Macro B
Machines ISO

Edition N°4 06.06.2017
Version 1.0/MC 0703

Contenu

1 Rendez vos programmes intelligents !!

1.1 Le "Macro B" c'est quoi ?

Le macro B est un langage de programmation paramétré sur les commandes numériques FANUC.
Toutes vos machines ISO de dernière génération disposent gratuitement de ce langage.
Il vous permet de rendre vos programmes intelligents. Nous verrons dans le chapitre 3 des exemples concrets d'utilisation.
Et surtout ne vous fiez pas aux apparences, à la première lecture cela peut paraître compliqué mais en réalité il est vraiment très simple de s'en servir dans vos programmes de tous les jours.

1.2 Le saviez-vous ?

Le macro B est un langage de programmation simplissime, qui offre une multitude de possibilités.
Par exemple, saviez-vous que la majorité des macros Tornos de votre machine ISO sont composés de plus de 20'000 lignes de code utilisant ce même langage ?

2 Instructions de base du macro B

2.1 Les variables

Pour pouvoir rendre un programme intelligent il faut pouvoir remplacer des valeurs par des variables.

Une variable est une donnée à laquelle on peut attribuer une valeur.
La syntaxe d'une variable est identifiable par un "#" et son numéro d'identification.

Exemple de variable: #153
Exemple d'attribution d'une valeur à une variable: #153 = 12.4 *(ainsi la variable #153 contient la valeur 12.4)*

Une fois que vos variables contiennent une valeur, elles peuvent par exemple être additionnées, multipliées, être utilisées comme position, comme vitesse, comme avance ou encore utilisées dans une expression conditionnelle.

Voyons désormais quelles sont les variables que vous pouvez utiliser.

La variable nulle:

La variable nulle est une variable qui ne contient jamais aucune valeur.

Il s'agit de la variable: **#0**

Les variables locales:

Les variables locales, sont des variables qui se réinitialisent à "nulle" dès le moment où l'on quitte le programme dans lequel une valeur leur a été attribuée.

Il s'agit des variables: **#1 - #33**

Les variables globales par canal:

Les variables globales par canal sont des variables qui ne se réinitialisent pas à "nulle" lorsque l'on quitte le programme dans lequel une valeur leur a été attribuée.
Le fait qu'elles soient par canal signifie que lorsque l'on attribue une valeur à une variable, la variable va contenir cette valeur uniquement dans le canal dans lequel la variable a eu son attribution.

Il s'agit des variables: **#150 - #199**

Les variables globales communes:

Les variables globales communes sont des variables qui ne se réinitialisent pas à "nulle" lorsque l'on quitte le programme dans lequel une valeur leur a été attribuée.
Le fait qu'elles soient communes signifie que lorsque l'on attribue une valeur à une variable, la variable va contenir cette valeur dans tous les canaux de la machine.

Il s'agit des variables: **#600 - #699**

Les variables systèmes:

Les variables systèmes peuvent être utilisées pour lire et écrire des données de la commande numérique, comme par exemple des variables de compensation d'outil, des données de position d'axes, etc.
Pour plus d'informations à propos des variables systèmes, il est possible de se reporter aux instructions FANUC.

Il s'agit des variables: **> #1000**

2.2 Les opérations arithmétiques et logiques

Les opérations arithmétiques:

Les opérations arithmétiques les plus souvent utilisées sont :

Addition	"+"
Soustraction	"-"
Multiplication	"*"
Division	"/"

Exemple d'utilisation:
#603 = [#601 + #602] / 4

Les fonctions:

Les fonctions les plus souvent utilisées sont :

Sinus	"SIN[#...]"
Cosinus	"COS[#...]"
Tangente	"TAN[#...]"
Racine carrée	"SQRT[#...]"
Valeur absolue	"ABS[#...]"
Puissance	"POW[#..., #...]"
Arrondi au nombre entier inférieur	"FIX[#...]"
Arrondi la valeur	"ROUND[#...]"

Exemple d'utilisation:
#603 = COS[#602]

NB : l'unité d'angle utilisée est le degré. Par exemple : 90 degrés et 30 minutes s'écrit 90.5 degrés.

Les opérateurs de relations:

Les opérateurs de relations permettent de comparer deux variables dans une expression conditionnelle.

