NUM
1020/1040/1060

MANUEL DE PROGRAMMATION COMPLEMENTAIRE

0100938872/2

Malgré tout le soin apporté à l'élaboration de ce document, NUM ne peut garantir l'exactitude de toutes les informations qu'il contient et ne peut être tenu responsable, ni des erreurs qu'il pourrait comporter, ni des dommages qui pourraient résulter de son utilisation ou de son application.

Les produits matériels, logiciels et services présentés dans ce document sont à tout moment susceptibles d'évolutions quant à leurs caractéristiques de présentation, fonctionnement ou utilisation. Leur description ne peut en aucun cas revêtir un aspect contractuel.

Les exemples de programmation sont décrits dans ce manuel à titre didactique. Leur utilisation dans des programmes d'applications industrielles nécessite des adaptations spécifiques selon l'automatisme concerné et en fonction du niveau de sécurité demandé.

Table des matières

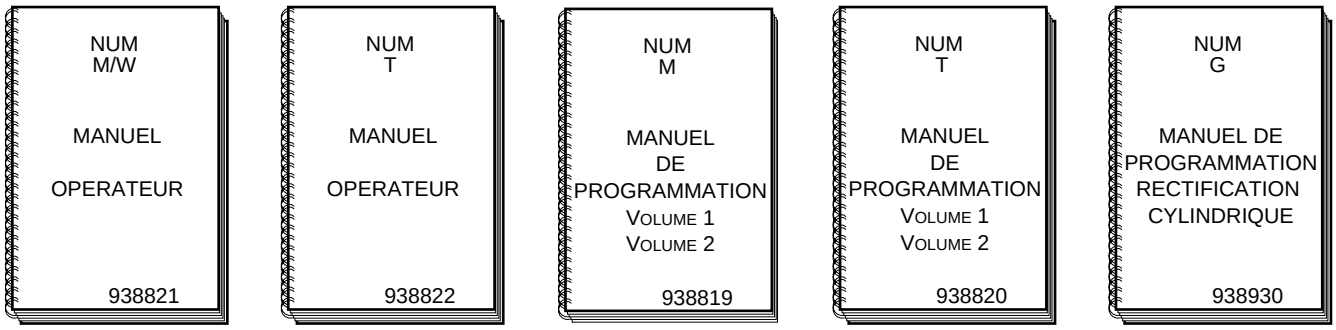
Tableau des mises à jour

EVOLUTIONS DE LA DOCUMENTATION		
Date	**Indice**	**Nature des évolutions**
02-93	0	Création du document (Conforme au logiciel indice D)
01-95	1	Mise en conformité avec l'indice G du logiciel Evolutions du manuel - Fonction RTCP - Fonction N/M AUTO Prise en compte des évolutions Logiciel indice E : - Adressage de la fonction appelante par [.RG80] dans sous programme appelé par fonction G - Adressage par [.IRDI(i)] définissant l'origine des décalages angulaires Logiciel indice F : - Transformations de coordonnées
04-96	2	Mise en conformité avec l'indice K du logiciel Evolutions du manuel : - Interpolation polynomiale lisse - En adressage par [.IBX(i)], [.IRX(i)] et [.IBI(i)], [.IRI(i)] ajout des index 10, 11, 4 et 5, liés au G21 et G22 Prise en compte des évolutions Logiciel indices H et J : - Mise à jour de la fonction N/M AUTO

Structure de la documentation produit NUM 1020/1040/1060

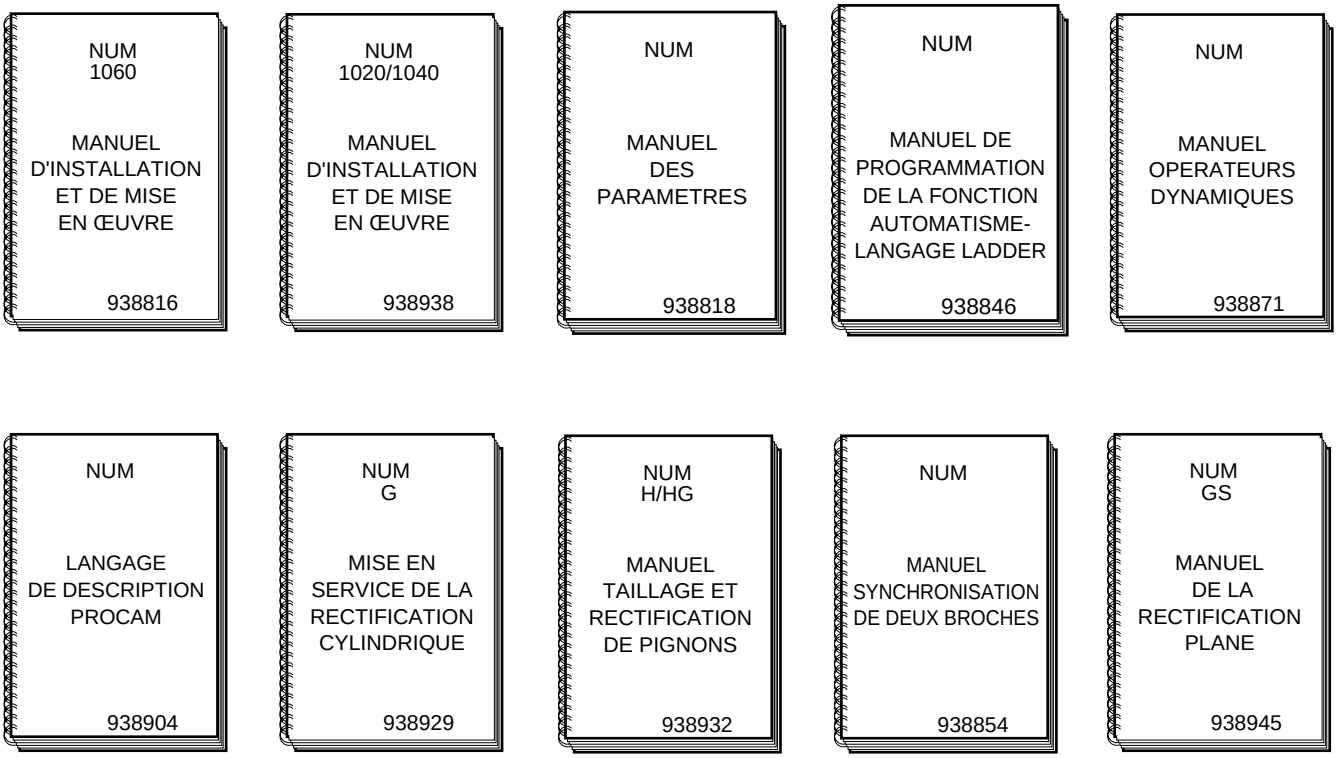
Documents utilisateur

Ces documents sont destinés à l'exploitation de la commande numérique.



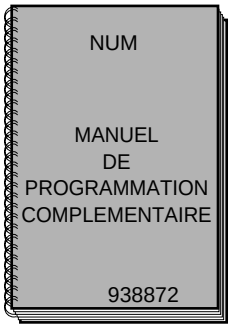
Documents intégrateur

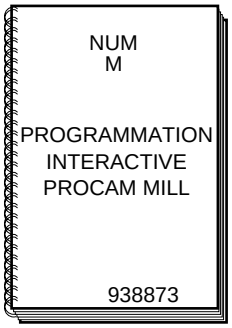
Ces documents sont destinés à la mise en œuvre de la commande numérique sur une machine

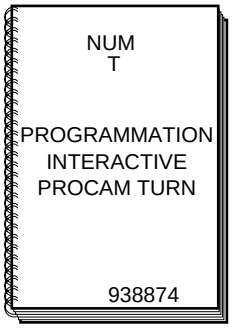


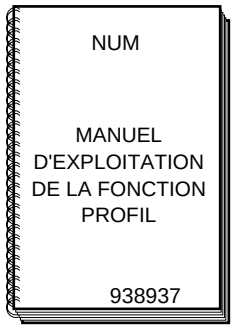
Documents spécifiques de programmation

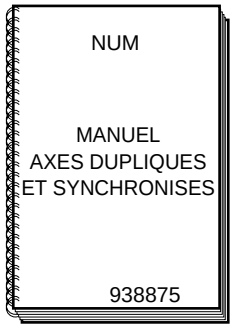
Ces documents concernent des applications spécifiques de programmation sur commande numérique.

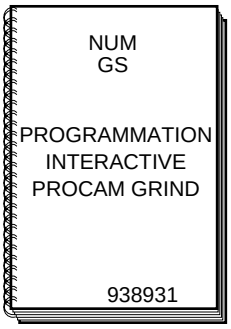


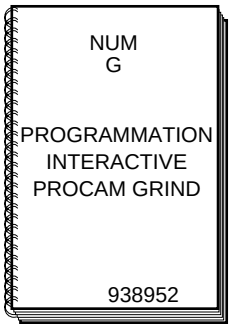






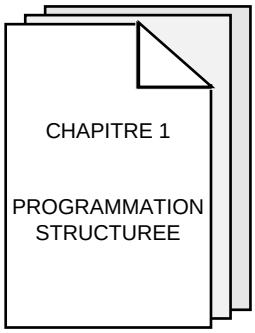




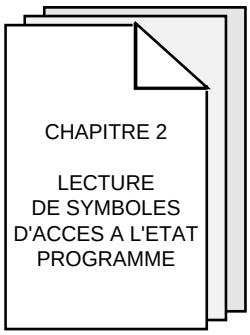


Manuel de programmation complémentaire

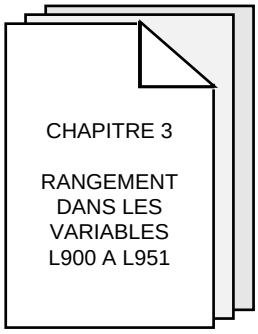
Contenu du manuel



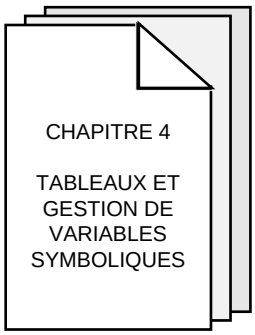
Présentation des commandes offrant la possibilité de programmer des sauts et des boucles sous forme structurée.



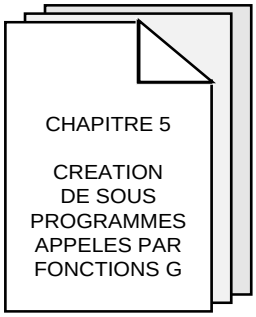
Présentation des symboles permettant d'obtenir la visibilité des fonctions programmées et du contexte programme lors de l'appel d'un cycle par fonction G.



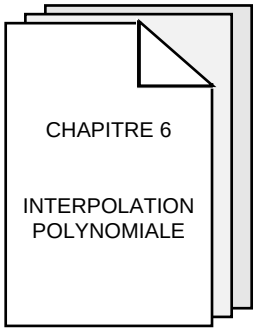
Possibilité de rangement des valeurs liées aux arguments ou fonctions programmées dans les variables L900 à L951 lors de l'appel d'un cycle par fonction G.



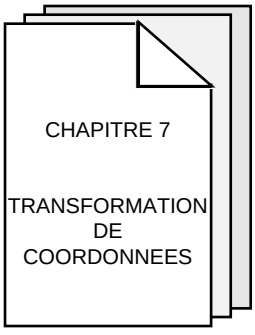
Possibilité de créer et d’assurer la gestion des tableaux de variables symboliques dans le but de ranger des fonctions et trajectoires d’un profil.



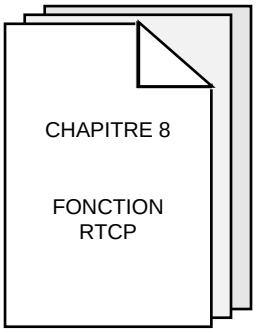
Possibilité de créer des sous programmes appelés par fonctions G.



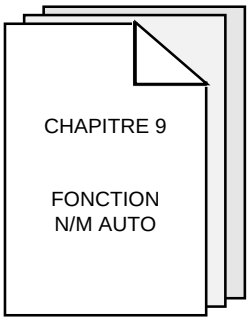
Possibilité de définir des trajectoires à partir de polynômes.



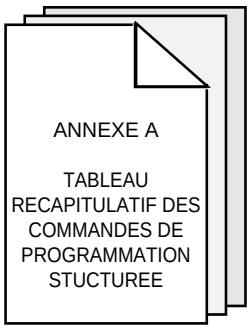
Possibilité de transformer des coordonnées à l'aide d'une matrice carrée.



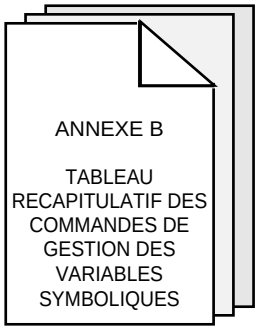
Possibilité de prendre en compte la chaîne cinématique d'une machine permettant d'orienter l'outil par rapport à la pièce et de le faire pivoter autour de son centre.



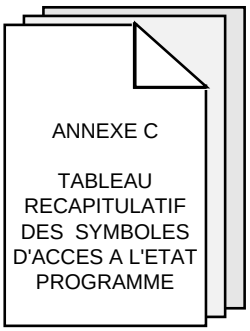
Possibilité de commander des axes N/M AUTO alors que d'autres axes de la machine suivent une trajectoire programmée.



Présentation sous forme de tableau des commandes de programmation structurée.

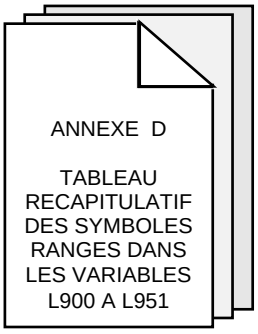


Présentation sous forme de tableau des commandes de gestion des variables symboliques.



Présentation sous forme de tableau des listes des symboles d'accès à l'état programme.

- adressage des fonctions G,
- adressage des fonctions M,
- adressage d'une liste de bits,
- adressage d'une valeur,
- adressage d'une liste de valeurs.



Présentation sous forme de tableau des listes des symboles de rangement dans les variables L900 à L951.

- Symboles de rangement dans les variables L900 à L925
- Symboles de rangement dans les variables L926 à L951

Utilisation du manuel de programmation complémentaire

Conventions d'écriture des syntaxes

Les lignes d'écriture (blocs) utilisées en programmation sont constituées de commandes, symboles, variables, fonctions ou arguments.

Chacune des fonctions présentées dans le manuel est soumise à une syntaxe d'utilisation ; l'ensemble des syntaxes fixe les règles d'écriture des blocs du programme.

Certaines syntaxes sont présentées sous formes de une ou plusieurs lignes dont l'écriture est simplifiée par l'utilisation des conventions suivantes :

- la (ou les) fonctionnalité à laquelle est rattachée la syntaxe est mise en évidence par l'utilisation de caractères gras,
- les «..» ou «lettres minuscules» après une ou plusieurs lettres adresses ou signe remplacent une valeur numérique (par exemple N..),
- les «...» remplacent une chaîne de caractères ou d'adresses sensiblement identiques à celles précédemment utilisées dans le bloc (par exemple [Symb1] / [Symb2] ...),
- les «xx» après une ou plusieurs lettres adresses remplacent des caractères alphanumériques (par exemple [.IBxx(i)]),
- les «xxx» après une lettres adresses remplacent des valeurs numériques (par exemple Gxxx).

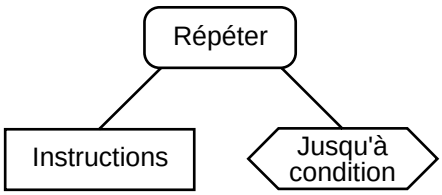
Exemples

Syntaxe de création d'un tableau de variables symboliques

P.BUILD [TAB(7,NB)] H.. N.. +n N..+n

Syntaxe de boucles «jusqu'à» et sa représentation graphique

REPEAT
(instructions)
UNTIL (condition)



Index

L'index figure en fin de volume et permet d'accéder à des renseignements ponctuels par mots clés.

Questionnaire

Afin de nous aider à améliorer la qualité de notre documentation, nous vous demandons de bien vouloir nous retourner le questionnaire figurant en fin de volume.

Agences

La liste des agences NUM figure en fin de volume.

1 Programmation structurée

Le système offre la possibilité de programmer les sauts et les boucles sous une forme structurée, ce qui permet une meilleure lisibilité des programmes et une plus grande facilité de programmation pour des programmes complexes.

Les outils de programmation décrits dans ce chapitre sont utilisés pour la création de sous programme appelé par fonction G (Voir chapitre 5).

1.1 Généralités

Un ensemble structuré commence et se termine toujours par des mots clés.

Il débute par les mots clés suivants :
IF
REPEAT
WHILE
FOR

Il se termine par :
ENDI avec IF
UNTIL avec REPEAT
ENDW avec WHILE
ENDF avec FOR

Le mot ELSE peut s'intercaler entre les mots IF et ENDI.

1.1.1 Commandes utilisées dans les ensembles structurés

- Conditions d'exécution d'instructions : IF, THEN, ELSE, ENDI
- Boucles "répéter jusqu'à" : REPEAT, UNTIL
- Boucles "répéter tant que" : WHILE, DO, ENDW
- Boucles avec variable de contrôle : FOR, TO, DOWNTO, BY, DO, ENDF
- Sortie d'une boucle : EXIT

1.1.2 Règles générales de syntaxe

Les mots IF, REPEAT, WHILE, FOR, ENDI, UNTIL, ENDW et ENDF doivent être les premiers mots du bloc (pas de numéro de séquence).

Les mots IF, REPEAT, THEN, ELSE, UNTIL, WHILE, DO et DOWNTO doivent être obligatoirement suivis d'un espace, par exemple :

WHILEL0 <3 n'est pas reconnu par le système, il faut programmer WHILE L0 <3

Les mots DO et THEN doivent suivre immédiatement la condition; si toutefois ces deux mots ne sont pas situés dans le même bloc que les mots IF, WHILE et FOR, ils doivent être les premiers mots du bloc suivant.

Des blocs avec numéro de séquence (N..) sont permis dans les boucles.

Les blocs commençant par les mots ENDI, ENDW, ENDF, EXIT ou UNTIL ne doivent pas comporter de fonctions de programmation ISO.

Dans un même bloc, l'un des mots DO, THEN ou ELSE peut être suivi de fonctions de programmation ISO.

Par exemple :

```
WHILE L1 < 3 DO G91 X12
```

ou

```
WHILE L1 < 3
DO G91 X12
```

La programmation suivante est refusée :

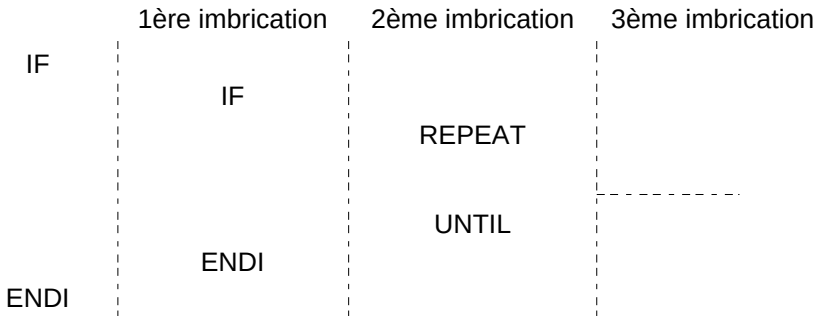
```
WHILE L0 < 3 G91 X10
DO         interdit dans
           la condition
```

1.1.3 Imbrications et sauts

Imbrications

15 niveaux d'imbrications d'ensembles structurés sont possibles et sont indépendants des appels de sous programmes par la fonction G77 ...

Par exemple :



Sauts

La programmation d'un saut, conditionnel ou non G79 ... est autorisée dans un ensemble structuré, mais il doit impérativement mener au plus bas niveau d'imbrication du programme ou du sous programme courant.

Par exemple :

```
%1
IF
     WHILE
          G79 N100  autorisé
          G77 H2
          G79 N50   erreur
          G79 N30   erreur
          N30
     ENDW
     N50
ENDI
N100
M02
```

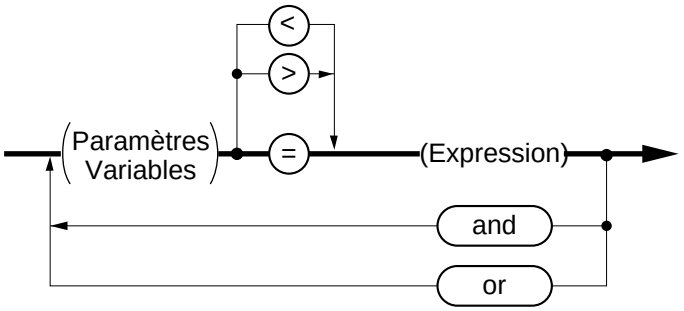
```
%2
G79 N100            autorisé
REPEAT
     IF
          G79 N100  autorisé
          G79 N50   interdit
     ENDI
     N50
UNTIL
N100
```

1.2 Commandes de programmation structurée

1.2.1 Diagramme d'une condition

Une condition doit suivre l'un des mots IF, WHILE ou UNTIL et doit être positionnée dans le même bloc. Si le bloc contient des bornes et éventuellement l'incrément, ceux-ci doivent suivre le mot FOR.

Diagramme d'une condition



Variables : Toutes les variables utilisées en programmation paramétrée : variables L, paramètres E et variables symboliques.

Expression : Suite de paramètres et de valeurs immédiates chaînés par les symboles " +, -, *, /, !, & " (Voir chapitre 6 du manuel de programmation). Les calculs sont effectués en série de gauche à droite.

1.2.2 Conditions d'exécution d'instructions

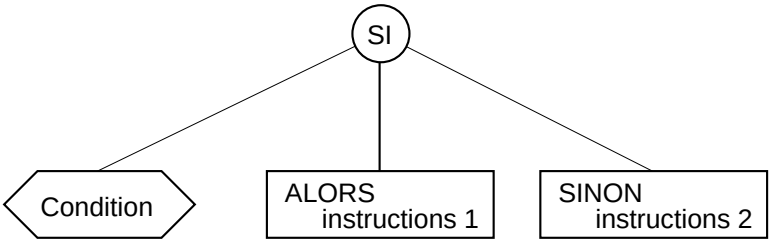
Syntaxe

```
IF(condition) THEN
   (instructions 1)
ELSE
   (instructions 2)
ENDI
```

Si la condition est vraie, les "instructions 1" seront exécutées. Dans le cas contraire les "instructions 2" seront exécutées.

