
$TC_DP 5	長さ 3	
ジオメトリ	**半径**	
$TC_DP 6	半径	
$TC_DP 7	スロッティングソーのスロット幅 b, フライス加工ツール用の丸め半径	
$TC_DP 8	オーバハング k	スロッティングソー 用のみ
$TC_DP 11	円すいフライス加工ツール用の角度	
磨耗	**ツール長および半径補正**	
$TC_DP 12	長さ 1	
$TC_DP 13	長さ 2	
$TC_DP 14	長さ 3	
$TC_DP 15	半径	
$TC_DP 16	スロッティングソー用のスロット幅 b, フライス加工ツール用の丸め半径	
$TC_DP 17	オーバハング k	スロッティングソー 用のみ
$TC_DP 20	円すいフライス加工ツール用の角度	
底面寸法／アダプタ	**ツールの長さ補正**	
$TC_DP 21	長さ 1	
$TC_DP 22	長さ 2	
$TC_DP 23	長さ 3	
テクノロジ		
$TC_DP 24	クリアランス角度	旋盤ツール用

8.2 ツール管理用の言語コマンド

コマンドの説明

T="WZ"	名前を使用してツールを選択する
NEWT("WZ",DUPLO_NO)	新規ツールを作成する、duplo 番号（オプション）
DELT("WZ",DUPLO_NO)	ツールを削除する ,duplo 番号（オプション）
GETT("WZ",DUPLO_NO)	T 番号を確定する
SETPIECE(x,y)	ピース番号をセットする
GETSELT(x)	事前に選択されたツール番号 (T No.) を読み出す
"WZ"	ツール名
DUPLO_NO	量
x	主軸番号 , エントリはオプション

ツール管理を使用すると、名前（例えば、T="DRILL" 、T="123" など）によってツールを生成しコールすることができます。

NEWT 機能

NEWT 機能を使用すると、NC プログラム内の名前を使用して新規ツールを生成することができます。この機能は作成された T 番号を自動的に送り返すので、その番号を使用してツールをアドレスすることができます。

リターンパラメータ =NEWT("WZ", DUPLO_NO)

duplo　番号が指定されていない場合、ツール管理よって自動的に生成されます。

例 :

DEF INT DUPLO_NO	
DEF INT T_NO	
DUPLO_NO = 7	
T_NO=NEWT("DRILL", DUPLO_NO)	duplo　番号 7 を使用して "DRILL" を生成する。生成された T 番号は T_NO に格納される。

DELT 機能

DELT 機能を使用して、T 番号を参照にせずにツールを削除できます。

DELT("WZ",DUPLO_NO)

GETT 機能

GETT 機能は、この名前によってしかわからない
ツール用のツールデータをセットするのに必要な T
番号を送り返します。

リターン Parameter=GETT("WZ", DUPLO_NO)

指定された同じ名前を持ついくつかのツールが存在
する場合、最初に使用される可能性がある T 番号が
送り返されます。

リターン Parameter =-1: ツール名や duplo 番号をツー
ルに割り当てることはできません。

例 :

T="DRILL"	
R10=GETT("DRILL", DUPLO_NO)	duplo 番号 = DUPLO_NO を使用する DRILL 用の T 番号を送り返す

"DRILL" は、NEWT または
$TC_TP1[ ] を使用して最初に宣言される必要がある

$TC_DP1[GETT("DRILL", DUPLO_NO),1]=100	ツール名を使用してツールパラメータを書き込む

SETPIECE 機能

この機能を使用すると、部品数モニタリングデータ
をアップデートすることができます。
この機能は、決められた主軸番号に対して
SETPIECE が最後に起動されてから後に変更された
ツールエッジを全てカウントします。

SETPIECE(x,y)

x	完了したワークの数量
y	y 主軸番号 , 0 はマスタ主軸を表す (デフォルト設定)

GETSELT 機能

この機能は、主軸に事前選択されたツールの T 番号を送り返します。

この機能を使用すると、M6 より先にツールオフセットデータにアクセスすることができるので、メインラン同期化の設定がいくぶん早くなります。

ツール管理を使用するツール変更の例

T1　ツールを事前に選択します、すなわちツールマガジンはマシニングに平行なツール位置に設定されます。

M6　事前に選択されたツールをロードします (マシンデータの設定により、M6 を使用せずにプログラムすることもできます).

例 :

T1 M6	ツール 1 のロード
D1	ツールの長さ補正の選択
G1 X10 ...	T1 を使用するマシン
T="DRILL"	ドリルの事前選択
D2 Y20 ...	切削エッジ T1 の変更
X10 ...	T1 を使用する切削
M6	ツールドリルのロード
SETPIECE(4)	完了したワーク数
D1 G1 X10 ...	ドリルを使用する作業

プログラミング編 基本説明書の付録のシステム変数のリストに、ツール管理に必要な全変数の一覧を掲載しています。

8.3 オンラインツールオフセット PUTFTOCF, PUTFTOC, FTOCON, FTOCOF

プログラミング

FCTDEF(Polynomial no., LLimit, ULimit,a0,a1,a2,a3)

PUTFTOCF(Polynomial No., Ref_value, Length1_2_3, Channel, Spindle)

PUTFTOC(Value, Length1_2_3, Channel, Spindle)

FTOCON

FTOCOF

コマンドの説明

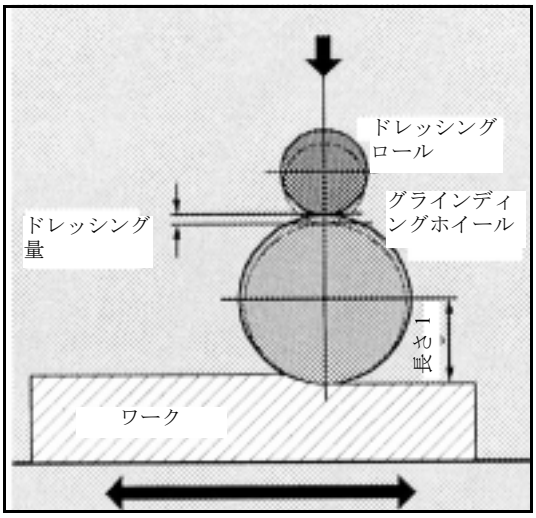
PUTFTOCF	オンラインでツールオフセットを連続して書き込む
FCTDEF	PUTFTOCF 機能用のパラメータを定義する
PUTFTOC	オンラインでツールオフセットを直接書き込む
FTOCON	オンラインでツールオフセットを起動する
FTOCOF	オンラインでツールオフセットを停止する

パラメータの説明

Polynomial_No.	値 1 ～ 3: 最大 3 つの多項式を同時にプログラムすることができる；多項式は最高 3 次まで
Ref_value	オフセット値が生成される基準値
Length1_2_3	ツールオフセット値が追加される磨耗パラメータ
Channel	ツールオフセットが起動されているチャンネルの番号；チャンネルが現チャンネルと異なる場合にのみ指定される
Spindle	オンラインツールオフセットが作動する主軸の番号；有効でないグラインディングホイールに対して指定する場合にのみ必要
LLimit	Upper limit 上限
ULimit	Lower limit 下限
a0,a1,a2,a3	多項式機能の係数
Value	ツールオフセット値が追加される値

機能

この機能は、オンラインのツールの長さ補正によってマシニングの結果生じるツールオフセット用の公差を即時に作成します（CD ドレッシング（研削）：グラインディングホイールは切削に平行してドレッシングされます)。ツールの長さ補正は、マシニングチャンネルまたは平行チャンネル（ドレッサチャンネル）から変更できます。



オンラインツールオフセットが適用できるのは、グラインディングホイールだけです。

オンライン TO に関する一般情報

ドレッシング処理のタイミングによって、次の機能を使用してオンラインツールオフセットに書き込むことができます：

- 連続書き込み , ノンモーダル : PUTFTOCF
- 連続書き込み , モーダル : ID=1 DO FTOC
  (synchronized action を参照してください)
- 不連続書き込み : PUTFTOC

評価機能の起動に続いての連続書き込み（各補間パルスに対して）の場合、それぞれの変更は、セットポイントがジャンプしないように、磨耗メモリで相加的に計算されます。

両方の場合：

オンラインオフセットは、磨耗パラメータの主軸または長さ 1, 2, 3 のそれぞれに基づいて挙動します。

現在の平面を基準にして、ジオメトリ軸に長さを割当てます。

GWPSON または TMON 使用するツールデータを基準にして、主軸はツールに割当てられます、ただし有効なグラインディングホイールは除きます（参照、プログラミング編 基本説明書)。

オフセットは、現在のツールサイド用または有効でないツールの左手側ツールサイド用の磨耗パラメータには必ず適用されます。

いくつかのツールサイド用のオフセットが同一である場合、チェーニングルールによってその値は第 2 のツールサイドに自動的に転送されます（ユーザーズマニュアル操作編の説明を参照してください)。

マシニングチャンネル用のオンラインオフセットが定義されている場合、マシニングプログラムから、あるいはオペレータの入力によって、このチャンネル上の現在のツール用の磨耗値を変更することはできません。

オンラインツールオフセットは、ツールモニタリング (TMON) および心なしグラインディングの他に、一定のグラインディング周囲速度 (GWPS) に関しても適用されます。

動作

PUTFTOCF = 連続書き込み

ドレッシングプロセスはマシニングと同時に行われます：

ドレッサロールまたはドレッサダイアモンドを使用して、グラインディングホイールの端から端まで完全にドレッシングします。

マシニングおよびドレッシングを別のチャンネルで実行することもできます。チャンネルがプログラムされていない場合、オフセットは有効なチャンネル内で有効となります。

PUTFTOCF(Polynomial_No., Ref_value, Length1_2_3, Channel, Spindle)

ツールオフセット、1 次、2 次および 3 次の多項式機能（FCTDEF を使用して事前に定義しておく必要があります）を基準にしてマシン軸チャンネル上で連続して変更されます。例えば実際の値の変更のような、このオフセットは、"Reference value" （基準値）変数から生成されます。

主軸番号がプログラミングされていない場合、オフセットは有効なツールに適用されます。

FCTDEF 機能用のパラメータのセット

このパラメータは個別のブロックに定義されます :

FCTDEF(Polynomial_NO., LLimit, ULimit,a0,a1,a2,a3)

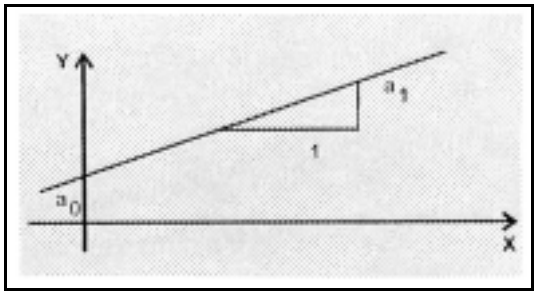
多項式は、1 次、2 次および 3 次の多項式となります。

制限は、制限値と同意です (LLimit = 下限 , ULimit = 上限)

例 :

傾き 1 を持つ直線 (y = a0 + a1x)

FCTDEF(1, -1000, 1000, -$AA_IW[X], 1)



オンラインオフセットを不連続に書き込む : PUTFTOC

このコマンドを使用すると、オフセット値を一度だけ書き込むことができます。このオフセットはターゲットチャンネル上で即時に起動されます。

PUTFTOC の応用 :

グラインディングホイールは平行チャンネルからドレッシングすることができますが、マシニングと同時に行うことはできません。

PUTFTOC(Value, Length1_2_3, Channel, Spindle)

指定された長さ 1 、2 または 3 のオンラインツールオフセットはその指定された値によって変更されます、すなわちその値は磨耗パラメータに追加されます。

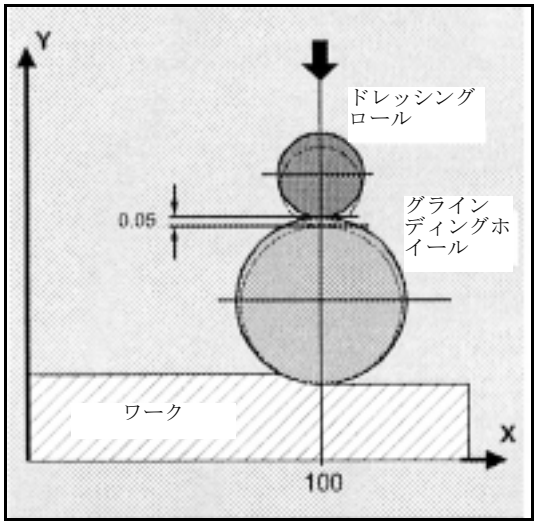
オンラインツールオフセットを含む : FTOCON, FTOCOF

ターゲットチャンネルは、FTOCON が有効な場合にのみ、オンラインでツールオフセットを受け取ることができます。

- FTOCON は、オフセットが起動されるチャンネルに書き込む必要があります。
  FTOCOF を使用する場合、オフセットは適用されません。ただし、PUTFTOC を使用して書きこまれた完全値は、ツールエッジ別オフセットデータ内で修正されます。

- FTOCOF は常にリセット設定となります。
- PUTFTOCF は以降の移動ブロック上で有効となります。
- オンラインツールオフセットは、FTOC を使用してモーダルに選択することができます。詳細については、
  10. シンクロナイズドアクション挙動セクションを参照してください。

プログラミング例



タスク

次のパラメータを使用する表面グラインディングマシン上では、グラインディングホイールは x100 でグラインディング動作がスタートした後、0.05 だけドレッシングされることになります。ドレッシング量は、連続したオンラインオフセット書き込みでアクティブにすることができます。

Y: グラインディングホイール用のインフィード軸

V: ドレッサロール用のインフィード軸

Machine: 軸 X, Z, Y を使用するチャンネル 1

Dress: 軸 V を使用するチャンネル 2

チャンネル 1 のマシニングプログラム :

%_N_BEARB_MPF	
...	
N110 G1 G18 F10 G90	基本設定
N120 T1 D1	現在のツールを選択する
N130 S100 M3 X100	主軸起動 , スタート位置に移動する
N140 INIT (2, "DRESS", "S")	チャンネル 2 でドレッシングプログラムを選択する
N150 START (2)	チャンネル 2 でドレッシングプログラムをスタートする
N160 X200	目的の位置へ移動する
N170 FTOCON	オンラインオフセットを起動する
N... G1 X100	連続マシニング
N...M30	

チャンネル 2 のドレッシングプログラム :

%_N_ABRICHT_MPF	
...	
N40 FCTDEF (1, -1000, 1000, -$AA_IW[V], 1)	機能を定義する : 直線
N50 PUTFTOCF (1, $AA_IW[V], 3, 1)	オンラインオフセットを連続して書き込む : 現在のグラインディングホイールの長さ 3 が、V 軸の動作から生成され、チャンネル 1 で修正される。
N60 V-0.05 G1 F0.01 G91	ドレッシングのインフィード動作 , PUTFTOCF 過去のブロックでのみ有効である
...	
N...M30	

ドレッシングプログラム、モーダル：

%_N_ABRICHT_MPF	
FCTDEF(1,-1000,1000,-$AA_IW[V],1)	機能を定義する：
ID=1 DO FTOC(1,$AA_IW[V],3,1)	オンラインツール補正を選択する：V 軸の実際の値は多項式 1 の入力値である；その演算結果に、チャンネル 1 の有効なグラインディングホイールの長さ 3 がオフセット値として加えられる。
WAITM(1,1,2)	マシニングチャンネルとの同期化
G1 V-0.05 F0.01, G91	ドレッシング用のインフィード動作
G1 V-0.05 F0.02	
...	
CANCEL(1)	オンラインオフセットの選択解除
...	

8.4 3D ツールオフセットの起動

説明

CUT3DC	周囲フライス加工用の 3D 半径オフセットの起動
CUT3DFS	一定向き調整を使用する表面フライス加工用の 3D ツールオフセット。ツールの向き調整は G17 ～ G19 によって確定され Frames （フレーム）に影響をうけない。
CUT3DFF	一定向き調整を使用する表面フライス加工用の 3D ツールオフセット。ツールの向き調整は G17 ～ G19 によって確定された方向となり、Frames （フレーム）によって影響をうけない。
CUT3DF	向き調整チェンジを使用する表面フライス加工用の 3D ツールオフセット (有効な 5 軸変換の場合のみ).
G40 X Y Z	停止する : ジオメトリ軸を使用する直線ブロック G0/G1
ISD=value	深さの挿入

このコマンドはモーダルで、CUT2D および CUT2DF と同じグループに属しています。

このコマンドは、現在の平面で次の動作が実行されてはじめて選択が解除されます。このコマンドは常に G40 に適用されます。CUT コマンドとは無関係です。

機能

ツール向き調整の変更は、円柱形ツール用のツール径補正に考慮されます。

3D ツール径補正には、2D ツール半径補正と同じコマンドが適用されます。G41/G42 を使用する場合、右手／左手補正が動作の方向に指定されます。アプローチ方法は必ず NORM です。

例

N10 A0 B0 X0 Y0 Z0 F5000	
N20 T1 D1	ツールのコール , ツールオフセット値をコールする
N30 TRAORI(1)	変換の選択
N40 CUT3DC	3D ツール径補正の選択
N50 G42 X10 Y10	ツール径補正の選択
N60 X60	
N70 ...	

追加説明

3D ツール径補正を使用する場合、中間ブロックが使用できます。2 1/2D ツール径補正用のルールが適用されます。

2 1/2D ツール径補正

3D ツール径補正は、5 軸変換が選択されて初めてアクティブになります。

円のブロックは必ず外側コーナーで挿入されます。

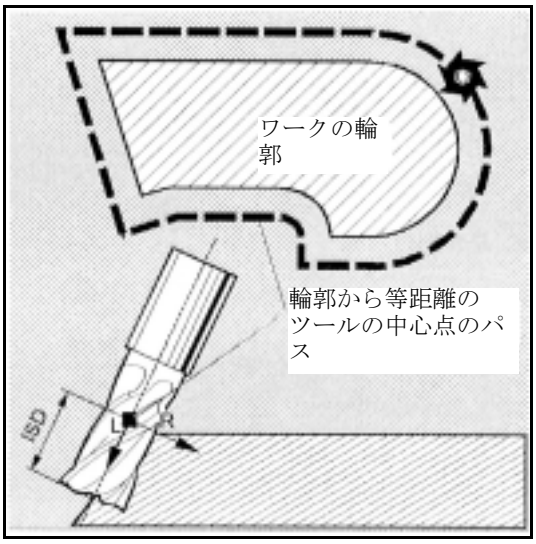
G450/G451 は有効ではありません。

DISC コマンドは無視されます。

2 1/2 D ツール径補正と 3D ツール径補正の違い

3D 径補正では、ツールの向き調整は変更可能です。

2 1/2 D ツール径補正は、一定の向き調整を使用するツールを使用することを前提としています。

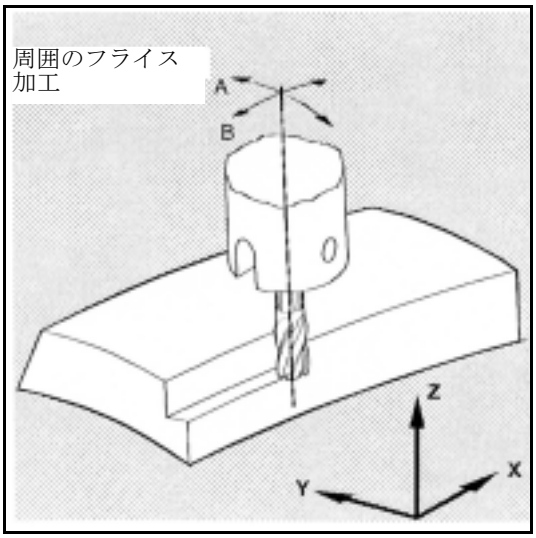


空間のツールの向き調整に自由度 5 を使用することができるので、3D ツール径補正は、5D ツール径補正とも呼ばれます。

周辺のフライス加工

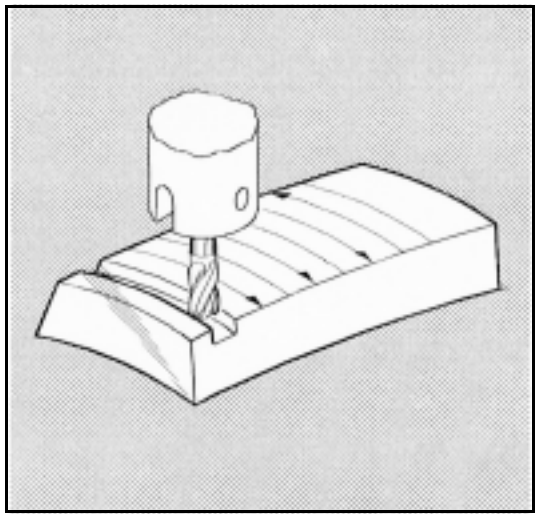
ここで使用される種類のフライス加工は、パス（ガイドライン）と対応する向き調整を定義することによって実行されます。

このタイプのマシニングでは、パス上のツールの形は関係ありません。ツール挿入点の半径のみが、決定要素となります。



3D TRC 機能は、円柱形ツールでのみ使用できます。

表面フライス加工



このタイプの 3D フライス加工では、ワーク表面上の 3D パスをラインバイラインに（1 列 1 列）定義する必要があります。

通常、ツールの形および寸法を用いて、CAM で計算されます。

NC ブロックの他に、ポストプロセッサがツールの向き調整（5 軸変換が有効である場合）ならびに要求された 3D ツールオフセット用の G コードをパートプログラムに書き込みます。

この機能により、マシンオペレータは NC パスの計算に使用されるツールよりいくぶん小さなツールを使用する選択肢が与えられます。

例 :

NC ブロックは、10 mm ミルを使用して計算されています。

その場合、ワークもまた 9.9 mm の直径のミルで加工することもできますが、表面プロファイルはまったく違ったものとなります。

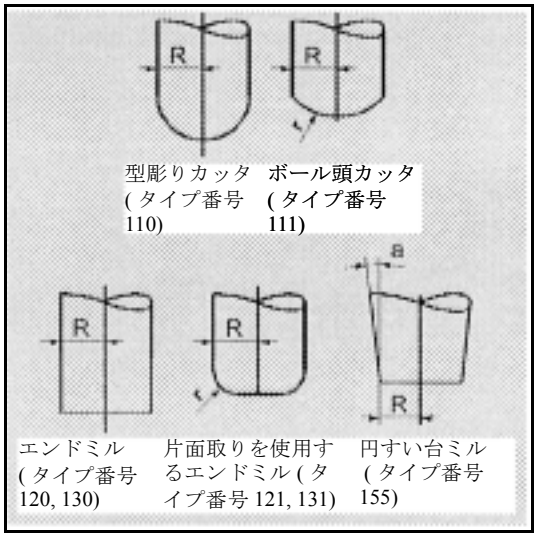
ミルの形、ツールデータ

下の表に、表面フライス加工オペレーションに使用されるツールの形とツールデータ制限値の概要を挙げています。

ツールシャフトの形は考慮されませんが、120 と 150 のツールが効果の点では同じです。

NC プログラムで表に列挙したものとは違うタイプ番号が使用される場合、システムは自動的にタイプ番号 110 の型彫りカッタを使用します。

ツールデータ制限値を超えると、アラームが出力されます。



型彫りカッタ (タイプ番号 110)　ボール頭カッタ (タイプ番号 111)

エンドミル (タイプ番号 120, 130)　片面取りを使用するエンドミル (タイプ番号 121, 131)　円すい台ミル (タイプ番号 155)

フライス加工ツールの種類	タイプ番号	R	r	a
円柱形ミラー	110	>0	X	X
ボールエンドミル	111	0	>R	X
エンドミル , アングルヘッドカッタ	120, 130	>0	X	X
エンドミル , 片面取りを使用するアングルヘッドカッタ	121, 131	>r	>0	X
円すい台ミル	155	>0	X	>0

X= は評価されません。

ツールの長さ補正

ツールの先端は長さ補正の基準点です（交点、縦方向の軸／表面）

3D ツールオフセット、ツール変更

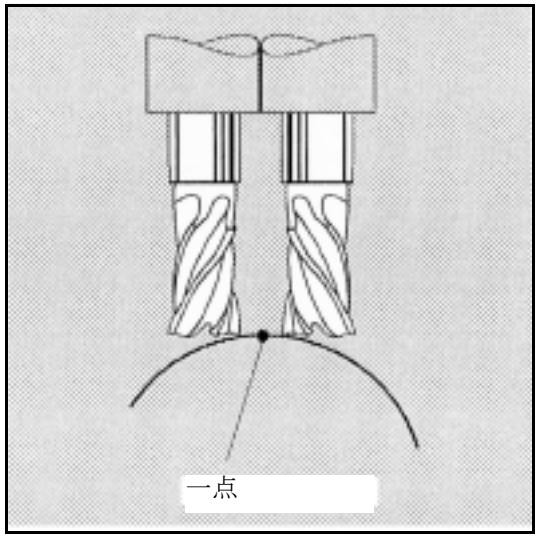
変更されたディメンジョン (R, r, a) を持つ新規のツールまたは別の形を持つツールは、G41 または G42 のプログラミングを通じてしか指定できません (G40 から G41 または

G42 へ変換 , G42 の G41 を再プログラミング)。

これは他のどのツールデータ（例えば、ツールの長さ）にも適応されません。したがって、このデータが適用されるツールは、G41 または G42 を使用せずに適合されるます。

パスの修正

表面フライス加工を使用する場合、右の例に示したようなツール表面上で接触点が「ジャンプ」するとどのようなことが発生するか必ず検査してください。右図では、凸面が垂直に位置決めされたツールを使用して加工されています。



一般原則では、要求されている表面プロファイルを生成するのに適したツールの形とツールの向き調整を選択する必要があります。

したがって、例に示した使用法は境界例とみなすことができます。

このような境界例は制御装置によって監視されています。制御装置は、ツールとノーマル表面ベクトルの間の角度アプローチモーションに基づきマシニングポイントにおける突然の変化を検出します。制御装置は、モーションが実行されるようにこれらの位置に直線ブロックを挿入します。

このような直線ブロックは、マシンデータに保存されているサイド角度用の許容可能な角度範囲に基づいて計算されます。

マシンデータに格納されている制限値が違反された場合、アラームが出力されます。

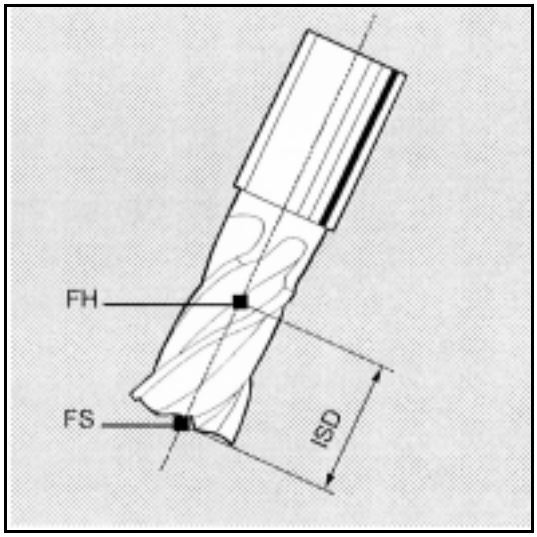
パス曲率

パスの曲率は監視されません。したがって、輪郭を損なうようなタイプのツールは絶対に使用しないでください。

深さの挿入

プログラムコマンド ISD (挿入深さ) を使用して、外周削り操作用のツール挿入深さをプログラミングすることができます。その結果、ツールの外面上のマシニングポイントの位置を変更することができます。

ISD は、カッタ先端 (FS) とカッタ基準点 (FH) の間の距離を指定します。点 FH は、ツール軸に沿ってプログラムされたマシニングポイントを投影することによって生成されます。3D ツール径補正が有効な場合にのみ ISD は評価されます。

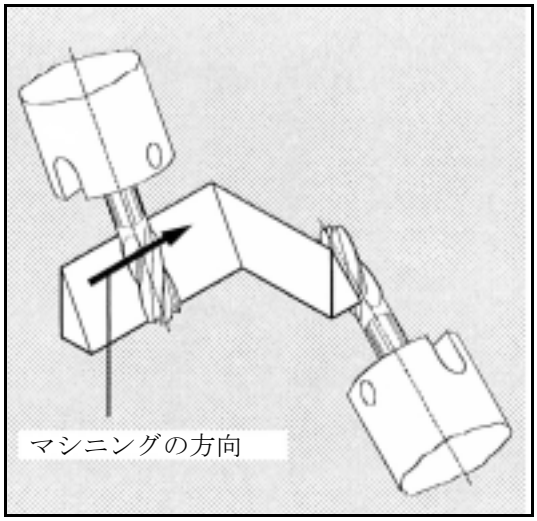


内側コーナー／外側コーナー

内側コーナーと外側コーナーは、別々に取り扱われます。

内側コーナー、外側コーナーという用語は、ツールの向き調整によって変わってきます。

コーナーで向きが変更された場合、コーナーの内側外側はマシニング中に入れ替わります。このような場合、マシニングは停止し、エラーメッセージが出力されます。



3D 補正用の挿入手順 :

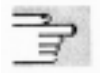
3D 外周削りを使用すると、G コード G450/G451 はコーナーで評価されます : すなわち、オフセット曲線の挿入をしてアプローチされたことになります。この新しい機能は、典型的な CAD 生成プログラムにとっては特に都合の良いものです。それらは短い直線ブロックで構成されていることが多い（滑らかな曲線に近づけるため）ので、近接するブロック間の変化はほぼ接線方向になります。

現在までのところ、輪郭の外側でのツール半径補正を使用する場合、概して円は外側コーナーを周回するように挿入されました。

ほとんど接線的な変化を使用するとこれらのブロックは非常に短くなるので、速度が不必要に落ちることはなくなります。

この場合、2 1/2 D 径補正を使用する場合と同様、関連する曲線は両方とも距離が長くなり、長くなった両方の曲線の挿入をしてアプローチされます。

両方のブロックのオフセット曲線を延長し、さらにツール向き調整に垂直な平面のコーナーでのそれらの挿入を定義することによって、挿入が確定されます。このような挿入がない場合、コーナーはこれまでどおりに取り扱われます、すなわち円は挿入されません。

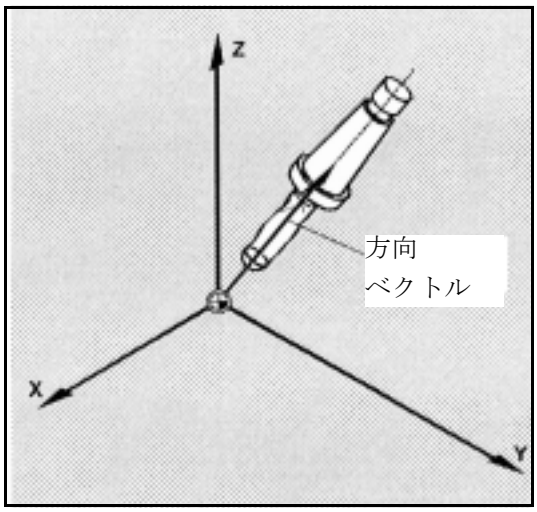


挿入手順の詳細については、3 次元工具径補正 (W5) を参照してください。

8.5 ツールの向き調整

ツールの向き調整は、空間のツールのジオメトリ心合わせをいう用語です。

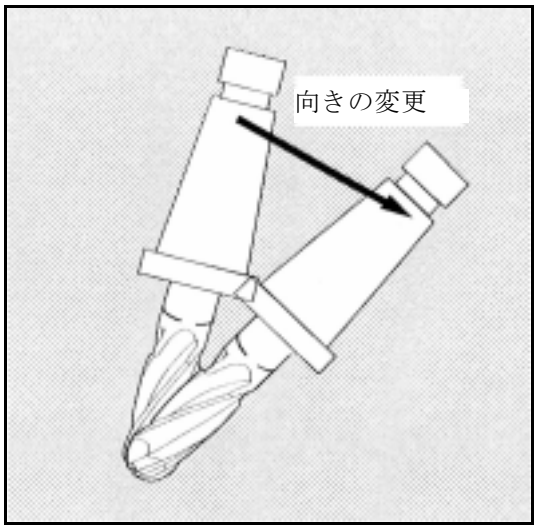
5 軸マシンツールでは、ツールの向き調整はプログラムコマンドを使用して制御されます。



ツール向き調整のプログラミング

ツール向き調整における変更は、以下を使用してプログラミングできます：

- 回転軸の直接プログラミング
- Euler 角度または RPY 角度
- 方向ベクトル
- LEAD/TILT （正面フライス加工）



基準座標系は、マシン座標系 (ORIMKS) または現在のワーク座標系 (ORIWKS) のいずれかとなります。

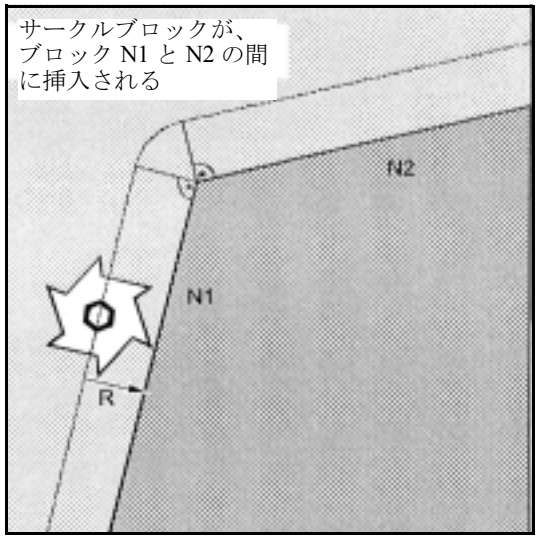
向き調整の変更は、以下によって制御されます：

ORIC	向きとパス動作が平行
ORID	向きとパス動作が連続
OSOF	向きが平滑でない
OSC	向き一定
OSS	ブロックの開始時のみ向き平滑
OSSE	ブロックの開始時と終了時に向き平滑
ORIS	有効な向き平滑化を使用する向き調整の速度（度／ mm ）；OSS および OSSE で有効

外側コーナーでの挙動

カッタの半径を使用する縁ブロックは、外側コーナーで必ず挿入されます。

プログラムコマンド ORIC および ORID を使用して、ブロック N1 と N2 の間にプログラムされた向き調整の変更が、挿入された円ブロックの開始の前に実行されるか、開始と同時に実行されるかを定義することができます。



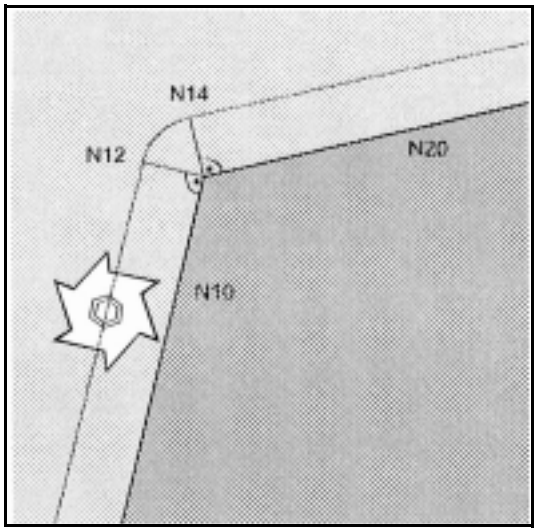
外側コーナーで向き調整変更が必要である場合、この変更は補間と同時に行うか、あるいは補間とは別にパス動作と一緒に行うことができます。

ORID を使用する場合、挿入されたブロックは最初はパス動作を伴わずに実行されます。コーナーを生成する円ブロックは、2 つの移動ブロックの後者の直前に挿入されます。

1 つの外側コーナーでいくつかの向き調整ブロックが挿入され、かつ ORIC が選択された場合、向き調整変更の値に基づいてそれぞれの挿入されたブロック間で円動作は分割されます。

ORIC のプログラミング例

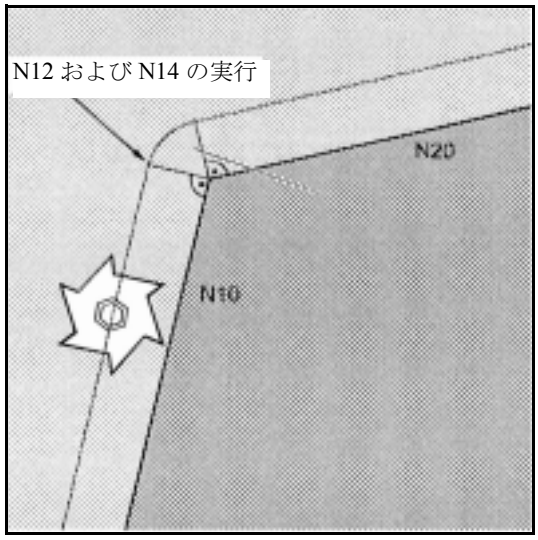
向き調整変更を使用する 2 つまたはそれ以上のブロック（例えば、A2= B2= C2= ）が、移動ブロック N10 および N20 の間にプログラミングされ、かつ ORIC が有効である場合、挿入されたブロックは角度変更の値にしたがってこれらの中間ブロック間で分割されます。



ORIC	
N8 A2=... B2=... C2=...	
N10 X... Y... Z...	
N12 C2=... B2=...N14 C2=... B2=...	外側コーナーで挿入された円ブロックは、向き調整の変更にしたがって N12 から N14 の間で分割される。円動作および向き調整変更は、平行して実行される。
N20 X =...Y=... Z=... G1 F200	

ORID のプログラミング例

ORID が有効である場合、2 つの移動ブロックの間の全てのブロックが最初の移動ブロックの終了時に実行されます。一定向き調整を使用する円ブロックは、第 2 の移動ブロックの直前で実行されます。



ORID	
N8 A2=... B2=... C2=...	
N10 X... Y... Z...	
N12 A2=... B2=... C2=...	ブロック N12 および N14 は、N10 の終了時に実行される。その結果。現在の向き調整を使用する円ブロックが移動される。
N14 M20	補助機能など .
N20 X... Y... Z...	

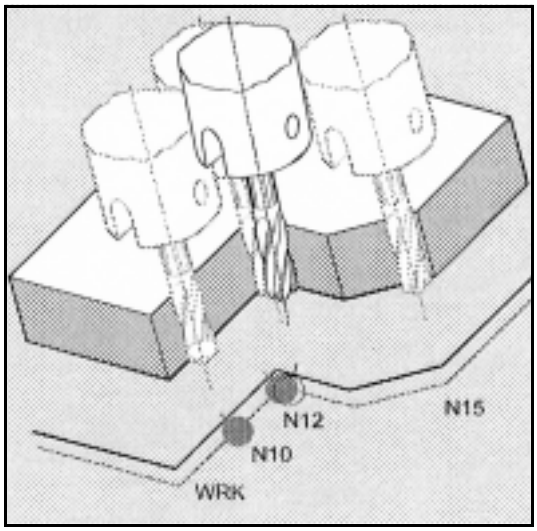
外側コーナーの最初の移動ブロックで有効になっているプログラムコマンドが、向き調整変更のタイプを確定します。

向き調整変更を使用しない場合

向き調整がブロック境界で変更されない場合、ツールの断面は輪郭の両方に接触する円となります。

プログラミング例

内側コーナーでの向き調整の変更



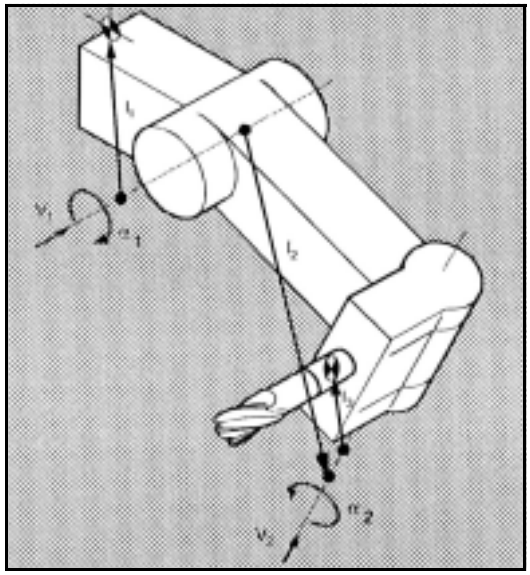
ORIC
N10 X ...Y... Z... G1 F500
N12 X ...Y... Z... A2=... B2=..., C2=...
N15 X Y Z A2 B2 C2

8.6 ツールホルダキネマティックス

最大 2 つの回転軸を使用するツールホルダキネマティックスは、17 のシステム変数 $TC_CARR1[m] から $TC_CARR17[m] を使用してプログラミングされます。

プログラムホルダの説明：

- 最初の回転軸とツールホルダ基準点の間のベクトル的距離 l_1 、最初の回転軸と 2 番目の回転軸の間のベクトル的距離 l_2 、2 番目の回転軸とツール基準点の間のベクトル的距離 l_3
- 両回転軸の基準ベクトル V_1, V_2
- 両軸の回りの回転角度 α_1, α_2。回転角度は、回転軸ベクトルの方向から見た場合時計方向を正としてカウントされます。



	x 構成要素	y 構成要素	z 構成要素
l_1	$TC_CARR1[m]	$TC_CARR2[m]	$TC_CARR3[m]
l_2	$TC_CARR4[m]	$TC_CARR5[m]	$TC_CARR6[m]
v_1	$TC_CARR7[m]	$TC_CARR8[m]	$TC_CARR9[m]
v_2	$TC_CARR10[m]	$TC_CARR11[m]	$TC_CARR12[m]
α_1 α_2	回転角度 = $TC_CARR13[m] 回転角度 = $TC_CARR14[m]		
l_3	$TC_CARR15[m]	$TC_CARR16[m]	$TC_CARR17[m]

追加説明

プログラムするそれぞれのツールホルダの番号は、"m" で指定されます。

軸上の距離ベクトルのスタート／エンドポイントは、自由に選択することができます。2 つの軸の回りの回転角度は、ツールホルダの初期ステータス（0°）で定義されます。このように、ツールホルダのキネマティックスの記述は、どのような番号に対してでも明確に記述されます。

（注）2 つの回転軸が交差する場合、その 2 つの軸の間の距離は指定する必要はありません。回転軸を全く持たない、あるいは 1 つだけ持つツールホルダは、回転軸の両方または片方の方向ベクトルをゼロに設定することによって記述することができます。回転軸を持たないツールホルダの場合、距離ベクトルはマシン平面の変更によってその構成要素が影響を受けない追加ツールオフセットとして作用します（G17 および G19）。

ツールホルダデータの削除

すべてのツールホルダデータセットのデータは、$TC_CARR1[0] = 0 によって削除されます。

ツールホルダデータの変更

記述された値はいずれも、パートプログラムに新規の値を割り当てることによって変更することができます。

ツールホルダデータの読み出し

記述された値はいずれも、その値をパートプログラムの変数に割り当てることによって変更することができます。

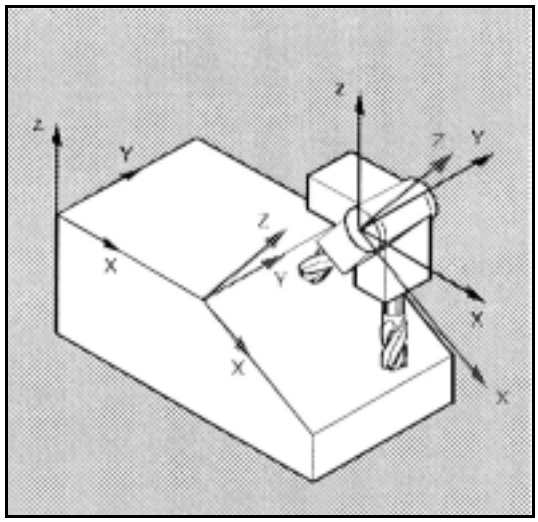
補足条件

次の場合、ツールホルダは、空間の全ての方向にツールを向き調整することができます。

- 2 つの回転軸がある
- その回転軸が相互に垂直の位置関係にある
- ツールの長さ軸が 2 番目の回転軸と垂直である

プログラミング例

次の例に使用されるツールホルダは、Y 軸回りの回転によって完全に描くことができます。



N10 $TC_CARR8[1]=1	ツールホルダ 1 の最初の回転軸の Y 要素の定義
N20 $TC_DP1[1,1]=120	エンドミルの定義¥
N30 $TC_DP3[1,1]=20	長さ 20 mm を使用する¥
N40 $TC_DP6[1,1]=5	かつ、半径 5 mm を使用する¥
N50 ROT Y37	y 軸回りを 37° 回転するフレーム定義
N60 X0 Y0 Z0 F10000	初期位置にアプローチ
N70 G42 CUT2DF TCOFR TCARR=1 T1 D1 X10	半径補正、回転したフレームでのツールの長さ補正を設定する、ツールホルダ 1, ツール 1 を選択する
N80 X40	37° 回転の条件下でマシニングを実行する
N90 Y40	
N100 X0	
N110 Y0	
N120 M30.9	

９ 軌跡移動における動作

9.1 タンジェンシャル制御 TANG, TANGON, TANGOF

プログラミング

TANG (FAxisF,LAxis1,LAxis2,Coupling,CS)

TANGON (FAxis,Angle)

TANGOF (FAxis)

TLIFT (FAxis)

コマンドの説明

TANG	タンジェンシャルフォローアップの定義用の準備命令
TANGON	追従軸とオフセット角度を指定するタンジェンシャル制御を起動する
TANGOF	追従軸を指定するタンジェンシャル制御を停止する
TLIFT	輪郭コーナーで中間ブロックを挿入する

パラメータの説明

FAxis	追従軸 : 追加のタンジェンシャル追従回転軸
LAxis1, LAxis2	リーディング軸 : 追従軸のタンジェントを決定するパス軸
Coupling	カップリング係数 : タンジェントの角度変更と追従軸の関係。パラメータ（オプション）; デフォルト : 1
CS	座標系の識別子 "B" = 基本座標系 ; "W" = ワーク座標系 パラメータ（オプション）; デフォルト "B"
Angle	追従軸のオフセット角度

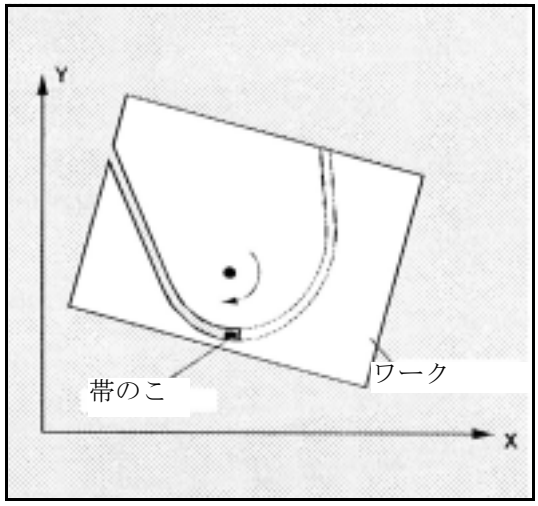
機能

回転軸 (= 追従軸) は、2 つのリーディング軸のプログラムされたパスを追従します。追従軸は、パスタンジェントを基準に定義されたオフセット角度で位置決めされます。

アプリケーション

タンジェンシャル制御は次のようなアプリケーションで使用できます :

- 帯のこ上のツール向き調整のフローアップ（図を参照のこと）
- ガラスまたは紙加工用のカッティングホイールの位置決め
- 5 軸溶接におけるワイヤのタンジェンシャルインフィード



動作

追従軸およびリーディング軸の定義

TANG を使用して、追従軸およびリーディング軸を定義することができます。

カップリング係数が、タンジェント軸と追従軸の角度変更の関係を指定します。この値は、通常の場合、1 となります (デフォルト)。

基本座標系 "B" (デフォルト) またはワーク座標系 "W" で、フォローアップを行うことができます。

例 :

TANG(C,X,Y,1,"B")

意味 :

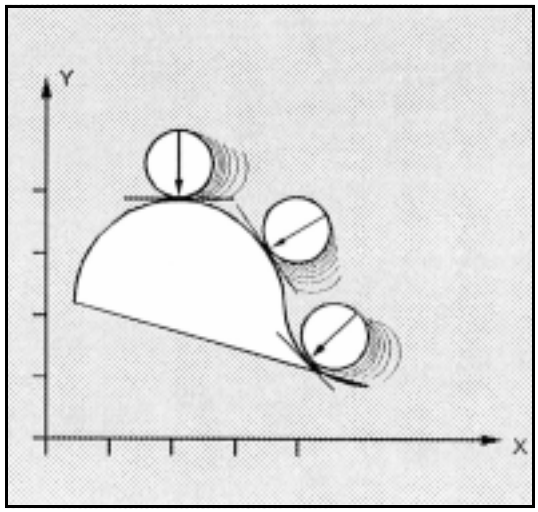
回転軸 C は、ジオメトリ軸 X と Y を追従します。

タンジェンシャル制御の起動／停止：

TANGON, TANGOF

タンジェンシャル制御は、追従軸ならびに追従軸の目標オフセット角度を指定する TANGON を使用してコールされます：

TANGON(C,90)



意味：

C 軸は追従軸です。パス軸がどのような動作をしても、C 軸はパスタンジェントに対して の 90° の位置になるように回転されます。

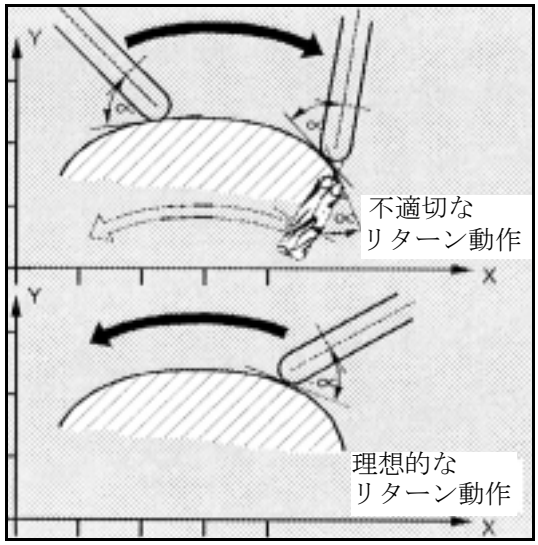
追従軸が指定されて、タンジェンシャル制御が停止されます：

TANGOF(C)

作業エリア制限による角度制限

前後に往復するパス動作では、タンジェントはパス上の折り返し点で 180° ジャンプし、それにしたがって追従軸の向きが変わります。

このような挙動は一般に望ましくありません：リターン動作はアプローチ動作と同じ負のオフセット角度で移動しなければなりません。



これは、追従軸 (G25, G26) の作業エリアを制限すると実行することができます。作業エリア制限は、パスリバーサル (WALIMON) と同時にアクティブになる必要があります。

オフセット角度が作業エリア制限の外側にある場合、負のオフセット角度を使用して許容可能な作業エリアに戻すようにしてください。

輪郭コーナーでの中間ブロックの挿入 , TLIFT

輪郭のコーナーでタンジェントが変化すると、それにしたがって追従軸の指令位置が変化します。

追従軸は通常最大限の速度を出してこのステップを補正しようとしますが、それによって要求タンジェンシャル位置から輪郭上のコーナーから先の不特定の位置へずれを生じてしまいます。そのようなずれは技術上の理由から許容されない場合、命令 TILIFT を使用して、制御装置をコーナーで停止させ、追従軸を自動的に生成された中間ブロックの新規のタンジェント方向へ向きを変更させます。追従軸は最大許容速度で回転されます。

TLIFT(...) 命令は、TANG(...) を使用して軸割当てが行われた直後にプログラムしてください。

例 :

TANG(C,X,Y...)