Les opérateurs de relations sont :

Egal à	"EQ"
Différent de	"NE"
Supérieur à	"GT"
Supérieur ou égal à	"GE"
Inférieur à	"LT"
Inférieur ou égal à	"LE"

Les opérations logiques:

Les opérations logiques permettent de tester plusieurs conditions dans une seule expression conditionnelle.

Les opérations logiques les plus souvent utilisées sont :

ET logique	"AND"
OU logique	"OR"

2.3 Les branchements et répétitions

L'instruction inconditionnelle "GOTO":

L'instruction "N*n*" en début de ligne permet de renseigner le numéro de bloc.
L'instruction "GOTO*n*" est très simple à comprendre, GOTO 5 signifie: sauter jusqu'au bloc N5.

Exemple d'utilisation:

```
...
GOTO 5
G0 X2 T35
M103 S200          } Cette partie de programme n'est pas exécutée par la CN
G0 Z6
N5 G0 X3 T26
...
```

L'instruction conditionnelle "IF" - "THEN":

L'instruction "IF" signifie: uniquement **si** l'expression conditionnelle est satisfaite on exécute se qui suit.
L'instruction "THEN" signifie **alors**.

Exemple d'utilisation:

IF [#600 EQ #601] THEN #602 = 18 *Si les valeurs de #600 et de #601 sont identiques alors #602 prend la valeur 18*

NB: le "EQ" peut être remplacé par d'autres opérateurs de relations

---

L'instruction conditionnelle "IF" - "GOTO":

L'instruction "IF" signifie: uniquement **si** l'expression conditionnelle est satisfaite on exécute se qui suit.
L'instruction "GOTO" signifie **aller à**.

Exemple d'utilisation:

```
IF [#600 EQ #601] GOTO 10   Si les valeurs de #600 et de #601 sont identiques alors on saute au bloc N10
G0 X2 T35
M103 S200
G0 Z6                       } Cette partie de programme est exécutée par la CN uniquement si #600 est différent de #601
...
...
N10 G0 X3 T26
```

NB: le "EQ" peut être remplacé par d'autres opérateurs de relations

---

L'instruction de répétition "WHILE":

L'instruction "WHILE" signifie **boucle**.
L'instruction "DO" signifie **faire**.

Exemple d'utilisation:

```
WHILE [#600 GT #601] DO1
G0 X2 T35
M103 S200
G0 Z6
...
...
#601 = #601 + 1
END1
```

Cette partie de programme est exécutée par la CN tant que la valeur de #600 est plus grande que #601

(L'incrémentation du #601 permettra la sortie de la boucle)

NB: le "GT" peut être remplacé par d'autres opérateurs de relations

Exemple de boucles imbriquées:

```
WHILE[...] DO1
...
...
    WHILE[...] DO2
    ...
    ...
        WHILE[...] DO3
        ...
        ...
        ...
        END3
    ...
    ...
    END2
...
...
END1
```

NB: il est possible d'imbriquer jusqu'à 3 boucles les unes dans les autres

2.4 Les sous-programmes

Les avantages d'un sous-programme:

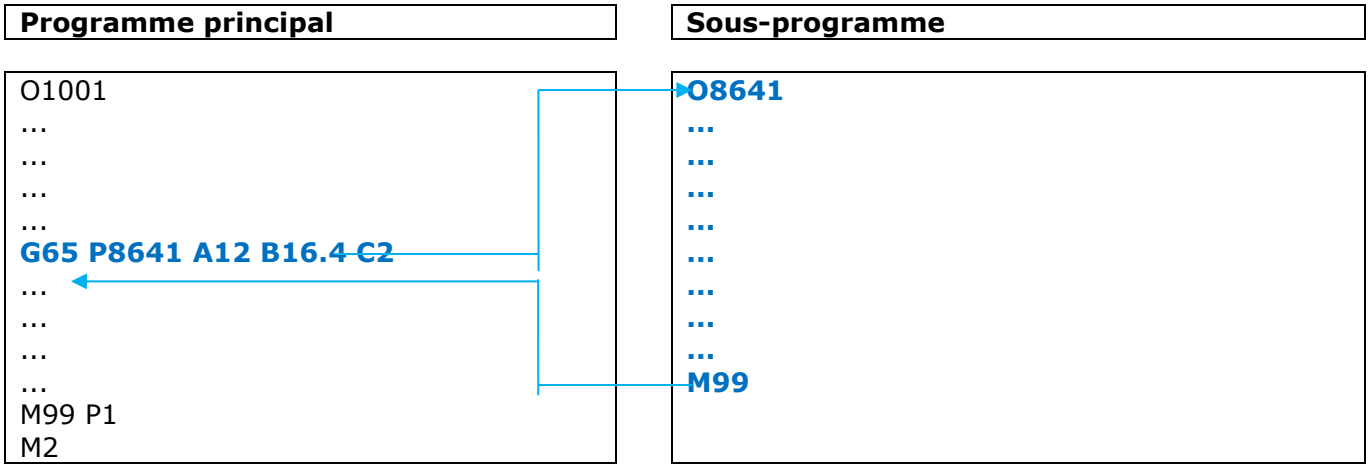
Le principal avantage d'un sous-programme réside dans le fait de pouvoir l'appeler à plusieurs reprises sans avoir à le recoder.
Il peut donc être appelé à plusieurs reprises depuis un seul programme, mais également être appelé depuis plusieurs programmes différents.

