Section 18: Macros

Macros and Parametric Programming

Macros

Macros give the programmer the ability to perform arithmetic and comparison functions within a CNC program. Values for variables (V1- V100) can be entered from input statements or passed to the macro from R variables or from the fixture, tool data, and tool time tables. Values from macro statements can be passed to the CNC program by using parametric variables (R0-R9). Macros may be used for probe functions and

Parametric Programming

Parametric programming gives the operator the ability to use a parametric R variable to represent the value of any program coding word. The variables include: R0 and R1 - R9. Parametric programming is also used to transfer data from macro variables V1-V100 to the CNC program for machine motion, feed and speed.

EXAMPLE:

N13 R1 + 10. R2 + 5. (R1 AND R2 ARE ASSIGNED VALUES
N14 R3 + 7.5 R4 + 5.5 R5 + 0 (R3, R4 AND R5 ARE ASSIGNED VALUES
N15 X - R1 Y + R2. G1 F34. (THIS LINE READS X10. Y5. G1 F34.
N16 X + R3. I - R4 J - R5 G3 (THIS LINE READS X7.5 I5.5 J0 G3

Parametric variables are used when programming words require change during the execution of a program or need to be changed at different runs. One typical use of these variables would be in the use of a subroutine or subprogram to cut a pocket. The Z words in the routine may use a variable to cut many different levels. The L word for the subroutine call may use a variable to change the number of times that the sub is called. The R variable used for the L word must indicate the subroutine number and number of repetitions (see the example below). The X, Y, I, and J of a circular move can all be changed by parametric programming. All program words can use a parametric variable. Parametric programming is also used when programmer is considering a generic program for a family of parts.

Note: When variables are used as values, a positive or negative symbol must be used between the two variables, R9+R2, R8-R2. This is also true when using radius designation for circular moves or for the minimum clearance plane description with fixed cycles.

R Variable Definition R variables are defined by coding the R word, + or - symbol and a value.

EXAMPLE:
N12 R8+50.0 This is variable R8 defined as 50.0
N13 F+R8 The R8 transfers a value of 50.0 for the feed rate

Note: Variables must be defined in the beginning of the program or just before they are used in the program. Variables are modal and retain their values after the termination of a program, after an HO, and after exiting from MDI. R variables do not have table storage like macro V variables. The programmer should always specify a value for an R variable; otherwise, the last programmed value will be used and will result in unpredictable machining.

Example Program for Parametric Programming The parametric R variables are defined in the main program, on lines 32 and 37, and in the subroutine on line 19 of the program below. The R variables are modal; they remain effective until the same R variable is redefined. Note that on line 21 of the program below, the R1 variable is changed but the R2 variable remains in effect.

N01 L100(SUB TO CUT SLOT
N02 (R1=Z, R2=SUB & REPS, R3=RADIUS OF SLOT, R4=DIA OF SLOT
N03 X-1. Z-R1 G1
N04 X-R3 G41
N05 L+R2
N06 X+R3 G40
N07 X1.
N08 L200(SUB FOR SLOT
N09 X+R3 Y-R3 I+R3 G3
N10 X1.
N11 Y+R4 J+R3 G3
N12 X-1.
N13 X-R3 Y-R3 J-R3 G3
N14 L300(CUT SLOT
N15 Z0.01 G1
N16 R1+0.1 R2+201.
N17 G91
N18 L102
N19 R1+0.05
N20 L101

N21 R1+0.01 R2+202.
N22 L101
N23 G0 G49 G90 Z0.1
N24 M17
N25 M30
N26 (
N27 (MAIN PROGRAM
N28 G0 G90 S8000 M3 E1 X1.5 Y-0.876
N29 (R1=Z, R2=SUB & REPS, R3=RADIUS OF SLOT, R4=DIA OF SLOT
N30 H1 Z0.1 M7
N31 Z0.01 G1 F30.
N32 R3+0.19 R4+0.38
N33 L301
N34 X4. Y-0.876
N35 L301
N36 X4. Y-3.45
N37 R3+0.255 R4+0.51
N38 L301
N39 X1.5 Y-3.45

Macros Each macro line is identified by using the # symbol at the beginning of a line in the program. Macros can reside in a main program, subroutine, or in a subprogram. Macros may be used during **AUTO, MDI** (Manual Data Input), **DNC**, and at the command line. The maximum number of characters per macro line is 63.

Parametric R variables are used to transfer macro data to the CNC program.

EXAMPLE:
N1 # V1 = V1 + 2.
N2 # V2 = V2 - 1.
N3 # R1 = V1
N4 # R2 = V2
N5 G0 G90 S8000 M3 E1 X+R1 Y-R2

Machine program code **cannot** be coded in a macro line.

EXAMPLE: *N1 G0 G90 # V1=INT(V5 * 10000)/10000 This line is incorrect*
*N1 # V1=INT(V5 * 10000)/10000 This line is correct*
N2 G0 G90 This line is correct

Example Macro Program for a Rectangle with PRINT and INPUT Statements:

0600 (SUB FOR RECTANGULAR POCKET FINISH
#CLEAR
#PRINT "ENTER THE POCKET LENGTH: ",
#INPUT V1 Keyboard entry is transferred to V1
#PRINT "ENTER THE POCKET WIDTH: ",
#INPUT V2 Keyboard entry is transferred to V2
#R3=V1/2 R3 equals V1 divided by 2
#R4=V2/2 R4 equals V2 divided by 2
#R2=V1 Transfers the V1 value to R2 parameter
#R5=V2 Transfers the V2 value to R5 parameter
G0G90Z.1
X0Y0
D1
Z-.250G1F20
G91 G41 Y-R4
X+R3
Y+R5
X-R2
Y-R5
X+R3
Y+R4 G40 G90 Cancel CRC
G0G90Z.1 Safety Z move
M99 End of sub program

O700 (SUB FOR RECTANGULAR POCKET FINISH WITH CORNER RADIUS

#CLEAR
#PRINT "ENTER THE POCKET LENGTH: ",
#INPUT V1 Keyboard entry is transferred to V1 (X VALUE)
#PRINT "ENTER THE POCKET WIDTH: ",
#INPUT V2 Keyboard entry is transferred to V2 (Y VALUE)
#PRINT "ENTER THE CORNER RADIUS:"
#INPUT V3 Keyboard entry is transferred to V3
#V4=V3+V3 Diameter
#R2=V1-V4 Transfers the V1-V4 value to R2 parameter (X

#R3=V1/2 R3 equals V1 divided by 2 (X VALUE)
#R4=V2/2 R4 equals V2 divided by 2 (Y VALUE)
#R5=V2-V4 Transfers the V2-V4 value to R5 parameter (Y
#R6=V3 Radius
#R7=(V2/2) V3R7 equals V2 divided by 2, minus the radius value (Y
#R8=(V1/2) V3R3 equals V1 divided by 2, minus the radius value (X
G0G90Z.1 H1Safety Z move
X0Y0 Locate to the center of pocket
D1 Diameter of tool must be in tool table
Z-.250G1F20. Feed down into pocket
G91 G41 Y-R4CRC climb cut
X+R8 Parametric transfer to axis movement
X+R6 Y+R6 R0+R6 G3
Y+R5
X-R6 Y+R6 R0+R6 G3
X-R2 Macro calculations determine the X Y moves here
X-R6 Y-R6 R0+R6 G3
Y-R5
X+R6 Y-R6 R0+R6 G3
X+R8
Y+R4 G40 G90 Cancel CRC
G0G90Z.1 Safety Z move
M99 End of subprogram

Mathematical Functions

Mathematical Function Macros

Macros operate with the use of mathematical functions, commands and variables. The variables are used in conjunction with the functions to perform calculations. The characters below describe the mathematical function capabilities. Older controls are required to type the Alpha Keys for the characters not on the keyboard.

Table 1: Mathmatical Function Macros

Character	Description	Alpha Keys
	Blank	
+	Plus, Positive, Addition	
-	Minus, Negative, Subtraction	
*	Multiply, Multiplication	
/	Divide, Division	
ABS	Absolute value	
ATN	Arc tangent	
COS	Cosine function	
INT	Integer value	
RND	Rounding function	
SGN	Return the sign	
SIN	Sine function	
SQR	Square root	
(	Left Parenthesis	
)	Right Parenthesis	
=	Equal	EQ
<	Less Than	LT
>	Greater Than	GT
<=	Less Than or Equal to	LE
>=	Greater Than or Equal to	GE
<>	Not Equal to	NE
'	Remark in macro line	

Order of Calculation

Arithmetic operations are performed in standard mathematical order:

Parentheses ( )

Functions (SIN,COS ...)

- (negation)

* (multiplication)and / (division)

+ (addition)and - (subtraction).

AND, OR, NOT

Parentheses can be used to establish order of arithmetic operations, overriding standard mathematical order.

Decimals Numbers without decimals will be considered integers. **This is true only for macro lines.**

Exponential Form Exponential form is not allowed. For 1.23^5 use 123000.

Comments Comments are very important part of Macro programming. They explain the program to others, and remind the programmer why they wrote it the way they did. Comments on a macro line begin with an apostrophe (‘), and can extend to a total line length of 63 characters.

N3 # V100=1.2345 ‘V100 IS THE LENGTH OF THE PART 2

Calculations The mathematical operations of addition, subtraction, multiplication and division are represented by + , - , *(asterisk) , and / (slash), respectively.

EXAMPLE: *N11 # V10=3/V12*

Variables

Arrays Variable arrays can now be used; that is, the subscript of a variable can now be a variable or arithmetic expression. The subscript variables are D, FX, FY, FZ, H, PX, PY, PZ, PA, PB, R, and V. The subscript for a variable must not exceed the allowed range for the variable. For instance, the subscript for a V variable must be 1 through 99 and the subscript for a FX variable must be 1 through 48.

EXAMPLE: *If the variables V1=5 and V2=1 then the following are equivalent:*

#V5=1
#V(5)=1
#V(V1)=1
#V(V1+V2-1)=1
#V(V1/V2)=1
#V(V(SQR(V2))=1

AX, AY, AZ, AA, AB, AC: Axis Position Variables Axis position variables are used to read the current location of the desired axis.

EXAMPLE: *N79 # IF AZ >12.050 THEN GOTO :GETNEXT*

In the above example, if the Z axis is at 12.050 inches by the time this line is executed then the control will jump to the line in the program with the label :GETNEXT.

EXAMPLE: *N99 E2 X0 Y0*
N100 Z-30 G31 F50.
N101 G91 Z.05
N102 G90
N103 Z-30. G31 F1. The probe will stop the Z axis when it touches
N104 #IF AZ > 12.050 THEN GOTO :GETNEXT

CP: Clock from Power On This variable will return the accumulated time stored since power on.

CR: Clock for All Run Time This variable will return the accumulated time for all of the time while the control was in auto mode.

CC: Clock for Current Part This variable will return the accumulated time of the program currently being executed. The SETP page must have the TIMERS ON for the accumulated time to register.

D1 - D99: Tool Diameter/Radius Variables

These variables are used to read the current value of any diameter/radius from the tool table. It can also be used to establish or write a value in the table. The current value for any diameter can be changed by placing the variable on the left side of the equal sign.

EXAMPLE: **Reading Diameters**

N20 # IF D5 >.505 THEN GOTO :END
N...
N349 # :END
N350 M0
N350 #PRINT " TOOL DIAMETER #5 SHOULD BE NO LARGER THAN .505",

EXAMPLE: **Writing Diameters**

N89 # D2 = D2 + .01
This line will read the current value of diameter offset #2, then write a new diameter for #2, .01 larger, into the tool table.

FX1-FX48, FY1-FY48, FZ1-FZ48, FA1-FA48, FB1-FB48: Fixture Offset Variables

These variables are used to read the current value of each fixture offset. It can also be used to establish or write a value in the table. The current value of any fixture offset can be changed by placing the variable on the left side of the equal sign.

FY3 Read/write the Y of fixture three
FZ34 Read/Write the Z of fixture thirty four

EXAMPLE: Reading

N15 G91 G10 L02 X2. P1
N16 # IF FX1 >30. THEN GOTO :SUB3 Read the X value of fixture one
N...
N24 # :SUB3.
N25 G91 G10 L02 X-2. P1
N26 E0 X0 Y-1.
N27 L315

EXAMPLE: **Writing**

N63 # FX1 = FX1 - 2.