TLIFT(C)

TLIFT の終了

TLIFT を終了する場合、後で TLIFT(...) を挿入せずに軸割当て TANG(...) を繰り返してください。

中間ブロックが自動的に挿入される角度変更制限は、マシンデータ $MA_EPS_TLIFT_TANG_STEP を介して定義されます。

追加説明

変換に与える影響

追従回転軸の位置は、変換の入力値となります。

追従軸の関数位置決め

リーディング軸を追従している軸が関数にて位置決めされている場合、その位置はプログラムオフセット角度に追加されます。

全てのパス定義を行うことができます：パスおよび軸動作の位置決め

カップリングステータス

次のシステム変数を使用して、NC プログラムにカップリングのステータスを問い合わせることができます：

$AA_COUP_ACT[Axis]

0　　カップリングはアクティブでない

1,2,3　タンジェンシャルフォローアップは有効

9.2 カップリング動作 TRAILON, TRAILOF

プログラミング

TRAILON(FAxis,LAxis,Coupling)

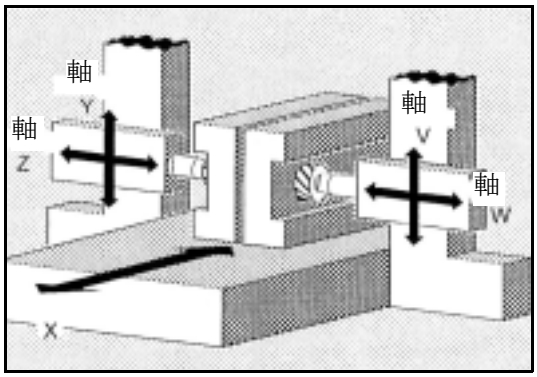
TRAILOF(FAxis,LAxis,Axis2)

コマンドおよびパラメータの説明

TRAILON	カップリング軸の起動および定義 ; モーダル
TRAILOF	カップリング軸の停止
FAxis	従軸の軸名
LAxis	リーディング軸の軸名
Coupling	カップリング係数 = 従軸の距離／リーディング軸 の距離 デフォルト = 1

機能

定義されたリーディング軸が移動されると、その軸に割当てられた従軸 (= フォローイング軸) がリーディング軸が記述した距離を移動します（カップリング係数が考慮されます)。また、リーディング軸および追従軸がカップリングされた軸となります。



アプリケーション

- シミュレートされた軸による軸の変換。
  リーディング軸はシミュレートされた軸となり、従軸は実際の軸となります。
  実際の軸は、カップリング係数を考慮して移動することができます。
- 2 つの軸を組合せて使用する両面加工：
  1 番目のリーディング軸 Y, 従軸 V
  2 番目のリーディング軸 Z, 従軸 W

動作

カップリング軸のグループの定義 , TRAILON

カップリング軸は、モーダル言語コマンド TRAILON を使用して同時に、定義され、起動されます。

TRAILON(V,Y)

V = 従軸 , Y = リーディング軸

同時に起動することができるカップリング軸数は、マシン上で可能な軸の組合せによって制限されます。

カップリング動作は、必ず基本座標系 (BCS) で行われます。

カップリング軸タイプ

カップリング軸グループは、直線軸と回転軸のどのような組合せでも構成することができます。シミュレーション軸も、リーディング軸として定義することができます。

カップリング動作軸

最高 2 つのリーディング軸を従軸に同時に割り当てることができます。カップリング軸をさまざまに組合せて割り当てることができます。

すべての使用可能なモーションコマンド (G0, G1, G2, G3, etc.) を使用して、従軸をプログラムすることができます。

個別に定義したパスの他に、従軸はそのリーディング軸から生じた距離を移動します（カップリング係数が考慮されます)。

従軸はまた、他の従軸に対してリーディング軸として振る舞うことができます。このようにして、カップリング軸のさまざまな組合せをセットアップすることができます。

カップリング係数

カップリング係数は、従軸およびリーディング軸のパスの理想比を指定します。

$$\text{カップリング係数} = \frac{\text{従軸のパス}}{\text{リーディング軸のパス}}$$

カップリング係数がプログラム内に定義されていない場合、
デフォルトとしてカップリング係数 1 が自動的に割当てられます。

係数は、小数として入力されます (タイプ REAL)。負値を入力すると、リーディング軸および従軸上に反対方向の移動動作が生じます。

カップリング軸の終了

次の言語コマンドは、リーディング軸とのカップリングを終了します :

TRAILOF(V,Y)

V = 従軸 **, Y** = リーディング軸

2 つのパラメータを持つ TRAILOF は、ただ 1 つのリーディング軸へのカップリングを終了します。

1 つの従軸が **2** つのリーディング軸に割当てられている
（例えば、**V=** 従軸 そして **X,Y=** リーディング軸）場合、**3** つのパラメータを使用して **TRAILOF** をコールし、カップリングを終了することができます :

TRAILOF(V,X,Y)

追加説明

加速および速度

組合わされた軸の加速および速度制限は、組合された軸ペアの中の " 最も低い軸 " によって決定されます。

カップリングステータス

次のシステム変数を使用して、NC プログラムにおけるカップリングのステータスを問い合わせることができます :

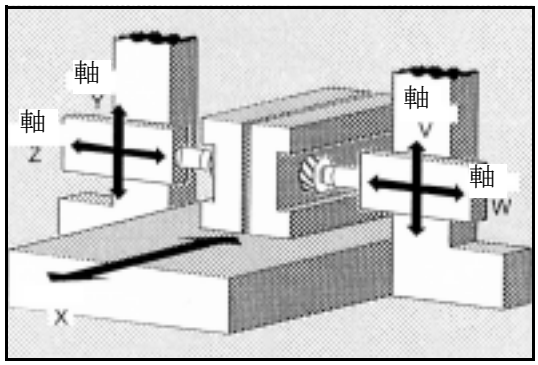
$AA_COUP_ACT[axis]

0　カップリングは有効でない

8　カップリングされた動作は有効

プログラミング例

図に示した軸構成を使用すると、ワークを両面加工することができます。それには、カップリングされた軸の組合せを 2 つ作成する必要があります。



...	
N100 TRAILON(V,Y)	有効な第 1 組の組合わされた軸
N110 TRAILON(W,Z,-1)	有効な第 2 組の組合わされた軸 , カップリング係数 負 : 従軸はリーディング軸と反対方向に移動する
N120 G0 Z10	軸方向が反対の Z 軸と W 軸のインフィード
N130 G0 Y20	軸方向が同一の Y 軸および V 軸のインフィード

...	
N200 G1 Y22 V25 F200	従軸 "V" の依存動作と非依存動作を重ね合わせる
...	
TRAILOF(V,Y)	第 1 のカップリング軸の終了
TRAILOF(W,Z)	第 2 のカップリング軸の終了

9.3 カーブテーブル , CTABDEF, CTABEND, CTAB, CTABINV

プログラミング

カーブテーブルは、パートプログラムに定義されます。

CTABDEF(FAxis,LAxis,n,applim)	カーブテーブル開始の定義
CTABEND()	カーブテーブル終了の定義
CTABDEL(n)	カーブテーブルの削除
R10=CTAB(LW,n,grad,FAxis,LAxis)	リーディング値用の追従値
R10=CTABINV(FW,aproxLW,n,grad,FAxis, LAxis)	追従値を基準にしたリーディング値

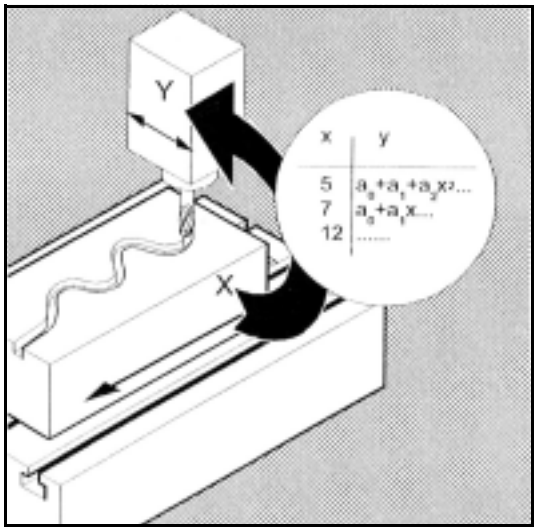
リーディング値および追従値の詳細については、本セクションの " 軸リーディング値カップリング " および " パスリーディング値カップリング " のセクションを参照してください。

説明

FAxis	追従軸 : カーブテーブルを介してプログラミングされた軸
LAxis	リーディング軸 リーディング値を使用してプログラミングされた軸
n	カーブテーブルの数
applim	テーブル周期性の識別子 : 0 テーブルは周期性でない 1 テーブルは周期性である
LW	リーディング値 その値を基準に追従値が計算されるリーディング軸の位置値
grad	勾配パラメータのパラメータ名
FW	追従値 その値を基準にリーディング値が計算される追従軸の位置値
aproxLW	追従値に対して確定される特定のリーディング値がない場合のリーディング値 用の近似解
FAxis,LAxis	追従値および／またはリーディング軸のオプション仕様

機能

カーブテーブルを使用して、2 軸の位置関係および速度関係をプログラムすることができます。



例 : メカニカルカムプレートの置換

カーブテーブルは、リーディング値と追従地の機能的関係を生成することによって、軸リーディング値カップリングの基礎を構築します :

制御装置は、リーディング軸と追従軸の相対的な位置をもとにしてカムプレートに対する多項式を計算します。

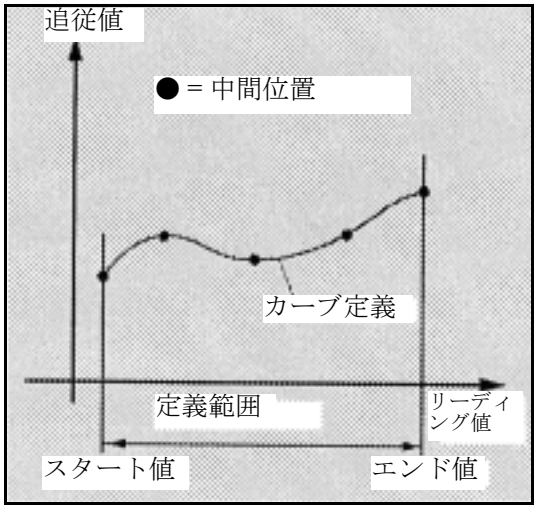
追加説明

カーブテーブルを作成する場合、マシンデータを設定してメモリスペースを予約する必要があります。

カーブテーブルの定義

CTABDEF, CTABEND

カーブテーブルは、パートプログラムあるいはパートプログラムの 1 セクション（開始を CTABDEF で、終了を CTABEND で囲まれている）を表します。



このパートプログラムセクション内では、ユニークな追従軸位置は移動ステートメントによってリーディング軸の個々の位置に割り当てられ、3 次までの多項式の形でカーブ定義を計算する際に中間位置として使用されます。

カーブテーブルの定義の開始用のスターティング値は、カーブテーブル内に指定された最初の関連軸位置（第 1 移動ステートメント）となります。カーブテーブルの定義のエンド値は、最後の移動コマンドによって確定されます。

カーブテーブルの定義では、全ての NC 言語が使用できます。

追加説明

以下のことは許容されません :

- 前処理の停止
- 工具径補正
- リーディング軸動作への割込み (例えば、変換の変更時における)
- 追従軸のみの移動ステートメント
- リーディング軸の反転、すなわちリーディング軸の位置は常にユニークでなければならない
- カーブテーブル定義内で行われた全てのモーダル指令は，テーブル定義が終了した時点で無効となります。したがって、テーブル定義が行われたパートプログラムは，テーブル定義の前と後ろに，同じ水準で置かれます。

R パラメータ割当てがリセットされます。

例 :

```
...
R10=5 R11=20
...
CTABDEF
G1 X=10 Y=20 F1000
R10=R11+5 ;R10=25
X=R10
CTABEND
... ;R10=5
```

カーブテーブルの繰り返し使用

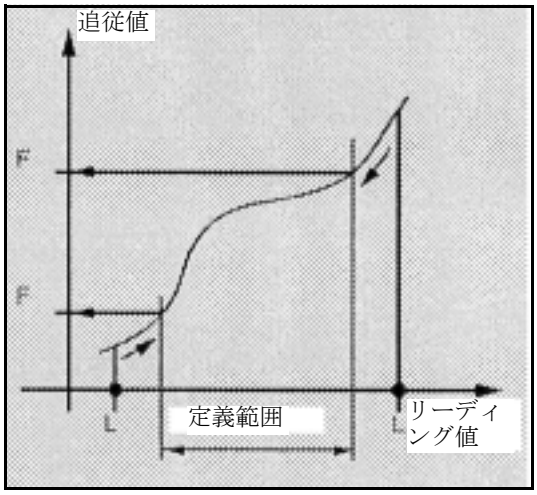
カーブテーブルを通じて計算されたリーディング軸と追従軸の機能関係は、パートプログラムのエンド外のテーブル番号の下で、パワーオフ中に、保持されます。

生成されたカーブテーブルは、カーブテーブルの生成にどのような軸が使用されていたとしても、リーディング軸と追従軸のどんな軸組み合わせにも適用することができます。

カーブテーブルエッジでの挙動

非周期性カーブテーブル

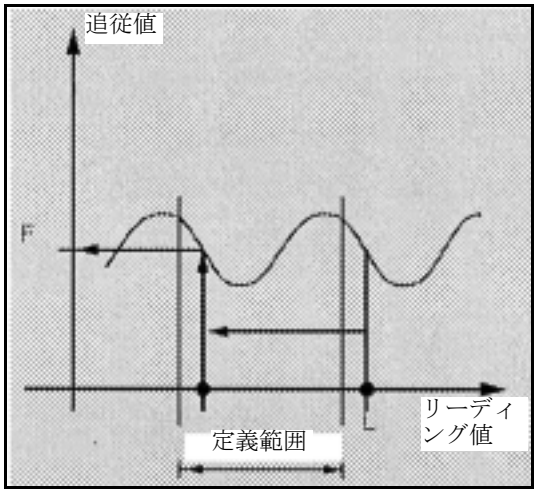
リーディング値が定義範囲外にある場合、追従値出力は上限または下限のいずれかになります。



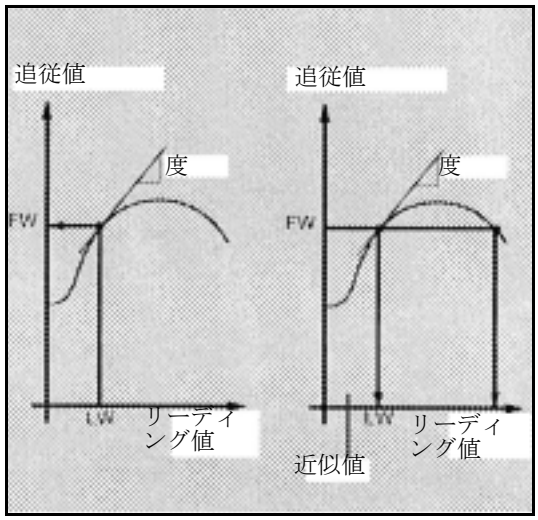
周期性カーブテーブル

リーディング値が定義範囲外にある場合。

そのリーディング値は定義範囲の 周期性 を評価され、対応する追従値が出力されます。



テーブル位置の読みこみ , CTAB, CTABINV



CTAB を使用すると、パートプログラムまたはシンクロナイズドアクションから直接リーディング値用の追従値を読み込むことができます (セクション 10)。

CTABINV を使用すると、追従値用のリーディング値を読み込むことができます。
この割当ては必ずしもユニークになる必要はありません。したがって、CTABINV は期待されたリーディング値用の近似値 (aproxLW) を要求します。
CTABINV は、近似値に最も近いリーディング値を返します。この近似値は、その前の補間サイクルから得られたリーディング値になる場合もあります。

どちらの機能も、勾配パラメータ (grad) を基準に正確な位置でテーブル機能の勾配を出力します。このようにして、対応する位置においてリーディング軸や追従軸の速度を計算することができます。

追加説明

CTAB/CTABINV 用のリーディング軸または追従軸のオプション仕様は、リーディング軸や追従軸が異なる長さの単位で構成されている場合、特に重要となります。

カーブテーブルの削除 , CTABDEL

CTABDEL によって、カーブテーブルを削除することができます。
カップリング中に有効になっているカーブテーブルは削除することはできません。

カーブテーブルの上書き

カーブテーブルは、その番号が他のテーブル定義に使用されるとすぐに上書きされます。有効なテーブルは上書きできません。

追加説明

カーブテーブルの上書きの際にはアラームは出力されません。

追加説明

システム変数 $P_CTABDEF を使用すると、パートプログラムの内部からカーブテーブルが有効であるかどうか問い合わせることができます。

パートプログラムセクションは、ステートメントを除外した後にカーブテーブル定義として使用することができるので、再び実際のパートプログラムとして使用することができます。

プログラミング例

プログラムセクションは変更せずにカーブテーブルの定義に使用することができます。前処理停止 用のコマンド STOPRE は継続されるので、プログラムセクションがテーブル定義に使用されていず、かつ CTABDEF と CTABEND が削除された直後に、再び有効になります：

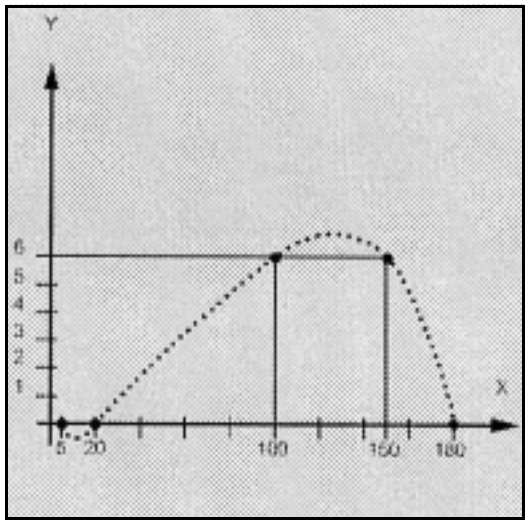
```
CTABDEF(Y,X,1,1)
...
...
IF NOT ($P_CTABDEF)
STOPRE
ENDIF
...
...
CTABEND
```

カーブテーブルおよびさまざまなオペレーティングステータス

ブロック検索が有効である間は、カーブテーブルの計算は使用できません。ターゲットブロックがカーブテーブルの定義内にある場合、CTABEND に達するとアラームが出力されます。

プログラミング例 1

カーブテーブルの定義



N100 CTABDEF(Y,X,3,0)	番号 3 をもつ周期性のないカーブテーブルの定義の開始
N110 X0 Y0	1. 移動ステートメントがスターティング値および第 1 補間点を定義する : リーディング値 : 5; 追従値 : 0
N120 X20 Y0	2. 中間点 : リーディング値 : 0...20; 追従値 : スターティング値 ...0
N130 X100 Y6	3. 中間点 : リーディング値 : 20...100; 追従値 : 0 ... 6
N140 X150 Y6	4. 中間点 : リーディング値 : 100...150; 追従値 : 6 ... 6
N150 X180 Y0	5. 中間点 : リーディング値 : 150...180; 追従値 : 6 ... 0
N200 CTABEND	定義の終了 ; カーブテーブルは、最高 3 次までの多項式としてその内部で表現され生成されます。カーブ定義の計算はモーダル選択された補間タイプ（円補間、直線補間、スプライン補間）によって異なります ; 定義の開始以前のパートプログラムステータスは、復活します。

プログラミング例 2

番号 2 、リーディング値範囲 0 から 360 、追従軸
モーション 0 から 45 そして 0 に戻るとする、周期性
カーブテーブルの定義 :

N10 DEF REAL DEPPOS;	
N20 DEF REAL GRADIENT;	
N30 CTABDEF(Y,X,2,1)	定義の開始
N40 G1 X=0 Y=0	
N50 POLY	
N60 PO[X]=(45.0)	
N70 PO[X]=(90.0) PO[Y]=(45.0,135.0,-90)	
N80 PO[X]=(270.0)	
N90 PO[X]=(315.0) PO[Y]=(0.0,-135.0,90)	
N100 PO[X]=(360.0)	
N110 CTABEND	定義の終了

Y を X にカップリングしてカーブをテストする :
N120 G1 F1000 X0
N130 LEADON(Y,X,2)
N140 X360
N150 X0
N160 LEADOF(Y,X)

テーブル機能からリーディング値 75.0 を読み取る :
N170 DEPPOS=CTAB(75.0,2,GRA s DIENT)

リーディング軸および追従軸の位置決め :
N180 G0 X75 Y=DEPPOS

カップリングの起動後、追従軸の同期性は要求さ れない :
N200 G1 X110 F1000
N210 LEADOF(Y,X)
N220 M30

9.4 軸リーディング値カップリング , LEADON, LEADOF

プログラミング

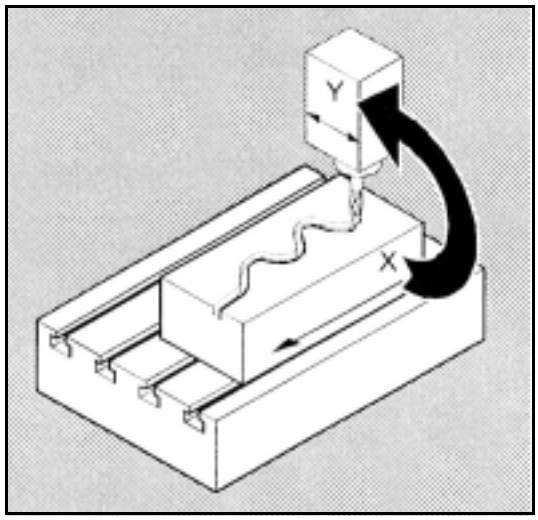
LEADON(FAxis,LAxis,n)

LEADOF(FAxis,LAxis,n)

説明

LEADON	リーディング値カップリングの起動
LEADOF	リーディング値カップリングの停止
FAxis	追従軸 :
LAxis	リーディング軸
n	カーブテーブル数

機能



軸リーディング値カップリングを使用すると、リーディング値および追従軸は同期して動きます。カーブテーブルまたは多項式の演算結果を介して、追従軸の位置をリーディング軸の位置に割り当てることができます（必要があれば、シミュレートされます)。

リーディング軸とは、カーブテーブル用の入力値を与える軸のことです。追従軸とは、カーブテーブルによって計算された位置を指令とする軸のことです。

リーディング値カップリングは、パートプログラムからでも動作中には同期化されたクションからでも、起動および停止することができます。

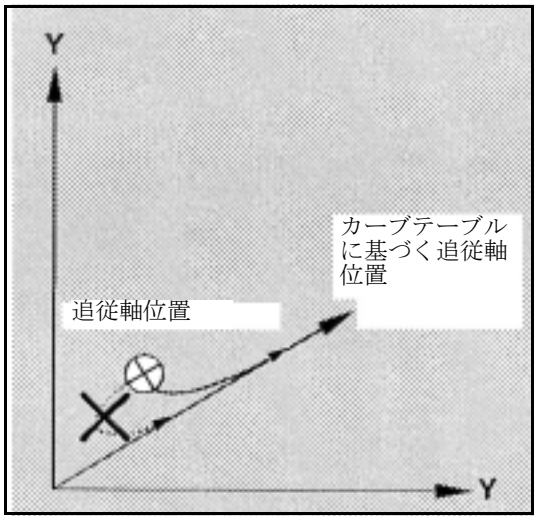
リーディング値カップリングは、必ず基本座標系において作用します。

カーブテーブル生成の詳細については、本セクションの " カーブテーブル " のセクションを参照してください。リーディング値カップリングの詳細については、/FB/, M3, カップリング動作とリーディング値カップリングを参照してください。

動作

リーディング値 カップリングには、リーディング軸と追従軸の同期化が必要です。この同期化をするには、リーディング値カップリングが起動していて、追従軸がカーブテーブルに基づき計算されたカーブ定義の許容範囲内になければなりません。

追従軸の位置の許容範囲は、マシンデータ 37200COUPLE_POS_TOL_COARSE を介して定義されます。



リーディング値カップリングが起動されているのに追従軸がまだ正しい位置にならない場合、追従軸用に計算された位置指令値がほぼ実際の追従軸位置になるとすぐに、同期化ランは自動的に初期化されます。同期化手順の間は、追従軸は追従軸の指令速度によって定義された方向に移動されます（マスタ主軸と CTAB に基づいて計算されます)。

追加説明

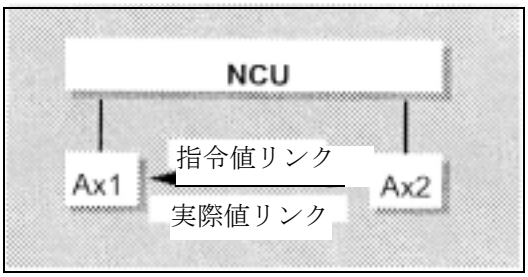
リーディング値カップリングが起動されると、計算された追従軸位置が現在の追従軸位置から離れた場合には、同期化を設定することはできません。

実際値と指令値のカップリング

以下をリーディング値として使用することができます、すなわち追従軸の位置計算用の出力値として使用することができます：

- リーディング軸位置の実際値 : 実際値カップリング
- リーディング軸位置の指令値 : 指令値カップリング

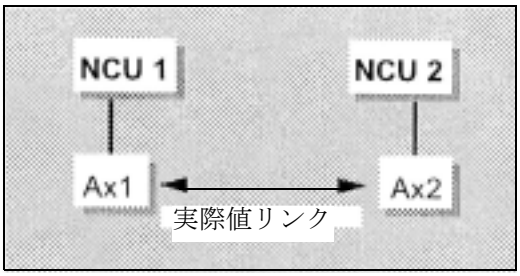
追加説明



指令値カップリングの方が、実際値カップリングよりリーディング軸と追従軸の同期がよりうまくできるので、デフォルト値によってセットされています。

指令値カップリングは、リーディング軸と追従軸が同じ NCU によって補間されていなければ使用できません。

外部リーディング軸を使用すると、実際値を介して追従軸だけをリーディング軸にカップリングすることができます。



実際値カップリング指令値カップリングの切替え

切替えは、セッティングデータ $SA_LEAD_TYPE を介してプログラミングすることができます。

追従軸が停止すると必ず実際値カップリングと指令値カップリングを切替える必要があります。切替え後の軸の停止中にのみ、再同期化することができます。

アプリケーション 例 :

マシンが大きく振動中に実際値を読み取ると必ず誤差が発生します。プレス移動にリーディング値カップリングを使用する場合、最大の振動が生じる作業ステップでは実際値カップリングからセットポイントカップリングに切替える必要がある場合もあります。

指令値カップリングを使用するリーディング値シミュレーション

マシンデータを使用して、サーボとリーディング軸の補間器の接続を断つことができます。これにより、実際にリーディング軸を移動させることなく指令値カップリング用の指令値を生成することができます。

指令値リンクから生成されたリーディング値は、次の変数から読み出して、例えばシンクロナイズドアクションで使用することができます :

- $AA_LEAD_P　　リーディング値位置
- $AA_LEAD_V　　リーディング値速度

追加説明

オプションとして、リーディング値は他の自己プログラミング方法を使用しても生成されます。そのような方法で生成されたリーディング値は変数

- $AA_LEAD_SP　　リーディング値位置
- -- $AA_LEAD_SV　　リーディング値速度

に書き込まれたり読み出されたりします。これらの変数を使用するには、まずそのための設定データ $SA_LEAD_TYPE = 2 をセットする必要があります。

カップリングのステータス

次のシステム変数 :

$AA_COUP_ACT[axis]

を使用して、NC プログラムにおけるカップリングのステータスを問い合わせることができます。

0　カップリングは有効でない

16　リーディング値カップリングは有効である

リーディング値カップリングの停止 , LEADOF

リーディング値カップリングを停止すると、追従軸はノーマルなコマンド軸に戻ります !

軸リーディング値カップリングおよびさまざまな操作ステータス

マシンデータでの設定によって、リーディング値カップリングは RESET を使用して停止されます。

プログラミング例

プレス機では、リーディング軸（スタンチョンシャフト）ならびにトランスファ軸と補助軸で構成されたトランスファシステムを機械的にカップリングする通常の方法は、エレクトロニックカップリングシステムに取り替わられつつあります。

ここでは、機械的なトランスファシステムがどのようにエレクトリックトランスファシステムに切替えられるかを実例を挙げて説明します。カップリングとカップリング解除のイベントは静的なシンクロナイズドアクションとして構築されます。

リーディング軸 LW （スタンチョンシャフト）から、トランスファ軸と補助軸はカーブテーブルを介して定義された追従軸として制御されます。

追従軸

X フィード軸または縦軸

YL クロージング軸または横軸

ZL ストローク軸

U ローラフィード , 補助軸

V ガイディングヘッド , 補助軸

W グリーシング , 補助軸

ステータス管理

スイッチングイベントおよびカップリングイベントは、リアルタイム変数によって管理されます :

$AC_MARKER[i]=n

この場合 :

i マーカ番号

n ステータス値

アクション

発生するアクションには、例えば次のようなシンクロナイズドアクションが含まれます :

- カップリングを起動する , LEADON(追従軸 , リーディング軸 , カーブテーブル番号)
- カップリングを停止する , LEADOF(追従軸 , リーディング軸)
- 実際値をセットする , PRESETON(軸、値)
- マーカをセットする , $AC_MARKER[i]= 値
- カップリングタイプ : 実際の / 仮想のリーディング値
- 軸位置にアプローチ、POS[axis]= 値

条件

評価用のブール演算子 AND を条件として使用して、迅速なデジタル入力、リアルタイム変数 $AC_MARKER および位置の比較をリンクすることができます

（注）次に挙げる例では、プログラムをさらに読みやすくするために、行変更、インデント、および太字が使用されています。制御装置に対しては、行番号以降は 1 行となります。

コメント

```
                ; 全ての静的なシンクロナイズドアクションを
                定義します
                ; **** リセットマーカ
N2              $AC_MARKER[0]=0 $AC_MARKER[1]=0
                $AC_MARKER[2]=0 $AC_MARKER[3]=0
                $AC_MARKER[4]=0 $AC_MARKER[5]=0
                $AC_MARKER[6]=0 $AC_MARKER[7]=0
                                                ; **** E1 0=>1 カップリングトランスファ ON
N10             IDS=1 EVERY ($A_IN[1]==1) AND
($A_IN[16]==1) AND ($AC_MARKER[0]==0)
DO              LEADON(X,LW,1) LEADON(YL,LW,2)
                LEADON(ZL,LW,3) $AC_MARKER[0]=1
                                                ;**** E1 0=>1 カップリング ローラフィード ON
N20             IDS=11 EVERY ($A_IN[1]==1) AND
($A_IN[5]==0) AND ($AC_MARKER[5]==0)
DO              LEADON(U,LW,4) PRESETON(U,0)
                $AC_MARKER[5]=1
                                                ; **** E1 0->1 カップリングガイドヘッド ON
N21             IDS=12 EVERY ($A_IN[1]==1) AND
($A_IN[5]==0) AND ($AC_MARKER[6]==0)
DO              LEADON(V,LW,4) PRESETON(V,0)
                $AC_MARKER[6]=1
                                                ; **** E1 0->1 カップリンググリーシング ON
N22             IDS=13 EVERY ($A_IN[1]==1) AND
($A_IN[5]==0) AND ($AC_MARKER[7]==0)
DO              LEADON(W,LW,4) PRESETON(W,0)
                $AC_MARKER[7]=1
                                                ; **** E2 0=>1 カップリング OFF
N30             IDS=3 EVERY ($A_IN[2]==1)
DO              LEADOF(X,LW) LEADOF(YL,LW)
                LEADOF(ZL,LW) LEADOF(U,LW)
LEADOF(V,LW) LEADOF(W,LW) $AC_MARKER[0]=0
$AC_MARKER[1]=0 $AC_MARKER[3]=0
$AC_MARKER[4]=0 $AC_MARKER[5]=0
$AC_MARKER[6]=0 $AC_MARKER[7]=0
....
N110            G04 F01
N120            M30
```

9.5 パスリーディング値カップリング , LEADONP, LEADOFP

プログラミング

LEADONP(LAxis,activation position,coupling)	パスリーディング値カップリングタイプ A を起動する
LEADONP(LAxis,ActivationPosition)	パスリーディング値カップリングタイプ B を起動する
LEADOFP	パスリーディング値カップリングを停止する

説明

LAxis	リーディング軸
Activation position	タイプ A の場合 : カップリングされたパスがフライ上で同期する場合のリーディング値。リーディング値が起動位置を通過した直後にカップリング は起動される。 タイプ B の場合 : その位置からカップリングが起動されるリーディング軸の位置
カップリング	カップリング係数 負の符号（カップリング係数が無視される場合）

機能

パスリーディング値カップリングを使用すると、全てのパス軸を

リーディング軸にカップリングすることができます (カーブテーブルを使用しない場合):

タイプ A: リーディング軸はパス軸ではない。

タイプ B: リーディング軸は、NC プログラムで使用されるパス軸である。

追加説明

RESET ならびにプログラムエンドで、パスリーディング値カップリングは停止されます。

カーブテーブル生成についての詳細は , 本セクションのサブセクション

" カーブテーブル " を参照してください

リーディング値 カップリングの詳細については、

/FB/, M3, カップリングされた軸およびリーディング値 カップリングを参照してください。

9.5.1 パスリーディング値カップリング タイプ A

動作

リーディング軸は、パス軸ではない外部軸となることができます。

定義された起動位置から、カップリングされたパス軸の送り速度はリーディング軸の送り速度に同期します。

プログラムされたパスに沿ったパス送り速度は、LEADONP によってパスリーディング値カップリングをコールするブロック内ではゼロとなります。

パス送り速度は定義された起動位置までリーディング軸送り速度に同期します。

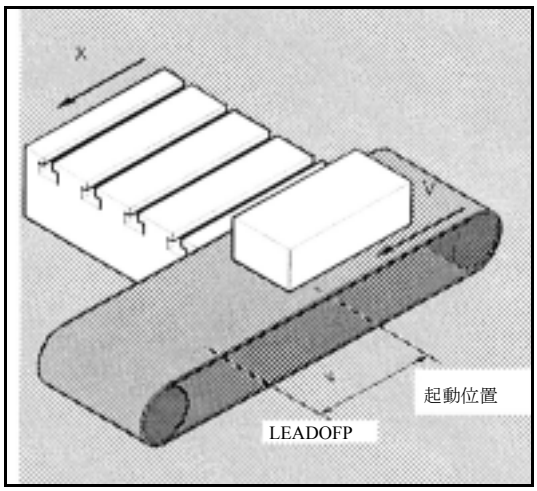
追加説明

制限事項

- リーディング軸の方向は反転することはできません。
- パスリーディング値カップリングには連続パスモード (G64) が起動されなければなりません。

プログラミング例

マシンが外部で制御されたコンベヤベルトからワークを受け取ります。この場合、コンベヤベルト V に平行なマシンの X 軸は、一時的にコンベヤベルトに同期されます。



...	
N100 G64 X0	連続パスモードのスイッチオン
N110 LEADONP(V,100,1)	パスリーディング値カップリングを起動する。V はリーディング軸である。X 軸は位置 V=100 で同期化される。カップリング係数 1. X は移動中に位置 V=100 にて同期化される。
N120 X150	ワーク移動は位置 X=150 で完了される。
N130 LEADOFP	リーディング値カップリングの停止 ...
N140 X160	... なめらかな接線方向移動 ...
N150 X... Y... Z...	... マシニング .
...	

9.5.2 パスリーディング値 カップリングタイプ B

動作

リーディング軸は、パス軸としてプログラムされる外部軸となることができます。NC プログラムが実行されると、全ての他のパス軸の送り速度がこのリーディング軸を基準に調節されます。

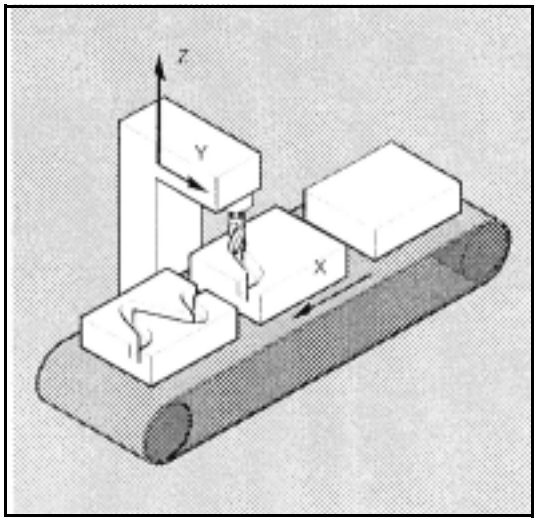
プログラムされたパスのパス送り速度は、LEADONP にてカップリングしているパスリーディング値をコールするブロック内ではゼロとなります。

追加説明

制限事項

- リーディング軸が移動すると、カップリングされたパス軸も移動することができます。
- リーディング軸 の方向は、反転することはできません。
- 連続パスモード (G64) は、パスリーディング値カップリングに対して起動する必要があります。

プログラミング例



連続運転する場合、標準 NC プログラムがマシンツール上で実行されますが、関連する軸の 1 つが NC によって制御されない、あるいはチャンネルの 1 つに問題がある、ということがしばしば発生します：

木工機上のマシニングステーションは、X 軸、Y 軸および Z 軸で NC プログラムを実行しなければなりません。X は、外部で制御されるトランスファ軸で、この軸に関しては実際値しかわかりません。パス軸 Y および Z の送り速度を X 軸を基準に調整するには、パスリーディング値カップリング用の起動コマンドがマシニングプログラムよりも先行する必要があります。

...	
N90 G64 X0	連続パスモードのスイッチオン
N100 LEADONP(X, XPOS)	X はリーディング軸。X=XPOS でカップリングをスタートする。
N110 X... Y... Z...	
...	

9.6 フィード特性 , FNORM, FLIN, FCUB, FPO

プログラミング

F... FNORM

F... FLIN

F... FCUB

F=FPO(...,...,...)

説明

FNORM	基本設定。送り速度値はブロックの移動パスの機能として指定され、その後モーダル値として有効になります。
FLIN	直線パス速度プロファイル : 送り速度値は、ブロック開始の速度値からブロックエンドの速度値まで移動パスを介して直線にアプローチされ、その後モーダル値として有効になります。
FCUB	３次パス速度プロファイル : ノンモーダルにプログラムされた F 値は、ブロックエンドポイントを基準にしたスプラインによって修正されます。スプラインは、前及び後の送り速度仕様で接線的に開始し終了します。 F アドレスがブロックから欠落している場合、プログラムされている最後の F 値が使用されます。
F=FPO...	多項式パス速度プロファイル : F アドレスは、多項式を介して現在の値からブロックエンドまで送り速度特性を定義します。その後、エンド値はモーダル値として有効になります。

機能

フィード特性の定義を柔軟に行うには、DIN 66205 に従った送り速度プログラミングを直線および３次特性によって拡張しておく必要があります。３次特性は直接設定するか、あるいは補間スプラインとしてプログラミングすることができます。

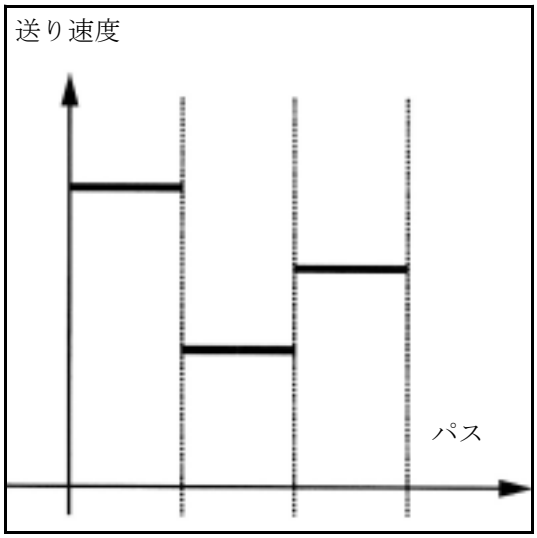
これらの特性を追加することにより、加工されるワークの曲率に従って連続したなめらかな速度特性をプログラムすることができます。

動作

FNORM

送り速度アドレス F は、パス送り速度を DIN 66025 を基準とする一定値として定義します。

これに関する詳細については、プログラミング編 基本説明書を参照してください。

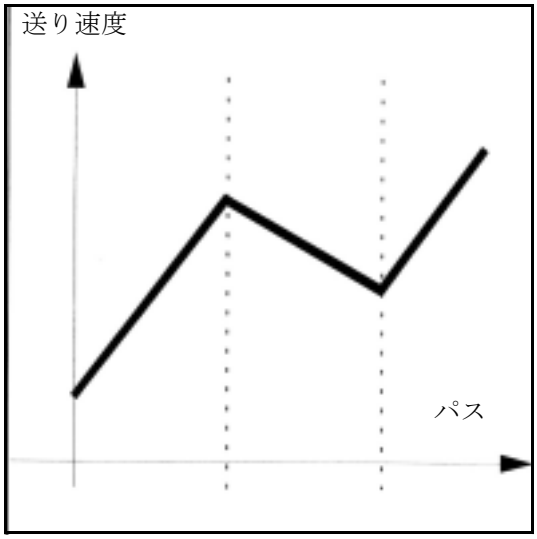


FLIN

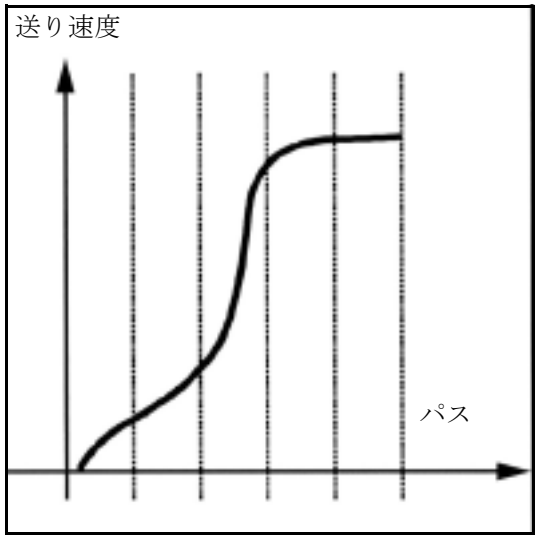
送り速度特性は、現在の値からプログラムされた F 値まで、そのブロックが終了するまで直線的にアプローチされます。

例 :

N30 F1400 FLIN X50



FCUB



３次特性に従って、送り速度は現在の値からプログラムされた F 値まで、そのブロックが終了するまでアプローチします。制御装置はスプラインを使用して、アクティブな FCUB を持つノンモーダルにプログラムされた送り速度値を全て修正します。

この場合、送り速度値はスプライン補間の計算用の補間点として作用します。

例 :

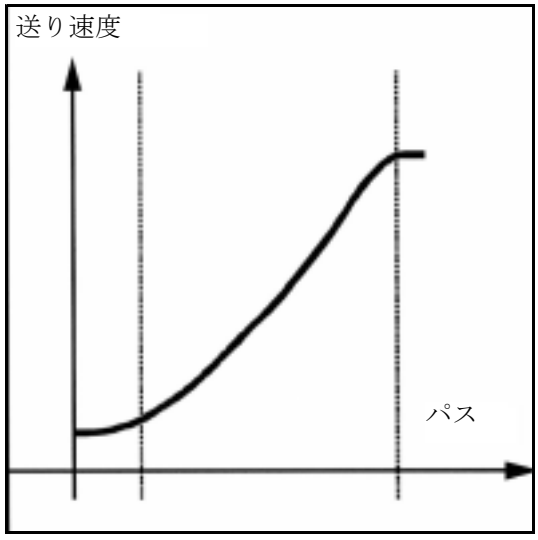
N50 F1400 FCUB X50

N60 F2000 X47

N70 F3800 X52

...

