11 - MACRO B

A função MACRO B é utilizada quando se deseja trabalhar na programação de peças complexas, “famílias de peças” e outras funções especiais tais como: operações aritméticas, desvios condicionais, contador, comandos de comparação, etc...

Este tipo de programação é feito através da atribuição de valores à variáveis.

Um programa pode invocar uma MACRO utilizando o comando G65 para a chamada de um sub-programa..

Exemplo:

PROGRAMA DE USINAGEM MACRO

```
O0001 (PROGRAMA PRINCIPAL)
G17 G21 G90 G94
  :
  :
G65 P9130 R50 L2 G02 X#1 Y#1 R#1
  :
  :
M30
```

```
O9130 (PROGRAMA MACRO)
:
#1=#18/2
G01 G42 X#1 Y#1 F300
:
:
M99
```

Explanação:

Quando definimos uma variável, especificamos um símbolo (#) seguido pelo numero da variável. Exemplo: #1

Uma expressão pode ser usada para especificar o número de uma variável, nesse caso, a expressão deve ser expressa entre colchetes.

Exemplo: # [#1+#2-12]

11.1 - TIPOS DE VARIÁVEIS

As variáveis são classificadas em 4 tipos:

a) #0 - Sempre nula, ou seja, nenhum valor pode ser atribuido para esta variável;

b) #1 - #33 - Variáveis locais. Podem apenas ser usadas em macro para carregar dados como resultado de operações. Quando o comando é inicializado, as variáveis locais são inicializadas sem valores (nulas).

Quando uma macro é invocada, valores podem ser atribuidos para a variáveis locais;

c) #100 - #149 (#199) / #500 - #531 (#999) - Variáveis comuns. Podem estar parcialmente entre diferentes programas Macros. Quando o comando é desligado, as variáveis #100 a #531 mantém o último valor que a elas foi atribuido. Dentro da gama de variáveis comuns ainda temos as variáveis #150 a #199 e #532 a #999 ;

d) #1000 - Variáveis de Sistema. São usadas para ler uma série de dados NC como: posição atual, valores de compensação de ferramenta, etc...

11.2 - GAMA DE VALORES PARA AS VARIÁVEIS

Variáveis locais e comuns podem ter valor = 0 ou um valor na seguinte faixa

-10^{47} a -10^{-29}
10^{-29} a 10^{47}

Se o resultado do calculo for invalido, ou seja, estiver fora desta faixa de valores, o alarme 111 será mostrado.

11.3 - OMISSÃO DO PONTO DECIMAL

Quando um valor de variável for definido em um programa, o ponto decimal pode ser omitido.

Exemplo: Quando #1=123 for definido, o valor real da variável #1 é 123.000

11.4 - REFERENCIANDO VARIÁVEIS

Para referenciar o valor de uma variável em um programa, especifique o endereço seguido pelo numero da variável. Quando uma expressão for usada para especificar uma variável, inclua a expressão entre colchetes.

Exemplo:

G01 X[#1+#2] F#3

Um valor de variável é automaticamente arredondado de acordo com o mínimo incremento do endereço

Exemplo:

Quando G0 X#1 é programado e o valor da #1 é 12,3456, se o CNC apresentar um valor mínimo de programação de 0,001 mm, o comando a ser executado será G0 X12,346.

Para reverter o sinal do valor de uma variável, programe o sinal menos (-).

Exemplo:

G0 X-#1

11.5 - OPERAÇÕES ARITMÉTICAS E OPERAÇÕES LÓGICAS

As operações listadas na tabela seguinte podem ser executadas com variáveis.

A expressão a direita da operação pode conter constantes e/ou variáveis combinadas por uma função ou operação.

As variáveis #J e #K podem ser substituídas por uma constante.

As variáveis da esquerda também podem ser substituídas por uma expressão.

11.5.1 - Tabela de operações aritméticas e operações lógicas

FUNÇÃO	FORMATO
DEFINICAO	#I=#J
SOMA	#I=#J+#K
DIFERENCA	#I=#J-#K
PRODUTO	#I=#J*#K
QUOCIENTE	#I=#J/#K
SENO	#I=SIN[#J]
COSSENO	#I=COS[#J]
TANGENTE	#I=TAN[#J]
ARCO TANGENTE	#I=ATAN[#J]/[#K]
RAIZ QUADRADA	#I=SQRT[#J]
VALOR ABSOLUTO	#I=ABS[#J]
ARREDONDAMENTO	#I=ROUND[#J]
ARREDONDAMENTO DOWN	#I=FIX[#J]
ARREDONDAMENTO UP	#I=FUP[#J]
OR	#I=#J OR #K
XOR	#I=#J XOR #K
AND	#I=#J AND #K
CONVERSAO DE BCD A BIN	#I=BIN[#J]
CONVERSAO DE BIN A BCD	#I=BCD[#J]

OBSERVAÇÃO: *Uma operação lógica se executa em números binários bit a bit.*

Explicação:

UNIDADES DE ÂNGULO - As unidades de ângulos usadas com as funções SIN, COS, TAN e ATAN são em graus.
Exemplo: 90°30’ = 90,5°.

FUNÇÃO ATAN - Após a FUNÇÃO ATAN, especificando o comprimento de dois lados separados por uma barra se obtém um resultado onde 0< resultado< 360.
Exemplo: Quando #1=ATAN[1]/[1], o valor da variável #1 é 135.

FUNÇÃO ARREDONDAMENTO - Quando se inclui uma função de arredondamento em uma operação aritmética ou lógica, a função ROUND arredonda a primeira casa decimal.
Exemplo: Quando se executa #1=ROUND[#2] onde a variável #2 contem o valor 1,2345, o valor para a variável #1 é 1.

A função de arredondamento aproxima o valor especificado segundo o incremento mínimo de entrada.
Exemplo: Um programa de furacão que realiza um movimento segundo os valores das variáveis #1 e #2 e logo retorna a posição inicial.

Supondo que o sistema apresente incrementos mínimos de 1/1000mm, a variável #1 contem o valor armazenado de 1,2347 e a variável #2 contem o valor armazenado de 2,3456.
Dai temos:

```
  G00 G91 X-#1          Movimento de 1,235mm
  G01 X-#2 F300     Movimento de 2,346mm
* G00 X[#1+#2]
```

Considerando que 1,2347+2,3456=3,5803, a distancia real de deslocamento será 3,580 e, desta forma, a ferramenta não retorna a posição inicial.
Para que este retorno ocorra deve-se programar:

```
* G0 X[ROUND[#1]+ROUND[#2]]
```

Exemplo sobre as funções FUP e FIX.
Suponha que #1=1,2 e #2=-1,2;
Quando o comando #3 = FUP[#1] é executado, o valor 2 e assinalado para a variável 3. Quando o comando #3 = FIX[#1] é executado, o valor 1 é assinalado para a variável 3. Quando o comando #3 = FUP[#2] é executado, o valor -2 é assinalado para a variável 3. Quando o comando #3 = FIX[#2] é executada, o valor -1 é assinalado para a variável 3.