Le mot ELSE est facultatif.

Représentation graphique



Exemple

```
IF E70000> 100 AND E70000 <200 THEN
     G77 H100
ELSE
     G77 H500
ENDI
```

Le mot THEN peut être éventuellement programmé en tête du bloc suivant et être suivi de fonctions à exécuter.

Par exemple :

```
IF L4 < 8
THEN L6 = L2+1 XL6
ENDI
```

1.2.3 Boucles "répéter jusqu'à"

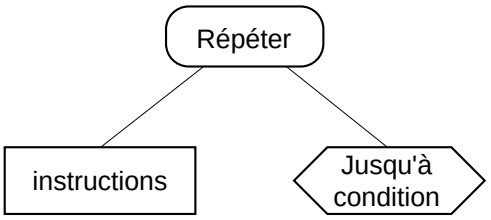
Syntaxe

REPEAT
(instructions)
UNTIL (condition)

Les instructions seront exécutées, puis répétées jusqu'à ce que la condition soit vraie.

Même si la condition est vraie depuis le début, les instructions seront exécutées une fois.

Représentation graphique



Exemple

Attente d'une réponse correcte

```
REPEAT
     $ SORTIE DU PROGRAMME (O/N) ? :
     [REPONSE]= $
UNTIL [REPONSE] = 14 OR [REPONSE] = 15
```

Un appui sur "O" renvoie la valeur 15 et "N" la valeur 14 (rang des lettres O et N dans l'alphabet)

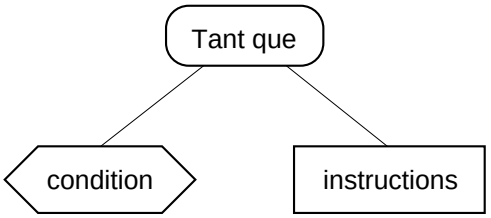
1.2.4 Boucles "répéter tant que"

Syntaxe

```
WHILE (condition) DO
      (instructions)
ENDW
```

Tant que la condition est vraie, les instructions seront exécutées. Cependant, si la condition est fausse dès le début, il n'y aura pas exécution des instructions, ce qui diffère de la structure REPEAT UNTIL.

Représentation graphique



Le mot DO peut être éventuellement programmé en tête du bloc suivant et être suivi de fonctions à exécuter.

Par exemple :

```
WHILE (condition)
DO XL6
ENDW
```

1.2.5 Boucles avec variable de contrôle

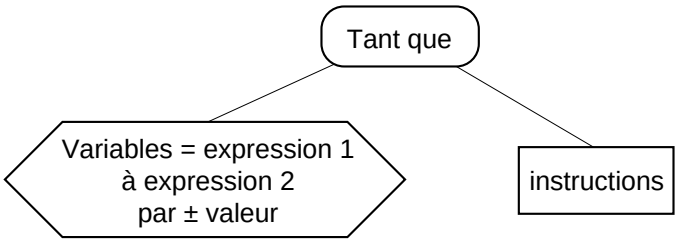
Syntaxe

FOR (variable) = (expression 1) **TO/DOWNTO** (expression 2) **BY** (valeur) **DO**
(instructions)
ENDF

Les instructions seront exécutées avec "variable = expression 1". Ensuite cette "variable" sera incrémentée (TO) ou décrémentée (DOWNTO) de "valeur" avant une nouvelle exécution des "instructions" et ce jusqu'à ce que la "variable" soit égale à "expression 2".

Le mot BY est facultatif.

Représentation graphique



La variable employée peut être :
- une variable programme L,
- une variable symbolique,
- un paramètre E80000, E81000 ou E82000 (attention à l'arrêt des calculs pour L100 à L199, Exxxxx et variables symboliques).

Les valeurs des expressions sont entières, positives ou négatives. Le système effectue éventuellement un arrondi (au chiffre inférieur jusqu'à 0.5 et au chiffre supérieur pour > 0.5)

Avec le mot TO, la boucle n'est exécutée que lorsque la valeur courante de la variable est inférieure ou égale à la valeur finale (incrémentation de la variable).

Avec le mot DOWNTO, La boucle n'est pas exécutée lorsque la valeur courante est supérieure ou égale à la valeur finale (décrémentation de la variable).

Par défaut, la valeur du mot BY est la valeur incrémentée ou décrémentée de la variable à chaque passage (par défaut la valeur est égale à 1).

La valeur du mot BY doit être toujours positive. Si celle-ci est négative, il y a inversion des mots TO et DOWNTO.

Exemple

Mise à zéro des données outils

```
FOR L1=1 TO 20
DO
  L2=56000+L1
  EL2=0
ENDF
```

1.2.6 Sortie de boucle

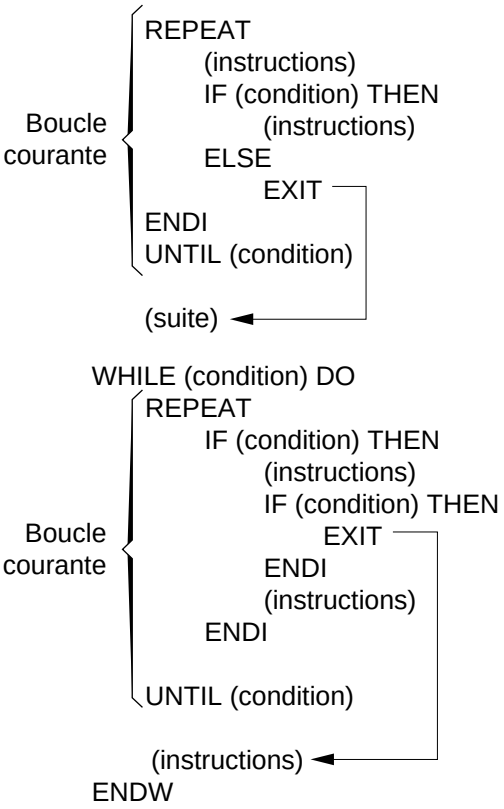
Syntaxe

EXIT

L'instruction EXIT permet de sortir de la boucle itéractive (FOR, WHILE ou REPEAT) et de descendre au niveau d'imbrication inférieur en éliminant les mots IF intérieurs à la boucle.

Le mot EXIT est toujours conditionné par le mot IF.

Exemple

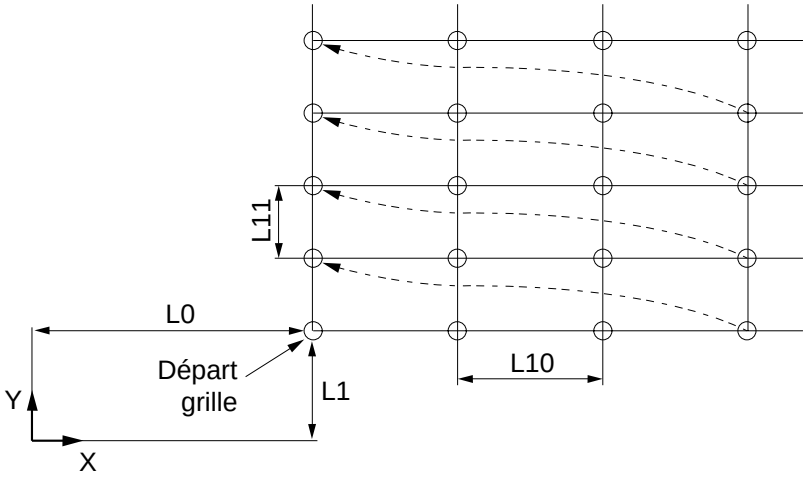


1.3 Exemple de programmation structurée

Perçages de trous suivant une grille

L2 = nombre de trous suivant X

L3 = nombre de trous suivant Y



```
%20
$ X DEPART :
L0=$
$ Y DEPART :
L1=$
$ PAS EN X :
L10=$
$ PAS EN Y :
L11=$
L4=0
WHILE L4 <> 15 DO                          Test réponse oui : O
     REPEAT $ NOMBRE DE POINTS EN X :
          L2=$
     UNTIL L2>0                            Test nombre de colonnes supérieur à 0
     REPEAT $ NOMBRE DE POINTS EN Y :
          L3=$
     UNTIL L3>0                            Test nombre de lignes supérieur à 0
     REPEAT $ GRILLE PTS /X:
          $=L2
          $+ /Y:
          $=L3
          $+ OK (O/N) :
          L4=$
     UNTIL L4=14 OR L4=15                  Attente réponse : N (non) ou O (oui)
ENDW
$
N.. G00 G52 X0 Z0
N.. T1 D1 M06
N..
FOR L5=1 TO L3 DO                          Boucle sur les lignes
     $ POSITIONNEMENT LIGNE:
     $=L5
     XL0 L6=L5-1*L11+L1 YL6
     G00 Z0
     FOR L4=1 TO L2 DO                     Boucle sur les colonnes
          $ PERCAGE DU POINT LIGNE:
          $=L5
          $+COLONNE:
          $=L4
          L6=L4-1*L10+L0 XL6
          G01 Z-10 F50
          G00 Z0
     ENDF
ENDF
N..
M02
```

2 Lecture des symboles d'accès à l'état programme

Les outils de programmation décrits dans ce chapitre sont nécessaires à la création de sous programme appelé par fonction G (Voir chapitre 5).

2.1 Généralités

Les symboles d'accès à l'état programme permettent d'obtenir la visibilité des fonctions programmées dans le bloc d'appel d'un cycle d'usinage par fonction G ainsi que des renseignements sur le contexte programme pièce au moment de l'appel.

Ces symboles permettent de lire les données modales du bloc courant.

Ces symboles à lecture seule sont accessibles par programmation paramétrée.

Ces symboles peuvent être :
- des symboles d'accès aux données du bloc courant,
- des symboles d'accès aux données du bloc précédent.

Chaque symbole adresse une donnée ou une liste de données qui a la forme d'un tableau à une dimension.

2.2 Symboles d'accès aux données du bloc courant

Ces symboles peuvent être :
- des symboles adressant des valeurs booléennes,
- des symboles adressant des valeurs numériques.

Syntaxe générale

Variable = [•symbole(i)]

Variable	Variable programme L, variable symbolique [symb], paramètre E.
[•symbole(i)]	Symbole entre crochets, précédé d'un point décimal, suivi éventuellement d'un index (i).

2.2.1 Symboles adressant des valeurs booléennes

Les symboles adressant des valeurs booléennes associés aux fonctions programmées permettent de déterminer si celles-ci sont actives ou non.

Les valeurs booléennes sont définies par 0 ou 1.

2.2.1.1 Adressage des fonctions G

[•BGxx]	Adressage des fonctions G.

Le symbole [•BGxx] permet de déterminer si les fonctions G spécifiées par xx sont actives ou non, par exemple :
[•BGxx]=0 : fonction Gxx révoquée
[•BGxx]=1 : fonction Gxx active

Liste des fonctions G

[•BG00] [•BG01] [•BG02] [•BG03] [•BG17] [•BG18] [•BG19]
[•BG20] [•BG21] [•BG22] [•BG29] [•BG40] [•BG41] [•BG42]
[•BG70] [•BG71 [•BG80] [•BG81] [•BG82] [•BG83] [•BG84]
[•BG85] [•BG86] [•BG87] [•BG88] [•BG89] [•BG90] [•BG91]
[•BG93] [•BG94] [•BG95] [•BG96] [•BG97]

2.2.1.2 Adressage des fonctions M

[•BMxx]	Adressage des fonctions M.

Le symbole [•BMxx] permet de déterminer si les fonctions M spécifiées par xx sont actives ou non, par exemple :
[•BMxx]=0 : fonction Mxx révoquée
[•BMxx]=1 : fonction Mxx active

Liste des fonctions M

[•BM03] [•BM04] [•BM05] [•BM07] [•BM08] [•BM09] [•BM10] [•BM11]
[•BM19] [•BM40] [•BM41] [•BM42] [•BM43] [•BM44] [•BM45] [•BM48]
[•BM49] [•BM62] [•BM63] [•BM64] [•BM65] [•BM66] [•BM67] [•BM68]
[•BM69] [•BM997] [•BM998] [•BM999]

Exemple

```
%100
N10 G00 G52 Z0 G71
N..
N40 G97 S1000 M03 M41
N50 M60 G77 H9000                          Appel de sous-programme de
                                           contrôle outil
N..
N350 M02

%9000
VAR
[GPLAN] [MGAMME] [MSENS] [GINCH] [GABS]
[XRETOUR] [YRETOUR] [ZRETOUR]
ENDV

$ SAUVEGARDE DU CONTEXTE
N10 [GPLAN]=17*[•BG17]
    [GPLAN]=18*[•BG18]+[GPLAN]             } Plan d'interpolation
    [GPLAN]=19*[•BG19]+[GPLAN]
N20 [MGAMME]=40*[•BM40]                    Gamme
    [MGAMME]=41*[•BM41]+[MGAMME]
N30 [MSENS]=03*[•BM03]
    [MSENS]=04*[•BM04]+[MSENS]             Sens de rotation
    [MENS]=05*[•BM05]+[MSENS]
N40 [GINCH]=70*[•BG70]                     Pouces
    [GINCH]=71*[•BG71]+[GINCH]
N50 [GABS]=90*[•BG90]                      Absolu
    [GABS]=91*[•BG91]+[GABS]
[XRETOUR]=E70000
[YRETOUR]=E71000
[ZRETOUR]=E72000
N60 G90 G70 G00 G52 Z0
N70 G52 X100 Y100 M05                      Position de contrôle outil
N80 G52 G10 Z-500
N90 G52 Z0
N100 G52 Z [ZRETOUR]                       Retour à la position Z
N110 G52 X [XRETOUR]
N120 G52 Y [YRETOUR]
G[GPLAN] M[MGAMME]
M[MSENS] G[GINCH] G[GABS]                  Restauration du contexte
```

2.2.1.3 Adressage d'une liste de bits

[•IBxx(i)]	Adressage d'une liste de bits.

Le symbole [•IBxx(i)] permet d'adresser une liste de bits correspondant aux éléments spécifiés par xx.

Ces valeurs sont booléennes, 0 ou 1.

L'index (i) définit le rang de l'élément dans la liste.

[•IBX(i)] Liste des axes programmés dans le bloc courant.
Index i = 1 à 11.
Cette liste non modale peut rester mémorisée et donc lue par programmation paramétrée si le système se trouve dans l'état G999.
i = 1 : axe X
i = 2 : axe Y
i = 3 : axe Z
i = 4 : axe U
i = 5 : axe V
i = 6 : axe W
i = 7 : axe A
i = 8 : axe B
i = 9 : axe C
i = 10 : en G21, l'index 10 adresse l'axe Y.
: en G22, l'index 10 adresse l'axe Z.
i = 11 : en G21, l'index 11 adresse l'axe X.
: en G22, l'index 11 adresse l'axe Y.

[•IBX1(i)] Liste des axes programmés depuis le début du programme jusqu'au bloc courant.
Index i = 1 à 9, identiques à la liste des axes programmés dans le bloc courant (Voir [•IBX(i)]).
Les axes programmés par rapport à l'origine mesure (G52) sont révoqués dans cette liste, mais y figurent à nouveau s'il y a retour à la programmation par rapport à l'origine programme.

[•IBX2(i)] Liste mémorisant les derniers axes programmés : primaires X Y Z ou secondaires U V W.
Index i = 1 à 6, à l'exclusion des axes A B et C, la liste est identique aux axes programmés dans le bloc courant (Voir [•IBX(i)]).
La programmation de l'axe d'un groupe révoque son axe équivalent dans l'autre groupe, par exemple : l'axe V programmé met à 0 le bit relatif à X et à 1 celui de V.

[•IBXM(i)] Etat miroir ou non miroir sur les axes. L'état miroir est signalé par un bit à 1, l'état non miroir par un bit à 0.
Index i = 1 à 9, identiques à la liste des axes programmés dans le bloc courant (Voir [•IBX(i)]).

[•IBI(i)] Liste des arguments I, J et K programmés dans le bloc courant.
Index i = 1 à 5.
Liste uniquement modale dans l'état G999.
i = 1 : argument I
i = 2 : argument J
i = 3 : argument K
i = 4 : en G21, l'index 4 adresse la composante J.
: en G22, l'index 4 adresse la composante K.
i = 5 : en G21, l'index 5 adresse la composante I.
: en G22, l'index 5 adresse la composante J.

[•IBP(i)] Liste des arguments P, Q et R programmés dans le bloc courant.
Index i = 1 à 3.
Liste uniquement modale dans l'état G999.
i = 1 : argument P
i = 2 : argument Q
i = 3 : argument R

2.2.2 Symboles adressant des valeurs numériques

Ces symboles adressent des valeurs numériques permettent de lire les données modales du bloc courant.

2.2.2.1 Adressage d'une valeur

[•Rxx] Adressage d'une valeur.

Le symbole [•Rxx] permet d'adresser une valeur correspondant aux éléments spécifiés par xx.

[•RF] Valeur de la vitesse d'avance
(unité selon la fonction programmée G93, G94 ou G95).

[•RS] Valeur de la vitesse de broche
(G97 : format selon les caractéristiques de broche déclarées dans le paramètre machine P7).

[•RT] Numéro d'outil.

[•RD] Numéro du correcteur d'outil.

[•RN] Numéro de la dernière séquence (bloc) rencontrée.
Si le numéro de bloc est absent, c'est le dernier bloc numéroté qui a été analysé.

[•RED] Valeur du décalage angulaire.

[•REC] Valeur de l'indexation de broche (en fraisage).

[•RG4] Valeur de la temporisation programmée (G04 F..).
Fonction non modale pouvant cependant rester mémorisée ; sa valeur peut donc être lue si le système se trouve dans l'état G999 ou G998.

[•RG80] Numéro de la fonction appelante dans un appel de sous programme par fonction G.
Dans un sous programme appelé par fonction G, le numéro de la fonction appelante est adressé par [.RG80] (Dans l'état G80, sa valeur est nulle).

[•RNC] Valeur de NC (numéro de courbe spline).

[•RDX] Orientation de l'axe de l'outil.
Elle est définie par les signes et valeurs suivants :

+1 pour G16 P+	+2 pour G16 Q+	+3 pour G16 R+
- 1 pour G16 P-	- 2 pour G16 Q-	- 3 pour G16 R-

[•RXH] Rang d'imbrication du sous programme courant.
1 : programme principal
2 : première imbrication
3 : deuxième imbrication, etc... (8 niveaux d'imbrications)

2.2.2.2 Adressage d'une liste de valeurs

[•IRxx(i)]	Adressage d'une liste de valeurs.

Le symbole [•IRxx(i)] permet d'adresser une liste de valeurs correspondant aux éléments spécifiés par xx.

L'index (i) définit le rang de l'élément dans la liste.

[•IRX(i)] Valeurs des cotes programmées sur les axes. Index i = 1 à 11.
i = 1 : valeur de X
i = 2 : valeur de Y
i = 3 : valeur de Z
i = 4 : valeur de U
i = 5 : valeur de V
i = 6 : valeur de W
i = 7 : valeur de A
i = 8 : valeur de B
i = 9 : valeur de C
i = 10 : en G21, l'index 10 adresse l'axe Y.
: en G22, l'index 10 adresse l'axe Z.
i = 11 : en G21, l'index 11 adresse l'axe X.
: en G22, l'index 11 adresse l'axe Y.

[•IRTX(i)] Valeurs des décalages programmés sur les axes.
Index i = 1 à 9, identiques aux valeurs des cotes programmées sur les axes (Voir [•IRX(i)]).

[•IRI(i)] Valeurs des arguments I, J et K. Index i = 1 à 5.
i = 1 : valeur de I
i = 2 : valeur de J
i = 3 : valeur de K
i = 4 : en G21, l'index 4 adresse la composante J.
: en G22, l'index 4 adresse la composante K.
i = 5 : en G21, l'index 5 adresse la composante I.
: en G22, l'index 5 adresse la composante J.

[•IRP(i)] Valeurs des arguments P, Q et R. Index i = 1 à 3.
i = 1 : valeur de P
i = 2 : valeur de Q
i = 3 : valeur de R

[•IRH(i)] Numéros de programmes ou sous programmes courants ou de niveaux d'imbrication inférieurs. Index i = 1 à n sous programme.
i = 1 : adresse le programme principal
i = 2 : adresse le sous programme appelé par le programme principal
i = 3 : adresse le sous programme suivant, etc... (8 niveaux d'imbrications)

[•IRDI(i)] Valeurs définissant l'origine des décalages angulaires programmés (G59 I.. J.. K..). Index i = 1 à 3.
i = 1 : valeur de I
i = 2 : valeur de J
i = 3 : valeur de K

2.3 Symboles d’accès aux données du bloc précédent

Les données accessibles du dernier bloc précédent sont identiques à celles du bloc courant (Voir 2.2). Leurs symboles sont les mêmes, mais précédés de deux points décimaux (au lieu d’un seul).

Ces symboles peuvent être :
- des symboles adressant des valeurs booléennes,
- des symboles adressant des valeurs numériques.

Ces symboles permettent de lire les données modales du bloc précédent.

Ces données sont celles du dernier bloc précédent exécutable (ou du dernier bloc peut-être déjà exécuté).

Cet adressage n’a d’intérêt que lorsque l’exécution du bloc courant est suspendue par la programmation de la fonction G999.

Syntaxe générale

Variable = [••symbole(i)]

Variable	Variable programme L, variable symbolique [symb], paramètre E.
[••symbole(i)]	Symbole entre crochets, précédé de deux points décimaux, suivi éventuellement d’un index (i).

3 Rangement dans les variables L900 à L951

Les outils de programmation décrits dans ce chapitre sont nécéssaires à la création de sous programme appelé par fonction G (Voir chapitre 5).

3.1 Généralités

Dans la programmation de cycles d'usinage, certains arguments ou fonctions peuvent avoir des significations différentes. Les symboles d'état [•IBE0(i)] et [•IBE1(i)] permettent de détecter leur présence dans les blocs comportant une fonction G d'appel de sous programme.

C'est au sous programme d'adresser correctement ces arguments ou fonctions selon leurs significations et de ranger leurs valeurs respectives dans les variables L900 à L951.

Les bits de [•IBE0(i)] et de [•IBE1(i)] (chacun d'entre eux valant 0 ou 1) sont accessibles en lecture par programmation paramétrée.

3.2 Rangement de F, S, T, H et N dans les variables L900 à L925

Rangement de F, S et T

Le symbole d'état [•IBE0(i)] constitué d'une liste de bits permet de détecter la programmation de F, S, T dans les blocs comportant une fonction Gxxx.