F=FPO(...,...,...)



送り速度特性は多項式を介して直接プログラムされます。多項式の係数は、多項式補間に使用されるのと同一の方法で指定されます。

例 :

F=FPO(endfeed, quadf, cubf)

endfeed, quadf および cubf は、以前に定義された変数です。

endfeed:	ブロック終了時の送り速度
quadf:	2 次方程式係数
cubf:	3 次方程式係数

アクティブな FCUB を使用すると、スプラインはブロックの開始および終了時に FPO を介して定義された特性に接線的にリンクされます。

補足条件

パス移動特性のプログラミング用の機能は、プログラムされた送り速度特性とは関係なく適用されます。

プログラムされた送り速度特性は常に絶対です（G90 または G91 とは無関係）。

追加説明

コンプレッサ

アクティブなコンプレッサ COMPON を使用する場合、いくつかのブロックが連結されてスプラインセグメントを構成する際に以下のものが適用されます：

FNORM:

グループ内の最終ブロックの F ワードがスプラインセグメントに適用されます。

FLIN:

グループ内の最終ブロックの F ワードがスプラインセグメントに適用されます。

プログラムされた F 値は、そのセグメントの終了時まで適用され、その後直線にアプローチされます。

FCUB:

生成された送り速度スプラインは、マシンデータにセットされた値を超えない分だけプログラムされたエンドポイントからはずれます。

$MC_COMPESS_VELO_TOL.

F=FPO(...,...,...)

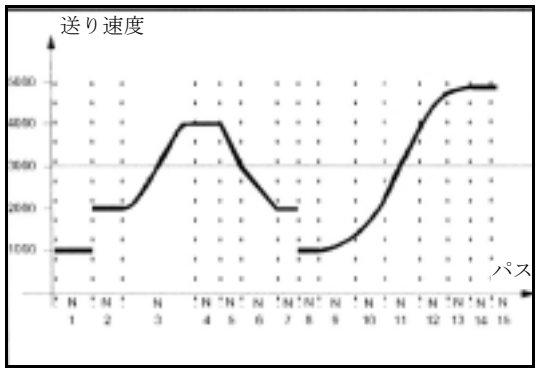
これらのブロックは圧縮されません。

カーブしたパスセクションの送り速度の最適化

送り速度多項式 F-FPO および送り速度スプライン FCUB は、必ず一定の切削速度 CFC で移動する必要があります。その結果、ジャークのない指令送り速度プロファイルを生成することができます。

プログラミング例

この例では、さまざまな送り速度プロファイルのプログラミングおよび図を示したいと思います。



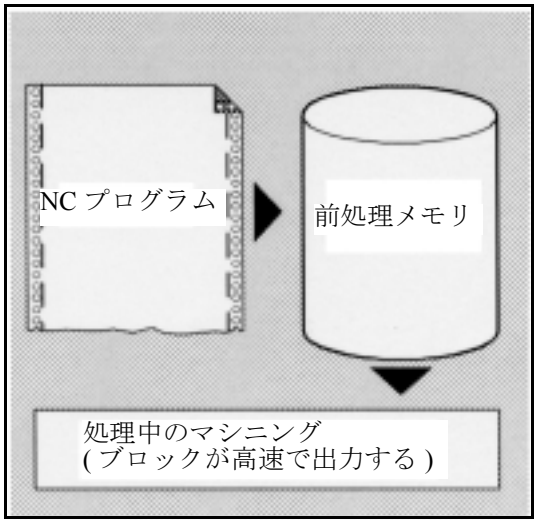
N1 F1000 FNORM G1 X8 G91 G64	一定の送り速度プロファイル、増分寸法記入
N2 F2000 X7	指令速度のステップ変更
N3 F=FPO(4000, 6000, -4000)	ブロックエンドで速度 4000 を持つ多項式を介する送り速度プロファイル
N4 X6	多項式速度 4000 はモーダル値として作用する
N5 F3000 FLIN X5	直線速度プロファイル
N6 F2000 X8	直線速度プロファイル
N7 X5	直線速度はモーダル値として作用する
N8 F1000 FNORM X5	加速速度の急速な変化を伴う一定速度プロファイル
N9 F1400 FCUB X8	その結果、全ての非モーダルにプログラムされた F 値はスプラインを介して修正される。
N10 F2200 X6	
N11 F3900 X7	
N12 F4600 X7	
N13 F4900 X5	スプラインプロファイルを停止する
N14 FNORM X5	
N15 X20	

9.7 前処理メモリを使用したプログラムラン ,STARTFIFO, STOPFIFO, STOPRE

コマンドの説明

STARTFIFO	高速処理セクションのエンド , 前処理メモリの充填
STOPFIFO	高速処理セクションのスタート
STOPRE	前処理の停止

機能



その拡張レベルに応じて、制御系は特定のいわゆる前処理メモリ容量を保有しています。このメモリ内に、準備されたブロックはプログラムが実行されるまで格納され、マシニングの進行に伴い高速ブロックシーケンスとして出力されます。

これらのシーケンスによってショートパスは高速で移動することができます。

利用可能な制御時間が充分に残っていると仮定して、前処理メモリは常に充填されています。

STARTFIFO は、前処理メモリがいっぱいになるまで、あるいは STOPFIFO または STOPRE が検出されるまでマシニングを停止します。

動作

処理セクションのマーク

前処理メモリ内にバッファされる高速処理セクションは、STARTFIFO と STOPFIFO を使用する開始時と終了時のそれぞれでマークされます。

例 :

N10 STARTFIFO

N20...

N100

N110 STOPFIFO

前処理メモリが充填される、あるいはコマンド STOPFIFO が検出されて初めて、これらのブロックの実行が開始されます。

制限事項

処理セクションがバッファリングできないオペレーション（例えば、原点復帰、測定機能など）を要求するコマンドを含んでいる場合、前処理メモリの充填が実行されないか、あるいは充填プロセスが中断されます。

前処理の中止

STOPRE がプログラムされると、以前に準備され格納されているブロックが全て完全に実行されるまで、後のブロックの処理は行われません。STOPRE 以前のブロックは、イグザクトストップ (G9) を使用して中止されます。

例 :

N10 ...

N30 MEAW=1 G1 F1000 X100 Y100 Z50

N40 STOPRE

制御系は、マシンのステータスデータ ($A...) へのアクセス中に内部前処理停止を初期化します。

例 :

```
R10 = $AA_IM[X]                ; X 軸の実際値を読み込む
```

（注）ツールオフセットまたはスプライン補間が有効な場合、STOPRE コマンドをプログラミングすると連続ブロックシーケンスの中断を引き起こすため、本コマンドのプログラミングは行わないでください。

9.8 輪郭上での再位置決め , REPOSA, REPOSL, REPOSQ, REPOSH

プログラミング

REPOSA RMI DISPR=... または REPOSA RMB または REPOSA RME

REPOSL RMI DISPR=... または REPOSL RMB または REPOSL RME

REPOSQ RMI DISPR=... DISR=... または REPOSQ RMB DISR=... または REPOSQ RME DISR=... または REPOSQA
DISR=...

REPOSH RMI DISPR=... DISR=... または REPOSH RMB DISR=... または REPOSH RME DISR=... または
REPOSHA DISR=...

コマンドの説明

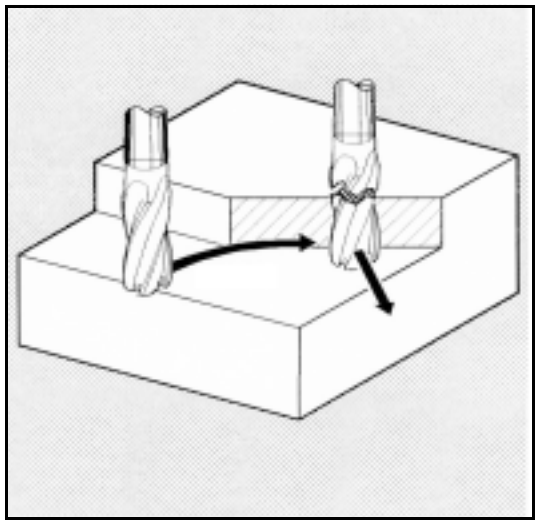
アプローチパス

REPOSA	全ての軸上のラインに沿ってアプローチする
REPOSL	ラインに沿ってアプローチする
REPOSQ DISR=...	半径 DISR を持つ４分の１円弧に沿ってアプローチする
REPOSQA DISR=...	全ての軸上で、半径 DISR を持つ４分の１円弧に沿ってアプローチする
REPOSH DISR=...	直径 DISR を持つ半円に沿ってアプローチする
REPOSHA DISR=...	全ての軸上で、直径 DISR を持つ半円に沿ってアプローチする

再位置決め点

RMI	中断点にアプローチする
RMI DISPR=...	中断点の前で距離 DISPR （mm/inch）の点を進入点とする
RMB	ブロックスタート点にアプローチする
RME DISPR=...	エンド点の前で距離 DISPR （mm/inch）の点を進入点とする
A0 B0 C0	アプローチが行われる軸

機能

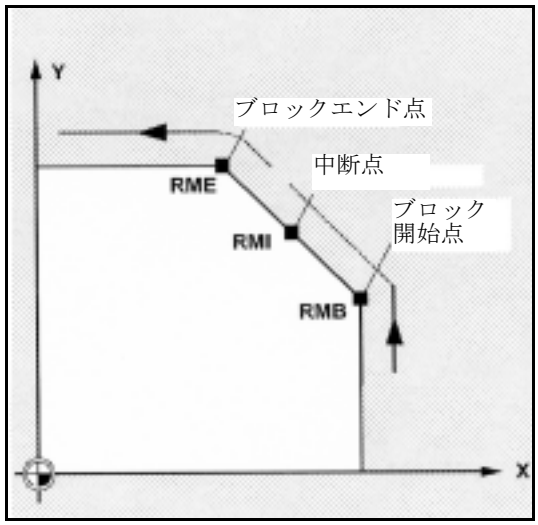


例えば、ツールが破損した、あるいは計測によるチェックを行うなどの理由で、マシニングオペレーション中にプログラムランを中断し、ツールから後退する場合、プログラムの制御の下で輪郭上の任意の点で再位置決めすることができます。

REPOS コマンドは、サブプログラムリターンジャンプ（例えば、M17 を介した）と同様に機能します。中断ルーチンで本コマンドの後にプログラムされたブロックは、実行されません。

プログラムランの中断の詳細については、プログラミング編 上級説明書のセクション「中断ルーチン」を参照してください。

動作



再位置決め点の定義

プログラムランが中断された NC ブロックを基準にして、3 つの異なる再位置決め点から 1 つを選択することができます：

- RMI, 中断点
  RMB, ブロック開始点または前回のエンド点
- RME, ブロックエンド点

RMI DISPR=... ブロックの開始点と中断点の間にある再位置決め点を選択することができるようにします。

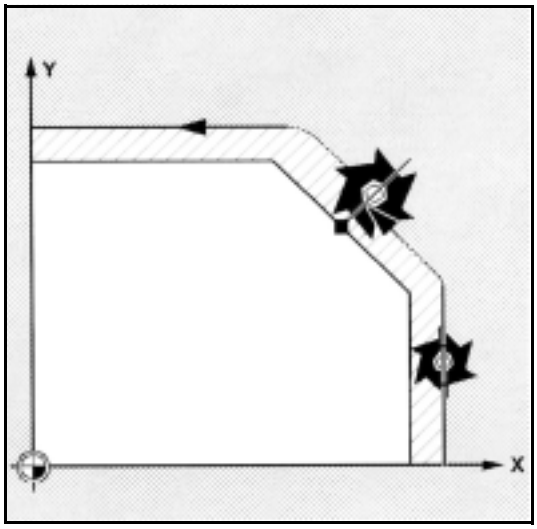
DISPR=... 再位置決め点と中断点またはエンド点の間で、輪郭距離を記述できるようにします（単位 mm/inch ）。高い値の場合でも、この点はブロックスタート点から遠ざかることはできません。

If no DISPR=... コマンドがプログラミングされ、その後中断点に応じて

DISPR=0 が適用されます。

新規ツールを使用したアプローチ

ツールの損傷によってプログラムを停止した場合：
新規 D 番号をプログラミングすると、再位置決め点で修正されたツールオフセット値を使用して、マシニングプログラムが続行されます。



ツールオフセット値が変更されている場合、中断点に再アプローチすることができない場合もあります。そのような場合、新規輪郭上の中断点に最も近い点がアプローチされます（例えば、DISPR によって変更された場合）。

輪郭のアプローチ

輪郭上でツールが再位置決めされるときの動作を、プログラミングすることができます。移動される軸のアドレスにゼロを入力して下さい。

コマンド REPOSA, REPOSQA および REPOSHA は、すべての軸を自動的に再位置決めします。ここの軸名を指定する必要はありません。

コマンド REPOSA, REPOSQA および REPOSHA をプログラミングすると、全てのジオメトリ軸が自動的に移動されます、すなわち、コマンドに軸の名前を入れる必要はありません。再位置決めする他の軸は全て、コマンドに指定する必要があります。

直線に沿ったアプローチ , REPOSA,REPOSL

ツールは直線に沿って再位置決め点にアプローチします。

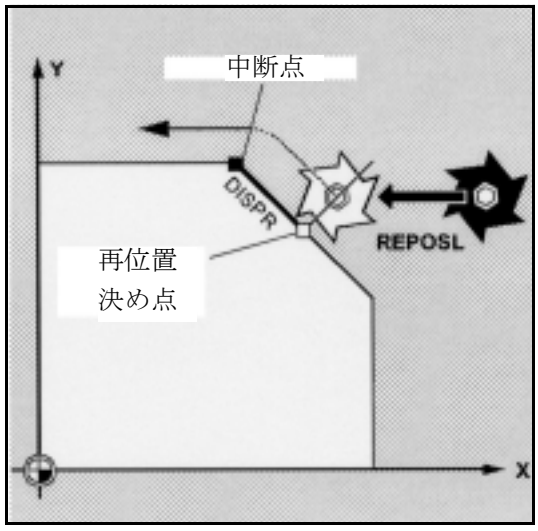
コマンド REPOSA を使用すると、全ての軸が自動的に移動されます。REPOSL を使用すると、移動する軸を指定することができます。

例 :

REPOSL RMI DISPR=6 F400

または

REPOSA RMI DISPR=6 F400

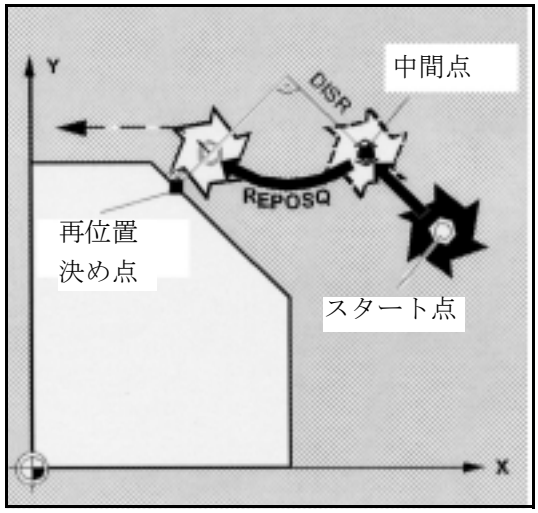


４分の１円弧に沿ったアプローチ , REPOSQ, REPOSQA

ツールは、半径 DISR=.... を持つ４分の１円弧に沿って再位置決め位置にアプローチします。制御系は、スタート点と再位置決め点の間の中間点を自動的に計算します。

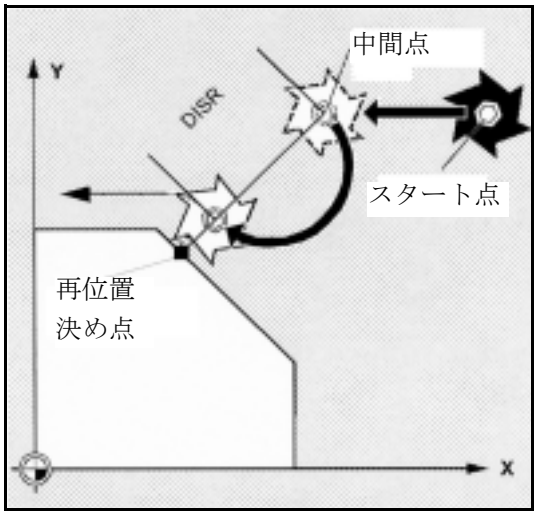
例 :

REPOSQ RMI DISR=10 F400



半円に沿ったアプローチ , REPOSH, REPOSHA

ツールは、直径 DISR=.... を持つ半円に沿って再位置決め点にアプローチします。制御系は、スタート点と再位置決め点の間の中間点を自動的に計算します。



例 :

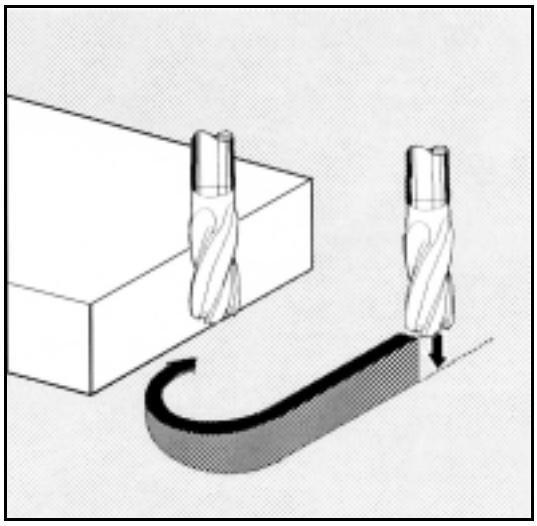
REPOSH RMI DISR=20 F400

円動作

REPOSH および REPOSQ の場合、次のようになります :

円は指定された工作平面 G17 から G19 で移動します。

アプローチブロックに第 3 のジオメトリ軸（インフィード方向）を指定する場合、ツール位置とインフィード方向にプログラムされた位置が合致していないと再位置決め位置へは螺旋形状に沿ってアプローチします。



次のような場合、制御装置は自動的に直線アプローチ REPOSL に切替えます :

DISR に値は指定されていません。

- 定義されたアプローチ方向はいずれも使用できない場合（移動情報を使用せずにブロックに中断をプログラムする）
- 現在の工作平面に垂直なアプローチ方向を使用する場合。

10 シンクロナイズドアクション挙動

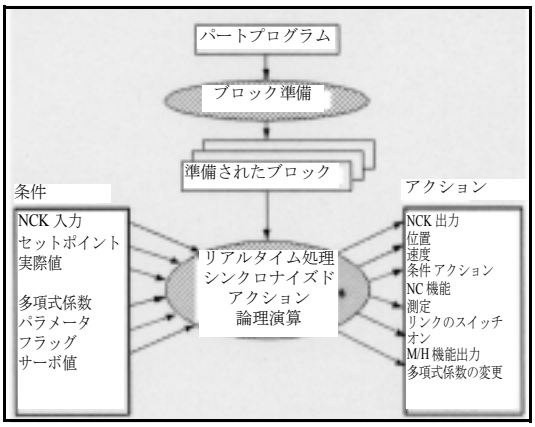
10.1 構造，基本情報

機能

シンクロナイズドアクションを使用すると，現在のパートプログラムから別のアクションを起動してそれらを同期的に実行することができます。

これらのアクションの開始点は，リアルタイムで（補間サイクル中で）評価される条件を使用して定義できます。したがって，これらのアクションはリアルタイムイベントに対する応答となり，アクションの実行はブロック境界の制約を受けません。

シンクロナイズドアクションには，アクションの有効性に関する情報，およびプログラムされたリアルタイム変数がスキャンされる頻度に関する情報（アクションが開始される頻度に関する情報）が含まれています。このため，アクションは補間サイクル中で一回だけトリガすることも周期的にトリガすることもできます。



プログラミング

DO action1 action2 ...
VOCABULARY_WORD condition DO action1 action2 ...
ID=n VOCABULARY_WORD condition DO action1 action2 ...
IDS=n VOCABULARY_WORD condition DO action1 action2 ...

説明

識別番号 ID/IDS	
ID=n	自動モードでのモーダルシンクロナイズドアクション，プログラムごとに，n = 1 ～ 255
IDS=n	各モードでのモーダルシンクロナイズドアクション，静的，n = 1 ～ 255
ID/IDS がない場合	自動モードでのノンモーダルシンクロナイズドアクション
ボキャブラリワード	
ボキャブラリワードがない場合	アクションは無条件で実行される。どの補間サイクルでも周期的に実行される。

WHEN	条件は 1 回満足されるまでチェックされる。関連するアクションは 1 回だけ実行される。
WHENEVER	条件は周期的にチェックされる。関連するアクションは条件が満足されている間は周期的に実行される。
FROM	条件が一度満足されると，シンクロナイズドアクションが有効である限りはアクションは周期的に実行される。
EVERY	条件が満足されるとアクションが 1 回実行され，条件が偽から真に切替るたびにまた実行される。条件は周期的にチェックされる。条件が満足されるたびに関連するアクションが実行される。
condition（条件）	リアルタイム変数のためのゲートロジック。条件は補間サイクル中にチェックされる。条件評価用に 一部の G コードをシンクロナイズドアクションにプログラムすることができる。
DO	条件が満足されていればアクションをトリガする。
Action（アクション）	条件が満足されたときに実行されるアクション。アクションの例としては，変数の割当て，軸カップリングの起動，NCK 出力のセット，M および H 機能の出力などがある。条件評価用に一部の G コードをシンクロナイズドアクションにプログラムすることができる。
シンクロナイズドアクション／テクノロジサイクルの協調	
CANCEL[n]	シンクロナイズドアクションをキャンセルする。
LOCK[n]	テクノロジサイクルを禁止する。
UNLOCK[n]	テクノロジサイクルをイネーブルする。
RESET	テクノロジサイクルをリセットする。

プログラミング例

WHEN $AA_IW[Q1]>5 DO M172 H510	軸 Q1 の実際値が 5 mm を超えると，補助機能 M172 と H510 が PLC インタフェースに出力される。

パートプログラム中でリアルタイム変数（実際値，デジタル入力あるいは出力の位置など）が見つかると，前のブロックが実行されてリアルタイム変数の値が得られるまで前処理が中断します。

使用されたリアルタイム変数は補間サイクル中に評価されます。

シンクロナイズドアクションの長所 :

前処理が中断されない。

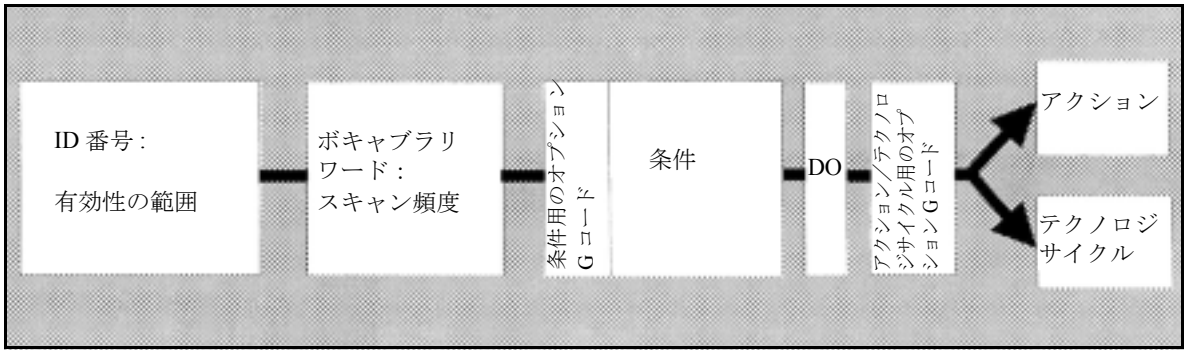
アプリケーション例

- 実行時間が重要なアプリケーション（ツールチェンジなど）の最適化
- 外部イベントに対する迅速な応答
- AC 制御のプログラミング
- 安全機能のセットアップ

10.1.1 プログラミングおよびコマンド要素

機能

シンクロナイズドアクションは単独に別のブロックでプログラムします。シンクロナイズドアクションは，次の実行ブロックでマシン機能（G0，G1，G2，G3 を使用した移動動作，補助機能出力を有するブロックなど）をトリガします。シンクロナイズドアクションは 5 つのコマンド要素から構成されます。各コマンド要素はそれぞれ別のタスクを有しています。



例：

ID=1	WHENEVER	$A_IN[1]==1	DO	$A_OUT[1]=1
シンクロナイズドアクション番号 1	いつでも	入力 1 がセットされたとき	次のことを行う	出力 1 をセット

10.1.2 有効性の範囲：識別番号 ID

機能

シンクロナイズドアクションの有効性の範囲は識別番号（モーダル ID）によって定義されます：

- モーダル ID がない場合
  シンクロナイズドアクションは自動モードでしか有効になりません。シンクロナイズドアクションは次の実行可能ブロック（動作命令またはその他のマシンアクションを有するブロック）にしか適用されず，非モーダルです。
  例：

```
WHEN $A_IN[3]==TRUE DO $A_OUTA[4]=10
G1 X20                                  ; 実行可能ブロック
```

- ID=n; n=1...255
  シンクロナイズドアクションは以降のブロックでモーダルに適用され，CANCEL(n) によって，または同じ ID を持つ新たなシンクロナイズドアクションをプログラミングすることによって停止します。
  M30 ブロックで適用されたシンクロナイズドアクションも有効のままです（必要であれば CANCEL コマンドで停止してください）。
  ID シンクロナイズドアクションは自動モードでしか適用されません。
  例：

```
ID=2 EVERY $A_IN[1]==1 DO POS[X]=0
```

- IDS=n; n=1...255
  これらの静的シンクロナイズドアクションは全運転モードでモーダルに適用されます。
  これらの静的シンクロナイズドアクションは，パートプログラムから開始だけでなく，PLC による非同期サブプログラム (ASUP) からも直接（電源オン後に）定義できます。

このようにしてアクションは起動でき，NC で選択されるモードとは関係なく実行されます。

例 **:**

```
IDS=1 EVERY $A_IN[1]==1 DO POS[X]=100
```

アプリケーション :

- JOG モードでの AC ループ
- 統合された安全用の論理演算
- 監視機能，全てのモードでのマシンステータスに対する応答

実行の順序

モーダルに，または静的に適用されるシンクロナイズドアクションは，ID 番号順に（補間サイクルで）実行されます。

非モーダルシンクロナイズドアクション（ID 番号なし）は，モーダルシンクロナイズドアクションが実行完了後，プログラム順に実行されます。

10.1.3 ボキャブラリワード

機能

ボキャブラリワードは，下記の条件が何回スキャンされるか，またそれに関連したアクションが何回実行されるべきかを指定します。

- ボキャブラリワードがない場合：
  ボキャブラリワードがプログラムされていない場合，常に条件が満たされていると考慮されます。同期コマンドは周期的に実行されます。
- WHEN
  条件は 1 回満足されるまで補間サイクルごとにチェックされます。その結果アクションは 1 回だけ実行されます。
- WHENEVER
  条件は補間サイクルごとにチェックされます。アクションは条件が満足されている間は補間サイクルごとに実行されます。

例：

```
DO $A_OUTA[1]=$AA_IN[X]
        ; アナログ出力での実際値の出力
```

- FROM

  条件は 1 回満足されるまで補間サイクルごとにチェックされます。シンクロナイズドアクションが有効である限りは，つまり条件が満足されなくなってからも，アクションは実行されます。
- EVERY

  条件は補間サイクルごとにチェックされます。条件が満足されるたびに，アクションが 1 回実行されます。

  パルスエッジ制御：

  条件が偽から真に切替るたびにアクションが再開されます。

例：

ID=1 EVERY $AA_IM[B]>75 DO

POS[U]=IC(10) FA[U]=900;

マシン座標において軸 B の実際値が値 75 を超えると，U 軸が軸フィード 10 だけ前進するべきです。

条件

2 つのリアルタイム変数を比較することによってアクションを実行するのか，1 つのリアルタイム変数と前処理中に計算された式とを比較することによってアクションを実行するのかを決めます。

条件（）中の比較の結果を，ブール演算子を使用してゲートをかけることもできます。

条件は補間サイクルごとにチェックされます。条件が満足されると関連するアクションが実行されます。

条件に一部の G コードを指定できます。これは，条件評価用に定義された設定およびアクション／テクノロジサイクル用に定義された設定を，現在有効になっているパートプログラムとは無関係に有することができるという意味です。シンクロナイズドアクションはトリガ条件が満足された場合常に，定義された初期ステータスでそれぞれのアクションを実行するようになっているので，プログラミング環境から切り離しておく必要があります。

アプリケーション例：

条件評価およびアクション用の測定系を G コード，G70，G71，G700，G710 で定義します。

条件用に指定された G コードは，その条件の評価用に適用されるのと同様に，アクション用に別の G コードが指定されていない場合はそのアクションに対しても適用されます。

各条件パートには G コードグループの G コードを 1 つしかプログラムできません。

プログラミング例条件の例：

WHENEVER $AA_IM[X] > 10.5*SIN(45) DO ...	前処理中に計算された式との比較
WHENEVER $AA_IM[X] > $AA_IM[X1] DO ...	他のリアルタイム変数との比較
WHENEVER ($A_IN[1]==1) OR ($A_IN[3]==0) DO ...	2 つの論理ゲートをかけられた比較

条件の例：

- リアルタイム変数の比較（アナログ／デジタル，入力／出力など）
- 比較結果のブール演算子を使用したゲーティング
- リアルタイム式の計算
- ブロックの先頭からの時間／距離
- ブロックエンドからの距離
- 測定された値，測定された結果
- サーボ値
- 速度，軸ステータス

10.1.4 アクション

機能

シンクロナイズドアクションのアクションには複数プログラムできます。1 つのブロックにプログラムされたアクションは全て同じ補間サイクルでスタートします。

アクションはアクション／テクノロジサイクルについて一部の G コードとともに使用できます。この G コードは，必要であれば，ブロックおよびテクノロジサイクルのアクション全てに対する条件用に設定された G コードから，別の G コードに変更することができます。アクションパートにテクノロジサイクルがあれば，テクノロジサイクルが完了した後 G コードは次の G コードまで，以降のアクション全てに対してモーダルに適用され続けます。

G コードグループ（G70，G71，G700，G710）からの G コードしかプログラムできません。

可能なアクション：

- 変数の割当て
- 設定データの書込み
- 制御パラメータの設定
- DELDTG：迅速な移動距離の削除
- RDISABLE：読込みディスエーブルの設定
- M，S および H 補助機能の出力
- STOPREOF：前処理停止のキャンセル
- FTOC：オンラインツールオフセット
- 評価機能（多項式）の定義
- SYNFCT：評価機能の起動：AC 制御
- バイナリ信号およびアナログ信号に応じてプログラムされたブロック内で数段階の送り速度を切換える
- 送り速度オーバライド
- 位置決め軸 (POS) および主軸 (SPOS) をスタート／位置決め／停止する
- PRESETON：実際値の設定
- カップリングされた軸動作／リーディング値カップリングを起動または停止する

- 測定
- 追加の安全機能のセットアップ
- デジタル信号およびアナログ信号の出力

プログラミング例

2 つのアクションを有するシンクロナイズドアクション

WHEN $AA_IM[Y] >= 35.7 DO M135 $AC_PARAM=50
条件が満足されると，M135 が PLC に出力され，オーバライドは 50% に設定されます。

アクションと同じく，プログラム（単一軸プログラム，テクノロジサイクル）も指定できます。このことは，シンクロナイズドアクションに個々にプログラムすることもできるアクションを比較するだけで行なえます。そのようなプログラムの個々のアクションは補間サイクルで連続して実行されます。

（注）アクションはどのようなモードを選択した場合でも実行できます。しかし、下記のアクションは，プログラムが有効なときに自動モードでしか有効になりません。

- STOPREOF
- DELDTG

10.1.5 シンクロナイズドアクションの概要

- ユーザレベル（パートプログラム）での，補間サイクル中の順番のプログラミング
- 補間サイクル中のイベント／ステータスへの応答
- リアルタイムのゲーティング論理
- I/O，制御ステータス，およびマシンステータスへのアクセス
- 補間サイクル中に実行される周期の順番のプログラミング
- 特定の NC 機能（読込みディスエーブル，軸方向にオーバーレイされたモーション）のトリガ
- パス動作と並行したテクノロジ機能の実行
- ブロック境界の制約を受けないテクノロジ機能のトリガ

- シンクロナイズドアクション用に診断が使用できます
- シンクロナイズドアクションで使用されるメインラン変数の拡張
- シンクロナイズドアクション中の複雑な条件
- シンクロナイズドアクション中の式の拡張：
  基本算術演算を使用したリアルタイム変数と補間サイクル中の機能の組合せ。インデックスを介してのメインラン変数の間接的なアドレッシングをオンラインで変更できます。
  シンクロナイズドアクションからの設定データをオンラインで変更したり評価したりできます。
- 構成：マシンデータを介して，同時にアクティブになるシンクロナイズドアクションの数が設定できます。
- シンクロナイズドアクションからの位置決め軸動作および主軸のスタート（コマンド軸）
- シンクロナイズドアクションからのプリセット
- 軸カップリングの起動，停止，パラメータ化：
  リーディング値カップリング，カップリングされた軸の動作
- 軸測定機能の起動／停止

- ソフトウェアドグ
- 前処理停止なしの移動距離の削除
- 単一軸プログラム，テクノロジサイクル
- シンクロナイズドアクションは JOG モードではプログラム境界を超えてアクティブ
- PLC からの影響を受ける可能性のあるシンクロナイズドアクション
- 保護されたシンクロナイズドアクション
- オーバーレイされた動作／クリアランス制御用の拡張

10.2 条件用およびアクション用の基本モジュール

リアルタイム変数

リアルタイム変数は，補間サイクル中に評価され，書込まれます。

リアルタイム変数とは，

- $A... ，メインラン変数
- $V... ，サーボ変数

です。

これらの変数を特に識別するために，$$ を付けてプログラムすることができます：

例えば $AA_IM[X] は $$AA_IM[X] と同じです。

補間サイクル中で評価／割当てが行なわれる時は，設定データおよびマシンデータは $$ を付けて識別しなければなりません。

変数表が 14 章に記載されています。

リアルタイムでの計算

リアルタイムでの計算は，INT，REAL，BOOL のデータタイプでしか行なうことができません。

リアルタイム式とは，補間サイクル中に実行され，NC アドレスと変数への割当て用に，規定の条件とアクションの下で使用される計算式です。

- **比較**

  条件の中で，同じデータタイプの変数または部分式を比較することができます。結果は常にデータタイプ BOOL になります。

  通常の比較演算子は全て使用できます (==, <>, <, >, <=, >=)。

- **ブール演算子**

  変数，定数，および比較は，通常のブール演算子 (NOT, AND, OR, XOR) を使用してゲートをかけることができます。

- **ビット演算子**

  ビット演算子 B_NOT，B_AND，B_OR，B_XOR が使用できます。

  オペランドは INTEGER タイプの変数または定数です。

- **基本算術オペレーション**

  INTEGER タイプおよび REAL タイプのリアルタイム変数の場合，基本算術オペレーション (+, -, *, /, DIV, MOD) に従って変数どうしを計算したり，定数とともに計算したりすることができます。

- **数学的機能**

  数学的機能はデータタイプ REAL のリアルタイム変数には適用できません。
  (SIN, COS, TAN, ASIN, ACOS, ABS, TRUNC, ROUND, LN, EXP, ATAN2, ATAN, POT, SQRT, CTAB, CTABINV)

  例：

```
DO $AC_PARAM[3] = COS($AC_PARAM[1])
```

（注）同じデータタイプの変数しかゲートをかけることはできません。

正：$R10=$AC_PARAM[1]
誤：$R10=$AC_MARKER[1]

掛算および割算は足算および引算より先に行なわれます。式を括弧でくくることもできます。

演算子 DIV および MOD をデータタイプ REAL に使用できます。

例：

DO $AC_PARAM[3] = $A_INA[1]-$AA_IM[Z1]	; 2 つのリアルタイム変数の引算
WHENEVER $AA_IM[x2] < $AA_IM[x1]-1.9 DO $A_OUT[5] = 1	
	; リアルタイム変数から定数を引く
DO $AC_PARAM[3] = $INA[1]-4*SIN(45.7 $P_EP[Y])*R4	
	; 定数式，前処理中に計算される

- **インデックス**

  リアルタイム変数はリアルタイム変数を使用してインデックスすることができます。

（注）リアルタイムでつくられない変数を，リアルタイム変数を使用してインデックスしてはいけません。

例：

WHEN...DO $AC_PARAM[$AC_MARKER[1]] = 3

不正：

$AC_PARAM[1] = $P_EP[$AC_MARKER]

プログラミング例

リアルタイム式の例

ID=1 WHENEVER ($AA_IM[Y]>30) AND ($AA_IM[Y]<40) DO $AA_OVR[S1]=80	位置ウィンドウの選択
ID=67 DO $A_OUT[1]=$A_IN[2] XOR $AN_MARKER[1]	2 つのブール信号の評価
ID=89 DO $A_OUT[4]=$A_IN[1] OR ($AA_IM[Y]>10)	比較結果の出力

10.3 シンクロナイズドアクション用の特殊なリアルタイム変数

下記に挙げられているリアルタイム変数がシンクロナイズドアクションで使用できます：

10.3.1 フラグ／カウンタ $AC_MARKER[n]

機能

マーカ変数をシンクロナイズドアクション内で読取り／書込みできます。

チャンネル別フラグ／カウンタ $AC_MARKER[n]

データタイプ：INTEGER
チャンネル別フラグ変数は，同じ名前の下では各チャンネルで 1 つだけしか存在しません。

例：

```
WHEN ... DO $AC_MARKER[0] = 2
WHEN ... DO $AC_MARKER[0] = 3
WHEN $AC_MARKER == 3 DO $AC_OVR=50
```

10.3.2 タイマ変数 $AC_TIMER[n]

機能

システム変数 $AC_TIMER[n] を使用すると，定義された待ち時間が経ってからアクションをスタートさせることができます。

データタイプ：REAL

単位：s

n：タイマ変数の番号

- **タイマのセット**

  $AC_TIMER[n]= 値という指定をすることによってタイマのカウントが始まります。

n：タイマ変数の番号

value：開始値（通常 0）

- **タイマの停止**

  $AC_TIMER[n]=-1 というように負の値を代入すると，タイマのカウントが停止します。

- **タイマの読取り**

  現在の時間値は，タイマが作動している時でも，停止してからでも読取ることができます。

  -1 の値が代入されてタイマが停止すると，最新のタイマ値がそのまま残されて，それが読取られます。

例：

デジタル入力の検出から 500 ms 遅れたアナログ出力による実際値の出力

```
WHEN $A_IN[1] == 1 DO $AC_TIMER[1]=0                    ；タイマのリセットおよび起動
WHEN $AC_TIMER[1]>=0.5 DO $A_OUTA[3]=$AA_IM[X] $AC_TIMER[1]=-1
```

10.3.3 シンクロナイズドアクションパラメータ $AC_PARAM[n]

機能

データタイプ REAL

n：パラメータの番号：0-n

シンクロナイズドアクションパラメータ $AC_PARAM[n] はシンクロナイズドアクション中で，計算用に，またバッファとして使用されます。

チャンネルごとに使用可能な AC パラメータ変数の数が，マシンデータ MD 28254:MM_NUM_AC_PARAM を使用して定義されます。

パラメータはチャンネルごとに 1 度だけ同じ名前で使用できます。フラグはダイナミックメモリに保存されます。

10.3.4 R パラメータ $Rxx へのアクセス

機能

データタイプ REAL

静的変数はパートプログラムなどで計算用に使用されます。静的変数は $ を付けることによって，補間サイクル中でアドレス可能です。

例：

WHEN $AA_IM[X]>=40.5 DO $R10=$AA_MM[Y]	R パラメータ 10 への書込みアクセス。
WHEN $AA_IM[X]>=6.7 DO $R[$AC_MARKER[1]]=30.6	; フラグ 1 に番号が与えられている R パラメータへの読取りアクセス

(注)

アプリケーション：

シンクロナイズドアクションで R パラメータを使用することによって，下記のことができます。

- プログラムが終了した後，NC リセットの後，および電源オン後もそのまま残しておきたい値の保存
- 保存された値を R パラメータ表示に表示する
- シンクロナイズドアクション用に指定された値のアーカイブ

R パラメータは「ノーマルな」算術変数 Rxx として使用するか，リアルタイム変数 $Rxx として使用しなければなりません。

R パラメータをシンクロナイズドアクションで使用した後でもう 1 度「ノーマルな」算術変数として使用したい場合は，前処理停止が，処理とメインランを同期化するために，STOPRE を使用してプログラムされていることを確認してください：

例

WHEN $AA_IM[X]>=40.5 DO $R10=$AA_MM[Y]	シンクロナイズドアクションでの R10 の使用
G01 X500 Y70 F1000	
STOPRE	前処理停止
IF R10>20	算術変数の評価

10.3.5 マシンデータおよび設定データの読取り／書込み

機能

シンクロナイズドアクションのマシンデータおよび設定データ (MD, SD) の読取りおよび書込みができます。

- **固定の MD，SD を読取る場合**

  固定の MD および SD は，通常のパートプログラムコマンドと同様に，シンクロナイズドアクション内で処理されます。固定の MD および SD の先頭には文字 $ が付きます。

例

```
ID=2 WHENEVER $AA_IM[z]<$SA_OSCILL_REVERSE_POS2[Z]-6 DO $AA_OVR[X]=0
  ; この例では，発振のための逆転位置 2 がアドレスされる。その逆転位置 2 は変更できないと考えられる。
```

- **変更可能な MD，SD を読取る場合**

  変更可能な MD および SD は，シンクロナイズドアクションの内部からアドレスされます。MD および SD の先頭には $$ という記号が付けられ，補間サイクルで評価されます。

  例

```
ID=1 WHENEVER $AA_IM[z]<$$SA_OSCILL_REVERSE_POS2[Z]-6 DO $AA_OVR[X]=0
  ; ここでは，逆転位置は加工中にコマンドを使用して変更できると考えられる。
```

- **MD，SD を書込む場合**

  前提条件：

  現在のアクセス許可設定は，書込みアクセスを許可する設定になっていなければなりません。変更を直ちに有効にしたい場合にのみシンクロナイズドアクションから MD および SD を

  変更してください。すべての MD と SD について有効ステータスが下記のリファレンスに示されています。

  参照： /LIS/, Lists

アドレス：
変更される MD および SD は，先頭に $$ を付けてアドレスしなければなりません。

例

```
ID=1 WHEN $AA_IW[X]>10 DO $$SN_SW_CAM_PLUS_POS_TAB_1[0]=20
                          $$SN_SW_CAM_MINUS_POS_TAB_1[0]=30
                          ; SW カムの切換え位置の変更。注記：切換え位置は，2 ～ 3 補間サイクル早めに正規の位置に到達するように早めに変更されなければなりません。
```

10.3.6 FIFO 変数 $AC_FIFO1[n] ...$AC_FIFO10[n]

機能

データタイプ REAL

関連するデータの順番を保存するのに 10 個の FIFO 変数（循環式保管）が使用できます。

アプリケーション：

- 周期測定
- パスの実行

各要素は読取りまたは書込みモードでアクセスできます。

使用できる FIFO 変数の数はマシンデータ MD 28260: NUM_AC_FIFO を使用して定義されます。

FIFO 変数に書込める値の数はマシンデータ MD 28264: LEN_AC_FIFO を使用して定義されます。
FIFO 変数は全て同じ長さです。

0 から 5 までのインデックスには特別な意味があります：

n=0: 書込み中：新たな値が FIFO に保存される
読取り中：1 番古い要素が読取られ，FIFO から取出される

n=1: 1 番先に保存された要素へのアクセス

n=2: 1 番後で保存された要素へのアクセス

n=3: 全 FIFO 要素の合計

n=4: FIFO で使用できる要素の数

各要素への読取りおよび書込みアクセスが可能。

要素の数をリセットすることによって FIFO 変数をリセットする。例：第 1 FIFO 変数の場合：$AC_FIFO1[4]=0

n=5: FIFO のスタートに関連する現在の書込みインデックス

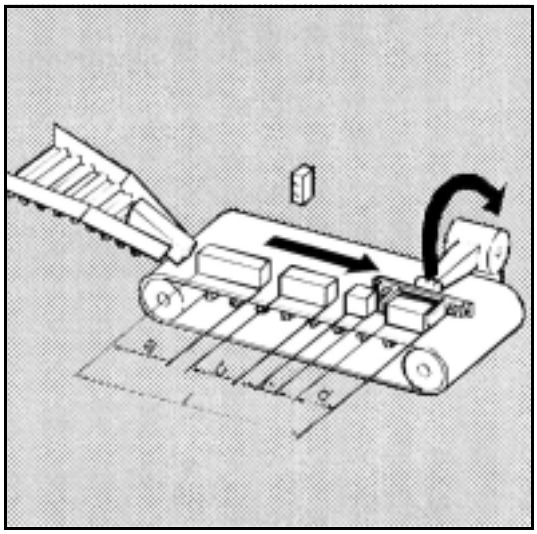
n=6 ～ 6+n max:

n 番目の FIFO 要素へのアクセス

プログラミング例

循環式保管

製造作業中，異なった長さ (a, b, c, d) の製品を運ぶのにベルトコンベアが使用されます。従って，運搬長さ "I" のベルトコンベアが運搬する製品の数は，プロセスに含まれる個々の製品の長さに応じて変わります。運搬速度が一定の場合，ベルトコンベアから製品を取出すための機能は，製品によって変化する到着時間に適応させなければなりません。



DEF REAL INTV=2.5	ベルトに置かれた製品どうしの距離が一定
DEF REAL TOTAL=270	長さの測定と取出し位置との間の距離
EVERY $A_IN[1]==1 DO $AC_FIFO1[4]=0	プロセスの先頭で FIFO をリセットする
EVERY $A_IN[2]==1 DO $AC_TIMER[0]=0	製品が光バリアをさえぎったら，タイミングを開始する。
EVERY $A_IN[2]==0 DO $AC_FIFO1[0]=$AC_TIMER[0]*$AA_VACTM[B]	
; 遮光が終わると，測定された時間と運搬速度から製品の長さを計算して FIFO に保存する。	
EVERY $AC_FIFO1[3]+$AC_FIFO1[4]*ZWI>=GESAMT DO POS[Y]=-30 $R1=$AC_FIFO1[0]	
; 全ての製品の長さと製品どうしの間隔の合計が，配置位置と取出し位置との間の長さより大きくなるかその長さと同じになったらただちに，取出し位置でベルトコンベアから製品を取出し，FIFO から製品の長さを読出す。	

10.4 シンクロナイズドアクションの中のアクション

10.4.1 補助機能の出力

機能

条件が満足されていれば，M，H および S 機能を加工ブロックごとに最高 10 個まで出力することができます。

補助機能出力はアクションコードワード "DO" を使用して起動します。

補助機能は補間サイクル中にただちに出力されます。補助機能用にマシンデータで定義された出力のタイミングは有効ではありません。出力のタイミングは，条件が満足されたときに指定されます。

例：

特定の軸位置で冷却剤をオンにする：

WHEN $AA_IM[X]>=15 DO M07

動作

補助機能は，（モーダル ID がない）ノンモーダルシンクロナイズドアクションでボキャビュラリワード WHEN または EVERY を使用してしかプログラムできません。補助機能を有効にするかしないかは PLC によって，例えば NC のスタートを介して指定されます。

（注）

下記のことは動作シンクロナイズドアクションから行なえません：

- M0, M1, M2, M17, M30: プログラム停止／終了（M2，M17，M30 はテクノロジサイクルに対しては使用可能）
- M70: 主軸機能
- M6 によってまたはマシンデータを使用して設定されたツールチェンジ用の M 機能
- M40, M41, M42, M43, M44, M45: ギアチェンジ

プログラミング例

WHEN $AA_IW[Q1]>5 DO M172 H510	軸 Q1 の実際値が 5 mm を超えると，補助機能 M172 および H510 が PLC へ出力されます。

10.4.2 読込みディスエーブルの設定 RDISABLE

機能

条件が満足されていれば，RDISABLE を使用すると，メインプログラムでそれ以上ブロックが実行されなくなります。

しかし、プログラムされた同期モーションアクションは実行され，それ以降のブロックも用意されます。

ブロックの先頭で RDISABLE が使用されると，RDISABLE が有効であるかどうかに関わらず，常に正確な位置決めがトリガされます。

プログラミング例

外部の入力に応じて補間サイクルでプログラムをスタートしてください。

...	
WHENEVER $A_INA[2]<7000 DO RDISABLE	; 入力 2 の電圧が 7V を超えるとプログラムが停止する (1000 = 1V)。
N10 G1 X10	; 条件が満足されていれば，N10 の最後で読込みディスエーブルが有効になる
N20 G1 X10 Y20	
...	

