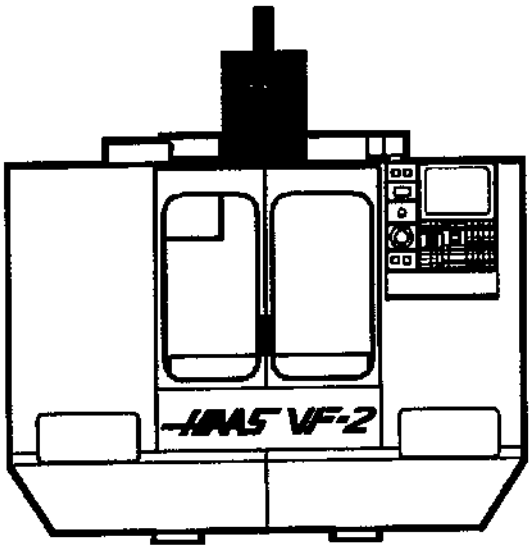


VF SERIES
VERTICAL MACHINING CENTER

PROGRAMMING and OPERATION MANUAL

Haas Automation, Inc.

9601 Lurline Ave.
Chatsworth, CA 91311
(818) 885-6050

November 19, 1993

CONTENTS

WARRANTY REGISTRATION CERTIFICATE

The Haas Automation Models VF-0, VF-1, VF-2, VF-3, and VF-4 CNC mills are warranted by Haas Automation as follows:

CONTROL: (All components within the main control cabinet and pendant cabinet):
Warranted against defects in material and workmanship for a period of two years from date of purchase.

NON-CONTROL:
The remainder of the machine is warranted against defects in material and workmanship for a period of one year from date of purchase.

This warranty is void if the unit is subject to misuse, neglect, accident, improper installation, or improper application. Haas Automation is not liable for any additional or incidental damage to parts, fixtures, machines, or loss of time that may be caused by malfunctions.

Warranty service is available from your dealer. Units out of warranty will be repaired promptly and at reasonable cost.

"Date of purchase" is the date the machine is shipped to the original purchaser.

If you have a problem with your unit, a re-reading of the manuals might solve the problem. If you still have a problem, a phone call to your dealer should solve the problem. **Please call your dealer first.** If you cannot get the problem taken care of with either of the above, a phone call to Haas maybe able to solve the problem.

Haas Automation
9601 Lurline Ave.
Chatsworth, CA 91311
Phone: (818) 885-6050 FAX: (818) 885-8372

In order to record the end user of this machine for updates and for product safety notices, we must have the machine registration returned immediately. Please fill out completely and mail to the above address to ATTENTION (VF-0, VF-1, etc. — whichever is applicable) REGISTRATIONS. **Please include a copy of your invoice** to validate your warranty date and to cover any additional options you may have purchased.

Company Name: _____ Contact Name: _____
Address: _____ Dealer: _____
_____ Date Purchased: _____
Model No.: _____ Serial Number: _____
Telephone: ( ) _____ FAX: _____

IMPORTANT NOTICE!!!! PLEASE READ IMMEDIATELY!!!!

This machine is equipped with an electronically-recorded serial number that cannot be altered. This is done to protect you in case of theft and to track machines when sold to other owners. After approximately 800 hours of use, the machine will automatically shut down if it has not been unlocked by Haas Automation. To unlock the machine, we must have the above registration with the serial number and the authorization from your dealer. You will receive a number from Haas that you will write in over the serial number on setting page (#26). The authorization from the dealer will come upon final acceptance of the machine. If, for any reason, the serial number of the machine is erased in memory, the machine will revert back to 200 hour limit for your protection.

We are sorry for any inconvenience, but because of the threat of lawsuits, it is important that we know the locations of machines to inform you of any potential safety problems.

I. GENERAL INFORMATION

The operation of the VF-Series Vertical Machining Center requires that a part program be designed, written, and entered into the memory of the controller. The most common way of writing part programs is off-line, that is, away from the CNC in a facility that can save the program and subsequently send it to the CNC control.

The most common way of sending a part program to the CNC is via an RS-232 interface. The HAAS VF-Series Vertical Machining Center has an RS-232 interface that is compatible with most existing computers and CNC's.

The HAAS VF-Series Vertical Machining Center control can store and save more than one part program even when the machine's power is turned off. Programming of the part program can also be done at the CNC by using the EDIT function. Editing can even be done while another program is being run.

This manual can be used as both an operator's manual and as a programmer's manual. The following two sections are to be used as a **basic** introduction to programming. However, a CNC programmer needs much more training and information available to him or her before attempting to program on the machine.

The remainder of this manual is divided into a ***Programming*** chapter (**II**) and an ***Operation*** chapter (**III**).

1. Basic Programming

An "NC" machine is a machine that is numerically controlled, i.e., the machine tool is controlled by a code system that enables it to be operated with little or no personal supervision and with a great deal of repeatability. Thus, the same task may be performed over and over with minimum human error. "CNC" is the same type of operating system, with the exception that the machine tool is monitored by a computer. The term CNC stands for **C**omputerized **N**umerical **C**ontrol.

Anyone who can operate a manual machine can learn to program an NC or CNC machine. The main difference is that instead of cranking handles to position a slide to a certain point, that dimension would be stored in the memory of the machine control **once**. The control thereafter will move the machine automatically when called out in the program, on each individual part.

Programming for parts that are complex is done by building several steps, or operations, together to form the finished product.

If a person sets out to operate and program an CNC controlled machine, that person must have a basic understanding of machining practices and a working knowledge of math. He/she needs also to become familiar with the control console and the placement of the keys, switches, displays, etc., that are pertinent to the operation of the machine.

This section is to be used as a supplementary teaching aid to users of the HAAS Vertical Machining Center. The information in this section may apply in whole or in part to the operation of other CNC machines. Its use is intended only as an aid in the operation HAAS Vertical Machining Center.

The purpose of this section is to give a basic understanding of CNC programming and its applications. It is not intended as an in-depth study of all ranges of machine use, but as an overview of common and potential situations facing CNC programmers.

1.1 The Coordinate System

The first diagram we are concerned with is called a NUMBER LINE. This number line has a reference point zero that is called ABSOLUTE ZERO and may be placed at any point along the line.

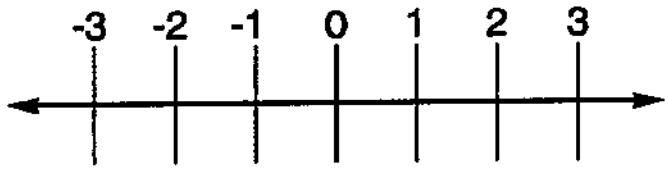


Fig. 1-1 *Horizontal number line.*

The number line also has numbered increments on either side of absolute zero. Moving away from zero to the right are positive increments. Moving away from zero to the left are negative increments. The "+", or positive increments, are understood, therefore no sign is needed.

We use positive and negative along with the increment's value to indicate its relationship to zero on the line. In the case of the previous line, if we choose to move to the third increment on the minus (-) side of zero, we would call for -3. If we choose the second increment in the plus range, we would call for 2. Our concern is with distance and direction from zero.

Remember that zero may be placed at any point along the line, and that once placed, one side of zero has negative increments and the other side has positive increments.

The next illustration (Fig. 1-3) pictures the three directions of travel on a vertical machining center. To carry the number line idea a little further, imagine such a line placed along each axis of the machine.

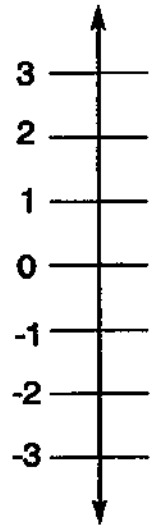


Fig. 1-2

The first number line is easy to conceive as belonging to the left-to-right, or "X", axis of the machine. If we place a similar number line along the front-to-back, or "Y", axis, the increments toward the operator are the negative increments, and the increments on the other side of zero away from the operator are the positive increments.

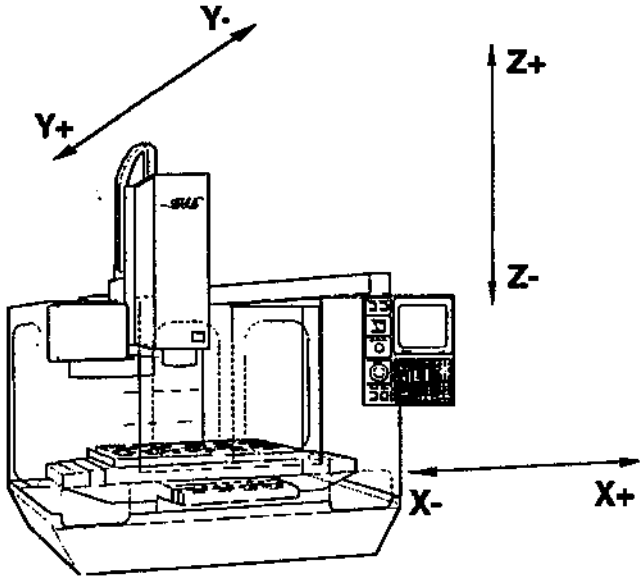


Fig. 1-3 *VF-1 showing X, Y, and Z axis lines.*

The final axis of travel on our machine is the up-and-down, or "Z", axis. When we place a number line on the **Z** travel, the positive increments are up — above zero — and the negative values are down — below zero. Actually, the increments on each number line on the HAAS machining centers equals .0001 inches. Also, while a line theoretically travels infinitely in either direction once established, the three lines placed along the **X**, **Y**, and **Z** axes of the machine do not have unlimited accessibility. That is to say, we are limited by the range of travel on the machine. For the HAAS VF-1 and VF-0, we have access to 20 inches in the X-axis, 16 inches in the Y-axis, and 20 inches in the Z-axis. For the VF-2, we have access to 30 inches in the X-axis, 16 inches in the Y-axis, and 20 inches in the Z-axis. For the VF-3, we have access to 40 inches in the X-axis, 20 inches in the Y-axis, and 25 inches in the Z-axis.

Remember, when we are moving the machine, we are concerned with positioning the spindle. Although the

machine table is the moving part, we have to keep in mind our coordinates are based off our theoretical spindle movement.

Keep in mind that the zero position may be placed at any point along each of the three number lines, and in fact will probably be different for each setup of the machine. It is noteworthy to mention here that the Z-axis is usually set with the machine zero position in the full upward position, or the tool change position. This will place all the **Z** moves in a negative range of travel. However, the work zero in the Z-axis is usually set at the top of the part surface, this will be entered in the tool length offset as a negative value. The range of travel on the HAAS VF Series is a total of 20 inches, four of these inches are above tool change position and are listed as a positive tool length offset, and 16 of these inches are below tool change position and are listed as a negative tool length offset.

The diagram at right shows a top view of the grid as it would appear on the machine tool. This view shows the **X** and **Y** axes as the operator faces the machine tool. Note that at the intersection of the two lines, a common zero point is established. The four areas to the sides and above and below the lines are called "QUADRANTS" and make up the basis for what is known as rectangular coordinate programming.

THE TOP LEFT QUADRANT IS = X—, Y+
THE BOTTOM LEFT QUADRANT IS = X—, Y—
THE TOP RIGHT QUADRANT IS = X+, Y+
THE BOTTOM RIGHT QUADRANT IS= X+, Y--

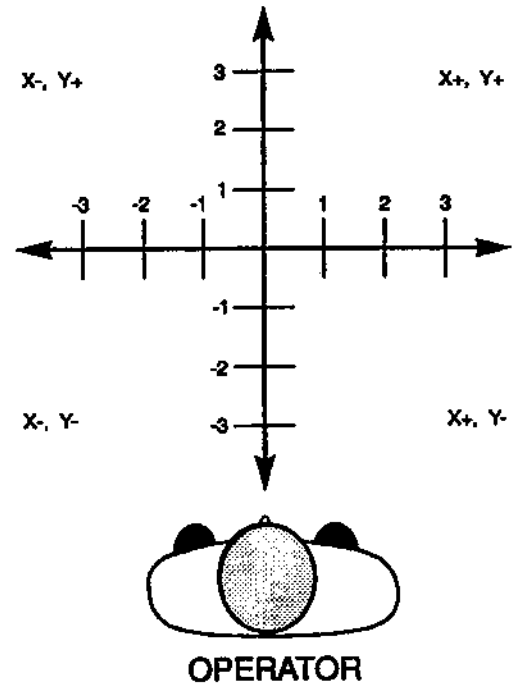


Fig. 1-4 *X,Y grid as it would appear from top view.*

Whenever we set a zero somewhere on the X-axis and somewhere on the Y-axis, we have automatically caused an intersection of the two lines. This intersection where the two zeros come together will automatically have the four quadrants to its sides, above, and below it. How much of a quadrant we will be able to access is determined by where we placed the zero within the travel of the machine axis.

For example, if we set zero exactly in the middle of the travel of **X** and **Y** (table center), we have created four quadrants that are 10 inches by 8 inches in size.

The layout of the **A** and **B** axes on the HAAS five-axis control are depicted at right and on the next page. The **A** axis is rotary motion about the X axis, while the **B** axis determines rotary motion about the **Y** axis. The right hand rule can be used to determine axis rotation for the **A** and **B** axes. When placing the thumb of the right hand along the positive X axis, the fingers of the right hand will point in the direction of tool movement for a positive **A** axis command. Likewise, when placing the thumb of the right hand along the positive **Y** axis, the fingers of the right hand will point in the direction of tool movement for a positive **B** axis command. It is important to remember that the right hand rule determines direction of tool movement and not the table

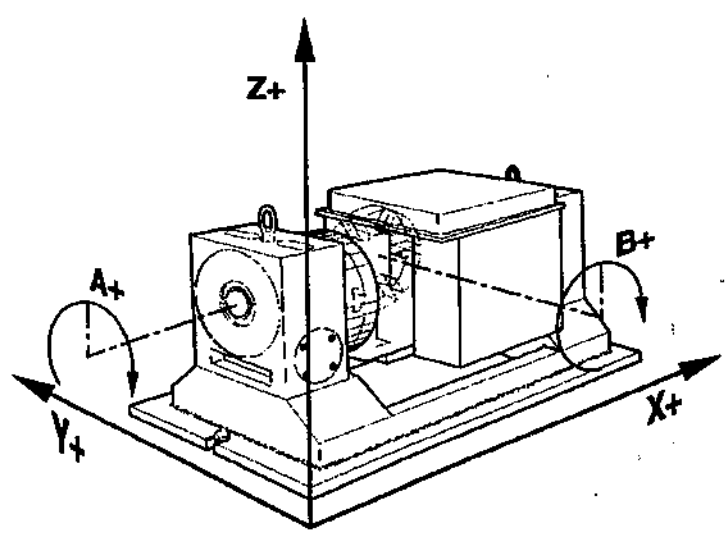


Fig. 1-5 *Positive tool movement for fifth axis machines.*

movement direction. For the right hand rule, the fingers will point opposite of the positive rotary table movement. Refer to Figures 1-5 and 1-6.