Index (i) : de 1 à 26 pour les adresses A à Z

Adressage du symbole d'état :
- le bit [•IBE0(6)] est mis à 1 sur programmation de F
- le bit [•IBE0(19)] est mis à 1 sur programmation de S
- le bit [•IBE0(20)] est mis à 1 sur programmation de T

Les valeurs de F, S et T sont rangées dans les variables (L900 à L925) suivantes :
- F dans L905,
- S dans L918,
- T dans L919.

Rangement de H et N

Le rangement de H et N n'est possible que lorsque H et N ne sont pas précédés des fonctions G75, G76, G77, ou G79 qui leurs sont liées en programmation ISO.

Un deuxième N (définissant la dernière séquence d'appel d'un sous programme N.. à N.. et qui suit le premier) est rangé dans la variable L914 (ce N ne peut en aucun cas être le numéro de bloc programmé en début de séquence).

Adressage du symbole d'état :
- le bit [•IBE0(8)] est mis à 1 sur programmation de H
- le bit [•IBE0(14)] est mis à 1 sur programmation d'un premier N
- le bit [•IBE0(15)] est mis à 1 sur programmation d'un second N

Les valeurs de H et N sont rangées dans les variables suivantes :
- H dans L907,
- premier N dans L913,
- second N dans L914.

3.3 Rangement de EA à EZ dans les variables L926 à L951

Les valeurs de EA à EZ à ranger dans les variables L926 à L951 sont définis par deux caractères alphabétiques :
- le premier est la lettre E,
- le second est une lettre comprise entre A et Z.

Le symbole d'état [•IBE1(i)] constitué d'une liste de 26 bits permet de détecter la présence des fonctions EA à EZ dans les blocs comportant une fonction Gxxx (i = index du rang alphabétique de la seconde lettre suivant E).

Adressage du symbole d'état :
- le bit [•IBE1(1)] adresse le bit correspondant à EA
- le bit [•IBE1(2)] adresse le bit correspondant à EB, et ainsi de suite jusqu'à EZ
- le bit [•IBE1(26)] adresse le bit correspondant à EZ

Les valeurs sont rangées dans les variables (L926 à L951) suivantes :
- EA dans L926,
- EB dans L927, et ainsi de suite jusqu'à EZ,
- EZ dans L951.

3.4 Adressage symbolique des variables L900 à L951

Les variables L900 à L925 et L926 à L951 peuvent être adressées respectivement par des symboles alphabétiques précédés du caractère « ' » (apostrophe), que ces variables soient situées dans le premier ou second terme d'une expression.

Variables L900 à L925

L900 à L925 peuvent être adressées par les symboles 'A à 'Z.

Par exemple :

'C = 'A + 'B est équivalent à L902 = L900 + L901

Variables L926 à L951

L926 à L951 peuvent être adressées par les symboles 'EA à 'EZ ('EA = L926, 'EB = L927, etc... jusqu'à 'EZ).

Par exemple :

'A = 'B - 'EA / 'EZ est équivalent à L900 = L901 - L926 / L951

4 Tableaux et gestion de variables symboliques

Les outils de programmation décrits dans ce chapitre sont utilisés pour la création de sous programme appelé par fonction G (Voir chapitre 5).

4.1 Création de tableaux de variables symboliques

Les règles d'écriture des variables symboliques utilisées en création de tableaux sont identiques à celles définies en programmation paramétrée (Voir chapitre 7 du manuel de programmation).

4.1.1 Définition d'un tableau

Un tableau est déclaré dans une variable symbolique entre les mots VAR et ENDV.

La définition d'un tableau est effectuée en insérant la ou les dimensions de ce tableau entre parenthèses après le dernier caractère de la variable symbolique.

Lorsqu'un tableau comprend plusieurs dimensions, les différentes dimensions sont séparées par l'écriture d'un des caractères virgule « , » ou point virgule « ; »

Syntaxe

```
VAR
[TABLn(a,b,c ...)]
ENDV
```

VAR	Déclaration de variables symboliques.
[TABLn(a,b,c...)]	TABLn : nom du tableau dans la pile. (a,b,c ...) : dimensions du tableau.
ENDV	Fin de déclaration de variables symboliques.

Exemple de définition de tableaux

```
VAR
[TABL1(10)] [TABL2(2,5,3)]
ENDV
```

Pour le tableau 2 [TABL2] ci-dessus, il est réservé :
2 x 5 x 3 = 30 éléments, soit 5 groupes de 2 puis 3 groupes de 10.

4.1.2 Dimensions des tableaux

Les tableaux peuvent comprendre de 1 à 4 dimensions.

Pour les tableaux à une dimension :
- la valeur d'une dimension doit être comprise entre 1 et 65535.

Pour les tableaux à 2, 3 ou 4 dimensions :
- la valeur d'une dimension doit être comprise entre 1 et 255.

La valeur d'une dimension doit être déclarée sous la forme d'une valeur immédiate ou d'une variable symbolique déjà initialisée.

Par exemple :

```
[VAR1] = 10
[VAR] = [TAB1 (VAR1,5)] est équivalent à : [TAB1 (10,5)]
```

Les variables symboliques ont des valeurs réelles.

Les index des tableaux sont des valeurs immédiates ou des variables symboliques.

Il est interdit de définir la valeur d'une dimension par :
- les variables programme L,
- les paramètres externes E.

Par exemple :

Si la variable symbolique [TAB(L0,3)] est programmée, on ne va pas chercher la variable programme L0 mais la variable symbolique [L0].

Les index de tableaux peuvent être constitués de sommes et/ou de différences de valeurs ou de variables symboliques.

Par exemple :

```
VAR   [IX] [COSX] [SINX] [NBT] = 4
[TABL(2,NBT)] = 0, 0, 10, 5, 20, 8, 30, -2
ENDV
FOR [IX] = 1 TO [NBT] -1 DO
  [COSX] = [TABL(1,IX+1)] - [TABL(1,IX)]
  [SINX] = [TABL(2,IX+1)] - [TABL(2,IX)]
  L0 = [COSX] * [COSX] L0 = [SINX] * [SINX] + L0
  [COSX] = [COSX] / RL0  [SINX] = [SINX] / RL0
  X [TABL(1,IX+1)] Y [TABL(2,IX+1)]
ENDF
```

Structure des tableaux à plusieurs dimensions

Les postes de la première dimension sont implantés en tête de ce tableau, puis ils sont multipliés par le nombre de postes de la seconde dimension. L'ensemble résultant est ensuite multiplié par le nombre de la dimension suivante et ainsi de suite.

4.1.3 Initialisation des variables et des tableaux

Les valeurs initialisées sont à 0 par défaut.

L'initialisation par des valeurs quelconques est effectuée en déclarant le caractère = suivi de la valeur ou des valeurs initiales séparées par le caractère virgule «,»

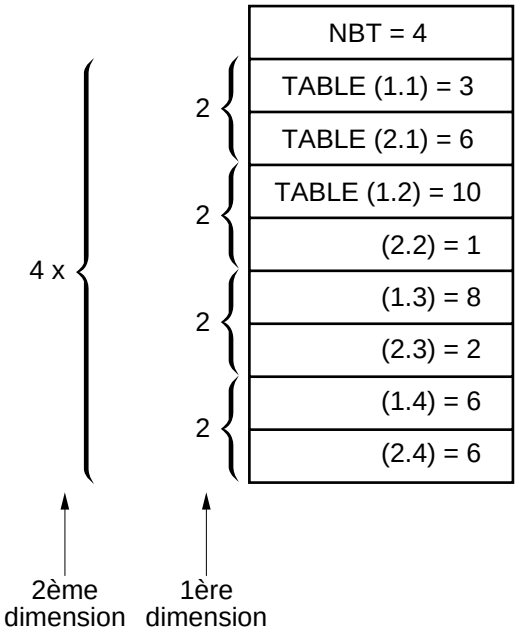
Les valeurs initiales peuvent être déclarées sur plusieurs blocs ; dans ce cas le caractère = sera répété devant la ou les valeurs définies dans le nouveau bloc.

Par exemple :

```
VAR [NTB] = 4 [TABLE(2,NTB)] = 3,6
                             = 10,1,8,2,6,6
ENDV

Si L0= [TABLE(2,3)]
   L0= 2 (6 chiffres)
```

Localisation dans la mémoire



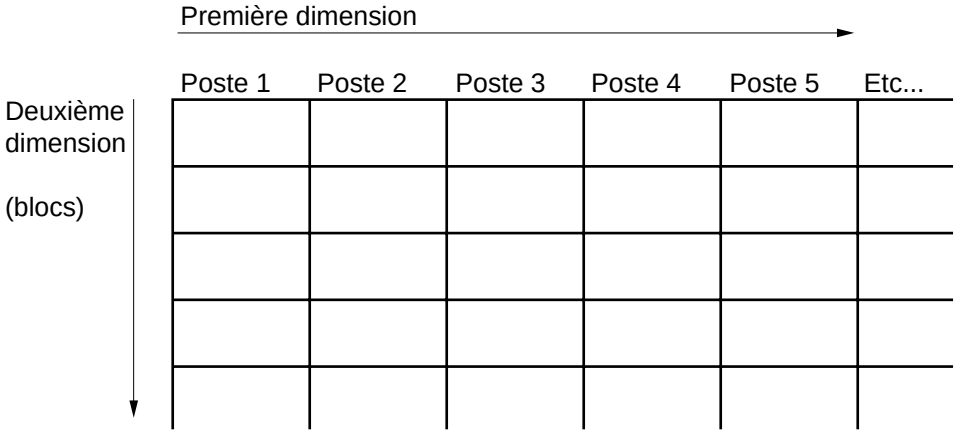
4.1.4 Création de tableaux pour rangement de profils

Le système offre la possibilité de ranger un profil écrit en ISO ou en PGP dans un tableau de la pile programme. La construction du tableau est effectuée au fur et à mesure de la lecture des blocs du profil.

Les blocs programmés sont rangés dans un tableau à deux dimensions :
- la première dimension est constituée de l'ensemble des fonctions à sauvegarder dans un bloc,
- la deuxième dimension correspond au nombre de blocs du profil.

La programmation paramétrée permet ensuite d'accéder aux données de ce tableau.

Par exemple :



4.1.5 Données susceptibles d'être rangées dans un tableau

Les données de programmation ISO suivantes sont susceptibles d'être rangées dans un tableau.

Fonctions G

Il n'est possible de ranger qu'une seule fonction G par bloc.

Lors d'un changement de plan d'interpolation dans un bloc, c'est la nouvelle fonction qui est rangée (G17, G18 ou G19 en fraisage. G20, G21, ou G22 en tournage); sinon c'est l'une des fonctions modales G00, G01, G02 ou G03 qui est rangée.

Valeurs des axes programmés

X, Y, Z, U, V, W, A, B, C (selon les axes déclarés dans le paramètre machine P0)

Ce sont les valeurs modales programmées avec les axes qui sont rangées.

Valeurs de I, J, K, P, Q, R

Le rangement des valeurs des fonctions n'est effectué que si celles-ci sont présentes dans le bloc, sinon c'est une valeur nulle qui est rangée dans leurs postes.

Vitesse d'avance F

C'est la valeur modale liée à la fonction F qui est rangée.

Vitesse de broche S

La valeur de S n'est rangée dans le tableau que si elle est présente dans le bloc, sinon c'est une valeur nulle qui est rangée.

Appel d'outil T

La fonction T n'est rangée dans le tableau que si elle est présente dans le bloc, sinon c'est une valeur nulle qui est rangée.

4.2 Commandes de gestion des variables symboliques

4.2.1 Rangement d'un profil quelconque

BUILD	Création d'un tableau pour rangement des trajectoires d'un profil.

La fonction BUILD permet le rangement du profil dans un tableau à deux dimensions :
- la première dimension est limitée à 16 postes.
- la deuxième dimension est limitée à 255 postes.

Syntaxe

BUILD [TAB(G / X / Y / I / J,NB)] H.. N..+n N..+n

BUILD	Création d'un tableau pour rangement d'un profil.
TAB	Nom du tableau dans la pile.
G / X / Y / I / J	Types de données dont les valeurs sont rangées dans les postes de la première dimension du tableau (16 postes maximum).
NB	Nom de la variable contenant le nombre de blocs de la deuxième dimension du tableau (255 blocs maximum).
H.. N.. N.. H.. N.. N.. N..+n N..+n H N..+n N..+n	Définition des bornes du profil.

Particularités

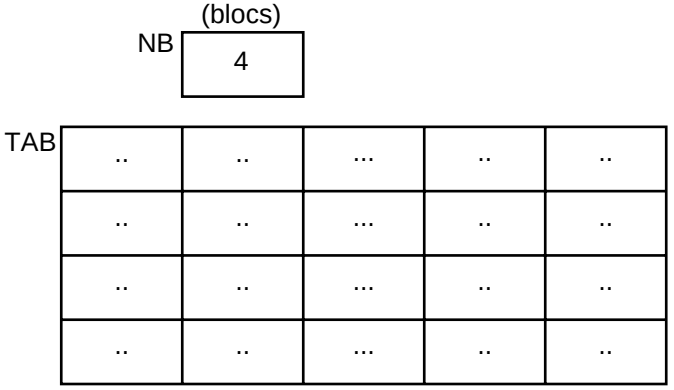
La fonction BUILD doit être le premier mot du bloc et le nom du tableau TAB le second, ils doivent être séparés par au moins un espace.

La programmation du nom du tableau TAB et de la variable NB crée automatiquement un tableau à deux dimensions.

Par exemple :

```
%555                                  %55
N10 ...                               G.. ...   Premier bloc du profil
N..                                   G.. ...
N..                                   G.. ...
BUILD [TAB(G/X/Y/I/J,NB)] H55         G.. ...   Dernier bloc du profil
N..
```

Tableau à deux dimensions créé par la programmation ci-dessus :
- première dimension : 5 postes,
- deuxième dimension : 4 postes (blocs).



Réservation de postes supplémentaires dans un champ de la fonction BUILD

Dans la première dimension du tableau, des postes supplémentaires peuvent être réservés s'ils sont non initialisés par des données de blocs; dans ce cas, les postes sont déclarés par des chiffres 0 séparés par le caractère / .

Par exemple :

```
%55
N10 ...
N..
N110                                  Premier bloc du profil
N..
N..
N220                                  Dernier bloc du profil
N..
BUILD [PROF1(G/X/Y/Z/0/0,NB)] N110 N220   La première dimension du tableau
                                          comprend 6 postes
```

Déclaration des postes par une liste de bits dans un champ de la fonction BUILD

Dans une variable symbolique certains des axes et des arguments suivants peuvent être déclarés sous la forme d'une liste de bits :
- axes X, Y , Z etc...
- arguments I, J, K,
- arguments P, Q, R.

La déclaration d'une liste de bits est réalisée en programmant dans un champ du BUILD l'une des adresses X, I ou P suivie d'un point décimal et du nom de la variable symbolique.

Par exemple :

```
BUILD [TAB1(G/I.Symb/R,NB)] H..
```
Déclaration de I.Symb

Le contenu de la variable symbolique [Symb] est une somme. Cette somme de valeurs est définie d'après les index des symboles d'adressage [••IBX(i)], [••IBI(i)] et [••IBP(i)] soit :
- 1 pour l'index i = 1
- 2 pour l'index i = 2
- 4 pour l'index i = 3
- 8 pour l'index i = 4 etc...

Donc la valeur de la variable comprenant I, J et K de [••IBI(i)] est égale à $2^{I-1} + 2^{J-1} + 2^{K-1}$.

Par exemple :

```
VAR [LIST] = 6
ENDV
BUILD [PROF(G/X.LIST/R,NB)] N.. N..
BUILD [PROF(G/Y/Z/R,NB)] N.. N..
```
est équivalent au bloc suivant :

4.2.2 Rangement d'un profil interpolé dans le plan

P.BUILD	Création d'un tableau pour rangement des cotes du plan d'interpolation d'un profil.

La fonction P.BUILD permet le rangement du profil dans un tableau à deux dimensions :
- la première dimension est limitée à 7 postes,
- la deuxième dimension est limitée à 255 postes.

Syntaxe

P.BUILD [TAB(7,NB)] H.. N..+n N..+n

BUILD	Création d'un tableau pour rangement d'un profil.
TAB	Nom du tableau dans la pile.
7	Nombre de postes dans la première dimension (7 postes maximum).
NB	Nom de la variable contenant le nombre de blocs (255 blocs maximum).
H.. N.. N.. H.. N.. N.. N..+n N..+n H.. N..+n N..+n	Définition des bornes du profil.

Particularités

La fonction P.BUILD doit être le premier mot du bloc et le nom du tableau TAB le second, ils doivent être séparés par au moins un espace.

Définition des 7 postes de la première dimension avec la fonction P.BUILD

- poste 1 : Type d'interpolation :
  valeur 0 pour une interpolation linéaire,
  valeur -1 pour une interpolation circulaire sens antitrigonométrique,
  valeur +1 pour une interpolation circulaire sens trigonométrique.
- poste 2 : Point à atteindre, valeur de la cote en abscisse.
- poste 3 : Point à atteindre, valeur de la cote en ordonnée.
  poste 4 : Position du centre :
  valeur en abscisse si interpolation circulaire, sinon valeur 0.
- poste 5 : Position du centre :
  valeur en ordonnée si interpolation circulaire , sinon valeur 0.
- poste 6 : Point de départ, valeur de la cote en abscisse.
- poste 7 : Point de départ, valeur de la cote en ordonnée.

Exemple

```
%70
N10 G17 X10 Y10
P.BUILD [TAB(7,NB] H80 N110 N130
N..
```

```
%80
N10 ...
N..
N..
N110 G01 X20
N120 G03 X20 Y50 I20 J30
N130 G01 X10 Y20
N.. ...
```

Le nom du tableau TAB et la variable NB créent le tableau à 7 postes suivant :

	(blocs)
NB	3

TAB							
	0	20	10	0	0	10	10
	+1	20	50	20	30	20	10
	0	10	20	0	0	20	50

4.2.3 Déport d'un profil ouvert et réactualisation du tableau

R.OFF	Déport normal d'un profil ouvert.

La fonction R.OFF permet le déport normal d'un profil initial créé dans un tableau par la fonction P.BUILD ou un tableau de même format soit [Pa(7,Nb)].

Après exécution de la fonction R.OFF, le profil déporté se situe dans ce même tableau et la variable spécifiant son nombre de blocs est réactualisée car des blocs intermédiaires ont pu être créés (voir figure 1).

Syntaxe

R.OFF [Pa(7,Nb)] / ±1 / R

R.OFF	Déport normal d'un profil.
Pa	Nom du tableau.
7	Nombre de postes du tableau.
Nb	Nom de la variable contenant le nombre de blocs du tableau Pa.
±1	Valeur +1 : déport à droite du profil, Valeur -1 : déport à gauche du profil.
R	Valeur du rayon exprimé dans la même unité que les cotes.

Particularités

La fonction R.OFF doit être le premier mot du bloc.

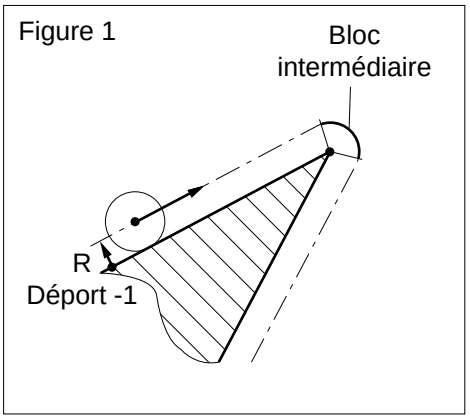
Le profil effectué avec la fonction R.OFF doit être un profil ouvert, c'est à dire que le point de départ du profil doit être différent de son point d'arrivée (voir figure 2).

La fonction R.OFF ne peut traiter que les profils récupérés dans P.BUILD qui ne comportent pas d'alternance de déport à gauche ou à droite pendant leur exécution.

Création d'un bloc intermédiaire par le système

Lorsque le profil comporte des trajectoires particulières le système peut être amené à créer un bloc de raccordement.

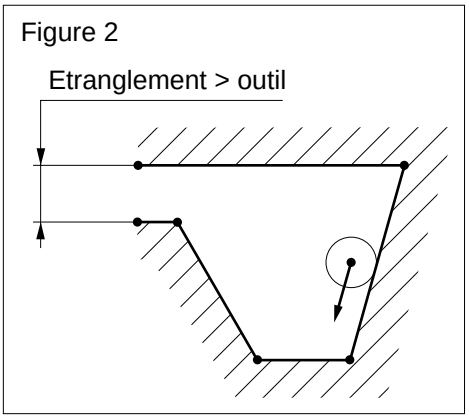
Figure 1



Profil ouvert

Lorsque le profil comporte un étranglement, celui-ci doit être de taille suffisante pour permettre le passage de l'outil, sinon le système considère le profil comme étant fermé.

Figure 2



Exemple

```
%92
N..
P.BUILD [P(7,NB)] N110 N200
...
...
R.OFF [PA(7,NB)] /-1 /10
```

Déport de 10 à gauche du profil

4.2.4 Redéfinition d'un profil selon l'angle de dépouille de l'outil

CUT	Elimination des gorges ou parties de gorges situées en deça de l'angle de dépouille de l'outil.

La fonction CUT s'applique aux gorges situées sur la trajectoire d'un profil plan créé dans un tableau par la fonction P.BUILD ou un tableau de même format soit [Pa(7,Nb)].

Après exécution de la fonction CUT le nouveau profil est situé dans ce même tableau et la variable spécifiant son nombre de blocs est réactualisée.

Syntaxe

CUT * [Pa(7,Nb)] / Angle

CUT	Elimination des gorges ou parties de gorges situées en deça de l'angle de dépouille de l'outil.
*	Lorsque le caractère * précède le nom du tableau, toutes les gorges situées en deça de l'angle de dépouille sont traitées. Lorsque le caractère * est absent devant le nom du tableau, seule la première gorge située en deça de l'angle de dépouille est traitée.
Pa	Nom du tableau.
7	Nombre de postes du tableau.
Nb	Nom de la variable contenant le nombre de blocs du tableau Pa.
Angle	Angle de dépouille exprimé en degrés.