10.4.3 前処理停止のキャンセル STOPREOF

機能

明確にプログラムされた前処理停止 STOPRE または有効なシンクロナイズドアクションによって暗示的に起動された前処理停止の場合，STOPREOF は，条件が満足されるとただちに，次の加工ブロックの後で前処理停止をキャンセルします。

（注）STOPREOF は，ボキャビュラリワード WHEN を使用して，ノンモーダル（ID 番号なし）にプログラムしなければなりません。

プログラミング例

ブロックエンドでの高速プログラム分岐。

WHEN $AC_DTEB<5 DO STOPREOF	; ブロックエンドまでの距離が 5 mm 未満であれば前処理停止がキャンセルされる。
G01 X100	; 直線補間が実行された後で前処理停止がキャンセルされる。
IF $A_INA[7]>500 GOTOF MARKE1=X100	; 入力 7 の電圧が 5 V を超えると，ラベル 1 にジャンプする。

10.4.4 残移動量の削除

パスおよび指定された軸について，条件に応じて残移動量の削除をトリガすることができます。

- 迅速な，あらかじめ準備された残移動量削除
- あらかじめ準備されていない残移動量削除

ができます。

10.4.5 あらかじめ準備された残移動量削除, DELDTG, DELTG（軸 1,..）

機能

DELTDG によってあらかじめ準備された残移動量削除を行なうと，トリガイベントに対して迅速に応答できます。そのため，例えば下記のようなサイクルタイムが重要なアプリケーションに対してあらかじめ準備された残移動量削除が使用されます。

- 残移動量削除を行なってから次のブロックがスタートするまでの時間を非常に短くしなければならない場合
- 残移動量削除のための条件がほぼ確実に満足されるであろう場合

動作

あらかじめ準備された残移動量削除がトリガされた移動ブロックエンドで，前処理停止が暗示的に起動されます。

従って連続パスモードまたは位置決め軸動作は，迅速な残移動量削除によってブロックエンドで中断，または停止されます。

残移動量は，システム変数 $AC_DELT または $AC_DELT[軸] を使用して求めることができます。

プログラミング例

迅速残移動量削除パス

WHEN $A_IN[1]==1 DO DELDTG	
N100 G01 X100 Y100 F1000	; 入力がセットされたら，動作がキャンセルされる。
N110 G01 X...	
IF $AC_DELT>50...	

プログラミング例

迅速な軸の残移動量削除

POS[X1]=100 G1 Z100 F1000

プログラムされた位置決め動作の停止：

ID=1 WHEN $A_IN[1]==1 DO MOV[V]=3 FA[V]=700	軸のスタート
WHEN $A_IN[2]==1 DO DELDTG(V)	残移動量削除，MOV=0 を使用して軸を停止する

入力された電圧に応じての残移動量削除：

WHEN $A_INA[5]>8000 DO DELDTG(X1)

; 入力 5 の電圧が 8 V を超えると，ただちに軸 X1 の残移動量を削除する。
パス動作は続行する。

制限事項：

あらかじめ準備された残移動量削除は，

- ツール半径補償が有効な場合は使用できません。
- このアクションは，ノンモーダルシンクロナイズドアクション（ID 番号なし）中にしかプログラムできません。

10.4.6 多項式定義 FCTDEF，ブロック同期

ブロックに同期させた多項式定義 , FCTDEF

プログラミング

FCTDEF(Polynomial_No.,LLIMIT,ULIMIT,a0,a1,a2,a3)

説明

Polynomial_No.	3 次多項式の番号
LLIMIT	機能値に対する下限
ULIMIT	機能値に対する上限
a_0,a_1,a_2,a_3	多項式係数

機能

FCTDEF を使用すると，3 次多項式を $y=a_0+a_1・x+a_2・x^2+a_3・x^3$ として定義することができます。これらの多項式は，メインラン変数（リアルタイム変数）から機能値を計算するために，オンラインツールオフセット FTOC および評価機能 SYNFCT によって使用されます。

多項式は，機能 FCTDEF を使用するか（ブロック定義），あるいはシステム変数を介して定義します：

$AC_FCTLL[n]	機能値に対する下限
$AC_FCTUL[n]	機能値に対する上限
$AC_FCT0[n]	a_0
$AC_FCT1[n]	a_1
$AC_FCT2[n]	a_2
$AC_FCT3[n]	a_3
n	多項式の番号

（注）

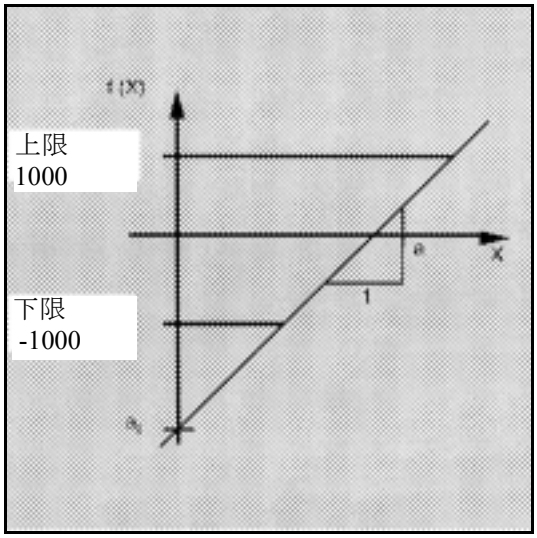
- システム変数は，パートプログラムまたはシンクロナイズドアクションから書込むことができます。パートプログラムから書込むときは，書込みがブロックと同期していることを確認するために STOPRE をプログラムしてください。
  システム変数 $AC_FCTLL[n]，$AC_FCTUL[n], $AC_FCT0[n] ～ $AC_FCTn[n] は，シンクロナイズドアクション内から変更できます。

  シンクロナイズドアクションから書込むときは，多項式係数および機能値リミットはただちに有効になります。

プログラミング例

直線部分用の多項式：

上限 1000，下限 -1000，縦軸部分 a0=$AA_IM[X]，
直線の傾き 1 の多項式は：




```
FCTDEF(1, -1000,1000,$AA_IM[X],1)
```

10.4.7 評価機能 SYNFCT

プログラミング

SYNFCT(Polynomial_No., re s al-time variable output, real-time variable input)

説明

Polynomial_No.	FCTDEF を使用して定義された多項式についている番号（サブセクション「多項式定義」を参照）。
Real-time variable output	リアルタイム変数の書込み
Real-time variable input	リアルタイム変数の読取り

機能

SYNFCT は（例えばアナログ入力 , 実際値などの）実行と同期してリアルタイム変数を読取ります。そして，読取ったリアルタイム変数を，評価多項式 (FCTDEF) を使用して機能値を最高 3 段階まで（例えばオーバライド，速度，軸位置など）計算するために使用します。計算結果はリアルタイム変数に出力され，FCTDEF によって上限と下限の制限を受けます（セクション 10.4.8 を参照してください）。

変数（リアルタイム変数）は，選択して下記のような処理演算に直接含めることができます。

- 足算の処理演算
- 掛算の処理演算
- 位置オフセットとしての処理演算

アプリケーション

評価機能は,

- AC 制御（適応制御）で,
- レーザ出力制御で,
- 位置フィードフォワードとともに，使用されます。

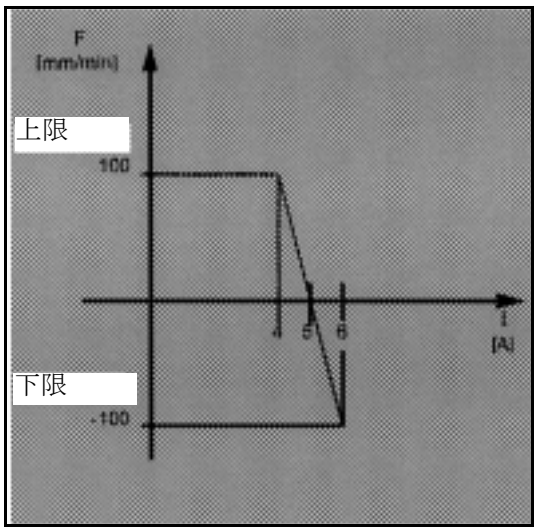
10.4.8 AC 制御（足算）

プログラミング例

足算によるプログラムされた送り速度の調整

プログラムされた送り速度は，X 軸（インフィード軸）の電流を使用して，足算をすることによって制御するようになっています：

送り速度は +/- 100 mm/min の範囲内で変化するべきです。電流は作用点 5A を中心として +/-1A の範囲で変動します。



1. 多項式定義

係数の求め方

$y = f(x) = a_0 + a_1x + a_2x^2 + a_3x^3$

a_1 = -100mm/1 min A

a_0 = -(-100)*5 =500

$a_2 = a_3 = 0$（2 次または 3 次の要素なし)

上限 = 100

下限 = -100

従って：

```
FCTDEF(1,-100,100,500,-100,0,0)
```

2. AC 制御の起動

```
ID=1 DO SYNFCT(1,$AC_VC,$AA_LOAD[x])
```

; $AA_LOAD[x] を使用して現在の軸の負荷（最大ドライブ電流値の %）を読取り，上記で定義された多項式を使用してパス送り速度オーバライドを計算する。

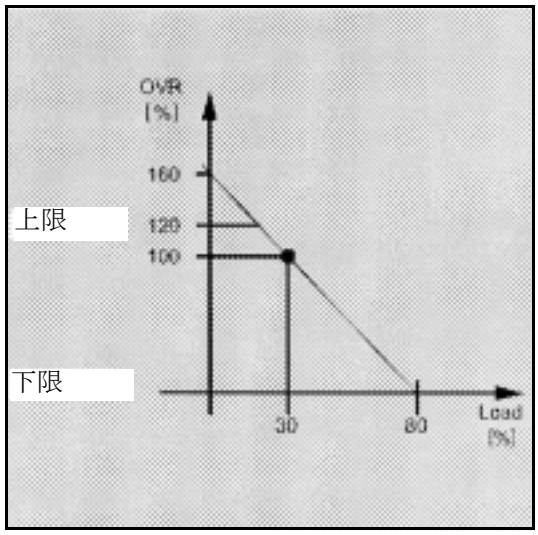
10.4.9 AC 制御 (掛算)

プログラミング例

掛算によるプログラムされた送り速度の調整

プログラムされた送り速度を掛算によって調整することが目的です。送り速度は（ドライブにかかる負荷に応じて）一定のリミットを越えてはいけません：

- 送り速度はドライブ負荷が 80% になると停止するようになっています：オーバライド = 0
- ドライブ負荷が 30% であれば，プログラムされた送り速度で移動できます：オーバライド = 100%
- 送り速度は 20% だけ超過することができます：最大オーバライド = 120%



1. 多項式定義

係数の求め方

$y = f(x) = a_0 + a_1x + a_2x^2 + a_3x^3$

$a_1 = -100\%/(80-30)\% = -2$

$a_0 = 100 + (2*30) = 160$

$a_2 = a_3 = 0$（2 次または 3 次の要素なし）

上限 = 120

下限 = 0

従って：

```
FCTDEF(2,0,120,160,-2,0,0)
```

2. AC 制御の起動

```
ID=1 DO SYNFCT(2,$AC_OVR,$AA_LOAD[x])
```

; $AA_LOAD[x] を使用して現在の軸の負荷（最大ドライブ電流値の %）を読取り，上記で定義された多項式を使用して送り速度オーバライドを計算する。

10.4.10 オンラインツールオフセット FTOC

プログラミング

FTOC(Polynomial_No., RV, length_2_3 or Radius4,
channel, spindle)

説明

Polynomial_No.	FCTDEF を使用して定義された多項式用の番号，このセクションのサブセクション「多項式定義」を参照してください。
RV	リアルタイム変数。このリアルタイム変数用に，指定された多項式用の機能値が計算されるようになっている。
長さ 1_2_3 Radius4	計算された機能値が加えられる長さ補償（$TC_DP1 から 3）または半径補償。
チャンネル	オフセットが有効になっているチャンネルの番号。ここでは，アクティブチャンネル内のオフセットについての指定は行なわれない。FTOCON はターゲットチャンネルから起動させなければならない。
主軸	補償されるべき主軸が有効でない場合しか指定されない。

機能

FTOC を使用すると，多項式が FCTDEF を使用して，例えば軸の実際値などの基準値に応じてプログラムされた後に，ジオメトリ軸がオーバーレイされた動作をすることができます。

このことは，モーダルなオンラインツール補償またはクリアランス制御を，シンクロナイズドアクションとしてプログラムすることもできるということを意味します。

アプリケーション

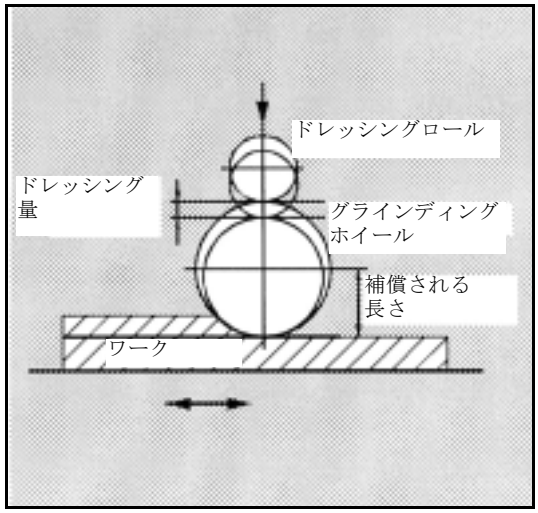
同じチャンネル内または別々のチャンネル内（加工およびドレッシングチャンネル）のワークの加工およびグラインディングホイールのドレッシング。

グラインディングホイールのドレッシングについての追加条件および指定が，PUTFTOCF を使用するツールオフセットに適用されるのと同じ方法で FTOC に適用されます。

詳細についてはセクション 8「ツールオフセット」を参照してください。

プログラミング例

この例では，有効なグラインディングホイールの長さを補償しようとしています。



%_N_DRESS_MPF	
FCTDEF(1,-1000,1000,-$AA_IW[V],1)	機能の定義：
ID=1 DO FTOC(1,$AA_IW[V],3,1)	オンラインツール補償の選択： V 軸 の実際値が多項式 1 への入力値である；結果が，チャンネル 1 内の有効なグラインディングホイールの長さ 3 に，オフセット値として加えられる。
WAITM(1,1,2)	加工チャンネルとの同期化
G1 V-0.05 F0.01 G91	ドレッシングするためのインフィード動作
G1 V-0.05 F0.02	
...	
CANCEL(1)	オンラインツールオフセットの選択解除
...	

10.4.11 位置決めモーション

機能

軸は，パートプログラムに関しては完全に非同期で，シンクロナイズドアクションから位置決めすることができます。イベントに強く依存している周期の順番またはオペレーションについては，シンクロナイズドアクションから位置決め軸をプログラミングすることが望ましいです。シンクロナイズドクションからプログラムされた軸をコマンド軸といいます。

G コード G70/G71/G700/G710 をシンクロナイズドアクションにプログラムできます。それらの G コードは，位置決めタスク用の測定系の定義をするためにシンクロナイズドアクションで使用されます。

参照： /PG/ Section 3 ″Specifying Paths"

/FBSY/ "Starting Command Axes"

測定系は G70/G71/G700/G710 を使用して定義されます。

G 機能をシンクロナイズドアクションにプログラミングすることによって，シンクロナイズドアクション用の INCH/METRIC 評価を，パートプログラムの文脈とは無関係に定義できます。

例 1

N100 R1=0		
N110 G0 X0 Z0		
N120 WAITP(X)		
N130 ID=1 WHENEVER $R==1 DO POS[X]=10		
N140 R1=1		
N150 G71 Z10 F10	Z=10 mm	X=10 mm
N160 G70 Z10 F10	Z=254 mm	X=254 mm
N170 G71 Z10 F10	Z=10 mm	X=10 mm
N180 M30		

例 2

N100 R1=0		
N110 G0 X0 Z0		
N120 WAITP(X)		
N130 ID=1 WHENEVER $R==1 DO G71 POS[X]=10		
N140 R1=1		
N150 G71 Z10 F10	Z=10 mm	X=10 mm
N160 G70 Z10 F10	Z=254 mm	X=10 mm（X は常に 10 mm に位置決めされる）
N170 G71 Z10 F10	Z=10 mm	X=10 mm
N180 M30		

プログラミング例

プログラムされた軸動作のディスエーブル

ブロックの先頭で軸動作をスタートさせたくなければ，シンクロナイズドアクションから適当な時間まで軸に対するオーバライドを 0 にしておくことができます。

WHENEVER $A_IN[1]==0 DO $AA_OVR[W]=0 POS[W]=1500	
	FA=1000
; 位置決め軸は，デジタル入力 1 =0 になるまで停止されます。	

10.4.12 位置決め軸：POS

機能

POS[axis]=value

パートプログラムからのプログラミングとは違って,
位置決め軸動作はパートプログラムの実行に対して
何の影響も及ぼしません。

説明

Axis:	移動させる軸の名称
Value:	移動させる値

プログラミング例

ID=1 EVERY $AA_IM[B]>75 DO POS[U]=100	
	軸 U は，制御原点から相対的に 100 (inch/mm) だけ動かされるか，または移動モードとは無関係に 100 (inch/mm) に位置決めされる。
ID=1 EVERY $AA_IM[B]>75 DO POS[U]=$AA_MW[V]-$AA_IM[W]+13.5	
	; 軸 U は，リアルタイム変数から計算されたパスだけ動かされる。

10.4.13 軸のスタート／停止：MOV

プログラミング

MOV [軸]= 値

説明

Axis:	スタートさせる軸の名称
Value:	移動／停止動作用のスタートコマンド 符号によって動作の方向を指定する。 値のデータタイプは INTEGER である。
Value>0（通常は +1）: 正の方向	正の方向
Value <0（通常は -1）:	負の方向
Value ==0:	軸動作の停止

機能

MOV[軸]= 値を使用すると，エンド位置を指定せずにコマンド軸をスタートさせることができます。

軸は，別の動作コマンドまたは位置決めコマンドによって別の動作がセットされるまで，または停止コマンドによって軸が停止するまで，プログラムされた方向に動かされます。

プログラミング例

... DO MOV[U]=0	軸 U が停止する

（注）インデックス軸が MOV[Axis]=0 によって停止したら，軸は次のインデックス位置で停止します。

10.4.14 軸フィード：FA

プログラミング例

FA[軸] = 送り速度

```
ID=1 EVERY $AA_IM[B]>75 DO POS[U]=100 FA[U]=990
                                                    ; 固定フィードレート値の定義
ID=1 EVERY $AA_IM[B]>75 DO POS[U]=100 FA[U]=$AA_VACTM[W]+100
                                                    ; リアルタイム変数からのフィードレート値の計算
```

10.4.15 ソフトウェアリミットスイッチ

機能

コマンド軸に対して，G25/G26 でプログラムされた作業エリア制限が，設定データ $SA_WORKAREA_PLUS_ENABLE に応じて考慮されます。

パートプログラム中の G 機能 WALIMON/WALIMOF を使用して作業エリア制限をオンにしたりオフにしたりしてもコマンド軸には影響ありません。

10.4.16 軸協調

機能

一般的に軸は，動作ブロック内のパートプログラムから動かされるか，シンクロナイズドアクションから位置決め軸として動かされます。

同じ軸が，パス軸または位置決め軸としてパートプログラムとシンクロナイズドアクションの両方から交互に移動するようになっている場合でも，両方の軸動作の間では協調させた移動が行なわれます。

それに続いてコマンド軸がパートプログラムから移動する場合は，前処理を再編成しなければなりません。このため，パートプログラム処理中に，前処理停止に匹敵する中断がかわるがわる起こります。

プログラミング例

X 軸をパートプログラムまたはシンクロナイズドアクションのどちらかから動かす：

N10 G01 X100 Y200 F1000	パートプログラムにプログラムされた X 軸
...	
N20 ID=1 WHEN $A_IN[1]==1 DO POS[X]=150 FA[X]=200	デジタル入力がセットされていればシンクロナイズドアクションから位置決めをスタートする
...	
CANCEL(1)	シンクロナイズドアクションの選択解除
...	
N100 G01 X240 Y200 F1000	

; X はパス軸になる；デジタル入力が 1 であり，X がシンクロナイズドアクションから位置決めされた場合，動作の前に遅延が起こる

プログラミング例

同一の軸に対する移動コマンドの変更：
ID=1 EVERY $A_IN[1]>=1 DO POS[V]=100 FA[V]=560
; デジタル入力 >= 1 であればシンクロナイズドアクションから位置決めをスタートする
ID=2 EVERY $A_IN[2]>=1 DO POS[V]=$AA_IM[V] FA[V]=790

軸が追従し，2 番目の入力がセットされる。つまり，2 つのシンクロナイズドアクションが同時にアクティブであれば，動作中に V 軸に対するエンド位置およびフィードが連続的に追従される。

10.4.17 実際値のプリセット

機能

PRESETON（軸，値）が実行されると，現在の軸位置は変更されず，新しい値が割当てられます。

(注）

次のような場合に PRESETON をシンクロナイズドアクション内から実行することができます :

- パートプログラムからスタートしたモジュロ回転軸の場合
- シンクロナイズドアクションからスタートした全てのコマンド軸の場合

制限事項 :

PRESETON は，変換に関係する軸に対しては使用できません。

プログラミング例

```
WHEN $AA_IM[a] >= 89.5 DO PRESETON(a4,10.5)
                ; 軸の制御原点を正の軸方向に 10.5 の長さ（インチ または mm）だけオフセットする
```

制限事項

時間をずらさなければ，全く同一の軸をパートプログラムとシンクロナイズドアクションの両方から動かすことはできません。このため，同一の軸が先にシンクロナイズドアクションにプログラムされていた場合，パートプログラムからの軸のプログラミングに遅延が生じます。

同一の軸が交互に使用される場合，2 つの軸動作間の移動は協調させられます。そのため，パートプログラムの実行は中断されます。

10.4.18 主軸モーション

機能

主軸は，パートプログラムに関しては完全に非同期で，シンクロナイズドアクションから位置決めすることができます。イベントに強く依存している周期の順番またはオペレーションについては，このタイプのプログラミングが望ましいです。

プログラミング例

主軸のスタート／停止／位置決め

ID=1 EVERY $A_IN[1]==1 DO M3 S1000	回転の方向および速度の設定
ID=2 EVERY $A_IN[2]==1 DO SPOS=270	主軸の位置決め

動作

同時に有効になっているシンクロナイズドアクションによって，競合するコマンドが主軸に対して発行された場合，最新の主軸コマンドが優先されます.

プログラミング例

回転方向および回転速度の設定／主軸の位置決め

ID=1 EVERY $A_IN[1]==1 DO M3 S300	回転方向および回転速度の設定
ID=2 EVERY $A_IN[2]==1 DO M4 S500	新たな回転方向および新たな回転速度の指定
ID=3 EVERY $A_IN[3]==1 DO S1000	新たな速度の指定
ID=4 EVERY ($A_IN[4]==1) AND ($A_IN[1]==0) DO SPOS=0	主軸の位置決め

10.4.19 カップリングされた動作：TRAILON, TRAILOF

機能

DO TRAILON(following axis, leading axis, coupling factor)	カップリングされた軸動作の起動
DO TRAILOF(following axis, leading axis, leading axis 2)	カップリングされた軸動作の停止

シンクロナイズドアクションからカップリングが起動されると，リーディング軸を動かすことができます。この場合，追従軸は設定された速度まで加速されます。速度が同期化される時のリーディング軸の位置が，カップリングされた軸動作に対するスタート位置です。カップリングされた軸動作の機能性の説明がセクション "Path traversing behavior"（パス移動挙動）に記載されています。

非同期のカップリングされた動作の起動：

... DO TRAILON(FA, LA, CF)

ただし： FA: 追従軸
LA: リーディング軸
CF: カップリング係数

非同期のカップリングされた動作停止：

... DO TRAILOF(FA, LA, LA2)

ただし： FA: 追従軸
LA: リーディング軸
LA2: リーディング軸 2, オプション

プログラミング例

$A_IN[1]==0 DO TRAILON(Y,V,1)	デジタル入力が 1 のとき，第 1 のカップリングされた軸のペアを起動する
$A_IN[2]==0 DO TRAILON(Z,W,-1)	第 2 のカップリングされた軸のペアを起動する
G0 Z10	反対の軸方向での Z 軸および W 軸のインフィード
G0 Y20	同じ軸方向での Y 軸および V 軸のインフィード
...	
G1 Y22 V25	追従軸 "V" の従属した動作および独立した動作を重ねる
...	
TRAILOF(Y,V)	第 1 のカップリングされた軸を停止する
TRAILOF(Z,W)	第 2 のカップリングされた軸を停止する

10.4.20 リーディング値カップリング： LEADON, LEADOF

機能

軸リーディング値カップリングは，制約なしでシンクロナイズドアクションにプログラムすることができます。

軸リーディング値カップリングの起動：

...DO LEADON(FA,LA,NR)

ただし：FA: 追従軸

LA: リーディング軸

NR: 保存されたカーブテーブルの番号

軸リーディング値カップリングの停止：

...DO LEADOF(FA,LA)

ただし：FA: 追従軸

LA: リーディング軸

カップリングする予定の軸は，RELEASE 機能を使用することによってシンクロナイズドアクションアクセスのために解放されます。

例

RELEASE (XKAN)

ID=1 every SR1==1 to LEADON(CACH,XKAN,1)

プログラミング例

即時の切断

切断装置の作業エリアまで入ってきている連続して走る連続材料は，同じ長さでいくつかに切断されます。

X 軸：連続材料が走る方向を表す軸。WCS

X1 軸：連続材料のマシン軸，MCS

Y 軸：連続材料と同期して切断装置が「トラベルする」方向を表す軸

切断ツールの位置決めおよび制御は PLC によって制御されることになっています。PLC インタフェースの信号を，連続材料と切断ツールとの同期の度合いを指定するために評価することができます。

アクション　カップリングの起動，LEADON

カップリングの停止，LEADOF

実際値のセット，PRESETON

%_N_SCHERE1_MPF ;$PATH=/_N_WKS_DIR/_N_DEMOFBE_WPD	
N100 R3=1500	; 切断される部分の長さ
N200 R2=100000 R13=R2/300	
N300 R4=100000	
N400 R6=30	; Y 軸のスタート位置
N500 R1=1	; コンベア軸用のスタート条件
N600 LEADOF(Y,X)	; 既存のカップリングの削除
N700 CTABDEF(Y,X,1,0)	; テーブルの定義
N800 X=30 Y=30	; 値のペア
N900 X=R13 Y=R13	
N1000 X=2*R13 Y=30	
N1100 CTABEND	; テーブルの定義の終了
N1200 PRESETON(X1,0)	; 開始のための PRESET
N1300 Y=R6 G0	; Y 軸のスタート位置，軸は直線
N1400 ID=1 WHENEVER $AA_IW[X]>$R3 DO PESETON(X1,0)	
	; 長さ R3 移動した後の PRESET，次の切断を新たにスタートする
N1500 RELEASE(Y)	
N1800 ID=6 EVERY $AA_IM[X]<10 DO LEADON(Y,X,1)	
	; X < 10 の場合，テーブル 1 を使用して Y を X にカップリングする
N1900 ID=10 EVERY $AA_IM[X]>$R3-30 DO EADOF(Y,X)	
	; 移動された切断距離より 30 以上前であれば，カップリングを停止する
N2000 WAITP(X)	
N2100 ID=7 WHEN $R1==1 DO MOV[X]=1 FA[X]=$R4	; 連続した動作で材料軸を配置する
N2200 M30	

10.4.21 測定

パートプログラムの移動ブロックで使用する場合に比べ，測定機能を思い通りに起動させたり停止させたりできます。

- 残移動量削除なしの軸の測定

MEAWA[axis]=(mode, trigger event_1, ..._4

- 残移動量削除なしの連続した測定

MEAC[axis]=(mode, measurement memory, trigger event_1, ..._4

測定についての詳細な情報が必要な場合は，セクション 5.6" 拡張された測定機能 MEASA, MEAWA, MEAC" を参照してください。

10.4.22 待機マーカのセット／クリア：SETM, CLEARM

機能

SETM(MarkerNumber)	チャンネル用の待機マーカのセット
CLEARM(MarkerNumber)	チャンネル用の待機マーカのクリア

例えばチャンネルを協調させるといった目的で，シンクロナイズドアクションに待機マーカをセットすることができます。

SETM

SETM コマンドは，パートプログラムおよびシンクロナイズドアクションのアクションのパートに書込むことができます。SETM コマンドは，コマンドが実行されるチャンネル用のマーカ番号のマーカをセットします。

CLEARM

CLEARM コマンドは，パートプログラムおよびシンクロナイズドアクションのアクションのパートに書込むことができます。CLEARM コマンドは，コマンドが実行されるチャンネル用のマーカ番号のフラッグ（マーカ）をリセットします。

10.4.23 エラーリアクション

機能

ステータス変数をスキャンすることによって，および適当なアクションをトリガすることによって，誤った応答がシンクロナイズドアクションによってプログラムされる可能性があります。

エラー状態に対する応答例：

- 軸の停止：オーバライド =0
- アラームのセット : SETAL を使用すると，シンクロナイズドアクションからの周期アラームがセットできます。
- 出力の設定
- シンクロナイズドアクションの中で行なうことができる全てのアクション

プログラミング例

```
ID=67 WHENEVER ($AA_IM[X1]-$AA_IM[X2])<4.567 DO $AA_OVR[X2]=0
        ; 軸 X1 と X2 との間の安全距離が小さすぎる場合，軸 X2 を停止する。
ID=67 WHENEVER ($AA_IM[X1]-$AA_IM[X2])<4.567 DO SETAL(61000)
        ; 軸 X1 と X2 との間の安全距離が小さすぎる場合，アラームをセットする。
```

10.5 テクノロジサイクル

機能

シンクロナイズドアクションの中のアクションとして，プログラムを呼出すことができます。これらのプログラムは，シンクロナイズドアクションの中のアクションとして許容できる機能のみから構成されていなければなりません。

このタイプのプログラムをテクノロジサイクルといいます。

テクノロジサイクルは，サブルーチンとして制御装置に保存されます。ユーザに関する限りでは，テクノロジサイクルは，サブルーチンのように呼出すことができます。パラメータの転送はできません。

1 つのチャンネル内で異なったテクノロジサイクル，つまりアクションを並行して処理できます。

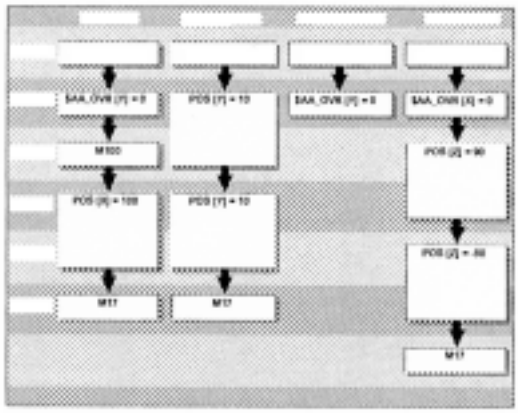
プログラムエンドは M02/M17/M30/RET でプログラムされます。ブロックごとの 1 つの軸動作の最大値をプログラムできます。

アプリケーション

軸プログラムとしてのテクノロジサイクル：テクノロジサイクルはそれぞれ 1 つの軸しか制御しません。この方法で，イベント制御されている同一の補間サイクル中で異なった軸動作をスタートすることができます。この時点でパートプログラムは，極端な場合，シンクロナイズドアクションを管理するためだけにしか使用されなくなります。

プログラミング例

軸プログラムは，デジタル入力をセットすることによってスタートします。



メインプログラム：

ID=1 EVERY $A_IN[1]==1 DO AXIS_X	入力 1 が 1 にセットされていれば，軸プログラム X がスタートする
ID=2 EVERY $A_IN[2]==1 DO AXIS_Y	入力 2 が 1 にセットされていれば，軸プログラム Y がスタートする
ID=3 EVERY $A_IN[3]==1 DO $AA_OVR[Y]=0	入力 3 が 1 にセットされていれば，軸 Y に対するオーバライドが 0 になる
ID=4 EVERY $A_IN[4]==1 DO AXIS_Z	入力 4 が 1 にセットされていれば，軸プログラム Z がスタートする
M30	

テクノロジサイクル AXIS_X:

$AA_OVR[Y]=0
M100
POS[X]=100 FA[X]=300
M17

テクノロジサイクル AXIS_Y:

POS[Y]=10 FA[Y]=200
POS[Y]=-10
M17

テクノロジサイクル AXIS_Z:

$AA_OVR[X]=0
POS[Z]=90 FA[Z]=250
POS[Z]=-90
M17

テクノロジサイクルは，条件が満足されるとただちにスタートします。位置決め軸の場合，実行するためには複数の補間サイクルが必要です。その他の機能 (OVR) は 1 サイクルで実行されます。

テクノロジサイクルでは，ブロックは連続して実行されます。

（注）

相互排他的なアクションが同一の補間サイクル中でコールされた場合，ID 番号がより上の方のシンクロナイズドアクションからコールされたアクションがスタートします。

10.5.1 ロック，ロック解除，リセット：LOCK, UNLOCK, RESET

プログラミング

LOCK (n, n, ...)	テクノロジサイクルのロック，有効なアクションが中断される
UNLOCK (n, n, ...)	テクノロジサイクルのロック解除
RESET (n, n, ...)	テクノロジサイクルのリセット，有効なアクションが中断される
n	シンクロナイズドアクションの識別番号

機能

シンクロナイズドアクション内から，またはテクノロジサイクルから，テクノロジサイクルの実行をロック，ロック解除，またはリセットすることができます。

テクノロジサイクルのロック，LOCK

LOCK を使用して他のシンクロナイズドアクションまたはテクノロジサイクルからテクノロジサイクルをロックできます。

例

```
N100 ID=1 WHENEVER $A_IN[1]==1 DO M130
...
N200 ID=2 WHENEVER $A_IN[2]==1 DO LOCK(1)
```

テクノロジサイクルのロック解除，UNLOCK

ロックされたテクノロジサイクルは，UNLOCK を使用して他のシンクロナイズドアクション／テクノロジサイクルから再びロック解除することができます。

UNLOCK を使用して，ロック解除は現在の位置で続けられます。ロック解除は，中断された位置決めプロシージャにも適用されます。

例

```
N100 ID=1 WHENEVER $A_IN[1]==1 DO M130
...
N200 ID=2 WHENEVER $A_IN[2]==1 DO LOCK(1)
...
N250 ID=3 WHENEVER $A_IN[3]==1 DO UNLOCK(1)
```

テクノロジサイクルのリセット，RESET

RESET を使用して他のシンクロナイズドアクションまたはテクノロジサイクルからテクノロジサイクルをリセットできます。

例

```
N100 ID=1 WHENEVER $A_IN[1]==1 DO M130
...
N200 ID=2 WHENEVER $A_IN[2]==1 DO RESET(1)
```

PLC サイドでのロック

モーダルシンクロナイズドアクションは，ID 番号 n=1 ～ 64 を使用して PLC からインタロックできます。関連する条件は評価されなくなり，関連する機能の実行は NCK 内でロックされます。

シンクロナイズドアクションは全て，PLC インタフェース内で 1 つの信号で無差別にロックすることができます。

（注）

プログラムされたシンクロナイズドアクションは標準通りに有効であり，マシンデータの設定によって上書き／ロックから保護することができます。

アプリケーション：

機械メーカーによって定義されたアクションをエンドユーザが変更することはできません。

10.6 シンクロナイズドアクションのキャンセル : CANCEL

プログラミング

CANCEL(n,n,...)	シンクロナイズドアクションのキャンセル
n	シンクロナイズドアクションの識別番号

説明

識別子 ID(S)=n が付いたモーダルシンクロナイズドアクションは，CANCEL を使用してパートプログラムから直接キャンセルすることしかできません。

例

N100 ID=2 WHENEVER $A_IN[1]==1 DO M130	
...	
N200 CANCEL(2)	番号 2 のシンクロナイズドアクションのキャンセル

（注）

キャンセルされたシンクロナイズドアクションからスタートしている未完了の動作は，プログラムされた通りに完了されます。

10.7 追加条件

- 電源オン
  電源をオンにしてもシンクロナイズドアクションは有効にはなりません。
  しかし，静的シンクロナイズドアクションは，PLC によってスタートする非同期サブルーチン (ASUP) を使用して電源をオンにした時に起動させることができます。

- モード変更
  ボキャブラリワード IDS を使用して起動されたシンクロナイズドアクションは，運転モードの切換え後も有効なままです。
  その他のシンクロナイズドアクションは全て，（例えば軸の位置決めなどの）運転モードの切換え後は無効になり，再位置決めおよび自動モードへの復帰後に再び有効になります。

- リセット
  NC リセットによって，シンクロナイズドアクションからスタートしたアクションは全て停止されます。静的シンクロナイズドアクションはアクティブなままです。静的シンクロナイズドアクションは新たなアクションをスタートすることができます。
  モーダルに有効なシンクロナイズドアクションをリセットするために，シンクロナイズドアクションまたはテクノロジサイクルから RESET コマンドを使用することができます。シンクロナイズドアクションから起動された位置決め軸動作がアクティブなうちにそのシンクロナイズドアクションがリセットされると，位置決め軸動作は中断されます。
  すでに実行された WHEN タイプのシンクロナイズドアクションは，RESET 後に再実行されません。

RESET に続く応答		
シンクロナイズドアクション／テクノロジサイクル	モーダル／ノンモーダル	スタティック (IDS)
	有効なアクションはリセットされ，シンクロナイズドアクションはキャンセルされる	有効なアクションはキャンセルされ，テクノロジサイクルはリセットされる
軸／位置決め主軸	動作はリセットされる	動作はリセットされる
速度制御された主軸	$MA_SPIND_ACTIVE_AFTER_RESET==1: 主軸は有効なままになる $MA_SPIND_ACTIVE_AFTER_RESET==0: 主軸は停止する。	$MA_SPIND_ACTIVE_AFTER_RESET==1: 主軸は有効なままになる $MA_SPIND_ACTIVE_AFTER_RESET==0: 主軸は停止する。
リーディング値カップリング	$MC_RESET_MODE_MASK, Bit13 == 1: リーディング値カップリングは有効なままになる $MC_RESET_MODE_MASK, Bit13 == 0: リーディング値カップリングは分離される	$MC_RESET_MODE_MASK, Bit13 == 1: リーディング値カップリングは有効なままになる $MC_RESET_MODE_MASK, Bit13 == 0: リーディング値カップリングは分離される
測定プロシージャ	シンクロナイズドアクションからスタートした測定はキャンセルされる。	静的シンクロナイズドアクションからスタートした測定はキャンセルされる。

- NC ストップ
  静的シンクロナイズドアクションは，NC ストップが行なわれても有効なままです。静的シンクロナイズドアクションからスタートされた動作はキャンセルされません。
  プログラムに対してローカルであり，有効なブロックに属しているシンクロナイズドアクションは有効なままになり，それらのシンクロナイズドアクションからスタートされた動作は停止します。

- プログラムの終了
  プログラムの終了とシンクロナイズドアクションは互いに影響を及ぼしあいません。
  現在のシンクロナイズドアクションは，プログラム終了後でも完了されます。
  M30 ブロックでアクティブなシンクロナイズドアクションは有効なままです。このような状態にしたくなければ，プログラムが終了する前に CANCEL を使用してキャンセルしてください（前のサブセクションを参照してください）。

プログラムの終了に続く応答		
シンクロナイズドアクション／テクノロジサイクル	モーダルおよびノンモーダルはリセットされる	スタティック (IDS) は有効なままになる
軸／位置決め主軸	軸／主軸が静止状態になるまで M30 は遅延する	動作は続行される
軸／位置決め主軸	動作はリセットされる	動作はリセットされる
速度制御された主軸	プログラムの終了： $MA_SPIND_ACTIVE_AFTER_RESET ==1: 主軸は有効なままになる $MA_SPIND_ACTIVE_AFTER_RESET ==0: 主軸は停止する 主軸は運転モードの切換え後も有効なままになる	主軸は有効なままになる
リーディング値カップリング	$MC_RESET_MODE_MASK, Bit13 == 1: リーディング値カップリングは有効なままになる $MC_RESET_MODE_MASK, Bit13 == 0: リーディング値カップリングは分離される	静的シンクロナイズドアクションからスタートされたカップリングはそのままになる
測定プロシージャ	シンクロナイズドアクションからスタートされた測定はキャンセルされる。	静的シンクロナイズドアクションからスタートされた測定はアクティブなままになる。

- ブロックサーチ
  ブロックサーチ中にみつかったシンクロナイズドアクションは，集められて NC スタートの時に評価されます；必要であれば，関連したアクションがその後スタートされます。
  ブロックサーチの間，静的シンクロナイズドアクションは有効です。
  ブロックサーチ中に，FCTDEF を使用してプログラムされた多項式係数がみつかった場合，それらの多項式係数は直接設定データに書込まれます。

- 非同期サブルーチンによるプログラムの中断
  ASUP スタート：
  モーダルなモーションシンクロナイズドアクションおよび静的なモーションシンクロナイズドアクションはアクティブなままになり，非同期サブルーチンの中でも有効です。
  ASUP 終了：
  Repos によって非同期サブルーチンが再開されなければ，非同期サブルーチンの中で変更されたモーダルなモーションシンクロナイズドアクションおよび静的モーションシンクロナイズドアクションはメインプログラム内で有効なままになります。
- 再位置決め
  再位置決め REPOS によって，中断されたブロックで有効になっていたシンクロナイズドアクションが再起動されます。
  非同期サブルーチンから変更されたモーダルシンクロナイズドアクションは，REPOS の後，残りのブロックが実行される時には有効にはなりません。
  FCTDEF を使用してプログラムされた多項式係数は，非同期サブルーチンおよび REPOS の影響を受けません。それらの多項式係数は，どこでプログラムされたかに関わらず，REPOS の実行後も，非同期サブルーチンおよびメインプログラムでいつでも使用することができます。

- CANCEL を使用した選択解除
  有効なシンクロナイズドアクションが CANCEL を使用して選択解除されても，有効になっているアクションには影響はありません。
  位置決め動作はプログラミングに従って終了します。
  CANCEL コマンドは，モーダルに有効なシンクロナイズドアクションまたは静的に有効なシンクロナイズドアクションを中断するために使用されます。
  シンクロナイズドアクションから起動された位置決め軸動作が有効なうちにそのシンクロナイズドアクションがキャンセルされると，位置決め軸動作は中断されます。このようにしたくなければ，CANCEL コマンドの前に軸の残移動量削除を使用して軸動作を減速することができます：

例

ID=17 EVERY $A_IN[3]==1 DO POS[X]=15 FA[X]=1500	; 位置決め軸動作のスタート
...	
WHEN ... DO DELDTG(X)	; 位置決め軸動作の終了
CANCEL(1)	

11 揺動

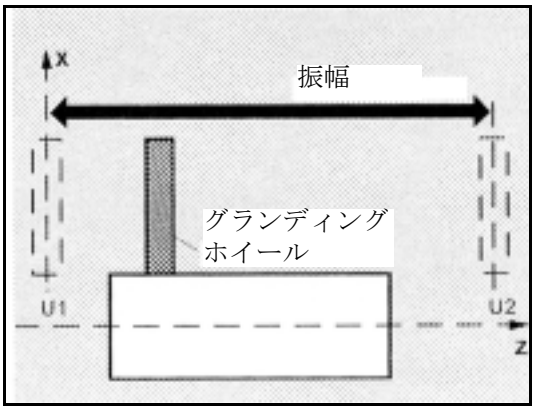
11.1 非同期揺動

コマンドの説明

OSP1[軸]= OSP2[軸]=	逆転点 1 の位置 逆転点 2 の位置
OST1[軸]= OST2[軸]=	逆転点での停止時間（単位：秒）
FA[軸]=	揺動軸に対するフィード
OSCTRL[軸]=	（オプションの設定，リセット）
OSNSC[軸]=	スパークアウトストロークの数
OSE[軸]=	エンド位置
OS[軸]=	1 = 揺動の起動；0 = 揺動の停止

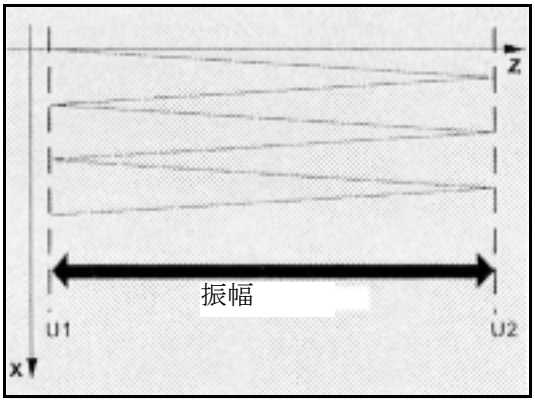
機能

揺動軸は，揺動動作が終了するまで，逆転点 1 と 2 の間を定義された送り速度で往復します。



その他の軸は，揺動動作中に必要に応じて補間できます。

一定のインフィードを達成するためにパス動作または位置決め軸を使用することができますが，発揺動作とインフィード動作とは無関係です。



揺動軸

揺動軸に対して次のことが適用されます：

- どのような軸でも揺動軸として使用できる。
- 異なった揺動軸が同時に有効になることが可能（最大数：位置決め軸の数）。
- 直線補間 G1 は揺動軸に対して常に有効であり，プログラム内で現在有効な G コマンドとは無関係。

揺動軸は次のことができます：

- 動的変換用の入力軸としての働きをする
- ガントリ軸や複合動作軸のガイド軸としての働きをする
- 以下の移動を行う。
  - ジャーク制限無し (BRISK)
  - ジャーク制限有り (SOFT)
  - 膝型加速曲線に沿って（位置決め軸として）移動させる。

揺動逆転点

揺動位置が下記のように定義される時，現在のオフセットを考慮する必要があります：

- 絶対指定

OSP1[Z]= 値

逆転点の位置 = オフセットの合計 + プログラムされた値

- 相対指定

OSP1[Z]=IC（値）

逆転点の位置 = 逆転点 1 + プログラムされた値

例

N10 OSP1[Z]=100 OSP2[Z]=110

.