H1 - H99: Tool Length Offset Variables

These variables are used to read the current value of any tool length offset from the tool table. It can also be used to establish or write a value in the table. The current value for any tool length offset can be changed by placing the variable on the left side of the equal sign.

EXAMPLE: **Reading**

N34 # IF H16 <-10.67 THEN GOTO :NOTOOL
N35 # IF H16 = 0 THEN GOTO :STOPIT
N...
N144 # :NOTOOL
N145 #R6=V6+1
N146 M6 T+R6

EXAMPLE: **Writing**

N654 # H33 = H33 - 16.

I, G, O, HO, TL,VF: Macro Variables

The TL and VF macros are single variables returning the index of the tool offset table, and index of the fixture offset table. The I-, O-, HO- and G macros are arrays, returning inputs, manipulating spare outputs, relative user selected home positions, and the current value of the G codes. All inputs and outputs are available from the 1040-2A card.

Fadal reserves the right to use and change any input and output pins at any time.

EXAMPLE:
N10#PRINT I(3)
N20#O(2) = 1

EXAMPLE: **Layout of I Macro:**

The inputs 1 – 6 are user inputs available from the J2 connector on the 1040-2A card. Pin 1 on the J2 connector is given by a little arrow visible on top of the connector.

I(1)	Input 1	J2 pin 17	I(2)	Input 2	J2 pin 18
I(3)	Input 3	J2 pin 19	I(4)	Input 4	J2 pin 20
I(5)	Input 5	J2 pin 21	I(6)	Input 6	J2 pin 22
I(7)	Renishaw Probe		I(8)	Scale Error	
I(9)	Ext. Slide Hold		I(10)	Slide Home	
I(11)	Spindle Orient		I(12)	Turret return	
I(13)	Slide external		I(14)	Orient Arm Home	
I(15)	Low range out		I(16)	Indexer stop sw.	
I(17)	Indexer arm sw.		I(18)	ATC fault	
I(19)	Drawbar down		I(20)	High range out	
I(21)	Oiler		I(22)	Arm at Table	J2 pin 9
I(23)	Arm at storage	J2 pin 10	I(24)	Y Aligned	J2 pin 11
I(25)	Pallet at A	J2 pin 12	I(26)	Pallet at B	J2 pin 13
I(27)	Door Open	J2 pin 14	I(28)	Door Closed	J2 pin 15
I(29)	Inhibit	J2 pin 16			

The input values contained in the I-array are either 0 (the input is low/on) or 1 (the input is high/off).

EXAMPLE: **Layout of G Macro:**

G(l) (Group A G-codes) current interpolation type.

Contains G-code plus one as follows:

1 = (G0) point to point.

2 = (G1) linear.

3 = (G2) circular clockwise (CW).

4 = (G3) circular counter clockwise (CCW).

G(2) (Group B G-codes) current plane for circular algorithms.

G(3) (Group C G-codes) current cutter radius compensation mode.

G(4) (Group D G-codes) current Canned Cycle (CC) mode.

G(5) (Group E G-codes) current absolute/incremental mode

G(6) (Group F G-codes) current mode for return in Canned Cycle.

G(7) current location of spindle with respect to starting point (reference plane)

0 = spindle at 0 (initial plane)

1 = spindle at "R1'

EXAMPLE: **Layout of O Macro**

O(1) Output 1 J2 pin 6
O(2) Output 2 J2 pin 7
O(3) Output 3 J2 pin 8

The output is updated when a value is assigned to one of the slots in O-array. If a zero (0) is assigned to O(1), then the output on J2 pin 6 is pulled low. Any other value than zero assigned to the output will cause the output to float.

EXAMPLE: **Layout of HO Macro:**

The HO-array contains values that represent the location of user selected home relative to absolute position for all axes.

HO(1) X – axis HO(6) W – axis

HO(2) Y – axis HO(7) A – axis

HO(3) Z – axis HO(8) B – axis

HO(4) U – axis HO(9) C – axis

HO(5)V - axis

PX1 - PB1, PX2 - PB2, PX3 - PB3: Probe Touch Point Variables

These variables are used to read the current value of the touch points. PX1 would read the current X axis value of touch point P1. PY1 would return the value for the Y axis location of touch point P1 and PZ1 returns the Z axis value. The values for ONLY three touch points are available. The values for the X, Y, Z, A, and B axis may be read.

EXAMPLE

```
N197 #V1=1
N198 #R8=V1
N199 G1 G31 X20. Y25. F25. P+R1
N200 # PRINT “POINT #”,V1," IS X",PX1," Y",PY1
```

After touching a point with the probe, the position (P1) is now in memory and each axis position can be used for any calculation or can be output through the RS-232 port with the print command.

The printed line will read as follows:

POINT #1 IS X18.5 Y17.65

R0-R9: Parametric Variables These variables are used to pass data from the macro to the CNC program lines. Parametric parameters R0-R9 are also used for fixed subs and fixed cycles (see Parametric Programming).

EXAMPLE *N15 # R2 = V56*
N16 Z-R2 G1 F20.
N17 L9201 R0+.1 R1+0 R2+.25 R3+0 R4+1 Z-R9 F+R8 (ENGRAVING

TN: Tool Number This variable will return the numerical value of the current tool in the spindle. This number is determined by the last M6 T# executed from the program.

EXAMPLE: *N32 #V1=TN This would result in V1 being equal to the current tool number*

TT: Tool Time This variable will return the time entered in the time table for life expectancy of a specified offset.

Note: The TT and TU macro are active when the SETP parameters for tool time are enabled (On).One of the options (2-5) must be selected.

EXAMPLE: *N12 #V1=TT3 This will result in V1 being equal to the tool time entered in the time table*

TU: Tool Used TU variables accumulate the total amount of time that a specific tool has been in use. For example, if tool 5 is used for ten minutes, then TU5 = 10. If tool 5 is used for another seven minutes, then TU5=17. TU variables are retained between part runs and when powering down the machine.

After a tool change, declare a tool length offset (H1 - H99). The current tool length offset applied determines which TU variable will be used. For example, if H7 is in effect, then TU7 will be active. The TU variable increases whenever the control is in the interpolation mode only.

Note: The TT and TU macro are active when the SETP parameters for tool time are enabled (On) by choosing one of the options (see SETP for tool life management options).

EXAMPLE: *N31 #IF TU3=3 THEN GOTO :CHANGETOOL*
N300 #:CHANGETOOL

This will cause a jump to the label :CHANGETOOL when the time is equal to 3. Using the option 2 of tool life management in the SETP page allows the user to program the tool time displayed in the tool time table and the TU to read the used time in the tool time table. The combination of the TT and the TU macro allows the user to customize a macro to include a specialize tool life management application.

V1-V100: Macro Variables These variables are used in the macro lines.

EXAMPLE: *N88 # V1 = V2 + V3*

Functions

ABS ABS will return the absolute value of a number.

EXAMPLE: *V2 = ABS (V1)*
If V1 is equal to -2.3 then V2 would be 2.3, and if V1 is equal to 2.3, then V2 would be 2.3. The absolute value of a number is always the positive value of the number.

ATN ATN will return the Arc tangent of a number. See the SET command for degrees or radians.

EXAMPLE: *V5 = ATN (V6/V7) If V6 is the opposite side of a right angle and V7 is the adjacent side of the right angle, then V5 would be equal to the angle between the sides.* To calculate mathematical function Tangent of an angle use the following formula: TANGENT(angle) = SIN(angle) / COS(angle)

COS COS will return the Cosine of an angle. See the SET command for degrees or radians.

EXAMPLE: *V56 = COS (V34 + V72)*
If V34 and V72 represent angles, V56 would then be equal to the cosine of the sum of the two angles.
To calculate mathematical function Inverse Cosine of an angle use the following formula:

ARCCOS(angle) = 1570796-ATN(angle / SQR(1-angle*angle)

INT INT will return the integer of a number.

EXAMPLE: *V100 = INT(V23)*
If V23 = 12.345 then V100 = 12.
If V23 = -12.345 then V100 = -12.
If V23 = 12.513 then V100 = 12.
If V23 = -12.513 then V100 = -12.

The integer value uses only the whole number portion of the number.

RND RND will return a rounded value of a number. The number of places to round to is set using the #SET RND# command. To set the rounding to four places enter the command below.

EXAMPLE: *N5 #SET RND4 This line sets the number of places to round to*
N6 # V1 = RND(V1) This rounds the V1 value to 4 places

Note: When higher accuracy is desired, the rounding may be calculated. This eliminates the need to use the #SET RND# command. To round off numbers to the fifth place use this equation: V1 = INT(V5 * 10000)/10000. The value of V5 will be rounded to the fifth place and V1 will be used for the rounded number. This example only affects the individual line.

SGN SGN is used to determine the sign of a value.

EXAMPLE: *V50 = SGN (V77 - V78)*
If (V77 - V78) is a positive value, then V50 is +1.
If (V77 - V78) is zero, then V50 is +1.
If (V77 - V78) is a negative value, then V50 is -1.

SIN SIN will return the sine of an angle. See the macro SET command for selecting degrees or radians.

EXAMPLE: *V56 = SIN (V34 + V72)*
If V34 and V72 represent angles, V56 would then be equal to the sine of the sum of the two angles.
To calculate mathematical function Inverse Sine of an angle use the following formula: ARCSIN(angle) = ATN(angle) / SQR(1-angle*angle)

SIN/COS To calculate mathematical function Tangent of an angle use the following

formula: TANGENT(angle) = SIN(angle) / COS(angle)

EXAMPLE: *V12 = SIN(30) / COS(30)*
V12 would be equal to the tangent function of a 30 degree angle

SQR SQR will return the square root of a number. If the number is negative, an error
is printed and the program will halt.

EXAMPLE: *V46 = SQR (9) V46 would be equal to 3*
*V45 = SQR (V1 - (V63*V29)) V45 would be equal to the square root of (V1 - (V63*V29))*

Macro Commands Macro commands help direct each equation. Processing will stop if the commands are incorrectly used. Computational errors will cause an error message to be printed on the screen. The processing of macros can be executed step by step using the DEBUG command.

CLEAR CLEAR is used to zero macro variables. The variables may need to be cleared at the beginning of a macro routine or at some place within the program. The CLEAR macro can zero one variable at a time or a range or list of variables. If the CLEAR statement is left blank, ALL variables (V1-V100) are cleared.

Note: It is recommended that all variables be cleared at the start of each program.

EXAMPLE: *CLEAR This zeroes all macro variables*

EXAMPLE: *# CLEAR V1 This line will zero only variable V1*
 # CLEAR V1-V20 This line will zero variables V1 through V20
 # CLEAR V3-V7,V15,V30,V45-V60 This line will zero variables V3 through V7, then V15 and V30, then V45 through V60

GOTO The GOTO statement is used to redirect the program. When used separately, the program is redirected to the line number or label indicated.

EXAMPLE: *N3 #V1= 1 This sets V1 equal to 1*
N4 # IF V1 >2 THEN GOTO :INSIDE
This checks the value of V1. If V1 is less than or equal to 1 continue at N5, if the value of V1 is greater than 1 then continue at label :INSIDE
N5 X1.0 Y-2.3 G1
N6 X3.6 Y-3.0
N15# :INSIDE This is label :INSIDE

*N16 # R3 = V2 * V17 This line transfers the multiplication of V2 * V17 to R3 parameter*
N25 X+ R3 F20. G1 This transferred R3 to X
N26 #GOTO :INSIDE Continue at label :INSIDE at N15

Labels Labels are used to identify a GOTO location in the program. It is usually best to choose labels that are descriptive of the area or instruction to which they are assigned. The colon (:) assigns the LABEL field.

EXAMPLE: *N5 #V1=1 This sets the value of V1 to be 1*
. . .
N9 #GOTO :DOMATH This sends the operation to line N100 to label DOMATH
N10 #:WORK This the label WORK the end destination of line N110
N11 #V1=V1+1 This changes variable V1 by adding 1 to its value
N12 #IF V1=5 THEN GOTO :DOMATH
This checks the value of V1 If V1 is equal to a value of 5 it goes to N100, label DOMATH,if not the program continues at line N14
N13 #R9=R9+V1 This line transfers the value of V1 to parameter R9
N14 T+R9 M6 This line transfers the value R9 to the Tool number
N...
N100 #:DOMATH This is the end destination from the GOTO on line N5.
N...
N110 #GOTO :WORK This sends the operation to line N10 to label WORK
Labels may not have a comment on the same line. See below:

N2 #:TEST 'THIS IS A TEST A comment is not allowed on a label line

IF - THEN The IF-THEN command is used for comparisons. The IF part of the macro line will examine a variable or equation and if it is true, it will execute the THEN part of the macro line. If the IF part of the macro is not true, it will not execute the THEN part of the macro line and will continue with the next line in the program. When N words are used for the THEN part of the statement, they are NOT renumbered when the NU command is used in the control. If the program is renumbered, then the N word must be altered to match the program.