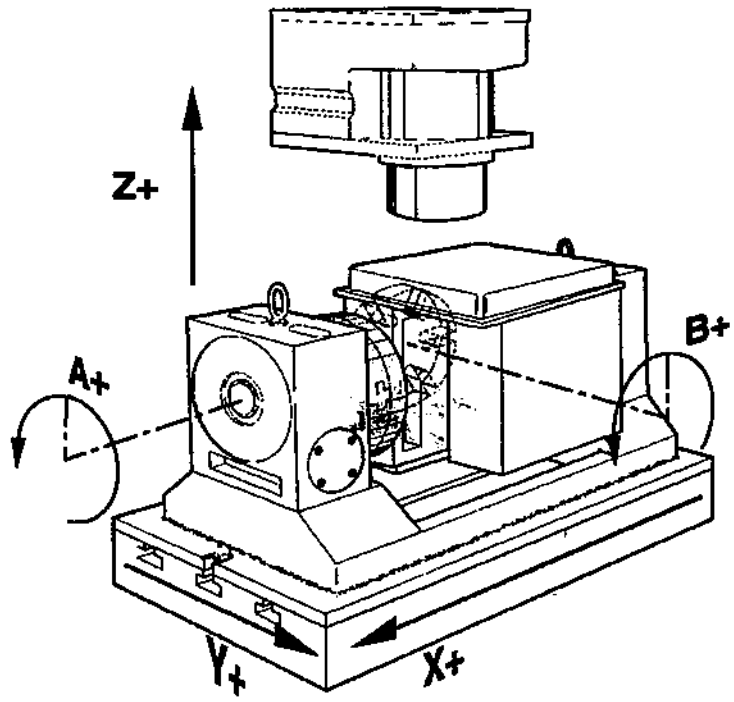


Fig. 1-6 *Positive mounting surface movement for all axes.*

NOTE: Figures 1-5 and 1-6 represent one of many possible machine tool and table configurations. You may have different table movements for positive directions, depending on the equipment, parameter settings, or five-axis software being used.

1.2 Machine Home

The principal may be seen when doing a manual reference return of all machine axes. When a zero return (ZERO RET) is performed at machine start up, all three axes are brought to the extreme positive direction until the limit switch is reached. When this condition is satisfied, the only way to move any of the three axes is in the negative direction. This is because a new zero was set for each of the three axes automatically when the machine was brought Home. This is placed at the edge of each axes travel. In effect, now the positive quadrants cannot be reached, and all the **X** and **Y** moves will be found to be in the X- , Y- quadrant. It is only by setting a new part zero somewhere within the travel of each axis that other quadrants are able to be reached.

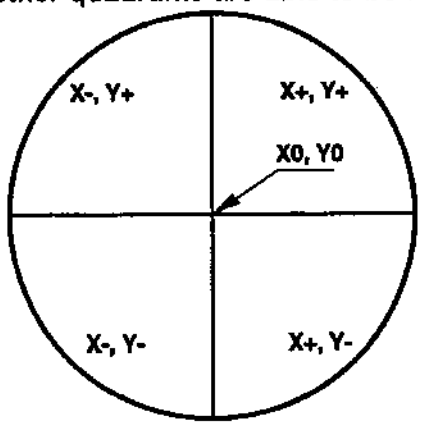


Fig. 1-7 *All four quadrants will have to be accessed to machine this part.*

Sometimes it is useful in the machining of a part to utilize more than one of the X,Y quadrants. A good example of this is a round part that has it's datum lines running through the center. The setup of such a part may look like this:

These are just some examples of how to make use of the four quadrants of the **X** and **Y** axes on the machine. As more experience is gained in the machine tool programming and setup techniques, each programmer and setup person develops their own methods and style. Some methods will be faster than others, but each individual will have to determine the needs of each job in question, and reflect back on notes and the previous jobs completed.

1.3 Absolute and Incremental Positioning

Up to this point, we have dealt with a system of positioning the tool that is known as absolute programming. In absolute or A.B.S., all coordinate points are given with regard to their relationship to the origin, a fixed zero point, or considered as part zero. This is the most common type of positioning.

Another type of positioning is called incremental positioning. Incremental, or I.N.C. positioning concerns itself with distance and direction. A new coordinate is entered in terms of its relationship to the previous position, and not from a fixed zero or origin. In other words, after a block of information has been executed, the position that the tool is now at is the new zero point for the next move to be made.

An example of the use of the incremental system is below. Note that to move from X 4.25 to X 2.025 on the scale, an incremental move of X -2.225 was made, even though the move still places the tool on the plus side of the scale. Therefore the move was determined from the last point, with no regard for the zero position. The + and - signs are used in terms of direction, and not in regard to the position of zero.

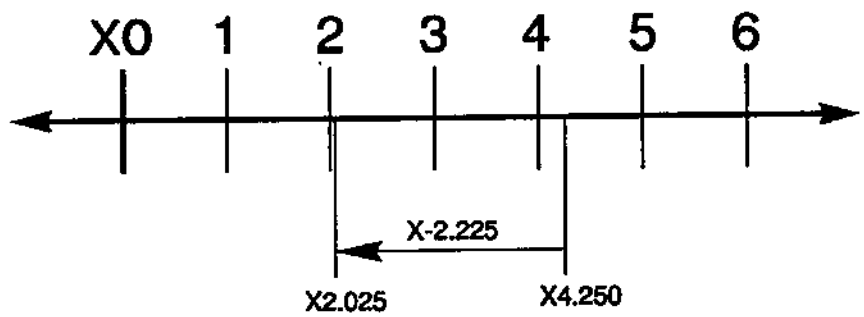


Fig. 1-8 *An example of an incremental move.*

Keep in mind that when positioning in **absolute**, we are concerned with distance and direction from a fixed zero reference point, and when positioning in **incremental** we are concerned with distance and direction from the last position.

1.4 Programming With Codes

A program is written as a set of instructions given in the order they are to be performed. The instructions, if given in English, might look like this:

LINE #1 = SELECT CUTTING TOOL.
LINE #2 = TURN THE SPINDLE ON AND SELECT THE RPM.
LINE #3 = TURN THE COOLANT ON.
LINE #4 = RAPID TO THE STARTING POSITION OF THE PART.
LINE #5 = CHOOSE THE PROPER FEED RATE AND MAKE THE CUT(S).
LINE #6 = TURN OFF THE SPINDLE AND THE COOLANT.
LINE #7 = RETURN TOOL TO HOLDING POSITION AND SELECT NEXT TOOL.

and so on. But our machine control understands only these messages when given in machine code.

Before considering the meaning and the use of codes, it is helpful to lay down a few guidelines:

1) Codes come in groups. Each group of codes will have a specific group number.

2) A **G** code from the same group can be replaced by another code in the same group. By doing this the programmer establishes modes of operation. The universal rule here is that codes from the same group cannot be used more than once on the same line.

3) There are modal **G** codes which, once established, remain effective until replaced with another code from the same group.

4) There are non-modal **G** codes which, once called, are effective only in the calling block, and are immediately forgotten by the control.

The rules above govern the use of all codes for programming the Haas (and other) controls. The concept of grouping codes and rules that apply will have to be remembered if we are to effectively program the machine tool. The following is a discussion of the codes most basic to the operation of the machine.

G CODES:

G00 Rapid traverse motion; Used for positioning and during non-cutting moves.
NOTE: Machine rapids at 500 inches per minute (IPM).

G01 Linear interpolation motion; Used for actual machining and metal removal. Governed by programmed feed rate in inches per minute.

G02 Circular interpolation - Clockwise.

G03 Circular interpolation - Counterclockwise.

G28 Machine home (Rapid traverse).

G40 Cutter compensation cancel.

G41 Cutter compensation to **left** of path.

G42 Cutter compensation to **right** of path.

G43 Read tool length compensation.

G54 Work coordinate #1 (Part zero).

G80 Canned cycle cancel.

G81 Drill canned cycle.

G82 Spot drill canned cycle.

G83 Peck drill canned cycle.

G84 Tapping canned cycle.

G90 Absolute programming.

G91 Incremental programming.

G98 Initial point return.

G99 Reference plane return.

M CODES:

M00 Program stop. Press CYCLE START button to continue.

M01 Optional stop program. Press optional stop key on control panel on M01 code.

M02 End of program. Cannot continue.

M03 Start spindle forward (Clockwise). Must be accompanied by a spindle speed.

M04 Start spindle reverse (Counterclockwise). Must have a spindle speed.

M05 Spindle stop.

M06 Tool change command. Must have a tool number in the same line. This command will automatically stop the spindle.

M08 Coolant **ON** command.

M09 Coolant **OFF** command.

M30 Program end and rewind to beginning of program.

M97 Local subroutine call.

M98 Subprogram call.

M99 Subprogram return, or loop.

NOTE: Only one "M" code can be used per line.
The "M" code will be the last item of code to be performed, regardless of where it is located in the line.

1.5 Machine Defaults

A **default** is an automatic function of the machine tool control. When powering up the machine, the control looks for the home position of all axes, then will read the default values or the preset "G" codes. If you have ever wondered why the machine went to the part zero that was entered in the G54 column when it was never specified in the actual program, it is because the machine automatically reads the G54 column upon start-up. That is a **default**.

The defaults for the Haas VF Series are listed in the Operator/Programming Manual, and are indicated by an asterisk (*) next to the specific G codes.

The control automatically reads these G codes when power is turned on.

G00 Rapid traverse
G17 X,Y Circular plane selection
G40 Cutter Compensation cancel
G49 Tool length compensation cancel
G54 Work coordinate zero #1 (1 of 26 available)
G64 Exact stop cancel
G80 Canned cycle cancel

G90 Absolute programming
G98 Initial point return

There is no default FEED RATE (**F** code), but once an **F** code is programmed, it will apply until another is entered.

1.6 Program Format

Program format, or program style is an important part of CNC machining. Each individual will format their programs differently and, in most cases, a programmer could not identify a program written by themselves. The point is that a programmer needs to be consistent and efficient, writing code in the way it is listed and in the order it appears in the program. For example:

X, **Y**, **Z** is in order of appearance. The machine will read **X**, **Y**, or **Z** in any order, but we want to be consistent. Write **X** first, **Y** second, **Z** third.

The first line or block in a program using active G codes should be a tool number and tool change command. This would be a good safety measure.

The second line or block will contain a rapid command (G00), an absolute or incremental command (G90, G91), a work zero for **X** and **Y** (G54), a positioning **X** and **Y** coordinate, a spindle speed command (S____), and a spindle ON clockwise command (M03).

The third line or block will contain a "Read tool length compensation" command (G43), a tool length offset number (H01), a Z-axis positioning move (Z.1), and an optional coolant ON command (M08).

An example program's first three lines will look like this:

```
T1  M06;
G00  G90  G54  X0  Y0  S2500  M03;
G43  H01  Z.1  M08;
```

All the necessary codes for each operation are listed above. This format is a good practice and will separate your style from other programmers.

QUESTION:
If G00, G90, and G54 are defaults, why do we list them in the second line of a program and for each different tool?

ANSWER:
G00, G90, and G54 are listed for an operator/setup person's aid so he/she can determine if the machine will rapid position, if the machine is in fact in the absolute coordinate mode, and most important, the work zero. The work zero is always different between setups, and multiple work zeros are very common.

QUESTION:
Can we combine the second and third lines, excluding the M08 code? If so, why do we write the lines separate?

ANSWER:
Yes. The four G codes G00, G90, G54, and G43 all belong to different groups. Remember, no two G codes of the same group can be listed on the same line.

The main reason for using two lines is SAFETY. Remember, only one line of information can be executed at a time. The **X** and **Y** coordinates will position first, then the tool length and the **Z** coordinate will execute. If combined, all three axes will move simultaneously, and any interfering clamps or fixtures can be struck and/or destroyed. When combining **X**, **Y**, and **Z** in positioning, chances of crashing the machine are greater.

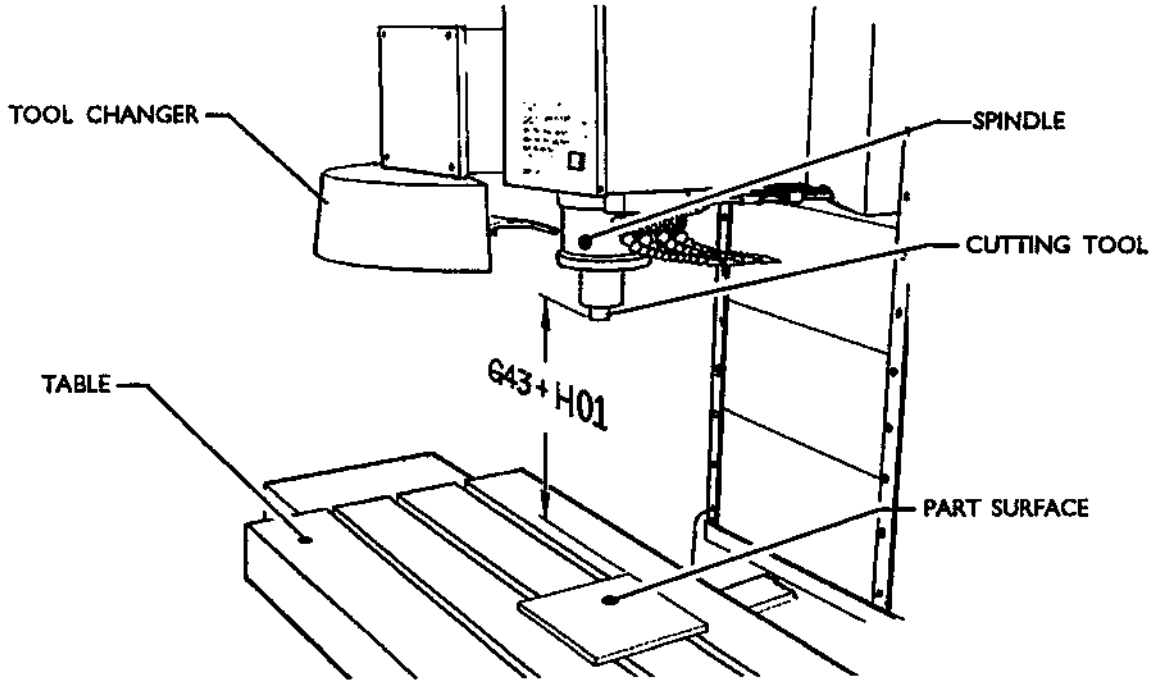


Fig. 1-9 *Tool length offset and tool length compensation.*

Tool number should always remain numerically matched with the tool length offset number. Setting 15 (the H & T agreement) will ensure the tool number and tool length offset will match. (Ex. T1 in line #1 should have H01 in line #3, and T2 should have H02 in line #3.)

1.7 Canned Cycles

A canned cycle is used to simplify programming of a part. Canned cycles are defined for most common Z-axis repetitive operations such as drilling, tapping, and boring. Once selected, a canned cycle is active until canceled with the G80 code. There are six operations involved in every canned cycle:

1) Positioning of **X** and **Y** axes (optional A, rotary axis).

2) Rapid traverse to the reference plane.

3) Drilling, boring, or tapping action.

4) Operation at the bottom of the hole.

5) Retraction to the reference plane.

6) Rapid traverse to the initial starting point.

A canned cycle is presently limited to operations in the Z-axis; that is, only the G17 plane is allowed. This means the canned cycle will be executed in the Z-axis whenever a new position is selected in the **X** or **Y** axis. The operation of a canned cycle will vary according to whether incremental (G91) or absolute (G90) is active. Incremental motion in a canned cycle is often useful as a loop count (L) and can be used to repeat the operation with an incremental **X** or **Y** move between each cycle.

G98 and G99 are modal commands which change the way the canned cycles operate. When G98 (the system default) is active, the Z-axis will be returned to the starting position at the completion of the canned cycle. When G99 is active, the Z-axis will be returned to the reference plane when the canned cycle is completed.

NOTE: If an **LQ** is in the canned cycle line, the cycle will not execute until the control reads an **X** or **Y** location.

For more detailed information on canned cycles, see Chapter II, Section 3.17.

2. Programming Examples

2.1 G81 Drilling Canned Cycle

FORMAT:

G81 Z- ___ F ___ R ___

Z = Position of the bottom of the hole being drilled.

F = Feed rate in inches per minute.

R = Reference plane, or a position placed above Z0.

NOTE: The **Z**, **F**, and **R** codes are required data for all canned cycles.

NOTE: The optional **X** and **Y** can be included in the canned cycle line. In most cases, this would be the location of the first hole to be drilled.