.

N40 OSP1[Z]=IC(3)

非同期揺動の特性

- 非同期揺動は，軸別に，ブロックリミットを超えてアクティブです。
- ブロック向けの揺動動作の起動はパートプログラムによって保証されます。
- 複数の軸の合同補間および振幅の重畳はできません。

設定データ

非同期揺動に必要な設定データをパートプログラムに設定できます。

設定データがプログラムに直接書込まれると，変更は前処理中に有効になります。STOPRE によって同期応答が行なわれます。

例

反転位置をオンライン変更した場合の揺動

$SA_OSCILL_REVERSE_POS1[Z]=-10	
$SA_OSCILL_REVERSE_POS2[Z]=10	
G0 X0 Z0	
WAITP(Z)	
ID=1 WHENEVER $AA_IM[Z] < $$AA_OSCILL_REVERSE_POS1[Z] DO $AA_OVR[X]=0	
ID=2 WHENEVER $AA_IM[Z] < $$AA_OSCILL_REVERSE_POS2[Z] DO $AA_OVR[X]=0	
	; 揺動軸の実際値が逆転点を超えると，インフィード軸が停止する。
OS[Z]=1 FA[X]=1000 POS[X]=40	; 揺動オン
OS[Z]=0	; 揺動オフ
M30	

個々の機能についての注

下記のアドレスによって，非同期揺動をパートプログラムから起動および制御することができます。

プログラムされた値は，対応する設定データにブロックと同期させて入力され，メインラン中は再び変更されるまで有効なままになります。

揺動の起動と停止：OS

OS[軸] = 1: 起動

OS[軸] = 0: 停止

WAITP（軸）：

- ジオメトリ軸を使用して揺動が行なわれることになっている場合，WAITP を使用してこの軸を揺動イネーブルにしなければなりません。
- 揺動が終了したとき，このコマンドを使用して，通常の使い方をするために揺動軸を再び位置決め軸として入力します。

逆転点での停止時間：
OST1, OST2

ホールド時間	逆転点でのイグザクト・ストップエリア内の動作
-2	イグザクト・ストップに到達するのを待つことなく補間が続けられる
-1	イグザクト・ストップ（粗）に到達するまで待つ
0	イグザクト・ストップ（微）に到達するまで待つ
>0	イグザクト・ストップ（微）に到達するまで待った後、停止時間の間待つ

停止時間用の単位は，G4 でプログラムされた停止時間の単位と同じです。

送りの設定 FA

送り速度とは，定義された位置決め軸の送り速度のことです。

送り速度が定義されていない場合は，マシンデータに保存された値が適用されます。

動作シーケンスの定義：OSCTRL

セットおよびリセットオプションを使用して，動作についての制御の設定を行ないます。

リセットオプション

これらのオプションは無効になります（すでにセットオプションで有効の場合のみ)。

セットオプション

これらのオプションは切換えられます。OSE（エンド位置）がプログラムされると，オプション 4 が暗示的に起動されます。

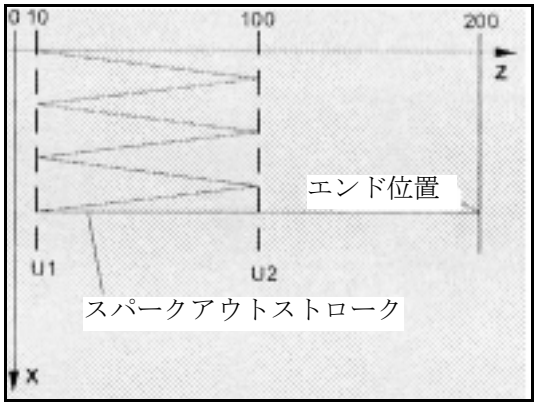
オプション値	意味
0	揺動が停止されると，次の逆転点で停止する（デフォルト） 値 1 および 2 をリセットすることによってのみ可能
1	揺動が停止されると，逆転点 1 で停止する
2	揺動が停止されると，逆転点 2 で停止する
3	揺動が停止されても，スパークアウトストロークがプログラムされていなければ，逆転点にアプローチしない
4	スパークアウトの後、エンド位置にアプローチする
8	移動距離の削除によって揺動動作がキャンセルされると： スパークアウトストロークを実行し，エンド位置にアプローチする（適用される場合）
16	移動距離の削除によって揺動動作がキャンセルされると： 停止される場合と同じく反転位置にアプローチする
32	新しいフィードは次の逆転点の後にしか有効にならない
64	FA = 0: パスオーバーレイが有効 FA 0: 速度オーバーレイが有効
128	回転軸 DC（最短パス）用

プラスの符号を使用すると複数のオプションが追加できます。

例

OSCTRL[Z] = (1+4,16+32+64)

プログラミング例



揺動軸 Z は 10 から 100 の間で往復するようになっています。

逆転点 1 にはイグザクト・ストップ（微）を使用してアプローチし，逆転点 2 にはイグザクト・ストップ（粗）使用してアプローチしてください。揺動軸については送り速度 250 で加工が行なわれます。加工オペレーションの最後で 3 回のスパークアウトストロークが実行され，揺動軸がエンド位置 200 にアプローチしなければなりません。

インフィード軸用の送りは 1 で，X 方向のインフィードの終点位置は 15 です。

WAITP(X,Y,Z)	開始位置
G0 X100 Y100 Z100	位置決め軸オペレーションでの切換え
N40 WAITP(X,Z)	
N50 OSP1[Z]=10 OSP2[Z]=100 ->	逆転点 1，逆転点 2
-> OSE[Z]=200 ->	エンド位置
-> OST1[Z]=0 OST2[Z]=-1 ->	U1 での停止時間：イグザクト・ストップ（微） U2 での停止時間：イグザクト・ストップ（粗）
-> FA[Z]=250 FA[X]=1 ->	揺動軸用，インフィード軸用のフィード
-> OSCTRL[Z]=(4,0) ->	設定オプション
-> OSNSC[Z]=3 ->	3 回のスパークアウトストローク
N60 OS[Z]=1	揺動のスタート
N70 WHEN $A_IN[3]==TRUE -> -> DO DELDTG(X)	移動距離削除
N80 POS[X]=15	X 軸の開始位置
N90 POS[X]=50	
N100 OS[Z]=0	揺動の停止
M30	

-> 単一のブロック内にプログラムできます。

11.2 シンクロナイズドアクションによって制御された揺動

プログラミング

1. 揺動用のパラメータの定義
2. モーションシンクロナイズドアクションの定義
3. 軸の割当て，インフィードの定義

揺動用のパラメータ

OSP1[揺動軸]=	逆転点 1 の位置
OSP2[揺動軸]=	逆転点 2 の位置
OST1[揺動軸]=	逆転点 1 での停止時間（単位：秒）
OST2[揺動軸]=	逆転点 2 での停止時間（単位：秒）
FA[揺動軸]=	揺動軸用のフィード
OSCTRL[揺動軸]=	セットオプションまたはリセットオプション
OSNSC[揺動軸]=	スパークアウトストロークの回数
OSE[揺動軸]=	エンド位置
WAITP(揺動軸)	揺動用に軸をイネーブルする

軸の割当て，インフィード

OSCILL[OscillationAxis] = (InfeedAxis1, InfeedAxis2, InfeedAxis3)

POSP[InfeedAxis] = (Endpos, Partial length, Mode)

OSCILL	インフィード軸または揺動軸用の軸の割当て
POSP	完全インフィードおよび部分インフィードの定義（セクション 3 を参照）
Endpos	全ての部分インフィードが移動した後のインフィード軸用のエンド位置
Partial length	逆転点／逆転エリアでの部分インフィードの長さ
Mode	完全インフィードをいくつかの部分インフィードに分割する 0 = 2 つの残りのステップを等しいサイズに分割する（初期設定）; 1 = 全ての部分インフィードを等しいサイズに分割する

モーションシンクロナイズドアクション

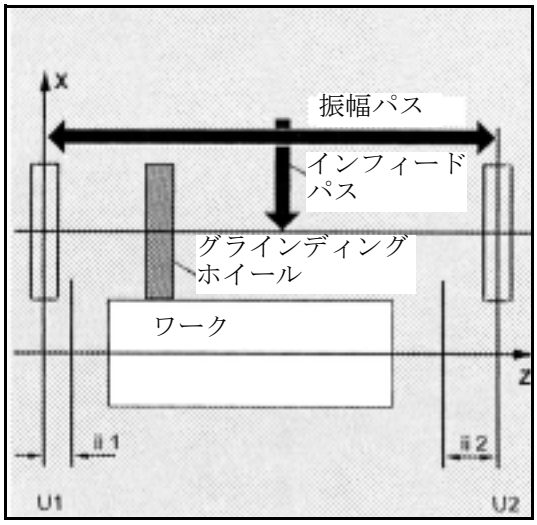
WHEN ... DO ...	もし ... なら ... する
WHENEVER ... DO	... の場合は常に ... する

シンクロナイズドアクションを介した揺動の制御

この揺動モードでは，インフィードモーションは，逆転点または定義された逆転エリア内でしか実行できません。

必要に応じて，揺動動作を：

- 続行する，または
- インフィードの実行が終了するまで停止することができます。



動作

1. 揺動パラメータの定義

揺動用のパラメータは，インフィードおよび揺動軸の割当てとインフィードの定義（「非同期揺動」を参照）を含む動作ブロックの前に定義して下さい。

2. モーションシンクロナイズドアクションの定義

   下記の同期条件を定義できます：

- 揺動軸が逆転エリア内 (ii1, ii2) または逆転点 (U1, U2) に到達するまでインフィードを行わない。
- 逆転点でのインフィードの間，揺動動作を停止する。
- 部分インフィードが完了状態での揺動動作。
- 次の部分インフィードのスタートを定義する。

3. 部分インフィードおよび完全インフィードと同様の揺動軸およびインフィード軸割当て。

揺動およびインフィード軸の割当て：

OSCILL

OSCILL[OscillationAxis] = (InfeedAxis1, InfeedAxis2, InfeedAxis3)

軸の割当ておよび揺動動作のスタートは，OSCILL コマンドで定義されます。

1 つの揺動軸に対してインフィード軸を 3 つまで割当てることができます。

揺動がスタートする前に，軸の挙動についての同期条件を定義しなければなりません。

インフィードの定義：POSP

POSP[InfeedAxis] = (EndPosition, Part, Mode)

POSP コマンドによって下記のことを制御装置に知らせます：

- 完全インフィード(エンド位置を基準にした)
- 逆転点，または逆転エリア内での部分インフィードの長さ
- エンド位置に到達した時の（モードを基準にした）部分インフィード応答

モード = 0	残りの 2 つの部分インフィードについての移動先までの移動距離を 2 つの同じサイズのステップに分割する。(初期設定)
モード = 1	部分インフィードは全て同じサイズになる。そのサイズは完全インフィードから計算される。

シンクロナイズドアクション

この章の最後に示されているシンクロナイズドアクションが通常の揺動に使用されます。

個々のタスクに対するソリューション例が示されています。それらのタスクは，ユーザごとの揺動動作を作成するためのモジュールとして使用できます。

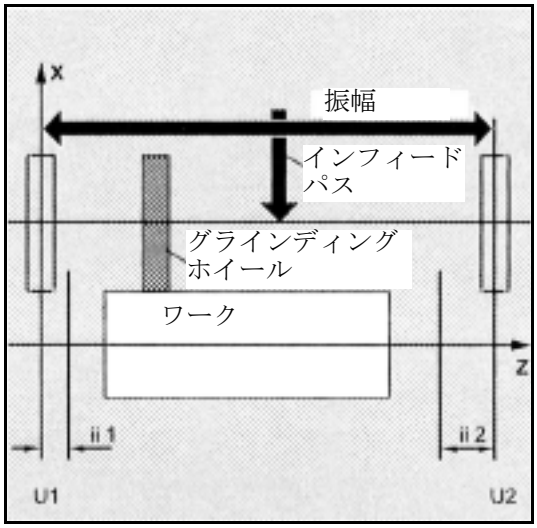
それぞれのケースに合わせて，同期条件を変えてプログラムすることができます。

ボキャブラリワード

WHEN ... DO ...	もし ... なら ... する
WHENEVER ... DO	... の場合は常に ... する

下記の機能を実行するのに，以降で詳しく述べられている言語リソースを使用することができます：

1. 逆転点でのインフィード
2. 逆転エリアでのインフィード
3. 両方の逆転点でのインフィード
4. 逆転点での揺動動作の停止
5. 揺動動作の再スタート
6. 部分インフィードのスタートを早くしすぎない



下記は，ここに挙げられているシンクロナイズドアクションの例全てに対する仮定条件です：

- 逆転点 1 < 逆転点 2
- Z = 揺動軸
- X = インフィード軸

ここで述べられている以外のシンクロナイズドアクションについての情報がセクション 10.3 に記載されています。

逆転エリア内のインフィード

インフィードモーションは，逆転点に到達する前に逆転エリア内でスタートしなければなりません。

これらのシンクロナイズドアクションは，揺動軸が逆転エリア内に入るまでインフィード動作を禁止します。

下記の命令は，上記の仮定条件を前提として使用されます：

逆転エリア 1:

```
WHENEVER
$AA_IM[Z]>$SA_OSCILL_REVERSE_POS1[Z]+ii1
DO $AA_OVR[X]=0
```

MCS 内の揺動軸の現在位置が、逆転エリア 1 のスタートを超えているときはいつでもインフィード軸の軸オーバライドを 0% に設定する。

逆転エリア 2 :

```
WHENEVER $AA_IM[Z]
<$SA_OSCILL_REVERSE_POS2[Z]+ii2 DO
$AA_OVR[X]=0
```

MCS 内の揺動軸の現在位置が逆転エリア 2 のスタートより小さいときはいつでもインフィード軸の軸オーバライドを 0% に設定する。

逆転点でのインフィード

揺動軸が逆転点に到達していなければ，インフィード軸ではどんな動作も行なわれません。

下記の命令は，上記の仮定条件を前提として使用されます：

逆転点 1：

```
WHENEVER
$AA_IM[Z]<>$SA_OSCILL_REVERSE_POS1[Z] DO
$AA_OVR[X]=0 ->
-> $AA_OVR[Z]=100
```

MCS 内の揺動軸 Z の現在位置が逆転点 1 の位置を超えているかその位置より小さいときはいつでもインフィード軸 X の軸オーバライドを 0% に設定し，揺動軸 Z の軸オーバライドを 100% に設定する。

逆転点 2：

逆転点 2 に対して：

```
WHENEVER
$AA_IM[Z]<>$SA_OSCILL_REVERSE_POS2[Z] DO
$AA_OVR[X]=0 ->
-> $AA_OVR[Z]=100
```

MCS 内の揺動軸 Z の現在位置が逆転点 2 の位置を超えているかその位置より小さいときはいつでもインフィード軸 X の軸オーバライドを 0% に設定し，揺動軸 Z の軸オーバライドを 100% に設定する。

逆転点での揺動動作の停止

揺動軸が逆転点で停止すると同時にインフィードモーションがスタートします。

インフィード動作が完了すると，揺動動作が続行されます。

このシンクロナイズドアクションは，有効なままになっている前のシンクロナイズドアクションによってインフィード動作が停止している場合に，このインフィード動作をスタートさせるために使用することもできます。

下記の命令は，上記の仮定条件を前提として使用されます：

逆転点 1：

```
WHENEVER
$SA_IM[Z]==$SA_OSCILL_REVERSE_POS1[Z]DO
$AA_OVR[Z]=0 ->
-> $AA_OVR[X] = 100
```

MCS 内の揺動軸の現在位置が逆転点 1 の位置と等しいときはいつでも揺動軸の軸オーバライドを 0% に設定し，インフィード軸の軸オーバライドを 100% に設定する。

逆転点 2：

```
WHENEVER $SA_IM[Z]
==$SA_OSCILL_REVERSE_POS2[Z]DO
$AA_OVR[Z]= 0 ->
-> $AA_OVR[X]=100
```

MCS 内の揺動軸の現在位置が逆転点 2 の位置と等しいときはいつでも揺動軸の軸オーバライドを 0% に設定し，インフィード軸の軸オーバライドを 100% に設定する。

逆転点のオンライン評価

$$ を付けてコード化されたメインラン変数が比較の右側にあれば，2 つの変数は IPO サイクルで連続的に評価され，お互いに比較されます。

ここで述べられている以外の情報については，セクション「モーションシンクロナイズドアクション」を参照してください。

揺動動作の再スタート

このシンクロナイズドアクションは，部分インフィード動作が完了したときに揺動動作を続行するために使用されます。

下記の命令は，上記の仮定条件を前提として使用されます：

WHENEVER $AA_DTEPW[X]==0 DO $AA_OVR[Z]=100

WCS 内のインフィード軸 X の部分インフィード用の移動距離が 0 であるときはいつでも揺動軸の軸オーバライドを 100% に設定する。

次の部分インフィード

インフィードが完了するとき，次の部分インフィードの早すぎるスタートは禁止しなければなりません。

このためにチャンネル別マーカ（$AC_MARKER[インデックス]）が使用されます。チャンネル別マーカは部分インフィードの最後（部分移動距離 0）でイネーブルされ，軸が逆転エリアから離れると削除されます。

シンクロナイズドアクションはそのとき次のインフィード動作を禁止するために使用されます。

与えられた仮定条件に基づいて，下記の命令が逆転点 1 に適用されます：

1. **マーカの設定**：

WHENEVER $AA_DTEPW[X] == 0 DO
$AC_MARKER[1]=1

WCS 内のインフィード軸 X 上の部分インフィード用の移動距離が 0 であるときはいつでもインデックス 1 が付いたマーカを 1 に設定する。

2. **マーカのクリア**

WHENEVER
$AA_IM[Z]<>$SA_OSCILL_REVERSE_POS1[Z] D0
$AC_MARKER[1]=0

MCS 内の揺動軸 Z の現在位置が逆転点 1 の位置を超えているかその位置より小さいときはいつでもマーカ 1 を 0 に設定する。

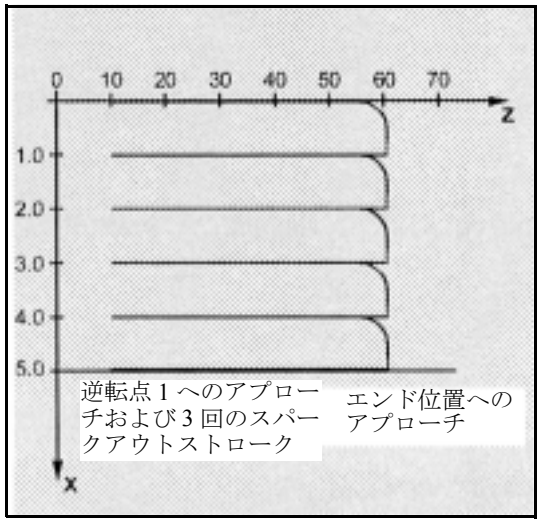
3. **インフィードの禁止**

WHENEVER $AC_MARKER[1]==1 DO
$AA_OVR[X]=0

マーカ 1 が 1 であるときはいつでもインフィード軸の軸オーバライドを 0% に設定する

プログラミング例

逆転点 1 でインフィードが行なわれないようにします。逆転点 2 では，インフィードは逆転点 2 より ii2 の距離だけ前でスタートし，揺動軸は逆転点で部分インフィードが終了するまで待たないようにします。軸 Z は揺動軸で，軸 X はインフィード軸です。



プログラム例

1. 揺動用のパラメータの定義

DEF INT ii2	逆転エリア 2 についての変数の定義
OSP1[Z]=10 OSP2[Z]=60	逆転点 1 および 2 の定義
OST1[Z]=0 OST2[Z]=0	逆転点 1：イグザクト・ストップ（微）
	逆転点 2：イグザクト・ストップ（微）
FA[Z]=150 FA[X]=0.5	揺動軸 Z のフィードレート，インフィード軸 X のフィードレート
OSCTRL[Z]=(2+8+16,1)	逆転点 2 での揺動動作の停止；DTG の削除後，スパークアウトし，エンド位置にアプローチする；DTG の削除後，反転位置にアプローチする
OSNC[Z]=3	3 回のスパークアウトストローク
OSE[Z]=70	エンド位置 = 70
ii2=2	逆転エリアの設定
WAITP(Z)	Z 軸についての揺動イネーブル

2. モーションシンクロナイズドアクション

WHENEVER $AA_IM[Z]<$SA_OSCILL_REVERSE_POS2[Z]-ii2 DO -> -> $AA_OVR[X]=0 $AC_MARKER[0]=0 MCS 内の揺動軸 Z の現在位置が逆転エリア 2 のスタートより小さいときはいつでもインフィード軸 X の軸オーバライドを 0% に設定し，インデックス 0 が付いたマーカを値 0 に設定する。
WHENEVER $AA_IM[Z]>=$SA_OSCILL_REVERSE_POS2[Z] DO $AA_OVR[Z]=0 MCS 内の揺動軸 Z の現在位置が逆転点 2 の位置を超えているか等しいときはいつでも揺動軸 Z の軸オーバライドを 0% に設定する。
WHENEVER $AA_DTEPW[X]==0 DO $AC_MARKER[0]=1 部分インフィードの移動距離が 0 であるときはいつでもインデックス 0 が付いたマーカを値 1 に設定する。
WHENEVER $AC_MARKER[0]==1 DO $AA_OVR[X]=0 $AA_OVR[Z]=100 インデックス 0 が付いたマーカが 1 であるときはいつでも早すぎるインフィード（揺動軸 Z がまだ逆転エリア 2 を出ていないのにインフィード軸 X が新たにインフィードをしようとする）を禁止するために，インフィード軸 X の軸オーバライドを 0% に設定し，揺動軸 Z の軸オーバライドを 100% に設定する（これにより 2 番目のシンクロナイズドアクションがキャンセルされる)。

-> これらは独立したブロックにプログラムしなけれ
ばならない。

3. 揺動のスタート

OSCILL[Z]=(X) POSP[X]=(5,1,1)	軸のスタート 軸 X を揺動軸 Z に対するインフィード軸として割当てる。 軸 X は 1 回のステップでエンド位置 5 まで移動する。
M30	プログラムエンド。

12 追加機能

12.1 軸機能 AXNAME, SPI, ISAXIS

プログラミング

AXNAME("TRANSVERSE AXIS")

AX[AXNAME("String")]

SPI(spindle number)

ISAXIS(geometry axis number)

コマンドの説明

AXNAME	入力文字列を軸識別子に変換する。
	入力文字列には有効な軸名が含まれていなければならない。
SPI	主軸番号を軸識別子に変換する。転送されたパラメータには有効な主軸番号が含まれていなければならない。
AX	可変軸識別子
ISAXIS	指定されたジオメトリ軸が存在するかどうかをチェックする。

機能

AXNAME は，例えば軸の名前がわからないときに一般的に適用できるサイクルを作成するために使用します（セクション 13.10. "String functions" も参照してください)。

SPI は，例えば軸機能が主軸同期などの主軸に使用されるときに使用します。

ISAXIS は，特定のジオメトリ軸が存在し，従ってそれに続く $P_AXNX コールがエラーメッセージによって強制終了されないことを確認するためにユニバーサルサイクルの中で使用します。

プログラミング例

向合った軸として定義された軸の移動。

OVRA[AXNAME("Transverse axis")]=10	トランズバース
AX[AXNAME("Transverse axis")]=50.2	トランズバース軸の最終位置
OVRA[SPI(1)]=70	主軸 1 用のオーバライド
IF ISAXIS(1) == FALSE GOTOF CONTINUE	横座標は存在しているか？
AX[$P_AXN1]=100	横座標の移動
CONTINUE:	

12.2 主軸同期

プログラミング

COUPDEF (FS,LS,SRFS,SRLS, block change
beh., coupling)
COUPDEL (FS,LS)
COUPRES (FS,LS)
COUPON (FS,LS,PSFS)
COUPOF (FS,LS,POSFS,POSLS)
WAITC (FS, block change beh., FS,
block change beh.)

コマンドの説明

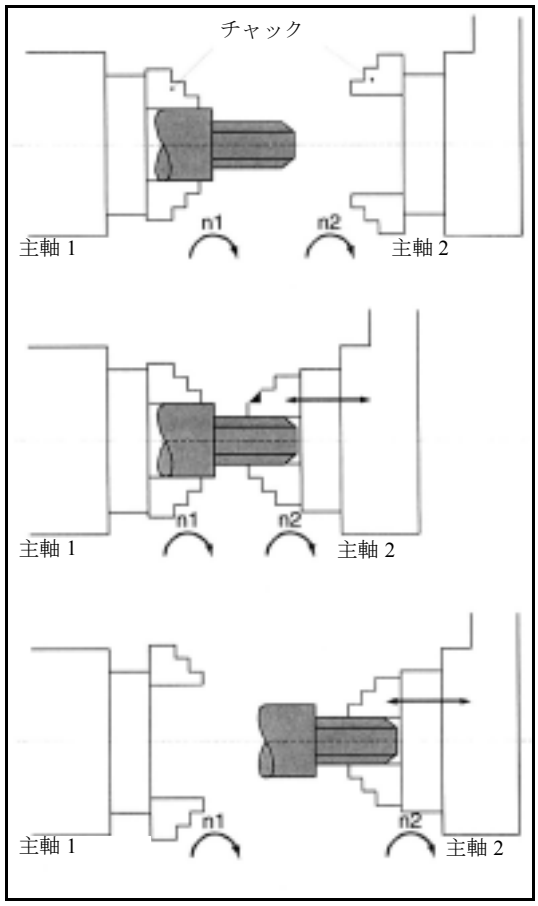
COUPDEF	ユーザカップリングの定義／変更
COUPON	カップリングの起動
COUPOF	カップリングの停止
COUPRES	カップリングパラメータのリセット
COUPDEL	ユーザ定義カップリングの削除
WAITC	同期条件が満足されるのを待つ

パラメータの説明

FS, LS	追従主軸およびリーディング主軸の名称；主軸番号で指定される：例 S2
SR_{FS}, SR_{LS}	追従主軸およびリーディング主軸用の速度比パラメータ 初期設定 = 1.0；分母の指定はオプション
Block change Behavior: ・"NOC" ・"FINE" ・"COARSE" ・"IPOSTOP"	ブロック変更の方法： 即時（デフォルト） 「同期ラン微」に応じて 「同期ラン粗」に応じて IPOSTOP に応じて（つまりセットポイント同期ランの後で）
Coupling ・"DV" ・"AV"	カップリングタイプ：FS と LS との間のカップリング セットポイントカップリング（デフォルト） 実際値カップリング
PS_{FS}	リーディング主軸と追従主軸との間の角度オフセット
POS_{FS}, POS_{LS}	追従主軸およびリーディング主軸のカップリング停止位置

機能

同期モードでは，リーディング主軸 (LS) と追従主軸 (FS) が存在します。それらは同期主軸ペアと呼ばれます。追従主軸はカップリングが有効（同期モード）なときに，パラメータ内で指定された機能的関係に従ってリーディング主軸の動作を追従します。

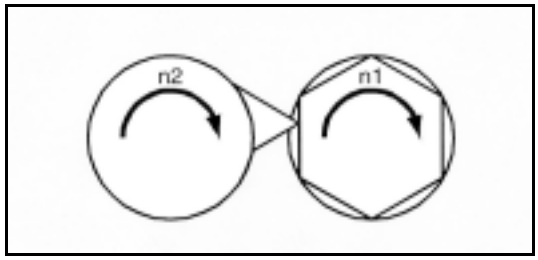


この機能によって，旋盤が，例えば最終加工のために主軸 1 から主軸 2 へ即時にワークの移送を行なうことがイネーブルされます。このため，例えば再チェックすることによって発生するダウンタイムをなくすことができます。

ワークの移送は，下記を使用して行うことができます：

- 速度同期 ($n_{FS} = n_{LS}$)
- 位置同期 （$\varphi_{FS} = \varphi_{LS}$）
- 角度オフセットを使用した位置同期
  （$\varphi_{FS} = \varphi_{LS} + \Delta\varphi$)

マルチエッジ加工（多角形旋削）用のメイン主軸と「ツールピンドル」との間で速度比を指定することもできます。



同期主軸ペアは，チャンネル別マシンデータを使用して各マシンについて恒久的に定義することもできるし，あるいは CNC パートプログラムの中でユーザが定義することもできます。

NC チャンネルごとに最高 2 つまでの同期主軸ペアを同時に操作することができます。

動作

同期主軸ペアの定義：オプション

カップリングの固定定義：

リーディング主軸および追従主軸はマシンデータの中で定義されます。

このカップリングを使用する場合は，LS および FS 用に定義されたマシン軸を NC パートプログラムから変更することはできません。それにもかかわらず，カップリングは，COUPDEF を使用して NC パートプログラム内でパラメータ化することができます（ライトプロテクトが有効でない場合）。

ユーザ定義カップリング：

NC パートプログラムで新たなカップリングを作成したり，既存のカップリングを変更したりするのに言語インストラクション COUPDEF を使用することができます。新たなカップリング関係を定義しようとする場合は，既存のユーザ定義カップリングは COUPDEL を使用して削除しなければなりません。

新たなカップリングの定義 COUPDEF

以下のパラグラフでは，予め定義されたサブルーチン用のパラメータを定義しています：

COUPDEF (FS,LS,SR_{FS} ,SR_{LS} , block change beh.,

coupling)

追従主軸およびリーディング主軸：FS および LS

軸名 FS および LS は，カップリングを個別に識別するのに使用されます。

それらは COUP 命令ごとにプログラムされなければなりません。それ以外のカップリングパラメータは，変更する場合しか定義する必要はありません（モーダル範囲）。

例

N... COUPDEF(S2,S1,SR_{FS},SR_{LS})

意味：

S2 = 追従主軸，S1 = リーディング主軸

追従主軸の位置決め：オプション

主軸同期カップリングが有効な時，追従主軸は，マスタ主軸によって開始されたモーションとは無関係に ± 180° の範囲内で位置決めすることもできます。

位置決め SPOS

追従主軸は，SPOS=.... によって補間することができます。

その他の SPOS に関する情報については，プログラミング編 基本説明書を参照してください。

例

N30 SPOS[2]=IC(-90)

FA, ACC, OVRA :

速度，加速

追従主軸用の位置決め速度および加速レートは，FA[SPI(Sn)] または FA[Sn]，ACC[SPI(Sn)] または ACC[Sn]，OVRA[SPI(n)] または OVRA[Sn] を使用してプログラムすることができます（プログラミング編 基本説明書を参照してください）。"n" は主軸番号 1 ～ n の略です。

プログラム可能なブロック変更 WAITC

WAITC は，例えばカップリングパラメータまたは位置決めオペレーションへの変更の後，プログラムを続行するために，様々な同期条件（粗，微，IPOSTOP）とともにブロック変更挙動を定義するために使用できます。

WAITC は，適当な同期条件が満足されるまで新たなブロックの挿入を遅らせます。それによって同期ステータスがより迅速に処理できるようになります。

同期条件が指定されない場合，関連するカップリング用にプログラム／構成されたブロック変更挙動が適用されます。

例

N200 WAITC

同期条件が指定されていない有効なスレーブ主軸全てについてこれらの同期条件が満足されるまで待つ。

N300 WAITC(S2,"FINE",S4,"COARSE")

スレーブ主軸 S2 および S4 について指定された「COARSE」同期条件が満足されるまで待つ。

速度比

速度比は，FS 用のパラメータ（分子）と LS 用のパラメータ（分母）で定義されます。

オプション：

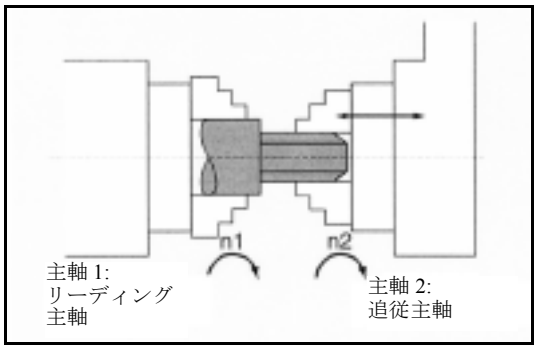
- 追従主軸とリーディング主軸の同じ速度で回転する（$n_{FS} = n_{LS}$; SR_T 正）。
- LS と FS を同一方向に回転させるか反対方向（SR_T 負）に回転させるか。
- 追従主軸とリーディング主軸の異なった速度での回転する ($n_{FS} = SR_T$ ；n_{LS} ; SR_T ；1)。
  アプリケーション：多面旋削

例

N... COUPDEF(S2, S1, 1.0, 4.0)

意味：

追従主軸 S2 およびリーディング主軸 S1 が速度比 0.25 で回転する。



- 分子をプログラムしなければなりません。分子がプログラムされない場合は，デフォルトで "1" になります。
- カップリングが有効ならば，速度比は即時に変更できます。

ブロック変更挙動

どこでブロック変更を行なうか指定するために，カップリングの定義中に，下記のオプションを選択することができます：

"NOC"	即時（デフォルト）の場合
"FINE"	「同期微」の場合
"COARSE"	「同期粗」の場合
"IPOSTOP"	IPOSTOP（つまりセットポイントサイドでの同期後）の場合

ブロック変更の方法を指定する時は，太字でタイプされた文字だけを指定すれば十分です。

ブロック変更の方法はモーダルです。

カップリングタイプ

"DV"	FS と LS との間のセットポイントカップリング（デフォルト）
"AV"	FS と LS との間の実際値カップリング

カップリングタイプはモーダルです。

注意

カップリングタイプは，カップリングが停止している場合しか変更できません！

同期モードの起動

- LS と FS との間の任意の基準角度でカップリングをできるだけ迅速に起動する：

N ... COUPON (S2, S1)

- 角度オフセット POS_{FS} を使用した起動
  プロファイルされたワークについての位置同期カップリング。
  POS_{FS} は，リーディング主軸の正の回転方向の 0° の位置を基準にします。

値の範囲 POS_{FS} : 0° ... 359.999° :

COUPON (S2,S1,30)

カップリングがすでに有効になっている場合でも，この方法を使用して角度オフセットを変更することができます。

同期モードの停止 COUPOF

3 つの方法が可能です：

- カップリングの迅速な起動および即時のブロック変更イネーブル用：

COUPOF (S2,S1)

- 停止位置を通過した後；停止位置
  POS_{FS}（場合によっては POS_{LS} も）を通過し終わるまでブロック変更はイネーブルされません。

値の範囲 0° ... 359.999° :

COUPOF (S2,S1,150)
COUPOF (S2,S1,150,30)

カップリングを削除する COUPDEL

ユーザ定義主軸同期

新たなカップリング関係を定義しようとしているときにユーザ構成可能なカップリング（1 または 2）がすでに定義されている場合は，既存のカップリングを削除しなければなりません。

N ... COUPON (S2,S1)

SPI(2) = 追従主軸，SPI(1) = リーディング主軸

カップリングは，まず停止 (COUPOF) させてからしか削除できません。

恒久的に構成されているカップリングは，
COUPDEL によって削除することはできません。

カップリングパラメータのリセット，COUPRES

言語命令 "COUPRES" は，下記のために使用されます：

- マシンデータおよセッティングデータに保存されたパラメータ（恒久的に定義されたカップリング）の起動
- プリセット値（ユーザ定義カップリング）の起動

COUPDEF を使用してプログラムされたパラメータ（変換比を含む）は，その後削除されます。

N ... COUPRES (S2,S1)

S2 = 追従主軸，S1 = リーディング主軸

システム変数

追従主軸の現在のカップリングステータス

追従主軸の現在のカップリングステータスは，NC パートプログラムで下記の軸システム変数を使用して読取ることができます：

$AA_COUP_ACT[FS]

FS = 例えば S2 のような主軸番号がついた追従主軸の軸名

読取られる値は，追従主軸に対して次のような意味を持っています：

0: カップリングは無効

4: 主軸同期カップリングが有効

現在の角度オフセット

LS を基準にした FS の現在の位置オフセットのセットポイントは，パートプログラムで下記の軸システム変数を使用して読取ることができます：

$AA_COUP_OFFS[S2]

現在の位置オフセット用の実際値は，

$VA_COUP_OFFS[S2]

を使用して読取ることができます。

FS = 例えば S2 というような主軸番号がついた追従主軸の軸名

アクティブカップリングおよびフォローアップモードの間にコントローラがディスエーブルされた後再イネーブルされた場合，コントローラが再イネーブルされた時の位置オフセットは，プログラムされた元の値とは異なります。この場合，新たな位置オフセットを読取り，必要であれば NC パートプログラムで訂正することができます。

プログラミング例

マスタ主軸とスレーブ主軸を使用した作業

	; リーディング主軸 = マスタ主軸 = 主軸 1
	; スレーブ主軸 = 主軸 2
N05 M3 S3000 M2=4 S2=500	; マスタ主軸は 3000 rpm で回転，スレーブ主軸は 500 rpm で回転
N10 COUPDEF (S2, S1, 1, 1, "NOC", "Dv")	; デフォルトのカップリング，構成することも可能
...	
N70 SPCON	; マスタ主軸を位置制御に含める（セットポイントカップリング）
N75 SPCON(2)	; スレーブ主軸を位置制御に含める
N80 COUPON (S2, S1, 45)	; オフセット位置 = 45 度への即時カップリング
...	
N200 FA [S2] = 100	; 位置決め速度 = 100 度／分
N205 SPOS[2] = IC(-90)	; 90° 負の方向へ移動する
N210 WAITC(S2, "Fine")	;「微」同期まで待つ
N212 G1 X... Y... F...	; 加工
...	
N215 SPOS[2] = IC(180)	; 180° で正の方向へ移動する
N220 G4 S50	; ドウェル時間 = マスタ主軸 50 回転
N225 FA [S2] = 0	; 構成された速度 (MD) の起動
N230 SPOS[2]=IC(-7200)	; 負の方向へ構成された速度で 20 回転
...	
N350 COUPOF (S2, S1)	; 即時のカップリング解除，S=S2=3000
N355 SPOSA[2] = 0	; 0 度でスレーブ主軸を停止する
N360 G0 X0 Y0	
N365 WAITS(2)	; 主軸 2 を待つ
N370 M5	; スレーブ主軸を停止する
N375 M30	

12.3 EG: 電子ギア

概要

「電子ギア」機能によって，5 つまでのリーディング軸の機能としての直線移動ブロックに従って，追従軸の動作を制御することができます。リーディング軸と追従軸との関係は，各リーディング軸に対するカップリング係数によって定義できます。

追従軸モーションパートは，それぞれのカップリング係数を掛けた個々のリーディング軸モーション部分の足算によって求められます。

EG 軸グルーピングを起動している時に，追従軸を定義された位置に従って同期させることができます。

ギアグループをパートプログラムから

- 定義する
- 起動する
- 停止する
- 削除する

ことができます。

追従軸動作はオプションとして

- リーディング軸のセットポイント
- リーディング軸の実際値

から得ることができます。

12.3.1 電子ギアの定義：EGDEF

機能

EG 軸グルーピングは，追従軸と，それぞれのカップリングタイプについて最低 1 つ，最高 5 つのリーディング軸を指定することによって定義されます：

EGDEF（追従軸，リーディング軸 1，カップリングタイプ 1，リーディング軸 2，カップリングタイプ 2, ...）

説明

Following axis	リーディング軸の影響を受ける軸
Leading axis 1, ... leading axis 5	追従軸に影響を及ぼす軸

Coupling type 1, ... coupling type 5	追従軸は下記の影響を受けます： 0: 各リーディング軸の実際値 1: 各リーディング軸のセットポイント

プログラミング

EGDEF(C, B,1, Z, 1, Y, 1)

B, Z, Y はセットポイントを介して C に影響を及ぼす

カップリングタイプは，全てのリーディング軸に対して同じである必要はないので，各リーディング軸に対して別個に指定されます。

カップリング係数は，EG カップリンググループの定義ために 0 でプリセットされます。

EG 軸グルーピング定義の前提条件：

追従軸に対してはまだ軸カップリングの定義をしてはいけません（必要であれば EGDEL を使用して既存のものを最初に削除しなければなりません)。

（注）

EGDEF は前処理停止をトリガします。

12.3.2 電子ギアの起動

起動コマンドについて 2 つの方法があります：

• 方法 1：

下記のコマンドを使用して，同期させずに選択的に EG 軸グルーピングを起動します：

EGON(FA, "Block change mode", LA1,

Z1, N1, LA2 , Z2, N2,..LA5, Z5, N5.)

説明

FA	追従軸
Block change mode	次のようなモードが使用できます： "NOC" 即時のブロック変更 "FINE" 「同期微」でブロック変更が行なわれる "COARSE" 「同期粗」でブロック変更が行なわれる "IPOSTOP" セットポイント同期ランでブロック変更が行なわれる
LA1, ... LA5	リーディング軸

Z1, ... Z5	カップリング係数 i 用のカウンタ
N1, ... N5	カップリング係数 i の分母 カップリング係数 i = カウンタ i ／分母 i

あらかじめ EGDEF を使用して指定されたリーディング軸しかプログラムできません。リーディング軸を最低 1 つはプログラムしなければなりません。

起動時のリーディング軸と追従軸の位置は，「同期位置」として保存されます。「同期位置」は，システム変数 $AA_EG_SYN によって読取ることができます。

- 方法 2：

下記のコマンドを使用して，同期させて選択的に EG 軸グルーピングを起動します：

EGONSYN(FA, "Block change mode", SynPosFA,[, LAi, SynPosLAi, Zi,Ni])

説明

FA	追従軸 :
Block change mode	次のようなモードが使用できます： "NOC" 即時のブロック変更 "FINE" 「同期微」でブロック変更が行なわれる "COARSE" 「同期粗」でブロック変更が行なわれる "IPOSTOP" セットポイント同期ランでブロック変更が行なわれる
[, LAi, SynPosLAi, Zi, Ni]	（角括弧は書かないこと） 最低 1 つ，最高 5 つのリーディング軸の順番：
LA1, ... LA5	リーディング軸
SynPosLAi	i 番目のリーディング軸についての同期位置
Z1, ... Z5	カップリング係数 i 用のカウンタ
N1, ... N5	カップリング係数 i の分母 カップリング係数 i = カウンタ i ／分母 i

あらかじめ EGDEF を使用して指定されたリーディング軸しかプログラムできません。

プログラムされた追従軸およびリーディング軸についての「同期位置」（SynPosFA および SynPosLA）を介して，カップリンググループが同期として有効な位置を定義します。起動時に電子ギアが同期ステータスになっていない場合は，追従軸は定義された同期位置まで移動します。

モジュロ軸がカップリンググループに含まれている場合，それらの係数軸の位置値は係数分を減らされます。これによって次の同期位置へのアプローチが確実になります（相対同期と呼ばれる：例えば次の歯のギャップなど)。インタフェース信号 "Enable following axis override"（追従軸オーバライドイネーブル）DB（30 + 軸番号），DBB26 ビット 4 が追従軸に対して発行される場合しか，同期位置へのアプローチは行なわれません。このインタフェース信号が発行されない場合は，プログラムは EGONSYN ブロックで停止し，前述の信号がセットされるまでセルフクリアアラーム 16771 が出力されます。

12.3.3 電子ギアの停止

有効な EG 軸グルーピングを停止する方法は 3 種類あります。

方法 1：

EGOFS(following axis)

電子ギアは停止します。追従軸は静止するまで減速します。

このコールが前処理停止をトリガします。

方法 2:

EGOFS(following axis, leading axis 1,

... leading axis 5)

このコマンドパラメータ設定によって，個々のリーディング軸の追従軸モーションに対する制御を選択的に取除くことができます。

リーディング軸を最低 1 つは指定しなければなりません。指定されたリーディング軸の追従軸への作用は，選択的にディスエーブルされます。

コールが前処理停止をトリガします。

リーディング軸が有効なままであれば，追従軸はリーディング軸に制御されたまま動きます。この方法で全てのリーディング軸がディスエーブルされた場合，追従軸は静止するまで減速します。

方法 3:

EGOFC(following spindle)

電子ギアは停止します。追従主軸は停止時に有効であった現在の速度で動き続けます。

このコールは前処理停止をトリガします。

（注）

この機能は主軸についてしか使用できません。

12.3.4 電子ギアの定義の削除

前のセクションで述べられた方法で EG 軸グルーピングを停止するまでは，EG 軸グルーピングの定義を削除することはできません。

EGDEL(following axis)

軸グルーピングのカップリング定義が削除されます。

同時に起動できる軸グルーピングの最大数に達するまで，EGDEF を使用して追加の軸グルーピングを定義できます。

このコールは前処理停止をトリガします。

12.3.5 回転フィードレート (G95) ／電子ギア

FPR() コマンドを使用して，電子ギアの追従軸を回転送り速度を指定する軸として定義することもできます。この場合，下記のことが適用されます：

- 送りは，電子ギアの追従軸のセットポイント速度に左右されます。
- セットポイント速度は，リーディング主軸および係数リーディング軸（パス軸ではない）の速度と，それらの主軸および軸に割当てられたカップリング係数から計算されます。
- 直線またはノンモジュロリーディング軸の速度および追従軸のオーバーライドは考慮されません。

12.3.6 電源オン，リセット，モード変更，ブロックサーチの際の EG の応答

電源オン後は，有効なカップリングはなくなります。

有効なカップリングは，リセットおよびモード変更後も保持されます。

ブロックサーチを使用すると，電子ギアを切換え，削除，定義するコマンドは，実行または保持されないで，スキップされます。

12.3.7 電子ギアのシステム変数

電子ギアのシステム変数によって，パートプログラムは EG 軸グルーピングの現在のステータスを指定し，必要な場合はそれらのステータスに反応することができます。

追加説明

電子ギアのシステム変数のリストが付録に示されています。それらの変数は先頭に $AA_EG_ ... または $VA_EG_ ... が付いた名称になっているのが特徴です。

12.4 軸コンテナ

機能

回転マシン／多軸マシンを使用すると，ワークをホールドしている軸が 1 つの加工ステーションから次の加工ステーションへと動きます。

加工ステーションはそれぞれ異なった NCU チャンネルによって制御されるので，ステーション／位置変更時には，ワークをホールドしている軸を適当な NCU チャンネルに動的に再割当てしなければなりません。

軸コンテナはこのことを行なうためのものです。

ローカルな加工ステーションでは，複数のワーククランプ軸／主軸を同時に有効にすることはできません。

軸コンテナは，クランピング軸／主軸との接続を全て集め，通常その中の 1 つだけを加工ステーション用に起動します。

軸コンテナを介して下記の軸を割当てることができます：

- ローカル軸および（または）
- リンク軸

軸コンテナ内の入力を切換えることによって，軸コンテナ内で定義された使用可能な軸を変更することができます。

この切換え機能は，パートプログラムからトリガすることができます。

リンク軸を有する軸コンテナは，NCU（NCU グローバル）全域で有効な，制御装置によって協調させられるツールです。

ローカル軸しか管理しない軸コンテナを持つこともできます。

軸コンテナの構成に関する詳細な情報については，/FB/, B3 を参照してください。

軸コンテナ内の入力は，コマンドを介しての増分 n によって切換えることができます：

プログラミング

AXCTSWE(CT1, CT 2, ...)