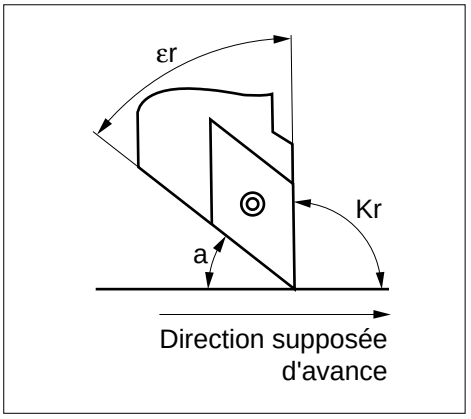
Particularités

La fonction CUT doit être le premier mot du bloc (pas de numéro de séquence).

Rappel sur les angles d'un outil de coupe défini dans le plan ZX

Angles caractéristiques :

- Kr : angle d'arête de coupe principale.
- εr : angle de pointe de l'outil.
- a : angle de dépouille ou d'arête secondaire.



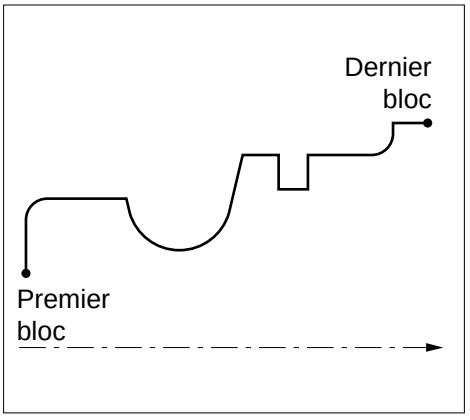
Traitement du tableau

L'analyse du tableau commence sur le premier bloc et se termine :
- sur le dernier bloc avec CUT * ...,
- dès la première coupure avec CUT ...

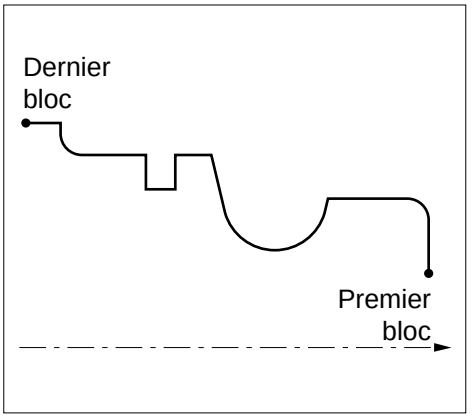
Dans le tableau, le profil doit être obligatoirement défini de telle sorte que sur l'axe des abscisses la cote de départ du profil (premier bloc) soit inférieure à celle de fin du profil (dernier bloc).

Par exemple :

Bonne définition du profil



Mauvaise définition du profil



Lorsque l'angle de dépouille est négatif ou nul (compris entre 0° et -180°), les zones du profil situées au dessous de cet angle sont éliminées.

Lorsque l'angle de dépouille est positif (compris entre 0° et +180°), les zones du profil situées au dessus de cet angle sont éliminées.

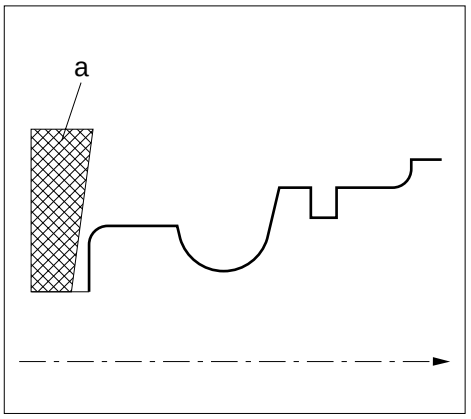
Exemples

«a» désigne les zones éliminées.

Exemple 1 :

Elimination de la première gorge située sur le profil (pas de * devant la variable)

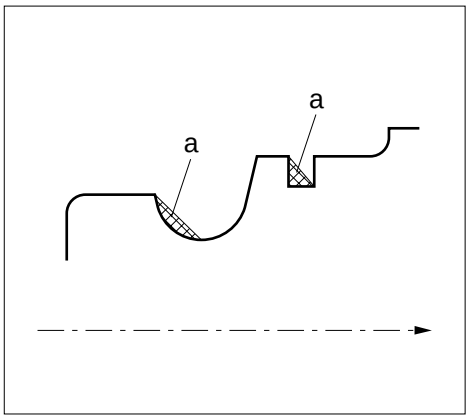
```
CUT [PA(7,NB)] / -95
```



Exemple 2 :

Elimination des gorges ou partie de gorges situées sur le profil (* devant la variable)

```
CUT * [PA(7,NB)] / -50
```



4.2.5 Validation ou invalidation de fonctions M et/ou d'axes. Positionnement ou suppression de bits

BSET	Validation de la programmation de fonctions M et/ou d'un ou plusieurs axes. Positionnement des bits de [•IBE0(i)] et [•IBE1(i)]

BCLR	Invalidation de la programmation de fonctions M et/ou d'un ou plusieurs axes. Suppression des bits de [•IBE0(i)] et [•IBE1(i)].

Syntaxe

BSET [•BMxx] / [•IBX(i)] / [•IBE0(i)] / [•IBE1(i)]

BSET	Validation de la programmation de fonctions M et/ou d'un ou plusieurs axes.
[•BMxx] / [•IBX(i)]	Lorsque le système se trouve dans l'état G999, la validation par BSET effectue la mise à 1 des bits de [•BMxx] et/ou [•IBX(i)].
[•IBE0(i)] / [•IBE1(i)]	Lorsqu'un sous programme est appelé par fonction Gxx, BSET permet également de positionner des bits de [•IBE0(i)] et [•IBE1(i)].

Syntaxe

BCLR [•BMxx] / [•IBX(i)] / [•IBE0(i)] / [•IBE1(i)]

BCLR	Invalidation de la programmation de fonctions M et/ou d'un ou plusieurs axes.
[•BMxx] / [•IBX(i)]	Lorsque le système se trouve dans l'état G999, l'invalidation par BCLR effectue la mise à 0 des bits de [•BMxx] et/ou [•IBX(i)].
[•IBE0(i)] / [•IBE1(i)]	Lorsqu'un sous programme est appelé par fonction Gxx, BCLR permet également de supprimer des bits de [•IBE0(i)] et [•IBE1(i)].

Particularités

Les fonctions BSET ou BCLR :
- doivent être les premiers mots du bloc (pas de numéro de séquence),
- doivent être séparées de la liste des symboles par au moins un espace, par contre aucun espace ne doit s'insérer dans la liste des symboles.
- sont suivies de la liste des symboles à valider ou invalider ; les différents symboles sont séparés par le caractère « / » .

Exemple

Dans le bloc N120 seuls les déplacements X10 et Z10 sont effectués. Les déplacements sur les axes Y et B ainsi que la sortie de la fonction M05 sont invalidés (Voir chapitre 2 pour les index (2) et (8) correspondant respectivement à Y et B).

```
N..
N..
N.. G999 X10 Y10 Z10 B30 M05
BCLR [•IBX(2)] / [•IBX(8)] / [•BM05]        Invalidation
N120 G997
```

4.2.6 Recherche de variables symboliques dans la pile

SEARCH	Recherche de la présence ou non d'une variable symbolique dans la pile.

Syntaxe

SEARCH [Symb] N..

SEARCH	Recherche de la présence ou non d'une variable symbolique dans la pile.
[Symb]	Nom de la variable symbolique. Lorsque la variable recherchée est présente l'analyse du bloc se poursuit.
N..	Numéro du bloc auquel doit être effectué un saut lorsque la variable symbolique est absente.

Exemple

```
%30
N..
VAR [Symb]
ENDV
N..

%35
N..
N90 ...
SEARCH [Symb] N100
N..
N100
N..
```

4.2.7 Mise à disposition d'une liste de variables symboliques

SAVE	Mise à disposition du programme principal et des sous programmes d'une liste de variables symboliques déclarées dans un sous programme quelconque.

Syntaxe

SAVE [Symb1] / [Symb2] ...

SAVE	Mise à disposition du programme principal et de tous sous programmes d'une liste de variables symboliques déclarées dans un sous programme quelconque.
[Symb1] / [Symb2] ...	Liste des variables symboliques.

Particularités

Les variables symboliques à mettre à disposition peuvent être déclarées à l'intérieur de sous programmes d'imbrications quelconques.

Les variables déclarées après SAVE doivent être séparées par le caractère « / ».

Exemple

Après retour du sous programme %30, le programme %10 et le sous programme %20 ainsi que les nouveaux sous programmes pourront utiliser les variables symboliques [V1] et [TB(4)].

```
%10
N..
N.. G77 H20
N..

%20
N..
N.. G77 H30
N..

%30
N..
VAR [V1] / [TB(4)]
ENDV
N..
SAVE [V1] / [TB(4)]
N..
```

4.2.8 Copie des blocs ou postes d'un tableau dans un autre tableau

MOVE	Copie de la totalité ou partie d'un tableau dans un autre tableau.

La fonction MOVE permet la copie des tableaux ayant les formats suivants :
- [P(m)] : m blocs de un élément ,
- [P(n,m)] : m blocs de n éléments.

Syntaxe générale

MOVE [Pj(nj,mj)],mj1,mj2 = [Pi(ni,mi)],mi1,mi2 / j1=i1 / j2=i2 /jn=in

La fonction MOVE comporte plusieurs possibilités de copie :
- copie simple des blocs,
- copie partielle des blocs,
- spécification des postes à copier.

Syntaxe copie simple des blocs

MOVE [Pj(nj,mj)] = [Pi(ni,mi)]

MOVE	Copie du contenu d'un tableau dans un autre tableau. Lors d'une copie simple, les deux tableaux doivent avoir des formats identiques, soit : nj = ni et mj = mi
Pj	Nom du tableau destination.
nj,mj	Postes et blocs dans le tableau destination.
Pi	Nom du tableau source.
ni,mi	Postes et blocs dans le tableau source.

Syntaxe copie partielle des blocs

MOVE [Pj(nj,mj)],mj1,mj2 = [Pi(ni,mi)],mi1,mi2

MOVE	Copie du contenu d'un tableau dans un autre tableau.
Pj	Nom du tableau destination.
nj,mj	Postes et blocs dans le tableau destination.
mj1,mj2	Bornes du tableau destination Pj entre lesquelles sont copiés les blocs indexés mi1 à mi2 du tableau source Pi. Les autres blocs de Pj ne sont pas modifiés.
Pi	Nom du tableau source.
ni,mi	Postes et blocs dans le tableau source.
mi1,mi2	Bornes indexées du tableau source Pi. Ces bornes ainsi que les blocs situés entre ces 2 bornes sont copiés dans le tableau Pj, entre les bornes mj1 et mj2.

Syntaxe spécification des postes à copier

MOVE [Pj(nj,mj)] = [Pi(ni,mi)] / j1=i1 / j2=i2 / jn=in

MOVE	Copie du contenu d'un tableau dans un autre tableau.
Pj	Nom du tableau destination.
nj,mj	Postes et blocs dans le tableau destination.
Pi	Nom du tableau source.
ni,mi	Postes et blocs dans le tableau source.
/ j1=i1 / j2=i2 / jn=in	Lorsque certains éléments d'un tableau ne doivent pas être copiés, il faut spécifier pour chaque poste à copier et après le caractère / , l'index du poste destination puis l'index du poste source séparés du caractère « = ». La valeur d'un poste destination copié peut être inversée si l'index du poste source est précédé du signe « - » (moins).

Particularités

La fonction MOVE doit être le premier mot du bloc (pas de numéro de séquence).

Le nombre de blocs d'un profil fini est limité à 95.

Il est possible d'inverser l'ordre d'une copie en inversant les index des bornes de début et de fin dans l'un des tableaux.

Les index peuvent être spécifiés dans des variables symboliques.

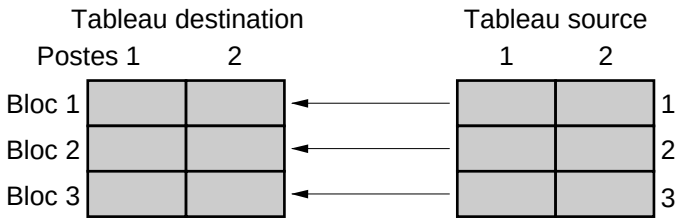
En cas d'erreur de progammation le système émet les erreurs suivantes :
ERREUR N°196 (incohérence dans la déclaration des index),
ERREUR N°199 (syntaxe incorrecte).

Exemples

MOVE copie simple des blocs.

Exemple : Copie du contenu d'un tableau dans un autre tableau.

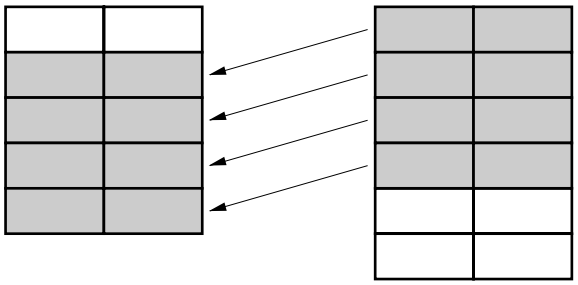
```
MOVE [PB(2,3)] = [PA(2,3)]
```



MOVE copie partielle des blocs.

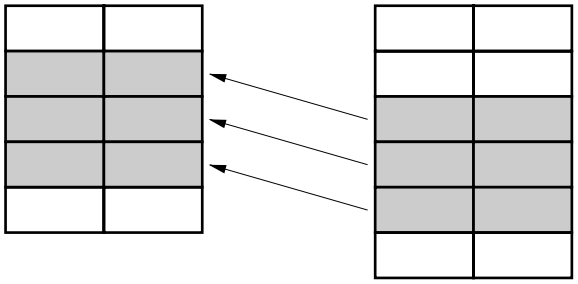
Exemple 1 : Copie d'une partie de tableau dans un autre tableau.

```
MOVE [PB(2,5)],2,5 = [PA(2,6)],1,4
```



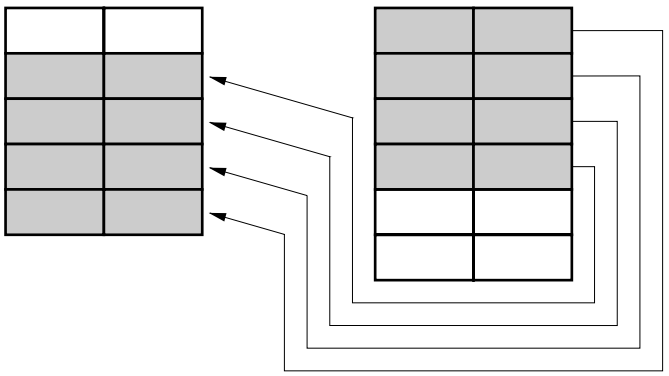
Exemple 2 : Copie d'une partie de tableau dans un autre tableau.

```
MOVE [PB(2,5)],2,4 = [PA(2,6)],3,5
```



Exemple 3 : Inversion des index des bornes et des blocs lors de la copie d'une partie de tableau dans un autre tableau.

```
MOVE [PB(2,5)],2,5 = [PA(2,6)],4,1
```

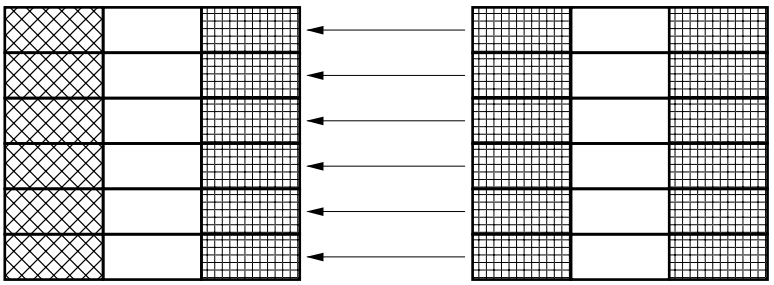


MOVE copie partielle des blocs et spécification des postes à copier

Exemple : Inversion des valeurs des postes lors de la copie d'une partie de tableau dans un autre tableau.

Dans le tableau destination PB, seuls les premiers et troisièmes postes sont copiés et les valeurs des premiers postes sont inversées.

```
MOVE [PB(3,6)] = [PA(3,6)] / 1=-1 / 3=3
```



Rappel

Fonction DELETE

La fonction DELETE (ou DELE) peut être utilisée pour la destruction programmée des variables symboliques (Voir chapitre 7 du manuel de programmation).

4.2.9 Adressage indirect de variables symboliques

Une variable ou un tableau de variables symboliques peuvent être référencés par une valeur.

Cet adressage permet la simplification de l'enchaînement de tableaux par utilisation de numéros à la place de noms de variables symboliques.

Cet adressage de variables symboliques est indirect, car effectué au travers d'une autre variable vecteur adresse dont la valeur est la référence d'une autre variable symbolique ou d'un tableau de variables symboliques.

L'adressage numérique est symbolisé par le caractère @ suivi du nom du vecteur adresse.

Les variables adressées par des numéros négatifs sont détruites automatiquement par la fonction G80.

Exemple

```
VAR  [no] = 7  [@no(10)]
  [Symb]
ENDV
[Symb] = 7     L0 = [@Symb(2)]
```

Le tableau à 10 postes est référencé par la valeur 7

«@Symb» adresse le même tableau déclaré sous «@no»

4.2.10 Exemples de programmation

Exemple 1

Utilisation de BUILD en fraisage (plan XY). En correction de rayon, trajectoires de 1 à 6 puis retour de 6 à 1.

Représentation de l'usinage

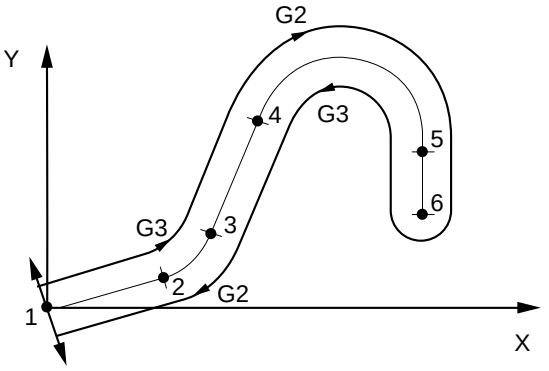


Tableau construit par le programme

Points

	G	X	Y	I	J
1	1	0	0	0	0
2	2	22.268	8.104	0	0
3	3	28.243	14.081	18.846	17.501
4	1	35.905	35.13	0	0
5	2	65	30	50	30
6	1	65	20	0	0

[NB] -1

G[TAB(1,I)]

X[TAB(2,I)]

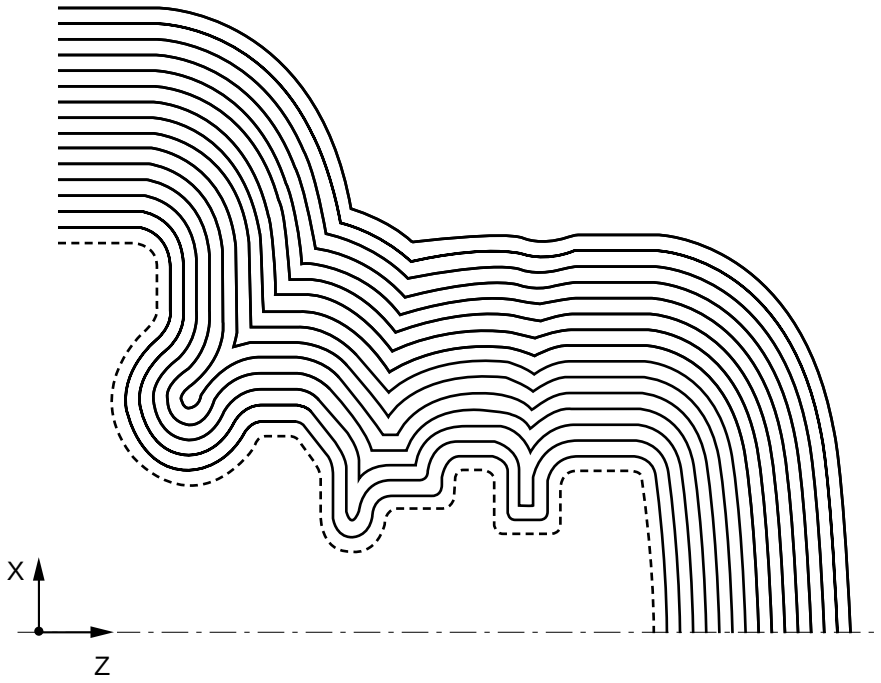
Valeur du tableau à l'exécution de BUILD

```
%350
Z0
M999 G79 N100
N10 G1 X Y
EA20 ES EB10
EA70                    } Définition (point 1 à 6)
G2 I50 J30 R15
N40 G1 Y20
N100
BUILD [TAB(G/X/Y/I/J,NB)]N10 N40
VAR [I]
ENDV
FOR [I]=1 T0 [NB] D0 G41 D1                Trajectoire de 1 à  6
   G[TAB(1,I)] X[TAB(2,I)] Y[TAB(3,I)] I[TAB(4,I)] J[TAB(5,I)]
ENDF
FOR [I]=[NB] -1 DOWNT0 1 D0                Retour de 6 à  1
   L0=[TAB(1,I+1)]
   IF L0>1 THEN L0=L0*3+1&3                Inversion des G2 et G3 pour le retour
   ENDI
   GLO X[TAB(2,I)] Y[TAB(3,I)] I[TAB(4,I+1)] J[TAB(5,I+1)]
ENDF
G40 X Y
M2
```

Exemple 2

Utilisation de P.BUILD en tournage. Exécution d'un profil avec déport et emploi des fonctions MOVE et R.OFF

Représentation de l'usinage



```
%46
P.BUILD [C(7,N)] N100 N110                  Création d'un tableau pour rangement
                                            du profil
VAR [R][G(3)]=2,1,3 [I] [V]
ENDV
...
FOR [R]= 30 DOWNTO 2 BY 2 DO
  VAR [M]=[N]-1 [P(7,M)]
  ENDV
  MOVE [P(7,M)] = [C(7,N)],2,[N]            Copie d'un tableau dans un autre
                                            tableau
  R.OFF [P((7,M)] / + 1 / [R]               Déport du profil à droite
  G X60 Z110
  X[P(7,1)] Z[P(6,1)]
  FOR [I] = 1 TO [M] DO [V] = [P(1,I)]+2
  G[G(V)] X[P(3,I)] Z[P(2,I)] I[P(5,I)] K[P(4,I)]
  ENDF
  DELE [P]/[M]
ENDF
M02
N100 X0 Z100                                \
G3 I0 K20 ES-                               |
G1 EA180 X20 Z85 EB2                        |
X12                                         |
Z75                                         |
X20                                         |
Z70                                         |
X15                                         |  Définition du profil
Z60                                         |
G2 X15 Z50 I15 K55                          |
G1 X25 EB-4                                 |
Z40                                         |
G2 I30 K30 ES+                              |
G1 EA90 X50 Z25 EB3                         |
N110 Z10                                    /
```

5 Création de sous programmes appelés par fonctions G

5.1 Appel de sous programme par fonctions G

Gxxx	Appel de sous programme par fonction G.