The following is the program to drill through the aluminum plate in Figure 2-1 on the facing page:

```
T1  M06
G00  G90 G54  X1.125  Y-1.875  S2500  M03
G43  H01  Z.1
G81  Z-.35  F15.  R.1
X2.0
X3.0  Y-3.0
X4.0  Y-5.625
X5.250  Y-1.375
G00  G80  Z1.0
M30
```

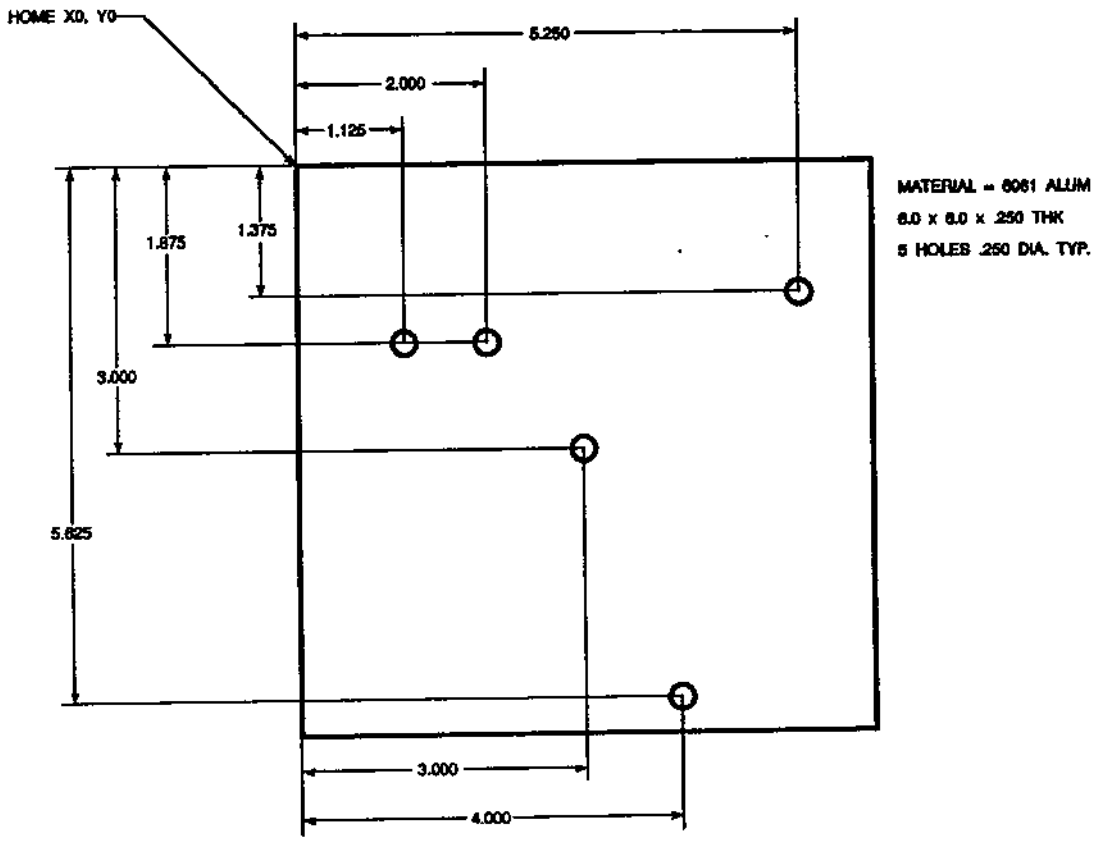


Fig. 2-1 *Canned cycle programming example using aluminum block.*

2.2 G82, G83, G84 Canned Cycles

G82 FORMAT -

G82 Z - ___ F ____ R ____ P ____;

These are the required codes for **spot drilling**.

P = Dwelling time in milliseconds at the bottom of the Z-axis move.

EX. P300 is approximately 1/3 second.

G83 FORMAT -

G83 Z - ___ F ____ R ____ Q ____;

These are the required codes for **peck drilling**.

Q = Incremental pecking amount in minus **Z** direction.

EX. Q.200 in a G83 line will peck toward the specified **Z** depth, taking .200 per peck until final depth is reached.

G84 FORMAT -

G84 Z - ___ F ____ R ____

These are the codes required for **tapping**.

No new codes to review.

NOTE: The most important item to be aware of during tapping is the speed and feed calculation.

FEED FORMULA: **Spindle Speed/Threads per inch on the tap = Feed Rate in inches/min.**

CANNED CYCLE PROGRAM

(HELPFUL NOTES)

```
%
O1234                                (Exercise program)
T1 M06                               (TOOL #1 IS A .5 X 90 degree spot drill)
G00 G90 G54 X.565 Y-1.875 S1275 M03
G43 H01 Z.1 M08
G82 Z-.175 F10. R.1 P300          >  (90 degree spot drill, the depth is)
X1.115 Y-2.750                       (half of the chamfer diameter)
X3.365 Y-2.875
X4.188 Y-3.313
X5.0 Y-4.0
G00 G80 Z1.0 M09
T2 M06                               (Tool #2 IS A .3125 stub drill)
G00 G90 G54 X.565 Y-1.875 S2500 M03
G43 H02 Z.1 M08
G83 Z-.620 F15. R.1 Q.175         >  (Drill point is 1/3 of the drill diameter.)
X1.115 Y-2.750
X3.365 Y-2.875
X4.188 Y-3.313
```

```
X5.0 Y-4.0
G00 G80 Z1.0 M09
T3 M06                          (Tool #3 is a 3/8-16 tap)
G00 G90 G54 X.565 Y-1.875 S900 M03
G43 H03 Z.2 M08
G84 Z-.600 F56.25 R.2           >  (900 RPM divided by 16 TPI = 56.25 IPM)
X1.115 Y-2.750
X3.365 Y-2.875
X4.188 Y-3.313
X5.0 Y-4.0
G00 G80 Z1.0 M09
G28 G91 Y0 Z0
M30
%
```

2.3 Subprograms and Canned Cycles

After reviewing the canned cycle, we can get a good idea of the amount of lines of code required to produce the five holes. The best way to conserve on program space and programming time is to use a subprogram. We can do this by grouping the **X** and **Y** locations of the holes into a separate program and then calling up this program when we need to tell a canned cycle the X,Y coordinates.

Instead of writing the X,Y locations once for each tool, we can write the X,Y locations once for any number of tools.

The canned cycle program that we reviewed on the previous page could use some constructive rearranging.

```
%
O1234 (Example program)
T1 M06
G00 G90 G54 X.565 Y-1.875 S1275 M03
G43 H01 Z.1 M08
G82 Z-.175 F10. R.1 P300
M98 P1000
G00 G80 Z1.0 M09
T2 M06
G00 G90 G54 X.565 Y-1.875 S2500 M03
G43 H02 Z.1 M08
G83 Z-.620 F15. R.1 Q.175
M98 P1000
G00 G80 Z1.0 M09
T3 M06
G00 G90 G54 X.565 Y-1.875 S900 M03
G43 H03 Z.2 M08
G84 Z-.600 F56.25 R.2
M98 P1000
G00 G80 Z1.0 M09
G28 G91 Y0 Z0
M30
%
```

```
%
O1000 (X,Y LOC. SUB)
X 1.115 Y-2.750
X 3.365 Y-2.875
X 4.188 Y-3.313
X 5.0 Y-4.0
M99
%
```

2.4 Subprograms With Multiple Fixtures

So far we have learned that using subprograms with canned cycles can save programming time and help reduce coordinate input error. Let's take this one step further. There are six vises mounted on the table. Each of these vises will use a new X,Y zero. They will be called up in the program as G54 through G59. The machine will have to be told where each of the vises is located on the table. By using an edge finder or an indicator, the zero point on each part can be established. Use the part zero set key in the work coordinate offset page to record each X,Y location. Once the X,Y zero position for each vise is in the offset page, the programming can begin.

By looking at the next page, we can get a good idea of what this setup would look like on the machine table.

For an example, each of these six parts will need to be drilled at the center. **X** and **Y** zero

```
%
O2000
T1 M06
G00 G90 G54 X0 Y0 S1500 M03
G43 H01 Z.1 M08
M98 P3000
G55
M98 P3000
G56
M98 P3000
G57
M98 P3000
G58
M98 P3000
G59
M98 P3000
G00 Z1.0 M09
G28 G91 Y0 Z0
M30
%
```

```
%
O3000
X0 Y0
G83 Z-1.0 F15. R.1 Q.2
G00 G80 Z.2
M99
%
```

The following illustration represents a multiple-fixture setup. Each vise will have an absolute zero once it is specified in the program. This is done by using G54 through G59 and G110 through G129, a total of 26 possible.

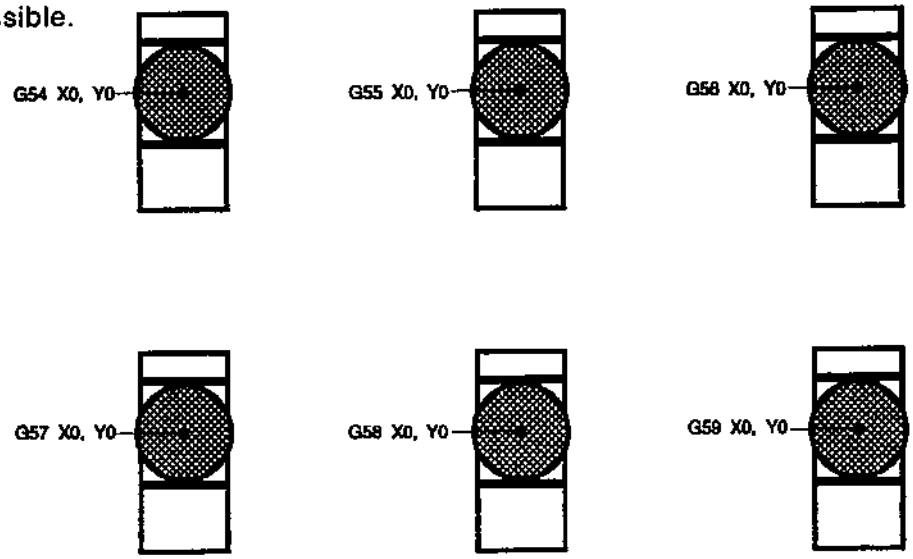


Fig. 2-2 *Multiple fixture setup.*

2.5 Looping Canned Cycles

Below is an example of a program using a drilling canned cycle that is incrementally looped. Compare the grid plate drawing on the next page to the program below.

```
%
O3400                          (Drilling grid plate)
T1 M06
G00 G90 G54 X1.0 Y-1.0 S2500 M03
G43 H01 Z.1 M08
G81 Z-1.5 F15. R.1
G91 X1.0 L9
G90 Y-2.0                      (Or stay in G91 and repeat Y-1.0)
G91 X-1.0 L9
G90 Y-3.0
G91 X1.0 L9
G90 Y-4.0
G91 X-1.0 L9
G90 Y-5.0
G91 X1.0 L9
G90 Y-6.0
G91 X-1.0 L9
G90 Y-7.0
G91 X1.0 L9
G90 Y-8.0
G91 X-1.0 L9
G90 Y-9.0
G91 X1.0 L9
G90 Y-10.0
G91 X-1.0 L9
G00 G90 G80 Z1.0 M09
G28 G91 Y0 Z0
M30
%
```

NOTE: The sequence of drilling used here is designed to save time and to follow the shortest path from hole to hole.

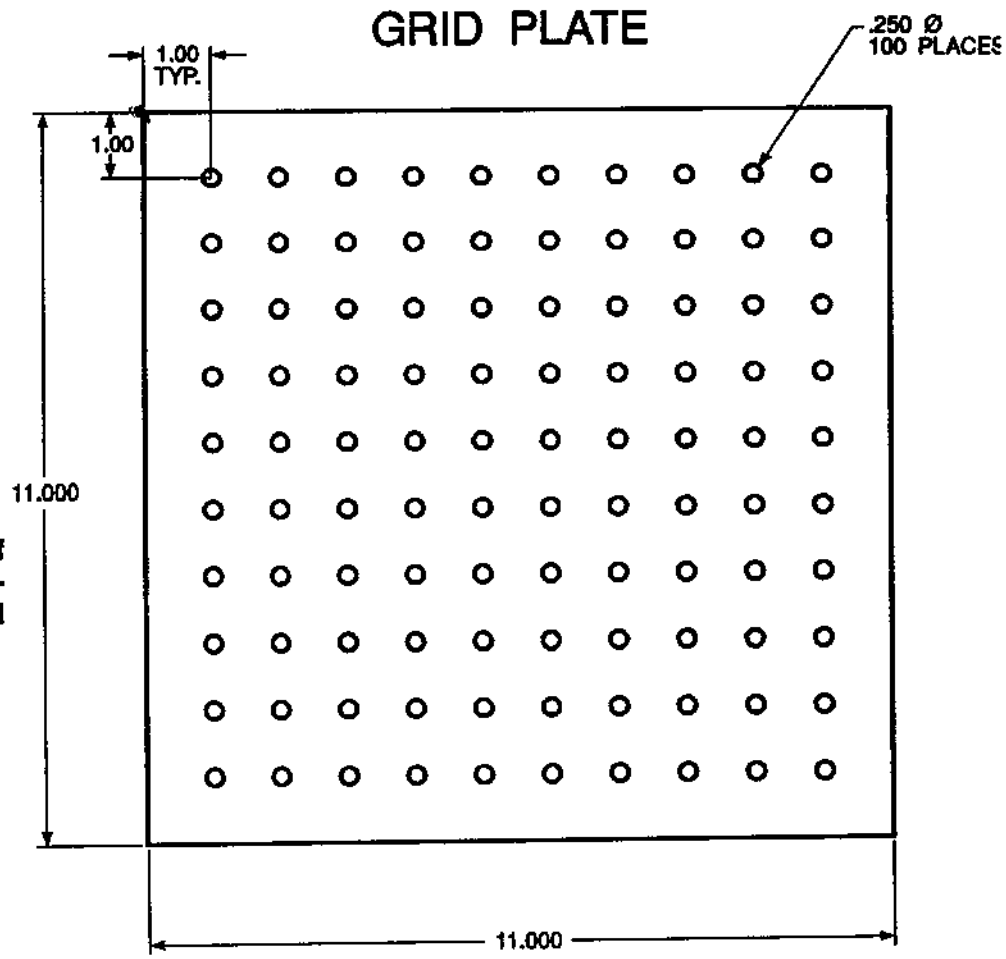


Fig. 2-3 *Grid plate for multiple-fixture subprogram exercise.*

2.6 Modifying Canned Cycles

In this section we will cover canned cycles that have to be customized in order to make the programming of difficult parts easier. In result, making the machining process more efficient.

Using G98 and G99 to clear clamps:

For example, we have a square part being held to the table with one inch tall table clamps. We need to write a program to clear the table clamps.

```
%
O4500
T1 M06
G00 G90 G54 X1.0 Y-1.0 S3500 M03
G43 H01 Z1.125 M08
G81 G99 Z-1.500 F20. R.05
X2.0 G98                    ( Will return to starting point after executing cycle )
X6.0 G99                    ( Will return to reference plane after executing cycle )
X8.0
X10.0
X12.0 G98
X16.0 G99
X18.0 G98
G00 G80 Z2.0 M09
G28 G91 Y0 Z0
M30
%
```

X,Y Plane Obstacle Avoidance In A Canned Cycle:

So far we have learned how G98 and G99 can be used to avoid an obstacle in the Z-axis. There is also a way to avoid an obstacle in the X,Y plane during a canned cycle by placing an L0 in a canned cycle line, we can tell the control to make an X,Y move without executing the Z-axis canned operation.