AXIS CONTAINER SWITCH ENABLE

(軸コンテナ切換えイネーブル)

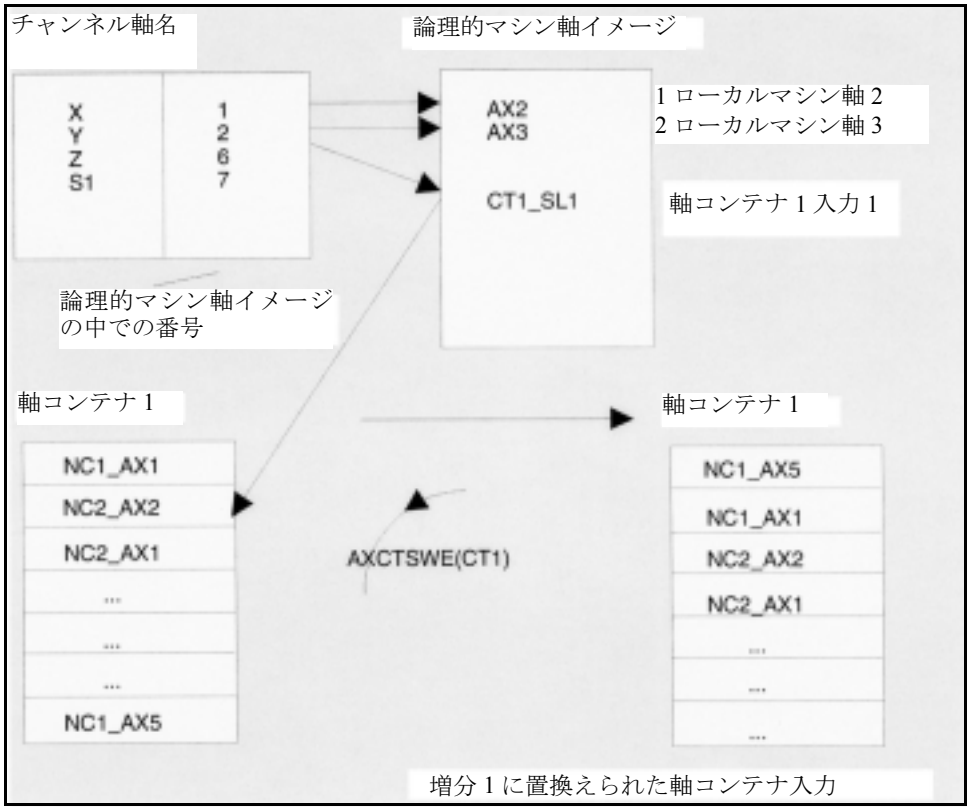
説明

CT1, CT 2 ...

構成要素を切換えようとしている軸コンテナの番号

機能

指定されたコンテナの中に入った軸を持つ各チャンネルは，位置／ステーションでの加工が終了したら，コンテナの回転に対するイネーブルコマンドを発行します。制御装置がコンテナの中に入っている全ての軸についてのそれぞれのチャンネルからイネーブルコマンドを全て受け取ると，コンテナは，MD で指定された増分だけ回転します。



前述の例では，軸コンテナが 1 だけ回転した後，NCU1 上の軸 AX1 の代わりに NCU1 上の軸 AX5 がチャンネル軸 Z に割当てられます。

軸コンテナが回転した後，論理的マシン軸イメージを介して回転した軸コンテナを照会するチャンネルを持つ全ての NCU には，新たな軸の割当てが適用されます。

13 ユーザストックリムーバブルプログラム

13.1 ストックリムーバブルのためのサポート機能

ユーザストックリムーバブルプログラム

あらかじめプログラムされたストックリムーバブルプログラムはストックリムーバブルに備えるものです。独自のストックリムーバブルプログラムを開発するために下記の機能を使用することもできます。

CONTPRON	表形式で輪郭準備を起動する
INTERSEC	2 つの輪郭要素の交点を計算する
EXECTAB	表の輪郭要素を 1 ブロックずつ実行する
CALCDAT	半径および中心点を計算する

これらの機能はストックリムーバブルのためだけでなく，全般的に使用することができます。

13.2 輪郭準備 CONTPRON

プログラミング

CONTPRON (TABNAME, BEARBART, NN, MODE)

EXECUTE (ERROR)

パラメータの説明

CONTPRON	輪郭準備の起動
TABNAME	輪郭表の名称
MACHINE	加工タイプ別パラメータ： "G"：縦旋削：内部加工 "L"：縦旋削：外部加工 "N"：正面旋削：内部加工 "P"：正面旋削：外部加工
NN	タイプ INT の結果変数におけるレリーフ切削の数
MODE (Software Version 4.4 and higher)	加工の方向，タイプ INT 0 = 前回と同様の輪郭準備（デフォルト値） 1 = 両方向の輪郭準備
EXECUTE	輪郭準備を終了する
ERROR	エラーチェックバック用変数，タイプ INT 1 = エラー；0 = エラーなし

機能

CONTPRON の後で実行されるブロックが，準備する予定の輪郭を記述します。

ブロックは処理されずに輪郭表にファイルされます。

各輪郭要素は，輪郭表の 2 次元配列の中の 1 行に対応します。

レリーフ切削の数が返されます。

EXECUTE は輪郭準備を停止し，通常の実行モードに戻します。

例

N30 CONTPRON(...)

N40 G1 X... Z...

N50...

N100 EXECUTE(...)

追加説明

コールのための前提条件

CONTPRON がコールされる前に，

- 衝突せずに加工できるように開始点にアプローチしなければなりません。
- G40 によるツールエッジ半径補正を停止しなければなりません。

使用可能な移動コマンド，座標系

丸めおよび面取りに加えて，輪郭プログラミングに対しては，G コマンド G0 ～ G3 しか使用できません。

CIP および CT を介しての円形パスプログラミングをサポートします。

SPLINE，POLYNOM，THREAD はエラーを発生させます。

CONTPRON と EXECUTE との間でフレームを起動させることによって座標系を変更することはできません。同じことが G70 と G71 との間の変更にも適用されます。

輪郭準備の終了

あらかじめ定義されたサブルーチン EXECUTE (ERROR) をコールすると輪郭準備は終了し，輪郭が記述されたら，システムは通常の実行に戻ります。復帰パラメータ 1 = エラー／ 0 = エラーなし（つまり輪郭は，エラーを起こさずに準備された）が関連する変数の中に保存されます。

レリーフカット要素

個々のレリーフ切削要素に対する輪郭の記述は，サブルーチンの中でも個々のブロック内でも行なえます。

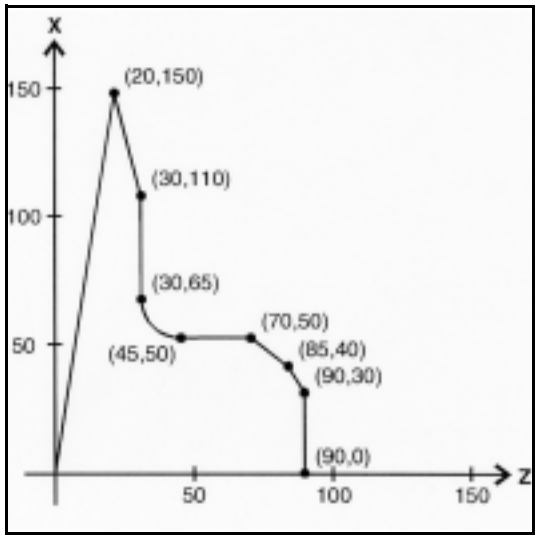
プログラムされた輪郭方向とは無関係のストックリムーバブル

CONTPRON がコールされたら，輪郭表はプログラムされた方向とは無関係に使用できます。

プログラミング例 1

- 名称 KTAB,
- 30 個までの輪郭要素（円，直線），
- リリーフカット要素の数用の変数
- エラーメッセージ用の変数

を有する輪郭テーブルを作成してください。



メインプログラムファイル

N10 DEF REAL KTAB[30,11]	KTAB という名称で，例えば最大 30 個の輪郭要素を持つ輪郭テーブル　パラメータ値 11 は固定サイズ。
N20 DEF INT ANZHINT	ANZHINT という名称の，レリーフカット要素の数用の変数
N30 DEF INT ERROR	確認応答用の変数 0 = エラー無し，1 = エラー
N40 G18	
N50 CONTPRON (KTAB,"G",ANZHINT)	輪郭準備コール
N60 G1 X150 Z20 N70 X110 Z30 N80 X50 RND=15 N90 Z70 N100 X40 Z85 N110 X30 Z90 N120 X0	N60 ～ N120 輪郭の記述
N130 EXECUTE(ERROR)	輪郭テーブルの記入を終了する，通常のプログラムの実行に切換える
N140 ...	テーブルの処理を続行する

テーブル KTAB

(0)	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9) (10)	
7	7	11	0	0	20	150	0	82.40535663	0	0
0	2	11	20	150	30	110	-1111	104.0362435	0	0
1	3	11	30	110	30	65	0	90	0	0
2	4	13	30	65	45	50	0	180	45	65
3	5	11	45	50	70	50	0	0	0	0
4	6	11	70	50	85	40	0	146.3099325	0	0
5	7	11	85	40	90	30	0	116.5650512	0	0
6	0	11	90	30	90	0	0	90	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

列の要素の説明

(0) 次の輪郭要素を指す（その列の行番号を指す）ポインタ

(1) 前の輪郭要素を指すポインタ

(2) 動作に対する輪郭モードのコード化

X = abc について使用可能な値

a = 10^2 G90 = 0 G91 = 1

b = 10^1 G70 = 0 G71 = 1

c = 10^0 G0 = 0 G1 = 1 G2 = 2 G3 = 3

(3), (4) 輪郭要素の開始点

現在の平面での (3) = 横座標，(4) = 縦座標

(5), (6) 輪郭要素の終了点

現在の平面での (5) = 横座標，(6) = 縦座標

(7) 最大／最小インジケータ：輪郭上のローカル最大値および最小値を識別する

(8) 輪郭要素と横座標との間の最大値（縦方向の加工用）または輪郭要素と縦座標との間の最大値（横方向の加工用）

角度はプログラムされた加工タイプによって決まります。

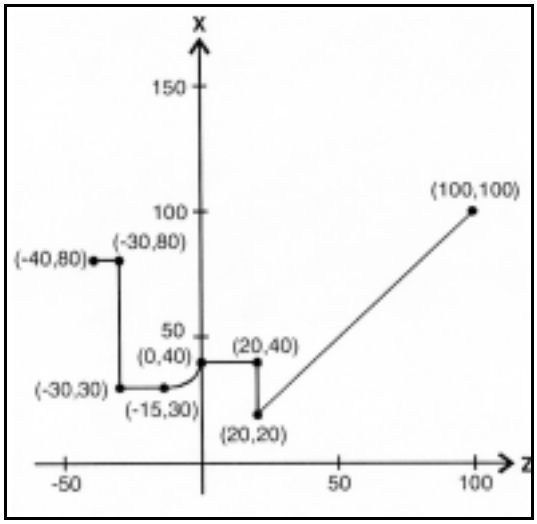
(9), (10) 輪郭要素が円ブロックの場合の輪郭要素の中心点座標

(9) = 横座標 , (10) = 縦座標

プログラミング例 2

- 名称 KTAB,
- 92 個までの輪郭要素（円，直線），
- モード：縦旋削，外部加工
- 前方への準備および後方への準備

を有する輪郭テーブルを作成してください。



メインプログラムファイル

N10 DEF REAL KTAB[92,11]	KTAB という名称で，例えば最大 92 個の輪郭要素を持つ輪郭表　パラメータ値 11 は固定サイズ。
N20 CHAR BT="L"	CONTPRON 用のモード：縦旋削，外部加工
N30 DEF INT HE=0	レリーフカット要素の数 = 0
N40 DEF INT MODE=1	前方への準備および後方への準備
N50 DEF INT ERR=0	エラーチェックバックメッセージ
...	
N100 G18 X100 Z100	
N105 CONTPRON (KTAB, BT, HE, MODE)	輪郭準備コール
N110 G1 G90 Z20 X20 N120 X40 N130 Z0 N140 G2 Z-15 X30 K=AC(-15) I=AC(45) N150 G1 Z-30 N160 X80 N170 Z-40	
N180 EXECUTE(ERR)	輪郭テーブルの記入を終了する，通常のプログラムの実行に切換える
...	

テーブル KTAB

輪郭準備が終了した後は，輪郭は両方向で加工できます。

行	列										
(0)	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	
0	6 [1)]	7 [2)]	11	100	100	20	20	0	45	0	0
1	0 [3)]	2	11	20	20	20	45	-3	90	0	0
2	1	3	11	20	45	0	45	0	0	0	0
3	2	4	12	0	45	-15	30	5	90	-15	45
4	3	5	11	-15	30	-30	30	0	0	0	0
5	4	7	11	-30	30	-30	45	-1111	90	0	0
6	7	0 [4)]	11	-30	80	-40	80	0	0	0	0
7	5	6	11	-30	45	-30	80	0	90	0	0
8	1 [5)]	2 [6)]	0	0	0	0	0	0	0	0	0
	...										
83	84	0 [7)]	11	20	45	20	80	0	90	0	0
84	90	83	11	20	20	20	45	-1111	90	0	0
85	0 [8)]	86	11	-40	80	-30	80	0	0	0	0
86	85	87	11	-30	80	-30	30	88	90	0	0
87	86	88	11	-30	30	-15	30	0	0	0	0
88	87	89	13	-15	30	0	45	-90	90	-15	45
89	88	90	11	0	45	20	45	0	0	0	0
90	89	84	11	20	45	20	20	84	90	0	0
91	83 [9)]	85 [10)]	11	20	20	100	100	0	45	0	0

列の要素の説明

(0) 次の輪郭要素を指す（その列の行番号を指す）ポインタ

(1) 前の輪郭要素を指すポインタ

(2) 動作に対する輪郭モードのコード化

X = abc について使用可能な値

a = 10^2 G90 = 0 G91 = 1

b = 10^1 G70 = 0 G71 = 1

c = 10^0 G0 = 0 G1 = 1 G2 = 2 G3 = 3

(3), (4) 輪郭要素の開始点

現在の平面での (3) = 横座標，(4) = 縦座標

(5), (6) 輪郭要素の終了点

現在の平面での (5) = 横座標，(6) = 縦座標

(7) 最大／最小インジケータ：輪郭上のローカル最大値および最小値を識別する

(8) 輪郭要素と横座標との間の最大値（縦方向の加工用）または輪郭要素と縦座標との間の最大値（トランズバース加工用）

角度はプログラムされた加工タイプによって決まります。

(9), (10) 輪郭要素が円ブロックの場合の輪郭要素の中心点座標

(9) = 横座標 , (10) = 縦座標

列中の注釈の説明

テーブルの行 0 では常に：

1) Previous: 行 n は輪郭終点（正順）を含んでいる。

2) Following: 行 n は輪郭表終点（正順）である。

各輪郭要素（正順）：

3) Previous: 輪郭始点（正順）

4) Following: 輪郭終点（正順）

行輪郭テーブル終点（正順）+ 1 では常に：

5) Previous: レリーフカットの数（正順）

6) Following] レリーフカットの数（逆順）

各輪郭要素（逆順）：

7) Following: 輪郭終点（逆順）

8) Previous: 輪郭始点（逆順）

テーブルの最後の行では常に：

9) Previous: 行 n は輪郭表始点（逆順）である。

10) Following: 行 n は輪郭始点（逆順）を含んでいる。

13.3 2 つの輪郭要素の交点

プログラミング

VARIB=INTERSEC (TABNAME1[n1], TABNAME2[n2], TABNAME3)

パラメータの説明

VARIB	ステータス用の変数 TRUE：交点あり FALSE：交点なし
TABNAME1[n1]	テーブルの名称およびテーブル 1 の n1 番目の輪郭要素
TABNAME2[n2]	テーブルの名称およびテーブル 2 の n2 番目の輪郭要素
TABNAME3	有効な面（G17 ～ G19）での交点座標用のテーブルの名称

機能

INTERSEC は，CONTPRON を使用して生成された輪郭テーブルから，2 つの正規化された輪郭要素の交点を計算します。示されたステータスによって，交点が存在するかどうかを指定します（TRUE = 交点あり，FALSE = 交点なし）。

追加説明

変数は，定義しなければ使用できません。

プログラミング例

テーブル KTAB1 の輪郭要素 3 と表 KTAB2 の輪郭要素 7 の交点を計算してください。有効な面の交点座標は CUT 内に保存されます（第 1 要素 = 横座標，第 2 要素 = 縦座標)。

交点が存在していなければ，プログラムは NOCUT（交点なし）へジャンプします。

DEF REAL KTAB1 [12, 11]	輪郭テーブル 1
DEF REAL KTAB2 [10, 11]	輪郭テーブル 2
DEF REAL CUT [2]	交点表
DEF BOOL ISPOINT	ステータス用の変数
...	
N10 ISPOINT=INTERSEC (KTAB1[3],KTAB2[7],SCHNITT)	
	輪郭要素の交点をコールする
N20 IF ISPOINT==FALSE GOTOF NOCUT	NOCUT へジャンプする
...	

13.4 テーブルからの輪郭要素の移動 EXECTAB

プログラミング

EXECTAB (TABNAME[n])

パラメータの説明

TABNAME[n]	要素の番号 n を含むテーブルの名称

機能

例えば CONTPRON コマンドを使用して生成された表内で，輪郭要素を 1 ブロックずつ移動するためにコマンド EXECTAB を使用することができます。

プログラミング例

表 KTAB に保存された輪郭要素を，サブルーチン EXECTAB によってノンモーダルに移動します。要素 0 ～ 2 を連続したコールで通過します。

N10 EXECTAB (KTAB[0])	テーブル KTAB の移動要素 0
N20 EXECTAB (KTAB[1])	テーブル KTAB の移動要素 1
N30 EXECTAB (KTAB[2])	テーブル KTAB の移動要素 2

13.5 円データの計算 - CALCDAT

プログラミング

VARIB = CALCDAT(PNT[n,2],NO,RES)

パラメータの説明

VARIB	ステータス用の変数　TRUE = 円，FALSE = 円ではない
PKT[n,2]	計算対象の点　n = 点の数（3 または 4）；2 = 点の座標
NO.	計算に使用される点の数：3 または 4
RES[3]	結果用の変数：円の中心点座標および半径の指定； 0 = 円の中心点の横座標，1 = 円の中心点の縦座標；2 = 半径

機能

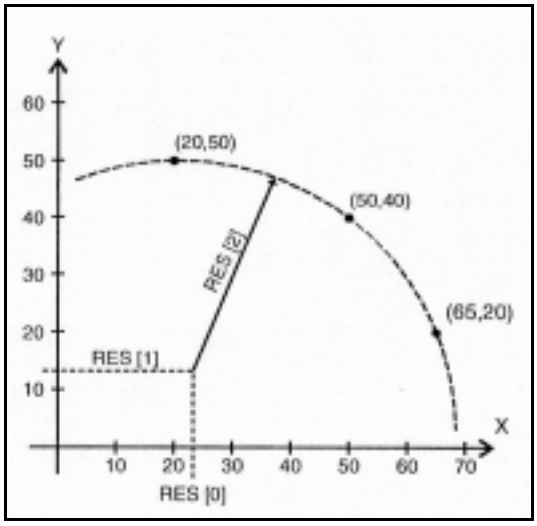
3 つまたは 4 つの既知の円弧の点から半径および円の中心点座標を計算します。

指定された点は，異なっていなければなりません。

4 つの点が直接円弧上にない場合は，円の中心点および半径には平均値がとられます。

プログラミング例

プログラムは，3 つの点が円弧に沿って位置しているかどうかを求めます。



N10 DEF REAL PNT[3,2]=(20,50,50,40,65,20)	ポイントの定義
N20 DEF REAL RES[3]	結果
N30 DEF BOOL STATUS	ステータス用の変数
N40 STATUS = CALCDAT(PNT,3,RES)	計算された円データをコールする
N50 IF STATUS == FALSE GOTOF ERROR	エラーへジャンプする

14 表

14.1 命令のリスト

凡例 :

[1] プログラム開始時の初期設定 (別の設定がプログラムされていない場合，制御システムのデリバリステータスにおいて).

[2] グループの番号は，テーブル "List of G functions/preparatory functions" （G 機能／準備機能のリスト）に対応します。

[3] 絶対終点 : モーダル ; 相対的終点：ノンモーダル ; その他の場合，モーダル / ノンモーダルは G 機能の構文によって決まります。

[4] IPO パラメータはアークセンタとして増分で機能します。AC を用い，絶対モードでプログラム可能。他の意味を持つとき（たとえば，ピッチ），アドレス修正は無視されます。

[5] OEM ユーザは，２種類の別の補間タイプを組み込んで，それらの名称を変更することができます。

[6] 拡張アドレスブロックフォーマットは，これらの機能用に使用することはできません。

名称	意味	値の割当て	説明，コメント	構文	モーダル／ノンモーダル	グループ [2]
:	ブロック番号 - 主要ブロック (N を参照) 0	... 9999 9999 整数値のみ , 符号なし	ブロック専用コード - N... の代わりに ; このブロックは，次に来る完全な加工セクションのための命令をすべて含んでいなければならない。例 :20			
A	軸	実数			m,s3	
A2	ツールの向き :Euler 角	実数			s	
A3	ツールの向き : 方向ベクトル構成要素	実数			s	
A4	ブロック始めについてのツールの向き	実数			s	
A5	ブロック終りについてののツールの向き ; 標準ベクトル構成要素	実数			s	
ABS	絶対値	実数				
AC	寸法入力，絶対	0, ..., 359.999・		X=AC(100)	s	
ACC	軸加速	実数，符号なし			m	
ACN	回転軸用絶対寸法設定，負方向のアプローチ位置			A=ACN(...) B=ACN(...) C=ACN(...)	s	
ACP	回転軸用絶対寸法設定，正方向のアプローチ位置			A=ACP(...) B=ACP(...) C=ACP(...)	s	

ACOS	アークコサイン (三角関数)	実数				
ADIS	パス機能 G1, G2, G3, ... の再サーフェシング距離	実数 , 符号なし			m	
ADISPOS	急速移動用再サーフェシング距離 G0	実数 , 符号なし			m	
ALF	角度リフトファスト	整数 , 符号なし			m	
AMIRROR	プログラム可能なミラリング (追加ミラー)			AMIRROR X0 Y0 Z0 ; 個別ブロック	s	3
AND	論理積 AND					
ANG	輪郭定義角	実数				
AP	極角度 (角　極)	0, ..., ± 360°			m,s [3]	
APR	読取り／表示アクセス保護 (アクセス保護読取り)	整数，符号なし				
APW	書き込みアクセス保護 (アクセス保護書き込み)	整数，符号なし				
AR	口角度 (円の角度)	0, ..., 360°			m,s [3]	
AROT	プログラム可能な回転 (追加回転)	1 番目のジオメトリ軸回りの回転 : -180° .. 180°　2 番目のジオメトリ軸 : -89.999° .. 9・3 番目のジオメトリ軸 : -180° .. 180°		AROT X...Y... Z... ; 分離した AROT RPL= ブロック	s	3
AS	マクロ定義	ストリング				
ASCALE	プログラム可能なスケーリング (追加スケール)			ASCALE X... Y... Z... ; 個別ブロック	s	3
ASIN	アークサイン (三角関数)	実数				
ASPLINE	アキマスプライン				m	1
ATAN2	アークタンジェント 2	実数				
ATRANS	追加プログラム可能オフセット (追加翻訳)			ATRANS X... Y... Z... ; 個別ブロック	s	3
AX	符号なし整数	実数			m,s [3]	
AXCSWAP	スイッチコンテナ軸			AXCSWAP(CTn,CTn+1,...)		25
AXIS	データタイプ軸名		ファイル名を追加できる			

AXNAME	入力ストリングを軸名に変換する (軸名を受取る)	ストリング	入力ストリングが有効な軸名称を含まない場合，アラームが生成される。			
AXSTRING	軸名称をストリングに変換する (軸をストリングとして受取る)	AXIS	ファイル名を追加できる			
B	軸	実数			m,s [3]	
B_AND	ビット AND					
B_NOT	ビット否定					
B_OR	ビット OR					
B_XOR	ビット排他的 OR					
B2	ツールの向き : Euler 角	実数			s	
B3	ツールの向き : 方向ベクトル構成要素	実数			s	
B4	ブロック始めのためのツールの向き	実数			s	
B5	ブロック終りのためのツールの向き ; 標準ベクトル構成要素	実数			s	
BAUTO	次に続く 3 点を使って最初のスプラインセグメントを定義する (begin not a knot)				m	19
BLSYNC	割込みルーチンの処理は，次のブロックチェンジで開始するためだけである。					
BNAT [1]	最初のスプラインブロックへの自然な遷移 (自然開始)				m	19
BOOL	データタイプ ブーリアン 値 TRUE / FALSE または 0 / 1					
BRISK [1]	活発なパス加速				m	21
BRISKA	プログラムされた軸について活発な軸加速を起動する					
BSPLINE	B スプライン				m	1
BTAN	最初のスプラインブロックにタンジェント遷移 (タンジェンシャル開始)				m	19
C	軸	実数			m,s [3]	
C2	5 ツールの向き : Euler 角	実数			s	
C3	ツールの向き : 方向ベクトル構成要素	実数			s	
C4	ブロック始めのためのツールの向き	実数			s	
C5	ブロック終りのためのツールの向き ; 標準ベクトル構成要素	実数			s	
CAC	位置の絶対アプローチ (コード化された位置 : 絶対座標)		コード化された値はテーブル指数である ; テーブル値にアプローチする。			
CACN	テーブルに保存された値の負方向に絶対アプローチする。(コード化された位置絶対負)		回転軸を位置決め軸としてプログラムすることができる。			
CACP	テーブルに保存された値の正方向に絶対アプローチする。(コード化された位置絶対正)					
CALCDAT	半径および 3，4 点からの中心点または円を計算する (円データを計算する)	VAR 実数 [3]	点は必ず異なっていること。			

CALL	間接的なサブルーチンコール			CALL PROGVAR		
CANCEL	モーダル同期アクションをキャンセルする	INT	指定された ID を使ってキャンセルする。 パラメータなしの場合 : すべてのモーダルシンクロナイズドアクションの選択を解除する。			
CASE	条件付のプログラムブランチ					
CDC	位置の直接アプローチ (コード化された位置 : 直接座標)		CAC 参照			
CDOF [1]	衝突検知 OFF				m	23
CDON	衝突検知 ON				m	23
CFC [1]	輪郭上での一定フィード				m	16
CFIN	内半径での一定フィード, 外半径では加速 (内半径での一定フィード)				m	16
CFTCP	ツール中心点での一定フィード (中心点パス) (ツール中心点内で一定フィード)				m	16
CHAN	データの有効範囲を指定する		チャンネルにつき 1 回			
CHANDATA	チャンネルデータアクセスについてチャンネル番号をセットする	INT	初期化ブロックでのみ可能			
CHAR	データタイプ ASCII 文字	0, ..., 255				
CHR	面取り ; 値 = 動作の方向での面取り長さ (面取り) 面取り ; 値 = 面取り長さ	実数 , 符号なし			s	
CHKDNO	D 番号チェック					
CIC	位置の相対的アプローチ (コード化された位置 : 増分座標)		CAC 参照			
CIP	中間点を通る円補間			CIP X... Y... Z... I1=... J1=... K1=...	m	1

CLAL	アラームをクリアする	INT	パラメータ : アラーム番号			
CLEARM	チャンネル座標について 1 ／複数のマーカをリセットする	INT, 1 - n	自身のチャンネルでの加工に影響を与える			
CLGOF	センタレスのグラインディング OFF の場合の一定ワーク速度					
CLGON	センタレスのグラインディング ON の場合の一定ワーク速度					
CLRINT	割込みの選択解除 :	INT	パラメータ : 割込み番号			
CMIRROR	座標軸におけるミラー	FRAME				
COMPOF [1]	コンプレッサ OFF				m	30
COMPON	コンプレッサ ON				m	30
COMPCURV	コンプレッサ ON 一定カーブ多項式				m	30
CONTPRON	輪郭の準備を起動する (輪郭準備 ON)				m	49
COS	コサイン (三角関数)	実数				
COUPDEF	ELG グループ／同期主軸グループ (カップリング定義)	ストリング	ブロック変更 (ソフトウエア) 応答 : NOC: ソフトウエア制御なし, FINE/COARSE: 「同期化　微／粗」, IPOSTOP: オーバーレイされた動作のセットポイントに依存した終了時のソフトウエア			
COUPDEL	ELG グループを削除する (カップリング削除)					
COUPOF	ELG グループ／同期主軸ペア OFF (カップリング OFF)					
COUPON	ELG グループ／同期主軸ペア ON (カップリング ON)					
COUPRES	ELG グループをリセットする (カップリングリセット)		プログラムされた値が無効 ; マシンデータ値が有効			
CP	パス動作 (連続パス)				m	49
CPRECOF [1]	プログラム可能な輪郭精度 OFF				m	39
CPRECON	プログラム可能な輪郭精度 ON				m	39
CPROT	チャンネル別保護ゾーン ON/OFF					
CPROTDEF	チャンネル別保護エリア定義					

CR	円半径	実数，符号なし			s	
CROT	現行の座標系の回転。	FRAME	パラメータの最大数 : 6			
CSCALE	マルチ軸用スケール係数。	FRAME	パラメータの最大数 : 2 * 軸数最大			
CSPLINE	立方スプライン				m	1
CTAB	カーブテーブルからのリーディング軸位置に従って，追従軸位置を定義する	実数	パラメータ 4/5 がプログラムされていない場合 : 標準スケーリング			
CTABDEF	テーブル定義オン					
CTABDEL	カーブテーブルを消去する					
CTABEND	テーブル定義オフ					
CTABINV	カーブテーブルからの追従軸位置に従って，リーディング軸位置を定義する	実数	CTAB 参照			
CT	タンジェント遷移で旋回する			CT X... Y.... Z...	m	1
CTRANS	Zero マルチ軸用オフセット	FRAME	最大 8 軸			
CUT2D [1]	2 1/2D ツールオフセット (2 次元のカッタ補正タイプ)				m	22
CUT2DF	2 1/2D ツールオフセット (カッタ補正タイプ 2 次元フレーム); このツールオフセットは，現在のフレーム に関して機能 (傾斜面) 。				m	22
CUT3DC	3 次元ツールオフセット周囲フライス削り (カッタ補正タイプ 3 次元周囲)				m	22
CUT3DF	3 次元ツールオフセット正面フライス加工 (カッタ補正タイプ 3 次元面)				m	22
CUT3DFF	有効なフレームの機能としてツール向きの一定な 3 次元ツールオフセッ正面フライス加工 (カッタ補正タイプ 3 次元面フレーム)				m	22
CUT3DFS	有効なフレームとは無関係なツール向きの一定名 3 次元ツールオフセット正面フライス加工 (カッタ補正タイプ 3 次元正面フレーム)				m	22
CUTCONO [1]	一定の半径補正 OFF				m	40
CUTCONON	一定の半径補正 ON				m	40
D	ツールオフセット番号	1, ... 32 000	指定ツール T... 用オフセットデータを含む ; D0 →ツール用オフセット値	D...		
DC	回転軸用絶対寸法設定，位置に直接アプローチする			A=DC(...) B=DC (...) C=DC(...) SPOS=DC(...)	s	
DEF	変数定義	整数 , 符号なし				
DEFAULT	CASE ブランチによるブランチ		式が指定値のいずれも満足しない場合に，ここへジャンプする			
DEFINE	マクロを定義する					
DELDTG	転送距離を削除する					
DELT	ツールを削除する		デュプロ番号を削除できる			
DIAMOF [1]	直径のプログラミング : OFF				m	29
DIAMON	直径のプログラミング : ON				m	29
DIAM90	G90 には直径プログラム，G91 には半径プログラム				m	29

DILF	急速リフト長さ				m	
DISABLE	割込み OFF					
DISC	ツール径補正において移動円の行き過ぎ量	0, ..., 100			m	
DISPLOF	現在のブロック表示を抑制する (表示 OFF)					
DISPR	再位置決めのための距離パス	実数，符号なし			s	
DISR	再位置決めのための距離	実数，符号なし			s	
DITE	スレッドランアウトパス	実数			m	
DITS	スレッドランインパス	実数			m	
DIV	整数除法					
DL	ツール合計補正	INT			m	
DRFOF	ハンドルオフセットを停止する (DRF)				m	
DRIVEA	プログラム軸のためのベント加速特性カーブをオンにする。					
DZERO	無効なチャンネルに割当てられた TO ユニットのすべてのツールの D 数をセットする					
EAUTO	最後の 3 点による最後のスプラインセグメントの定義 (end not a knot)				m	20
EGDEF	電子ギアの定義 (電子ギア定義)		1 追従軸について，最大 5 リーディング軸で			
EGDEL	追従軸用カップリング定義を削除する (電子ギア削除)		前処理停止をトリガする			
EGOFC	電子ギア連続のスイッチをオフにする (電子ギア OFF 連続)					
EGOFS	電子ギアを選択的にスイッチオフにする (電子ギア OFF 選択)					
EGON	電子ギアをスイッチオンにする (電子ギア ON)		同期なし			
EGONSYN	電子ギアのスイッチをオンにする (電子ギア ON 同期)		同期あり			
ELSE	ブランチをプログラムする，IF 条件が満たされない場合					
ENABLE	割込み ON					
ENAT [1]	次の移動ブロックへ自然カーブ遷移 (自然終了)				m	20
ENDFOR	FOR カウンタループ行を終了する					
ENDIF	IF ブランチ行を終了する					
ENDLOOP	エンドレスプログラムループ LOOP の行を終了する					
ENDPROC	開始行 PROC で，プログラムの行を終了する					
ENDWHILE	WHILE ループ行を終了する					
ETAN	スプライン開始時，次の移動ブロックにタンジェンシャルカーブ遷移 (タンジェンシャル終了)				m	20
EVERY	条件を FALSE(偽) から TRUE(真) に変更する場合，シンクロナイズドアクションを実行する					
EXECTAB	モーションテーブルからのエレメントを実行する (テーブル実行)					
EXECUTE	プログラム実行 ON		基準点エディットモードからあるいは保護ゾーンを作成後，標準プログラム実行に戻る			

EXP	指数関数 e^x	実数				
EXTERN	パラメータ転送でサブルーチンをブロードキャストする					
F	フィード値 (ドウェル時間も，G4 に関連づけて，F に従ってプログラムされる)					
0.001, ..., 99 999.999	ツール／ワークパス速度 ; G94 or G95 の機能として mm/min または mm/revolution で表した寸法 F=100 G1 ...					
FA	軸フィード (軸フィード)	0.001, ..., 999999.999 mm/min, degree/（min; 0.001, ..., 39999.9999 inch/min		FA[X]=100	m	
FAD	スムーズなアプローチと後退のための送り速度をインフィードする (フィードアプローチする / 離れる)	実数 , 符号なし				
FALSE	論理定数 : 偽	BOOL	BOOL は整数定数 0 で置き換えることができる			
FCTDEF	多項式機能を定義する		SYFCT または PUTFTOCF で評価される			
FCUB	立方スプラインによるフィード変数 (フィード立方)				m	37
FD	ハンドルオーバライドのためのパスフィード (フィード DRF)	実数 , 符号なし			s	
FDA	ハンドルオーバライドのための軸フィード (フィード DRF 軸)	実数 , 符号なし			s	
FFWOF [1]	フィードフォワード制御 OFF (フィードフォワード OFF)				m	24
FFWON	フィードフォワード制御 ON (フィードフォワード ON)				m	24
FGREF	基準半径				m	
FGROUP	パスフィードを用いて軸を定義する		F は，FGROUP の下でプログラムされたすべての軸に適用される FGROUP (Axis1, [Axis2], ...)			
FIFOLEN	プログラム可能な前処理深さ					
FL	同期軸の制限速度 (フィード制限)	実数 , 符号なし	G93, G94, G95 でセットされた単位が適用される (最高速移動)	FL [Axis] =...	m	
FLIN	直線的可変フィード (直線フィード)				m	37
FMA	マルチ軸をフィードする	実数 , 符号なし			m	
FNORM [1]	DIN66025 による標準フィード (フィード標準)				m	37
FOR	一定回数通過のカウンタループ					
FORI1	大円上で方向ベクトルを旋回するためのフィード				m	
FORI2	旋回方向ベクトル周囲をオーバレイして回転するためのフィード				m	
FP	固定点 : アプローチする固定点の数	整数 , 符号なし		G75 FP=1	s	

FPO	多項式を介してプログラムされたフィード特性 (フィード多項式)	実数	二次，三次多項式係数			
FPR	回転軸識別	0.001 ...999999.999		FPR (回転軸)		
FPRAOF	回転送り速度を停止する					
FPRAON	回転送り速度を起動する					
FRAME	座標系を定義するためのデータタイプ		各ジオメトリ軸について含まれる : オフセット，回転，せん断角度，スケーリング，ミラリング ; 各専用軸について : オフセット，スケーリング，ミラリング			
FRC	半径と面取りのためのフィード				s	
FRCM	半径と面取りモーダルのためのフィード				m	
FTOC	ツールオフセット（微）を変更する		FCTDEF で定義された自由度 3 の多項式の機能として			
FTOCOF [1]	オンラインツールオフセット (微) OFF (ツールオフセット (微）OFF)				m	33
FTOCON	オンラインツールオフセット (微) ON (ツールオフセット（微）ON)				m	33
FXS	固定停止への移動 ON (固定停止)	整数 , 符号なし	1 = 選択 , 0 = 選択解除		m	
FXST	固定停止への移動のためのトルクリミット (固定停止トルク)	%	オプショナル設定		m	
FXSW	固定停止への移動のための監視ウインドウ (固定停止ウインドウ)	mm，インチまたは度	オプショナル設定			

G 機能						
G	G 機能 (準備機能) G 機能は，G グループに分けられる。グループ内の G 機能のうち 1 つだけが，ブロック中にプログラムできる。G 機能は，モーダル的に有効であるか（同じグループの別の機能でキャンセルするまで），またはプログラムされるブロック内でのみ有効である（非モーダルで)。	整数，プリセット値のみ。		G..		
G0	位置決め（早送り)		モーション	G0 X... Z...	m	1
G1 [1]	直線補間		コマンド	G1 X... Z... F... m 1		
G2	右回りの円弧補間			G2 X... Z... I... K... F... ; 中心点および終点 G2 X... Z... CR=... F... ; 半径および終点 G2 AR=... I... K... F... ; 口径角度および中心点 G2 AR=... X... Z... F... ; 口径角度および終点	m	1
G3	左回りに円弧補間			G3 ... ; G2 については他	m	1
G4	ドウェル		特殊モーション	G4 F... ; 単位 s でのドウェル時間または G4 S... ; 主軸回転でのドウェル時間 ; 個別ブロック	s	2
G9	イグザクトストップ				s	11
G17 [1]	加工面 X/Y の選択		インフィード方向 Z		m	6
G18	加工面 Z/X の選択		インフィード方向 Y		m	6
G19	加工面 Y/Z の選択		インフィード方向 X		m	6
G25	低い作業エリア制限		チャンネル軸での値の割当て チャンネル軸	G25 X.. Y.. Z.. ; 個別ブロック	s	3
G26	高い作業エリア制限			G26 X.. Y.. Z..; 個別ブロック	s	3

G33	一定ピッチでのスレッド補間	0.001, ..., 2000.00 mm/rev	モーションコマンド	G33 Z... K... SF=... ; シリンダスレッド G33 X... I... SF=... ; 正面スレッド G33 Z... X... K... SF=... ; テーパスレッド (X より Z 軸において長いパス) G33 Z... X... I... SF=... ; テーパスレッド (Z 軸より X 軸において長いパス)	m	1
G40[1]	ツール径補正 OFF				m	7
G41	工具径補正左側				m	7
G42	工具径補正右側				m	7
G53	現在のゼロオフセットの抑止 (モーダルでない)		プログラムされたオフセットを含む		s	9
G54	ワーク座標系 1 へシフト				m	8
G55	ワーク座標系 2 へシフト				m	8
G56	ワーク座標系 3 へシフト				m	8
G57	ワーク座標系 4 へシフト				m	8
G58						
G59						
G60[1]	イグザクトストップ				m	10
G63	補正チャックを用いたタッピング			G63 Z... G1	s	2
G64	イグザクトストップモードキャンセル - 輪郭モード				m	10
G70	インチによる寸法				m	13
G71 [1]	メータによる寸法				m	13
G74	基準点アプローチ		マシン軸	G74 X... Z...; 個別ブロック	s	2
G75	固定点アプローチ			G75 FP=.. X1=... Z1=...; 独立	s	2
G90 [1]	アブソリュート指令			G90 X... Y... Z...(...) Y=AC(...) または X=AC Z=AC(...)	m s	14
G91	インクリメンタル指令			G91 X... Y... Z... または X=IC(...) Y=IC(...) Z=IC(...)	m s	14
G94 [1]	直線フィード F を mm/min あるいは inch/min および "/min で表す				m	15
G95	回転フィードレート F を mm/rev あるいは inch/rev で表す				m	15
G96	一定切削速度 ON			G96 S... LIMS=... F...	m	15
G97	一定切削速度 OFF				m	15

G110	最新のプログラム設定位置に関連する極プログラミング			G110 X.. Y.. Z..	s	3
G111	現在のワーク座標系のゼロ点に関連する極プログラミング			G110 X.. Y.. Z..	s	3
G112	最新の有効極に関連する極プログラミング			G110 X.. Y.. Z..	s	3
G140[1]	1 G41/G42 によって定義された アプローチ WAB の方向				m	43
G141	輪郭の左のアプローチ WAB の方向				m	43
G142	輪郭の右のアプローチ WAB の方向				m	43
G143	タンジェントに依存するアプローチ WAB の方向				m	43
G147	直線でスムースにアプローチ				s	2
G148	直線でスムースに後退				s	2
G153	ベースフレームを含む現在のフレームを抑止				s	9
G247	4 分の 1 円弧でスムースにアプローチ				s	2
G248	4 分の 1 円弧でスムースに後退				s	2
G331	タッピング	± 0.001, ...,	モーション		m	1
G332	後退 (タッピング)	2000.00 mm/rev	コマンド		m	1
G340 [1]	空間的なブロックへのアプローチ (深さおよび同時に平面において (らせん)		スムースなアプローチと後退のために		m	44
G341	垂直軸 (z) でアプローチしてから平面でアプローチする		スムースなアプローチと後退のため		m	44
G347	半円でスムースなアプローチ				s	2
G348	半円でスムースな後退				s	2
G450 [1]	遷移円		ツール補正応答		m	18
G451	コーナにおける等距離パスの補間				m	18
G460[1]	TRC でアプローチ／後退挙動				m	48
G461	TRC でアプローチ／後退挙動				m	48
G462	TRC でアプローチ／後退挙動				m	48
G500 [1]	G500 で値がない場合，すべての設定可能なフレームを停止				m	8
G505 .... G599	5. ... 99. 設定可能なゼロオフセット				m	8
G601 [1]	イグザクトストップ（微）に応答してブロック変更		プログラム可能な遷移を丸めて，有効な G60 または G9 に関連してのみ有効		m	12
G602	イグザクトストップ (粗）に応答してブロック変更				m	12
G603	ブロックの IPO 終りに応答してブロックチェンジ				m	12
G641	イグザクトストップ - 輪郭モード			G641 ADIS=...	m	10
G642	軸の精度で丸め				m	10
G700	インチおよびインチ／分での寸法				m	13
G710 [1]	ミリメートルおよびミリメートル／分でメートルによる寸法				m	13