La fonction Gxxx permet l'appel d'un sous programme et l'exécution de celui-ci.

La fonction Gxxx est utilisée pour l'exécution de cycles d'usinage. Les cycles ainsi créés sont éventuellement personnalisables. Cette fonctionnalité permet aussi la création de fonctions spécifiques.

Syntaxe

N.. **Gxxx** Paramètres spécifiques du cycle d'usinage

Gxxx	La fonction force l'appel du sous programme %10xxx de numéro correspondant au cycle d'usinage. Par exemple : - G81 appelle le sous programme %10081 - G199 appelle le sous programme %10199
Paramètres	Les paramètres (ou arguments) spécifiques au cycle d'usinage sont obligatoires derrière Gxxx.

Propriétés des fonctions

Les fonctions Gxxx appelant des sous programmes sont modales.

Une fonction Gxxx est non modale lorsque la fonction de révocation est intégrée au sous programme appelant le cycle.

Révocation

Les fonctions modales Gxxx sont révoquées par la fonction G80 (cette fonction n'appelle pas de sous programme).

Particularités

⚠ ATTENTION

Les fonctions G200 à G255 peuvent éventuellement servir à des applications NUM, il est par conséquent conseillé de n'utiliser que les fonctions de G100 à G199.

Liste des fonctions G forçant l'appel d'un sous programme :
- G06, G31, G33, G38, G45, G46, G48, G49, G63, G64, G65, G66, G81 à G89,
- G100 à G255 (rappel : G200 à G255 réservées à NUM).

Un appel de cycle par fonction Gxxx non accompagné d'arguments est ignoré par le système. Ces arguments sont définis dans le sous programme %10xxx appelé.

Les sous programmes appelés par fonctions G doivent avoir la visibilité du contexte programme et de toutes les fonctions programmées dans le bloc d'appel.

Le déroulement d'un sous programme appelé par fonction G ne peut être interrompu par une demande "immédiate" du mode modification (MODIF).

Un sous programme appelé par fonction G ne peut appeler lui même un autre sous programme par fonction G. Par contre, l'imbrication avec un autre type d'appel est possible (par fonction M ou processeur machine), mais dans tous les cas deux appels du même type ne peuvent s'imbriquer.

Fonctionnalités utilisées dans les sous programmes %10xxx

- Fonctions G997, G998 et G999,
- Variables programme L900 à L925 et L926 à L951,
- Paramètres externes E,
- Variables symboliques,
- Symboles d'accès à l'état programme,

L'appel de sous programme par fonction G positionne de manière implicite la fonction G999 (suspension de l'exécution et forçage de la concaténation des blocs); cette fonction devra être révoquée par programmation des fonctions G998 et/ou G997 positionnées dans le sous programme.

Lors du retour au programme pièce, l'état G999 est systématiquement repositionné tant que la fonction d'appel de sous programme (Gxxx) reste présente et active (pas de G80).

Pour informations complémentaires sur les fonctions G997, G998 et G999 se référer aux manuels de programmation :
- Fraisage (938819),
- Tournage (938820).

5.2 Non visualisation des sous programmes en cours d'exécution

Un sous programme et ses autres sous programmes internes en cours d'exécution peuvent être non visualisés en page programme (PROG).

Le caractère " : " placé derrière le numéro du sous programme définit la non visualisation et seul le bloc d'appel du sous programme est visualisé.

Par exemple :

Seul le bloc N150 comprenant la fonction G108 est visualisé durant l'exécution des sous programmes %10108 et %118.

```
%10                 %10108:             %118
N10                 N10                 N10
N..                 N..                 N..
N150 G108 ...       N80 G77 H118        N..
N..                 N..                 N..
```

5.3 Exemples de programmation

Exemple 1

Création d'un cycle particulier avec la fonction G199 (sous programme %10199).

Le cycle ci-dessous n'est donné qu'à titre d'exemple de création.

Le cycle permet l'exécution de plusieurs perçages ou pointages "P" répartis sur un cercle de rayon "R" et centré suivant X Y (G17).

Syntaxe et paramètres du cycle

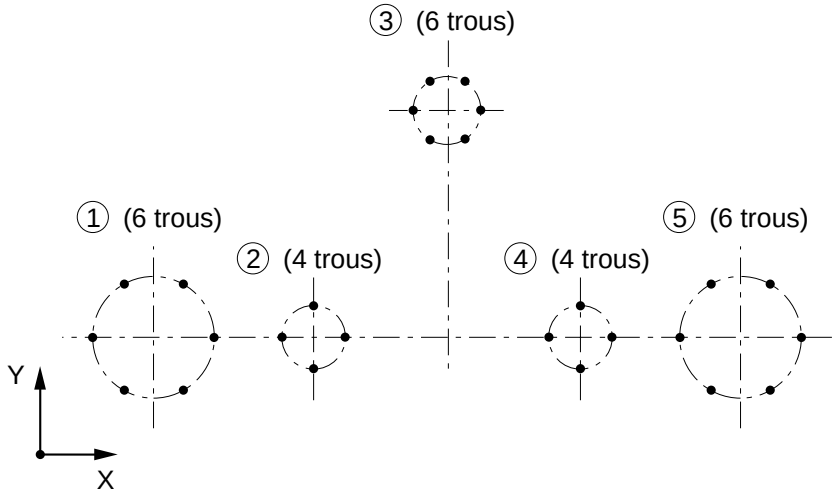
N.. G199 X.. Y.. ER.. Z.. P.. R.. F..

G199	Cycle de perçages équidistants sur cercle.
X.. Y..	Position du centre du cercle.
ER..	Position d'approche et de dégagement.
Z..	point à atteindre en usinage.
P..	Nombre de trous équidistants.
R..	Rayon du cercle de positionnement des perçages.
F..	Vitesse d'avance d'usinage

Programme principal d'usinage

```
%20
N10 G0 G52 Z0
N20 T1 D1 M06 (FORET)
N30 S1000 M40 M03
N40 X0 Y0 Z5
N50 G199 X50 Y50 ER2 Z-10 P6 R20 F90      Cycle cercle 1
N60 X100 Z-5 P4 R10                       Cycle cercle 2
N70 X150 Y150 Z-15 P6 R25                 Cycle cercle 3
N80 X200 Y50 Z-5 P4 R10                   Cycle cercle 4
N90 X250 Z-10 P6 R20                      Cycle cercle 5
N100 G80 G0 G52 M05                       Annulation cycle
N110 M02
```

Représentation de l'usinage



Sous programme du cycle

```
%10199: (Perçages équidistants sur cercle)
VAR
[G0/1] [RETOUR] [FEED] [G94/5]
ENDV

[G0/1]=3 * [..BG03] [G0/1]=2*[..BG02] + [G0/1]Mémo G0, G1, G2 ou G3
[G0/1]=1 * [..BG01] + [G0/1]
[FEED]=[.RF]                                  Mémo G94 ou G95
[G94/5]= 94 * [.BG94]
[G94/5]= 95 * [.BG95] + [G94/5]
PUSH L0 - L7

(Test si P et R programmés dans le bloc d'appel)
IF [..G80]= 1 THEN                            Premier bloc du cycle ?
L0= [.IBP(1)] * [.IBP(3)]
G79 L0= 0 N100                                Erreur si P ou R absents
ENDI
IF [.IBP(1)] = 1 THEN                         Prise en compte d'un nouveau P éventuel
L100= [.IRP(1)]
ENDI
  L100= [.IRP(1)]                             Mémo valeur P
  G79 L100 < 1 N101                           Erreur si P n'est pas un nombre entier
                                              positif
  IF [.IBX(3)] = 1 THEN L925 = [.IRX(3)]      Cote fond de perçage
ENDI
```

```
[RETOUR]= 'ER                          Cote de retour
BCLR [.IBX(3)]                         Invalidation axe Z

L0= [.IRX(1)]                          Cote position en X
L1= [.IRX(2)]                          Cote position en Y
L2= L0 + [.IRP(3)]                     Cote départ cycle
XL2                                    Modification cote fond de perçage

G997                                   Validation déplacements XY
FOR L4= 1 TO L100
  L5= L4 * 360 / L100                  Angle actuel
  L6= CL5 * [.IRP(3)] + L0             Cote X
  L7= SL5 * [.IRP(3)] + L1             Cote Y
  G3 G94 F5000 XL6 YL7 IL0 JL1         Positionnement circulaire
  M997                                 Forçage concaténation
  IF [.IBE0(6)]= 1 THEN                Vitesse d'avance
  G94 FL905
  ENDI
  G1 Z L925                            Exécution perçage
  G4 F1                                Temporisation fond de perçage
  G0 Z [RETOUR]                        Retour en Z
  M999
ENDF                                   Fin de cycle

G [G94/5] F [FEED]                     Retour aux conditions initiales
PULL L0 - L7
G79 N9999                              Fin du cycle

N100 E.500                             Numéro d'erreur (Voir %20500)
N101 E.501                             Numéro d'erreur (Voir %20500)
N9999
```

Programme de messages d'erreurs

```
%20500 (Messages d'erreurs du cycle G199)
N500 $ P ET R OBLIGATOIRES EN G199
N501 $ P DOIT ETRE UN ENTIER POSITIF EN G199
```

Exemple 2

Création d'un cycle particulier avec la fonction G177 (sous programme %10177).

Le cycle ci-dessous n'est donné qu'à titre d'exemple de création.

Le cycle permet l'exécution d'un profil par passes aller-retour avec possibilité d'utiliser ou non la correction de rayon.

Syntaxe et paramètres du cycle

G177 N.. N.. ER..	
G177	Cycle d'usinage par passes aller-retour.
N.. N..	Numéro du premier et du dernier bloc définissant le profil (lorsque les blocs sont inversés l'usinage du profil est inversé).
ER..	Argument forçant ou annulant la correction de rayon : ER 40 : usinage centre outil ER 41 : correction à gauche du profil ER 42 : correction à droite du profil

Programme principal d'usinage

```
%77
N10 G0 G52 Z0
N20 T1 D1 M06 (OUTIL R5)
N30 S2000 M40 M03
N40 G0 X-20 Y0
N50 G92 R1
N60 Z-10
N70 G79 N160
N80 G1 X-20 Y0
N90 EA20 ES
N100 EA50
N110 G2 I60 J25 X75 Y25
N120 G1 ET
N130 G2 I95 J10
N140 G3 I120 J15 X120 Y5
N150 G1 G40 X140
N160 G177 N80 N150 ER42
N170 G59 Y20
```

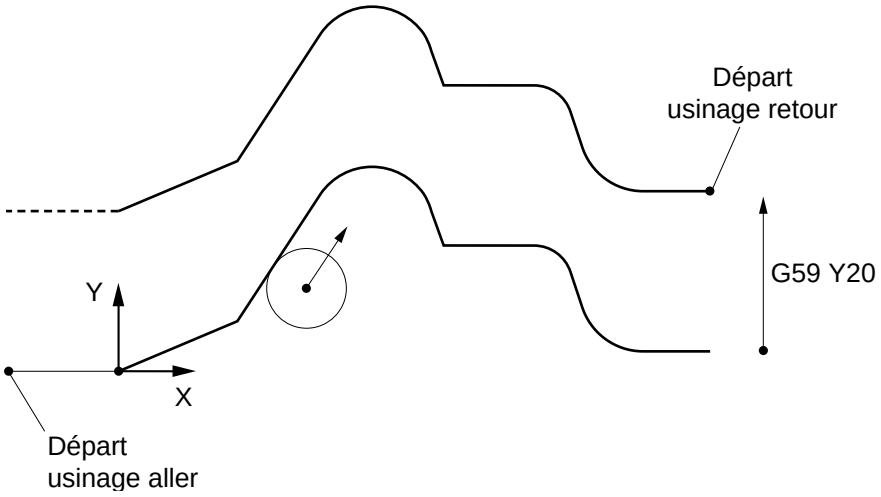
Définition du profil

Cycle aller

Décalage d'origine

```
N180 G177 N150 N80 ER41                    Cycle retour
N190 G0 G52 Z0
N200 G0 X-100 M5
N210 M02
```

Représentation de l'usinage



Sous programme du cycle

```
%10177: (Profil par passes aller-retour)
G998
VAR [N1] [N2] [PLANF] [G] [H] [diam] [NBLOC] [M998] [multi]=1
  [axe1] [axe2] [PLANT] [centre1] [centre2]
ENDV
IF 'ER<40 OR 'ER>42 THEN 'ER=40            Si ER mal programmé alors ER40
ENDI
[M998]=[.BM999]-[.BM997]+998 M998          Mémorisation M997, M998 et M999
[N1]=L913 [N2]=L914
IF L913>L914 THEN [N1]=L914 [N2]=L913      Mémorisation du sens d'exécution profil
ENDI
[H]=[.RXH]-1 [H]=[.IRH(H)]                 Rang d'imbrication
IF L913>L914 THEN
P.BUILD [TAB(7,NB)] H[H] N[N1] N[N2]       Création d'un tableau pour rangement
                                           du profil
G997
[PLANT]=[.BG20]+[.BG21]                    Choix du plan si tournage
[diam]=E70007                              Programmation au diamètre
```

```
IF [PLANT]<>0 THEN
    IF [.BG20]=1 THEN @Y=X @X=Z @J=I @I=K
    ELSE @Y=Y @X=X @J=J @I=I
    ENDI
  IF [diam]=1 THEN [multi]=2                    Si tournage au diamètre (x2)
  ENDI
ELSE
E11005=0
  IF [.BG17]=1 THEN @Y=Y @X=X @J=J @I=I
  ELSE
    IF [.BG18]=1 THEN @Y=X @X=Z @J=I @I=K       Equivalence d'adresses en fraisage
    ELSE
    @Y=Z @X=Y @J=K @I=J
    ENDI
  ENDI
ENDI
[axe1]=[TAB(3,NB)]*[multi][axe2][TAB(2,NB)]     Point de départ égal au point
                                                d'arrivée du profil
G1 GL943 @Y [axe1] @X [axe2]
G998
FOR [NBLOC]=[NB] DOWNTO 2 DO
[G]=[TAB(1,NBLOC)]
   IF [G]=0 THEN [G]=1
   ELSE
     IF [G]=-1 THEN [G]=3                       Boucle tant que le profil
     ELSE [G]=2                                 n'est pas terminé
     ENDI
   ENDI
[axe1]=[TAB(7,NBLOC)]*[multi][axe2]=[TAB(6,NBLOC)]
[centre1]=[TAB(5,NBLOC)]*[multi][centre2]=[TAB(4,NBLOC)]
G[G] @Y[axe1] @X[axe2] @J[centre1] @I[centre2]
ENDF
ELSE
G1 GL943 G77 H[H] N[N1] N[N2]                   Exécution du profil aller
ENDI
N9900 G80
E11005=[diam]                                   Restitution de la programmation au
                                                diamètre ou rayon définie en entrée
```

Exemple 3

Cycle de perçage débourrage créé par la société NUM et appelé par la fonction G83

Le cycle %10083 appelle le sous programme %10080 permettant l'analyse de tous les cycles créés par NUM (Voir sous programme %10080 à la suite du sous programme %10083).

Rappel de la syntaxe du cycle G83 en fraisage

N.. G83 X.. Y.. Z.. ER.. P.. Q.. F..

```
%10083:
(cycle debourrage)
VAR [M3/4][M998][G90/1][G0/1][RF][garde]=1 [diam]
   [IX][IY][IZ][LZ][I][cote][prof][Gplan][E]
ENDV
[diam]=E11005 E11005=0
G77 H10080(appeler module analyse)
(verification syntaxe: presence P si bloc precedent G80)
IF [..BG80]=1 AND [.IBP(1)]=0
  THEN E.889
ENDI
(chargement P et Q si programmes)
IF [.IBP(1)]=1 THEN 'P=[.IRP(1)]
   IF [.IBP(2)]=0 THEN 'Q='P
   ENDI
ENDI
IF [.IBP(2)]=1 THEN 'Q=[.IRP(2)]
ENDI
(conversion garde si INCH)
IF [.BG70]=1 THEN [garde]=[garde]/25.4
ENDI
(sens garde selon orientation outil)
IF [.RDX]<0 THEN [garde]=-[garde]
ENDI
(preparer le positionnement des axes)
IF [.BG95]=1 THEN G0
ELSE F5000
ENDI
IF [I]<>0 THEN G9 G998
   [cote] = 'ER [I]=[IZ]+10 G0 G77 H10080 N[I] N[I]
(donner signe correct a P et Q si programmes)
   IF 'P < 0 THEN 'P = -'P
   ENDI
   IF 'Q < 0 THEN 'Q = -'Q
```

```
   ENDI
   IF 'P > 0 THEN 'I=0  'L='ER-L[LZ]
(erreur si plan de remontee = fond de trou)
    IF 'L = 0 THEN E.891
    ENDI
      IF 'L < 0 THEN 'L=-'L
      ENDI
      IF 'Q = 0 OR 'Q > 'P THEN 'Q = 'P
      ENDI
      (calcul profondeur 1ere passe)
      'I = 'Q-'P * 'I/'L + 'P + 'I
      'J = 'L-'I
      IF 'J <= 0 THEN
      (descente fond de trou)
      [cote]=L[LZ] G1 F[RF] G77 H10080 N[I] N[I]
      G79N100
      ENDI
      IF 'J < 'Q/2 THEN  'I = -'Q/2 + 'L
      ENDI
      IF 'ER > L[LZ]  THEN  [prof] = -'I + 'ER
      ELSE    [prof] = 'I + 'ER
      ENDI
      (execution 1ere passe)
      [cote]=[prof] G9 G1 F[RF] G77 H10080 N[I] N[I]
      IF [.IBE1(6)]=1 THEN G4 FL931
      ENDI
      (retrait -> ER)
      [cote]='ER G0 G77 H10080 N[I] N[I]
      REPEAT
           (calcul cote approche)
           [cote]=[garde]+[prof]G0 G77 H10080 N[I] N[I]
           (calcul passes suivantes et execution)
           'I = 'Q-'P * 'I/'L + 'P + 'I   'J = 'L-'I
           IF    'J <= 0 THEN
      EXIT
           ENDI
           IF 'J < 'Q/2 THEN  'I = -'Q/2 + 'L
           ENDI
           IF 'ER > L[LZ]  THEN  [prof] = -'I + 'ER
           ELSE [prof] = 'I + 'ER
           ENDI
           [cote]=[prof] G9 G1 F[RF] G77 H10080 N[I] N[I]
           IF [.IBE1(6)]=1 THEN G4 FL931
           ENDI
           [cote]='ER G0 G77 H10080 N[I] N[I]
      UNTIL 'I = 'L (test si fin de boucle)
```

```
   ENDI
 (descente fond de trou)
 [cote]=L[LZ] G1 F[RF] G77 H10080 N[I] N[I]
ENDI
N100
(temporisation par EF)
IF [.IBE1(6)]=1 THEN G4 FL931
ENDI
(remontee a ER)
[cote] = 'ER  [I]=[IZ]+10 G0 G77 H10080 N[I] N[I]
G997 G9 M[M998]
G[G90/1] G[G0/1] F[RF] E11005=[diam]
```

Sous programme %10080 appelé par le cycle %10083

```
%10080
(analyse des cycles de percage_taraudage_etc..)
IF [.IBE0(6)] = 1 THEN FL905
ENDI
IF [.IBE0(19)] = 1 THEN SL918
ENDI
IF [.IBE0(20)] = 1 THEN TL919
ENDI
BCLR [.IBE0(6)]/[.IBE0(19)]/[.IBE0(20)]
(lire sens rotation broche et M d'enchainement blocs)
[M3/4]=3*[.BM03] [M3/4]=4*[.BM04]+[M3/4]
[M998]=[.BM999]-[.BM997]+998 M997
(lire le rang de l'axe d'outil)
[IZ] = [.RDX]
IF [IZ] < 0 THEN  [IZ] = -[IZ]
ENDI
(G21,G22 interdit en cycle usinage)
[E]=[.BG21]+[.BG22] G79 [E]>0 N85
(plan et axe outil compatible?)
IF [.BG20]=1 THEN [Gplan]=20
ELSE [Gplan]=[.BG19]-[.BG17]+18
[E]=[Gplan]+[IZ] G79 [E]<>20 N83
ENDI
(acquerir les rangs des axes et le poste dans L900 de l'axe d'outil)
[LZ]=922+[IZ] G79 N[IZ]
N1 [IX]=5 [IY]=6 G79 N3+1
N2 [IX]=4 [IY]=6 G79 N3+1
N3 [IX]=4 [IY]=5
(choisir l'axe primaire ou secondaire )
(sur axes perpendiculaires a axe outil)
IF [.IBX2(IX)] = 0 THEN [IX] = [IX]-3
ENDI
```

```
IF   [.IBX2(IY)] = 0 THEN [IY] = [IY]-3
ENDI
(et sur axe outil )
IF [.IBX2(IZ)] = 0 THEN [IZ] = [IZ]+3 [LZ]=[LZ]-3
 IF [.IBX2(IZ)] = 0 THEN E.880
 ENDI
ENDI
(mettre dans 'ER= la derniere cote Z, retour si ER n'a pas ete programme)
IF [.IBE1(18)] = 0 THEN  'ER = [..IRX(IZ)]
ELSE  [E]=[IZ]-1*1000+70007
(Si programmation au diametre, correction de 'ER)
IF E[E]=1 THEN 'ER='ER/2
ENDI
ENDI
[LZ]=[LZ]+26  ( LZ pointe les variables L926..L951 )
(sur 1er bloc initialiser la cote de fond de trou)
IF [..BG80] = 1 THEN L[LZ]=[..IRX(IZ)]
ENDI
(si programmee memoriser la nouvelle  cote de fond de trou )
IF [.IBX(IZ)] = 1 THEN  L[LZ] = [.IRX(IZ)]
ENDI
(test si orientation compatible avec sens usinage)
IF 'ER <> L[LZ] THEN
 IF 'ER>L[LZ] AND [.RDX]<0 THEN E.890
 ENDI
 IF 'ER<L[LZ] AND [.RDX]>0 THEN E.890
 ENDI
ENDI
(memoriser le type de positionnement)
[G0/1]=3*[.BG03][G0/1]=2*[.BG02]+[.BG01]+[G0/1] [G90/1]=90+[.BG91]
(memoriser vitesse , tempo )
[RF]=[.RF]
(si ED programme on le prend en compte , et si interpo lineaire )
(on force positionnement des axes deja programmes precedemment)
IF [.IBE1(4)] = 1 THEN  EDL929
BCLR [.IBE1(4)]
 IF [G0/1] < 2 THEN  G91 [cote]=0
  IF [.IBX1(IX)] = 1 THEN [I]=[IX]+10 G77 H10080 N[I] N[I]
  ENDI
  IF [.IBX1(IY)] = 1 THEN [I]=[IY]+10 G77 H10080 N[I] N[I]
  ENDI
 ENDI
ENDI
(memoriser presence puis invalider l'axe d'outil)
[I]=[.IBX(IZ)]+[.IBX(IX)]+[.IBX(IY)]  G90
BCLR [.IBX(IZ)]
```

```
(tst rot broche si axe programme)
IF [I]<>0 THEN [E]=[M3/4]*[.RS] G79 [E]=0 N81
ENDI
G79 N800
(message de defaut)
N81 E.831 (broche a l'arret)
N82 E.882 (fond de trou non programmee)
N83 E.890 (orientation outil incompatible)
N84 E.894 (ER interdit en G20)
N85 E.895 (G21,G22 interdit dans ce cycle)
(-deplacements-)
N11 X[cote]
N12 Y[cote]
N13 Z[cote]
N14 U[cote]
N15 V[cote]
N16 W[cote]
N800
```

6 Interpolation polynomiale

6.1 Généralités

L'interpolation polynomiale est un outil permettant la définition de trajectoires à partir de polynômes. Elle est utilisée pour la réalisation de courbe spline.