Note: The THEN part of the statement may also use labels. See Label definitions.

EXAMPLE: **Example 1:**

N200 # IF V1 > V2 THEN GOTO :PART3
N201 E2 X0 Y0
...
N299 # :PART3
N300 E3 X0 Y0

In line N200, if it is true that V1 is greater than V2, then the program will jump to line number N300. If V1 is not greater than V2 the program will skip to line N201.

The THEN part of the statement may contain any valid macro statement.

EXAMPLE: **Example 2:**

N100 # IF (V5 + V6) <= 0 THEN V7 = ABS(V5 + V6)

In line N100, if the sum of V5 plus V6 is less than or equal to zero, then let the value of V7 be the absolute value of the sum of V5 plus V6.

INDEX The INDEX macro is used to send indexer code directly to the FADAL indexer. This feature is only available when used with the FADAL indexer. For more information on the FADAL indexer see the indexer manual.

EXAMPLE: *N5 # INDEX "M1 A90. F300." This sends CNC code to the FADAL indexer to move the indexer 90 degrees at a feed rate of 300*

INPUT The INPUT command is used to allow the operator to enter program data during execution of the macro. When the INPUT statement is executed, the program processing stops until the operator presses the ENTER key. The PRINT command may be used to prompt the operator for the desired data to enter. Placing a comma at the end of the PRINT statement will move the cursor to the end of the text on the screen.

EXAMPLE: *N1 #CLEAR*
N2 #PRINT "ENTER THE POCKET LENGTH",
N3 #INPUT V1
N4 # PRINT "ENTER THE POCKET WIDTH",
N5 #INPUT V2

These statements prompt for the V1 and V2 data required for the program. The program suspends operation until each value is entered.

LABELS Labels designate a place in the program where program execution may be directed when preselected conditions have or have not been met. Labels are unaffected by program renumbering and take the form #:LABEL. Nothing else may appear on the line. The call to a label will include the colon followed by the name of the label. Each label in a program must have a unique name. If there are any duplicate names, program execution will always go to the first label in the program. Labels may be any alphanumeric the programmer chooses. Labels must be preceded by a colon (:) and there may be no comments or other code in the line with the label. A label indicates a place in the program where program execution may continue after a jump. This may be a routine that is to be repeated several times, or a routine that is to be executed only when certain conditions are met. To make a program easier to read, labels should describe the operation taking place. Such as :LOOP :JUMPBACK.

PRINT The PRINT statement is used to print text and data to the screen.

Note: Text that is to be displayed MUST be enclosed in quotation marks.

EXAMPLE: *N4 #V6=25.45*
N5 #PRINT "THE VALUE CALCULATED: ",V6,", IS THE X VALUE."

Using a comma to separate the variable from the text, the screen display is as follows:

THE VALUE CALCULATED: 25.45, IS THE X VALUE.

Note: Do not use a semicolon to separate the variable from the text.

SET The SET command is used to change macro parameters. There are five parameters to establish using the SET command. They are DEBUG, RUN, RND#, DEGREES, and RADIANS. Setting DEGREES or RADIANS affects the SIN, COS, and ATN commands. The SET commands are inputted after entering the machine command MA at the command prompt or in the macro program. The DEGREES and RADIANS parameters may also be set within a macro as described below. The SET command, without specifying the setting function, restores all SET parameters to the default.

EXAMPLE: **MA Example:**

MA
SET RND5
The "#" sign is not required when using the MA command. This Command is typed at ENTER NEXT COMMAND Prompt.

EXAMPLE: **Macro Example:**

N24 # SET RADIANS
N25 # V2 = ATN(V1)
N26 # PRINT, V2
N27 # SET DEGREES
N28 # V2 = ATN(V1)
N29 # PRINT, V2

Line N26 prints the angle V2 in radians. Line N28 prints the angle V2 in degrees.

SET DEBUG The debug command is used temporarily in the macros to proof the macro lines. Debug mode works only when SU from the command mode is used. The advantage to using the debug mode is that it will display the values of the variables. To start the Debug mode, enter #SET DEB in the macro program or SET DEB using the MA command. To end the Debug mode, to enter SET RUN using the MA command or enter #SET RUN in the macro program.

SET DEGREES / RADIANS SET DEG and SET RAD commands are modal and cancel each other. When using the SIN, COS, or ATN functions the calculations are based upon the DEGREE or RADIANS setting. The default setting is degrees.

*Radians = ((Angle in Degrees * PI) / 180)*
*Degrees = ((Angle in Radians * 180) / PI)*

PI=3.14159265

SET RND# This function will return a rounded value of a number. The number of places to round to is set using the SET RND# command. To set the rounding to four places enter the command below. The maximum number of places to round to is five.

EXAMPLE: *MA*
SET RND4

Note: When higher accuracy is desired, the rounding may be calculated. This eliminates the need to use the SET RND# command. To round off numbers to the fifth place use this equation: V1 = INT(V5 * 10000)/10000. The value of V5 will be rounded to the fifth place and V1 will be used for the rounded number. This example only affects the individual line.

SET RUN This command is used to exit the DEBUG mode. Using the MA command, enter SET RUN to exit the DEBUG mode or enter #SET RUN in the macro program. The program may then be executed.

SINPUT The SINPUT command is used to wait for and accept data though the RS-232 port during execution of a macro. When the SINPUT statement is executed, the program processing stops until the control receives data through the RS-232. The control will not look ahead of the line with the SPRINT macro command. The LOOK AHEAD process begins when data is received through the RS-232 port. The data sent to the port must terminate with a carriage return.

EXAMPLE:
N1 #CLEAR
N2 #PRINT "WAIT ! NOW ENTERING THE POCKET LENGTH DATA",
N3 #SINPUT V1
N4 #PRINT "WAIT ! NOW ENTERING THE POCKET WIDTH DATA",
N5 #SINPUT V2

SPRINT The SPRINT command is used to send data out through the RS-232 port. Text and variable data can be sent. The SPRINT statement prints through the RS-232 port the same as the PRINT statement displays to the screen.

Note: Text must be enclosed in quotation marks.

EXAMPLE:
N43 # SPRINT "PART NUMBER ",V4," DATA"
N44 # SPRINT "X ",AX," Y ",AY

In line N43 the words, part number, and data are considered to be text. If the value of V4 is 98645 then N43 would print out:

PART NUMBER 98645 DATA

START # The START macro is used to jump to another program. This is used in a situation where the program being called has subroutines. Subroutines are not allowed in a subprogram, this is the reason why this START command is used. This command can be used to link many user main programs. It executes a jump only. No return will occur. The START command would generally be used in place of an M2 or M30 at the end of each program being linked. When the next program is called, the operator must press the AUTO button again to run the next program.

EXAMPLE: *N3245 G0 G90 G49 Z0*
N3246 E0 X0 Y0
N3247 #START 7 This calls a jump to program O7
WAIT

This command temporarily pauses processing of the program lines at the line with the WAIT command. Processing will continue when the execution buffer is completely exhausted. This might be used to Print out a message when the program is completely finished. The control will print out through the RS-232 port during the preprocess period unless the WAIT command is used. The WAIT command is also used to have the machine absolutely stationary before printing out the current location of the machine using the axis variables.

EXAMPLE: *N256 G1 G31 Z-30. F1.*
N257 # WAIT
N258 # SPRINT "POINT #",V5,": X",AX," Y",AY

In line N257 the control will stop processing the program and wait for the execution buffer to be exhausted before executing line N258 which would print out the current location of the machine.

Note: The execution buffer is a part of memory used by the control to store processed program data. When the control "looks ahead", this processed data is stored in the execution buffer. Using the SETP command allows the programmer to vary the size of the buffer. On the machine parameter page the term "Binary buffer" is used for the execution buffer.

AND, OR, and NOT AND, OR, and NOT are logical operators that allow the programmer to construct compound tests from one or more expressions. In BASIC, a compound Boolean expression is created by connecting two Boolean expressions with a "logical operator". The FADAL macro language allows "logical operators". The number of logical operators in a statement is limited by the maximum number of characters allowed on a single line, 63. This includes all spaces and the pound sign (#).

A good rule for writing compound expressions is to always verify that the statement will evaluate to both a TRUE and a FALSE condition. The items being compared must be segregated with parentheses to avoid confusion in lengthy or complex statements. This will also ensure that the statement's components are evaluated in the order that the programmer intended.

EXAMPLE: **Example 1:**

AND

IF (V1 GT V2) AND (V1 LT V3) THEN GOTO :LOOP
This first example is true only if expression 1 and expression 2 are both true, then control jumps to the label :LOOP.

EXAMPLE: **Example 2:**

OR

IF (V1 GT V2) OR (V1 LT V3) THEN GOTO :LOOP
In this example, either condition can be true for control to jump to the label :LOOP.

EXAMPLE: **Example 3:**

NOT
IF NOT(V1=0) THEN GOTO :LOOP
In this example if V1 is NOT equal to zero then control jumps to the label :LOOP.

EXAMPLE: **Example 4:**

AND, OR, and NOT
IF (V1 LT V2) OR (V1 LT V3) AND (NOT(V1 GT V4)) THEN GOTO :LOOP

The fourth example shows multiple operators strung together in one statement. Combinations of AND, OR, and NOT can be used to build up very complex expressions. There is no practical limit to their complexity, except the obvious one of writing an expression that may be difficult to read and exceeds 63 characters.

Macro Tutorial

Overview This subsection is designed as a tutorial for FADAL's Macro Language. Several Macro Language commands are referenced here. For a complete list with applicable syntax, review the Macro Commands section.

Summary Macro programming uses a subset of BASIC to manipulate data for use in G code programs. If a shape can be defined with an equation or all of its elements with variables, then it may be advantageous to use macro programming.

The availability of compare statements and algebraic operators in the form of IF THEN loops and +, -, *, /, SIN, COS and ATN (to list a few), makes it possible to create complex geometry with a few lines of code.

This discussion will include examples of macro programs with their explanations and, where appropriate, the equations from which the geometry is derived. It will start with simple examples and proceed through to the more complex functions.

Conventions All macro lines must be preceded by a pound sign # immediately following a line number. The open paren (and asterisk * are macro operators and cannot be used in a macro statement to denote a comment. Instead use an apostrophe ‘ to denote a comment in a macro line. The open and closed parens (|) are grouping operators used to indicate order of operation, use as in an algebraic equation. The asterisk * is a multiplication symbol.

Comments Comments are very important part of macro programming. They explain the program to others, and remind the programmer why they wrote it the way they did. Comments on a macro line begin with an apostrophe (‘), and can extend to a total line length of 63 characters.

N3 # V100=1.2345 ‘V100 IS THE LENGTH OF THE PART 2

Calculations The mathematical operations of addition, subtraction, multiplication and division are represented by +, -, * (asterisk), and / (slash), respectively.

EXAMPLE: N11 # V10=3/V12

Variables

V1-V100: Macro Variables There are 100 available variable or ‘V’ registers numbered from V1 to V100. These registers may be used to manipulate data but may not be used directly in the G code program. The manipulated values must be passed to the R registers for use in the program. There are ten R registers numbered R0 to R9. Since the control uses R0 through R4 in various fixed cycles, it is usually advantageous to pass values to the R registers starting with R9 and proceeding backwards to R0. Be careful not to ‘step on’ any values in use. Re-assign values if necessary.

These variables are used in macro language statements.

EXAMPLE: *N88 # V1 = V2 + V3*

Arrays Variable arrays can now be used; that is, the subscript of a variable can now be a variable or arithmetic expression. The subscript variables are D, FX, FY, FZ, H, PX, PY, PZ, PA, PB, R, and V. The subscript for a variable must not exceed the allowed range for the variable. For instance, the subscript for a V variable must be 1 through 99 and the subscript for a FX variable must be 1 through 48.