G810[1], ..., G819	OEM ユーザ用に確保された G グループ					31
G820[1], ..., G829	OEM ユーザ用に確保された G グループ					32
GEOAX	ジオメトリ軸 1 - 3 に新しいチャンネル軸を割当てる		パラメータなし : MD 設定値有効			
GET	マシン軸を割当てる		軸は，必ず，RELEASE を使って他のチャンネルで解除すること。			
GETD	直接マシン軸を割当てる		GET 参照			
GETACTT	名前の同じツールのグループから有効なツールを得る					
GETSELT	選択された T 番号を得る					
GETT	ツール名についての T 番号を得る					
GOTOF	命令を前方にジャンプ (プログラムのエンドに向かって)					
GOTOB	命令を後方にジャンプ (プログラムのスタートに向かって)					
GWPSOF	一定のグラインディングホイール周辺速度を選択解除する (GWPS)			GWPSOF (T No.)	s	
GWPSON	一定のグラインディングホイール周辺速度を選択する (GWPS)			GWPSON (T No.)	s	
H...	PLC への補助機能出力	実数 /INT	MD (機械メーカー) を介して設定可能	H100 または H2=100		
I[4]	補間パラメータ	実数			s	
I1	補間点座標	実数			s	
IC	相対寸法設定	0, ..., ± 99999.99°		X=IC(10)	s	
IDS	静的シンクロナイズドアクションの識別					
IF	条件付きのジャンプを導く		構造 : IF - ELSE - ENDIF			
INDEX	入力ストリングに文字のインデックスを定義する	0, ..., INT	ストリング : パラメータ 1, 文字 : パラメータ 2			
INIT	チャンネルでの実行用ブロックを選択する					
INT	リーディング符号をもつデータタイプ整数	- (2^{31} -1), ..., 2^{31} -1				
INTERSEC	２つの輪郭要素間の補間を計算する	VAR 実数 [2]	エラーステータス BOOL			
IP	変数補間パラメータ	実数				
ISAXIS	パラメータとして指定されたジオメトリ軸 1 - 3 が存在するかどうかチェックする	BOOL				
ISD	挿入深さ	実数			m	

ISNUMBER	入力ストリングが数字に変換できるかどうかチェックする	BOOL				
J[4]	補間パラメータ	実数			s	
J1	中間点座標	実数			s	
JERKA	プログラムされた軸用マシンデータを介してセットされた加速応答を起動する					
K[4]	補間パラメータ	実数			s	
K1	中間点座標	実数			s	
KONT	ツール補正用輪郭の周囲を移動する				m	17
L	サブプログラム番号	整数，最高 7 位まで	リーディング用ゼロが関連する！	L10	s	
LEAD	リード角	実数			m	
LEADOF	リーディング値カップリング OFF (リーディングオフ)					
LEADOFP	パスリーディング値カップリング OFF (パスリーディングオフ)					
LEADON	リーディング値カップリング ON (リーディングオン)					
LEADONP	パスリーディング値カップリング ON (パス上のリーディング)					
LFOF[1]	スレッド切削の中断 OFF				m	41
LFON	スレッド切削の中断 ON				m	41
LFTXT[1]	リフト時ツール方向がタンジェント				m	46
LFWP	上昇時にツール方向がタンジェンシャルでない				m	46
LIFTFAST	中断ルーチンコール前の急速上昇					
LIMS	G96 で，主軸速度制限 (制限主軸速度)	0.001 ... 99 999.999			m	
LN	自然対数	実数				
LOCK	ID でシンクロナイズドアクションをディスエーブルにする (テクノロジサイクルを停止)					
LOG	(一般) 対数	実数				
LOOP	エンドレスループの採用		構造 : LOOP - ENDLOOP			
M...	切換え操作	0, ..., 9999 9999	最大 5 つのフリーな特別機能機械メーカーが定義する			
M0[6]	プログラムされた停止					
M1[6]	オプショナルの停止					
M2[6]	プログラムエンド，プログラムのスタートにリセットするメインプログラム					
M3	マスタ主軸用右主軸回転					
M4	マスタ主軸用左回り主軸回転					
M5	マスタ主軸用主軸停止					
M6	ツール交換					

M17 [6]	サブプログラムのエンド					
M30 [6]	プログラムエンド，M2 の場合					
M40	自動ギアチェンジ					
M41... M45	ギアステージ 1, ..., 5					
M70	軸運転に遷移					
MCALL	モーダルでサブプログラムをコール		サブプログラム名なし : 選択解除			
MEAC	残移動量削除せずに連続測定	整数 , 符号なし			s	
MEAFRAME	測定点からフレーム計算	FRAME				
MEAS	タッチトリガプローブで測定 (測定)	整数 , 符号なし			s	
MEASA	残移動量削除をして測定				s	
MEAW	残移動量削除せずにタッチトリガプローブを用いて測定 (残移動量削除なしの測定 o)	整数 , 符号なし			s	
MEAWA	残移動量削除せずに測定				s	
MI	フレームデータにアクセス : ミラリング					
MINDEX	入力ストリングの文字インデックスを定義する	0, ...,INT	ストリング : パラメータ 1, 文字 : パラメータ 2			
MIRROR	プログラム可能なミラー			MIRROR X0 Y0 Z0 ; 個別ブロック	s	3
MMC	MMC コマンドインタプリータへのコマンド	STRING				
MOD	モジューロ割算					
MOV	位置決め軸を始動する (移動位置決め軸スタート)	実数				
MSG	プログラム可能なメッセージ			MSG("message")	m	
MSGN	MMC テキスト番号を指定してテキストを示す (メッセージ番号)					
MSGS	単一ブロックモードでメッセージ (メッセージ単一ステップ)		次の NC スタートへ, 次のブロックを実行中			
N	サブブロック番号	0, ...,9999 9999 整数値のみ，符号なし	番号でブロックを識別するために用いることができる ; ブロック始めの位置	E.g. N20		
NCK	データの有効範囲を指定する		NCK につき 1 度			
NEWCONF	修正されたマシンデータを受け取る					
NEWT	新しいツールを作成する		デュプロ番号は省略可能			
NORM [1]	ツールオフセットについての始点および終点における標準設定				m	17
NOT	論理 NOT (否定)					
NPROT	マシン別保護ゾーン ON/OFF					

NPROTDEF	マシン別保護エリア定義 (NCK 別保護エリア定義)					
NUMBER	入力ストリングを数字に変換する		実数			
OEMIPO1 [5]	OEM 補間 1				m	1
OEMIPO2 [5]	OEM 補間 2				m	1
OF	CASE ブランチにおけるボキャブラリワード					
OFFN	プログラムされた輪郭の許容値			OFFN=5		
OMA1	OEM アドレス 1	実数			m	
OMA2	OEM アドレス 2	実数			m	
OMA3	OEM アドレス 3	実数			m	
OMA4	OEM アドレス 4	実数			m	
OMA5	OEM アドレス 5	実数			m	
OFFN	オフセット補正 - 標準	実数			m	
OR	論理的 OR					
ORIC [1]	外側コーナでの向き変更が入力される円ブロックにオーバレイされる (連続的に向き変更)				m	27
ORID	円ブロックの前で向き変更が実行される (不連続的に方向変更)				m	27
ORIEULER	Euler 角を使って向き調整角度				m	50
ORIMACHAX	マシン軸あるいは向き調整軸の直線補間				m	51
ORIMKS	マシン座標系でのツール向き				m	25
ORIPATH	ツール向きパス				m	51
ORIRPY	RPY 角を使って向き調整角度				m	50
ORIS	向き変更 (向きスムーシング係数)	実数	パスに従う		m	
ORIVIRT1	仮想の向き調整軸を使って向き角度 (定義 1)				m	50
ORIVIRT2	仮想の向き調整軸を使って向き角度 (定義 1)				m	50
ORIVIRTAX	大円補間				m	51
ORIWKS [1]	ワーク座標系におけるツール向き				m	25
OS	揺動 ON / OFF	整数，符号なし				
OSC	ツール向きのための一定スムーシング				m	34
OSCILL	発信のための軸割当て - 揺動を起動する		軸 : 1 - 3 インフィード軸		m	
OSCTRL	揺動制御オプション	整数，符号なし			m	
OSE	揺動 : 終点				m	
OSNSC	揺動 : スパークアウトサイクルの数				m	
OSOF [1]	ツール向きのための一定スムーシング OFF				m	34

OSP1	揺動 : 左向きの逆転点 (揺動 : 位置 1)	実数			m	
OSP2	揺動 : 右向きの逆転点 (揺動 : 位置 2)	実数			m	
OSS	ブロックの終りでのツール向きスムーシング				m	34
OSSE	ブロックの始めと終りでのツール向きスムーシング				m	34
OST1	揺動 : 左向きの逆転点で停止	実数			m	
OST2	揺動 : 右向きの逆転点で停止	実数			m	
OVR	主軸オーバライド	1, ..., 200%			m	
OVRA	軸主軸オーバライド	1, ..., 200%			m	
P	サブプログラム通過数	1 ... 9999, 整数 符号なし n		例 . L781 P...; 個別ブロック		
PDELAYOF	パンチングのための遅延 OFF (遅延 OFF にしてパンチ)				m	36
PDELAYON[1]	パンチングのための遅延 ON (遅延 ON にしてパンチ)				m	36
PL	パラメータのインターバル長さ	実数 , 符号なし			s	
PM	１ 分当たり			1 分当たりのフィード		
PO	多項式	実数 , 符号なし			s	
POLF	位置 LIFTFAST	実数 , 符号なし		POLF[Y]=10	m	
POLY	多項式補間				m	1
POS	軸を位置決めする			POS[X]=20		
POSA	ブロック境界を越えて軸を位置決めする			POSA[Y]=20		
POSP	パートセクションでの位置決め (揺動) (パーツで軸を位置決め)	実数 : 最終位置 , パート長さ ; 整数 : オプション				
POT	二乗 (算術機能)	実数				
PR	回転当たり			回転フィードレート		
PRESETON	プログラムされた軸について実際値をセットする		次のパラメータで該当する値を使って, 軸名をプログラムする。最大 8 軸が可能	PRESETON(X,10, Y,4.5)		
PRIO	割込み処理の優先順位をセットするためのボキャブラリワード					
PROC	プログラムにおける最初の命令			ブロック番号 - PROC - 識別子		
PTP	点から点への移動				m	49

PUTFTOC	並列ドレッシングのためのツールオフセット (微) (連続ドレッシング)					
PUTFTOCF	ツール訂正機能 (微) を従属にする : FCtDEF を使って定義された連続ドレッシング用の機能に依存するツールオフセット (微)					
PW	点重量	実数 , 符号なし			s	
QU	高速追加 (補助) 機能出力					
R...	算術パラメータ : 設定可能なアドレス識別子として, および数的拡張を用いて	± 0.0000001, ..., 9999 9999	R パラメータ番号は MD を介してセットできる。	R10=3 ;R パラメータ割当て X=R10 ; 軸値 R[R10]=6 ; 間接プログラミング		
RDISABLE	読み込みディスエーブル					
READAL	アラームを読取る		アラームが昇順サーチされる			
実数	リーディング符号を持つデータタイプ浮動小数点式変数 (実数)	プロセッサの 64 ビット浮動小数点フォーマットに対応する				
REDEF	マシンデータ用設定，これについてのユーザグループが，表示される					
RELEASE	マシン軸を解除する		マルチ軸がプログラム可能			
REP	同値で，配列のすべての要素を初期化するためのボキャブラリワード					
REPEAT	プログラムループを繰り返す		条件が満たされるまで (UNTIL)			
REPEATB	プログラム行を繰り返す		nnn 回			
REPOSA	すべての軸を直線に再位置決めする				s	2
REPOSH	半円に沿って再位置決めする				s	2
REPOSHA	すべての軸を半円に沿って再位置決めする : すべての軸，ジオメトリ軸を 4 分の 1 円弧に沿って再位置決めする :				s	2
REPOSL	直線で再位置決めする				s	2
REPOSQ	4 分の 1 円弧に沿って再位置決めする				s	2
REPOSQA	4 分の 1 円弧に沿ってすべての軸を再位置決めする すべての軸を直線で，ジオメトリ軸を 4 分の 1 円弧に沿って再位置決めする				s	2
RESET	テクノロジサイクルをリセットする		1 つまたは複数の ID をプログラムできる			
RET	サブプログラムのエンド		連続パスモードを維持するために，M2 の代わりに使用	RET		
RINDEX	入力ストリングに文字のインデックスを定義する	0, ...,INT	ストリング : パラメータ 1, 文字 : パラメータ 2			

RMB	ブロックの始めに再位置決めする (Repos モードのブロック始め)				m	26
RME	ブロックの終りに再位置決めする (Repos モードのブロック終り)				m	26
RMI [1]	中断点で再位置決めする (Repos モード中断)				m	26
RND	輪郭コーナを丸める	実数 , 符号なし		RND=...	s	
RNDM	モーダル丸め	実数 , 符号なし		RNDM=... RNDM=0: モーダル丸めを停止する	m	
ROT	プログラム可能な回転	1 番目のジオメトリ軸 : -180° .. 180° 2 番目のジオメトリ軸 : -89.999° ,..., 90° 3 番目のジオメトリ軸 : -180° .. 180° の回りを回転		ROT X... Y... Z... ROT RPL= ; 個別ブロック	s	3
ROUND	少数位を丸める	実数				
RP	極半径 (半径　極)	実数			m,s [3]	
RPL	平面での回転 (回転面)	実数 , 符号なし			s	
RT	フレームデータへのアクセス用パラメータ : 回転					
S	主軸速度あるいは (G4, G96 を使って) 別の意味	0.1 ... 99999999	rev/min で表した主軸速度 G4: 主軸回転の滞在時間 G96: m/min で表した切削速度	S...: マスタ主軸の主軸速度 S1...: 主軸 1 の主軸速度	m,s	
SAVE	サブルーチンコールで情報を保存するための特性		以下の情報が保存される : すべてのモーダル G 機能および現在のフレーム			
SBLOF	単一ブロックを抑止する (単一ブロック OFF)		1 つのブロックのように単一ブロック内で，次のブロックが実行される。			
SBLON	単一ブロック抑止を消去する (単一ブロック ON)					
SC	フレームデータへのアクセス用パラメータ : スケーリング (スケール)					
SCALE	プログラム可能なスケーリング (スケール)			SCALE X... Y... Z... ; 個別ブロック	s	3

SD	スプライン度数	整数 , 符号なし			s	
SET	列挙された値の配列のすべての要素を初期化するためのボキャブラリワード					
SETAL	アラームをセットする					
SETDNO	ツール (T) の D 番号およびその切削エッジを新しくセットする					
SETINT	NCK 入力の表示時，どの割込みルーチンを起動するかを定義する		エッジ 0 が評価される→ 1			
SETM	チャンネル座標用マーカ（1 つまたは複数）をセットする		ローカルチャンネルでの加工には影響しない。			
SETMS	マシンデータにプログラムされたマスタ主軸に戻る					
SETMS(n)	主軸 n は，必ず，マスタ主軸として機能すること。					
SETPIECE	主軸に割当てられたすべてのツールの個数をセットする。		主軸番号なし : マスタ主軸について有効			
SF	スレッド切削のための点オフセットを開始する (スプラインオフセット)	0.0000, ..., 359.999°			m	
SIN	サイン (三角関数)	実数				
SOFT	ソフト軸加速				m	21
SOFTA	プログラムされた軸のためのソフト軸加速をスイッチオンにする					
SPATH [1]	FGROUP 軸のパス基準はアーク長さである				m	45
SPCOF	マスタ主軸または主軸 (n) を速度制御から位置制御へ切替える			SPCON SPCON (n)		
SPCON	マスタ主軸または主軸 (n) を位置制御から速度制御に切替える			SPCONSPCON (n)		
SPLINE PATH	スプライングルーピングを定義する		最大 8 軸			
SPN	ブロック当たりのパスセクションの数 (ストローク / パンチの数)	整数			s	
SPP	パスセクションの長さ (ストローク / パンチのパス)	整数			m	
SPOS	主軸の位置			SPOS=10 または SPOS[n]=10	m	
SPOSA	ブロック境界を越えた主軸位置			SPOSA=5 または SPOSA[n]=5	m	
SQRT	平方根 ; 算術機能	実数				
SR	後退パス (スパークアウト後退パス)	実数 , 符号なし			s	
SRA	外部入力での後退パス，軸について (スパークアウト後退)			SRA[Y]=0.2	m	

ST	スパークアウト時間	実数 , 符号なし			s	
STA	スパークアウト時間，軸について				m	
START	現在のプログラムから複数のチャンネルで同時に，選択プログラムを開始する		ローカルチャンネルには無効			
STAT	関節ジョイントの位置	整数			s	
STARTFIFO [1]	実行する ; 並行して前処理バッファを満たす				m	4
STOPFIFO	処理を停止する ; STARTFIFO，前処理バッファ "full" またはプログラムのエンドを見つけるまで前処理バッファを満たす,				m	4
STOPRE	すべての準備されたブロックがメインランで実行されるまで前処理を停止する					
STOPREOF	前処理停止 OFF					
STRING	データタイプ : ストリング	最大 200 文字				
STRLEN	ストリング長さを定義する	INT				
SUBSTR	入力ストリングに文字のインデックスを定義する	実数	ストリング : パラメータ 1, 文字 : パラメータ 2			
SUPA	現在のゼロオフセットを抑止		プログラム，オフセット，ハンドルオフセット (DRF), 外部ゼロオフセットおよび PRESET オフセットを含む		s	9
SYNFCT	モーションシンクロナイズドアクションで条件の機能として多項式を評価	VAR REAL				
SYNR	変数は，同期して，つまり実行時間に読取られる (同期読取り)					
SYNRW	変数は読取られ，同期して，つまり実行時間に書き込まれる (同期読取り－書込み)					
SYNW	変数は，同期して，つまり実行時間に書き込まれる (同期書込み)					
T	ツールをコールする (マシンデータにそのように定義してあれば，変更する ; そうでなければ，M6 コマンドが必要になる)	1 ... 32 000	T 番号を介してコール : またはツール名でコール :	E.g. T3 または T=3 E.g. T="DRILL"		
TAN	タンジェント (三角関数)	実数				
TANG	指定された両方のリーディング軸からフォローアップのためのタンジェントを定義する					
TANGOF	タンジェントフォローアップモード OFF					
TANGON	タンジェントフォローアップモード ON					

TCARR	ツールホルダをリクエストする (番号 "m")	整数	m=0: 有効なツールホルダを選択解除する	TCARR=1		
TCOABS	1 現在のツール向きからツール長さの構成要素を定義する		マシンをリセット後，たとえば手動設定で，要求する		m	42
TCOFR	有効なフレームの向きからツール長さ構成要素を定義する				m	42
TILT	サイド角	実数			m	
TMOF	ツール監視機能を選択解除する		T 番号のあるツールが有効でない場合のみ，T 番号が要求される。 T No. = 0: すべてのツールについて監視機能を停止する	TMOF (T No.)		
TMON	ツール監視機能を選択する			TMON (T No.)		
TO	FOR カウンタループでのエンド値を定義する					
TOFRAME	現在のプログラム可能なフレームをツール座標系にセットする				s	3
TOLOWER	ストリングの文字を小文字に変換する					
TOUPPER	ストリングの文字を大文字に変換する					
TR	フレームデータへのアクセス用パラメータ : 翻訳					
TRAANG	変換傾斜軸		チャンネルごとに設定できる複数の変換			
TRACEOF	円形性テスト : 数値の転送 OFF					
TRACEON	円形性テスト : 数値の転送 ON					
TRACON	連結した変換					
TRACYL	円柱 : 周囲表面変換		TRAANG 参照			
TRAFOOF	変換をオフにする			TRAFOOF( )		
TRAILOF	軸の同期カップリングされたモーション OFF(トレイリング OFF)					
TRAILON	軸の同期カップリングされたモーション ON(トレイリング ON)					
TRANS	プログラム可能なオフセット (翻訳)			TRANS X. Y. Z. ; 個別ブロック	s	3
TRANSMIT	極変換		TRAANG 参照			
TRAORI	4- 軸 , 5- 軸変換 (方向付けされた変換)		TRAANG 参照			
TRUE	論理定数 : 真	BOOL	整数定数 1 で置き換え可能			
TRUNC	少数位を切り捨てる	実数				
TU	軸角度	整数		TU=2	s	
TURN	らせんの順序	0, ..., 999			s	
UNLOCK	ID を使ってシンクロナイズドアクションをイネーブルにする (テクノロジサイクルを継続する)					

UNTIL	REPEAT ループのエンドのための条件					
UPATH	カーブパラメータは, FGROUP 軸のパス基準である				m	45
VAR	ボキャブラリワード : パラメータの転送のタイプ		VAR を使って : リファレンスでコールする			
WAITC	軸／主軸のためのカップリングブロックチェンジ基準が満たされるまで，待機する (カップリング状態を待つ)		最大 2 軸 / 主軸がプログラム可能。	WAITM(1,1,2)		
WAITM	指定チャンネルでマーカを待つ ; イグザクトストップで前ブロックを終了する			WAITM(1,1,2)		
WAITMC	指定チャンネルでマーカを待つ ; 他のチャンネルがまだマーカに到達してない場合のみイグザクトストップ			WAITMC(1,1,2)		
WAITP	移動の終りを待つ			WAITP(X) ; 個別ブロック		
WAITS	主軸位置に到達するまで待つ			WAITS (メイン主軸) WAITS (n,n,n)		
WALIMOF	作業エリア制限 OFF			; 個別ブロック	m	28
WALIMON [1]	作業エリア制限 ON			; 個別ブロック	m	28
WHILE	WHILE プログラムループの開始		終り : ENDWHILE			
WRITE	ファイルシステムにブロックを書き込む					
X	軸	実数			m,s [3]	
XOR	排他的論理和 OR					
Y	軸	実数			m,s [3]	
Z	軸	実数			m,s [3]	

凡例 :

1 プログラム開始時の初期設定 (別の設定がプログラムされていない場合，制御システムのデリバリステータスにおいて).

2 グループの番号は，テーブル "List of G functions/preparatory functions" （G 機能／準備機能のリスト）に対応します。

3 絶対終点 : モーダル ; 相対的終点 : ノンモーダル ; その他の場合，モーダル / ノンモーダルは G 機能の構文でによって決まります。

4 IPO パラメータはアークセンタとして増分で機能します。AC を用い，絶対モードでプログラム可能。他の意味を持つとき（たとえば，ピッチ），アドレス修正は無視されます。

5 OEM ユーザは，２種類の別の補間タイプを組み込んで，それらの名称を変更することができます。

6 拡張アドレスブロックフォーマットは，これらの機能用に使用することはできません。

14.2 システム変数のリスト （パートプログラム）

ツールパラメータ

	名称	タイプ	コメント
ツールオフセットパラメータ	$TC_DPx[t,d]		t: T 番号 | 1 - 32000 または MD を介して定義される d: ツールエッジ番号 / D 番号 | 1 - 9 x: パラメータ数が MD にセットされる
	$TC_DP1	INT	ツールタイプ
	$TC_DP2	REAL	ツールエッジ位置
	$TC_DP3		ジオメトリ - 長さ 1
	$TC_DP4		ジオメトリ - 長さ 2
	$TC_DP5		ジオメトリ - 長さ 3
	$TC_DP6		ジオメトリ - 半径
	$TC_DP7		ジオメトリ - スロット幅 b/ フィレット半径
	$TC_DP8		ジオメトリ - オーバハング k
	$TC_DP11		ジオメトリ - 円錐フライスツールの角度
	$TC_DP12		磨耗 - 長さ 1
	$TC_DP13		磨耗 - 長さ 2
	$TC_DP14		磨耗 - 長さ 3
	$TC_DP15		磨耗 - 半径
	$TC_DP16		磨耗 - スロット幅 b/ フィレット半径
	$TC_DP17		磨耗 - オーバハング k
	$TC_DP20		磨耗 - 円錐フライスツールの角度
	$TC_DP21		ベース - 長さ 1
	$TC_DP22		ベース - 長さ 2
	$TC_DP23		ベース - 長さ 3
	$TC_DP24		クリアランス角度
切削エッジデータ，OEM ユーザ	$TC_DPCx[t,d]	REAL	t: T 番号 | 1 - 32000 または MD を介して定義される d: ツールエッジ番号 / D 番号 | 1 - 9 x: パラメータ数が MD にセットされる
	$TC_DPC1		
	$TC_DPC2		
	$TC_DPC3		
	$TC_DPC4		
	$TC_DPC5		
	$TC_DPC6		
	$TC_DPC7		
	$TC_DPC8		
	$TC_DPC9		

	名称	タイプ	コメント
	$TC_DPCE	INT	$TC_DPCE[t,d] = オフセットデータブロック t,d の ' ツールエッジ番号 ' 有効な ' フラット D 番号管理 ' 機能を用いて , 構文は次のとおり : $TC_DPCE[d] CE は <C>utting<E>dget を示す : T 番号 1 - SLMAXTOOLNUMBERd: ツールエッジ番号 / D 番号 1 - SLMAXCUTTINGEDGENUMBER
ツール向けのグラインディングデータ	$TC_TPGx[t]	REAL	t: T 番号 | 1 - 32000
	$TC_TPG1 I	NT	主軸番号
	$TC_TPG2		チェーニングルール
	$TC_TPG9		半径計算用のパラメータ番号
	$TC_TPG3	REAL	最小グラインディングホイール半径
	$TC_TPG4		最小グラインディングホイール幅
	$TC_TPG5		現在のグラインディングホイール幅
	$TC_TPG6		最大回転速度
	$TC_TPG7		最大表面速度
	$TC_TPG8		斜めグラインディングホイールの傾斜角
データツール管理の監視	$TC_MOPx[t,d]	INT	t: T 番号 | 1 - 32000 d: ツールエッジ番号 / D 番号 | 1 - 9
	$TC_MOP1		ツール寿命の前警告リミット
	$TC_MOP2		残りのツール寿命
	$TC_MOP3		ワーク数の前警告リミット
	$TC_MOP4		残りのワーク番号
	$TC_MOP5[t,d]		限界磨耗を事前警告 t: T 番号 1 - 3200 または MD を介して定義される d: ツールエッジ番号 / D 番号 1 - SLMAXCUTTINGEDGENUMBER
	$TC_MOP6[t,d]		残りの磨耗 t: T 番号 1 - 3200 または MD を介して定義される d: ツールエッジ番号 / D 番号 1 -SLMAXCUTTINGEDGENUMBER
	$TC_MOP11[t,d]		使用期間セットポイント t: T 番号 1 - 3200 または MD を介して定義される d: ツールエッジ番号 / D 番号 1 - SLMAXCUTTINGEDGENUMBER
	$TC_MOP13[t,d]		数量セットポイント t: T 番号 1 - 3200 MD を介して定義される d: ツールエッジ番号 / D 番号 1 -SLMAXCUTTINGEDGENUMBER
	$TC_MOP15[t,d]		磨耗セットポイント t: T 番号 1 - 3200 MD を介して定義される d: ツールエッジ番号 / D 番号 1 -SLMAXCUTTINGEDGENUMBER

	名称	タイプ	コメント
監視データ，OEM ユーザ	$TC_MOPCx[t,d]	INT	t: T 番号 |1 - 32000 d: ツールエッジ番号 / D 番号 | 1 - 9 x: パラメータ数が MD にセットされる
	$TC_MOPC1		
	$TC_MOPC2		
	$TC_MOPC3		
	$TC_MOPC4		
	$TC_MOPC5		
	$TC_MOPC6		
	$TC_MOPC7		
	$TC_MOPC8		
	$TC_MOPC9		
ツール関連データ ツール管理	$TC_TPx[t]		t: T 番号 1 - 32000
	$TC_TP2	STRING	ツール識別子
	$TC_TP1 I	NT	デュプロ番号
	$TC_TP3 S		左への寸法
	$TC_TP4		右への寸法
	$TC_TP5		上への寸法
	$TC_TP6		下への寸法
	$TC_TP7		マガジンロケーションタイプ
	$TC_TP8	INT	ステータス
	$TC_TP9		ツール監視のタイプ
	$TC_TP11		交換方法
	$TC_TP10		ツール情報
ツール関連データ，OEM ユーザ	$TC_TPCx[t]	REAL	t: T 番号 1 - 32000 x: パラメータ数は，MD にセットされる。
	$TC_TPC1		
	$TC_TPC2		
	$TC_TPC3		
	$TC_TPC4		
	$TC_TPC5		
	$TC_TPC6		
	$TC_TPC7		
	$TC_TPC8		
	$TC_TPC9		
	$TC_TPC10		

	名称	タイプ	コメント
マガジンロケーションデータ ツール管理	$TC_MPPx[n,m]		n: 物理マガジン番号 m: 物理ロケーション番号
	$TC_MPP3	BOOL	隣接ロケーション考慮 オン / オフ
	$TC_MPP1	INT	ロケーションクラス
	$TC_MPP2		ロケーションタイプ
	$TC_MPP6		このロケーションのツールの T 番号
	$TC_MPP4		ロケーションステータス
	$TC_MPP5		ロケーションクラスインデックス
	$TC_MPP7		このロケーション番号でのツールアダプタのアダプタ番号 : 物理マガジン番号 : 物理ロケーション番号
マガジンロケーションデータ OEM ユーザ	$TC_MPPCx[n,m]	INT	n: 物理マガジン番号 m: 物理ロケーション番号 x: パラメータ数が MD にセットされる
	$TC_MPPC1		
	$TC_MPPC2		
	$TC_MPPC3		
	$TC_MPPC4		
	$TC_MPPC5		
	$TC_MPPC6		
	$TC_MPPC7		
	$TC_MPPC8		
	$TC_MPPC9		
	$TC_MPPC10		
マガジンデータ : 変更位置からの距離	$TC_MDPx[n,m]	INT	n: 物理マガジン番号 m: 物理ロケーション番号
	$TC_MDP1		1 番目の内部マガジンのロケーション m からマガジンの変更ロケーション n までの距離
	$TC_MDP2		2 番目の内部マガジンのロケーション m からマガジンの変更ロケーション n までの距離
マガジンデータ : マガジンロケーションタイプの階層構造	$TC_MPTH[n,m]	INT	マガジンロケーションの階層構造 n: 階層構造 0 から 7 m: ロケーションタイプ 0 から 7
マガジン説明データ ツール管理	$TC_MAPx[n]		n: マガジン番号 1 から ... まで
	$TC_MAP2	STRING	マガジンに対応する識別子
	$TC_MAP1	INT	マガジンのタイプ
	$TC_MAP3		マガジンのステータス
	$TC_MAP4		次のマガジンとチェーニング

	名称	タイプ	コメント
	$TC_MAP5		前のマガジンとチェーニング
	$TC_MAP6		行数
	$TC_MAP7		列数
	$TC_MAP8		交換位置に関連する現在のマガジン位置
	$TC_MAP9		現在の磨耗グループ数: マガジン番号 1 から ... まで
	$TC_MAMPx[n]		マガジン番号 1 から ... まで
	$TC_MAMP1	STRING	マガジンモジュールに対応する識別子
	$TC_MAMP2	INT	ツールサーチのタイプおよびフリーロケーションサーチのタイプ
	$TC_MAMP3	INT	磨耗グループスカラ変数を用いたツールのハンドリング
マガジン説明データ OEM ユーザ	$TC_MAPCx[n]	INT	n: マガジン番号 1 - ... x: パラメータ数が MD にセットされる
	$TC_MAPC1		
	$TC_MAPC2		
	$TC_MAPC3		
	$TC_MAPC4		
	$TC_MAPC5		
	$TC_MAPC6		
	$TC_MAPC7		
	$TC_MAPC8		
	$TC_MAPC9		
	$TC_MAPC10		

マガジンデータ

	名称	タイプ	コメント
マガジンロケーションと主軸の関係	$TC_MLSR[n,m]=0	INT	バッファロケーション n とバッファロケーション m との間の割当て : n: 物理マガジンロケーション番号 ., ロケーションタイプ ? ' 主軸 ', m: 物理マガジンロケーション番号 ., ロケーションタイプ = ' 主軸 ' この割当て方法を使って，ユーザは，たとえば，グリッパをどの主軸に割当てるかを定義することができる パラメータ値がゼロに固定される。 書込み操作で，関係を定義付け，読取り操作で，指定した関係が存在しているかどうかをチェックする。存在しなければ，読取り中にアラームが出力される : " 主軸とグリッパをリンクすることを定義する "。

ツール管理

名称	タイプ	アクセス		意味 | 値の範囲	インデックス | 値の範囲
		パートプログラム	シンクロナイズドアクション		
$AC_TC_FCT	INT	R/W	R/W	コマンド番号 . どのアクションが必要かを指定する	
$AC_TC_LFN	INT	R/W	R/W	新規ツールのソースロケーション番号 0: 新規ツールはなし。	
$AC_TC_LFO	INT	R/W	R/W	古いツールのソースロケーション番号 (取替えられる).0: 古いツールはなし。	
$AC_TC_LTN	INT	R/W	R/W	新規ツールのターゲットロケーション番号 0: 新規ツールはなし。	
$AC_TC_LTO	INT	R/W	R/W	古いツールのターゲットロケーション番号 (取替えられる). 0: 古いツールはなし。	
$AC_TC_MFN	INT	R/W	R/W	新規ツールのソースマガジン番号 0: 新規ツールはなし。	
$AC_TC_MFO	INT	R/W	R/W	古いツールのソースマガジン番号 (取替えられる). 0: 古いツールはなし。	
$AC_TC_MTN	INT	R/W	R/W	新規ツールのターゲットマガジン番号 0: 新規ツールはなし。	
$AC_TC_MTO	INT	R/W	R/W	古いツールのターゲットマガジン番号 (取替えられる). 0: 古いツールはなし	
$AC_TC_STATUS	INT	R/W	R/W	コマンドステータス - $AC_TC_FCT を介して読取る	
$AC_TC_THNO	INT	R/W	R/W	新規ツールが変換されるツールホルダ番号 (パート主軸番号中の)	
$AC_TC_TNO	INT	R/W	R/W	新規ツールの NCK-INTernal T 番号 (取替えられる). 0: 新規ツールはなし。	
$TC_ECP13	DOUBLE	R/W		$TC_DP3 のオフセット : $TC_DP12[t,d] に類似した $TC_ECP13[t,d]	t: T 番号 1 - MD (SLMAXTOOLNUMBER) を介して定義される

名称	タイプ	アクセス		意味 | 値の範囲	インデックス | 値の範囲
		パートプログラム	シンクロナイズドアクション		
				有効な ' フラット D 番号管理 ' 機能を用いて , 構文は次のとおり : $TC_ECP13[d]t: T 番号 | 1 -32000 MD を介して定義される d: ツールエッジ番号 / D 番号 1 - n (MD を介して定義された n)	
$TC_ECP14[t,d]	DOUBLE	R/W		$TC_DP4 のオフセット : $TC_DP13[t,d] に類似	
$TC_ECP15[t,d]	DOUBLE	R/W		$TC_DP5 のオフセット , $TC_DP14[t,d] に類似	
$TC_ECP16[t,d]	DOUBLE	R/W		$TC_DP6 のオフセット , $TC_DP15[t,d] に類似	
$TC_ECP17[t,d]	DOUBLE	R/W		$TC_DP7 のオフセット , $TC_DP16[t,d] に類似	
$TC_ECP18[t,d]	DOUBLE	R/W		$TC_DP8 のオフセット , $TC_DP17[t,d] に類似	
$TC_ECP19[t,d]	DOUBLE	R/W		$TC_DP9 のオフセット , $TC_DP18[t,d] に類似	
$TC_ECP20[t,d]	DOUBLE	R/W		$TC_DP10 のオフセット, $TC_DP19[t,d] に類似	
$TC_ECP21[t,d]	DOUBLE	R/W		$TC_DP11 のオフセット, $TC_DP20[t,d] に類似	
$TC_ECP23[t,d]	DOUBLE	R/W		$TC_DP3 のオフセット , $TC_DP12[t,d] に類似	
$TC_ECP24[t,d]	DOUBLE	R/W		$TC_DP4 のオフセット , $TC_DP13[t,d] に類似	
$TC_ECP25[t,d]	DOUBLE	R/W		$TC_DP5 のオフセット , $TC_DP14[t,d] に類似	
$TC_ECP26[t,d]	DOUBLE	R/W		$TC_DP6 のオフセット , $TC_DP15[t,d] に類似	
$TC_ECP27[t,d]	DOUBLE	R/W		$TC_DP7 のオフセット , $TC_DP16[t,d] に類似	
$TC_ECP28[t,d]	DOUBLE	R/W		$TC_DP8 のオフセット , $TC_DP17[t,d] に類似	
$TC_ECP29[t,d]	DOUBLE	R/W		$TC_DP9 のオフセット , $TC_DP18[t,d] に類似	
$TC_ECP30[t,d]	DOUBLE	R/W		$TC_DP10 のオフセット, $TC_DP19[t,d] に類似	
$TC_ECP31[t,d]	DOUBLE	R/W		$TC_DP11 のオフセット, $TC_DP20[t,d] に類似	
$TC_ECP33[t,d]	DOUBLE	R/W		$TC_DP3 のオフセット , $TC_DP12[t,d] に類似	

名称	タイプ	アクセス		意味 | 値の範囲	インデックス | 値の範囲
		パートプログラム	シンクロナイズドアクション		
$TC_ECP34[t,d]	DOUBLE	R/W		$TC_DP4 のオフセット, $TC_DP13[t,d] に類似	
$TC_ECP35[t,d]	DOUBLE	R/W		$TC_DP5 のオフセット, $TC_DP14[t,d] に類似	
$TC_ECP36[t,d]	DOUBLE	R/W		$TC_DP6 のオフセット, $TC_DP15[t,d] に類似	
$TC_ECP37[t,d]	DOUBLE	R/W		$TC_DP7 のオフセット, $TC_DP16[t,d] に類似	
$TC_ECP38[t,d]	DOUBLE	R/W		$TC_DP8 のオフセット, $TC_DP17[t,d] に類似	
$TC_ECP39[t,d]	DOUBLE	R/W		$TC_DP9 のオフセット, $TC_DP18[t,d] に類似	
$TC_ECP40[t,d]	DOUBLE	R/W		$TC_DP10 のオフセット ,$TC_DP19[t,d] に類似	
$TC_ECP41[t,d]	DOUBLE	R/W		$TC_DP11 のオフセット, $TC_DP20[t,d] に類似	
$TC_ECP43[t,d]	DOUBLE	R/W		$TC_DP3 のオフセット, $TC_DP12[t,d] に類似	
$TC_ECP44[t,d]	DOUBLE	R/W		$TC_DP4 のオフセット: $TC_DP13[t,d] に類似	
$TC_ECP45[t,d]	DOUBLE	R/W		$TC_DP5 のオフセット, $TC_DP14[t,d] に類似	
$TC_ECP46[t,d]	DOUBLE	R/W		$TC_DP6 のオフセット, $TC_DP15[t,d] に類似	
$TC_ECP47[t,d]	DOUBLE	R/W		$TC_DP7 のオフセット, $TC_DP16[t,d] に類似	
$TC_ECP48[t,d]	DOUBLE	R/W		$TC_DP8 のオフセット, $TC_DP17[t,d] に類似	
$TC_ECP49[t,d]	DOUBLE	R/W		$TC_DP9 のオフセット, $TC_DP18[t,d] に類似	
$TC_ECP50[t,d]	DOUBLE	R/W		$TC_DP10 のオフセット, $TC_DP19[t,d] に類似	
$TC_ECP51[t,d]	DOUBLE	R/W		$TC_DP11 のオフセット, $TC_DP20[t,d] に類似	
$TC_ECP53[t,d]	DOUBLE	R/W		$TC_DP3 のオフセット, $TC_DP12[t,d] に類似	
$TC_ECP54[t,d]	DOUBLE	R/W		$TC_DP4 のオフセット, $TC_DP13[t,d] に類似	
$TC_ECP55[t,d]	DOUBLE	R/W		$TC_DP5 のオフセット, $TC_DP14[t,d] に類似	
$TC_ECP56[t,d]	DOUBLE	R/W		$TC_DP6 のオフセット, $TC_DP15[t,d] に類似	
$TC_ECP57[t,d]	DOUBLE	R/W		$TC_DP7 のオフセット, $TC_DP16[t,d] に類似	

名称	タイプ	アクセス		意味 | 値の範囲	インデックス | 値の範囲
		パートプログラム	シンクロナイズドアクション		
$TC_ECP58[t,d]	DOUBLE	R/W		$TC_DP8 のオフセット , $TC_DP17[t,d] に類似	
$TC_ECP59[t,d]	DOUBLE	R/W		$TC_DP9 のオフセット , $TC_DP18[t,d] に類似	
$TC_ECP60[t,d]	DOUBLE	R/W		$TC_DP10 のオフセット, $TC_DP19[t,d] に類似	
$TC_ECP61[t,d]	DOUBLE	R/W		$TC_DP11 のオフセット, $TC_DP20[t,d] に類似	
$TC_ECP63[t,d]	DOUBLE	R/W		$TC_DP3 のオフセット , $TC_DP12[t,d] に類似	
$TC_ECP64[t,d]	DOUBLE	R/W		$TC_DP4 のオフセット , $TC_DP13[t,d] に類似	
$TC_ECP65[t,d]	DOUBLE	R/W		$TC_DP のオフセット , $TC_DP14[t,d] に類似	
$TC_ECP66[t,d]	DOUBLE	R/W		$TC_DP6 のオフセット , $TC_DP15[t,d] に類似	
$TC_ECP67[t,d]	DOUBLE	R/W		$TC_DP7 のオフセット , $TC_DP16[t,d] に類似	
$TC_ECP68[t,d]	DOUBLE	R/W		$TC_DP8 のオフセット , $TC_DP17[t,d] に類似	
$TC_ECP69[t,d]	DOUBLE	R/W		$TC_DP9 のオフセット , $TC_DP18[t,d] に類似	
$TC_ECP70[t,d]	DOUBLE	R/W		$TC_DP10 のオフセット, $TC_DP19[t,d] に類似	
$TC_ECP71[t,d]	DOUBLE	R/W		$TC_DP11 のオフセット, $TC_DP20[t,d] に類似	
$TC_SCP13	DOUBLE	R/W		$TC_DP3 のオフセット , に類似した $TC_DP12[t,d] 有効な 'Flat D number management'（フラット D 番号管理）の機能を用いて , 構文は次のとおり : $TC_SCP13[d]t: T 番号 | 1 -32000 または MD を介して定義される d: ツールエッジ番号 / D 番号 1 - n (MD を介して定義される n)	t: T 番号 1 - MD を介して定義される (SLMAXTOOLNUMBER)
$TC_SCP14[t,d]	DOUBLE	R/W		$TC_DP4 のオフセット : $TC_DP13[t,d] に類似	
$TC_SCP15[t,d]	DOUBLE	R/W		$TC_DP5 のオフセット , $TC_DP14[t,d] に類似	
$TC_SCP16[t,d]	DOUBLE	R/W		$TC_DP6 のオフセット , $TC_DP15[t,d] に類似	

名称	タイプ	アクセス		意味 | 値の範囲	インデックス | 値の範囲
		パートプログラム	シンクロナイズドアクション		
$TC_SCP17[t,d]	DOUBLE	R/W		$TC_DP7 のオフセット, $TC_DP16[t,d] に類似	
$TC_SCP18[t,d]	DOUBLE	R/W		$TC_DP8 のオフセット, $TC_DP17[t,d] に類似	
$TC_SCP19[t,d]	DOUBLE	R/W		$TC_DP9 のオフセット, $TC_DP18[t,d] に類似	
$TC_SCP20[t,d]	DOUBLE	R/W		$TC_DP10 のオフセット ,$TC_DP19[t,d] に類似	
$TC_SCP21[t,d]	DOUBLE	R/W		$TC_DP11 のオフセット ,$TC_DP20[t,d] に類似	
$TC_SCP23[t,d]	DOUBLE	R/W		$TC_DP3 のオフセット, $TC_DP12[t,d] に類似	
$TC_SCP24[t,d]	DOUBLE	R/W		$TC_DP4 のオフセット, $TC_DP13[t,d] に類似	
$TC_SCP25[t,d]	DOUBLE	R/W		$TC_DP5 のオフセット, $TC_DP14[t,d] に類似	
$TC_SCP26[t,d]	DOUBLE	R/W		$TC_DP6 のオフセット, $TC_DP15[t,d] に類似	
$TC_SCP27[t,d]	DOUBLE	R/W		$TC_DP7 のオフセット, $TC_DP16[t,d] に 類似	
$TC_SCP28[t,d]	DOUBLE	R/W		$TC_DP8 のオフセット, $TC_DP17[t,d] に 類似	
$TC_SCP29[t,d]	DOUBLE	R/W		$TC_DP9 のオフセット, $TC_DP18[t,d] に類似	
$TC_SCP30[t,d]	DOUBLE	R/W		$TC_DP10 のオフセット, $TC_DP19[t,d] に類似	
$TC_SCP31[t,d]	DOUBLE	R/W		$TC_DP11 のオフセット, $TC_DP20[t,d] に類似	
$TC_SCP33[t,d]	DOUBLE	R/W		$TC_DP3 のオフセット, $TC_DP12[t,d] に類似	
$TC_SCP34[t,d]	DOUBLE	R/W		$TC_DP4 のオフセット, $TC_DP13[t,d] に類似	
$TC_SCP35[t,d]	DOUBLE	R/W		$TC_DP5 のオフセット, $TC_DP14[t,d] に類似	
$TC_SCP36[t,d]	DOUBLE	R/W		$TC_DP6 のオフセット, $TC_DP15[t,d] に類似	
$TC_SCP37[t,d]	DOUBLE	R/W		$TC_DP7 のオフセット, $TC_DP16[t,d] に類似	
$TC_SCP38[t,d]	DOUBLE	R/W		$TC_DP8 のオフセット, $TC_DP17[t,d] に類似	
$TC_SCP39[t,d]	DOUBLE	R/W		$TC_DP9 のオフセット, $TC_DP18[t,d] に類似	
$TC_SCP40[t,d]	DOUBLE	R/W		$TC_DP10 のオフセット, $TC_DP19[t,d] に類似	