La position sur chacun des axes est définie par un polynôme basé sur un paramètre indépendant (sans dimension) variant de 0 à 1 du début à la fin de la trajectoire.

On distingue deux variantes de l'interpolation polynomiale :
- l'interpolation polynomiale segmentée,
- l'interpolation polynomiale lisse.

Distinction entre interpolation polynomiale segmentée et lisse

En interpolation polynomiale segmentée, le pas de segmentation est fonction de la vitesse programmée; chaque segment est calculé de façon à être exécuté en 10 ms et est interpolé linéairement à chaque échantillonage.

En interpolation polynomiale lisse, l'interpolation est exécutée en temps réel ; un point sur la courbe étant calculé à chaque échantillonage.

Fonctionnalités en option

L'utilisation de l'interpolation polynomiale lisse nécessite que l'option N°52 (Interpolation polynomiale lisse) soit validée ; si l'option N°52 n'est pas validée, la programmation de l'argument I.. dans la syntaxe est ignorée et c'est une interpolation segmentée qui est réalisée si l'option N°51 (courbe spline) est validée.

L'interpolation polynomiale segmentée (absence du I.. dans la syntaxe) est acceptée si l'une ou l'autre des options N°51 ou N°52 est validée.

6.2 Programmation de l'interpolation polynomiale segmentée

Dans la syntaxe du bloc, chaque polynôme est caractérisé par la position d'arrivée suivie des coefficients de degré croissant séparés par le caractère « / » .

Syntaxe

N.. G01 X../ Coefficients / Coeff n^{ème} dg Y../ Coefficients Z../ Coefficients ...

G01	Fonction d'interpolation linéaire et polynomiale.
X..	Cote d'arrivée de l'interpolation suivant X.
/ Coeff n^{ème} degré	Coefficients (du 1er, 2ème degré etc...) du polynôme.
Y.. Z..	Cote d'arrivée de l'interpolation suivant Y, Z et autres axes.

6.2.1 Particularités des axes et coefficients programmés

La somme des coefficients de degrés supérieurs à 0 doit être égale au déplacement relatif de l'axe dans le bloc.

REMARQUE *On notera qu'il n'y a pas d'émission de message en cas d'erreur dans la somme des coefficients.*

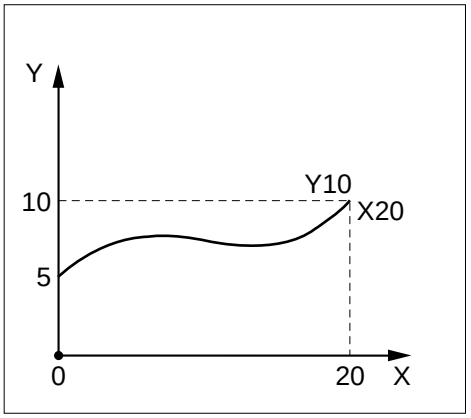
Les coefficients sont exprimés :
- en mm (ou en pouce si G70) pour les axes linéaires,
- en degré pour les axes rotatifs.

Dans un même bloc d'interpolation polynomiale :
- certains axes peuvent être programmés avec des degrés différents,
- une interpolation linéaire peut être programmée.

Par exemple :

```
N..
N80 G01 X0 Y5 Z-5
N90 X20/9.5/22/-11.5  Y10/18/-33/20  Z10
N..
```

L'axe Z est interpolé linéairement



6.2.2 Transformations géométriques

Toutes les transformations géométriques suivantes peuvent être appliquées à la courbe :
- décalage programmé (G59),
- miroir (G51 ...),
- facteur d'échelle (G74).
- décalage angulaire (ED..),

REMARQUE *Afin d'effectuer un décalage angulaire, les 2 axes du plan d'interpolation doivent être programmés et doivent avoir le même nombre de coefficients, il faut éventuellement compléter le polynôme d'un axe par des coefficients de valeur 0 pour qu'il dispose du même nombre de coefficients que son partenaire, si ce n'est pas le cas le système émet le message d'erreur 133.*

Par exemple :

Sans décalage angulaire

```
G1 X20  Y0 /50/-180/240/-110
```

Avec décalage angulaire ED..

```
G1 X20 /20/0/0  Y0 /50/-180/240/-110
```

Correction de rayon en interpolation polynomiale

En correction de rayon (G41 ou G42) le déport normal de l'outil n'est effectué que si les deux axes du plan d'interpolation sont programmés.

Durant toute l'interpolation (du point de départ au point d'arrivée), l'outil est maintenu normal à la courbe dans le plan d'interpolation.

Des courbes polynomiales qui se suivent doivent être tangentes. En cas d'enchaînement non tangent d'une courbe avec une autre courbe polynomiale (droite ou cercle), le raccordement est effectué par un cercle de raccordement qui positionne l'outil normal à la nouvelle courbe en son point de départ.

Lorsqu'il y a enchaînement de deux courbes et que le diamètre de l'outil est trop important pour se positionner à la normale de l'une des trajectoires programmées (rayon de raccordement inférieur au rayon d'outil, ou trajectoire inaccessible), le système applique malgré cela les règles de continuité des trajectoires; il en résulte un enlèvement de matière indésirable.

6.2.3 Vitesse d'avance en interpolation

Le pas de segmentation de la courbe est directement lié à la vitesse d'avance programmée :

- en G93 (V/L) et G94 (mm/min) le pas de segmentation est calculé de façon à être exécuté en 10 millisecondes (ms) si la période d'échantillonnage est inférieure à 5 ms.
- en G95 (mm/tour) le pas de segmentation est égal à l'avance par tour programmée.

6.2.4 Limitation du nombre de coefficients

Pour chaque bloc, les coefficients sont rangés dans la pile programme qui comporte au maximum 32 postes. Chaque coefficient est rangé dans un poste; de plus pour chaque axe, un poste est occupé par l'adresse physique de l'axe suivi de son nombre de coefficients.

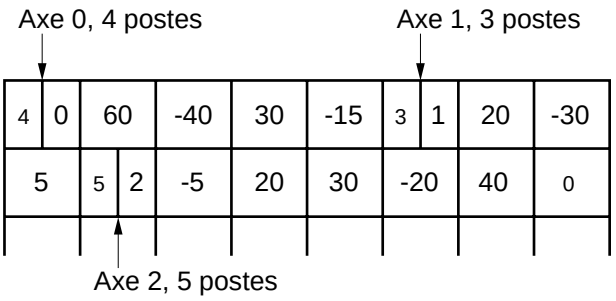
En cas de saturation de la pile , le système émet le message d'erreur 6.

Les coefficients ont le même format que celui des cotes et ont les mêmes valeurs limites.

Par exemple :

Rangement des coefficients d'un bloc dans la pile.

```
N20 X100/60/-40/30/-15  Y70/20/-30/5  Z80/-5/20/30/-20/40
```



6.3 Programmation de l'interpolation polynomiale lisse

Dans la syntaxe du bloc, chaque polynôme est caractérisé par la position d'arrivée suivie des coefficients de degré croissant séparés par le caractère « / » .

L'argument I.. dans la syntaxe différencie de l'interpolation polynomiale lisse de l'interpolation polynomiale segmentée (voir 6.1).

Syntaxe

N.. G01 X../ Coefficients / Coeff n^{ème} dg Y../ Coefficients Z../ Coefficients ... I..

G01	Fonction d'interpolation linéaire et polynomiale.
X..	Cote d'arrivée de l'interpolation suivant X.
/ Coeff n^{ème} degré	Coefficients (du 1er, 2ème degré, etc...) du polynôme.
Y.. Z..	Cote d'arrivée de l'interpolation suivant Y, Z et autres axes.
I..	Longueur le courbe (trajectoire sur laquelle s'applique la vitesse d'avance programmée).

6.3.1 Particularités en interpolation polynomiale lisse

Dans la déclaration des polynômes, la présence du coefficient de degré le plus élevé est facultative; la somme des coefficients d'un axe étant égale à sa cote relative, le système a la possibilité de le déterminer.

Par exemple :

```
...
G01 X0 Y..
X20/10/-5/          Le coefficient de degré 3 est égal à (20-0)-(10-5) = 15
...
```

Application particulière

En interpolation polynomiale lisse le contrôle du paramètre I.. permet aussi d'appliquer la vitesse programmée, non pas à la trajectoire, mais par exemple à un seul axe :

Application de la vitesse à l'axe X

```
...
G01 X10 Y10 F1000
X20  Y25/30/  I10     Interpolation linéaire en X et 2ème degré
                      (25-10)-30 = -15 en Y
X35  Y50/25/  I15     Interpolation linéaire en X et en Y
                      ( coefficient du 2ème degré = 0 )
...
```

6.3.2 Restrictions en interpolation polynomiale lisse

En interpolation polynomiale lisse la correction d'outil en G41, G42 et G29 ne peut être utilisée.

Le recul sur trajectoire défini par une interpolation polynomiale lisse est impossible.

7 Transformations de coordonnées

7.1 Généralités

Les transformations de coodonnées sont effectuées par l'utilisation d'une matrice carrée. Cet outil est utilisé par l'application NUM traitant l'usinage dans un plan incliné.

La transformation des mouvements $\vec{X}' = K . \vec{X}$ est effectuée en aval des interpolateurs par la matrice représentée ci-après.

Matrice de transformation des coordonnées

	X	Y	Z
X'	Kxx	Kxy	Kzx
Y'	Kxy	Kyy	Kzy
Z'	Kxz	Kyz	Kzz

REMARQUE *Les contrôles de courses, les limitations de vitesses et d'accélérations sont traités en amont, au niveau de la préparation des interpolations.*

7

7.2 Mise en œuvre de la matrice de transformation des coordonnées

La mise en œuvre de la matrice de transformation est réalisée par la syntaxe de programmation ci-après. Cette syntaxe contient la fonction de validation de la matrice suivie des coefficients P, Q et R séparés par le caractère « / » (ces coefficients sont normés à 1).

Syntaxe

N.. **G24**+ X.. Y.. Z.. P(Kxx)/(Kyx)/(Kzx) Q(Kxy)/(Kyy)/(Kzy) R(Kxz)/(Kyz)/(Kzz)

G24+	Fonction de validation de la matrice de transformation des coordonnées.
X.. Y.. Z..	Origine du référenciel de la matrice de transformation.
P(Kxx)/(Kyx)/(Kzx)	Coefficients suivant X de la matrice.
Q(Kxy)/(Kyy)/(Kzy)	Coefficients suivant Y de la matrice.
R(Kxz)/(Kyz)/(Kzz)	Coefficients suivant Z de la matrice.

Révocation

La transformation validée (G24+ ...) est révoquée :
- par G24- ne comportant aucun argument,
- à la remise sous tension.

Propriétés de la fonction

Les arguments de transformation sont modaux. Après une révocation par G24-, il est possible de reprogrammer G24+ suivi de tous les arguments ou G24+ seul dans le bloc.

La transformation définie par la fonction G24+ est appliquée immédiatement et reste maintenue sur une RAZ.

Particularités

Les origines X, Y et Z définissent l'origine du référenciel de la matrice par rapport au référenciel de base déclaré par PREF + DEC1 (+ éventuellement DEC3). Ces origines doivent obligatoirement :
- être positionnées derrière la fonction G24+,
- précéder les coefficients P, Q et R.

Il est possible de valider la matrice de transformation alors que la prise d'origine (POM) n'est pas encore effectuée sur les axes (par exemple, pour dégager l'outil selon l'orientation de la tête après la mise sous tension).

La prise d'origine (POM) sur un des axes du plan ne doit par être demandée avec la matrice de transformation validée; sinon en mode "POM" le système émet le message d'erreur 24.

Il est possible d'utiliser la fonction G24 pour d'autres appliquations qu'une rotation du plan de programmation dans l'espace (par exemple pour la correction d'axes non orthogonaux); dans ce cas, les valeurs des coefficients de la matrice de transformation doivent être comprises entre -8 et +8 (en cas contraire, le système émet le message d'erreur 24). Il en est de même pour les valeurs de sa matrice inverse qui est calculée par le système. On notera que les vitesses de passages d'angles et les accélérations limites peuvent ne pas être respectées.

Numéros et messages d'erreurs

Erreur 2 : Pas de signe + ou - derrière la fonction G24.

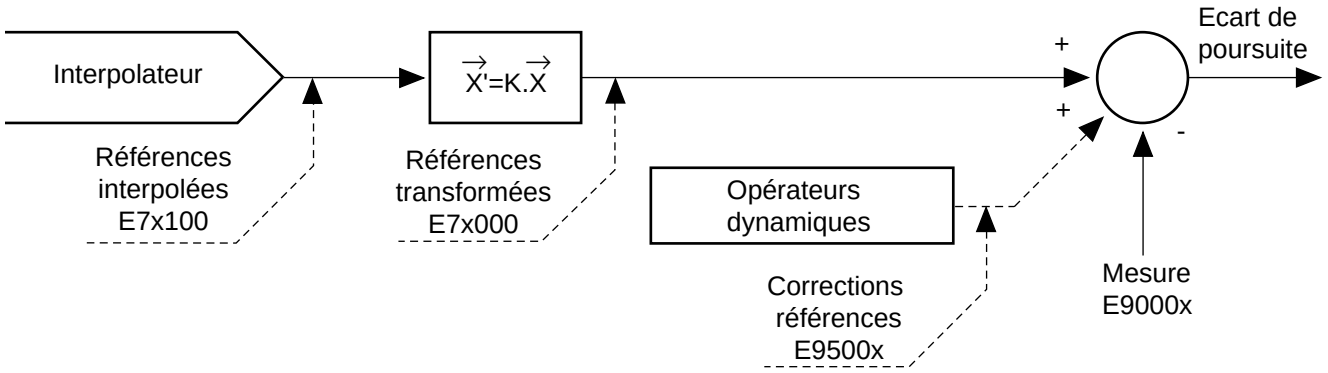
Erreur 14 : Option plan incliné invalide.

Erreur 24 : Erreur dans la déclaration d'un plan incliné.
- Nouvelle activation de la fonction alors qu'elle est déjà présente.
- Déclaration incomplète des arguments de la fonction.
- Axe du point de pivot inexistant ou non asservi.
- Valeur incohérente d'un des termes de la matrice.
- Programmation d'un axe du plan incliné interdite sous G52.

7.3 Application de la transformation de coordonnées

La transformation de coordonnées doit être appliquée derrière les interpolateurs, mais avant les traitements éventuels des opérateurs dynamiques et de la calibration inter-axes.

Représentation schématique



La calibration inter-axes s'appuie sur les références transformées pour calculer les corrections. Le paramètre E7x100 (à lecture seule) adresse les références de positions issues des interpolateurs.

7.3.1 Restrictions et conditions d'utilisation

La copie (avec ou sans transformation) des références interpolées E7x100 vers les références utilisées dans les asservissements E7x000, n'est effectuée que pour les axes déclarés dans les paramètres machine P2 ou P3 (modifiée éventuellement par le paramètre E9100x). Il est nécessaire d'en tenir compte dans les applications avec les opérateurs dynamiques qui utilisent des axes fictifs.

Il ne faut pas déclarer des opérateurs dynamiques qui calculent des corrections de références (E9500x : opérande destination) lorsque la matrice de transformation est active ; il est nécessaire au préalable :
- de révoquer la matrice de transformation (G24-),
- de positionner les opérateurs dynamiques,
- de rétablir la matrice de transformation (G24+...).

Les autres déclarations pouvant être effectuées avec la matrice de transformation inactive sont les suivantes :
- activation et désactivation de la calibration inter -axes, par E9400x,
- déclaration axe asservi ou non par E9100x,
- activation et désactivation des opérateurs dynamiques N° 15, 20 et 21,
- prise d'origine (POM).

7.3.2 Temps de traitement

Temps de traitement de la transformation : 80 microsecondes.
Temps de préparation des interpolations : 500 microsecondes de plus.

7.4 Exemple de sous programme d'application

Sous programme d'application %10150

Cet exemple n'est donné qu'à titre indicatif.

```
%10150
VAR  [A]=[.RX(7)] [B]=[.RX(8)] [C]=[.RX(9)]
     [K11] [K12] [K13]
     [K21] [K22] [K23]
     [K31] [K32] [K33]
     [V] [X] [Y] [Z]
ENDV

(Remettre les cotes du bloc précédent dans A B et C)
IF [.IBX(7)] = 1 THEN
       [V]=[..IRX(7)] A[V]
ENDI
IF [.IBX(8)] = 1 THEN
       [V]=[..IRX(8)] B[V]
ENDI
IF [.IBX(9)] = 1 THEN
       [V]=[..IRX(9)] C[V]
ENDI

(Lire le point de pivot du plan incliné)
(et remettre les cotes du bloc précédent dans X Y et Z
[X]=[.IRX(1)]  [V]=[..IRX(1)] X[V]
[Y]=[.IRX(2)]  [V]=[..IRX(2)] Y[V]
[Z]=[.IRX(3)]  [V]=[..IRX(3)] Z[V]

(Modifier si nécessaire [X] [Y] [Z] du déport nez de broche dû à la rotation tête)
(Faire en sorte qu'il n'y ait pas de déplacement)
BCLR  [.IBX(1)]/[.IBX(2)]/[.IBX(3)]/[.IBX(7)]/[.IBX(8)]/[.IBX(9)]

(Calculer les coefficient de la matrice)
[K11]=C[C]*C[B]
[K12]=S[C]*C[A] [K12]=C[C]*S[B]*S[A]-[K12]
[K13]=S[C]*S[A] [K13]=C[C]*S[B]*S[A]+[K13]
[K21]=S[C]*C[B]
[K22]=C[C]*C[A] [K22]=S[C]*S[B]*S[A]+[K22]
[K23]=C[C]*S[A] [K23]=S[C]*S[B]*C[A]-[K23]
[K31]=-S[B]
[K32]=C[B]*S[A]
[K33]=C[B]*C[A]

G24- (Invalider la matrice précédente)
(Valider la nouvelle matrice)
G24+ X[X] Y[Y] Z[Z] P[K11]/[K12]/[K13] Q[K21]/[K22]/[K23] R[K31]/[K32]/[K33]
G997 G80
```

Appel de la fonction de validation du plan incliné

```
%...
N..
N.. G150 X.. Y.. Z.. A.. B..
N..
```

8 Fonction RTCP

8.1 Généralités

La fonction RTCP (Rotating Tool Center Point) permet la prise en compte, de façon continue, de la partie cinématique de la machine entraînant l'orientation de l'outil par rapport à la pièce en le faisant pivoter autour de son centre.

Le programme d'usinage contient les coordonnées cartésiennes du bout de l'outil ou du point de contact du référentiel de la pièce.

Une application créée par NUM permet de traiter la majorité des cas de figures par fonction RTCP.

8.1.1 Commande des axes rotatifs

Selon le mode de commande des axes rotatifs, deux cas peuvent se présenter :

- les coordonnées rotatives sont programmées et associées aux coordonnées cartésiennes dans le programme d'usinage ; ce mode est alors désigné par le terme "RTCP programmé",
- les coordonnées rotatives ne sont pas programmées et les axes rotatifs sont pilotés manuellement par manivelles ou manipulateurs ; ce mode est alors désigné par le terme "3/5 AUTO" (voir fonction N/M AUTO dans le chapitre 9 du présent manuel).

8.1.2 Traitements effectués sur les axes

Les traitements effectués sur les axes permettent :

- de corriger de manière continue (selon l'évolution des axes rotatifs) les références des axes cartésiens (X, Y, Z) de façon à maintenir le point de contact de l'outil sur la trajectoire programmée. Cette correction est effectuée tant en "RTCP programmé" qu'en "3/5 AUTO",
- de s'assurer que les cotes des axes cartésiens (X, Y, Z) "corrigés" par la fonction RTCP restent à l'intérieur des courses machine. Ce contrôle n'est effectué qu'en "RTCP programmé",
- d'analyser les survitesses et les suraccélérations induites sur les axes cartésiens (X, Y, Z) par le déplacement des axes rotatifs, afin de déterminer l'accélération optimale sur la trajectoire et de limiter éventuellement la vitesse programmée de telle sorte qu'aucun axe ne soit sollicité au delà de sa vitesse maximum. Cette analyse n'est effectuée qu'en "RTCP programmé".

8.2 Mise en oeuvre de la fonction RTCP

La mise en œuvre de la fonction RTCP est réalisée dans le programme pièce par la syntaxe de programmation définie ci-après. Cette syntaxe contient le code de validation de la fonction RTCP suivie des arguments désignant les axes rotatifs (axes de tête "twist" et/ou de plateaux") induisant des mouvements sur les axes cartésiens et décrivant la cinématique machine.