For example, we have a six inch square aluminum block, with a one inch by one inch deep flange on each side. The print calls for two holes centered on each side of the flange. We need to write a program to avoid each of the corners on the block.

```
%
O4600
(X0,Y0 is at the top left corner)
(Z0 is at the top of the part)
T1 M06
G00 G90 G54 X2.0 Y-.5 S3500 M03
G43 H01 Z-.9 M08
G81 Z-2.0 F15. R-.9
X4.0
X5.5 L0                     (angular corner avoidance)
Y-2.0
Y-4.0
Y-5.5 L0
X4.0
X2.0
X.5 L0
Y-4.0
Y-2.0
G00 G80 Z1.0 M09
G28 G91 Y0 Z0
M30
%
```

2.7 Special Canned Cycles

In this section, we will cover the special canned cycles that the Haas control offers. These canned cycles are used in conjunction with other drilling, boring, and tapping cycles.

G70 = BOLT HOLE CIRCLE

G71 = BOLT HOLE ARC

G72 = BOLT HOLES ALONG AN ANGLE

The sample program below will show the format for using a G70 to drill a three inch diameter bolt hole pattern combined with a G81 drilling canned cycle.

```
%
O5000
T1  M06
G00  G90  G54  X0  Y0  S1500  M03
G43  H01  Z.1  M08
G70  I1.5  J0  L8  G81  Z-1.0  F15.  R.1
G00  G80  Z1.0  M09
G28  G91  Y0  Z0
M30
%
```

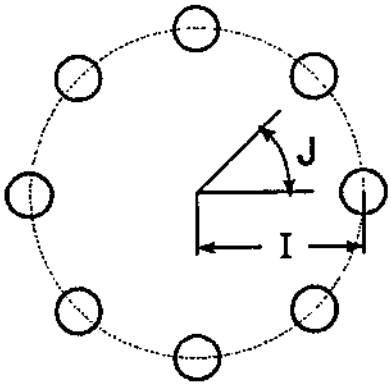
RULES FOR BOLT PATTERN CANNED CYCLES:

1. The tool must be placed at the center of the bolt pattern before the canned cycle execution. The center is usually X0, Y0.

2. The **J** code is the angular starting position and is always 0 to 360 degrees counterclockwise from the three o'clock position.

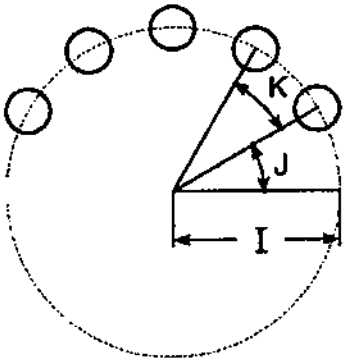
[See following page for illustrations of G70, G71, and G72.]

G70
BOLT HOLE CIRCLE



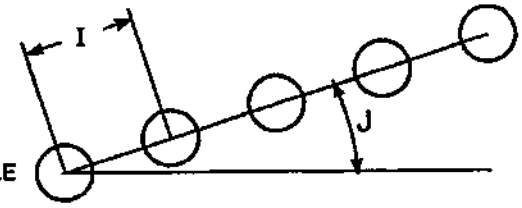
I = Radius of the bolt circle
J = Starting angle from three o'clock position
L = Number of holes (evenly spaced)

G71
BOLT HOLE ARC



I = Radius of the bolt circle
J = Starting angle from three o'clock position
K = Angular spacing between holes
L = Number of holes (evenly spaced)

G72
BOLT HOLES
ALONG AN ANGLE



I = Distance between holes
J = Angle from the three o'clock position
L = Number of holes

2.8 Circular Interpolation and Cutter Compensation

In this section, we will cover the usage of G02 (Circular Interpolation Clockwise) and G03 (Circular Interpolation Counterclockwise) and Cutter Compensation (G41: Cutter Compensation Left, G42: Cutter Compensation Right).

Using G02 and G03, we can program the machine to cut circular moves and radii. Generally, when programming a profile or a contour, the easiest way to describe a radius between two points is with an **R** and a value. For complete circular moves (360°), an **I** or a **J** with a value must be specified. The circle section illustration below will describe the different sections of a circle.

By using cutter compensation in this section we, the programmers, will be able to shift a cutter by the amount of the cutter radius and be able to program a profile or a contour to the exact print dimensions. By shifting the cutter radius, the programming time and the likelihood of calculation error is reduced.

Before we get into circular interpolation and how it is used, below are a few rules about cutter compensation that have to be closely followed in order to perform successful machining operations. Always refer to these rules when programming!

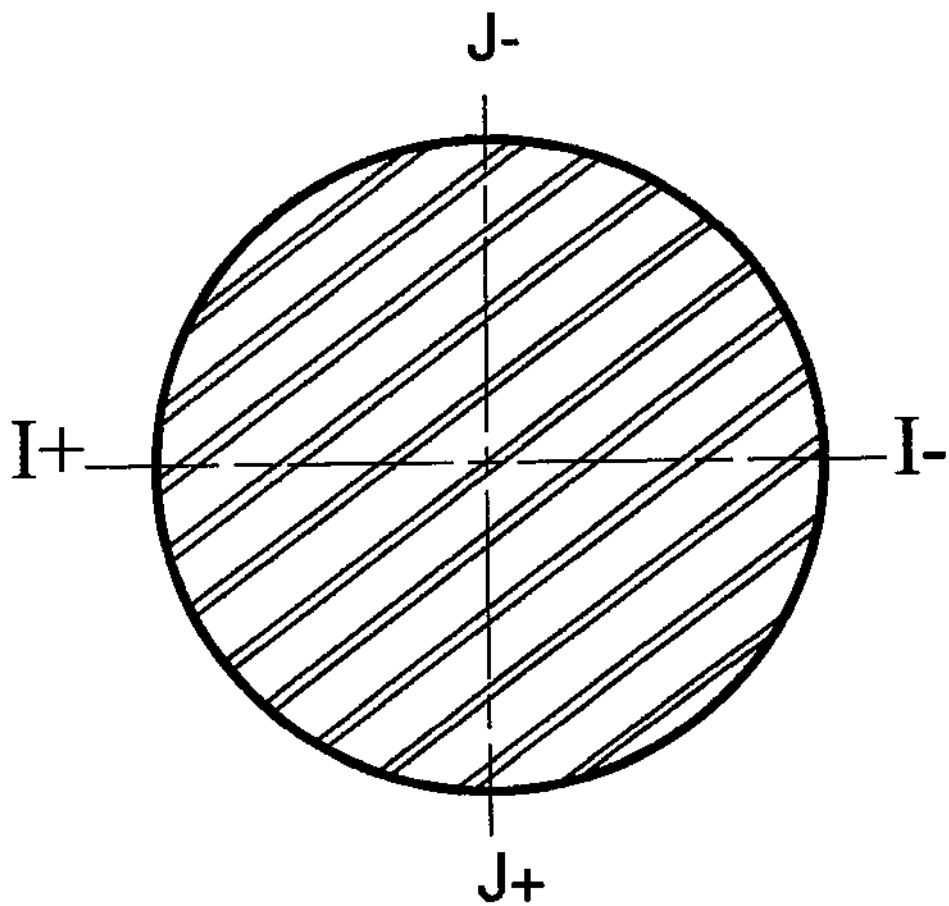
1) Cutter compensation must be turned ON during a G00 or G01 X,Y move that is equal to or greater than the cutter radius, or the amount being compensated for.

2) When an operation using cutter compensation is done, the cutter compensation will need to be turned OFF, using the same rules as the turn ON process, i.e., what is put in must be taken out.

3) In most machines, during cutter compensation, a linear X,Y move that is smaller than the cutter radius may not work. (Setting 58 - set to Fanuc - for positive results.)

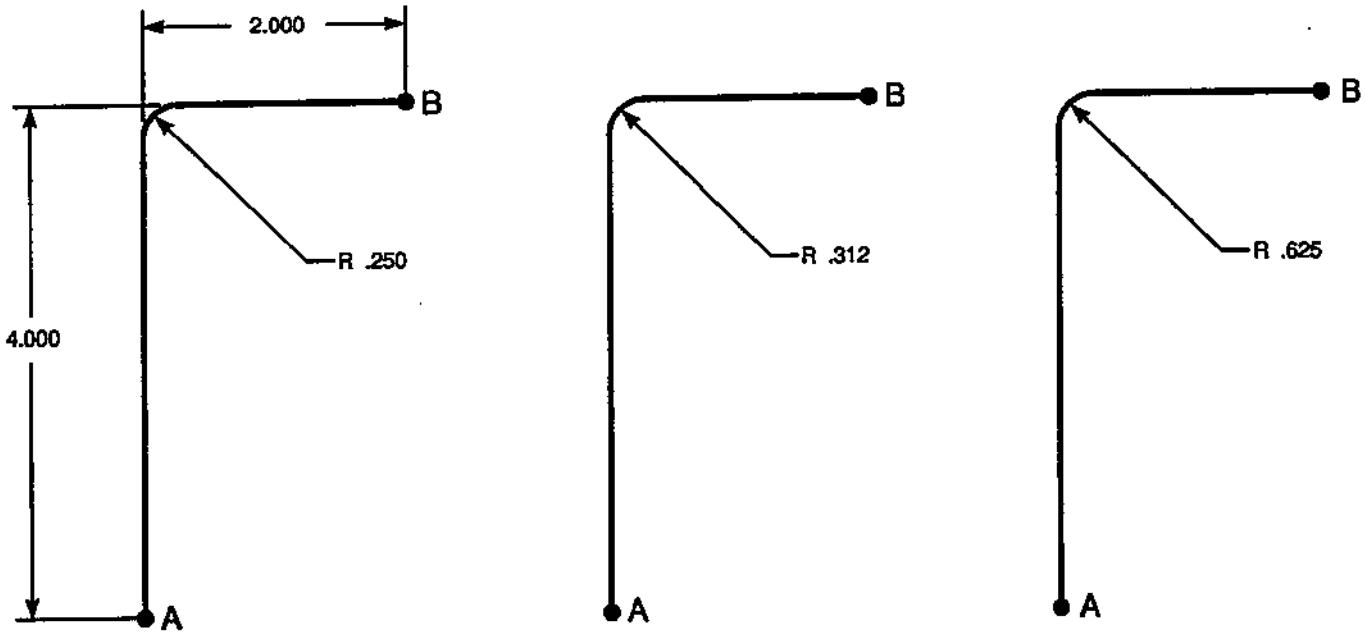
4) Cutter compensation cannot be turned ON or OFF in a G02 or G03 arc movement.

CIRCLE SECTIONS



DO THE FOLLOWING EXERCISE FOR PRACTICE.

WRITE THE CODE NECESSARY TO GET FROM POINT A TO POINT B



G01	Y3.75		
G02	X.250	Y4.0	R.250
G01	X2.0		

G01	[illegible]		
G02	X [illegible]		
G01	X2.0		

G[illegible]	[illegible]		
G[illegible]	[illegible]	[illegible]	[illegible]

FIRST MOVE: Overall dimension and subtract the given radius.

SECOND MOVE: Add radius to current **X** and **Y** values. Always program the ending points in circular interpolation.

THIRD MOVE: Linear move to the next point.

WRITE A PROGRAM TO PRODUCE THE PART SHOWN BELOW.

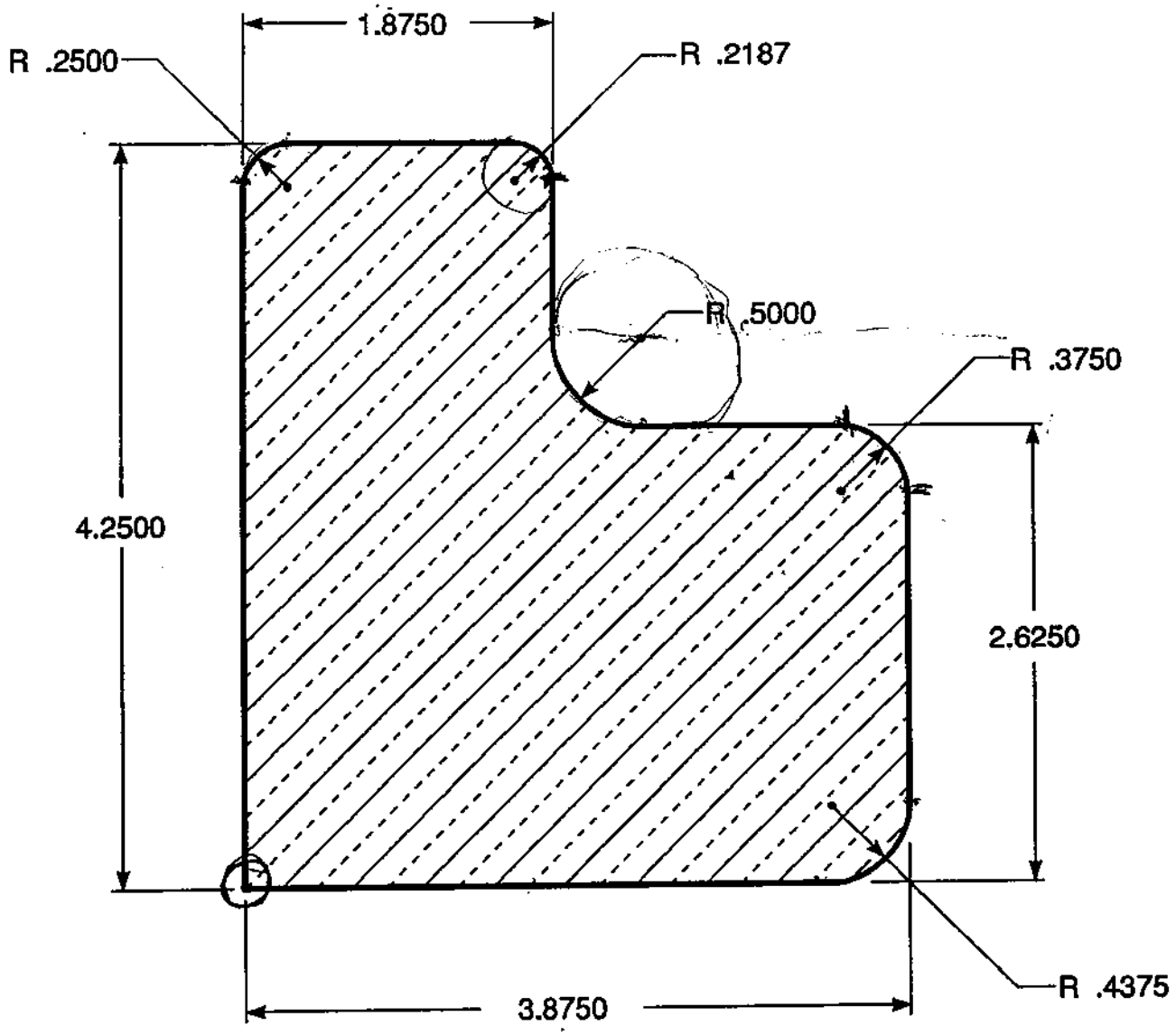


Fig. 2-4 *Part for G02, G03, and R practice.*

See following page for the correct program.

The correct program for Figure 2-4 is as follows:

```
%
O600
T1  M06                                  (Tool is a .250 diameter two-flute end mill)
G00  G90  G54  X-.2  Y-.2  S5000  M03
G43  H01  Z.1  M08
Z-1.0
G41  D01  X0                             (Turn ON cutter compensation, .200 move)
G01  Y4.0  F25
G02  X.250  Y4.250  R.250
G01  X1.6562
G02  X1.875  Y4.0313  R.2187
G01  Y3.125
G03  X2.375  Y2.625  R.500
G01  X3.5
G02  X3.875  Y2.25  R.375
G01  Y.4375
G02  X3.4375  Y0  R.4375
G01  X-.1
G00  G40  X-.3                           (Turn OFF cutter compensation, .200 move)
Z1.0  M09
G28  G91  Y0  Z0
M30
%
```

The following illustration shows how the tool path is calculated for the cutter compensation.