11.6 - PRIORIDADES DE OPERAÇÕES

1 - Funções
2 - Operações como multiplicação e divisão (*,/,AND)
3 - Operações como adição e subtração (+,-,OR,XOR)

Exemplo:
#1=#2+#3*SIN[#4]

Primeira resolução SIN[#4].
Segunda resolução #3*SIN[#4].
Terceira resolução #2+#3*SIN[#4].

11.7 - NÍVEIS DE COLCHETES

Para modificar as ordens das operações deve-se usar colchete [].
Os colchetes podem ser usados em até 5 níveis, incluindo os colchetes usados para fechar a expressão. Quando um nível de 5 colchetes for ultrapassado um alarme 118 ocorrerá.

Exemplo:
#1=SIN[[[#2+#3]*#4+#5]*#6]

1ª operação [#2+#3]
2ª operação [#2+#3]*#4
3ª operação [[#2+#3]*#4+#5]
4ª operação [[#2+#3]*#4+#5]*#6
5ª operação SIN[[[#2+#3]*#4+#5]*#6]

11.8 - DESVIO E REPETIÇÃO

Em um programa o fluxo do controle pode modificar-se usando a declaração GOTO e a declaração IF de desvio e repetições.
Três tipos de operações são usadas:

1ª GOTO - desvio incondicional
2ª IF - desvio condicional: SE, ENTAO.
3ª WHILE - repetição: ENQUANTO.

11.8.1 - Desvio incondicional - GOTO

Executa um desvio para o número de seqüência N.
Quando se especifica um número de seqüência não compreendido entre 1 ate 99999, um alarme 128 será mostrado.
Também pode-se especificar um número de seqüência usando uma expressão.

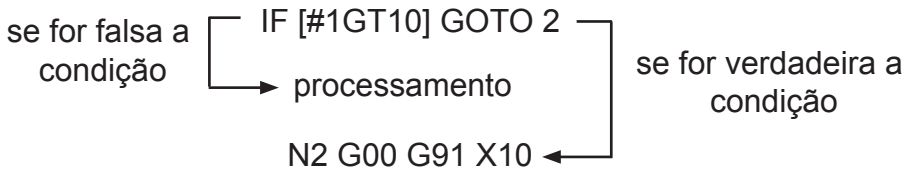
GOTO N N - número de seqüência (1 ate 99999)

Exemplo:
GOTO500; (desvia para o bloco N500)

11.8.2 - Desvio condicional - IF

Especifique uma expressão condicional depois de “If”. Se a expressão condicional for verdadeira executa-se um desvio para o numero de seqüência N. Se a expressão condicional for falsa executa-se o bloco seguinte.

Exemplo:
Se o valor da variável #1 for superior a 10, executa-se um desvio ao número de seqüência N2.



Explicações:

Expressão condicional - Uma expressão condicional deve incluir um operador colocado entre as variáveis ou entre uma variável e uma constante e deve estar entre colchetes.
No lugar de uma variável pode ser usada uma expressão.

Operadores - Os operadores são formados por duas letras e são usados para comparar dois valores com a finalidade de determinar se são iguais ou se um valor é menor ou maior que outro valor.

OPERADOR	SIGNIFICADO
EQ	Igual a
NE	Diferente de
GT	Maior que
GE	Maior ou igual a
LT	Menor que
LE	Menor ou igual a

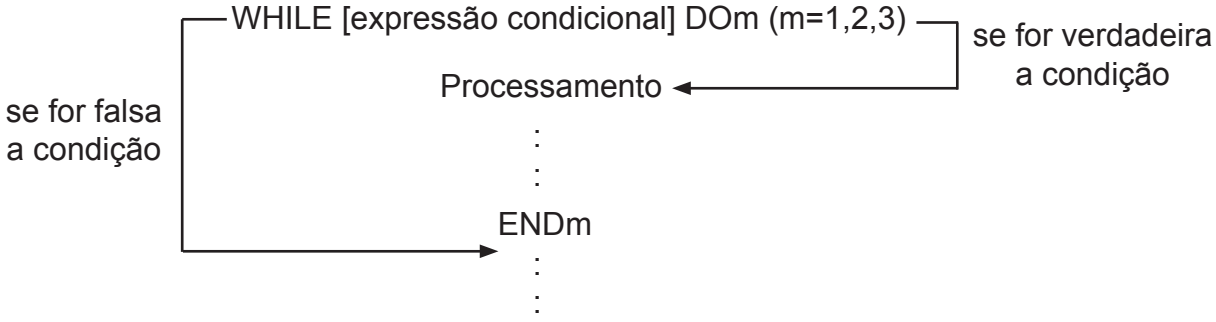
Programa exemplo: Determinar a soma dos números de 1 a 10.

```
O9100
#2=1                    - Valor inicial da variável #2=1
N1 IF[#2 GT10] GOTO2    - Desviar para N2 se #2 for maior que 10
#2=#2+1                 - Incrementando a variável
GOTO 1                  - Desviar para N1
N2 M30                  - Fim do programa
```

Os valores das variáveis #2 a cada etapa.
#2=2,3,4,5,6,7,7,8,9,10,11.

11.8.3 - Repetição - WHILE

Especifique uma expressão condicional depois de WHILE. Enquanto a condição especificada for verdadeira, o programa vai sendo executado desde a declaração DO até a declaração END. Se a condição especificada for falsa o programa passa a ser executado no bloco que vem em seguida a declaração END.



Explicação:

Enquanto a condição especificada depois de WHILE for verdadeira,o programa continua sendo executado desde a declaração DO até a declaração END.

Se a condição especificada for falsa o programa continua sendo executado a partir do bloco que vem depois de END.

Um número depois de DO e um número depois de END são números de identificação para especificar um intervalo de execução.

Deve-se usar os números 1, 2 e 3. Quando usa-se um número diferente de 1, 2 e 3 será mostrado o alarme 126.

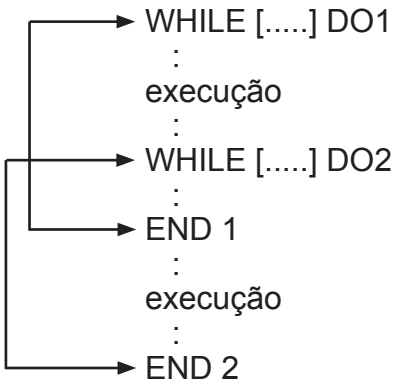
11.8.4 - Níveis de rotinas usando a função WHILE

Os números de identificação de 1 até 3 em um desvio DO-END podem ser usados quantas vezes desejado. Note porém que quando um programa inclui rotinas de repetição entrelaçados (intervalos do sobrepostos) um alarme 124 ocorrerá.