EXAMPLE: If the variables V1=5 and V2=1 then the following are equivalent:

#V5=1
#V(5)=1
#V(V1)=1
#V(V1+V2-1)=1
#V(V1/V2)=1
#V(V(SQR(V2))=1

AX, AY, AZ, AA, AB, AC: Axis Position Variables

Axis position variables are used to read the current location of the desired axis.

N79 # IF AZ >12.050 THEN GOTO :GETNEXT

In the above example, if the Z axis is at 12.050 inches by the time this line is executed, then the control will jump to the line in the program with the label :GETNEXT.

N99 E2 X0 Y0
N100 Z-30 G31 F50.
N101 G91 Z.05
N102 G90
N103 Z-30. G31 F1. The probe will stop the Z axis when it touches
N104 #IF AZ > 12.050 THEN GOTO :GETNEXT

CLEAR The CLEAR statement is used to zero the values in the variable table. If used alone it will clear all of the variables in the table. If a variable number is included

EXAMPLE: *#CLEAR Clears all V registers OR*
#CLEAR V1 Clears only V1 OR
#CLEAR V10-V30 Clears V10 through V30

In normal programming practice a CLEAR statement at the beginning of the program will ensure that all of the variables will be zeroed. This is done to prevent a variable register that the program will use from having a value that may shorten a loop sequence or exceed the test value.

For example, if a statement reads #IF V1 <|> 10 THEN GOTO :JUMP with the intent of looping 10 times, and the value of V1 is 15 from a previous program, the program will always go to the label :JUMP. This line is an example of a bad programming practice. The only way a loop written with this statement will ever end would be if V1 were equal to 10. If the loop were counting up it would be better written as #IF V1 <= 10. In this form once the count exceeded 10 the program would drop out of the loop. If a specific value is required in a register it must be entered by the programmer with an assign statement.

To assign a value to a V variable use the equal sign. Ex: #V1=10 #V1=V2 #V1=R9 #V1=SIN(V10) any valid Macro function may be used to assign a value to a Macro variable. The R variables may also be assigned values in the same manner as a V variable. In addition, values may be directly assigned outside of a Macro statement using a + or - . The value passed to an R variable in this manner must be either a number or another R variable.

EXAMPLE: *R9+10. R8-18. R7+R6*

Operator Interaction

PRINT PRINT statements are used to display messages on the screen for the operator. Anything written that will not change must be enclosed in quotes. Any values represented by V or R variables must be outside of the quotes (") and separated by a comma (,).

#PRINT V10
#PRINT "CHANGE PART"
#PRINT "THIS IS PART NUMBER ",V10
#PRINT "THIS IS PART NUMBER ",V10,"THERE ARE ",V11,"PARTS LEFT"

SCREEN DISPLAY:

36920
CHANGE PART
THIS IS PART NUMBER 36920
THIS IS PART NUMBER 36920 THERE ARE 121 PARTS LEFT

INPUT An INPUT statement requires that a variable register be specified to receive the input value, this may be requested by the PRINT statement. The operator will enter a requested value and that value will be placed in the variable register specified in the INPUT statement then continue program operation. Pressing the return key without entering a value will enter a zero (0) in the variable register specified and continue program execution.

The INPUT statement is used to halt program execution to receive data input and to display a print statement. It is assumed that a print statement will request some action from the operator. A PRINT statement only displays to the screen. PRINT will not halt program execution and therefore may scroll off of the screen before the operator can read it.

EXAMPLE: *#PRINT "ENTER THE DIAMETER OF THE END MILL"*
#INPUT V100

Program Branching

LABELS Labels designate a place in the program where program execution may be directed when preselected conditions have or have not been met. Labels are unaffected by program renumbering and take the form #:LABEL. Nothing else may appear on the line. The call to a label will include the colon followed by the name of the label. Each label in a program must have a unique name. If there are any duplicate names, program execution will always go to the first label in the program. Labels may be any alphanumeric the programmer chooses. Labels must be preceded by a colon **(:)** and there may be no comments or other code in the line with the label. A label indicates a place in the program where program execution may continue after a jump. This may be a routine that is to be repeated several times, or a routine that is to be executed only when certain conditions are met. To make a program easier to read, labels should describe the operation taking place. Such as :LOOP :JUMPBACK.

GOTO The GOTO statement may also be used in a line by itself to redirect program execution. In this form it is an unconditional jump, meaning there are no conditions to be met for the jump. Program execution is directed to the specified line or label.

EXAMPLE *#GOTO N45 OR*
#GOTO :LABEL

These lines will cause program execution to branch to the line or label specified. GOTO statements may address a line number directly; however, if the program is re-numbered it will be necessary to verify that all of the addresses have not changed. The control WILL NOT update any N words in GOTO statements when a program is re-numbered. A preferred method is to use macro labels.

IF-THEN The IF-THEN statement provides the FADAL macro programming language the flexibility necessary for compact, powerful programs. It takes the form IF some condition is true THEN do something or go to an address.

The condition to be met may be a simple statement comparing some A to some B:

EXAMPLE: *#IF V1=V2*
This compare statement looks for equal values in variables V1 and V2
#IF V1=8

This compare statement checks to see if the content of variable V1 is equal to 8

Some valid equalities: V1=R1 R1=V1 V1=SIN(V2) V5=TN. The condition may also be a Boolean Equation.

EXAMPLE *#IF V1*V2=3 Which reads: if V1 times V2 equals V3*
or
*#IF V1*V2< >V3 Which reads: if V1 times V2 is not equal to V3*

Symbolic Operators The Boolean operators used in the FADAL Macro language are:

=	*or*	*EQ for Equal*
<	*or*	*LT for Less than*
>	*or*	*GT for Greater than*
<=	*or*	*LE for Less than or equal to*
>=	*or*	*GE for Greater than or equal to*
<|>	*or*	*NE for Not equal to*

The THEN portion of the statement may be used to assign or re-assign a value to a variable or register.

EXAMPLE *THEN V1=V5 or THEN R6=SIN(V34)*

The THEN statement may also be used to re-direct program execution.

EXAMPLE *THEN GOTO N45 Re-directs program execution to line number N45*
or
THEN GOTO :LOOP Re-directs program execution to the label :LOOP If program execution is re-directed to a line number, it will be necessary to verify that the line number is correct any time the program is re-numbered. When the program is re-numbered the control WILL NOT update the line numbers in an IF THEN loop or a GOTO statement. For this reason it is preferable to use labels to re-direct program execution.

A complete IF THEN statement will take the form:

#IF V1 <= 28 THEN GOTO :LOOP

This line reads, if the contents of variable register V1 are less than or equal to 28, then go to LOOP. This line could be used in a program with V1 as a counter to perform some operation until the value of V1 is greater than 28, once the

value of V1 exceeds 28, program operation will continue with the program line following the IF THEN statement.

#IF V1 = 1 THEN V20 = V5

This line reads, if the contents of variable register V1 are equal to 1, then make the contents of variable register V20 equal to the contents of variable V5. This IF THEN statement would be used to set or reset the value of variable register V20 to the value in V5 once V1 is set to 1.

AND, OR, and NOT Logical Operators

AND, OR, and NOT are logical operators that allow the programmer to construct compound tests from one or more expressions. In BASIC, a compound Boolean expression is created by connecting two Boolean expressions with a “logical operator”. The FADAL macro language allows “logical operators”. The number of logical operators in a statement is limited by the maximum number of characters allowed on a single line, 63. This includes all spaces and the pound sign (#).

A good rule for writing compound expressions is to always verify that the statement will evaluate to both a TRUE and a FALSE condition. The items being compared must be segregated with parentheses to avoid confusion in lengthy or complex statements. This will also ensure that the statements components are evaluated in the order that the programmer intended.

EXAMPLE: **Example 1:**

AND
IF (V1 GT V2) AND (V1 LT V3) THEN GOTO :LOOP
This first example is true only if expression 1 and expression 2 are both true, then control jumps to the label :LOOP.

EXAMPLE: **: Example 2:**

OR
IF (V1 GT V2) OR (V1 LT V3) THEN GOTO :LOOP
In this example either condition can be true for control to jump to the label :LOOP.

EXAMPLE: **Example 3:**

NOT
IF NOT(V1=0) THEN GOTO :LOOP
In this example if V1 is NOT equal to zero then control jumps to the label :LOOP.

EXAMPLE: **Example 4:**

AND, OR, and NOT
IF (V1 LT V2) OR (V1 LT V3) AND (NOT(V1 GT V4))
THEN GOTO :LOOP

The fourth example shows multiple operators strung together in one statement. Combinations of AND, OR, and NOT can be used to build up very complex expressions. There is no practical limit to their complexity, except the obvious one of writing an expression that may be difficult to read and exceeds 63 characters.

Counting Loops Counting loops may be initiated to count the number of parts machined, holes drilled, etc. The basic code generally takes the form:

 # V49 = V49 + 1
 # D98 = D98 + 1

The values may be placed in any unused register, for example, any valid V, D, R, H, TT, or FO register. When using a register for counting, care must be taken to insure that the value in the register is not overwritten by another function in use by the control.

Inaccuracies in the count may be caused by the look-ahead feature in the control. This may be overcome with the use of a macro WAIT statement in the line preceding the count statement. The WAIT statement does not cause the machine to stop, it stops look ahead past this point until the program execution actually reaches this point in the program. This will keep the counter from incrementing until the event being counted actually occurs.

Counting loops are best placed at the end of the program.

FADAL Macro Language Examples

D-Hole Macro

1) Move to the center of the D-Hole.

   a. This move can be made incrementally, but this macro switches to absolute. So remember, if you are programming in incremental, enter the G91 code on the next line of the program after the call to this macro.

2) Position the Z axis to the desired Z depth.

3) Establish a feed rate.

4) Use a D word to establish the tool diameter offset to use.

   a. This macro assumes that the tool diameter is entered into the tool table.
   b. For tool tables set up for radius Y, enter the radius.

5) Type the R words and M98 P800.

   a. R5: Use 0 for CW, and 1 for CCW
   b. R6: Circle radius
   c. R7: Blend radius
   d. R8: Distance to the line
   e. R9: Angle

EXAMPLE:

```
O1 (D-HOLE EXAMPLE: MAIN PROGRAM
(TOOL #1, .5 DRILL
M6 T1
G90 G0 S5000 M3 E1 X2. Y-3.
H1 Z.1 M8
G81 G99 R0+.1 Z-.6 F40. X2. Y-3.
G80
M5 M9
G90 G0 G49 Z0
(TOOL #2, .5 HSS 2FL EM
M6 T2
G90 G0 S7000 M3 E1 X2. Y-3.
H2 Z.1 M8
Z-.3 G1 F30.
F45.
D2
R5+1. R6+2. R7+.5 R8-.7 R9+0 M98 P800
Z.1 G0
M5 M9
G49 Z0
E0 X0 Y0
     M2
```