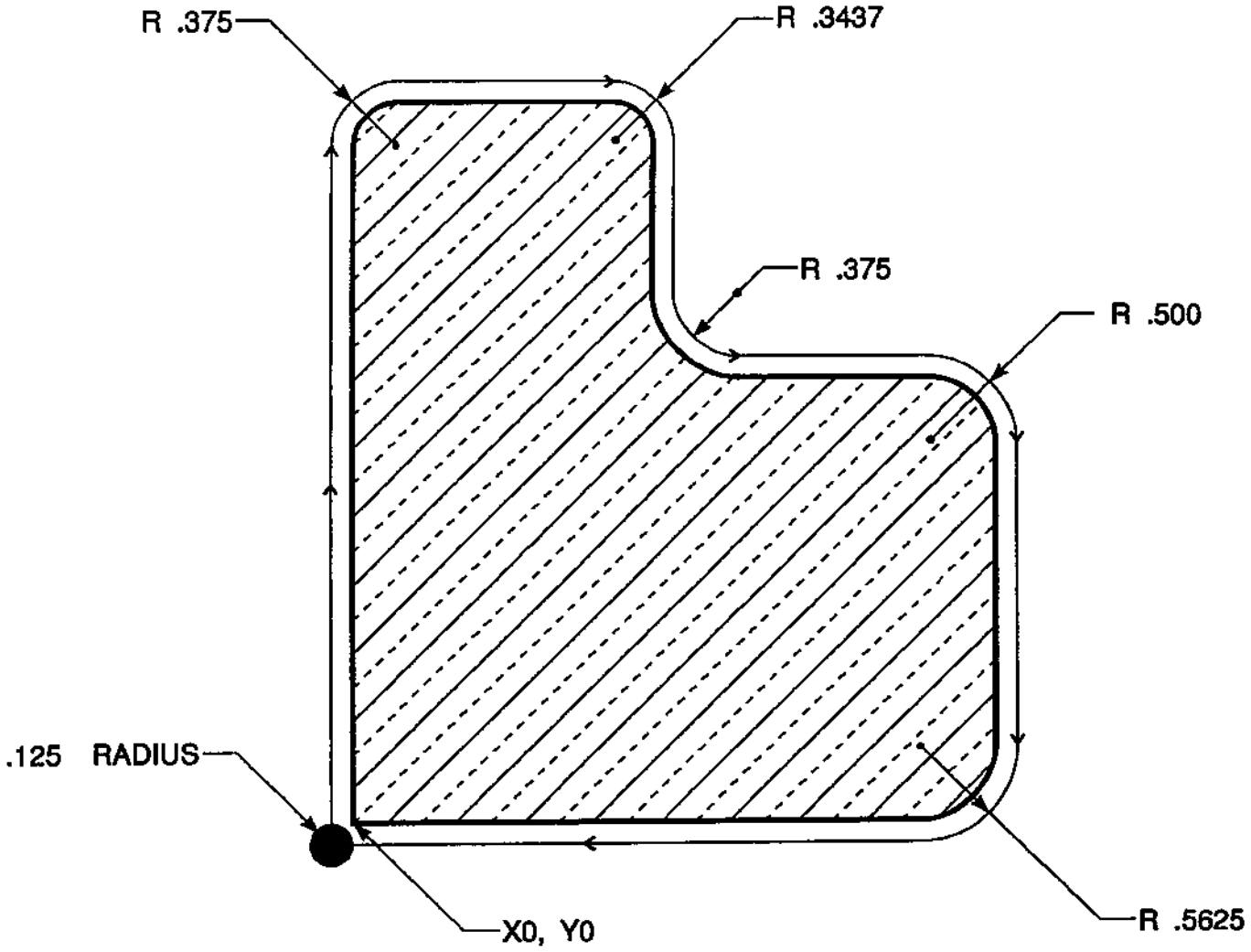


Fig. 2-5 *Programming exercise showing tool path.*

This program uses no cutter compensation. Tool path is programmed to centerline of the cutter. This is also the way the control calculates for cutter compensation. Note the machine tool path in Figure 2-5.

```
%
O6100
T1  M06
G00  G90  G54  X-.125  Y-.2  S5000  M03
G43  H01  Z.1  M08
G01  Z-1.0  F50
Y4.125  F25
G02  X.250  Y4.375  R.375
G01  X1.6562
G02  X2.0  Y4.0313  R.3437
G01  Y3.125
```

```
G03  X2.375  Y2.750  R.375
G01  X3.5
G02  X4.0  Y2.25  R.5
G01  Y.4375
G02  X3.4375  Y-.125  R.5625
G01  X-.2
G00  Z1.0  M09
G28  G91  Y0  Z0
M30
%
```

2.9 Programming Using I and J

Most contour machining will use a radius value **R** for circular interpolation moves less than 360°. An **I** and **J** can also be used in the place of **R**, but this can be more confusing at times. The **I** and **J** are signed distances from the starting point to the center of the circle.

Referring back to Section 2.8, we can see the program below using the **R** or the **I** and **J**:

Using **R**:

```
G01  Y3.750
G02  X.250  Y4.0  R.250
G01  X2.0
```

Using **I** and **J**:

```
G01 Y3.750
G02  X.250  Y4.0  I.250  J0
G01  X2.0
```

Note the +1.250 move and compare it with the circle sections illustration on page 22.

NOTE: The G02 or G03 line will always need the X,Y end points, whenever **R** is used or **I** and **J**.

Programming a complete 360° circle can only be done by using an **I** or a **J**. For example: we have a one inch diameter hole and want to open it up to one and a half inches. The cutter is .750 diameter. If we took the finished hole diameter and subtracted the cutter diameter, we would have .750 left over — .375 each side.

The start move would be to position to the center of the hole — most likely X0,Y0. The first cutting move would be to move .375 in any direction. X+, X-, Y+, Y-. Let's go with the X+ direction.

```
G01  X.375
G03  I-.375      (An I or J will always be a radial value.)
G01  X0
```

NOTE: The start point is also the end point for a 360° move.

NOTE: If we decided to make the first move in the Y+ direction, the G03 line would contain a J-.375 move.

2.10 Circular Pocket Milling

The Haas control has included in its software a Yasnac style circular pocket milling program (G12 clockwise circular pocket, G13 counterclockwise pocket). These G codes imply the use of cutter compensation, i.e., a G41 or G42 is not required to be stated in the program line. However, a D____ offset number for cutter radius or diameter is required for the ability to adjust the circle diameter.

In this section, we will cover the G12 and G13 format, as well as the different ways these programs can be written for many various applications.

SINGLE PASS: Using **I** only.
APPLICATIONS: One-pass counterboring; rough and finish pocketing of smaller holes; I.D. keyway cutting; "O"-ring grooves.

MULTIPLE PASS: Using **I**, **K**, and **Q**.
APPLICATIONS: Multiple-pass counterboring, rough and finish pocketing of large holes with cutter overlap.

MULTIPLE Z DEPTH PASS: Using **I** only, or, **I**, **K**, and **Q**. (G91 and **L**)
APPLICATIONS: Deep rough and finish pocketing; incremental **Z** depth stepping.

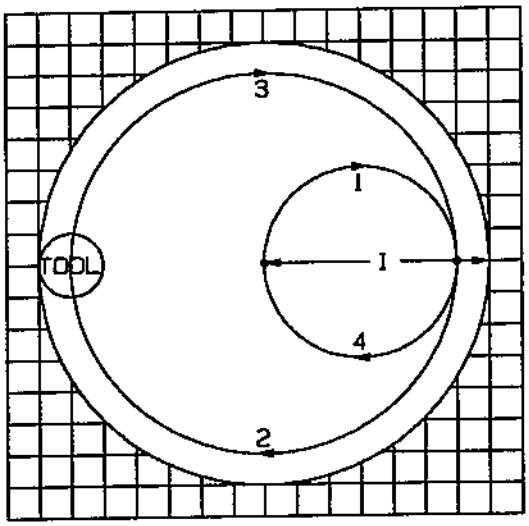
NOTE: The tool must be positioned at the center of the circle, either in a previous block or in the G12/G13 line by using **X** and **Y**.

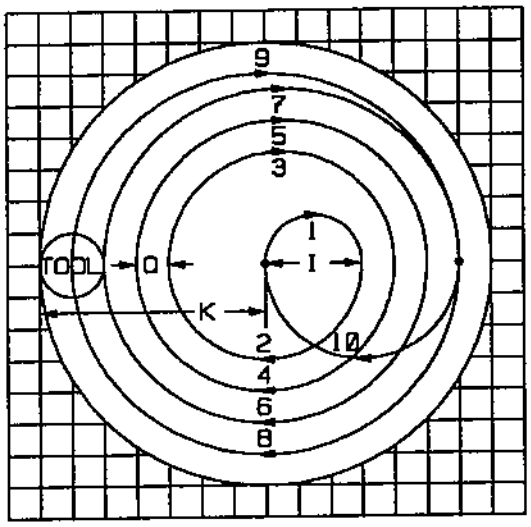
WARNING! Any software revision before 1.37, 2.8, and 3.2 will execute a rapid **Z** depth movement. Current software supports linear feed in the **Z** depth move.

The diagram on the next page shows the tool path during the G12 and G13 cycles. One uses **I** only and the other uses **I**, **K**, and **Q**.

Circular Pocket Milling

G12

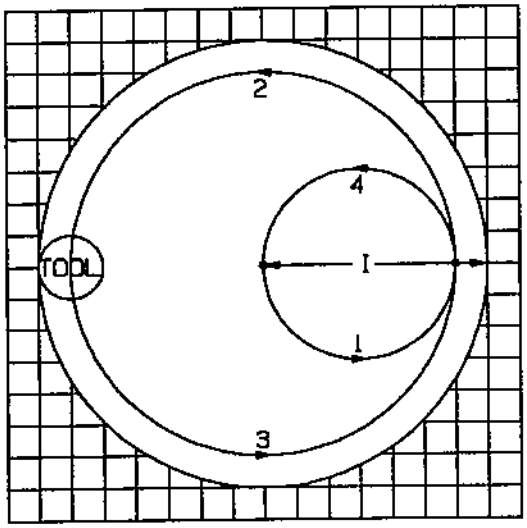


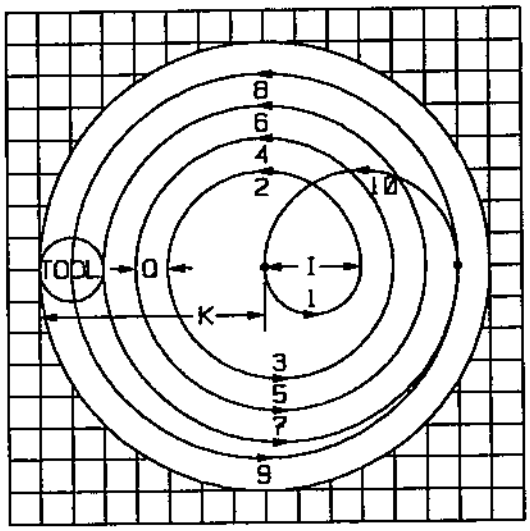


I ONLY

I, K, and Q

G13





PROGRAM LINE REQUIREMENTS:

Z = Depth of cut or increment.
F = Feed Rate.
I = Radius of first circle (Finished radius if no K specified.).
K = Radius of finished circle (If using I, K, and Q.).
Q = Radius increment or cutter overlap (Must use with K).
D = Tool geometry offset number (Not required).
L = Loop count for incremental Z depth stepping (Optional).

Ex. G13 single-pass using I only:

```
%
O2000                                   (.500 entered in the Radius/Diameter offset column)
T1  M06                                 (Tool #1 is a .500 diameter end mill)
G00  G90  G54  X0  Y0  S4000  M03
G43  H01  Z.1  M08
G13  Z-1.0  F20.  I.500  D01            (Will complete a one-inch diameter hole one-inch deep)
G00  Z1.0  M09
G28  G91  Y0  Z0
M30
%
```

Ex. G13 multiple-pass using **I**, **K**, and **Q**:

```
%
O3000                                   (.500 entered in the Radius/Diameter offset column)
T1  M06                                 (Tool #1 is a .500 diameter end mill)
G00  G90  G54  X0  Y0  S4000  M03
G43  H01  Z.1  M08
G13  Z-1.0  F20.  D01  I.400  K.500  Q.400
G00  Z1.0  M09
G28  G91  Y0  Z0
M30
%
```

This example will complete a three-inch diameter hole, one inch deep, with a cutter overlap of .100 thousandths of an inch.

Ex. G13 Multiple-pass using **I**, **K**, **Q**, **L**, and G91:

```
%
O4000                                   (.500 entered in the Radius/Diameter offset column)
T1  M06                                 (Tool #1 is a .500 diameter end mill)
G00  G90  G54  X0  Y0  S4000  M03
G43  H01  Z.1  M08
G01  Z0  F10.
G13  G91  Z-.5  F20.  D01  I.400  K2.0  Q.400  L4
G00  G90  Z1.0  M09
G28  G91  Y0  Z0
M30
%
```

The previous program uses G91 and an **L** count of **4**. This cycle will execute a total of four times. The **Z** depth increment is .500. This is multiplied by the **L** count, making the total depth of this hole 2.000.

The G91 and **L** count can also be used in G13 "**I** only" line.

NOTE: If the geometry column has a value inserted within, the circular pocket milling cycle will automatically read the data, regardless of the D01 being present or not. The only effective way to cancel the cutter compensation for the pocket milling is to insert a D00 in the program line. This will bypass the value in the geometry column.

2.11 General Purpose Pocket Milling

The general purpose pocket milling program is has been included in the Haas control. This program is used to mill irregular shapes and is capable of leaving islands and bosses within a contour. With the G150, there is a main program for technical input and a subprogram for contour definition.

G150 FORMAT EXAMPLE:

```
%
O4500
T1  M06
G00  G90  G54  X0  Y0 S3500  M03
G43  H01  Z.1  M08
G150  X__  Y__  Z__  F__  R__  Q__  I__ OR J__  K__  P4600  D__  G41 OR G42
G00  Z1.0  M09
G28 G40  G91  Y0  Z0
M30
%


%
O4600
G01  X__  Y__
X__
Y__
X__  Y__
M99
%
```

PROGRAM LINE REQUIREMENTS:

X = X-axis position of the starting hole.
Y = Y-axis position of the starting hole.
Z = Final depth of the hole.
F = Feed rate.
R = Reference plane.
Q = Incremental Z-axis cut depth per pass.
I = X-axis cut increment.
J = Y-axis cut increment.
K = finish cut allowance.
P = Subprogram number.
D = Geometry offset number.
G41 or G42 = Cutter compensation turn **ON**.

The shape of the pocket to be cut must be defined by a series of motions within a subprogram. One of either **I** or **J** must be specified. If **I** is used, the pocket is cut from a series of strokes in the X-axis. If **J** is used, the pocket is cut from a series of strokes in the Y-axis. The value entered with the **I** or **J** will be the shift amount or cutter overlap. The **K** amount is the finishing allowance for the walls of the pocket.

The subprogram must define a closed area by a series of G01, G02, or G03 motions on X- and Y-axes, and must end with an M99. The only other codes that can be used in the subprogram are: G90, G91, **I**, **J**, **R**, **X**, and **Y**. Any other codes are ignored. This subprogram must not exceed 20 strokes.

NOTE: When defining the contour in the subprogram, the idea to keep in mind is to only connect the contour — not to return to the starting point.