Le second avantage d'un sous-programme est de pouvoir lui passer des arguments de paramétrage.

L'appel d'un sous-programme:

Un sous-programme se code, se nomme et se transfère sur la machine de la même façon qu'un programme principal.
Exemple de nom "O8641".
L'appel d'un sous-programme se fait par la fonction G65 P*n* {A*n* B*n* C*n* ...}.

Exemple:
G65 P8641 A12 B16.4 C2 *Le sous-programme O8641 est appelé et les argument A, B, C lui sont transmis.*



Les arguments d'un sous-programme:

Le passage d'arguments de paramétrage à un sous-programme est optionnel.
Si l'on souhaite en utiliser, les valeurs des arguments sont transférées automatiquement dans les variables locales selon le tableau ci-dessous:

Adresse	Numéro variables		Adresse	Numéro variables		Adresse	Numéro variables
A	#1		I	#4		T	#20
B	#2		J	#5		U	#21
C	#3		K	#6		V	#22
D	#7		M	#13		W	#23
E	#8		Q	#17		X	#24
F	#9		R	#18		Y	#25
H	#11		S	#19		Z	#26

2.5 L'affichage d'une alarme

Il est également possible d'afficher une alarme sur la CN de la façon suivante:

```
#3000 = 1  (ALARME)
```

Lorsque la CN arrivera sur ce bloc, l'alarme "MC3001 ALARME" s'affichera et bloquera l'interprétation.

Exemple d'utilisation:

```
IF [#600 GE 0] GOTO 10
      #3000 = 2  (ERREUR VALEUR NE
N10
```

Si #600 est plus petit que 0 l'alarme "MC3002 ERREUR VALEUR NEGATIVE" s'affichera et bloquera l'interprétation.

2.6 L'affichage d'un message

Il est possible d'afficher un message sur la CN de la façon suivante:

```
#3006 = 1  (MESSAGE)
```

Lorsque la CN arrivera sur ce bloc, le message "MESSAGE" s'affichera mais ne bloquera pas l'interprétation.

Exemple d'utilisation:

```
IF [#600 GE 0] GOTO 10
      #3006 = 2  (ATTENTION VALEUR NE
N10
```

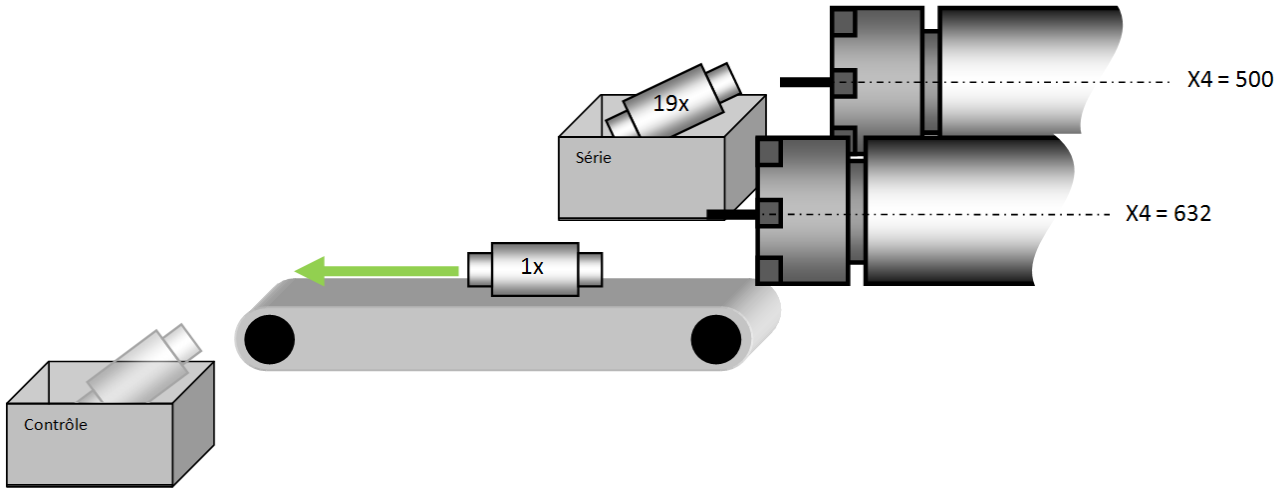
Si #600 est plus petit que 0 le message "ATTENTION VALEUR NEGATIVE" s'affichera mais ne bloquera pas l'interprétation.