Sub Program 800

```
N1O800(D-HOLE MACRO: SUB PROGRAM
N2#'1ST MOVE TO D HOLE CENTER, THEN POS. Z AXIS)
N3#'R5=0 FOR CW, 1 FOR CCW)
N4#'R6=RADIUS OF CIRCLE
N5#'R7=BLEND RADII
N6#'R8=X DIR & DIS TO LINE
N7#'R9=ANGLE
N8#V1=AX 'GET CURRENT X POSITION
N9#V2=AY 'GET CURRENT Y POSITION
N10#V3=R5 'CW OR CCW
N11#V4=R6 'RADIUS OF CIRCLE
N12#V5=R7 'BLEND RADIUS
N13#V6=R8 'X DIR & DIS TO LINE
N14#V8=V4-V5
N15#V10=-(ABS(V6)-V5) 'X CENTER OF BLEND
N16#V11=SQR((V8*V8)-(V10*V10)) 'Y CENTER OF BLEND
N17#V12=V11/V10 'SLOPE OF LINE THROUGH CENTER OF CIRCLE AND
CENTER
```

```
N18#V13=-SQR((V4*V4)/(1+(V12*V12))) ‘X END PNT ON BND RAD
N19#V14=V12*V13 ‘Y END PNT ON BND RAD
N20#IF R5=0 THEN GOTO :CW
N21#R3=V6 ‘FIRST X MOVE TO LINE
N22#R4=0
N23 G90 G8
N24 M98 P900
N25 X+R5 Y+R6 G1
N26#R3=V6
N27#R4=-V11 ‘Y MOVE TO BND RAD
N28 M98 P900
N29 X+R5 Y+R6
N30#R1=R5
N31#R2=R6
N32#R3=V10
N33#R4=-V11
N34 G8
N35 M98 P900
N36#R1=R5-R1 ‘I TO B.R. CENTER
N37#R2=R6-R2 ‘J TO B.R. CENTER
N38#R3=V13 ‘X BND RAD END POINT
N39#R4=-V14 ‘Y BND RAD END POINT
N40 M98 P900
N41 X+R5 Y+R6 I+R1 J+R2 G3
N42#R1=V1-R5
N43#R2=V2-R6
N44#R3=V13
N45#R4=V14 ‘Y CIRCLE END POINT
N46 M98 P900
N47 X+R5 Y+R6 I+R1 J+R2
N48#R1=R5
N49#R2=R6
N50#R3=V10
N51#R4=V11
N52 M98 P900
N53#R1=R5-R1
N54#R2=R6-R2
N55#R3=-V6 ‘X BND RAD END POINT
N56#R4=V11 ‘Y BND RAD END POINT
N57 M98 P900
N58 X+R5 Y+R6 I+R1 J+R2
N59#R3=-V6
```

```
N60#R4=0
N61 M98 P900
N62 X+R5 Y+R6
N63#R3=0
N64#R4=0
N65 M98 P900
N66 X+R5 Y+R6 G40
N67 M99
N68#:CW
N69#R3=V6 'FIRST X MOVE TO LINE
N70#R4=0
N71 G90 G8
N72 M98 P900
N73 X+R5 Y+R6 G1
N74#R3=V6
N75#R4=V11 'Y MOVE TO BND RAD
N76 M98 P900
N77 X+R5 Y+R6
N78#R1=R5
N79#R2=R6
N80#R3=V10
N81#R4=V11
N82 M98 P900
N83#R1=R5-R1 'I TO B.R. CENTER
N84#R2=R6-R2 'J TO B.R. CENTER
N85#R3=V13 'X BND RAD END POINT
N86#R4=V14 'Y BND RAD END POINT
N87 M98 P900
N88 X+R5 Y+R6 I+R1 J+R2 G2
N89#R1=V1-R5
N90#R2=V2-R6
N91#R3=V13
N92#R4=-V14 'Y CIRCLE END POINT
N93 M98 P900
N94 X+R5 Y+R6 I+R1 J+R2
N95#R1=R5
N96#R2=R6
N97#R3=V10
N98#R4=-V11
N99 M98 P900
N100#R1=R5-R1
N101#R2=R6-R2
```

N102#R3=V6 'X BND RAD END POINT
N103#R4=-V11 'Y BND RAD END POINT
N104 M98 P900
N105 X+R5 Y+R6 I+R1 J+R2
N106#R3=V6
N107#R4=0
N108 M98 P900
N109 X+R5 Y+R6
N110#R3=0
N111#R4=0
N112 M98 P900
N113 X+R5 Y+R6 G40
N114 M99

Sub Program 900

N1O900(ROTATE X & Y)
*N2#V30=(R3*COS(R9)-R4*SIN(R9))*
*N3#V31=(R3*SIN(R9)+R4*COS(R9))*
N4#R5=V30+V1
N5#R6=V31+V2
N6 M99

Row Column Pattern Macro

1) Start the fixed cycle

2) Type the R words:

R9 = Number of holes across R7 = Number of holes down
R8 = Space between X axis holes R6 = Space between Y axis holes

EXAMPLE:

O10(ROW & COLUMN: MAIN PROGRAM#CLEAR
#SET RUN
G90 G0 G40 G80 G17
E0 X0 Y0
G90 G0 S1000 M3 E1 X0 Y0
H1 Z0.1 M8
G81 G99 Z-1.R+0.1 F100. M45
R9+3.R8+1.R7+3.R6+1.
M98 P810
G80
M5 M9
G90 G0 G49 Z0
E0 X0 Y0
M2

Sub Program 810

```
O810(ROW & COLUMN: SUBPROGRAM
#'R9 IS THE NUMBER OF HOLES ACROSS
#'R8 IS THE SPACE BETWEEN X HOLES
#'R7 IS THE NUMBER OF HOLES DOWN
#'R6 IS THE SPACE BETWEEN Y HOLES
#V1=R9
#V2=R8
#V3=R7
#V4=R6
#V5=V1
#R5=V5
#V50=0'V50 IS THE HOLE COUNTER ACROSS
#V60=0'V60 IS THE HOLE COUNTER DOWN
#V70=0'V70 IS LAST ROW COUNTER
G91
#:RIGHT
#IF V50=V1 THEN GOTO:BACK
#V50=V50+1
X+R8
#GOTO:RIGHT
#:BACK
#V60=V60+1
#V50=0
#R4=(R9*R8)-R8
X-R4 Y-R6
#IF V60=V3 THEN GOTO:LAST
#GOTO:RIGHT
#:LAST
X+R8
#V70=V70+1
#IF V70=V5 THEN GOTO:END
#GOTO:LAST
#:END
M99
```

Spiral Cut Macro

```
O5757
#CLEAR Clears all macro variables
G0 G90 G80 G40 G49
S1800 M3 M7
G0 G90 X0 Y0
H1 Z.1
G1 Z-.25 F20.
```

#V7=0 - Sets V7 to zero
#:LOOP - Label :LOOP
#V7=V7+1 - Add 1 to V7
#IF V7>=360 THEN V7=0 Reset angle if over 360 degrees
#V1=V1+.00077 Set radial increments per degree
*#V2=SIN (V7)*V1 Calculate X component*
*#V3=COS(V7)*V1 Calculate Y component*
#R9=V2 Transfer V2 value to R9
#R8=V3 Transfer V3 value to R8
#R7=V1 Transfer V1 value to R7
G2 X+R9 Y+R8 F30. R+R7 Machines part to macro calculations
#IF V1<10. THEN GOTO :LOOP Stop when 10. inches
N20 G0 Z.1 M5 M9
N21 G28
N22 M2

Drill Grid Pattern Macro – Whole Number Increments

The following is a simple program to drill a grid of 100 holes in a 10 by 10 pattern. The program begins with normal G code programming, then clears the values in the variable registers to ensure a proper count for our loops. It uses the variable V1 for the X position counter and V2 for the Y position counter. These values will be passed to the R9 and R8 registers for use in the G code portion of the program. Only one label will be required for the loop. The first IF THEN statement will check to see when all of the holes in the X axis are drilled. When the count reaches 10, program execution will 'fall out' of the loop. The next statement zeroes (re-initializes) the X axis counter and the following statement increments the Y axis value. The second IF THEN statement checks to see when all of the rows in the Y axis have been drilled. In this program the values in the variable registers serve as counters as well as the value to increment for each X and Y move.

O1234(DRILL GRID 10-1" X STEPS, 10-1" Y STEPS
G90 G80 G40 G0 Z0
S2500 M3 M7
G0 X0 Y0 E1
Z.1 H1
G81 X0 Y0 Z-.75 R0.1 G98 F20.
#CLEAR Sets all variable registers to zero
#:LOOP Label for the loop
#V1=V1+1 Increases the value in V1 by 1
#R9=V1 Assigns the value in V1 to R9
#R8=V2 Assigns the value in V2 to R8
X+R9 Y+R8 Moves to locations specified by R8 and R9

#IF V1<=9 THEN GOTO :LOOP Tests if 10 holes have been drilled
#V1=0 Resets V1 to zero for the next row
#V2=V2+1 Increases the value in V2 by 1
#IF V2<=9 THEN GOTO :LOOP Tests if 10 rows have been drilled
G80 M5 M9
G49 Z0
G0 E0 X0 Y0 Z0
M2

This program drills 100 holes with 20 lines of code. An equivalent G code program would use approximately 109 lines to accomplish the same thing.

Drill Grid Pattern Macro – Decimal Increments

If it is necessary to program the moves in other than whole number increments, then it will be necessary to change the program format only slightly. The R9 and R8 values will be incremented by the desired values and must be initialized prior to the loop statement. The variables V1 and V2 are retained as counters.

O1235(DRILL GRID 10-.375" X STEPS, 10-.5" Y STEPS
G90 G80 G40 G0 Z0
S2500 M3 M7
G0 X0 Y0 E1
Z.1 H1
G81 X0 Y0 Z-.75 R0.1 G98 F20.
R9+0 R8+0 Clears the R9 and R8 registers
#CLEAR Sets all variable registers to zero
#:LOOP Label for the loop
#V1=V1+1 Increases the value in V1 by 1
#R9=R9+.375 Increases the value in R9 for the next hole
X+R9 Y+R8 Moves to the locations specified by R8 and R9
#IF V19 THEN GOTO :LOOP Tests if 10 holes have been drilled
#V1=0 Resets V1 to zero for the next row
#V2=V2+1 Increases the value in V2 by 1
#R9=0 Resets R9 to zero to start at beginning of line
#R8=R8+.5 Increases the value in R8 for the next row
#IF V29 THEN GOTO :LOOP Tests if 10 rows have been drilled
G80 M5 M9
G49 Z0
G0 E0 X0 Y0 Z0
M2

Tutorial Program Summaries

Synopsis The following programs have been designed as an integral part of this tutorial. Each program is summarized in this section and then each program is explained line-by-line in the following section. The last section contains listings of each program.

Program Number 1 This is a simple program to check the length of tool number 1 and enter this length into the tool offset table. Fixture offset number 23 must contain the distance from the Z zero surface to the top of the tool probe. Fixture offset number 24 must contain the X and Y location of the tool probe.

Program Number 2 This program is similar to program 1 in that it uses the TS-20 or TS-27 tool setting probe to check for tool condition. The function of this program is to check for broken tools.

Program Number 3 This program gets the tool number for the tool in the spindle then uses that tool to drill a set of holes. It then checks the time that the drill is cutting against the time entered in the tool table. If the time is less than the total time in the table then the routine will continue. Once the time has exceeded the time allowed, the program will increment the tool number and load the next tool. This process will continue until all tools in the tool changer have been used. In order for this program to function properly, the timers must be set to 2)DO NOT CHECK. This function is found on page two of the parameters. In addition, all tools to be used must be loaded and their respective tool lengths entered in the tool table.

Program Number 4 This program takes the diameter of a large drill then calculates the amount of each peck and the minimum dwell time required to relieve pressure on the drill. This allows the machine to more efficiently drill a large hole with less strain on the machine.

The formula used takes the number of milliseconds in a minute 60,000 divided by the RPM multiplied by .75. 60000/RPM equals the dwell time for one complete revolution. Multiplying by .75 gives 3/4 revolution to relieve the pressure.

Program Number 5 This program will drill a circular bolt hole pattern as defined by the user. The Sine and Cosine functions are used to define the X and Y dimensions for the first hole. The program then rotates the hole an amount determined by the number of holes.

Program Number 6 This sub program probes two bores in line on the X axis. It then resets the fixture offset to the center of the left hand bore and rotates the coordinate system to align with the C/L of the bores. The left hand bore will be set to X0

Y0. The distance between bores is assumed to be 5". This dimension may be adjusted in line N14. Bores up to approximately 2.5" dia. may be checked. This value will be influenced by the probe diameter and may be changed by modifying the X and Y values in the L9101 R1+1 lines.

Program Number 7 This program will cut a rectangular pocket with tapered sides. The pocket will have been roughed to the bottom finish size. The angle of the sides in this example is five degrees and the Z step will be .01. The calculation for the X and Y axis step over is Tan 5 * .01 = the step over distance. The tangent point for the bottom of the pocket must be calculated for the start dimensions. If the pocket is through, the Z depth will be equal to the depth of the bottom of the pocket plus 1/2 the diameter of the ball nose end mill that is being used. The X Y shift amount would be equal to the radius of the cutter divided by the COS of the angle. EX. .1875/COS 5 .1875/.99619=.1882 would be the correction factor for a 3/8 end mill at 5 degrees. If the pocket is not a through pocket, Z depth will be equal to the depth of the pocket minus 1/2 the diameter of the ball end mill. The X Y shift amount would use the same calculations as the through pocket plus a correction factor for the difference in depth, which would be TAN of the Angle times the Radius of the cutter.