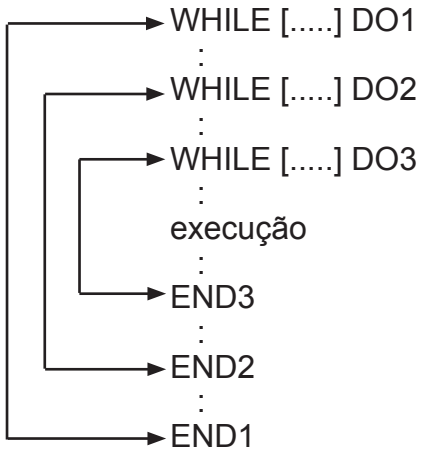
a) Os números de identificação (1 a 3) podem ser usados varias vezes como desejado.

```
WHILE [.....] DO1
  :
execução
  :
END1
  :
WHILE [.....] DO1
  :
execução
  :
END1
```

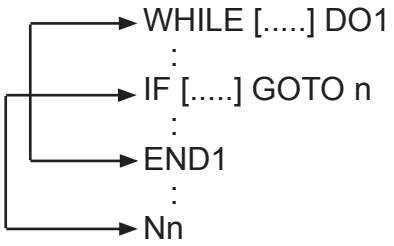
b) Não podemos sobrepor os intervalos DO



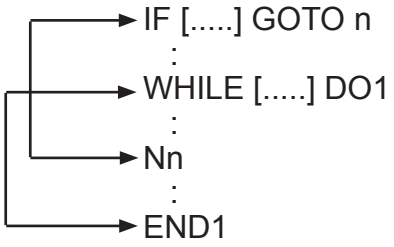
c) As rotinas DO podem ser entrelaçadas ate o máximo de 3 níveis.



d) O comando pode ser transferido para uma rotina externa.



e) Os desvios não podem ocorrer para um ponto dentro da rotina.



11.9 - LIMITES

Quando se especifica DOm sem especificar a declaração WHILE, se executa uma rotina infinita que vai desde DO até END.

Tempo de processamento: Quando se executa um desvio a um número de seqüência específico em uma declaração GOTO, busca-se um número de seqüência.

Por este motivo o processamento no sentido inverso (para trás), demora-se mais que o processamento no sentido direto (para frente).

Utilizando a declaração WHILE para repetição se reduz o tempo de processamento.

11.10 - VARIÁVEL NÃO DEFINIDA

Em uma expressão condicional que utiliza EQ ou NE, uma variável nula e o zero tem efeitos diferentes. Em outros tipos de expressões condicionais, um valor nulo é considerado zero.

11.11 - CHAMADA DE MACROS

Pode-se invocar um programa macro usando os métodos de chamada simples (G65) e chamada modal (G66).

11.11.1 - Diferenças entre chamadas de macro e chamadas de subprogramas

A chamada de macro (G65) é diferente da chamada de um subprograma (M98) como se descreve a seguir:

a) Com G65 pode-se especificar um argumento (dado transferido a uma macro), M98 não permite fazê-lo.

b) Quando um bloco M98 contém outro comando - ex.: G01 X100 M98 P___ -; se chama o subprograma depois de se executar o comando. Por outro lado, G65 chama incondicionalmente uma macro.

c) Quando um bloco M98 contem outro comando - ex.: G01 X100 M98 P__ -; a maquina pára no modo bloco a bloco, por outro lado o G65 não detém a maquina.

d) Com G65, o nível de variáveis locais variam, com M98 o nível de variáveis locais não varia.

11.11.2 - Chamada Simples (G65)

Quando se especifica G65, se chama uma macro especificado no endereço P. Os dados (argumentos) podem ser transferidos para um programa macro.

Sintaxe:

G65 P____ L_____

Onde:

P ___ ; número do programa que contem a macro
L ___ ; número de repetições (1=default)

Exemplo:

```
O0001                    O1000
G65 P1000 A1 B2          #3=#1+#2
M30                      IF[#3 GT360] GOTO9
                         G0 G91 X #3
                         N9 M99
```

Explanações:

Após G65 especifica-se o endereço P com o número do programa que contem a macro. Quando o número de repetições for necessário especifica-se o número de 1 ate 9999. Quando o número for omitido a repetição será única. Utilizando uma especificação do argumento se atribuem valores as correspondentes variáveis locais.

11.12 - ESPECIFICAÇÕES DE ARGUMENTOS

Existem dois tipos de especificações de argumentos. A especificação de argumentos I usa letras diferentes de G, L, O, N e P.

A especificação de argumentos II utiliza as letras A, B, C e também I, J, K até dez vezes.

O tipo de especificação do argumento está determinado automaticamente pelas letras utilizadas.

Exemplo:

```
O0001 (PROG. PRINCIPAL - ARG. TIPO  I)
G65 P1000 I0 J0 K0 D100 E50 F-20 H2
M30

OU

O0001 (PROG. PRINCIPAL - ARG. TIPO  II)
G65 P1000 I0 J0 K0  I100 J50 K-20 J2
M30
```

```
O1000 (MACRO)
G0 X[#4] Y[#5]
Z[#6+2]
WHILE [#6 GT #9] DO1
G1 Z[#6] F500
X[#7]
Y[#8]
X[#4]
Y[#5]
#6=#6-#11
END1
M99
```

11.12.1 - Especificação de argumentos I

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

OBSERVAÇÃO: *Os endereços G, L, N, O, e P não podem ser usados como argumento, os endereços que não se usam podem ser omitidos, as variáveis locais correspondentes a um endereço omitido se configuram como nulas.*

11.12.2 - Mesclagem das especificações de argumentos I e II

ENDEREÇO	NÚMERO DA VARIÁVEL
A	#1
B	#2
C	#3
I_1	#4
J_1	#5
K_1	#6
I_2	#7
J_2	#8
K_2	#9
I_3	#10
J_3	#11
K_3	#12
I_4	#13
J_4	#14
K_4	#15
I_5	#16
J_5	#17
K_5	#18
I_6	#19
J_6	#20
K_6	#21
I_7	#22
J_7	#23
K_7	#24
I_8	#25
J_8	#26
K_8	#27
I_9	#28
J_9	#29
K_9	#30
I_{10}	#31
J_{10}	#32
K_{10}	#33

***OBSERVAÇÃO**: Os subíndices de I, J, K, para indicar a ordem da especificação de argumentos não se registram no programa.*

11.13 - LIMITAÇÕES

Formato: antes de qualquer argumento deve-se especificar G65

Mescla de especificações de argumentos I e II: se existe a mescla dos argumentos I e II tem prioridade o tipo especificado em último lugar.

Níveis de chamadas: pode-se programar desvios de chamadas num máximo de 4 níveis. Aqui não se incluem as chamadas de um subprograma (M98)

11.14 - NÍVEIS DE VARIÁVEIS LOCAIS.

Existem variáveis locais desde o nível 0 até o nível 4 para programação de desvio. O nível do programa principal e o zero.

Cada vez que uma macro e chamada com G65 ou G66, o nível da variável local aumenta em 1. Os valores das variáveis locais do nível anterior se armazenam no CNC.