3 Exemples concrets d'utilisation

3.1 Echantillonnage de pièces

Imaginons que vous produisiez une série de pièces qui nécessite un contrôle d'une pièce tous les 20 cycles.
Voyons comment le macro B peut nous aider.

Principe:
Le principe réside dans le fait d'éjecter la pièce 19 fois dans le récupérateur à l'intérieur de la machine pour une seule fois sur le convoyeur de pièces afin qu'elle puisse être contrôlée hors machine.

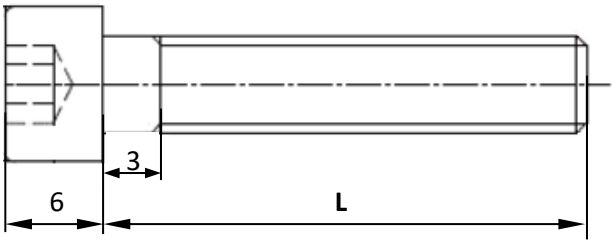


Programmation:

Canal contre-broche

```
#600 = 0                          (INITIALISATION COMPTEUR DE CYCLES)
#601 = 20                         (PIECE A CONTROLER TOUS LES COMBIEN DE CYCLES)
#602 = 500                        (POSITION X4 EXTRACTION SERIE)
#603 = 632                        (POSITION X4 EXTRACTION POUR CONTROLE)
...
...
N1 M120                           (DEBUT DE BOUCLE)
IF [#600 EQ #601] THEN            (AU CYCLE 20 REMISE A 0 DU COMPTEUR)
#600 = 0                          (INCREMENTATION COMPTEUR)
#600 = #600 + 1
...
...
IF [#600 EQ #601] GOTO 10         (CYCLE DIFFERENT DU 20EME ON EJECTE A X500)
G53 X500
GOTO 11
N10                               (CYCLE NUMERO 20 ON EJECTE A X630)
G53 X632
N11                               (EJECTION)
M84
...
...                               (FIN DE BOUCLE)
M121
```

3.2 Famille de pièces



Imaginons que vous produisiez une gamme de vis.
Toutes les vis sont identiques à l'exception de leur longueur "L".
Il pourrait être intéressant de conserver un seul programme pour toutes les vis au lieu d'un programme par vis.