EX: TAN 5 * .1875 = .087488*.1875 = .0164

This is the depth factor. Subtract this from the Angle factor .1882 for the total correction factor: .1882-.0164=.1718

Program Number 8 Sine Wave - A sine wave is constructed by laying out a line that represents one 360 degree revolution. In the case of a cam wrap, this would be equal to the circumference of the part to be cut. An example would be a 1 1/2" diameter that has a circumference of PI*dia=3.1416*1.5=4.7124. This line is then divided into equal segments determined by the for requirements on the blue print. Many segments give a high resolution and better form; few segments give a low resolution. Two degree segments are acceptable for most applications. Once the number of segments is determined, the line is divided by this number to obtain the Y axis step. Since our line is 360 degrees long, we find the number of steps by dividing 360 by the angular increment, in this case two. 360/2=180 steps. We then divide the length of our line (4.7124) by the number of steps (180) 4.7124/180=.0262. This will be the amount of Y axis movement for each two degree increment on the Sine wave. The X axis movement is determined by multiplying the Sine of the accumulated angle by the height of the wave. The height of the wave will be half of the total X axis movement. If we require a total movement of 1.25 inches in the X axis our wave
will be 1.25/2=.625.

Our formulas are:

PI*Dia = Circumference

360/Angular increment = Y axis move

Sine of the angle * Wave Height = X axis move

This program will ‘wrap’ a sine wave around a round part on a fourth axis.

Program Number 9 Ellipse - An ellipse is defined as a collection of points whose locations are the sum of the distances from two fixed points such that the sum of the distances is always equal. The formula is given as (X Sq/ a Sq)+(Y Sq / b Sq)=1, where a is equal to the X radius of the ellipse and b is equal to the Y radius of the ellipse. By re-arranging the formula to solve for Y we obtain +-Y = +/- Square Root of (b2 - ((X2 * b2) / a2)). Because the formula solves for plus or minus Y we will need two loops to complete the ellipse, one for the Y+ and one for the Y-. The X axis will be incremented in .01 steps.

Formula: +-Y = +/- Square Root of (b2 - ((X2 * b2) / a2))

Tutorial Program Explanations

Program Number 1 This is a simple program to check the length of tool number 1 and enter this length into the tool offset table. Fixture offset number 23 must contain the distance from the Z zero surface to the top of the Tool Probe. Fixture offset number 24 must contain the X and Y location of the tool probe.

N1 O99 (CHECK TOOL
Contains the program number and a comment
N2 G90 G0 E24 X0 Y.25 S250. M4
Sets absolute and rapid modes, moves to X0 Y.25 on the Tool Probe and turns the spindle on backward at 250. RPM
N3 M65
Turns on the TS-20 tool setting probe
N4 H1 E23 Z1.
Applies the tool length offset for tool #1 plus the Z axis correction for the Fixture location plus one inch. This moves the tool tip to one inch above the tool probe
N5 G1 G31 Z-.1 F20.
Commands the probe skip function and feeds the tool into the tool probe
N6 G91 Z.05
Commands an incremental Z axis move .05 off of the tool probe
N7 G90 S500
Sets absolute mode and increases the RPM to 500
N8 G1 G31 Z-.1 F1.

Commands probe skip function and feeds the Z axis back into the tool probe at a slower feed for improved accuracy
N9 #R9=AZ
Reads the current Z location (AZ) into register 9 (R9). This is the actual Z value from the Home position
N10 #R8=R9-FZ23
This macro statement subtracts the Z fixture location in FIXTURE OFFSET 23 (FZ23) from the value in register 9 (R9) and places it in register 8 (R8) This compensates for the difference in Z from the Z zero position to the top of the tool probe
N11 G10 L10 P1 R0+R8
Uses the G10 function to store the value in register 8 (R8) into tool offset number 1
N12 M5 G0
Turns off the spindle and sets rapid mode
N13 G0 G90 H0 Z0
Sets absolute mode, cancels the tool offset and moves to Z zero
N14 E0 X0 Y0 Z0
Cancels the fixture offset and moves to X0 Y0 and Z0
N15 M99
Code to return from sub program. Will return control to the main or calling program

Program Number 2 This program is similar to program 1 in that it uses the TS-20 or TS-27 tool setting probe to check for tool condition. The function of this program is to check for broken tools.

N1 O2(SUB TO CHECK FOR TOOL BREAKAGE
Contains the program number and a comment
N2 G90 G0 E24 X0 Y.25 S250. M4
Sets absolute and rapid modes, moves to X0 Y.25 on the touch probe and turns the spindle on backward at 250 RPM. Make sure the Y axis move will clear the tool probe
N3 #R8=TN
Sets register 8 (R8) equal to the tool number
N4 #H99=H(R8)
Sets tool offset number 99 (H99) equal to the tool offset indicated in register 8 (R8)
N5 #H99=H99+FZ23
Adds the correction factor for the difference from Z zero to the top of the probe

Note: Lines Four and Five may be combined to eliminate a step:

N4 #H99=H(R8)+FZ23
N6 M65
Turns on the tool probe
N7 H99 Z1. (TOOL LENGTH ENTERED IN OFFSET #99 TO PROBE
Applies the corrected tool length offset to one inch above the tool probe
N8 G1 Z-.1
Brings the tip of the tool below the edge of the probe
N9 G1 G31 Y0
Turns on the probe skip function and moves the tool into the tool probe
N10 #R9=AY
Reads the Y axis position into register number 9 (R9)
N11 G0 Y.25
Moves the tool off of the tool probe
N12 H0 Z0 G0 E0 X0 Y0
Cancels all offsets and sends the machine to the home position
N13 #IF R9>0 THEN GOTO :EXIT

Tests the value of R9. If the value of R9 is greater than Zero the tool is not broken. This is because once the tool makes contact with the probe, axis motion is stopped. If the tool is broken it will not make contact with the tool probe and will complete the move to Y0. If the value of R9 is greater than zero the THEN portion of the command will cause program execution to continue on the line with the :EXITlabel. If the value of R9 is zero, this line will cause no action. The program will continue with the next line.

N14 M0 Halts program execution. If the tool is broken the program will halt execution here. A #PRINT statement could be used here to alert the operator to a broken tool.
N15 #:EXIT This is the label for the jump from line 13 to continue program execution
N16 M99 Code to return from sub program. It will return control to the main or calling program

Program Number 3 This program gets the tool number for the tool in the spindle then uses that tool to drill a set of holes. It then checks the time that the drill is cutting against the time entered in the tool table. If the time is less than the total time in the table, then the routine will continue. Once the time has exceeded the time allowed, the program will increment the tool number and load the next tool. This process will continue until all tools in the tool changer have been used. In order for this program to function properly, the timers must be set to 2)DO NOT CHECK. This function is found on page two of the parameters. In addition, all tools to be used must be loaded and their respective tool lengths entered in the tool table.

*N1 03*TOOL TIME**
Contains the program number and a comment.
N2 G0 G90 E1 X0 Y0
Sets the rapid and absolute modes then moves to X0 Y0 at fixture offset number 1
N3 #V9=TN 'GET THE TOOL NUMBER
Assigns the number of the tool currently in the spindle to variable 9 (V9)
N4 #R9=V9 Assigns the tool number to register 9 (R9)
Note: Lines three and four may be combined to eliminate a step:

#R9=TN

N5 #:LOOP
Contains the label :LOOP. This is where an IF THEN loop will send the program
N6 M6 T+R9
Commands a tool change to the tool indicated by register 9 (R9)
N7 Z.1 H+R9
Applies the tool offset to .1 above Z zero
N8 G81 G98 Z-.1 R0+.1 F50. M45
Drills a hole
N9 G91 X.5 L20
Drills twenty holes in the X axis .5" apart
N10 G80 G90 M5M9
Cancels the drill cycle, turns off the spindle and coolant, resets the absolute mode
N11 X0 Y0
Moves back to X0 Y0.
N12 #IF TU(R9)<TT(R9) THEN GOTO :LOOP

Tests the time used against the time set in the Tool Time table. If the time used is less than the time in the table, program execution continues at the :LOOP label in line N5. If the tool time is expected to be the same for all of the tools TT(R9) may be changed to TT1. This will eliminate the need to set the tool time several places in the table, all times will be checked against the value in one

table. If the time used is more than the time set in the Tool Time Table, program execution will continue on the next line.

N13 #R9=R9+1
Increments the tool number by one
N14 #TU(R9)=0
Resets the Time used in the Tool Time table to Zero
N15 #IF R9<22 THEN GOTO :LOOP

Tests the value of R9 to see if all of the tools have been used. If the tool number is 21 or lower, program execution will continue at :LOOP line N5. If the tool number is greater than 21 program execution will continue on the next line. NOTE: The test value may be changed to test for more or fewer tools.

N16 M2 End program code

Program Number 4 This program takes the diameter of a large drill then calculates the amount of each peck and the minimum dwell time required to relieve pressure on the drill. This allows the machine to more efficiently drill a large hole with less strain on the machine.

The formula used takes the number of milliseconds in a minute 60,000 divided by the RPM multiplied by .75. 60000/RPM equals the dwell time for one

complete revolution. Multiplying by .75 gives 3/4 revolution to relieve the pressure.

N1 O4 (BIG DRILL
Contains the program number and a comment
N2 L100
Subroutine number one
N3 G1 G91 F5. Z-R6
Sets incremental mode feed at 5. IPM to Z minus the value in register 6 (R6)
N4 G4 P+R4
Dwell for the amount of time indicated by the value in R4
N5 #V10=AZ
Enters the current Z position into V10
N6 #IF V10>V1 THEN GOTO N3
Tests for the current location of Z. If full depth has not been reached loop back to N3
N7 #V10=0
Reset V10 to zero
N8 G90 G1 Z.1F50.
Set absolute mode. Feed out to .1 above Z zero
N9 M17
N10 M30
Lines nine and ten end the subroutine definitions
N11 #CLEAR
The program starts here
This statement clears all of the values in the variable table
N12 R9-1. R8+1000. R7+.05 (CHANGE THIS LINE FOR NEW VALUES.
Sets the R values for the drill to be used
N13 #PRINT "R9=ABSOLUTE Z DEPTH"
N14 #PRINT "R8=RPM TO BE USED"
N15 #PRINT "R7=DIAMETER OF DRILL BEING USED"
Lines 13, 14 and 15 inform the operator what the R values are used for
N16 #V1=R9
Passes the value in R9 to V1
N17 #V2=R8
Passes the value in R8 to V2
N18 #R4=(6000/V2).75*
Calculates the Dwell time and stores it in R4

N19 #V3=R7
Passes the value in R7 to V3

N20 #V4=V3.3 '.3 = 30 PERCENT OF THE DRILL DIAMETER*
Calculates the peck distance
N21 #R6=V4
Passes the value in V4 to R6
N22 G90 G0 S+R8 M3 E1 X0 Y0
Sets the absolute mode, turns on the spindle and rapids to X0 Y0 of fixture offset #1
N23 H1 Z1. M8
Brings the spindle to 1. above Z zero and turns on the coolant
N24 Z.1
Brings the tool .1 above Z zero
N25 L101 G66
Calls subroutine L100 as a modal routine. From this point the subroutine will execute after every position move until canceled by a G67
N26 M45
Will execute the subroutine at this location
N27 X1.
Moves to X1. and executes the subroutine
N28 X2.
Moves to X2. and executes the subroutine
N29 G67
Cancels execution of the subroutine
N30 G0 G90 H0 Z0
Sets absolute and rapid modes. Cancels tool length offset and moves to Z0
N31 E0 X0 Y0
Cancels the fixture offset and moves to X0 Y0
N32 M2
End of program

Program Number 5 This program will drill a circular bolt hole pattern as defined by the user. The Sine and Cosine functions are used to define the X and Y dimensions for the first hole. The program then rotates the hole an amount determined by the number of holes.