名称	タイプ	アクセス		意味 | 値の範囲	インデックス | 値の範囲
		パートプログラム	シンクロナイズドアクション		
$TC_SCP41[t,d]	DOUBLE	R/W		$TC_DP11 のオフセット, $TC_DP20[t,d] に類似	
$TC_SCP43[t,d]	DOUBLE	R/W		$TC_DP3 のオフセット, $TC_DP12[t,d] に類似	
$TC_SCP44[t,d]	DOUBLE	R/W		$TC_DP4 のオフセット, $TC_DP13[t,d] に類似	
$TC_SCP45[t,d]	DOUBLE	R/W		$TC_DP5 のオフセット, $TC_DP14[t,d] に類似	
$TC_SCP46[t,d]	DOUBLE	R/W		$TC_DP6 のオフセット, $TC_DP15[t,d] に類似	
$TC_SCP47[t,d]	DOUBLE	R/W		$TC_DP7 のオフセット, $TC_DP16[t,d] に類似	
$TC_SCP48[t,d]	DOUBLE	R/W		$TC_DP8 のオフセット, $TC_DP17[t,d] に類似	
$TC_SCP49[t,d]	DOUBLE	R/W		$TC_DP9 のオフセット, $TC_DP18[t,d] に類似	
$TC_SCP50[t,d]	DOUBLE	R/W		$TC_DP10 のオフセット, $TC_DP19[t,d] に類似	
$TC_SCP51[t,d]	DOUBLE	R/W		$TC_DP11 のオフセット, $TC_DP20[t,d] に類似	
$TC_SCP53[t,d]	DOUBLE	R/W		$TC_DP3 のオフセット, $TC_DP12[t,d] に類似	
$TC_SCP54[t,d]	DOUBLE	R/W		$TC_DP4 のオフセット, $TC_DP13[t,d] に類似	
$TC_SCP55[t,d]	DOUBLE	R/W		$TC_DP5 のオフセット, $TC_DP14[t,d] に類似	
$TC_SCP56[t,d]	DOUBLE	R/W		$TC_DP6 のオフセット, $TC_DP15[t,d] に類似	
$TC_SCP57[t,d]	DOUBLE	R/W		$TC_DP7 のオフセット, $TC_DP16[t,d] に類似	
$TC_ECP58[t,d]	DOUBLE	R/W		$TC_DP8 のオフセット, $TC_DP17[t,d] に類似	
$TC_SCP59[t,d]	DOUBLE	R/W		$TC_DP9 のオフセット, $TC_DP18[t,d] に類似	
$TC_SCP60[t,d]	DOUBLE	R/W		$TC_DP10 のオフセット, $TC_DP19[t,d] に類似	
$TC_SCP61[t,d]	DOUBLE	R/W		$TC_DP11 のオフセット, $TC_DP20[t,d] に類似	
$TC_SCP63[t,d]	DOUBLE	R/W		$TC_DP3 のオフセット, $TC_DP12[t,d] に類似	
$TC_SCP64[t,d]	DOUBLE	R/W		$TC_DP4 のオフセット, $TC_DP13[t,d] に類似	
$TC_SCP65[t,d]	DOUBLE	R/W		$TC_DP のオフセット, $TC_DP14[t,d] に類似	

名称	タイプ	アクセス		意味 | 値の範囲	インデックス | 値の範囲
		パートプログラム	シンクロナイズドアクション		
$TC_SCP66[t,d]	DOUBLE	R/W		$TC_DP6 のオフセット , $TC_DP15[t,d] に類似	
$TC_SCP67[t,d]	DOUBLE	R/W		$TC_DP7 のオフセット , $TC_DP16[t,d] に類似	
$TC_SCP68[t,d]	DOUBLE	R/W		$TC_DP8 のオフセット , $TC_DP17[t,d] に類似	
$TC_SCP69[t,d]	DOUBLE	R/W		$TC_DP9 のオフセット , $TC_DP18[t,d] に類似	
$TC_SCP70[t,d]	DOUBLE	R/W		$TC_DP10 のオフセット ,$TC_DP19[t,d] に類似	
$TC_SCP71[t,d]	DOUBLE	R/W			
$TC_ADPT1[a]	DOUBLE	R/W		アダプタジオメトリ：長さ 1a: アダプタ番号 1 - n (MD を介して定義された n) (SLMAXADAPTERNUMBER)	
$TC_ADPT2[a]	DOUBLE	R/W		アダプタジオメトリ：長さ 2a	
$TC_ADPT3[a]	DOUBLE	R/W		アダプタジオメトリ：長さ 3a	
$TC_ADPTT[a]	INT	R/W		アダプタ変換番号 a:	アダプタ番号 1 - n (MD を介して定義された n) (SLMAXADAPTERNUMBER)

補正

	名称	タイプ	コメント
測定システム補正値	$AA_ENC_COMP_x[n,m,a]		n: エンコーダ番号 0 - 1 m: 点番号 a: マシン軸
	$AA_ENC_COMP[n,m,a]	REAL	補正値
	$AA_ENC_COMP_STEP[n,a]		ステップ幅
	$AA_ENC_COMP_MIN[n,a]		補正開始位置
	$AA_ENC_COMP_MAX[n,a]		補正終了位置
	$AA_ENC_COMP_IS_MODULO[n,a]	BOOL	補正はモジューロである
補間補正	$AN_CEC_xxx_yyy[n,m]		n: 補正テーブル番号, 0 - 最大値 (MD を介して設定可能) m: 中間点の数, 0 - 最大値 (MD を介して設定可能)
	$AN_CEC[n,m]	REAL	補正値
	$AN_CEC_INPUT_AXIS[n]	AXIS	セットポイントが補正テーブル出力として機能する軸の名称
	$AN_CEC_OUTPUT_AXIS[n]		補正テーブル入力の影響を受ける軸の名称
	$AN_CEC_STEP[n]	REAL	補正値間の距離
	$AN_CEC_MIN[n]		補正テーブルの開始位置
	$AN_CEC_MAX[n]		補正テーブルの終了位置
	$AN_CEC_DIRECTION[n]	INT	補正テーブルの方向指示アクションを起動する
	$AN_CEC_MULT_BY_TABLE[n]		出力値が補正テーブルの出力値で掛算されるテーブルの番号 0: 基本軸が両方向に移動する 1: 基本軸が正方向に移動する -1: 基本軸が負方向に移動する
	$AN_CEC_IS_MODULO[n]	BOOL	1 (TRUE): 補正テーブルの周期的反復 0 (FALSE): 補正テーブルの周期的反復なし

保護ゾーン

	名称	タイプ	コメント
NCK 指定保護ゾーン	$SN_PA_xxx_yyy[n,m]		n: 保護ゾーンの番号, 0 - 最大値 (MD を介して設定可能) m: 輪郭要素番号 0 から 10 まで
	$SN_PA_ACTIV_IMMED[n]	BOOL	保護ゾーンは即座に有効になるか ?
	$SN_PA_T_W[n]	CHAR	ワーク関連／ツール関連保護ゾーン 0: ワーク関連保護ゾーン 3: ツール関連保護ゾーン
	$SN_PA_ORI[n]	INT	保護ゾーンの向き 0: 1 番目と 2 番目のジオメトリ軸からの面における多角形の定義 1: 3 番目と 1 番目のジオメトリ軸からの面における多角形の定義 2: 2 番目と 3 番目のジオメトリ軸からの面における多角形の定義
	$SN_PA_LIM_3DIM[n]		多角形の定義に垂直な軸における保護ゾーンの制限についての識別 0: 制限なし 1: 正方向の制限 2: 負方向の制限 3: 両方向の制限
	$SN_PA_PLUS_LIM[n]	REAL	多角形定義に垂直に位置決めされた軸の保護ゾーンの正の制限
	$SN_PA_MINUS_LIM[n]		多角形定義に垂直に位置決めされた軸の負方向の保護ゾーンの負の制限
	$SN_PA_CONT_NUM[n]	INT	有効な輪郭要素の数
	$SN_PA_CONT_TYPE[n,m]		輪郭要素タイプ (G1, G2, G3)
	$SN_PA_CONT_ORD[n,m]	REAL	輪郭要素の終点 （縦座標）
	$SN_PA_CONT_ABS[n,m]		輪郭要素の終点 (横座標)
	$SN_PA_CENT_ORD[n,m]		輪郭要素の中心点（縦座標）
	$SN_PA_CENT_ABS[n,m]		輪郭要素の中心点 (横座標）

14.3 システム変数のリスト

14.3.1 R パラメータ

名称	タイプ	アクセス		意味 | 値の範囲	インデックス | 値の範囲
		パートプログラム	シンクロナイズドアクション		
Rn または R[n]	REAL	R/W		静的メモリ内の算術変数	n: 算術変数の番号 | 0 - 最大 (MD を介して定義される)
$Rn または $R[n]			R/W		

14.3.2 フレーム

$P_PFRAME	FRAME	R/W		現在のプログラム可能なフレーム	
$P_ACTFRAME				現在の全体フレーム	
$P_IFRAME				現在の設定可能フレーム	
$P_UIFRNUM	INT	R		有効な $P_UIFR の番号	
$P_UIFR[n]	FRAME	R/W		設定可能なフレーム (G54 etc.)	n: 番号 | 1 - 100 (MD を介して設定可能)
$P_ACTBFRAME	FRAME	R/		$P_ACTBFRAME 現在のチェーニングされた完全なベースフレーム	
$P_CHBFRAME	FRAME	R/W		$P_CHBFRAME[n] チャンネルにおける現在のベースフレーム MD を介して，0 から 8 まで構成可能 $MC_MM_NUM_BASE_FRAMES. 寸法のチェックが変数アクセスにおいて実行される。	MD $MC_MM_NUM_BASE_FRAMES を介して，0 から 8 まで構成可能。寸法のチェックが変数アクセスにおいて実行される。
$P_CHBFRMASK	INT	R/W		$P_CHBFRMASK 完全なベースフレームの計算に含まれるべきチャンネル別ベースフレームを定義するためのビットマスク	
$P_NCBFRAME	FRAME	R/W		$P_NCBFRAME[n] 現在の NCU ベースフレーム 0 から 8 の NCU ベースフレームは，MD $MN_MM_NUM_GLOBAL_BASE_FRAMES を介して構成できる。	0 から 8 の NCU ベースフレームは，MD $MN_MM_NUM_GLOBAL_BASE_FRAMES を介して構成できる。

$P_CHBFR	FRAME	R/W		$P_CHBFR[n] チャンネルベースフレームは，G500, G54 .. G599 を介して起動できる。0 から 8 までのチャンネルベースフレームは，MD $MC_MM_NUM_BASE_FRAMES を介して構成できる。0 から 8 までのチャンネルベースフレームは，MD $MC_MM_NUM_BASE_FRAMES を介して構成できる。	

14.3.3 ツールホルダデータ

名称	タイプ	アクセス		意味 | 値の範囲	インデックス | 値の範囲
		パートプログラム	シンクロナイズドアクション		
$TC_CARRx[n]	REAL	R/W		初期設定は = 0; すなわち，NCK は，そのようなデータを持たない。	n: 最大パラメータ数 (MD を介して設定可能)
$TC_CARR1[n]	REAL	R/W		オフセットベクトル I1 の x 要素	
$TC_CARR2[n]	REAL	R/W		オフセットベクトル I1 の z 要素	
$TC_CARR3[n]	REAL	R/W		オフセットベクトル I1 の z 要素	
$TC_CARR4[n]	REAL	R/W		オフセットベクトル I2 の x 要素	
$TC_CARR5[n]	REAL	R/W		オフセットベクトル I2 の z 要素	
$TC_CARR6[n]	REAL	R/W		オフセットベクトル I2 の z 要素	
$TC_CARR7[n]	REAL	R/W		回転軸 v1 の x 要素	
$TC_CARR8[n]	REAL	R/W		回転軸 v1 の z 要素	
$TC_CARR9[n]	REAL	R/W		回転軸 v1 の z 要素	
$TC_CARR10[n]	REAL	R/W		回転軸 v2 の x 要素	
$TC_CARR11[n]	REAL	R/W		回転軸 v2 の z 要素	
$TC_CARR12[n]	REAL	R/W		回転軸 v2 の z 要素	
$TC_CARR13[n]	REAL	R/W		degrees (度) で表した回転角度 a1	
$TC_CARR14[n]	REAL	R/W		degrees(度) で表した回転角度 a2	
$TC_CARR15[n]	REAL	R/W		オフセットベクトル l_3 の x 要素	
$TC_CARR16[n]	REAL	R/W		オフセットベクトル l_3 の y 要素	
$TC_CARR17[n]	REAL	R/W		オフセットベクトル l_3 の z 要素	n: 角度 1 - 2
$P_TCANG	DOUBLE	R/		$P_TCANG[n] ツールホルダ軸の有効な角度 : 角度 1 - 2	

$TC_CARR15[n]	DOUBLE	R/W		ツールホルダの最大数がマシンデータを介してセットできる基本ベクトルの X 要素 初期設定 = 0; すなわち，NCK は，そのようなデータを持たない。	ツールホルダの最大数は，マシンデータを介して，セットできる。 初期設定 = 0; すなわち，NCK は，そのようなデータを持たない。
$TC_CARR16[n]	DOUBLE	R/W		ツールホルダの最大数がマシンデータを介してセットできる基本ベクトルの Y 要素 初期設定 = 0; すなわち，NCK は，そのようなデータを持たない。	ツールホルダの最大数は，マシンデータを介して，セットできる。 初期設定 = 0; すなわち，NCK は，そのようなデータを持たない。
$TC_CARR17[n]	DOUBLE	R/W		ツールホルダの最大数がマシンデータを介してセットできる基本ベクトルの Z 要素 初期設定 = 0; すなわち，NCK は，そのようなデータを持たない。	ツールホルダの最大数は，マシンデータを介して，セットできる。 初期設定 = 0; すなわち，NCK は，そのようなデータを持たない。

14.3.4 チャンネル別保護ゾーン

名称	タイプ	アクセス		意味 | 値の範囲	インデックス | 値の範囲
		パートプログラム	シンクロナイズドアクション		
$SC_PA_xxx_yyy[n,m]	REAL	R/W			n: 保護ゾーン番号 | 0 - 最大値 (MD を介して設定可能) m: 輪郭要素の番号 | 0 - 10
$SC_PA_ACTIV_IMM ED[n]	BOOL			保護ゾーンは即時に有効であるか ? | 0, 1	
$SC_PA_T_W[n]	CHAR			ワーク／ツール関連保護ゾーン | 0: ワーク関連保護ゾーン 3: ツール関連保護ゾーン	
$SC_PA_PA_ORI[n]	INT			保護ゾーンの向き : 平面 ... での多角形定義 | 0: 1 番目と 2 番目のジオメトリ軸から 1: 3 番目と 1 番目のジオメトリ軸から 2: 2 番目と 3 番目のジオメトリ軸から	

名称	タイプ	アクセス		意味 | 値の範囲	インデックス | 値の範囲
		パートプログラム	シンクロナイズドアクション		
$SN_PA_LIM_3DIM[n]				多角形の定義に垂直な軸での保護ゾーンの制限についての識別 0: 制限なし 1: 正方向での制限 2: 負方向での制限 3: 両方向での制限	
$SC_PA_PLUS_LIM[n]	REAL			多角形定義に垂直に位置決めされた軸の負方向で保護ゾーンの正の制限	
$SC_PA_MINUS_LIM[n]				多角形定義に垂直に位置決めされた軸の負方向の保護ゾーンの負の制限	
$SC_PA_CONT_NUM[n]	INT			有効な輪郭要素の数	
$SC_PA_CONT_TYP [n,m]				輪郭要素タイプ (G1, G2, G3)	
$SC_PA_CONT_ORD [n,m]	REAL	R/W		輪郭要素の終点 (縦座標)	
$SC_PA_CONT_ABS [n,m]	REAL	R/W		$SN_PA_CONT_ABS[n,m]	
$SN_PA_CENT_ORD[n,m]				輪郭要素の中心点（縦座標）	n: パラメータ番号 | 0-49
$SN_PA_CENT_ABS[n,m]				輪郭要素の中心点 (横座標）	
$AC_FIFOx[n]	REAL	R/W	R/W	モーションシンクロナイズドアクションおよび周期的測定のための FIFO x: 1-10	n: パラメータ番号 | 0 - 最大 FIFO 要素 n=0: 書込み : FIFO に新規値を保存する 読取り : 最も古い要素を読取り，FIFO から消去する n=1: 最も古い要素へのアクセスを読取る n=2: 最新の要素へのアクセスを読取る n=3: ビット 0 が MD $MC_MM_MODE_FIFO にセットされている場合，FIFO に保存されたすべての要素のトータル n=4: FIFO に保存された要素の現在の数へのアクセスを読取る n=5: 最も古い要素 n=6: 2 番目に古い要素，など。

14.3.5 入力／出力

名称	タイプ	アクセス		意味 | 値の範囲	インデックス | 値の範囲
		パートプログラム	同期アクション		
$A_xxx[n]					n: 入力／出力 番号 | 1 - マシンデータで定義される最高値
$A_IN[n]	BOOL	R	R	ディジタル入力 NC	
$A_OUT[n]		R/W	R/W	ディジタル出力 NC	
$A_INCO[n]	BOOL	R	R	コンパレータ入力 NC	

14.3.6 PLC 変数の読取りと書込み

名称	タイプ	アクセス		意味 | 値の範囲	インデックス | 値の範囲
		パートプログラム	シンクロナイズドアクション		
$A_DBx[n]		R/W			n: NC と PLC 間のデュアルポートラム内の位置オフセット
$A_DBB[n]	INT		R/W	PLC からの読取りと PLC への書込みのデータバイト (8 ビット)	
$A_DBW[n]			R/W	PLC からの読取りと PLC への書込みのデータワード (16 ビット)	
$A_DBD[n]			R/W	PLC からの読取りと PLC への書込みのダブルデータワード (32 ビット)	
$A_DBR[n]	REAL	R/W	R/W	PLC からの読取りと PLC への書込みの実数データ (32 ビット)	
$A_PBB_IN INT	R/			1 (TRUE) | 0 (FALSE) $A_PBB_IN[n] 制御システム電源オンおよび軸の基準化後に PLC IO から直接データバイト (8 ビット) を読取る n: PLC 入力範囲内 0 - ... の位置オフセット	n: PLC 入力範囲 0 - ... 内の位置オフセット
$A_PBB_OUT	INT	R/W	R/W	$A_PBB_OUT[n] PLC IO にデータバイト (8 ビット) を直接書き込む : PLC 出力範囲内 0 - ... の位置オフセット	n: PLC 出力範囲内 0 - ... 位置オフセット
$A_PBD_IN	INT	R	R	$A_PBD_IN[n] PLC IO からダブルデータワード (32 ビット) を直接読取る n: PLC 入力範囲内 0 - ... の位置オフセット	n: PLC 入力範囲内 0 - ... の位置オフセット
$A_PBD_OUT	INT	R/W	R/W	$A_PBD_OUT[n] PLC IO にダブルデータワードを直接書き込む : PLC 出力範囲内 0 - ... の位置オフセット	n: PLC 出力範囲内 0 - ... の位置オフセット

$A_PBR_OUT	DOUBLE	R/W	R/W	$A_PBR_OUT[n] PLC IO に実数データ (32 ビット) を直接書き込む n: PLC 出力範囲内 0 - ... での位置オフセット	n: PLC 出力範囲内 0 - ... の位置オフセット
$A_PBW_IN	INT	R	R	$A_PBW_IN[n] PLC IO からデータワード (16 ビット) を直接読取る n: PLC 入力範囲内 0 - ... の位置オフセット	n: PLC 入力範囲内 0 - ... の位置オフセット
$A_PBW_OUT	INT	R/W	R/W	$A_PBW_OUT[n] PLC IO からデータワード (16 ビット) を直接書込む n: PLC 出力範囲内 0 - ... の位置オフセット	n: PLC 出力範囲内 0 - ... の位置オフセット

14.3.7 タイマ

名称	タイプ	アクセス		意味 | 値の範囲	インデックス | 値の範囲
		パートプログラム	シンクロナイズドアクション		
$A_YEAR	INT	R	R	システム時間 - 年 | 0 - 99	
$A_MONTH				システム時間 - 月 | 1 - 12	
$A_DAY				システム時間 - 日 | 1- 31	
$A_HOUR				システム時間 - 時間 | 0 - 23	
$A_MINUTE				システム時間 - 分 | 0 - 59	
$A_SECOND				システム時間 - 秒 | 0 - 59	
$A_MSECOND				システム時間 - ミリ秒 | 0 - 999	
$AC_TIMER[n]	REAL	R/W	R/W	秒で示すタイマロケーション | 時間は，補間サイクルの倍数でカウントされる ;	n: タイマロケーションの番号 | 1 - ... (MD を介して定義される)
$AC_TIME	REAL	R	R	秒でのブロックの始めからの時間	
$AC_TIMEC				IPO クロックサイクルでのブロックの始めからの時間	

14.3.8 チャンネルステータス

名称	タイプ	アクセス		意味 | 値の範囲	インデックス | 値の範囲
		パートプログラム	同期アクション		
$AC_MEA[n]	INT	R	R	1 (TRUE): プローブは切替えられたステータスになる	n: プローブ番号 | 1 - 2
$AC_TRAFO		R		有効な変換のコード番号 ; 0 (FALSE): 変換 OFF	

名称	タイプ	アクセス		意味 | 値の範囲	インデックス | 値の範囲
		パートプログラム	シンクロナイズドアクション		
$AC_STAT			R	現在のチャンネルステータス -1: 無効 0: プログラムはリセット状態 1: プログラム停止 2. プログラムが有効 3: プログラム待機中 4: プログラム中断	
$AC_PROG			R	現在のプログラムステータス : -1: 無効 0: プログラムはリセット状態 1: プログラム停止 2. プログラムが有効 3: プログラム待機中 4: プログラム中断	
$AC_IPO_BUF		R	R	IPO バッファのステータスを満たす (あらかじめコード化されたブロック数を指定する) | 0 - $MC_MM_IPO_BUFFER_SIZE	
$AC_SYNA_MEM		R	R	モーションシンクロナイズドアクションのための利用可能なメモリ要素の数 | 0 - $MC_MM_NUM_SYNC_ELEMENTS	
$AC_LIFTFAST	INT	R/W		有効な変換のコード番号 0: リターンストロークが有効でなかった 1: リターンストロークが有効だった	
$P_ISTEST	BOOL	R		パートプログラムのテストモードをチェックする 0 (FALSE): プログラムテストが無効 1 (TRUE): プログラムテストが有効	
$AC_ALARM_STAT	INT	R	R	$AC_ALARM_STAT (選択された) シンクロナイズドアクションに対するアラーム反応 (SYNFCT)	

14.3.9 パスモーションの変数

パス

名称	タイプ	アクセス		意味 | 値の範囲	インデックス | 値の範囲
		パートプログラム	シンクロナイズドアクション		
$AC_PATHN	REAL		R	標準化されたパスパラメータ／次の間の値 0 = ブロック始めおよび 1 = ブロック終り	
$AC_DTBW				WCS のブロック始めからのジオメトリック距離	
$AC_DTBB				BCS のブロック始めからのジオメトリック距離	
$AC_DTEW				WCS のブロック終りからのジオメトリック距離	
$AC_DTEB				BCS のブロック終りからのジオメトリック距離	
$AC_PLTBB				BCS のブロック始めからのパス距離	
$AC_PLTEB				BCS のブロック終りからのパス距離	
$AC_DTEPW				WCS のインフィード揺動のための移動距離	
$AC_DTEPB				BCS のインフィード揺動のための移動距離	
$AC_DELT	REAL	R	R	モーションシンクロナイズドアクションのための DELDTG を使って，パス上の残移動量を削除した後，WCS で，パス上の残移動量のラッチを解除	

速度

名称	タイプ	アクセス		意味 | 値の範囲	インデックス | 値の範囲
		パートプログラム	シンクロナイズドアクション		
$AC_OVR	REAL		R/W	シンクロナイズドアクションのためのパスオーバライド	
$AC_VC				シンクロナイズドアクションのための追加パスフィード補正	
$AC_VACTW	REAL		R	ワーク座標系におけるパス速度	
$AC_VACTB				基本座標系におけるパス速度	
$P_F	REAL	R		最近プログラムされたパスフィード F	

14.3.10 シンクロナイズドアクションのための多項式の値

名称	タイプ	アクセス		意味 | 値の範囲		インデックス | 値の範囲
$AC_FCTxyL	REAL	R/W	R/W	制限値 (LmV) x. シンクロナイズドアクション SYNFCT / 評価機能 FCTDEF x についての多項式		
$AC_FCT1LL				低制限値	x =1	
$AC_FCT2LL				低制限値	x = 2	
$AC_FCT3LL				低制限値	x = 3	
$AC_FCT1UL				高制限値	x = 1	
$AC_FCT2UL				高制限値	x = 2	
$AC_FCT3UL				高制限値	x = 3	
$AC_FCTxC[n]	REAL	R/W	R/W	シンクロナイズドアクション／評価機能 FCTDEF x において多項式 x (SYNFCT) についての多項式係数 a0 - a3		n: 多項式係数の番号 a0 -a3 | 0 - 3
$AC_FCT1C[n]				x = 1		
$AC_FCT2C[n]				x = 2		
$AC_FCT3C[n]				x = 3		
$AC_FCTLL[n]	REAL	R/W	R/W	シンクロナイズドアクション SYNFCT ／ 評価機能 FCTDEF n のための多項式による低制限値		n: 多項式番号 |1 - MD 値 $MC_MM_NUM_FC_TDEF_ELEMENTS
$AC_FCTUL[n]	REAL	R/W	R/W	シンクロナイズドアクション SYNFCT ／ 評価機能 FCTDEF n のための多項式による高制限値 n		
$AC_FCTx[n]	REAL	R/W	R/W	シンクロナイズドアクション SYNFCT ／ 評価機能 FCTDEF n のための多項式による係数 n		n: 多項式番号 | 1 - MD 値 $MC_MM_NUM_FC_TDEF_ELEMENTS
$AC_FCT0[n]				x = 0		
$AC_FCT1[n]				x = 1		
$AC_FCT2[n]				x = 2		
$AC_FCT3[n]				x = 3		

14.3.11 軸別変数

位置およびパス

名称	タイプ	アクセス		意味 | 値の範囲	インデックス | 値の範囲
		パートプログラム	シンクロナイズドアクション		
$AA_IW[axis]	REAL	R	R	WCS での実際値	軸：マシン軸
$AA_IB[axis]				BCS での実際値	
$AA_IM[axis]				MCS での実際値	
$AA_ENC_ACTIVE[axis]	BOOL			1 (TRUE): 有効な測定系は，エンコーダ制限周波数以下で動作する	軸
$AA_ENC1_ACTIVE[axis]				1 (TRUE): エンコーダ 1 は，エンコーダ制限周波数以下で動作する	
$AA_ENC2_ACTIVE[axis]				1 (TRUE): エンコーダ 2 は，エンコーダ制限周波数以下で動作する	
$VA_IMx[axis]	REAL	R	R	MCS で（測定された）現在の実際値；補正 (温度 , CEC) が適用されている。エンコーダ周波数を超える場合に，変数が未定義の値を提供する。軸 : マシン軸	
$VA_IM[axis]				有効な測定系	
$VA_IM1[axis]				エンコーダ 1	
$VA_IM2[axis]				エンコーダ 2	
$AA_MW[axis]	REAL	R	R	ワーク座標系での測定値	軸：マシン軸
$AA_MM[axis]				マシン座標系での測定値	
$AC_DRF[axis] $AC_PRESET[axis]				DRF オフセット 最近指定されたプリセット値	
$AA_ETRANS[axis]	REAL	R/W	R/W	外部ゼロオフセット	
$AA_OFF[axis]			R/W	プログラムされた軸のオーバレイされたモーション	
$AC_RETPOINT[axis]		R		再アプローチ用輪郭でのリセット点	
$AA_SOFTENDP[axis]	REAL	R	R	ソフトウエア制限位置，正方向	軸：マシン軸
$AA_SOFTENDN[axis]				ソフトウエア制限位置，負方向	
$AA_DTxx[axis]	REAL	R	R	位置決め用軸方向パスおよびモーションシンクロナイズドアクション用同期軸	軸：マシン軸 | 位置決め軸 同期軸

$AA_DTBW[axis]	REAL	R	R	WCS で，ブロック始めから	
$AA_DTBB[axis]				BCS で，ブロック始めから	
$AA_DTEW[axis]				WCS で，ブロック終りまで	
$AA_DTEB[axis]				BCS で，ブロック終りまで	
$AA_DTEPW[axis]				WCS で，インフィード発振用軸方向移動距離	
$AA_DTEPB[axis]				BCS で，インフィード発振用軸方向移動距離	
$AA_DELT[axis]				モーションシンクロナイズドアクションのための DELDTG (軸) を使って，残移動量を軸方向に削除した後，WCS での軸方向の残移動量のラッチを解除する	

名称	タイプ	アクセス		意味 | 値の範囲	インデックス | 値の範囲
		パートプログラム	シンクロナイズドアクション		
$AA_OSCILL_REVERSE_POS1[axis]		REAL	R	揺動用の現在のリバース位置 1 ; $SA_OSCILL_REVERSE_POS1 の設定データ値は，シンクロナイズドアクションにおいてオンラインで評価される。軸：マシン軸	
$AA_OSCILL_REVERSE_POS2[axis]				揺動用の現在のリバース位置 2 ; $SA_OSCILL_REVERSE_POS2 の設定データ値は，シンクロナイズドアクションにおいてオンラインで評価される	
$AA_MWx[axis]	REAL	R/W	R/W	軸方向測定の測定結果	
$AA_MW1[axis]				WCS でのトリガイベント 1	
$AA_MW2[axis]				WCS でのトリガイベント 2	
$AA_MW3[axis]				WCS でのトリガイベント 3	
$AA_MW4[axis]				WCS でのトリガイベント 4	
$AA_MMx[axis]	REAL	R/W	R/W	軸方向測定の測定結果	
$AA_MM1[axis]				MCS でのトリガイベント 1	
$AA_MM2[axis]				MCS でのトリガイベント 2	
$AA_MM3[axis]				MCS でのトリガイベント 3	
$AA_MM4[axis]				MCS でのトリガイベント 4	
$AA_MEACT[axis]	BOOL	R		1 (TRUE): X について有効な軸方向測定	
$P_EP[axis]	REAL	R		最近プログラムされたセットポイント	
$P_FA[axis]	REAL	R		最近プログラムされた軸フィード	
$AA_ACT_INDEX_AX_POS_NO	INT	R/	R/	$AA_ACT_INDEX_AX_POS_NO[X] 0: インデックス軸がないので，インデックス位置が使用できない > 0: 最後に到達したまたは移動したインデックス位置の番号	
$AA_IBN	DOUBLE	R/	R/	$AA_IBN[X] 基本原点システムでの実際値 (BOS)	

名称	タイプ	アクセス		意味 | 値の範囲	インデックス | 値の範囲
		パートプログラム	シンクロナイズドアクション		
$AA_IEN DOUBLE	DOUBLE	R/	R/	$AA_IEN[X] 設定可能ゼロシステムでの実際値 (SZS).	
$AA_PROG_INDEX_A X_POS_NO	INT	R/	R/	$AA_PROG_INDEX_AX_POS_NO[X] 0: インデックス軸がないので，インデックス位置が使用できないか，またはインデックス軸が現在はインデックス位置にアプローチしていない。 > 0: プログラムされたインデックス位置の番号	
$AA_REF		INT	R/	$AA_REF[X] 軸ステータス : 0: 軸は参照されない 1: 軸が参照される	

速度，軸

名称	タイプ	アクセス		意味 | 値の範囲	インデックス | 値の範囲
		パートプログラム	シンクロナイズドアクション		
$AA_OVR[axis]	REAL		R/W	モーションシンクロナイズドアクションの軸オーバライド	主軸なしの軸
$AA_VC[axis]				パス送りまたは軸送りの追加補正値	
$xA_VACTy[axis]	REAL		R	軸速度　軸	
$AA_VACTW[axis]				WCS で	
$AA_VACTB[axis]				BCS で	
$AA_VACTM[axis]				MCS でのセットポイントとして	
$VA_VACTM[axis]				MCS での実際値として	

電子ギア

名称	タイプ	アクセス		意味 | 値の範囲	インデックス | 値の範囲
		パートプログラム	シンクロナイズドアクション		
$AA_EG_TYPE[a,b]	INT	R		カップリングのタイプ : 0: 実際値カップリング 1: セットポイントカップリング	軸識別子 a: 追従軸 b: リーディング軸
$AA_EG_NUMERA [a,b]	REAL	R		カップリング係数のカウンタ CF CF = 分子／分母 初期設定 : 0	軸識別子 a: 追従軸 b: リーディング軸
$AA_EG_DENOM [a,b]	REAL	R		カップリング係数の分母 CF CF = 分子／分母 初期設定 : 1 分母は正であること。	軸識別子 a: 追従軸 b: リーディング軸
$AA_EG_SYN[a,b]	REAL	R		指定されたリーディング軸についての同期位置 初期設定 : 0	軸識別子 a: 追従軸 b: リーディング軸
$AA_EG_SYNFA[a]	REAL	R		指定された追従軸についての同期位置 初期設定 : 0	軸識別子 a: 追従軸
$AA_EG_SYNFA	DOUBLE	R/	R/	$AA_EG_SYNFA[a] a: 追従軸 追従軸の同期位置	
$AA_EG_BC[a]	STRING	R		EG 起動コールのためのブロック変更基準 : EGON, EGONSYN: "NOC" 即時 "FINE" 同期（微） "COARSE" 同期 （粗） "IPOSTOP" セットポイント同期化	軸識別子 a: 追従軸

名称	タイプ	アクセス		意味 | 値の範囲	インデックス | 値の範囲
		パートプログラム	シンクロナイズドアクション		
$AA_EG_BC	STRING	R/	R/	$AA_EG_BC[a] ブロック変更基準 2. 自動的に MAXSTRINGLEN である TYPE_STRING をもつサイズ	
$AA_EG_NUM_LA[a]	INT	R		EGDEF を使って定義されたリーディング軸の番号。EGDEF を使って追従軸として定義される軸がない場合は，0。	軸識別子 a: 追従軸
$AA_EG_NUM_LA	INT	R/	R/	$AA_EG_NUM_LA[a] a: 追従軸 EGDEF を使って指定されたリーディング軸の番号	
$AA_EG_AX[a,n]	AXIS	R		インデックス n が指定されているリーディング軸の軸識別子。	軸識別子 a: 追従軸 n: EG グループのリーディング軸 0 ... 4
$AA_EG_ACTIVE[a,b]	BOOL	R		リーディング軸の起動ステータスを決定する : 0: 停止 1: 起動	軸識別子 a: 追従軸 b: リーディング軸
$VA_EG_SYNCDIFF [a]	REAL	R	R	同期差の実際値。MD $MA_COUPLE_POS_TOL_COARSE との比較および _FINE がインタフェース信号をリターンする。	軸識別子 a: 追従軸
$VA_EG_SYNCDIFF	DOUBLE	R/	R/	$VA_EG_SYNCDIFF[a] a: 追従軸 同期差	

ドライブデータ

名称	タイプ	アクセス		意味 | 値の範囲	インデックス | 値の範囲
		パートプログラム	同期アクション		
$AA_LOAD[axis]	REAL	R	R	ドライブ容量使用を % で示す	軸
$AA_TORQUE [axis]				ドライブトルクセットポイントを Nm で示す	
$AA_POWER[axis]				ドライブ有効電力を W で示す	
$AA_CURR[axis]				軸または主軸の実際現在値を A で示す	
$AA_SCTRACE[axis]	BOOL	R/W		書込み : 0 (FALSE): アクションなし 1 (TRUE): サーボトレースのために，IPO トリガを開始する 読取り : 常に 0 (なぜなら，セルフリセットトリガビットはインタフェースから読み戻されるから)	

軸のステータス

名称	タイプ	アクセス		意味 | 値の範囲	インデックス | 値の範囲
		パートプログラム	シンクロナイズドアクション		
$AA_STAT[axis]	INT	R	R	軸ステータス | 軸 0: 使用できる軸ステータスなし 1: 進行中の移動モーション 2: 軸はすでに IPO 終りに到達した (チャンネル内の軸にのみ適用) 3: すべての軸にとって適切な位置にある軸 （イグザクトストップ（粗）） 4: すべての軸にとって適切な位置に軸 （イグザクトストップ（微））	
$AA_TYP[axis]				軸タイプ 0: 別のチャンネルにある軸 1: 自身のチャンネル内のチャンネル 2: 中立軸 3: PLC 軸 4: 往復軸 5: 中立軸 , JOG にて 6: リーディング値が次の値とカップリングされる 7: 追従軸のカップリングされたモーション 8: コマンド軸 9: サイクル軸をコンパイルする	
$AA_FXS[axis]				ステータス "Travel to fixed stop"(固定停止まで移動) | 0: 固定停止点にはない軸 1: 順調に固定停止にアプローチする 2: 固定停止へのアプローチが失敗した	
$AA_OFF_LIMIT [axis]				軸オフセット用制限値 $AA_OFF... | 0: 到達しない 1: 正の方向に到達した -1: 負の軸方向に到達した	マシン軸

名称	タイプ	アクセス		意味 | 値の範囲	インデックス | 値の範囲
		パートプログラム	シンクロナイズドアクション		
$AA_LEAD_TYP[LW]	INT	R	R	リーディング値のソース | 1: 実際値 2: セットポイント 3: シュミレーションされたリーディング値	リーディング軸
$AA_LEAD_SP[LW]				シュミレーションされたリーディング値位置	
$AA_LEAD_SV[LW]				シュミレーションされたリーディング値速度	
$AA_LEAD_P[LW]				現実のリーディング値位置	
$AA_LEAD_V[LW]				現実のリーディング値速度	
$AA_SYNC[FA]				リーディング値カップリングにおける追従軸のカップリングステータス | 0: 同期していない 1: 同期性（粗） 2: 同期性（微） 3: 粗および微	
$AA_COUP_ACT[FA]				軸カップリングのタイプ | 0: カップリングされない 3: タンジェントでトレイルする 4: 同期主軸 8: カップリングされたモーション軸 16: リーディング値カップリングの追従軸	

14.3.12 主軸データ

名称	タイプ	アクセス		意味 | 値の範囲	インデックス | 値の範囲
		パートプログラム	シンクロナイズドアクション		
$AC_MSNUM	INT	R	R	マスタ主軸の番号	n: 主軸軸 0: マスタ主軸 | 0 - 最大主軸番号
$AC_SDIR[n]				主軸回転の現在有効な方向 |	
				3: 右回転	
				4: 左回転	
				5: 主軸停止	

名称	タイプ	アクセス		意味 | 値の範囲	インデックス | 値の範囲
		パートプログラム	シンクロナイズドアクション		
$AC_SMODE[n]	INT	R	R	現在有効な主軸モード |	
				0: インストールされた主軸はなし	
				1: 速度制御モード	
				2: 位置決めモード	
				3: 同期モード	
				4: 軸モード	
$AA_S[n]	REAL			実際主軸速度 ; 符号は，回転方向に対応する	
$AA_COUP_OFFS[axis]				セットポイントとして同期主軸の位置オフセット	主軸 (S1, S2) | 最大主軸番号

ドライブデータ主軸

名称	タイプ	アクセス		意味 | 値の範囲	インデックス | 値の範囲
		パートプログラム	シンクロナイズドアクション		
$VA_COUP_OFFS[axis]	REAL	R	R	実際値としての同期主軸の位置オフセット	主軸 (S1, S2) | 最大主軸番号
$VA_COUP_ACT[axis]				追従主軸の現在のカップリングステータス	
$P_S[n]	REAL			最新のプログラム主軸速度	
$P_SDIR[n]	INT			プログラムされるべき最新の主軸回転方向 3: 右回転 4: 左回転 5: 主軸停止	
$P_SMODE[n]				最新のプログラム主軸モード 0: 主軸がないかまたは別のチャンネルにプログラムされているかまたは PLC 主軸 1: 速度制御モード 2: 位置決めモード 3: 同期モード 4: 軸モード	
$P_GWPS[n]	BOOL	R		1 (TRUE): 一定のグラインディングホイール表面速度 ON	

14.3.13 プログラムされた値 (前処理で同期化する)

名称	タイプ	アクセス		意味 | 値の範囲	インデックス | 値の範囲
		パートプログラム	シンクロナイズドアクション		
$P_ACTID[n]	BOOL	R/W		1 (TRUE): ID n が有効で，モーダルなシンクロナイズドアクション	n: | 1 - 16
$P_AXN1	AXIS	R		ジオメトリ軸の現在アドレス - 横座標	
$P_AXN2				ジオメトリ軸の現在アドレス - 縦座標	
$P_AXN3				ジオメトリ軸の現在アドレス - 垂直座標	
$P_ACTGEOAX	AXIS	R		現在のジオメトリ軸割当て，面に依存する。GEOAX(1,X, 2,Y,3,Z) を使ってプログラムされた現在のジオメトリ軸割当てが行なわれる。	
$P_DRYRUN	BOOL			0 (FALSE): ドライラン ON 1 (TRUE): ドライラン OFF	
$P_SEARCH				1 (TRUE): ブロックサーチ (計算が有る場合と無い場合のいずれか) が有効である	
$P_SEARCH1				1 (TRUE): 計算のあるブロックサーチが有効である	
$P_SEARCH2	BOOL			1 (TRUE): 計算のないブロックサーチが有効である	
$P_SEARCHL				最新の選択ブロックサーチタイプの識別子をリターンする : 0 (ブロックサーチなし), 1 (計算のないブロックサーチ), 2 (輪郭における，計算のあるブロックサーチ), 4 (ブロック終点における，計算のあるブロックサーチ)	
$P_CTABDEF				1 (TRUE): カーブテーブルの定義が有効である	
$PI	REAL			円周率 π = 3.1415927	
$P_PROGPATH	STRING			現在のディレクトリからのサブプログラムのコール : PCALL $P_PROGPATH <<_N_MYSUB_SPF	

$P_SUBPAR[n]	BOOL			パラメータ転送を使って，サブプログラムコールのために，有効な転送パラメータがプログラムされたかどうかのクエリ。1 (TRUE): 転送パラメータがプログラムされた 0 (FALSE): 初期設定パラメータ	n= PROC 命令における定義による転送パラメータ番号
$P_MC	INT			0 (FALSE): モーダルサブプログラムコールなし 1 (TRUE): モーダルサブプログラムコール	
$P_REPINF	INT	R		REPOS コマンドを使って，再位置決めのためのステータス情報 0 (FALSE): 再位置決めが可能でない理由は次のとおり - コールが ASUP で伝えられない - リセットステータスで始動された ASUP からコールが送られる - JOG モードで始動された ASUP からコールが送られる 1 (TRUE): 再位置決め可能	
$P_SIM	BOOL	R		1 (TRUE): 進行中のシュミレーション	
$P_GG[n]	INT			1 つの G グループの現在の G 機能 |PLC インタフェースについての識別	n: G グループ番号
$P_EXTGG[n]	INT			シーメンスモードでのみ使用可能 : 外部 NC 言語を用いて，G グループの現在の G 機能 (PLC インタフェースとしてのインデックス)n: G グループ番号	n: G グループの番号

14.3.14 チャンネルへのコマンドとチャンネルからのコマンド

名称	タイプ	アクセス		意味 | 値の範囲	インデックス | 値の範囲
		パートプログラム	シンクロナイズドアクション		
$P_MMCA	STRING	R/W		MMC ジョブ認識	
$A_PROTO	BOOL			プロトコル機能を起動する／停止する	

14.3.15 ツールデータ

名称	タイプ	アクセス		意味 | 値の範囲	インデックス | 値の範囲
		パートプログラム	シンクロナイズドアクション		
$P_AD[n]	REAL	R/W		有効なツールオフセット	n: パラメータ番号 | 1 - 25
$P_AEP[<Achse>]	REAL	R		輪郭始点 P4 における WCS での軸位置 <axis> (有効なツール径補正をまったく考慮しない)	- ・< $P_APR[<axis>] < ?
$P_APDV	BOOL	R		$P_APR および $P_AEP の両方のシステム変数の内容が有効である場合，すなわち，システム変数が，プログラムされた最新の WAB アプローチブロックと関連した位置値を含む場合，TRUE をリターンする。	FALSE, TRUE
$P_APR[<axis>]	REAL			輪郭始点 P3 において，WCS での軸位置 <axis> (有効なツール径補正をまったく考慮しない)	- ・< $P_APR[<axis>] < ?
$P_ATPG[n]				現在のツール関連グラインディングデータ	n: パラメータ番号 | 1 - 9
$P_TOOLxxx[t]	REAL	R			t: T 番号 | 1 - SLMAXTOOLNUMBER
$P_TOOL	INT			有効なツールエッジ | D0 - D9	
$P_TOOLL[n]	REAL			有効なツールの全長	n: 長さ | 1 - 3
$P_TOOLNO	INT			有効なツール番号 | T0 - T32000	
$P_TOOLR	REAL			有効なツール径 (全体)	
$P_TOOLND[t]	INT			ツールエッジ番号 t	
$P_TOOLEXIST[t]	BOOL			1 (TRUE): T 番号のついたツール t が存在する	
$P_TOOLMN[t]	INT			ツール t のマガジン番号	
$P_TOOLMLN[t]				ツール t のマガジン番号	
$_VDITCP[n]	INT	R/W		VDI インタフェースを使ってツール管理に使用可能なパラメータ	n: | 1 - 3

14.3.16 軸コンテナ

名称	タイプ	アクセス		意味 | 値の範囲	インデックス | 値の範囲
		パートプログラム	シンクロナイズドアクション		
$AC_AXCTSWA	BOOL	R/	R/	IF $AC_AXCTSWA[n] == TRUE GOTOB MARK1 読取り： TRUE: チャンネルは，チャンネルは，軸コンテナ名 n について軸コンテナ回転をイネーブルし，まだ完了していない。 FALSE: 軸コンテナ回転が終了される。	
$AN_AXCTAS	INT	R/	R/	読取り : 軸コンテナ回転の現在の回転 : 軸コンテナが現在切替えてられているスロットがいくつあるかによって，軸コンテナ名 n を使って，軸コンテナを指定する。 数値範囲は，0 から，軸コンテナの使用中のスロットの最大数 -1 である。	
$AN_AXCTSWA	BOOL	R/	R/	EVERY $AN_AXCTSWA[n] == TRUE DO M99 Read: TRUE: 1　軸コンテナ回転が，軸名 n で，軸コンテナについて現在実行されている。 FALSE: 有効な軸コンテナ回転はない。	

Yaskawa Siemens CNC シリーズ

本製品の最終使用者が軍事関係であったり，用途が兵器などの製造用である場合には，「外国為替及び外国貿易法」の定める輸出規制の対象となることがありますので，輸出される際には十分な審査及び必要な輸出手続きをお取りください。

製品改良のため、定格、寸法などの一部を予告なしに変更することがあります。
この資料についてのお問い合わせは、当社代理店もしくは、下記の営業部門にお尋ねください。

製造

株式会社 安川電機　　シーメンスAG

販売

シーメンス・ジャパン株式会社

工作機械営業本部
東京都品川区大崎1-11-1 **ゲートシティ大崎ウエストタワー**　〒141-8644
TEL（03）3493-7411　FAX（03）3493-7422

アフターサービス

カスタマーサービス事業本部
TEL 0120-996095（フリーダイヤル）　FAX (03)3493-7433

シーメンス・ジャパン株式会社
http://www.siemens.co.jp

02- WSC02, 02-WSC09