Quando um M99 é executado num programa de macro, o comando retorna para o programa onde houve o desvio. Neste momento, o nível da variável local e decrementado em uma unidade. Os valores das variáveis locais armazenadas será restabelecido quando a macro for chamada.

Programa Principal	**Macro**	**Macro**	**Macro**	**Macro**
Nível 0	**Nível 1**	**Nível 2**	**Nível 3**	**Nível 4**
O0001 #1=1 G65 P2 A20 : G4 X[#1] (#1 = 1) : M30	O0002 : G65 P3 A45 : G1 X[#1] F.15 (#1 = 20) : M99	O0003 : G65 P4 A0.2 : G0 Z[-#1] (#1 = 45) : M99	O0004 : G65 P5 A50 : G1 X32 F[#1] (#1 = 0.2) : M99	O0005 : : G0 X[#1] (#1 = 50) : : M99

***OBSERVAÇÃO*:** *Para cada nível de subprograma podem ser utilizadas 33 variáveis locais, ou seja, da variável #1 até a #33.*

11.15 - VARIÁVEIS COMUNS

#100 - , #500 - Variáveis que podem ser lidas e gravadas por macros em diferentes níveis.

11.16 - CHAMADA MODAL (G66)

A função G66 faz com que a chamada de uma determinada macro se torne modal até que a mesma seja cancelada pela função G67.

Sintaxe:

G66 P___ L____ .....

Onde:

P - número do subprograma
L - número de repetições

Explicações:

Após especificar G66, programe o endereço P com o número do programa que contém a macro. Quando se deseja o número de repetições, o endereço L pode conter um número de 1 até 9999. Assim como usado na função G65, os dados são transferidos a um programa de macro através de argumentos.

Cancelamento: Quando se especifica um código G67 já não se excetua as chamadas modais nos blocos posteriores.

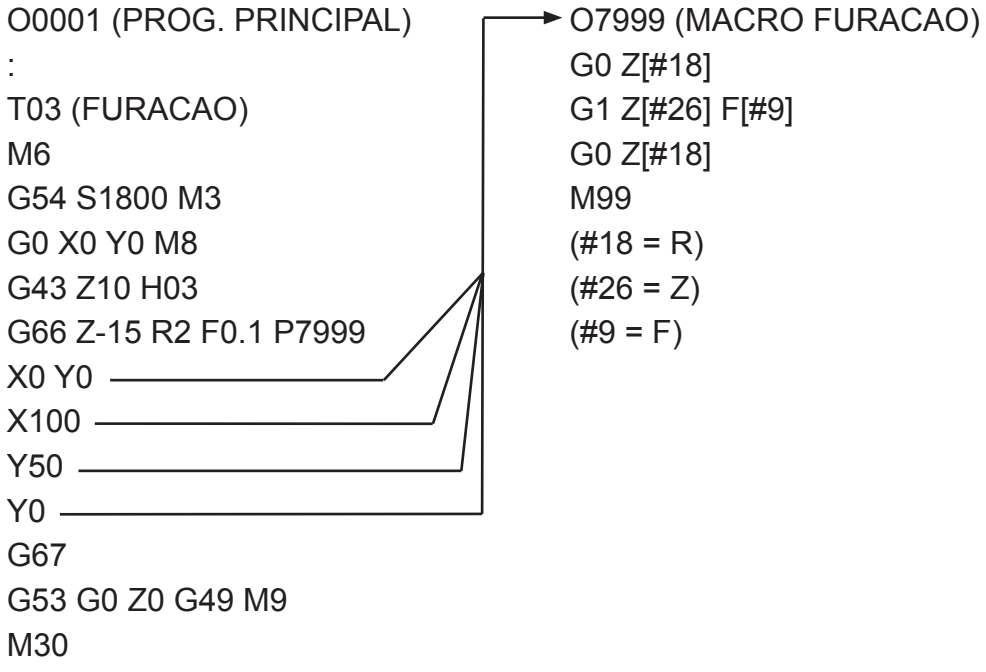
Níveis de chamadas: As chamadas podem ser especificadas usando desvios de até 4 níveis incluindo as chamadas modais.

Níveis de chamadas modais: Pode-se ativar uma chamada modal dentro de outras especificando outro código G66.

Limitações: Em um bloco G66 não se pode ativar macros. G66 deve ser especificado antes de qualquer argumento.

As variáveis locais (argumentos) podem ser definidas unicamente em blocos G66.

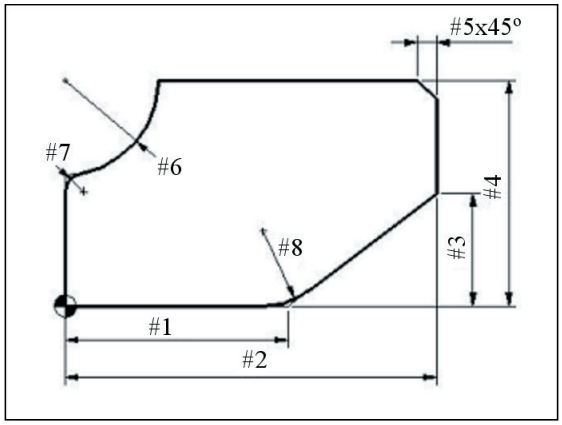
Exemplo:



11.17 - EXEMPLOS DE PROGRAMAÇÃO:

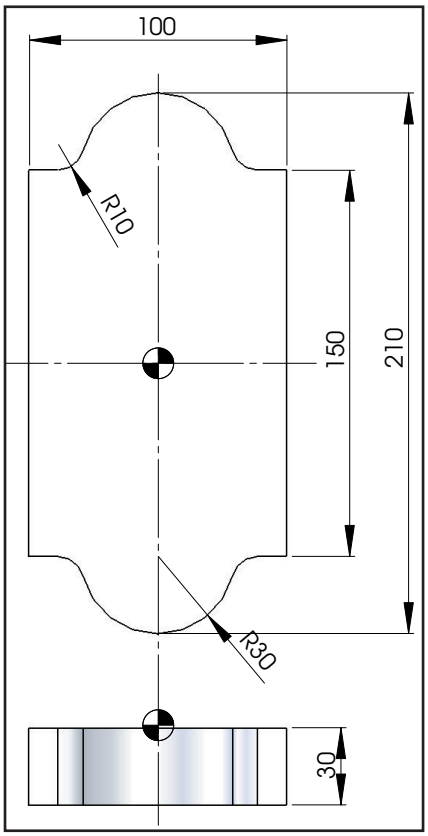
Abaixo estão alguns exemplos de aplicação da programação parametrizada.