G150 EXAMPLES -

4.0 x 4.0 x .500 DP. SQUARE POCKET:

```
%
O1000
T1  M06                          (Tool #1 is a .500 diameter end mill)
G00  G90  G54  S2000  M03
G43  H01  Z.1  M08
G01  Z0  F10.
G150  X0  Y0  Z-.5  F10.  R.1  Q.25  I.4  K.01  P500  D01  G41
G00  Z1.0  M09
G40  G28  G91  Y0  Z0
M30
%
```

ABSOLUTE SUBPROGRAM:

```
%
O0500
G01  Y2.0
X-2.0
Y-2.0
X2.0
Y2.0
X0
M99
%
```

INCREMENTAL SUBPROGRAM:

```
%
O0500
G01  G91  Y2.0
X-2.0
Y-4.0
X4.0
Y4.0
X-2.0
G90
M99
%
```

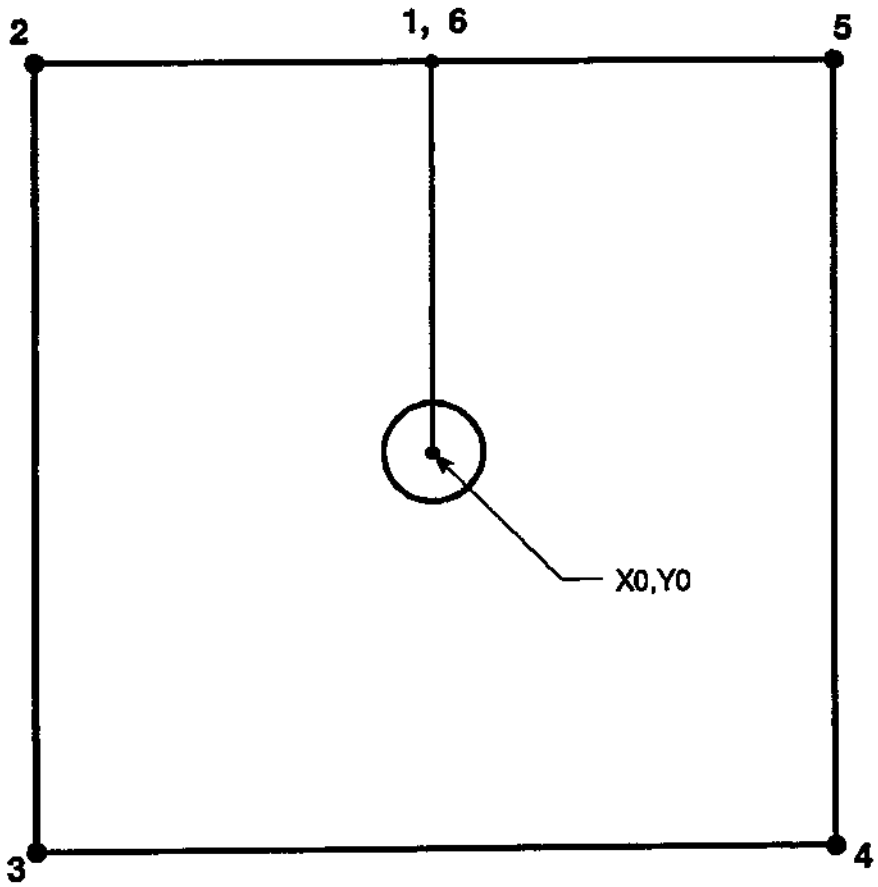


Fig. 2-6 *Pocket milling exercise for G150 operation.*

SQUARE ISLAND:

```
%
O1000
T1  M06                          (Tool is a .500 diameter end mill)
G00  G90  G54  S2500  M03
G43  H01  Z.1  M08
G01  Z0  F10.
G150  X1.0  Y1.0  Z-.5  F15.  R.1  Q.25  I.4  K.O1  P500  D01  G41
G00  Z1.0  M09
G40  G28  G91  Y0  Z0
M30
%


%
O0500
G01  X0  Y0
X6.0
Y6.0
X0
Y3.0
X2.0
Y4.0
X4.0
Y2.0
X2.0
Y3.0
X0
Y0
M99
%
```

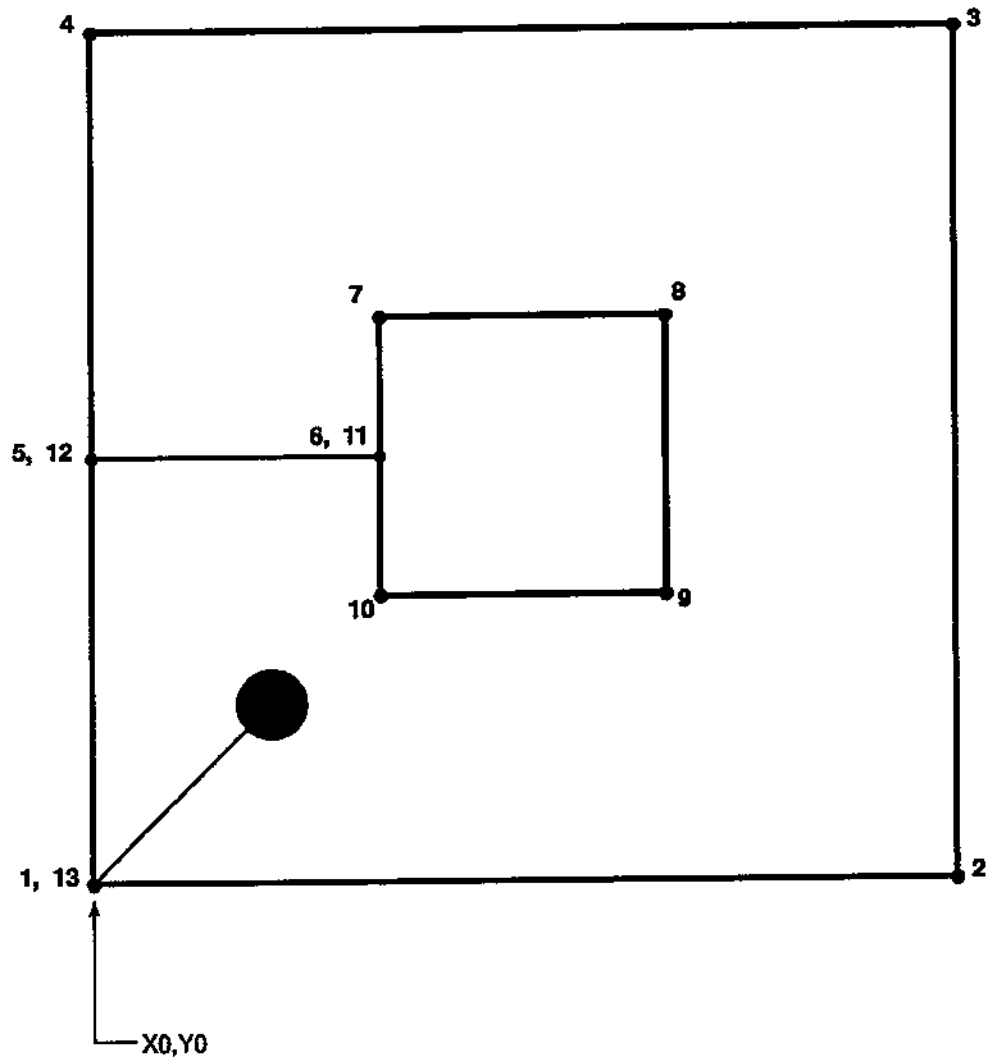


Fig. 2-7 *Square island programming exercise using G150.*

ROUND ISLAND:

```
%
O1000
T1  M06                          (Tool is a .500 diameter end mill)
G00  G90  G54  S2500  M03
G43  H01  Z.1  M08
G01  Z0  F10.
G150  X1.0  Y1.0  Z-.5  F15.  R.1  Q.25  I.4  K.01  P500  D01  G41
G00  Z1.0  M09
G40  G28  G91  Y0  Z0
M30
%

%
O0500
G01  X0  Y0
X6.0
Y6.0
X0
Y3.0
X2.0
G02  I1.0
G01  X0
Y0
M99
%
```

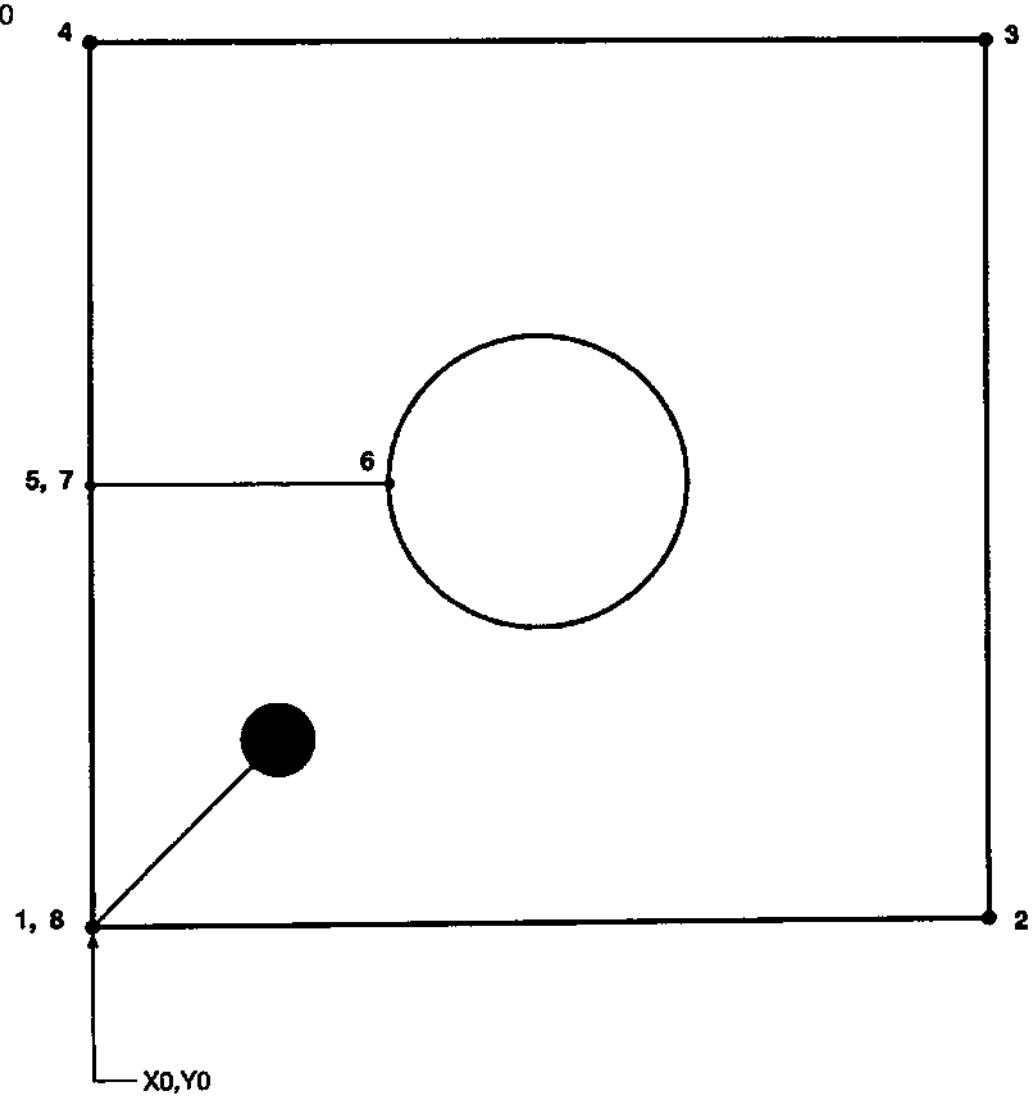


Fig. 2-8 *Round island programming exercise using G150.*

2.12 Programmable Mirror Image

Programmable mirror image can be turned on or off individually for any of the four axes. The two codes are non-modal, but the mirror status of each axis is modal. The bottom of the CRT screen will indicate when an axis is mirrored. These codes should be used in a command block without any other **G** codes and will not cause any axis movement. G101 will turn off mirror image for any axis listed in that block. The actual value given for the **X**, **Y**, **Z**, or **A** code has no effect and should be entered as zero value.

Ex.
G101 X0 = Will turn on mirror image for the X-axis.
G100 X0 = Will turn off mirror image for the X-axis.

Most mirror image applications would consist of irregular pockets and contours and would most likely be set up in subprograms for convenience.

NOTE: The first pocket or contour will need to be run before the mirror image function can duplicate the geometry. After completion of the first item, a Z-axis clearance move should be made. Then, the mirror image should be turned on with an axis specification. The following line needs the coordinates of the starting location of the original pocket. The following line will feed to the required Z-axis depth, the next line would contain a subprogram call or a contour definition, and last, a positive Z-axis clearance move.

The pockets should be arranged around a given origin, usually described as X0,Y0.

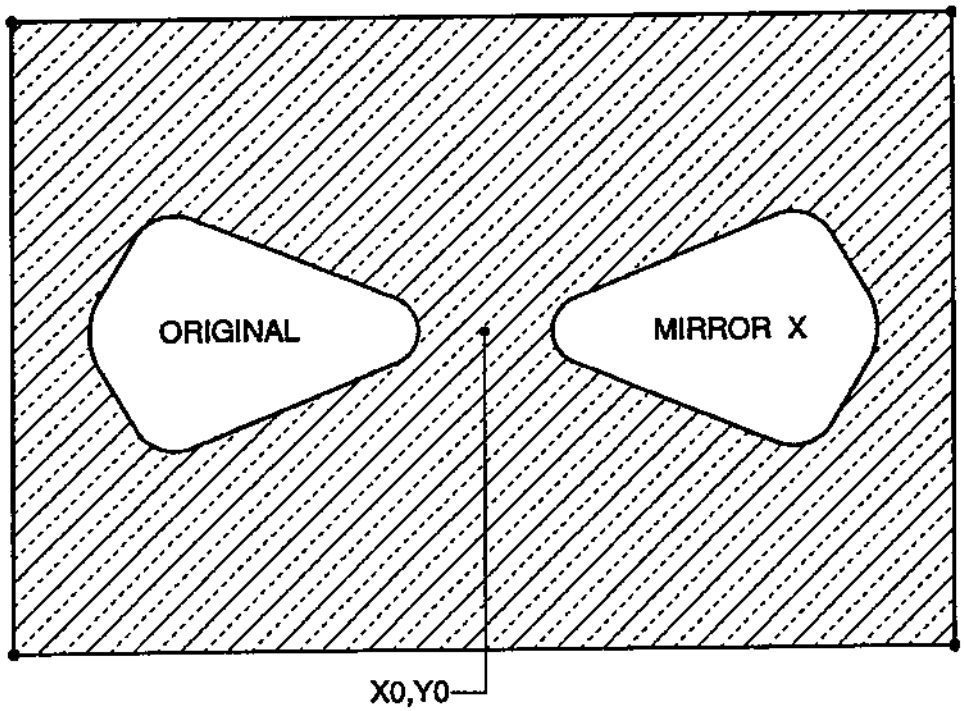


Fig. 2-9 *Mirror image and pocket milling exercise.*

NOTE: When milling a shape with X-Y motions, turning **on** MIRROR IMAGE for just one of the X and Y will change climb milling to conventional milling and/or conventional milling to climb milling. As a result, you may not get the type of cut or finish that was desired. Mirror image of both X and Y will eliminate this problem.