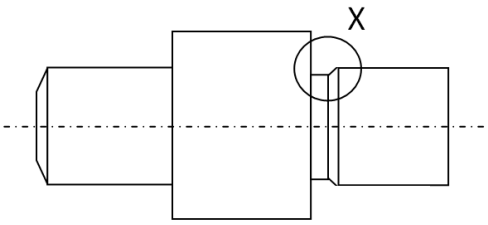
Canal 1

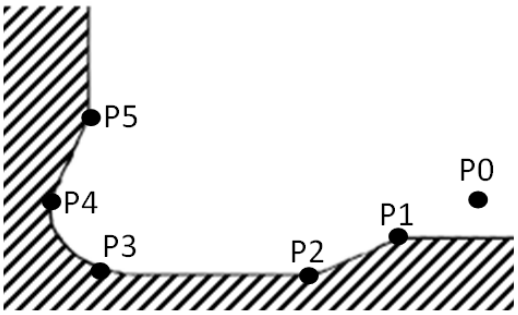
```
#600 = 53427                    (N° IDENTIFICATION DE VIS 53426, 53427, 53428, ...)
IF [#600 EQ 53426] GOTO 5
IF [#600 EQ 53427] GOTO 10
IF [#600 EQ 53428] GOTO 15
IF [#600 EQ 53429] GOTO 20
IF [#600 EQ 53430] GOTO 25
N5 #601 = 10                    (LONGUEUR "L" POUR VIS 53426)
GOTO 30
N10 #601 = 12                   (LONGUEUR "L" POUR VIS 53427)
GOTO 30
N15 #601 = 15                   (LONGUEUR "L" POUR VIS 53428)
GOTO 30
N20 #601 = 20                   (LONGUEUR "L" POUR VIS 53429)
GOTO 30
N25 #601 = 22                   (LONGUEUR "L" POUR VIS 53430)
N30
G800 A10 B[6+#601] C[#601+2]    (DONNEE DE CYCLE  B: LONGUEUR PIECE, C: DISTANCE
...                             PRISE DE PIECE)
...
N1 M120
...                             (DEBUT DE BOUCLE)
...
G0 X5 Z2 T12 D0
G1 Z-#601 F0.06
G1 X12                          (TOURNAGE DE LA PORTEE)
...
...
G78 P020060 Q500 R0.02
G78 X4.2 Z-[#601-3] R0 P4300    (FILETAGE)
Q900 F0.7                       (FILETAGE)
...
...
M121                            (FIN DE BOUCLE)
...
```

Ainsi il suffit de modifier le numéro d'identification de la vis sur la première ligne du programme pour changer de vis.