Syntaxe

G26+ Arguments désignant les axes rotatifs

G26+ Validation de la fonction RTCP.

REMARQUE *Les arguments désignant les axes rotatifs sont spécifiques à la cinématique machine (voir § 8.3*)

Révocation

La fonction RTCP (G26+) est révoquée :
- par G26- ne comportant aucun argument,
- sur une RAZ.

Particularités

La fonction RTCP peut être validée alors que les axes cartésiens et rotatifs occupent une position quelconque ; cette validation redéfinit automatiquement la nouvelle position des axes cartésiens dans le repère programme compte tenu de la cinématique machine.

Le paramètre externe E11018 (à lecture seule) indique l'état validé ou invalidé de la fonction RTCP :
- état "1", fonction validée (G26+),
- état "0", fonction invalidée (G26-).

Erreurs de programmation

Lorsqu'une erreur de programmation est détectée au moment de la validation de la fonction RTCP, le système émet le message d'erreur 16 regroupant les défauts suivants :

- programmation de G26+ alors que le RTCP ou le plan incliné sont déjà validés. G24+ et G26+ sont incompatibles,
- absence de programmation du vecteur IJK (ou d'une de ses composantes, ou d'une double déclaration d'une composante) avant la déclaration d'un axe "twist"; ou programmation de ce vecteur alors qu'aucun axe "twist" n'est déclaré,
- déclaration d'un axe "twist" après déclaration d'un "plateau". En "twist", la cinématique machine doit être décrite en partant de la position centre outil vers le bâti machine et en "plateau" la cinématique doit être décrite du bâti vers la pièce,
- déclaration d'un axe rotatif n'appartenant pas au groupe dans lequel le RTCP est validé,
- absence d'arguments ou trop d'arguments déparés du caractère "/". Les axes rotatifs et les points de pivot doivent être programmés avec trois arguments et les inclinaisons avec deux arguments,
- nombre d'articulations et d'inclinaisons de plans supérieurs à sept,
- absence du numéro de fonctionnalité RTCP en page "OPTIONS" de la CN.

Autres défauts

Les défauts suivants peuvent aussi entraîner l'émission d'un message d'erreur :

- le changement de correcteur d'outil alors que le RTCP est validé provoque aussi l'émission du message d'erreur 16,
- lorsque le RTCP est validé, la POM au clavier est refusée et la POM par manipulateurs génère le message d'erreur 24,
- lors de la validation du plan incliné, si la POM n'est pas effectuée sur les axes rotatifs le système émet le message d'erreur 159.

8.3 Description des cinématiques

Selon le cas "twist" ou "plateaux", la description de la cinématique machine est définie à partir de sa référence particulière :

- "twist" : la cinématique est décrite en partant de la position centre outil vers le bâti machine,
- "plateaux" : la cinématique est décrite en partant du bâti machine vers la pièce.

Les cinématiques machine "twist" et "plateaux" présentées dans la présente section du manuel sont définies un exemple de configuration correspondant a chaque cas. Selon le cas la validation de la fonction RTCP est donc appliquée avec des arguments spécifiques désignant les axes rotatifs.

Divers cas de configurations sont traités par une application NUM.

8.3.1 Description des "twist"

Dans le cas des têtes "twist", chaque articulation est décrite par un axe de rotation et un vecteur de translation. Le vecteur de translation est défini pour la position "0" de l'axe de rotation.

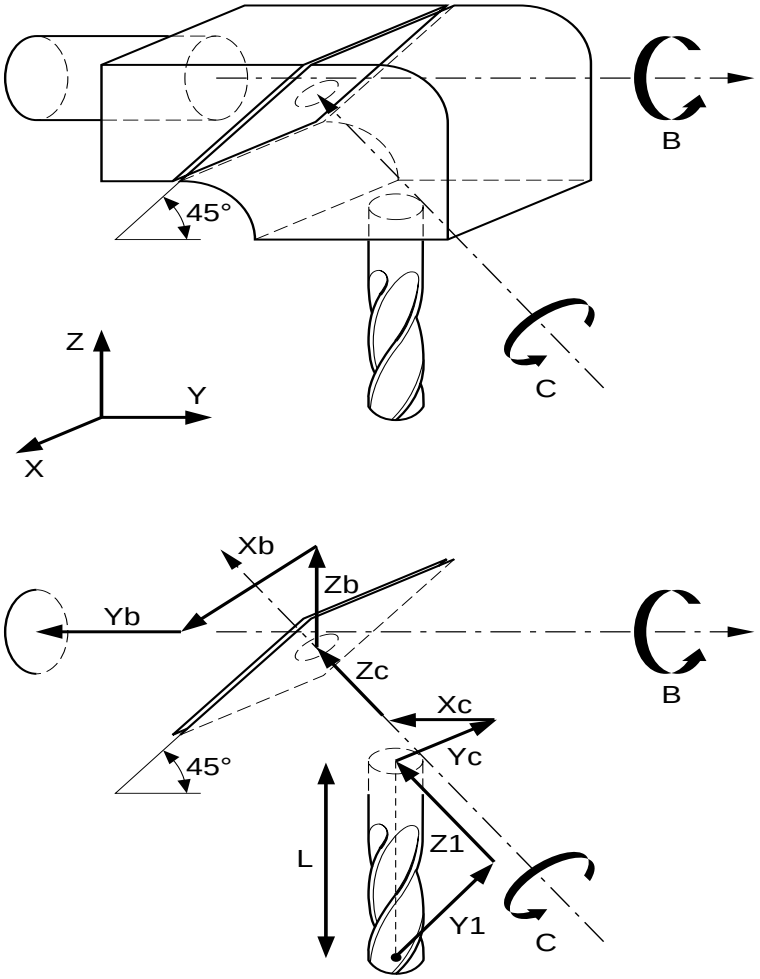
L'axe de rotation est déclaré par TA, TB, ou TC suivi des trois translations associées séparées par le caractère "/".

Si une articulation repose sur un plan incliné, celui-ci est déclaré par TI, TJ ou TK selon que le plan est parallèle à X, Y ou Z et est suivi du cosinus puis du sinus de l'angle d'inclinaison du plan.

La direction de l'outil est définie par les composantes d'un vecteur unitaire I, J, K orienté dans le plan de rotation de l'outil. Ce vecteur permet d'inclure dans la cinématique la jauge d'outil ainsi que ses corrections dynamiques qui peuvent être introduites en cours d'usinage.
Le vecteur I, J, K est le premier argument suivant la fonction RTCP (G26+).

Exemple

Représentation de la cinématique tête "twist"



Programmation

```
[xl]=Xl/L [yl]=Yl/L [zl]=Zl/L
G26+ I[xl] J[yl] K[zl] TCXc/Yc/Zc TIca/sa TBXb/Yb/Zb
```

8.3.2 Description des "plateaux"

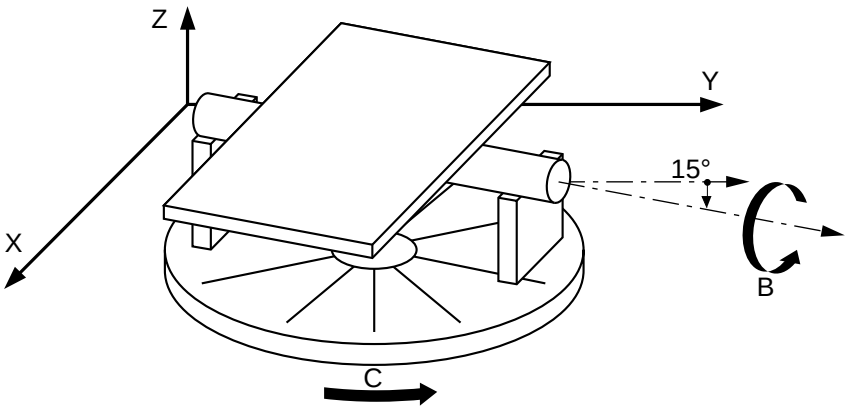
Dans le cas des "plateaux", chaque articulation est décrite par un axe de rotation et un vecteur de translation. Le premier vecteur de translation positionne en absolu le centre du premier plateau (plateau reposant sur le bâti machine) et le ou les plateaux suivants sont déclarés en relatif pour des position "0" des plateaux.

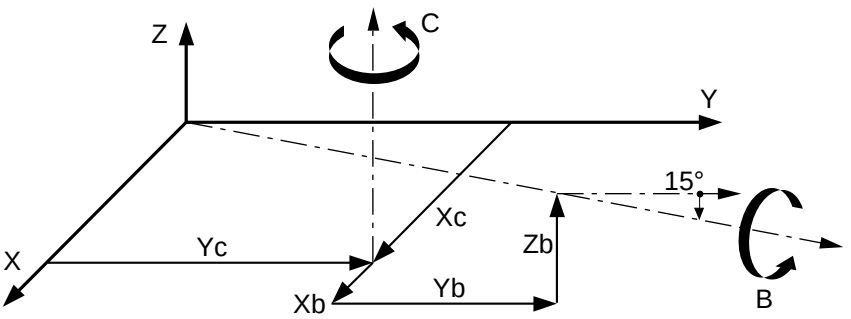
L'axe de rotation est déclaré par PA, PB ou PC suivi des trois translations associées séparées par le caractère "/".

Si une articulation repose sur un plan incliné, celui-ci est déclaré à sa suite par PI, PJ ou PK selon que le plan est parallèle à X, Y ou Z, suivi du cosinus puis du sinus de l'angle d'inclinaison du plan.

Exemple

Représentation de la cinématique "plateaux"





Programmation

[ca]=cos 15° [sa]=sinus 15°
G26+ PCxc/yc/0 PBxb/yb/zb PI[ca]/[sa]

8.4 Traitements liés à la fonction RTCP

Les autres traitements géométriques pouvant être associés à la fonction RTCP sont les suivants :

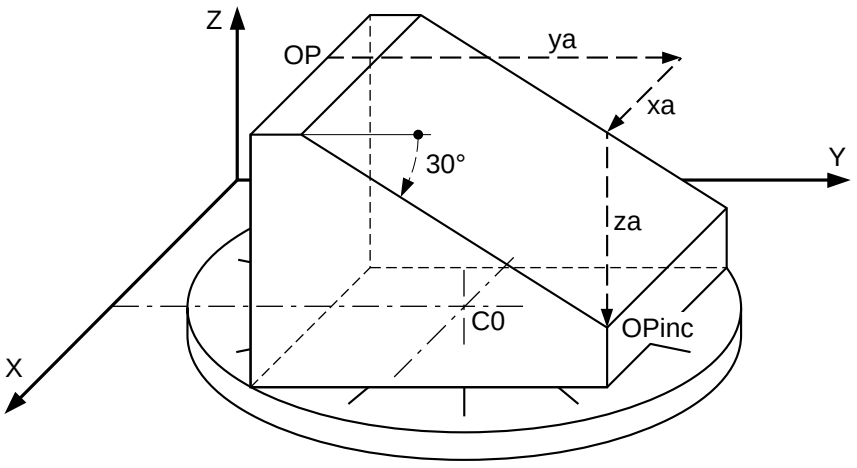
- pièce sur plan incliné,
- excentration de plateau,
- correction de longueur d'outil,
- correction dynamique d'outil,
- correction d'outil dans l'espace.

8.4.1 Pièce sur plan incliné

Lorsque la pièce devant être exécutée repose sur un plan incliné, ce plan est spécifié par un point de pivot déclaré dans la syntaxe de programmation par PD suivi des trois composantes d'un vecteur de translation puis de une ou deux inclinaisons définies par PI, PJ ou PK suivies du cosinus et du sinus des angles. Ce vecteur de translation positionne le point de pivot par rapport à l'origine programme et ce point de pivot devient la nouvelle origine programme.

Exemple

Représentation de la cinématique sur plan incliné



OP = ancienne origine programme
OPinc = nouvelle origine programme

Programmation

```
[ca]=cos-30° [sa]=sinus-30°
G26+ PCxc0/yc0/zc0 PDxa/ya/za PI[ca]/[sa]
```

8.4.2 Excentration de plateau rotatif (DEC3)

Les décalages liés à l'excentration du plateau rotatif (DEC3) ne sont plus traités en tant que tels lorsque la fonction RTCP est validée. La connaissance de la position de l'axe de rotation du plateau porte-pièce fait que cette correction est automatiquement incluse dans la fonction RTCP.

8.4.3 Correction de longueur d'outil

Lorsque l'outil est porté par un axe "twist", sa correction de longueur n'est plus traitée en tant que décalage à apporter sur l'axe de l'outil, mais est incluse dans la cinématique des "twist" avec l'orientation déclarée dans le vecteur IJK programmé (voir description des "twist").

Déclaration et dimension affectée au correcteur d'outil

Le correcteur d'outil doit être déclaré avant la validation de la fonction RTCP. La valeur affectée au correcteur est mémorisée au moment de la validation du RTCP et est conservée tant que le RTCP reste valide. Les modifications éventuelles de la jauge ne seront prises en compte qu'après invalidation du RTCP (G26-), puis d'une nouvelle validation du RTCP.

8.4.4 Correction dynamique d'outil

Les corrections dynamiques de longueur d'outil inférieures à 100µ (0,1 mm) sont immédiatement prises en compte et sont incluses (comme la correction de longueur) dans la cinématique des "twist" avec l'orientation déclarée dans le vecteur IJK programmé (voir description des "twist").

8.4.5 Correction d'outil dans l'espace (G29)

Correction d'outil 3 ou 5 axes en G29

Correction d'outil 3 axes

Lorsque la fonction RTCP est utilisée avec la correction d'outil "3 axes" (absence du vecteur d'orientation d'outil IJK dans les blocs en G29), le système ne traite que le cas de l'outil sphérique.

Correction d'outil 5 axes

Lorsque la fonction RTCP est utilisée avec la correction d'outil "5 axes" (présence du vecteur d'orientation d'outil IJK dans les blocs en G29), le système permet de traiter le cas de l'outil torique.

Visualisation de la position centre outil en G29

En cours d'interpolation, la page de coordonnées du point courant "AXES" par rapport à l'origine programme (OP) permet de visualiser les cotes centre outil.

Particularité de correction d'outil en G29 avec plateaux (sans axe "twist")

Lorsque la cinématique d'une machine ne comporte que des "plateaux" (pas d'axe "twist"), il est nécessaire pour que la correction dans l'espace (G29) soit prise en compte de déclarer derrière la fonction G26+ : le vecteur de direction d'outil IJK suivide l'argument TD avant de décrire la cinématique des "plateaux".

Par exemple :

```
G26+ IJK1 TD0/0/0  PC../../..  ...
...
```

8.5 Utilisation en modes "JOG" et "INTERV"

Dans les modes manuel "JOG" et intervention "INTERV" (suite à un arrêt d'usinage "ARUS") avec fonction RTCP validée, le déplacement de l'axe commandé par manipulateur est limité à une valeur telle qu'aucun axe ne franchisse ses fins de courses. De même l'accélération et la vitesse de déplacement de l'axe peuvent être réduites de manière à ne pas dépasser leurs limites sur les autres axes entraînés.

En "JOG" par manivelle, seul le contrôle des fins de courses est effectué.

8.6 Restrictions et conditions d'utilisation

Dans les applications par opérateurs dynamiques, certaines restrictions et conditions d'utilisation de la fonction RTCP sont à respecter.

Lorsque la fonction RTCP est active, il ne faut pas déclarer des opérateurs dynamiques qui calculent des corrections de référence (E9500x : opérande destination), il est nécessaire au préalable :
- de révoquer la fonction RTCP (G26-),
- de positionner les opérateurs dynamiques,
- de revalider la fonction RTCP (G26+ ...).

Les autres déclarations pouvant être mises en oeuvre avec le RTCP inactif sont les suivantes :
- activation et désactivation de la calibration inter-axes par E9400x,
- déclaration axe asservi ou non par E9100x,
- activation ou désactivation des opérateurs dynamiques N°15, 20 et 21,
- prise d'origine (POM).

9 Fonction N/MAUTO

9.1 Généralités

N/M AUTO signifie que N axes, parmi les M axes d'une machine sont contrôlés par le programme pièce, tandis que le ou les autres sont commandés manuellement. Par exemple :
- 2 axes interpolés et le troisième axe déplacé en manuel (2/3 AUTO) ,
- 3 axes interpolés et les deux autres déplacés en manuel (3/5 AUTO).

Le ou les axes CN devant être déplacés manuellement doivent être préalablement déclarés "non interpolés" (ces axes sont dits "axes NMA").

Le déplacement manuel de l'axe NMA peut être effectué :
- soit par les manipulateurs d'axes (JOG),
- soit par sa manivelle associée.

La fonction N/M AUTO est active alors que la machine est en cycle :
- parallèlement à la conduite des autres axes interpolés en mode continu (CONT), séquentiel (SEQ) (et cela à vitesse de déplacement rapide ou de travail),
- et que la CN n'est ni en mode manuel (JOG), ni en arrêt programmé (M12).

REMARQUE *La fonction RTCP (Voir chapitre 8 du présent manuel) peut éventuellement agir en aval.*

9.1.1 Conditions générales requises en N/M AUTO

Le fonctionnement N/M AUTO nécessite que :
- la fonction automatisme soit programmée en langage ladder avec PLCTOOL,
- la fonctionnalité N°16 soit présente en page "OPTION" de la CN.
- la version G (minimum) du logiciel soit installée ; si la fonction RTCP doit être activée, la version J du logiciel est nécessaire (de plus cette version réalise le contrôle des courses, vitesses et accélérations sur les axes entraînés).

9.1.2 Axes non interpolés et axe NMA

A un instant donné, cinq axes au plus peuvent être déclarés non interpolables ; un axe est déclaré non interpolable par programmation du paramètre externe E912xx à l'état 1 (xx=adressse physique de l'axe de 00 à 31).

Les axes déclarés non interpolables seront réellement non interpolés qu'après autorisation-validation en provenance de la fonction automatisme par la mise à 1 de la variable %W2.1 (C_NMAUTO) ; voir mise en œuvre de la fonction N/M AUTO en §9.2.

REMARQUE *Par définition un axe "non interpolé" ne sera plus déplacé sous contrôle du programme pièce.*

On notera que parmi les cinq axes non interpolés (à un instant donné) :
- seul un axe peut être piloté en NMA à l'aide de ses manipulateurs ou de sa manivelle,
- les quatre autres axes non interpolés (au plus) ne sont pas entraînés (au moins directement). Par contre si la fonction RTCP est active un axe non interpolé peut être entraîné indirectement,
- tous les autres axes sont normalement interpolés.

Pour être déclaré non interpolé l'axe CN doit :
- être asservi,
- ne pas appartenir à un autre groupe.

Les axes non interpolés peuvent appartenir à l'un des huit groupes ; cependant la fonction RTCP est uniquement limitée au groupe 0.

REMARQUE *En aucun cas la fonction N/M AUTO ne permet de superposer deux mouvements sur le même axe.*

9.1.3 Erreurs en N/M AUTO

Le message d'erreur 99 est généré quand :
- plus de 5 paramètres externes E912xx sont mis à 1,
- le paramètre E912xx est écrit alors que l'axe xx associé est non asservi,
- le paramètre E912xx est écrit dans un programme pièce alors que l'axe associé xx appartient à un autre groupe.

9.1.4 Cas de traitement

Par exemple, un axe rotatif C est piloté en NMA à l'aide de la manivelle. Cet axe entraîne, via la fonction RTCP, les axes X, Y et Z (les axes X, Y, Z sont eux programmés normalement).

A tout instant, la CN contrôle : courses, vitesses et accélérations sur les axes X, Y, Z et C.

Concernant l'exemple, on notera :
- qu'un axe déclaré non interpolable peut cependant être programmé,
- que dans le cas traité l'interpolation continue, mais de façon fictive (c'est à dire qu'elle n'entraîne pas de mouvement direct). Par contre, dès que l'axe cesse d'être non interpolé, il est possible de reprendre la trajectoire programmée sans même attendre la fin du bloc en cours.

9.1.5 Exemples de programmes et d'utilisation

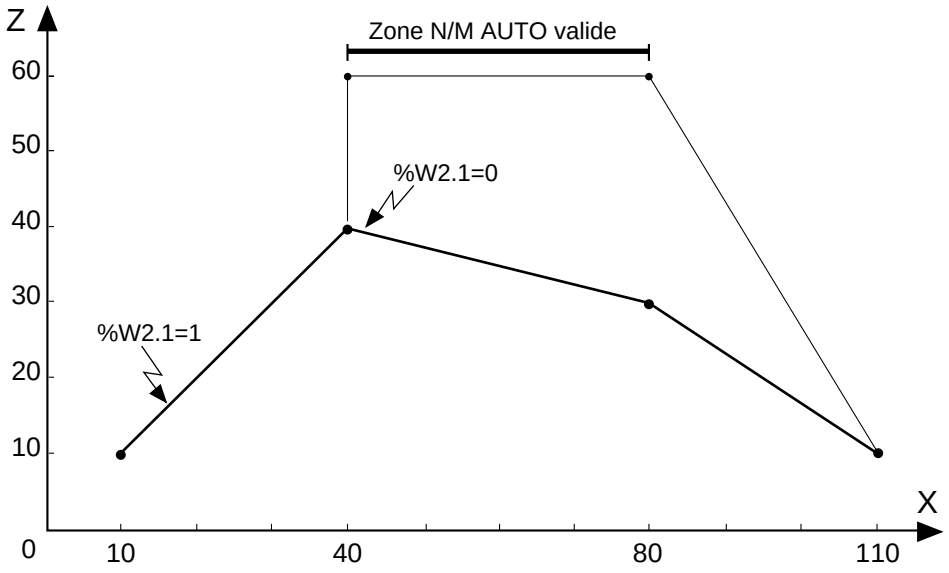
Les programmes ci-après définissent plusieurs cas possibles de validation et d'invalidation de la fonction N/M AUTO.