PROGRAM CODE FOR MIRROR IMAGE IN X-AXIS:

```
%
O3600                          (Mirror image X-axis)
T1 M06                         (Tool #1 is a .250 diameter end mill)
G00 G90 G54 X-.4653 7.052 S5000 M03
G43 H01 Z.1 M08
G01 Z-.25 F5.
```

```
F20.
M98  P3601
G00  Z.1
G101  X0.
X-.4653  Y.052
G01  Z-.25  F5.
F20.
M98  P3601
G00 Z.1
G100 X0.
G28  G91  Y0  Z0
M30
%

%
O3601                                (Contour subprogram)
G01  X-1.2153  Y.552
G03  X-1.3059  Y.528  R.0625
G01  X-1.5559  Y.028
G03  X-1.5559  Y-.028  R.0625
G01  X-1.3059  Y-.528
G03  X-1.2153  Y-.552  R.0625
G01  X-.4653  Y-.052
G03  X-.4653  Y.052  R.0625
M99
%
```

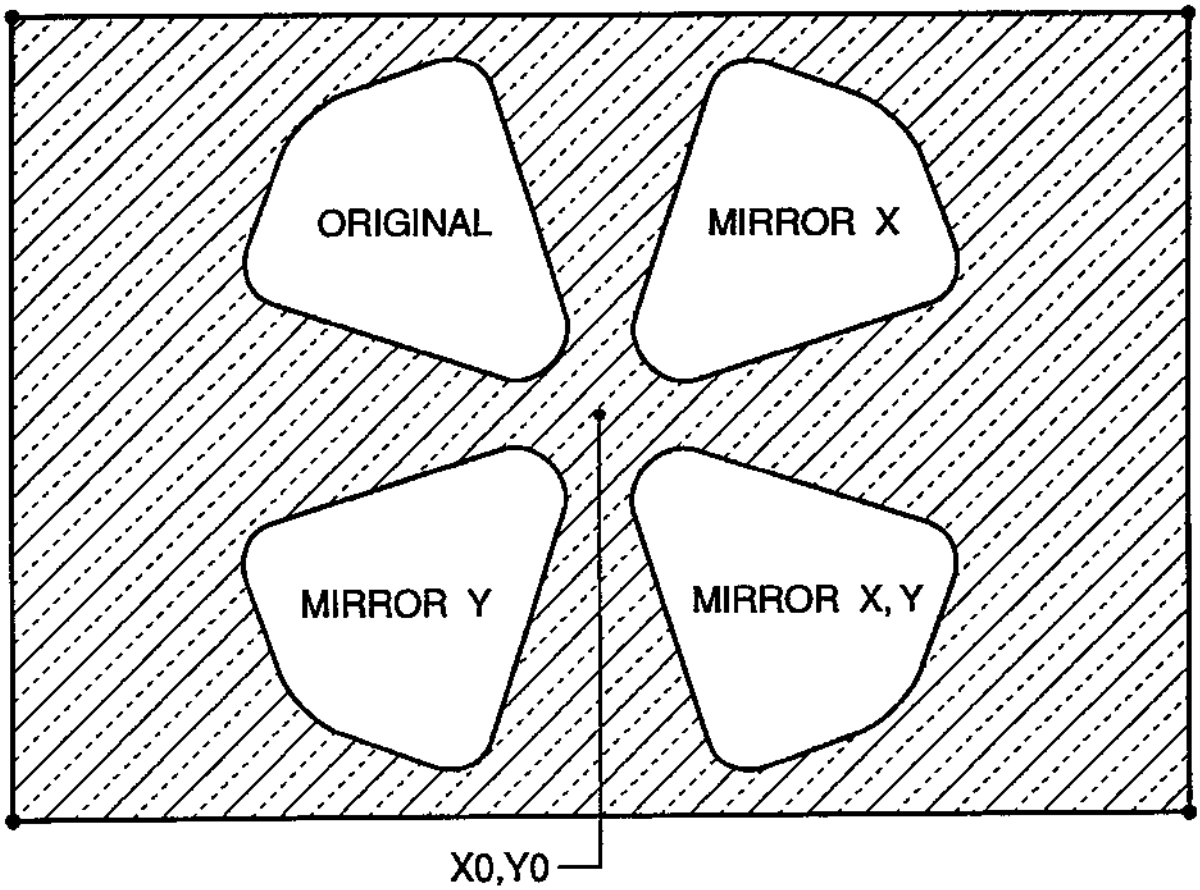


Fig. 2-10 *Mirror image (X, Y, and X-Y) and pocket milling exercise.*

PROGRAM CODE FOR MIRROR IMAGE IN THE X, Y, AND XY AXES:

```
%
O3700                                   (Mirror image X, Y, and XY axes)
T1  M06                                 (Tool #1 is a .250 diameter end mill)
G00  G90  G54  X-.2923  Y.3658  S5000  M03
G43  H01  Z.1  M08
G01  Z-.25  F5.
F20.
M98  P3701
G00  Z.1
G101  X0.                               (Turn on mirror image X-axis)
X-.2923  Y.3658                         (Position to original coordinates)
G01  Z-.25  F5.                         (Feed to Z depth)
F20.                                    (Pocket feed rate)
M98  P3701                              (Pocket contour subprogram call)
G00  Z.1                                (Part clearance)
G100  X0.                               (Cancel mirror image X-axis)
G101  Y0.                               (Turn on mirror image Y-axis)
X-.2923  Y.3658
G01  Z-.25  F5.
F20.
M98  P3701
G00  Z.1
G100  Y0.                               (Cancel mirror image Y-axis)
G101  X0.  Y0.                          (Turn on mirror image X and Y axes)
X-.2923  Y.3658
G01  Z-.25  F5.
F20.
M98  P3701
G00  Z.1
G100  X0.  Y0.                          (Cancel mirror image X and Y axes)
G28  G91  Y0  Z0
M30

O3701                                   (Contour subprogram)
G01  X-.469  Y1.2497
G03  X-.5501  Y1.2967  R.0625
G01  X-1.0804  Y1.12
G03  X-1.12  Y1.0804  R.0625
G01  X-1.2967  Y.5501
G03  X-1.2497  Y.469  R.0625
G01  X-.3658  Y.2923
G03  X-.2923  Y.3658  R.0625
M99
%
```

2.13 Thread Milling

We will use the following example and go through the thread milling procedures step-by-step to get the desired result:

DATA:

- I.D. Thread milling a 1.5 x 8 TPI hole.
- Using .750 diameter x 1.0 thread hob.
- Take the hole diameter 1.500.
- Subtract cutter diameter .750 = .750 Then divide by 2 = .375.

STEP 1: Within this space we need to turn on cutter compensation and ramp on to the circle to be machined.

STEP 2: Perform complete circle while simultaneously moving in the Z-axis the amount of one full pitch of the thread. This is called **helical interpolation**.

STEP 3: Ramp off the circle and turn off the cutter compensation.

NOTE: Always climb cut the cutter.
I.D. will be G03; O.D. will be G02.
An **I.D.** right hand thread will move **up** in the Z-axis by the amount of one thread pitch.
An **O.D.** right hand thread will move **down** in the Z-axis by the amount of one thread pitch.
PITCH = 1.0/Threads per inch

Ex. 1.0 divided by 8 TPI = .125

Cutter compensation cannot be turned off or on during an arc movement. A linear turn on and turn off movement must be made, either in the X- or Y-axis. This move will be the maximum compensation amount that can be adjusted.

I.D. THREAD MILLING

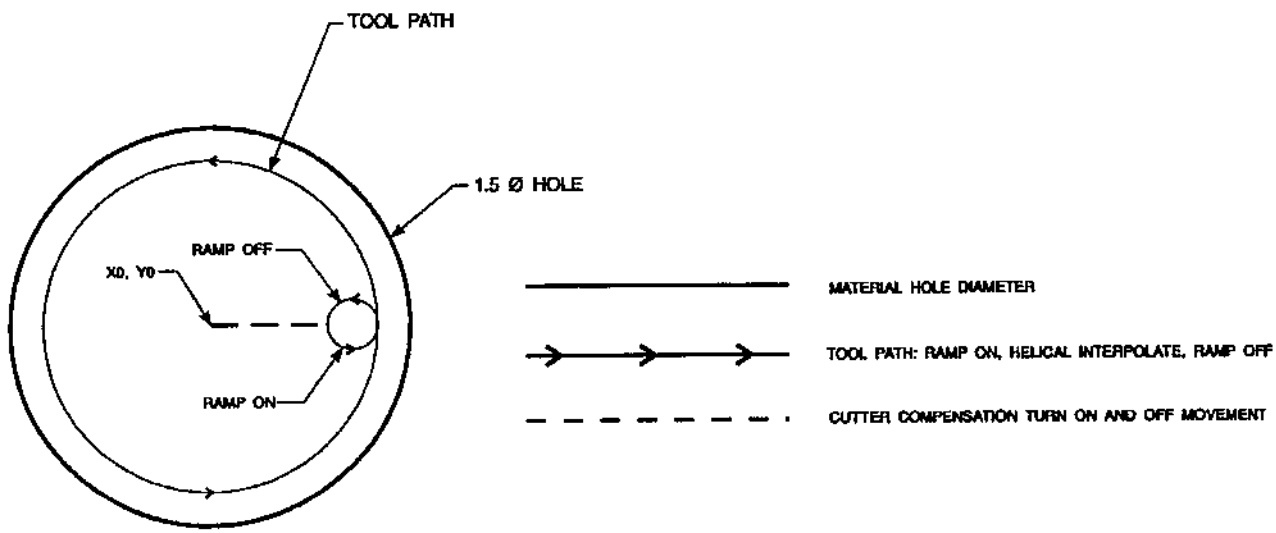


Fig. 2-11 *Thread milling exercise.*

The code for Figure 2-11 (previous page) is as follows:

```
%
O2300                                   (Thread milling 1.5 diameter x 8 TPI)
(X0, Y0 is at the center of the hole)
(Z0 is at the top of the part)
(Using .5 thick material)
G00  G90  G54  X0  Y0  S400  M03
G43  H01  Z.1  M08
Z-.6
G01  G41  D01  X.175  F25.
G03  X.375  R.200  F7.
G03  I-.375  Z-.475
G03  X.175  R.200
G01  G40  X0  Y0
G00  Z1.0  M09
G28  G91  Y0  Z0
M30
%
```

NOTE: Maximum cutter compensation adjustability is .175, which is more than enough for this application.

Start with zero in the diameter offset column and enter a negative number to increase the thread diameter.

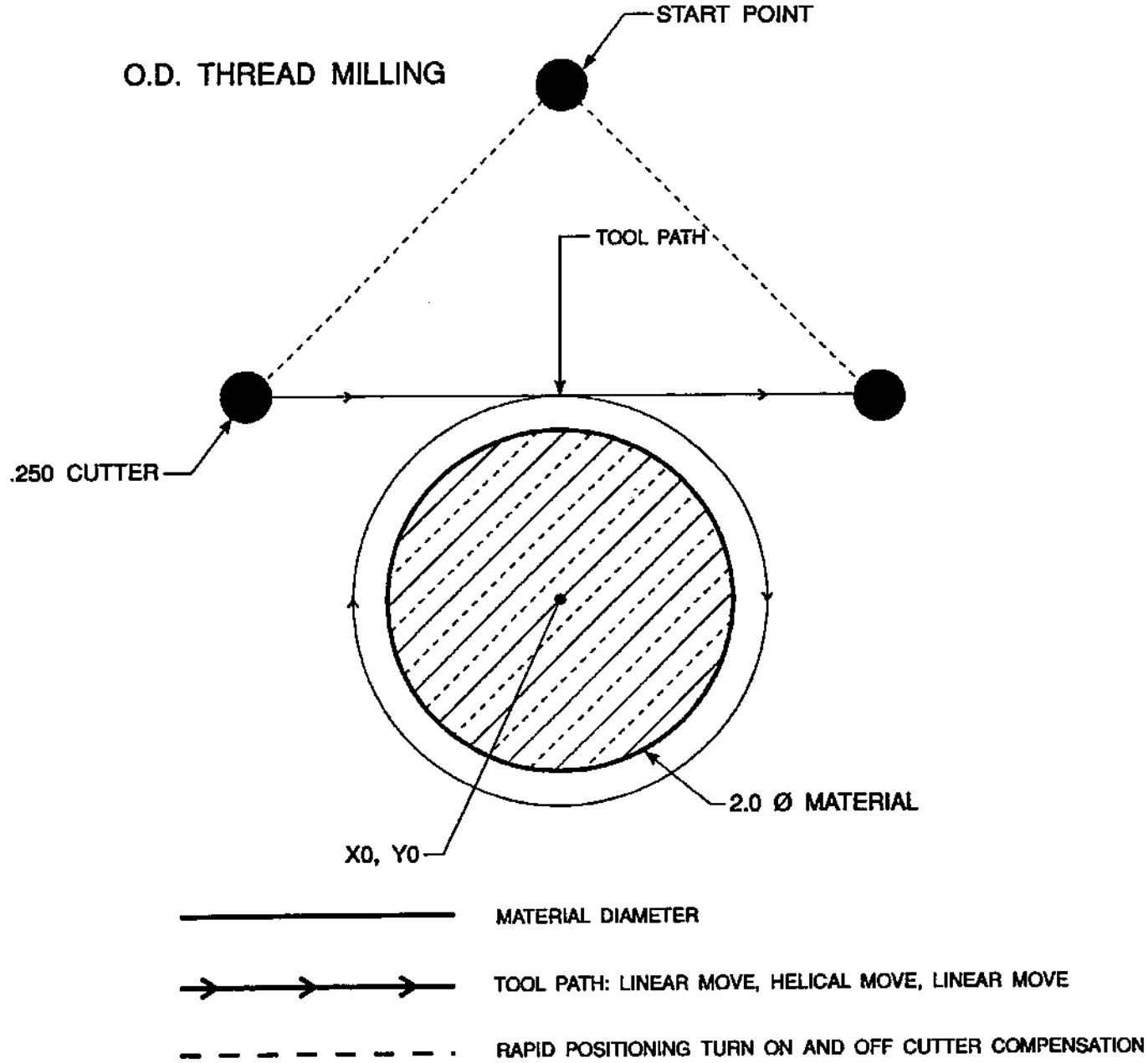


Fig. 2-12 *O.D. thread milling exercise.*

O.D. THREAD MILLING -

The code for the previous illustration is as follows:

```
%
O2400                                  (Thread milling a 2.0 diameter post x 16 TPI)
(X0,Y0 is at the center of the post)
(Z0 is at the top of the part)
(Post height is 1.125 inch)
G00  G90  G54  X0  Y2.0  S2000  M03
G43  H01  Z.1  M08
Z-1.0
G41  D01  X-1.5  Y1.125                (Turning on cutter compensation.)
G01  X0.  F15.                         (Linear interpolation onto the post.)
```

```
G02  J-1.125  Z-1.0625          (360° helical circle; negative Z move.)
G01  X1.5                       (Linear interpolation off the post.)
G00  G40  X0  Y2.0              (Turning off cutter compensation.)
Z1.0  M09
G28  G91  Y0  Z0
M30
%
```

NOTE: A cutter compensation turn on move can consist of any X or Y move from any position just as long as the move is greater than the amount being compensated for. The same rule applies for turning off cutter compensation.