Exemplo 1: Programa parametrizado para fazer uma família de peças, conforme o desenho abaixo:



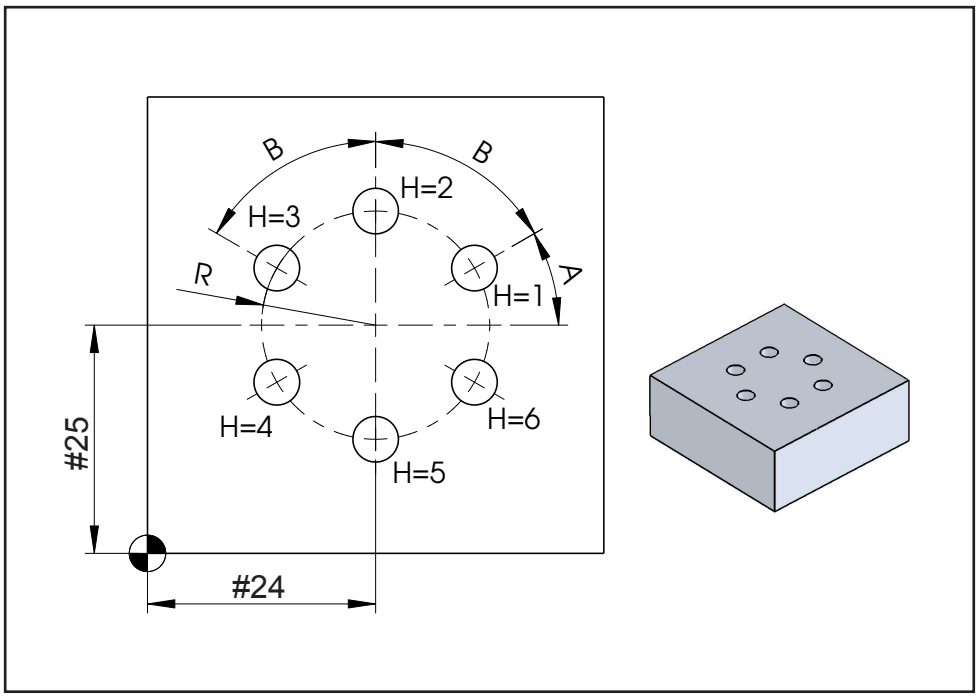

```
O0001 (FAMILIA 1)
N10 #1=60 (COMPRIMENTO 1)
N20 #2=100 (COMPRIMENTO TOTAL DA PECA)
N30 #3=30 (LARGURA 1)
N40 #4=60 (LARGURA TOTAL DA PECA
N50 #5=5  (LARG. DO CHANFRO x 45 GRAUS)
N60 #6=25 (RAIO 1)
N70 #7=7 (RAIO 2)
N80 #8=20 (RAIO 3)
N90 #9=20 (DIAMETRO DA FERRAMENTA)
N100 #10=400 (AVANCO)
N110 #11=0 (Z INICIAL)
N120 #12=-20 (Z FINAL )
N130 #13=2 (PROFUNDIDADE DE CORTE)
N140 #14=5 (FOLGA P/  APROXIM. EM X E Y)
N150 #15=5 (FOLGA P/ APROXIM. EM Z)
N160 #16=0 (RECONHECE ULTIMO PASSE)
N170 #9=#9/2(CALCULO DO RAIO FERRAM.)
N180 #20=#11- #13 (COORD. 1A. PASSADA)
N190 G17 G21 G90 G94
N200 G53 G0 Z0 G49
N210 T3
N220 M6
N230 G54 S3500 M3
N240 G0 X-[#9+#14] Y-[#9+#14] M8
N250 G43 Z[#11+#15] H3 D3
N260 G0 Z[#20]
N270 G42 G1 X0 Y0 F[#10]
N280 X[#1] ,R[#8]
N290 X[#2] Y[#3]
N300 Y[#4] ,C[#5]
N310 X[#6]
N320 G2 X0 Y[#4-#6] R[#6] ,R[#7]
N330 G1 Y0
N340 G40 X-[#9+#14] Y-[#9+#14]
N350 #20=[#20-#13]
N360 IF [#16 EQ 1] GOTO420
N370 IF [ #20 GT #12 ] GOTO260
N380 #16=1
N390 G0 Z[#12]
N400 GOTO270
N410 G0 Z[#11]
N420 G53 G0 Z0 G49 M5
N430 M30
```

Exemplo 2: Programa parametrizado para criar uma subrotina sem utilizar o recurso de subprograma:




```
O0001 (PECA COM SUBROTINA)
G53 G0 Z0 G49
T15
M06
G54 S3600 M03
G00 X-65 Y0
G43 Z10 H15 D15
#1=0 (Z INICIAL)
#2=-20 (Z FINAL)
#3=2 (PROF. CORTE)
#4=[#1- #3] (VAR. CALCULO)
G0 Z[#1+2]
WHILE [#4 GT #2] DO1
G0 Z[#4]
G41 G1 X-50 F1000
Y75
X-30 ,R10
G2 X30 Y75 R30 ,R10
G1 X50
Y-75
X30 ,R10
G2 X-30 Y-75 R30 ,R10
G1 X-50
Y0
G40 X-65 Y0 F5000
#4=[#4- #3]
END1
(***ACABAMENTO***)
G0 Z[#2]
G41 G1 X-50 F1000
Y75
X-30 ,R10
G2 X30 Y75 R30 ,R10
G1 X50
Y-75
X30 ,R10
G2 X-30 Y-75 R30 ,R10
G1 X-50
Y0
G40 X-65 Y0 F5000
G0 Z[#1+2]
G53 G0 Z0 G49
M30
```

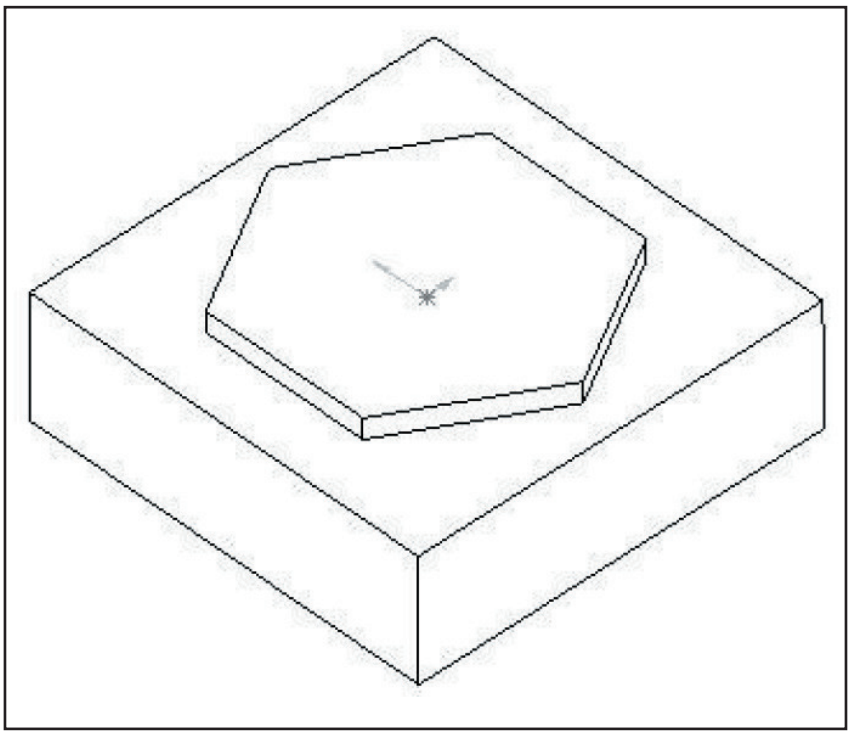
Exemplo 3: Programa parametrizado para fazer um arco (círculo) de posicionamentos, conforme o desenho abaixo:




```
O0001 (PROG. PRINCIPAL)
G17 G21 G90 G94
G53 G0 Z0 G49
T1 (BROCA D8.5 MM)
M6
G54 S1500 M3
G0 X0 Y0 M8
G43 Z10 H1 D1
G99 G81 Z-10 R2 F160 K0
G65 X80 Y50 R20 A45 B45 H3 P1000
G80
G53 G0 Z0 G49 M9 M5
M30
```

```
O1000 (SUBPROG. MACRO)
#3=1
N1
X[#24+[#18*COS[#1]]] Y[#25+[#18*SIN[#1]]]
#1=#1+#2
#3=#3+1
IF [#3 LE #11] GOTO1
M99
```