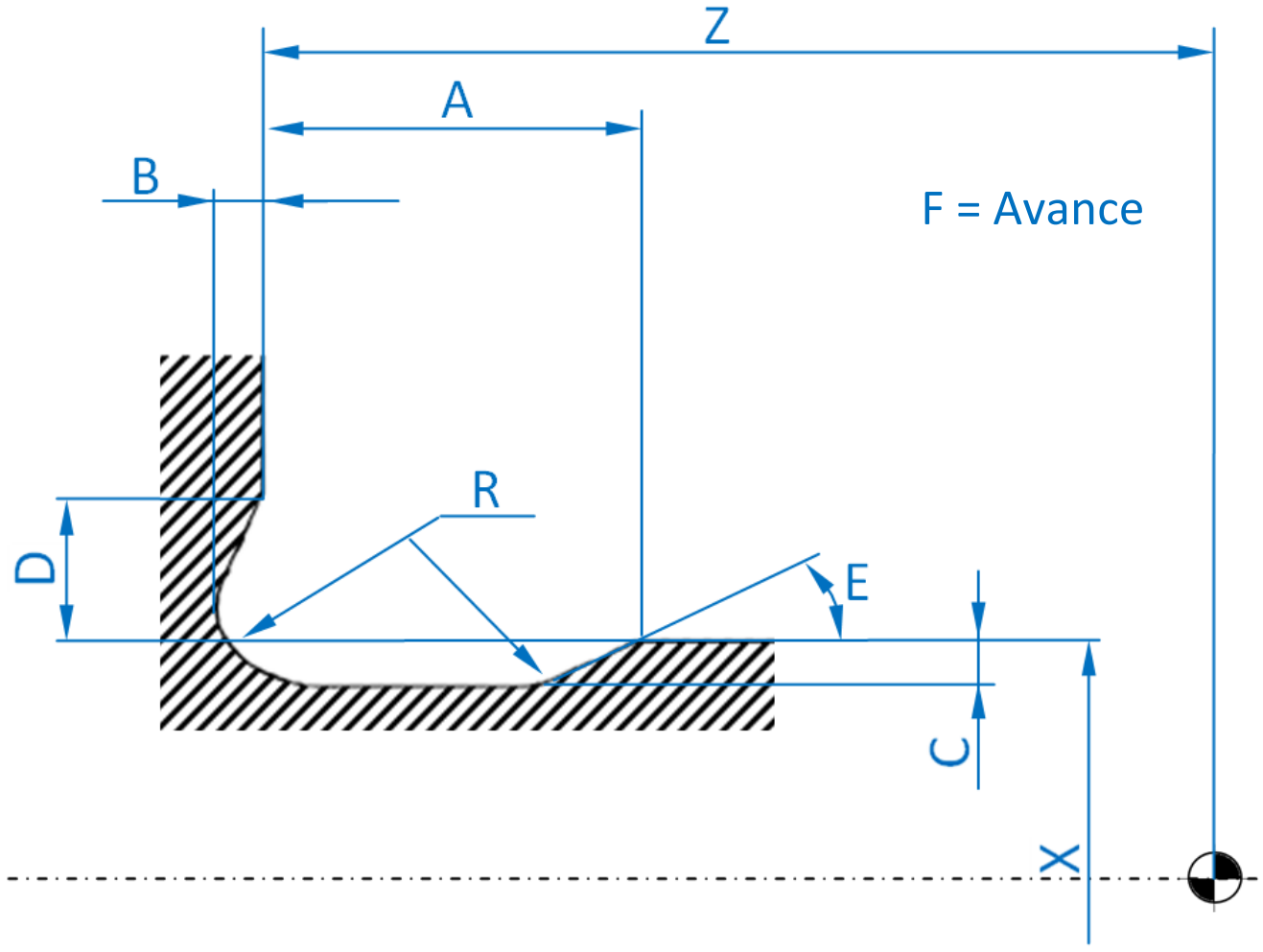
3.3 Votre propre macro d'usinage

Imaginons que vous soyez amenés à programmer régulièrement sur vos pièces des saignées sur lesquelles vous devez calculer plusieurs points. Pour vous simplifier la vie, vous pourriez créer votre propre macro d'usinage.





Programmation de la macro:



Programme principal
... **G65 P8512 A2 B0.1 C0.25 D1.15 E30 F0.05 R0.6 X12 Z36** (APPEL DU SOUS-PROGRAMME O8512) **...**

Sous-programme O8512 *(Macro d'usinage)*

```
(*** MEMORISATION ARGUMENTS ***)
#600 = #1                                   (ARGUMENT A)
#601 = #2                                   (ARGUMENT B)
#602 = #3                                   (ARGUMENT C)
#603 = #7                                   (ARGUMENT D)
#604 = #8                                   (ARGUMENT E)
#605 = #9                                   (ARGUMENT F)
#606 = #18                                  (ARGUMENT R)
#607 = #24                                  (ARGUMENT X)
#608 = #26                                  (ARGUMENT Z)
#609 = 2                                    (CONSTANTE DE SECURITE)

(*** CALCUL P0 ***)
#611 = #607 + #609                          (P0 X)
#612 = -[ABS[#608] - #600 -                 (P0 Z)
[[#609/2]/[TAN[#604]]]]
(*** CALCUL P1 ***)
#613 = #607                                 (P1 X)
#614 = -[ABS[#608] - #600]                  (P1 Z)

(*** CALCUL P2 ***)
#615 = #607 - [#602 * 2]                    (P2 X)
                                            (P2 Z)
#616 = -[ABS[#608] - #600 +
[#602/TAN[#604]]]
(*** CALCUL P3 ***)                         (P3 X)
#617 = #615                                 (P3 Z)
#618 = -[ABS[#608] + #601 - #606]

(*** CALCUL P4 ***)                         (P4 X)
#619 = #615 + [#606 * 2]                    (P4 Z)

#620 = -[ABS[#608] + #601]
                                            (P5 X)
(*** CALCUL P5 ***)                         (P5 Z)
#621 = #607 + [#603 * 2]

#622 = -[ABS[#608]]

(*** CODE ISO ***)                          (P0)
G90 G95                                     (P1)
G0 Y0                                       (P2)
G0 X#611 Z#612                              (P3)
G1 X#613 Z#614 F#605                        (P4)
G1 X#615 Z#616 ,R#606 F#605                 (P5)
G1 X#617 Z#618 F#605
G2 X#619 Z#620 I#606 K0 F#605               (SORTIE DU SOUS-PROGRAMME)
G1 X#621 Z#622 F#605

M99
```

3.4 Nettoyage discontinu de la machine

Imaginons que vous produisiez une série de pièces, sur laquelle il est nécessaire de nettoyer à fréquence régulière l'intérieur de votre machine pour évacuer les copeaux.
Voyons comment le macro B peut nous aider.

Principe:
Le principe consiste à nettoyer régulièrement vos outils de coupe par le biais d'une pompe haute pression mais sans faire tourner cette dernière en permanence.
L'avantage d'utiliser un tel procédé, est de diminuer bruit et consommation électrique dans votre atelier.
Dans l'exemple ci-dessous nous profitons également de nettoyer la pince contre-broche avec de l'air.