N1 O5 (BOLT PATTERN
Program number and comment
N2 #CLEAR
Clears all variable registers
N3 #PRINT "X0 Y0 IS THE CENTER OF THE PATTERN USING FIXTURE OFFSET
#1"
N4 #PRINT "R5 IS THE NUMBER OF HOLES TO BE DRILLED"
N5 #PRINT "R4 IS THE DIAMETER OF THE HOLE PATTERN"

N6 #PRINT "R3 IS THE STARTING ANGLE"
Lines three through six tell the operator where everything is
N7 R5+10. R4+5. R3+30.
Defines the parameters for the part
N8 #R7=360/R5 'ANGLE BETWEEN HOLES
Sets R7 as the angle between holes. It is to be used in the G68 line to rotate the hole
N9 #V1=R4/2 'RADIUS OF THE BOLT PATTERN
Determines the radius of the bolt pattern for X Y calculations
*N10 #R9=SIN(R3)*V1 'X STARTING LOCATION*
*N11 #R8=COS(R3)*V1 'Y STARTING LOCATION*
N12 T1M6
N13 G90 G80 G0 S5000 M3 E1 X+R9 Y+R8
N14 Z5. H1 M8
N15 G81 X+R9 Y+R8 Z-1. R0.1 F10.G98
N16 #R6=R7
Set R6 equal to R7 as a value to increment the angle for each rotation
N17 #:LOOP
Label to loop to
N18 G68 R+R7 X0 Y0
Rotate the hole by the angle defined by R7
N19 #R7=R7+R6
Add to the angle for rotation (next hole)
N20 #IF R7<(360-R6) THEN GOTO :LOOP
Test for last hole
N21 G80 M5 M9
N22 G49 Z0
N23 G28 X0 Y0 Z0
N24 M2

Program Number 6 This sub program probes two bores in line on the X axis. It then resets the fixture offset to the center of the left hand bore and rotates the coordinate system to align with the C/L of the bores. The left hand bore will be set to X0 Y0. The distance between bores is assumed to be 5". This dimension may be adjusted in line N14. Bores up to approximately 2.5" diameter may be checked. This value will be influenced by the probe diameter and may be changed by changing the X and Y values in the L9101 R1+1 lines.

N1 O6(SUB PROG TO CHECK BORE C/L AND ROTATE
Contains the program number and a comment
N2 G0 G90 G40 G49 G80 Z0
Safe start line

N3 X0 Y0 E1
Move to X0 Y0 at fixture offset 1
N4 Z0.1 H1
Moves the probe to .1 above Z zero
N5 G1 Z-0.25 F30.
Moves the probe .25 below Z zero
N6 M64
Turns on the probe
N7 L9101 R1+1. Y1.25 P1 F30.
Calls the probe routine to pick up the first point in the first bore
N8 L9101 R1+1. X-1.0825 Y-0.625 P2 F30.
Calls the probe routine to pick up the second point in the first bore
N9 L9101 R1+1 X1.0825 Y-0.625 P3 F30.
Calls the probe routine to pick up the third point in the first bore
N10 L9101 R1+2.
Calculates the diameter and center point of the first bore
N11 #V50=R1
Stores the X location of the first bore in V50
N12 #V51=R2
Stores the Y location of the first bore in V51
N13 G0 Z.1
Moves to Z.1
N14 X5. Y0
Moves to the estimated center point of the second bore
N15 G1 Z-0.25 F20.
Moves to Z-.25
N16 L9101 R1+1. X5. Y1.25 P1 F30.
Calls the probe routine to pick up the first point in the second bore
N17 L9101 R1+1. X3.9175 Y-0.625 P2 F30.
Calls the probe routine to pick up the second point in the second bore
N18 L9101 R1+1. X6.0825 Y-0.625 P3 F30.
Calls the probe routine to pick up the third point in the second bore
N19 L9101 R1+2.
Calculates the diameter and center point of the second bore
N20 #V60=R1
Stores the X location of the second bore in V60
N21 #V61=R2
Stores the Y location of the second bore in V61
N22 #V55=V60-V50
Calculates the X distance from the first bore to the second bore
N23 #V56=V61-V51
Calculates the Y distance from the first bore to the second bore

N24 #V57=ATN(V56/V55)
Calculates the angular rotation off of 0
N25 #FX1=FX1+V50
Sets fixture offset one X location to the center of the first bore
N26 #FY1=FY1+V50
Sets fixture offset one Y location to the center of the first bore
N27 #R9=V57
Sets R9 equal to the angular rotation of the second bore
N28 G49 Z0
Cancels tool length offset
N29 G0 X0 Y0 E1
Moves to X0 Y0 of fixture offset one
N30 G68 X0 Y0 R+R9
Rotates the coordinate system to align on the bores
N31 M99
Code to return to the main or calling program

Program Number 7 This program will cut a rectangular pocket with tapered sides. The pocket will have been roughed to the bottom finish size. The angle of the sides in this example is five degrees and the Z step will be .01. The calculation for the X and Y axis step over is Tan 5 * .01 = the step over distance. The tangent point for the bottom of the pocket must be calculated for the start dimensions. If the pocket is through, the Z depth will be equal to the depth of the bottom of the pocket plus 1/2 the diameter of the ball nose end mill that is being used. The X Y shift amount would be equal to the radius of the cutter divided by the COS of the angle. EX: .1875/COS 5 .1875/.99619=.1882 would be the correction factor for a 3/8 end mill at 5 degrees. If the pocket is not a through pocket Z depth will be equal to the depth of the pocket minus 1/2 the diameter of the ball end mill. The X Y shift amount would use the same calculations as the through pocket plus a correction factor for the difference in depth, which would be TAN of the Angle times the Radius of the cutter. EX: TAN 5 * .1875 = .087488*.1875 = .0164. This is the depth factor. Subtract this from the Angle factor .1882 for the total correction factor. .1882-.0164=.1718.

N1 O7 (TAPER RECTANGLE
Contains the program number and a comment
N2 G0 G90 G80 G40 G49 Z0
Safe start line
N3 S7500 M3 M8
Spindle on 7500 RPM coolant on
N4 G8 M92
Set intermediate gain turn off ramping. This increases the accuracy and smoothes the moves

N5 G0 X0 Y0 E1
Move to the center of the pocket
N6 Z0.05 H1
Move to .05 above Z zero
N7 G1 Z-1.687 F50.
Move to the bottom of the pocket. The tangent point must be calculated for the X and Y locations
N8 R9+0.8736
Set R9 equal to 1/2 of the Y dimension at the bottom of the pocket plus the tangent point on the ball end mill

N9 R8+1.3556
Set R8 equal to 1/2 of the X dimension at the bottom of the pocket plus the tangent point on the ball end mill
N10 R7+1.687
Set R7 equal to the Z depth. The pocket depth plus 1/2 of the endmill diameter
N11 #:LOOP
Label to loop to for each step
N12 G1 Y-R9
Mill from the center to Y- dimension
N13 X+R8
Mill from the center to the X+ dimension
N14 Y+R9
Mill to the Y+ dimension
N15 X-R8
Mill to the X- dimension
N16 Y-R9
Mill to the Y- dimension
N17 X0
Mill to X0
N18 #R9=R9+.00087
Increment the Y dimension by .00087. The Tangent of the angle times the .01 Z step. In this case the angle is five degrees
N19 #R8=R8+.00087
Increment the X dimension by .00087. The tangent of the angle times the .01 Z step. In this case the angle is five degrees
N20 #R7=R7-.01
Increment the Z dimension .01
N21 Z-R7.
Move the Z axis up .01
N22 #IF R7 GT 0 THEN GOTO :LOOP
Test to see if the top of the part has been reached

N23 G0 Z.5 M5 M9
Rapid to Z.5 turn off the spindle and coolant
N24 G49 Z0 E0 X0 Y0 Z0
Cancel offsets and return to the home position
N25 M2
End of program

Program Number 8 Sine Wave - A sine wave is constructed by laying out a line that represents one 360 degree revolution. In the case of a cam wrap, this would be equal to the circumference of the part to be cut. An example would be a 1 1/2" diameter that has a circumference of PI*dia=3.1416*1.5=4.7124. This line is then divided into equal segments determined by the for requirements on the blue print. Many segments give a high resolution and better form; few segments give a low resolution. Two degree segments are acceptable for most applications. Once the number of segments is determined, the line is divided by this number to obtain the Y axis step. Since our line is 360 degrees long we find the number of steps by dividing 360 by the angular increment, in this case two. 360/2=180 steps. We then divide the length of our line (4.7124) by the number of steps (180) 4.7124/180=.0262. This will be the amount of Y axis movement for each two degree increment on the Sine wave. The X axis movement is determined by multiplying the Sine of the accumulated angle by the height of the wave. The height of the wave will be half of the total X axis movement. If we require a total movement of 1.25 inches in the X axis our wave will be 1.25/2=.625.

Our formulas are: PI*Dia = Circumference, 360/Angular increment = Y axis move, Sine of the angle * Wave Height = X axis move

This program will ‘wrap’ a sine wave around a round part on a fourth axis.

N1O7(SINE WAVE DEMO
Program number and a comment
N2 L100
Sub routine label
N3 R1+0 R2+0 R3+0 R4+0 (ESTABLISH VALUES FOR R WORDS
Zero R variables
N4 #V1=1.5 ‘DIAMETER OF THE PART
Enter the diameter of the part
N5 #V2=.625 ‘RISE OF THE SINE WAVE
Enter the rise of the sine wave
*N6 #V3=3.141593*V1 ‘CIRCUMFERENCE OF THE PART*
Calculate the circumference of the part

*N7 #V4=90/(5*V3) 'CALCULATE THE Q WORD*
Calculate the Q word for Cam Wrapping
N8 #R1=V4 'TRANSFER Q WORD TO R1
Transfer the value of the Q word to R1
N9 G51.1 Y0 'SET Y AXIS MIRROR
Mirror the Y axis for Cam Wrapping
N10 G17 Q+R1 'TURN ON CAM WRAPPING
Turn on Cam Wrapping
N11 G90 F50.
Set absolute mode 50. IPM
N12 #V10=V3/180 'Y AXIS INCREMENT PER 2 DEG MOVE
Calculate the Y axis increment for each 2 deg
N13 #:LOOP
Label to loop to until finished
N14 #V12=V12+V10 'Y AXIS POSITION
Calculate the Y axis position
N15 #V13=V13+2 'ANGULAR COUNT
Increment the angle
*N16 #V14=SIN(V13)*V2 'X AXIS POSITION*
Calculate the X axis position
N17 #R4=V12 'Y AXIS PASS TO R4
Transfer the Y axis position to R4
N18 #R5=V14 'X AXIS PASS TO R5
Transfer the X axis position to R5
N19 G1 X+R5 Y+R4 F20. 'NEXT MOVE
Move to the next location
N20 #IF V13 LT 360 THEN GOTO :LOOP
Test if finished
N21 G0 Z1.
Move to clear part
N22 Y0
Unwrap move
N23 G50.1
Turn off Mirror
N24 G17
Turn off Cam Wrapping
N25 M17
N26 M30
End of subroutines
N27 G0 G80 G90 G40 G49 Z0
Safe start line
N28 T1 M6

Tool change
N29 S2500 M3 M8
Spindle and coolant on
N30 G0 X0 Y0 A0 Z0 E1
Move to position
N31 Z1. H1
Move to Z 1
N32 G1 Z0.65 F15.
Move to cut height. Z zero is the center of the part
N33 L101
Call Subroutine
N34 M5 M9
Turn off spindle and coolant
N35 G49 E0 X0 Y0 Z0
Cancel offsets
N36 G28
Go home
N37 M2
End of program

Program Number 9 Ellipse - An ellipse is defined as a collection of points whose locations are the sum of the distances from two fixed points such that the sum of the distances is always equal. The formula is given as (X Sq/ a Sq)+(Y Sq / b Sq)=1 Where a is equal to the X radius of the ellipse and b is equal to the Y radius of the ellipse. By re-arranging the formula to solve for Y we obtain +- Y = Square Root of (b2 - ((X2 * b2) / a2)). Because the formula solves for plus or minus Y, we will need two loops to complete the ellipse, one for the Y+ and one for the Y-. The X axis will be incremented in .01 steps.