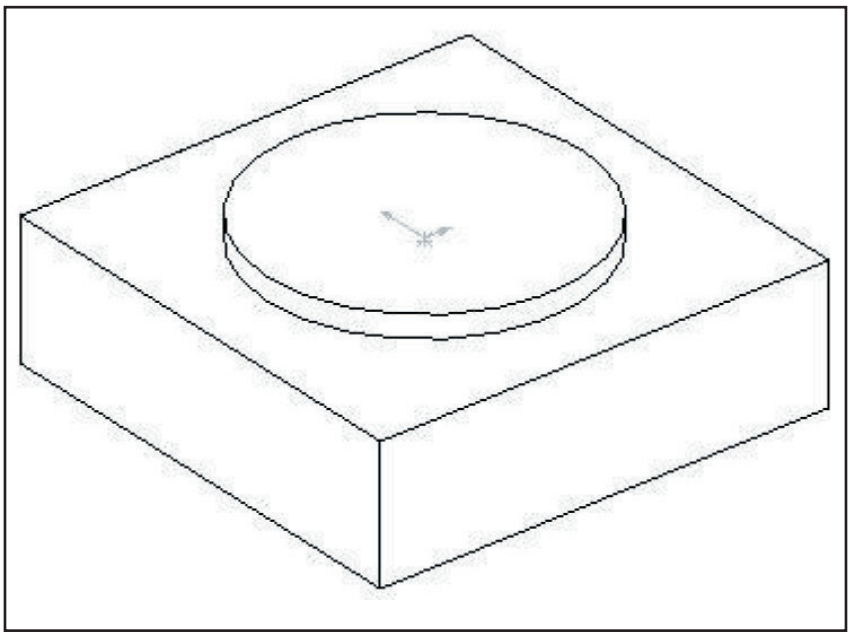
Exemplo 4: Programa parametrizado para fazer um sextavado inscrito num determinado círculo:



```
O0001 (MACRO SEXTAVADO)
G17 G21 G90 G94
G53 G0 Z0 G49
T12
M6
G54 S2700 M3
#1=50 (RAIO DO CIRCULO)
#3=0 (ANGULO INICIAL)
#4=3 (FOLGA P/ A APROXIMACAO)
#5=200 (VELOCIDADE DE AVANÇO)
#6=0 (CONTADOR DO NO. LADOS)
#10=#3 (DUPLICA A VARIAVEL R3)
G0 X[[#1+#4]*COS[#3]] Y[[#1+#4]*SIN[#3]]
G43 Z2 H12
G1 Z-5 F150
N12 G1 X[#1*COS[#3]] Y[#1*SIN[#3]] F[#5]
#3=#3+60
#6=#6+1
IF [#6 LE 6] GOTO12
G1 X[[#1+#4]*COS[#10]]
Y[[#1+#4]*SIN[#10]]
G53 G0 Z0 G49
M30
```

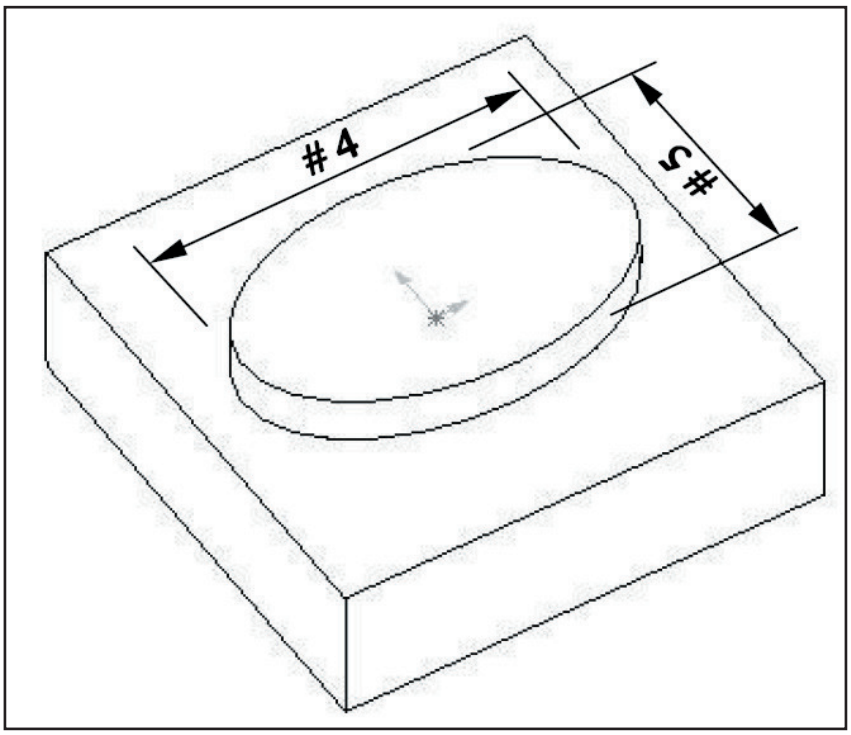
OBSERVAÇÃO: *No exemplo acima o raio da ferramenta não foi considerado,ou seja, o percurso programado corresponde a trajetória percorrida pelo centro da ferramenta*

Exemplo 5: Programa parametrizado para executar arcos de 0,001 a 360 graus de abertura utilizando a função G01.



```
O0005 (MACRO CIRCULAR)
G17 G21 G90 G94
G53 G0 Z0 G49
T1
M6
G54 S2500 M3
#1=50 (RAIO DO ARCO)
#20=10 (DIAMETRO FERRAMENTA)
#3=0 (ANGULO INICIAL/CORRENTE)
#4=360 (ANGULO FINAL)
#5=1 (INCREMENTO ANGULAR)
#6=#20/2 (DEFIN. RAIO FERRAMENTA)
#7=3 (FOLGA P/ A APROXIMACAO)
#10=200 (VELOCIDADE DE AVANÇO)
#1=#1+#6 (REDEFIN. RAIO DO ARCO)
G0 X[[#1+#7]*COS[#3]] Y[[#1+#7]*SIN[#3]]
G43 Z2 H1
G01 Z-5 F250
N11 G1 X[#1*COS[#3]] Y[#1*SIN[#3]] F[#10]
#3=#3+#5
IF [#4 GT #3] GOTO11
G1 X[#1*COS[#4]] Y[#1*SIN[#4]]
X[[#1+#7]*COS[#4]] Y[[#1+#7]*SIN[#4]]
G53 G0 Z0 G49
M30
```

Exemplo 6: Programa parametrizado para fazer uma elipse real de 360º.