Cas de validation/invalidation sans "ARUS"

Programme original

```
%10
E91202=1
X10 Z10
X40 Z40
X80 Z30
X110 Z10
M02
```

Représentation schématique



Invalidation prise en compte suite à l'arrêt du cycle par fonction M00

```
%10
E91202=1
X10 Z10
X40 Z40
M00                    Arrêt programmé
X80 Z30                En manuel, axe Z déplacé à 60
M00                    Arrêt programmé
X110 Z10               Axe Z interpolé de 60 à 10
M02
```

Invalidation prise en compte suite à modification du paramètre E91202

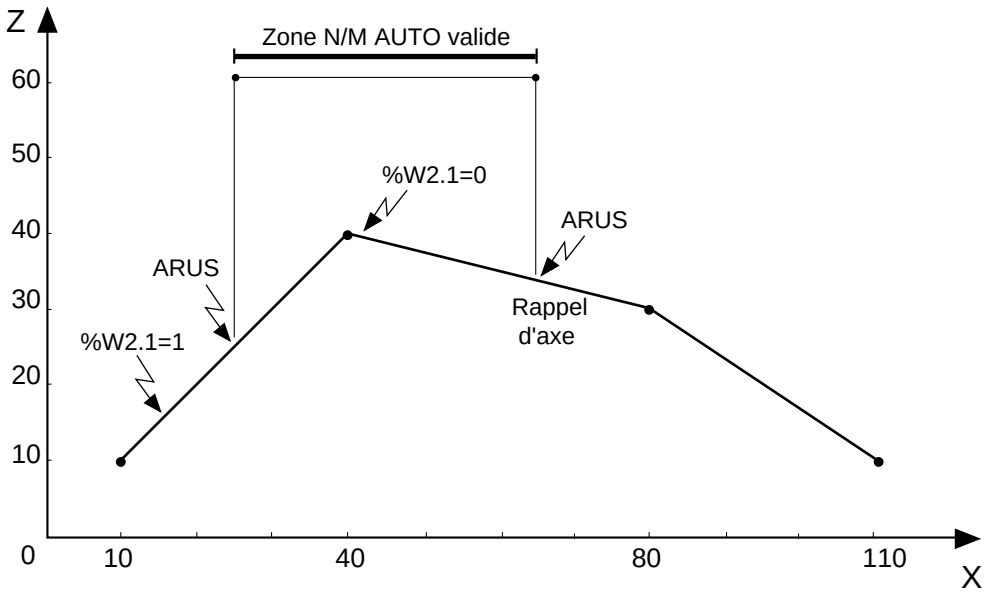
```
%10
E91202=1
X10 Z0
X40 Z40
M00                    Arrêt programmé
X80 Z30                En manuel, axe Z déplacé à 60
E91202=0               Modification de E91202
X110 Z10               Axe Z interpolé de 60 à 10
M02
```

Cas de validation/invalidation avec "ARUS"

Programme original

```
%10
E91202=1
X10 Z10
X40 Z40
X80 Z30
X110 Z10
M02
```

Représentation schématique



REMARQUE Les deux signaux peuvent apparaître simultanément.

9.2 Mise en œuvre de la fonction N/M AUTO

Lors de la mise en œuvre, un axe physique xx peut être déclaré non interpolable :
- "à la volée",
- "machine arrêtée" (c'est à dire voyant "ARUS" allumé ou voyant CYCLE éteint).

Le programme automate doit autoriser et/ou confirmer l'état N/M AUTO en effectuant %W2.1=1. L'axe physique xx est alors un axe non interpolé "déplaçable" par manipulateur ou manivelle.

%R2.1 (E_NMAUTO) est le compte rendu de la CN. Ce bit n'est accessible qu'en lecture et passe à 1 dès que la fonction N/M AUTO est valide, c'est à dire qu'au moins un axe est déplaçable en NMA.

%W802.B (potentiomètre du groupe 8) module la vitesse de l'axe NMA avec manipulateurs.

%W4.1 (VITMAN 1) et %W4.2 (VITMAN 2) sont pris en compte comme en manuel. Ces bits définissent la vitesse de déplacement lorsque la manivelle ou les manipulateurs sont utilisés.

Exemple de mise en œuvre

E91202=1 est programmé.

Suivant l'état de %W2.1 (C_NMAUTO) au moment du passage à 1 de E91202, deux cas sont possibles.

Cas 1 : axe déclaré non interpolable "à la volée"

Si l'automate a préalablement autorisé la fonction N/M AUTO :
Au moment de l'écriture du paramètre E91202 la variable %W2.1 (C_NMAUTO) était à 1 ; l'axe physique 2 est immédiatement mis dans l'état non interpolé ; un compte rendu est disponible pour l'automate et %R2.1 (E_NMAUTO) passe à 1.

Cas 2 : axe déclaré non interpolable "machine arrêtée"

Si l'automate n'a pas encore autorisé la fonction N/M AUTO :
Au moment de l'écriture du paramètre E91202 la variable %W2.1 (C_NMAUTO) était à 0 ; l'état de l'axe physique 2 reste inchangé ; un compte rendu est disponible pour l'automate et %R2.1 (E_NMAUTO) reste à 0.

Si après une non autorisation, l'automate autorise le N/M AUTO :
La variable %W2.1 (C_NMAUTO) passe donc à 1 ; l'état de l'axe physique 2 reste encore inchangé jusqu'au prochain arrêt de la machine.
Dès que l'état " machine arrêtée" intervient (voyant "ARUS" allumé et "CYCLE" éteint) l'axe physique 2 est mis dans l'état non interpolé et %R2.1 (E_NMAUTO) passe à 1.

9.3 Mode d'emploi après validation N/M AUTO

Le type de "JOG" définit si l'axe NMA sera déplacé :
- soit par les manipulateurs d'axe (en illimité exclusivement),
- soit par la manivelle.

Particularités liées aux modes

La fonction N/M AUTO reste valide dans tous les modes, mais ne peut être mise en œuvre que dans les modes suivants :
- continu "CONT",
- rapide "RAP",
- immédiat "IMD",
- intervention "INTERV",
- séquentiel "SEQ"
- recherche du numéro de séquence "RNS" (voir remarque ci-après).

REMARQUES *En RNS, seuls les axes définis comme interpolables sont concernés par le rappel d'axes.*
Il est possible de passer d'un mode à l'autre, mais dans tous les cas ce changement doit être effecté axe NMA arrêté.

9.3.1 Utilisation des manipulateurs

Rappel : Seul le "JOG" illimité est autorisé avec les manipulateurs.

L'axe MNA est entraîné dans le sens positif ou négatif tant que le bouton manipulateur est à 1.

La vitesse de déplacement est définie par la position de VITMAN 1 et VITMAN 2 (sélection de la vitesse de "JOG" en mode manuel) modulée par le potentiomètre (%W802.B).

9.3.1.1 Cas de blocage d'un axe alors que E912xx est à 1

Les raisons pouvant expliquer qu'un axe est bloqué sont les suivantes (même si E912xx=1) :
- potentiomètre du groupe 8 à l'état 0 (%W802.B)
- autorisation d'avance à 0,
- l'axe NMA ou un axe entraîné (si RTCP actif) en butée sur fin de course,
- mise à 1 du paramètre E912xx a été réalisée alors que %W2.1 (C_NMAUTO) était à 0. Il est donc nécessaire de générer soit un "ARUS" ou de programmer la fonction M00 pour la prise en compte.

9.3.2 Utilisation de la manivelle

9.3.2.1 Affectation d'axe

Afin qu'un axe NMA soit entraîné par manivelle, il est nécessaire que l'automate affecte cet axe non interpolé à la manivelle :

[%W1x.B := Nom de l'axe non interpolé (x= A, B, C, ou D)]

autorise les commandes d'avance des axes dans le (ou les) sens souhaité(s) :

[%W6.0 à %WD.7 := 1]

Si la machine comporte plusieurs manivelles, celles-ci doivent être déclarées dans le paramètre machine P14 et présentes dans le système. Par contre les manivelles présentes et non utilisées doivent être mises à -1.

9.3.2.2 Exemple d'utilisation avec manivelle

Trois manivelles sont présentes dans le système :
- l'axe 0 est affecté à la manivelle 0, sens + et -
- l'axe 4 est affecté à la manivelle 1, sens + et -
- aucun axe n'est affecté à la manivelle 2.

Affectation des axes aux manivelles :
- %W1A.B =0
- %W1B.B =4
- %W1C.B =-1

Autorisation de sens + et/ou - :
- %W9.0 =1 %WD.0 =1
- %W9.4 =1 %WD.4 =1

9.3.2.3 Déplacements avec manivelle

L'axe NMA est entraîné en tournant la manivelle correspondante ; les incréments seront éventuellement appliqués au diamètre.

La vitesse de l'axe est directement déduite de la vitesse de la manivelle en tenant compte du paramètre machine P13 et des variables VITMAN 1 et VITMAN 2.

Les positions entre la manivelle et l'axe NMA sont respectées en fin de mouvement sauf dans les cas suivants :
- si la vitesse a été écrêtée à la valeur max de l'axe NMA,
- si les demandes d'incréments ont entraînés l'axe NMA ou un axe entraîné si RTCP actif au delà des fins de course,
- si une manivelle est active ; le mouvement des autres manivelles est ignoré.

REMARQUE *Il est possible d'utiliser plusieurs manivelles, chacune se voyant affecter un axe non interpolé, à condition de ne pas être actionnées simultanément. Le système attend la fin du mouvement généré par une manivelle, avant de prendre en compte l'autre. Toutefois si une manivelle est active, le déplacement des autres manivelles est ignoré et perdu.*

9.3.2.4 Cas de blocage d'un axe alors que E912xx est à 1

Les raisons pouvant expliquer qu'un axe est bloqué sont les suivantes (même si E912xx=1) :

- variable %W2.1 à l'état 0 (C_NMAUTO),
- la manivelle n'est pas déclarée, n'a pas été reconnue ou certaines valeurs affectées aux paramètres machine sont incohérentes (voir P12, P13, P14),
- l'axe n'a pas été affecté à la manivelle utilisée (voir %W1x.B),
- les commandes d'avance et de sens de l'axe sont à O (voir %W6.0 à %WD.7),
- l'axe NMA ou un axe entraîné si RTCP est actif est en butée fin de course,
- la mise à 1 du E912xx a été réalisée alors que %2.1 est à 0 (C_NMAUTO), il faut soit générer un "ARUS", soit programmer un M0 pour la prise en compte,
- un axe était déjà entrainé par une autre manivelle.

9.4 Arrêt-Reprise en N/M AUTO

Afin de dévalider un axe précédemment déclaré non interpolé, trois possibilités sont utilisables.

Possibilité 1

Le programme pièce effectue : E912xx=0 ; la fonction N/M AUTO est toujours valide et seul l'axe xx redevient interpolé.
%R2.1 (E_NMAUTO) reste à 1.
Si l'axe NMA est en cours de déplacement (par manivelle ou manipulateurs) l'exécution du programme pourra reprendre une fois l'axe arrêté.

Possibilité 2

Le programme automate met %W2.1 (C_NMAUTO) à 0 et le voyant "CYCLE" est éteint ; la fonction N/M AUTO est dévalidée totalement.
%R2.1 (E_NMAUTO) passe à 0.
Il y a émission de l'erreur 107 si le bloc suivant définit un cercle. Cependant les E912xx sont conservés en l'état, mais deviennent inefficaces.

Dans le cas ou la fonction N/M AUTO devait être à nouveau validée , il n'est pas nécessaire de faire repasser explicitement les E912xx à 0 puis à 1.
Lorsque le programme automate fait repasser : %W2.1 (C_NMAUTO) de 0 à 1, dès que le voyant "CYCLE" est éteint et/ou "ARUS" actif, la fonction N/M AUTO est revalidée immédiatement et :
- les E912xx programmés redeviennent efficaces,
- le %R2.1(E_NMAUTO) repasse à 1.

Possibilité 3

Le programme automate met: %W2.1 (C_NMAUTO) à 0 ; alors que la machine est en "ARUS", "le voyant "RAPAX" s'allume et le rappel des axes NMA doit être effectué avant de pouvoir poursuivre le programme pièce.
La fonction N/M AUTO est dévalidée totalement.
%R2.1 (E_NMAUTO) passe à 0.
Cependant les E912xx sont désormais conservés en l'état mais deviennent inefficace.

Dans le cas ou la fonction N/M AUTO devait être à nouveau validée , il n'est pas nécessaire de faire repasser explicitement les E912xx à 0 puis à 1.
Lorsque le programme automate fait repasser : %W2.1 (C_NMAUTO) de 0 à 1, dès que le voyant "CYCLE" est éteint et/ou "ARUS" actif, la fonction N/M AUTO est revalidée immédiatement et :
- les E912xx programmés redeviennent efficaces,
- le %R2.1(E_NMAUTO) repasse à 1.

REMARQUE *A chaque RAZ tous les axes reviennent à l'état "interpolé "(E912xx=0).*

9.5 Contrôles réalisés en N/M AUTO

Les contrôles réalisés en N/M AUTO sont les suivants :
- contrôle d'accélération,
- contrôle de vitesse,
- contrôle des courses,
- contrôles divers.

9.5.1 Contrôles d'accélération

En cours de N/M AUTO, l'accélération maximale tolérée est l'accélération maximum de l'axe NMA correspondant (voir paramètre machine P32 (mot pair) : F pour vitesse de travail). Si cette accélération est dépassée, la vitesse de l'axe NMA sera limitée.

Si la fonction RTCP est active, une seconde limitation est apportée afin de ne pas dépasser les accélérations maximales applicables aux vitesses de travail des axes entraînés (voir P32).

Rappel : si un axe menant de RTCP est déclaré non interpolé, la variation des vitesses d'interpolation est préparée compte tenu d'une accélération moitié moindre que l'accélération maximale tolérée.

9.5.2 Contrôles de vitesse

La vitesse de l'axe NMA actionné par manivelle est la vitesse définie dans le paramètre machine P31. VITMAN 1 et VITMAN 2 sont pris en compte comme en manuel.

Si la fonction RTCP est active une seconde limitation est apportée afin de ne pas dépasser les vitesses maximales autorisées des axes entraînés (voir P30).

Rappel : les vitesses d'interpolation peuvent être modulées par les valeurs affectées aux paramètres E7x101.

9.5.3 Contrôles des courses

Le test de surcourse est effectué conformément à la définition du paramètre machine P17. L'approche des fins de courses de l'axe NMA provoque son arrêt.

Si la fonction RTCP es active et qu'un axe entraîné approche des fins de courses, un "ARUS" est automatiquement généré pour arrêter l'interpolateur.

Le dégagement des fins de courses est possible à l'aide des manipulateurs ou de la manivelle entraînant l'axe NMA.

Quand l'utilisateur le souhaite, le programme peut reprendre en appuyant sur le départ "CYCLE".

9.5.4 Contrôles divers

Aucun test de conflit avec le G12 (survitesse) n'est réalisé. La prise d'origine "POM" des axes NMA doit être effectuée.

Annexe A Tableau récapitulatif des commandes de programmation structurée

EXIT : Sortie de boucle (Voir 1.2.6)

Syntaxe :

EXIT

FOR : Boucles avec variable de contrôle (Voir 1.2.5)

Syntaxe :

FOR (variable) = (expression 1) TO/DOWNTO (expression 2) BY (valeur) DO
(instructions)
ENDF

IF : Conditions d'exécution d'instructions (Voir 1.2.2)

Syntaxe :

IF (condition) THEN
(instructions 1)
ELSE
(instructions 2)
ENDI

REPEAT : Boucles «répéter jusqu'à» (Voir 1.2.3)

Syntaxe :

REPEAT
(instructions)
UNTIL (condition)

WHILE : Boucles «répéter tant que» (Voir 1.2.4)

Syntaxe :

WHILE (condition) **DO**
(instructions)
ENDW

Annexe B Tableau récapitulatif des commandes de gestion des variables symboliques

BUILD : Création d'un tableau pour rangement des trajectoires d'un profil (Voir 4.2.1)

Syntaxe générale :

BUILD [TAB(G / X / Y / I / J,NB)] H.. N.. +n N..+n

BCLR : Invalidation de la programmation de fonctions M et/ou d'un ou plusieurs axes. Suppression des bits de [•IBE0(i)] et [•IBE1(i)] (Voir 4.2.5)

Syntaxe :

BCLR [•BMxx] / [•IBX(i)] / [•IBE0(i)] / [•IBE1(i)]

BSET : Validation de la programmation de fonctions M et/ou d'un ou plusieurs axes. Positionnement des bits de [•IBE0(i)] et [•IBE1(i)] (Voir 4.2.5)

Syntaxe :

BSET [•BMxx] / [•IBX(i)] / [•IBE0(i)] / [•IBE1(i)]

CUT : Elimination des gorges ou parties de gorges situées en deça de l'angle de dépouille de l'outil (Voir 4.2.4)

Syntaxe :

CUT * [Pa(7,Nb)] / Angle

MOVE : Copie de la totalité ou partie d'un tableau dans un autre tableau (Voir 4.2.8)

Syntaxe générale

MOVE [Pj(nj,mj)],mj1,mj2 = [Pi(ni,mi)],mi1,mi2 / j1=i1 / j2=i2 /jn=in

---

P.BUILD : Création d'un tableau pour rangement des cotes du plan d'interpolation d'un profil (Voir 4.2.2)

Syntaxe générale :

P.BUILD [TAB(7,NB)] H.. N.. +n N..+n

R.OFF : Déport normal d'un profil ouvert (Voir 4.2.3)

Syntaxe :

R.OFF [Pa(7,Nb)] / ±1 / [R]

SAVE : Mise à disposition du programme principal et des sous programmes d'une liste de variables symboliques déclarées dans un sous programme quelconque (Voir 4.2.7)

Syntaxe :

SAVE [Symb1] / [Symb2] ...

SEARCH : Recherche de la présence ou non d'une variable symbolique dans la pile (Voir 4.2.6)

Syntaxe :

SEARCH [Symb] N..

Annexe C Tableau récapitulatif des symboles d'accès à l'état programme

Symboles d'accès aux données du bloc courant ou du bloc précédent

Seuls les symboles d'accès aux données du bloc courant sont répertoriés dans le tableau. Les symboles d'accès aux données du bloc précédent qui n'y figurent pas sont exactement identiques à ceux du bloc courant, mais sont précédés de deux points décimaux au lieu d'un seul, par exemple : bloc courant [•BGxx], bloc précédent [••BGxx] etc ...

C.1 Adressage des fonctions G et M

Symboles	Voir	Désignation
[•BGxx]	2.2.1.1	Adressage des fonctions G
[•BMxx]	2.2.1.2	Adressage des fonctions M

C.2 Adressage d'une liste de bits

Symboles	Voir	Désignation
[•IBI(i)]	2.2.1.3	Liste des arguments I, J et K programmés dans le bloc courant
[•IBP(i)]	2.2.1.3	Liste des arguments P, Q et R programmés dans le bloc courant
[•IBX(i)]	2.2.1.3	Liste des axes programmés dans le bloc courant
[•IBX1(i)]	2.2.1.3	Liste des axes programmés depuis le début du programme jusqu'au bloc courant
[•IBX2(i)]	2.2.1.3	Liste mémorisant les derniers axes programmés X Y Z ou U V W
[•IBXM(i)]	2.2.1.3	Etat «miroir» ou «non miroir» sur les axes

C.3 Adressage d'une valeur

Symboles	Voir	Désignation
[•RD]	2.2.2.1	Numéro du correcteur d'outil
[•RDX]	2.2.2.1	Orientation de l'axe de l'outil (G16)
[•REC]	2.2.2.1	Valeur de l'indexation de broche
[•RED]	2.2.2.1	Valeur du décalage angulaire
[•RF]	2.2.2.1	Valeur de la vitesse d'avance
[•RG4]	2.2.2.1	Valeur de la temporisation programmée (G04 F..)
[•RG80]	2.2.2.1	Numéro de la fonction appelante dans appel de sous programme par fonction G

Symbole	Voir	Désignation
[•RN]	2.2.2.1	Numéro de la dernière séquence (bloc) rencontrée
[•RNC]	2.2.2.1	Valeur de la fonction NC
[•RS]	2.2.2.1	Valeur de la vitesse de broche
[•RT]	2.2.2.1	Numéro d'outil
[•RXH]	2.2.2.1	Rang d'imbrication du sous programme courant

C.4 Adressage d'une liste de valeurs

Symboles	Voir	Désignation
•IRDI(i)]	2.2.2.2	Valeur des décalages angulaires programmés (G59 I.. J.. K..)
[•IRH(i)]	2.2.2.2	Numéros de programmes ou sous programmes courants
[•IRI(i)]	2.2.2.2	Valeurs des arguments I, J et K
[•IRP(i)]	2.2.2.2	Valeurs des arguments P, Q et R
[•IRTX(i)]	2.2.2.2	Valeurs des décalages programmés sur les axes
[•IRX(i)]	2.2.2.2	Valeurs des cotes programmées sur les axes

Annexe D Tableau récapitulatif des symboles de rangement dans les variables L900 à L951

D.1 Symboles de rangement dans les variables L900 à L925

Symboles	Voir	Variables L	Adresses
[•IBE0(6)]	3.2	L905	F
[•IBE0(8)]	3.2	L907	H
[•IBE0(14)]	3.2	L913	N
[•IBE0(15)]	3.2	L914	N
[•IBE0(19)]	3.2	L918	S
[•IBE0(20)]	3.2	L919	T

D.2 Symboles de rangement dans les variables L926 à L951

Certains symboles figurant dans le tableau ne sont pas utilisés.

Symboles	Voir	Variables	Adresses
[•IBE1(1)]	3.3	L926	EA
[•IBE1(2)]	3.3	L927	EB
[•IBE1(3)]	3.3	L928	EC
[•IBE1(4)]	3.3	L929	ED
[•IBE1(5)]	3.3	L930	EE
[•IBE1(6)]	3.3	L931	EF
[•IBE1(7)]	3.3	L932	EG
[•IBE1(8)]	3.3	L933	EH
[•IBE1(9)]	3.3	L934	EI
[•IBE1(10)]	3.3	L935	EJ
[•IBE1(11)]	3.3	L936	EK
[•IBE1(12)]	3.3	L937	EL
[•IBE1(13)]	3.3	L938	EM
[•IBE1(14)]	3.3	L939	EN
[•IBE1(15)]	3.3	L940	EO

Symboles	Voir	Variables	Adresses
[.IBE1(16)]	3.3	L941	EP
[.IBE1(17)]	3.3	L942	EQ
[.IBE1(18)]	3.3	L943	ER
[.IBE1(19)]	3.3	L944	ES
[.IBE1(20)]	3.3	L945	ET
[.IBE1(21)]	3.3	L946	EU
[.IBE1(22)]	3.3	L947	EV
[.IBE1(23)]	3.3	L948	EW
[.IBE1(24)]	3.3	L949	EX
[.IBE1(25)]	3.3	L950	EY
[.IBE1(26)]	3.3	L951	EZ

Index

Symboles

A

B

C

D

E

F

G

I

L

M

N

P

R

S

T

U

V

W