Canal contre-broche

```
#600 = 0                    (INITIALISATION COMPTEUR DE CYCLES)
#601 = 100                  (MACHINE A NETTOYER TOUS LES COMBIEN DE CYCLES)
...
N1 M120                     (DEBUT DE BOUCLE)
IF [#600 EQ #601] THEN      (AU CYCLE 100 REMISE A 0 DU COMPTEUR)
#600 = 0                    (INCREMENTATION COMPTEUR)
#600 = #600 + 1
...
M11                         (EJECTION DE LA PIECE DE LA PINCE CONTRE-BROCHE)
M84
...
...                         (TOUS LES 100 CYCLES ON EXECUTE LE CODE JUSQU'A N10)
IF [#600 NE #601] GOTO      (OUVERTURE VANNE 1)
10                          (ENCLENCHEMENT POMPE HAUTE PRESSION)
M532 M11 M1                 (NETTOYAGE SYSTEME OUTILS 1)
M532 M1 M1                  (FERMTURE VANNE 1)
G4 X4                       (OUVERTURE VANNE 2)
M532 M11 M0                 (NETTOYAGE SYSTEME OUTILS 2)
M532 M12 M1                 (FERMTURE VANNE 2)
G4 X4                       (OUVERTURE VANNE 3)
M532 M12 M0                 (NETTOYAGE SYSTEME OUTILS 3)
M532 M13 M1                 (FERMTURE VANNE 3)
G4 X4                       (DECLENCHEMENT POMPE HAUTE PRESSION)
M532 M13 M0                 (SOUFFLAGE CENTRE CONTRE-BROCHE 3 SEC. POUR
M532 M1 M0                  NETTOYAGE PINCE)
M841 M2 M3000
N10
...
...                         (FIN DE BOUCLE)
M121
...
```

3.5 Arrêt machine prédéfini

Imaginons que votre machine produise toute la semaine mais que vous devez arrêter sa production le samedi soir afin d'éviter qu'une usure d'outil produise des pièces hors tolérance jusqu'au lundi matin.
Il serrait probablement intéressant pour vous de ne pas avoir à retourner à l'usine le samedi soir pour arrêter les machines. Voyons comment le macro B peut nous aider.

Principe:
Le principe consiste à contrôler à chaque cycle la date et l'heure courante de la CN, et d'arrêter la machine lorsque la date et l'heure configurées sont atteintes.

Dans l'exemple ci-dessous nous arrêtons la machine
le : 24.06.2017 (#600)
à : 20:35:00 (#601)

```
Canal 1
#600 = 203500                    (HEURE DE L'ARRET: HEURE - MINUTE - SECONDE)
#601 = 20170624                  (DATE DE L'ARRET MACHINE: ANNEE - MOIS - JOUR)
...
...
N1 M120                          (DEBUT DE BOUCLE)
...
...
IF [#601 NE #3011] GOTO 10       (CONTROLE SI NOUS SOMMES LE JOUR DE L'ARRET
IF [#600 LT #3012] GOTO 10       MACHINE)
M105                             (CONTROLE QUE L'HEURE D'ARRET MACHINE SOIT
M405                             PASSEE)
M1105                            (ARRET DES BROCHES)
M9
M0
N10                              (ARRET DE L'HUILE)
M121                             (ARRET DU CYCLE)
...
                                 (FIN DE BOUCLE)
```

NB: La #3011 = variable système qui renseigne la date actuelle sur la CN (année, mois, jour)
La #3012 = variable système qui renseigne l'heure actuelle sur la CN (heure, minute, seconde)

4 Bon à savoir

4.1 Temps de cycle

Nous vous recommandons de coder en macro B tous ce qui est possible avant la boucle d'usinage *(avant M120)*. Cela permet de réduire au maximum la perte de temps de cycle liées au traitement des conditions et des calculs.

4.2 Formation Tornos

Il peut être intéressant de savoir que Tornos propose des formations de programmation paramétrée, afin que vous puissiez devenir un vrai spécialiste et utiliser au mieux toutes les possibilités de ce langage.

4.3 Instruction FANUC

L'instruction FANUC B-63944 explique l'intégralité des possibilités du langage.