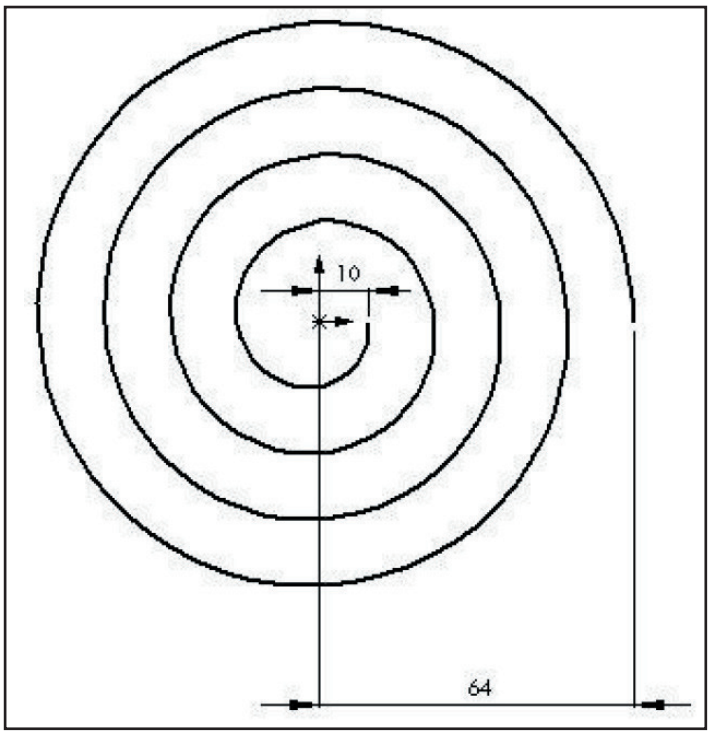



```
O0010 (PROG. PRINCIPAL)
N1 G17 G21 G90 G94
N2 G53 G0 Z0 G49
N3 T18 (FRESA D20 MM)
N4 M6
N5 G54 S3500 M3
N6 G0 X60 Y0 M8
N7 G43 Z10 H18 D18
N8 Z-2
N9 G65 X0 Y0 I80 J50 A0 B1 Q5
     R10 F520 P2000
N10 G53 G0 Z0 G49 M9 M5
N11 M30
```

```
O2000 (SUBPROG. MACRO)
N1 #4=[#4/2]+#18
N2 #5=[#5/2]+#18
N3 G00 X[[#4+#17]*COS[#1]] Y[[#5+#17]*SIN[#1]]
N4 WHILE [#1 LT 360] DO1
N5 G01 X[#4*COS[#1]] Y[#5*SIN[#1]] F[#9]
N6 #1=#1+#2
N7 END1
N8 G01 X[#4*COS[360]] Y[#5*SIN[360]]
N9 G00 X[[#4+#17]*COS[#1]] Y[[#5+#17]*SIN[#1]]
N10 M99
```

Exemplo 7: Programa parametrizado para executar uma espiral de arquimedes tendo:

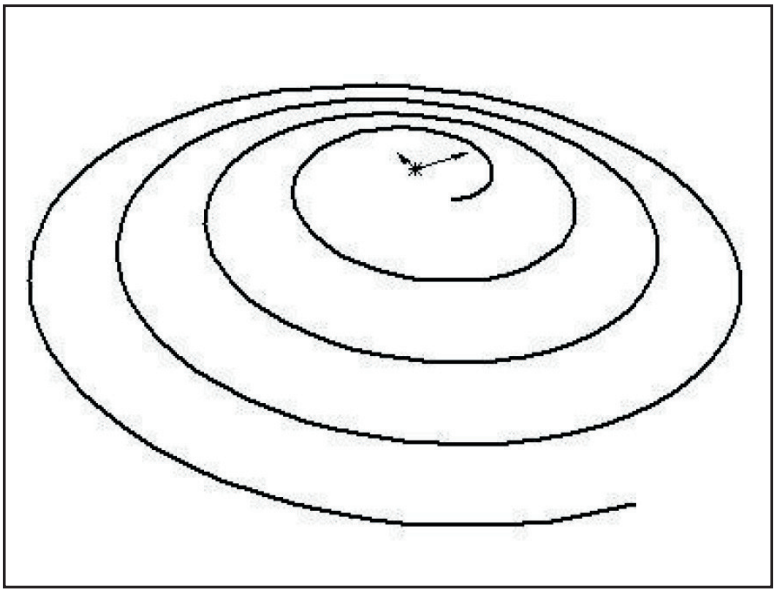
Raio inicial: 10mm
Raio final: 64 mm
N.espirais: 5
Posição inicial: 0 grau




```
O0010 (MACRO ESPIRAL)
G17 G21 G90 G94
G53 G0 Z0 G49
T12
M6
G54 S2250 M3
#1=10 (RAIO INICIAL)
#20=64 (RAIO FINAL)
#3=0 (ANGULO INICIAL/CORRENTE)
#7=5 (NUMERO DE ESPIRAIS)
#4=#7*360 (ANGULO FINAL)
#5=1 (INCREMENTO ANGULAR)
#8=[#20-#1]/#4 (INC/TO RADIAL X)
#9=#8*#5 (INC/TO RAD. P/ INC/TO ANG.)
#10=200 (VELOCIDADE DE AVANÇO)
G0 X[#1*COS[#3]] Y[#1*SIN[#3]]
G43 Z2 H12
G1 Z-5 F200
N45 G1 X[#1*COS[#3]] Y[#1*SIN[#3]] F[#10]
#3=#3+#5
#1=#1+#9
IF [#4 GT #3] GOTO45
G1 X[#20*COS[#4]] Y[#20*SIN[#4]]
G53 G0 Z0 G49
M30
```

Exemplo 8: Programa parametrizado para executar uma espiral de arquimedes conjugado com uma descida em “Z” tendo:

Raio inicial: 10 mm
Raio final: 64 mm
N.espirais: 5
Posição inicial: 0 graus
Pos “Z” inicial: 0
Pos “Z” final: 25