Formula: +-Y = +/- Square Root of (b2 - ((X2 * b2) / a2))

N1 O9 (ELLIPSE PROGRAM
Program number and a comment
N2 G0 G90 G80 G40 G49 Z0
Safe start line
N3 T1 M6
Call tool one
N4 S2500 M3 M7
Spindle on coolant on
N5 G0 X0 Y0 Z0 E1
Rapid to the middle of the part
N6 #:XDIM

Label to loop to if invalid data entered
N7 #PRINT “ENTER THE X DIMENSION”
Print statement asking the operator to enter data
N8 #INPUT V1
Input statement to accept the requested data
N9 #V1=V1/2
Calculates the X radius
N10 #IF V1 LE 0 THEN GOTO :XDIM
Tests for acceptable data
N11 #:YDIM
Label to loop to if invalid data is entered
N12 #PRINT “ENTER THE Y DIMENSION”
Print statement asking the operator for more information
N13 #INPUT V10
Input statement to accept requested information
N14 #V10=V10/2
Calculates the Y radius
N15 #IF V10 LE 0 THEN GOTO :YDIM
Tests for invalid information
N16 #V20=V1
Copies the value in V1 to V20
*N17 #V1=V1*V1*
Squares the contents of V1. This is the a value
*N18 #V10=V10*V10*
Squares the contents of V10. This is the b value

N19 #R9=V20
Transfers the X+ location to R9
N20 G0 X+R9 Y0
Moves to the start point on the Ellipse
N21 Z0.1 H1
Moves the tool to .1 above Z zero
N22 G1 Z-0.25 F20.
Feeds the tool to Z-.25
N23 #V25=V20
Copies the value of V20 into V25 for use as a counter
N24 #:LOOP
Label to loop to for each step
N25 #V20=V20-.01
Calculates the next X location
*N26 #V21=V20*V20*

Squares the X location for the X squared portion of the formula
*N27 #V30=(V21*V10)/V1*
Calculates the X squared times b squared divided by a squared portion of the formula
N28 #V31=SQR(V10-V30)
Calculates the square root of b squared minus V30 value
N29 #R9=V20
Copies the X location to R9
N30 #R8=V31
Copies the Y location to R8
N31 G1 X+R9 Y+R8 F20.
Moves to the next location
N32 #IF V20 > -V25 THEN GOTO :LOOP
Tests for the finish of the top portion of the Ellipse
N33 #:LOOP1
Label to loop to for the bottom portion of the Ellipse
N34 #V20=V20+.01
Reverses the X axis move
*N35 #V21=V20*V20*
Squares the X location for the X squared portion of the formula
*N36 #V30=(V21*V10)/V1*
First calculation of the formula
N37 #V31=SQR(V10-V30)
Final calculation of the formula
N38 #R9=V20
Copies the X location to R9
N39 #R8=V31
Copies the Y location to R8
N40 G1 X+R9 Y-R8 F20.
Moves to the next location
N41 #IF V20 < V25 THEN GOTO :LOOP1
Tests to see if the Ellipse is complete
N42 G0 Z0.1 M5 M9
Moves to Z.1 turns off the spindle and coolant
N43 G49 Z0 E0 X0 Y0
Cancels offsets returns to Zero
N44 G28
Returns home
N45 M2
End of program

Tutorial Program Listings

Program Number 1

```
N1 O99 (CHECK TOOL
N2 G90 G0 E24 X0 Y.25 S250. M4
N3 M65
N4 H1 E23 Z1.
N5 G1 G31 Z-.1 F20.
N6 G91 Z.05
N7 G90 S500
N8 G1 G31 Z-.1 F1.
N9 #R9=AZ
N10 #R8=R9-FZ23
N11 G10 L10 P1 R0+R8
N12 M5 G0
N13 G0 G90 H0 Z0
N14 E0 X0 Y0 Z0
N15 M99
```

Program Number 2

```
N1 O2(SUB TO CHECK FOR TOOL BREAKAGE
N2 G90 G0 E24 X0 Y.25 S250. M4
N3 #R8=TN
N4 #H99=H(R8)
N5 #H99=H99+F23
N6 M65
N7 H99 Z1. (TOOL LENGTH ENTERED IN OFFSET #99 TO PROBE
N8 G1 Z-.1
N9 G1 G31 Y0
N10 #R9=AY
N11 G0 Y.25
N12 H0 Z0 G0 E0 X0 Y0
N13 #IF R9>0 THEN GOTO :EXIT
N14 M0
N15 #:EXIT
N16 M99
```

Program Number 3

```
N1 03*TOOL TIME*
N2 G0 G90 E1 X0 Y0
N3 #V9=TN ‘GET THE TOOL NUMBER
N4 #R9=V9
N5 #:LOOP
N6 M6 T+R9
N7 Z.1 H+R9
N8 G81 G98 Z-.1 R0+.1 F50. M45
N9 G91 X.5 L20
N10 G80 G90 M5M9
N11 X0 Y0
N12 #IF TU(R9)<TT(R9) THEN GOTO :LOOP
N13 #R9=R9+1
N14 #TU(R9)=0
N15 #IF R9<22 THEN GOTO :LOOP
N16 M2
```

Program Number 4

```
N1 O4 (BIG DRILL
N2 L100
N3 G1 G91 F5. Z-R6
N4 G4 P+R4
N5 #V10=AZ
N6 #IF V10>V1 THEN GOTO N3
N7 #V10=0
N8 G90 G1 Z.1F50.
N9 M17
N10 M30
N11 #CLEAR
N12 R9-1. R8+1000. R7+.05 (CHANGE THIS LINE FOR NEW VALUES.
N13 #PRINT “R9=ABSOLUTE Z DEPTH”
N14 #PRINT “R8=RPM TO BE USED”
N15 #PRINT “R7=DIAMETER OF DRILL BEING USED”
N16 #V1=R9
N17 #V2=R8
N18 #R4=(6000/V2)*.75
N19 #V3=R7
N20 #V4=V3*.3 ‘.3 = 30 PERCENT OF THE DRILL DIAMETER
N21 #R6=V4
N22 G90 G0 S+R8 M3 E1 X0 Y0
N23 H1 Z1. M8
N24 Z.1
N25 L101 G66
```

N26 M45
N27 X1.
N28 X2.
N29 G67
N30 G0 G90 H0 Z0
N31 E0 X0 Y0
N32 M2

Program Number 5

N1 O5 (BOLT PATTERN
N2 #CLEAR
N3 #PRINT "X0 Y0 IS CENTER OF PATTERN USING FIXTURE OFFSET #1"
N4 #PRINT "R5 IS THE NUMBER OF HOLES TO BE DRILLED"
N5 #PRINT "R4 IS THE DIAMETER OF THE HOLE PATTERN"
N6 #PRINT "R3 IS THE STARTING ANGLE"
N7 R5+10. R4+5. R3+30.
N8 #R7=360/R5 'ANGLE BETWEEN HOLES
N9 #V1=R4/2 'RADIUS OF THE BOLT PATTERN
*N10 #R9=SIN(R3)*V1 'X STARTING LOCATION*
*N11 #R8=COS(R3)*V1 'Y STARTING LOCATION*
N12 T1M6
N13 G90 G80 G0 S5000 M3 E1 X+R9 Y+R8
N14 Z5. H1 M8
N15 G81 X+R9 Y+R8 Z-1. R0.1 F10.G98
N16 #R6=R7
N17 #:LOOP
N18 G68 R+R7 X0 Y0
N19 #R7=R7+R6
N20 #IF R7<(360-R6) THEN GOTO :LOOP
N21 G80 M5 M9
N22 G49 Z0
N23 G28 X0 Y0 Z0
N24 M2

Program Number 6

N1 O6(SUB PROG TO CHECK BORE C/L AND ROTATE
N2 G0 G90 G40 G49 G80 Z0
N3 X0 Y0 E1
N4 Z0.1 H1
N5 G1 Z-0.25 F30.
N6 M64
N7 L9101 R1+1. Y1.25 P1 F30.
N8 L9101 R1+1. X-1.0825 Y-0.625 P2 F30.
N9 L9101 R1+1 X1.0825 Y-0.625 P3 F30.

N10 L9101 R1+2.
N11 #V50=R1
N12 #V51=R2
N13 G0 Z.1
N14 X5. Y0
N15 G1 Z-0.25 F20.
N16 L9101 R1+1. X5. Y1.25 P1 F30.
N17 L9101 R1+1. X3.9175 Y-0.625 P2 F30.
N18 L9101 R1+1. X6.0825 Y-0.625 P3 F30.
N19 L9101 R1+2.
N20 #V60=R1
N21 #V61=R2
N22 #V55=V60-V50
N23 #V56=V61-V51
N24 #V57=ATN(V56/V55)
N25 #FX1=FX1+V50
N26 #FY1=FY1+V50
N27 #R9=V57
N28 G49 Z0
N29 G0 X0 Y0 E1
N30 G68 X0 Y0 R+R9
N31 M99

Program Number 7

N1 O7 (TAPER RECTANGLE
N2 G0 G90 G80 G40 G49 Z0
N3 S7500 M3 M8
N4 G8 M92
N5 G0 X0 Y0 E1
N6 Z0.05 H1
N7 G1 Z-1.687 F50.
N8 R9+0.8736
N9 R8+1.3556
N10 R7+1.687
N11 #:LOOP
N12 G1 Y-R9
N13 X+R8
N14 Y+R9
N15 X-R8
N16 Y-R9
N17 X0
N18 #R9=R9+.00087
N19 #R8=R8+.00087

N20 #R7=R7-.01
N21 Z-R7.
N22 #IF R7 GT 0 THEN GOTO :LOOP
N23 G0 Z.5 M5 M9
N24 G49 Z0 E0 X0 Y0 Z0
N25 M2

Program Number 8

N1O7(SINE WAVE DEMO
N2 L100
N3 R1+0 R2+0 R3+0 R4+0 (ESTABLISH VALUES FOR R WORDS
N4 #V1=1.5 'DIAMETER OF THE PART
N5 #V2=.625 'RISE OF THE SINE WAVE
*N6 #V3=3.141593*V1 'CIRCUMFERENCE OF THE PART*
*N7 #V4=90/(5*V3) 'CALCULATE THE Q WORD*
N8 #R1=V4 'TRANSFER Q WORD TO R1
N9 G51.1 Y0 'SET Y AXIS MIRROR
N10 G17 Q+R1 'TURN ON CAM WRAPPING
N11 G90 F50.
N12 #V10=V3/180 'Y AXIS INCREMENT PER 2 DEG MOVE
N13 #:LOOP
N14 #V12=V12+V10 'Y AXIS POSITION
N15 #V13=V13+2 'ANGULAR COUNT
*N16 #V14=SIN(V13)*V2 'X AXIS POSITION*
N17 #R4=V12 'Y AXIS PASS TO R4
N18 #R5=V14 'X AXIS PASS TO R5
N19 G1 X+R5 Y+R4 F20. 'NEXT MOVE
N20 #IF V13 LT 360 THEN GOTO :LOOP
N21 G0 Z1.
N22 Y0
N23 G50.1
N24 G17
N25 M17
N26 M30
N27 G0 G80 G90 G40 G49 Z0
N28 T1 M6
N29 S2500 M3 M8
N30 G0 X0 Y0 A0 Z0 E1
N31 Z1. H1
N32 G1 Z0.65 F15.
N33 L101
N34 M5 M9
N35 G49 E0 X0 Y0 Z0

N36 G28
N37 M2

Program Number 9

```
N1 O9 (ELLIPSE PROGRAM
N2 G0 G90 G80 G40 G49 Z0
N3 T1 M6
N4 S2500 M3 M7
N5 G0 X0 Y0 Z0 E1
N6 #:XDIM
N7 #PRINT “ENTER THE X DIMENSION”
N8 #INPUT V1
N9 #V1=V1/2
N10 #IF V1 LE 0 THEN GOTO :XDIM
N11 #:YDIM
N12 #PRINT “ENTER THE Y DIMENSION”
N13 #INPUT V10
N14 #V10=V10/2
N15 #IF V10 LE 0 THEN GOTO :YDIM
N16 #V20=V1
N17 #V1=V1*V1
N18 #V10=V10*V10
N19 #R9=V20
N20 G0 X+R9 Y0
N21 Z0.1 H1
N22 G1 Z-0.25 F20.
N23 #V25=V20
N24 #:LOOP
N25 #V20=V20-.01
N26 #V21=V20*V20
N27 #V30=(V21*V10)/V1
N28 #V31=SQR(V10-V30)
N29 #R9=V20
N30 #R8=V31
N31 G1 X+R9 Y+R8 F20.
N32 #IF V20 > -V25 THEN GOTO :LOOP
N33 #:LOOP1
N34 #V20=V20+.01
N35 #V21=V20*V20
N36 #V30=(V21*V10)/V1
N37 #V31=SQR(V10-V30)
N38 #R9=V20
```

N39 #R8=V31
N40 G1 X+R9 Y-R8 F20.
N41 #IF V20 < V25 THEN GOTO :LOOP1
N42 G0 Z0.1 M5 M9
N43 G49 Z0 E0 X0 Y0
N44 G28
N45 M

2