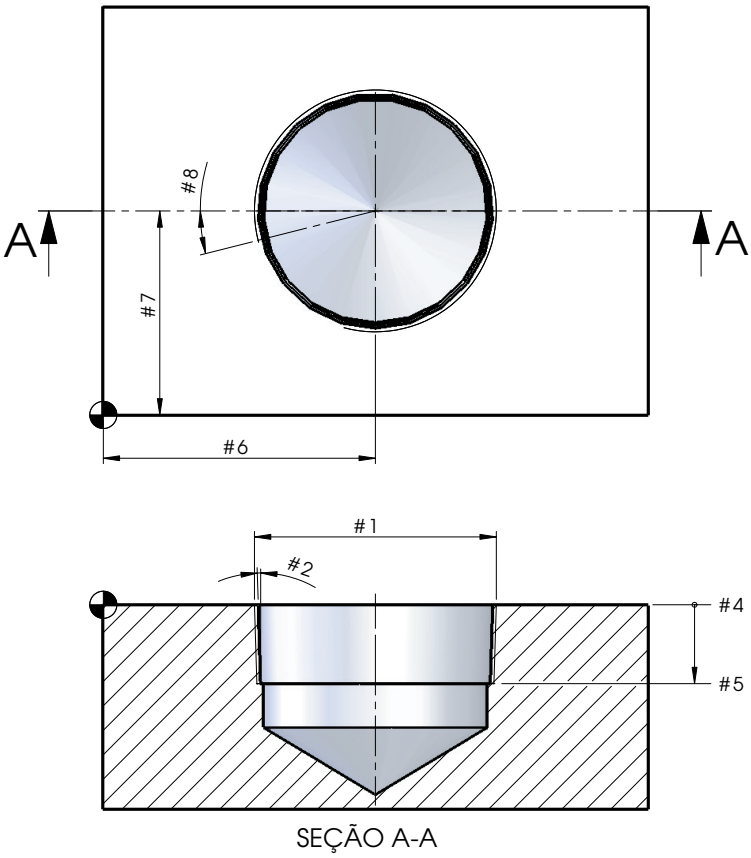


```
O0011 (MACRO ESPIRAL 2)
G17 G21 G90 G94
G53 G0 Z0 G49
T1
M6
G54 S3500 M3
#1=10 (RAIO INICIAL)
#20=64 (RAIO FINAL)
#3=0 (ANGULO INICIAL)
#7=5 (NUMERO DE ESPIRAIS)
#4=#7*360 (ANGULO FINAL)
#5=1 (INCREMENTO ANGULAR)
#8=[#20-#1]/#4 (INC/TO RADIAL X)
#9=#8*#5 (INC. RAD. P/ INC. ANG.)
#10=200 (VELOC. DE AVANÇO)
#11=0 (ALTURA Z INICIAL)
#12=25 (ALTURA Z FINAL)
#13=#12-#11 (PROFUNDIDADE)
#14=#13/#4 (INC/TO Z)
#15=#14*#5 (INC. Z P/ INC. ANG.)
G0 X[#1*COS[#3]] Y[#1*SIN[#3]]
G43 Z2 H1
G1 Z[#11] F[#10]
N22 G1 X[#1*COS[#3]] Y[#1*SIN[#3]] Z[#11]
#3=#3+#5
#1=#1+#9
#11=#11-#15
IF [#4 GT #3] GOTO22
G1 X[#20*COS[#4]] Y[#20*SIN[#4]] Z[-#12]
G53 G0 Z0 G49
M30
```

Exemplo 9: Programa parametrizado para interpolar uma rosca cônica interna utilizando uma ferramenta de roscar interno :

Diâmetro inicial: 88.9 mm
Âng. da rosca: 1.783º (1º 47')
Passo: 3.175 mm (8 fpp)
Pos "Z" inicial: 0 (abs)
Pos "Z" final: -28.98 (abs)
Pos "X" do centro da rosca: 0 (abs)
Pos "Y" do centro da rosca: 0 (abs)
Ângulo de entrada da rosca: 185º

Nota: Os valores acima são referentes a rosca 3" NPT - 8 fpp.



SEÇÃO A-A

```
O0001 (MACRO ROSCA CONICA DIREITA/ESQUERDA)
G17 G21 G90 G94
G53 G0 Z0 G49
T01
M6
G54 S3500 M3
G43 Z100 H01 D01
#1=88.9 (DIAM. INICIAL)
#2=1.783 (ANG. DA ROSCA)
#3=3.175 (PASSO DA ROSCA)
#4=0 (Z INICIAL)
#5=-28.98 (Z FINAL)
#6=100 (COORD. CENTRO EM X)
#7=75 (COORD. CENTRO EM Y)
#8=185 (ANG. ENTRADA DA ROSCA)
#9=1 (INCR. ANG. - RESOLUCAO CIRCULO)
#10=2 (DIAMETRO DA FERRAM.)
```

```
#11=350 (AVANCO PARA FRESAM.)
#12=0 (ROSCA DIR=0 - ROSCA ESQ=1)
(**** VARIAVEIS DE CALCULO - NAO MODIFICAR ****)
#20=[#1-#10]/2 (CALC. RAIO INICIAL DA ROSCA)
#21=#4-#5 (CALC. PROF. TOTAL)
#22=#9*#3/360 (CALC. PROF. POR INCR. ANG)
#23=TAN[#2]*#21 (CALC. CATETO RADIAL)
#24=TAN[#2]*#3 (CALC. INCR. RADIAL)
#25=#9*#24/360 (CALC. INCR. RAD. POR INCR. ANGULAR)
#26=0 (CONTADOR ANGULAR - SEMPRE 0)
#27=#22*#26 (CALC. PROF. PELO CONTADOR)
#28=#20-#23 (CALC. DO RAIO FINAL DA ROSCA)
#29=#8+[#21*360/#3] (CALC. DO ANGULO FINAL)
IF [#12 EQ 0] GOTO1
IF [#12 EQ 1] GOTO2
#3000=1 (DEFINIR TIPO DA ROSCA - DIR/ESQ)
N1 #30=1 (ROSCA DIREITA)
GOTO3
N2 #30=-1 (ROSCA ESQUERDA)
N3
(**** PROGRAMA MACRO - NAO MODIFICAR****)
G0 X[#6+[#20*COS[#8]]] Y[#7+[#20*SIN[#8]]]
Z[#4+2]
G1 Z[#4] F[#11]
WHILE [#27 LT #21] DO1
G1 X[#6+[[#20+[#30*#25*#26]]*COS[#8]]] Y[#7+[[#20+[#30*#25*#26]]*SIN[#8]]] Z[#4-#27]
#26=#26-[#30*#9]
#8=#8-[#30*#9]
#27=#27+#22
END1
G1X [#6+[#28*COS[#29]]] Y[#7+[#28*SIN[#29]]] Z[#4-#21]
X[#6] Y[#7]
G0 Z[#4+2]
G53 G0 Z0 G49
M30
```