2.14 Single-Point Thread Milling

Using the following data, we will write a program for single-point thread milling procedures:

DATA:

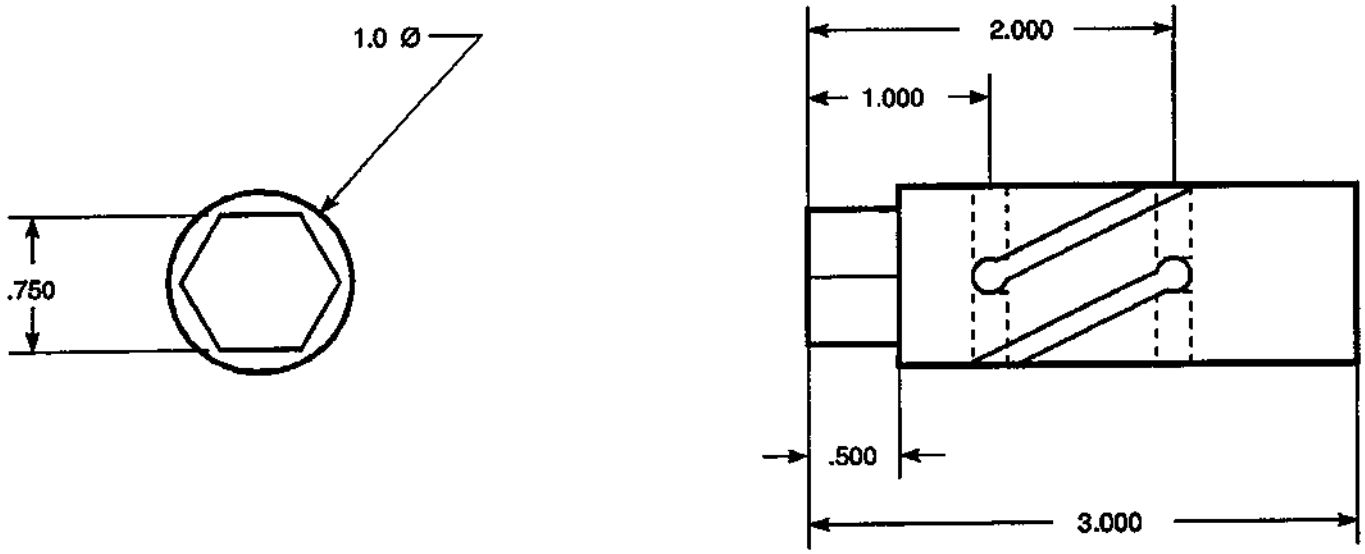
- 2.500 Ø hole
- Diameter of cutter (Subtract .750): 1.75
- Radial value (Divide by 2): .875
- Thread pitch: .0833 (12 TPI)
- Part thickness: 1.00

```
%
O1000                           (Main program)
(X0,Y0 is at the center of the hole)
(Z0 is at the top of the part)
T1  M06                         (Tool #1 is a .750 diameter single-point thread tool)
G00  G90  G54  X0  Y0  S2500  M03
G43  H01  Z.1  M08
G01  Z-1.083  F35.
G01  X.875  F15.                (Radial value)
M98  P1001  L14                 (Multiply .0833 pitch x 14 passes = 1.1662 = total in Z-axis)
G00  G90  Z1.0  M09
G28  G91  Y0  Z0
M30
%
```

```
%
O1001                           (Helical subprogram)
G91
G03  I-.875  Z.0833
M99
%
```

2.15 Fourth Axis Programming

The following is an example of fourth axis programming. Refer to Figure 2-13 for the program code on page 40.



EIGHT (8) HOLES CHAMFERED (.2 Ø)

EIGHT (8) HOLES .1875 Ø, 90° SPACING, .4 DP.

FOUR (4) CHANNELS, .125 WIDE

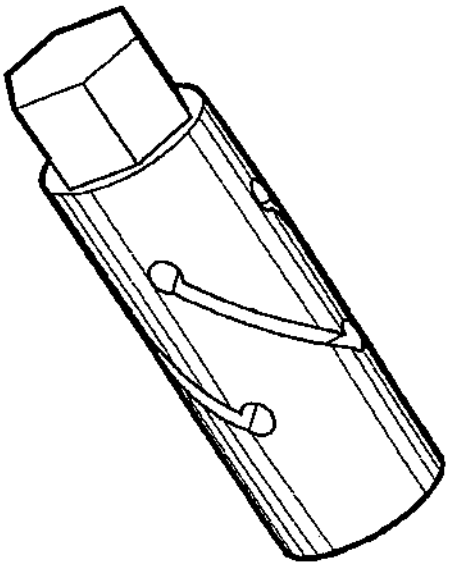


Fig. 2-13 *Fourth axis programming exercise.*

PROGRAM CODE FOR FIG. 2-19:

```
%
O1234                                          (Fourth axis program using a Haas Servo 5C)
(Material is 1.0 Ø round stock x 3.0 L)
(Set material in collet to protrude 2.25.)
(Set fixture parallel with the table T-slots on the right side.)
(X0 is the front of the material.)
(Y0 is the centerline of the spindle and material.)
(Z0 is the top of the part.)
T1  M06                                        (Tool #1 is a .500 end mill to mill hex.)
G00  G90  G54  X.250  Y-.500  A0  S4500  M03
G43  H01  Z.1  M08
M98  P1235  L6
G00  G90  Z.1  M09
T2  M06                                        (Tool #2 is a .375 Ø NC spot drill.)
G00 G90 G54 X1.0  Y0  A0  S5000  M03
G43  H02  Z.1  M08
G82  Z-.1  F10.  R.1  P300
X2.0
A90.
X1.0
A180.
X2.0
A270.
X1.0
G00  G80  Z.1  M09
T3  M06                                        (Tool #3 is a .1875 Ø stub twist drill.)
G00  G90  G54  X1.0  Y0  A0  S5000  M03
G43  H03  Z.1  M08
G83  Z-1.125  F12.  R.1  Q.25
X2.0
A90.
X1.0
G00  G80  Z.1  M09
T4  M06                                        (Tool #4 is a .125 Ø end mill.)
G00  G90  G54  X1.0  Y0  A0  S5000  M03
G43  H04  Z.1  M08
M98  P1236
G00  G90  Z.1  M09
G28  G91  Y0  Z0
M30
%

%
O1235                                          (Subprogram to mill hex.)
G01  Z-.125  F50.
Y.5  F35.
G00  Z.1
G91  Y-1.0  A60.
G90
M99
%
```

The following subprogram can be written in absolute or incremental programming. Examine each program and determine which style would be faster and easier to understand and to program in the future.

ABSOLUTE:

```
%
O1236  (Subprogram to mill channels.)
G01  Z-.25  F15.
X2.0  A90.
G00  Z.1
A180.
G01  Z-.25
X1.0  A90.
G00  Z.1
A180.
G01  Z-.25
X2.0  A270.
G00  Z.1
A360.
G01  Z-.25
X1.0  A270.
G00  Z.1
M99
%
```

INCREMENTAL:

```
%
O1236  (Subprogram to mill channels.)
G91
G01  Z-.35  F15.
X1.0  A90.
G00  Z.35
A90.
G01  Z-.35
X-1.0  A-90.
G00  Z.35
A90.
G01  Z-.35
X1.0  A90.
G00  Z.35
A90.
G01  Z-.35
X-1.0  A-90.
  G00  G90  Z.1
M99
%
```

2.16 Formulas

TAPPING -

STANDARD thread formula:
Revolutions per minute (RPM) divided by threads per inch (TPI) = Feed rate in inches per minute
RPM/TPI = F

METRIC thread formula:
Pitch (P) multiplied by .03937 = ____ multiplied by RPM = Feed rate in inches per minute
(P x .03937) x RPM = F

SPEED AND FEEDS -

S.F.M. (Surface Feet per Minute):
.262 multiplied by the cutter diameter multiplied by the RPM = SFM
.262 x Cutter Diameter x RPM = SFM

R.P.M. (Revolutions Per Minute):
3.82 multiplied by the recommended SFM divided by the cutter diameter = RPM
(3.82 x SFM) / Cutter Diameter = RPM

I.P.M. (Inch Per Minute):
Feed per tooth multiplied by the number of cutter teeth multiplied by the RPM = Feed rate in inches per minute.
(Feed/tooth x n) x RPM = IPM or F

CUBIC INCH PER MINUTE:

Effective diameter of cut multiplied by the depth of cut multiplied by the inch per minute feed rate = cubic inch per minute.

(E Diameter x d) x IPM = CIPM

II. PROGRAMMING

1. Introduction

The definition of a part program for any CNC consists of movements of the tool and speed changes to the tool RPM. It also contains auxiliary command functions such as tool changes, coolant on or off commands, or external **M** code commands.

Tool movements consist of rapid positioning commands, straight line movement of the tool at a controlled speed, and movement along an arc.

This machine has three (3) linear axes named **X**, **Y**, and **Z**. The X-axis moves the table left and right, the Y-axis moves it to and from the operator, and the **Z** moves the milling head up and down. The machine zero position is where the tool is at the right corner of the mill table farthest away from the front doors. Motion in the X-axis will move the table to the right for negative numbers and to the left for positive numbers. Motion in the Y-axis will move the table away from the operator for negative numbers and toward the operator for positive numbers. Motion in the Z-axis will move the tool down for negative numbers and up for positive numbers.

The optional fourth, or rotary, axis can be programmed for both rapid positioning commands and for feed commands either by itself or in conjunction with the other axes.

In addition to the above, there may be up to five, external, axes that can be programmed for rapid or feed motions, but one axis at a time only.

2. Program Structure

2.1 The Parts of a Program

A CNC part program consists of one or more blocks of commands. When viewing the program, a block is the same as a line of text. Blocks shown on the CRT are always terminated by the " ; " symbol which is called an EOB. Blocks are made up of alphabetical address codes and the " / " symbol. Address codes are always an alphabetical character followed by a numeric value. For instance, the specification of the position to move the X-axis would be a number preceded by the **X** symbol.

The " / " symbol, sometimes called a slash, is used to define an optional block. A block that contains this symbol can be optionally deleted with the BLKDEL button when running a program.

```
%                                          :PROGRAM MUST BEGIN AND END WITH %
O1234 (OP1 SAMPLE MILL PART)               :PROGRAM # AND COMMENT STATEMENT
N1 (TOOL #1 IS A ½ INCH STUB DRILL)        :(******) NOTES TO OPERATOR
N100 G00 X0 Y0 Z.5 G43 H1 M3 S1400 T2      :RAPID TO POS, OFFSET 1, SPIN FWD
N101 G01 Z.2 F30.                          :FEED 30 INCH/MINUTE TO Z DEPTH
N102 G83 G98 Z-.625 R.03 Q.2 F5.           :PECK TO Z-.625 START .03 ABOVE
N103 X1.5 Y1.5                             :DRILL ANOTHER HOLE AT NEW X,Y
N104 Y-1.5                                 :DRILL 3RD HOLE, PECK DEPTH IS .20
N105 X-1.5                                 :DRILL FOURTH HOLE
N106 Y1.5                                  :DRILL FIFTH HOLE
N107 G00 G80 Z.5                           :CANCEL CANNED CYCLE
N108 M06                                   :TOOL CHANGE TO TOOL #2
N2 (T #2 IS 5/8 90 DEG. COUNTERSINK)       :N### ARE LINE NUMBERS
N200 G00 X0 Y0 Z.5 G43 H2 M3 S500          :OFFSET 2, SPINDLE SPEED 500 RPM
N201 G01 Z.2 F30.                          :FEED TO Z AT 30 INCH PER MINUTE
N202 G82 G98 Z-.27 R.0 F5.                 :SPOT DRILL CYCLE, DRILL AT X0 Y0
N203 X1.5 Y1.5                             :SEC HOLE R=START PLANE ABOVE ZERO
N204 Y-1.5                                 :3RD HOLE G98=RETURN TO INIT POINT
N205 X-1.5                                 :FOURTH HOLE
N206 Y1.5                                  :FIFTH HOLE
N207 G00 G80 Z.5                           :RAPID TO Z.5
N208 G28 X0 Y0 Z2.0                        :ZERO RETURN AFTER MOVE TO X0, Y0
N209 M06 T03                               :TOOL CHANGE
N3 (TOOL #3 IS A ½ END MILL)               :N #'S ARE FOR YOUR CONVENIENCE
(SET DIAMETER VALUE TOOL #3)               :COMMENTS ARE IGNORED BY CONTROL
N300 G00 X0 Y0 Z.5 G43 H3 M3 S1000         :G43 = OFFSET Z IN MINUS DIRECTION
N301 G01 Z.2 F30.                          :G01 CAN BE SPECIFIED AS G1
N302 Z-.625 F5.                            :FEED TO DEPTH
N303 G01 G41 X-1.00                        :COMPENSATE CUTTER LEFT OF LINE
N304 G03 I1.0 D1                           :CUT CIRCLE CCW WITH TOOL DIA D1
N305 G00 G40 X00                           :RAPID TO CENTER, G40 CANCELS COMP
N306 G00 Z.5                               :RAPID OUT OF PART
N307 G28                                   :ZERO RETURN, Z GOES FIRST THAN X,Y
N308 M6 T1                                 :PUT TOOL 1 IN SPINDLE
M30                                        :RESET PROGRAM TO BEGINNING
%                                          :END OF TAPE
```

There is no positional requirement for the address codes. They may be placed in any order within the block. The following is a sample program as it would appear on the CRT. The words following the ":" are not part of the program but are put here as further explanation.

This program will drill four holes and mill a two-inch hole in a four-inch square plate with **X** and **Y** zero at the center. The program with comment statements would appear like this.

Please note that each tool has some slight variations. This is done to show the flexibility of the control. For example, to change tools, all that is needed is an M06 even without a G28 in the previous line. Also, a G28 can be specified as G28 X0 Y0 Z0 or simply as G28. A **"T"** command can be put in with the M06 or it can be specified earlier in the program. This gives the maximum compatibility with other controls.

More than one program can be stored in the memory of the CNC. Every program stored has an **Onnnn** address code to define the number of that program. Those numbers are used to identify the program for selection as the main program being run or as a subprogram called from a main program.

2.2 Alphabetical Address Codes

The following is a list of the Address Codes used in programming the CNC.

A Fourth axis rotary motion

The **A** address character is used to specify motion for the optional fourth, **A**, axis. It specifies an angle in degrees for the rotary axis. It is always followed by a signed number and up to three fractional decimal positions. If no decimal point is entered, the last digit is assumed to be 1/1000 degrees. The smallest magnitude is 0.001 degrees, the most negative value is -8380.000 degrees, and the largest number is 8380.000 degrees.

B Fifth axis rotary motion

The **B** address character is used to specify motion for the optional fifth, **B**, axis. It specifies an angle in degrees for the rotary axis. It is always followed by a signed number and up to three fractional decimal positions. If no decimal point is entered, the last digit is assumed to be 1/1000 degrees. The smallest magnitude is 0.001 degrees, the most negative value is -8380.000 degrees, and the largest number is 8380.000 degrees.

C Auxiliary external rotary axis

The **C** address character is used to specify motion for the optional external sixth, **C**, axis. It specifies an angle in degrees for the rotary axis. It is always followed by a signed number and up to three fractional decimal positions. If no decimal point is entered, the last digit is assumed to be 1/1000 degrees. The smallest magnitude is 0.001 degrees, the most negative value is -8380.000 degrees, and the largest number is 8380.000 degrees.

D Tool diameter selection

The **D** address character is used to select the tool diameter or radius used for cutter compensation. The number following must be between 0 and 50. D0 specifies that the tool size is zero and serves to cancel a previous **Dn**. Any other value of **D** selects the numbered entry from the tool diameter/radius list under the Offsets display.

E Not used

F Feed rate

The **F** address character is used to select the feed rate applied to any interpolation functions, including pocket milling and canned cycles. It is either in inches per minute with four fractional positions or mm per minute with three fractional positions.

G Preparatory Functions (G codes)

The **G** address character is used to specify the type of operation to occur in the block containing the **G** code. The **G** is followed by a two or three digit number between 0 and 150. Each **G** code defined in this control is part of a group of **G** codes. The Group 0 codes are non-modal; that is, they specify a function applicable to this block only and do not effect other blocks. The other groups are modal and the specification of one code in the group cancels the previous code applicable from that group. A modal **G** code applies to all subsequent blocks so those blocks do not need to re-specify the same **G** code. More than one **G** code can be placed in a block in order to specify all of the setup conditions for an operation. See Section 3 for a detailed list of **G** codes.

H Tool length offset selection

The **H** address character is used to select the tool length offset entry from the offsets memory. The **H** is followed by a two digit number between 0 and 50. H0 will cause no offset to be used and **Hn** will use the tool length entry **n** from the Offsets display. Note that G49 is the default condition disabling tool length offsets; so you must also select either G43 or G44 for tool offsets to work. The TOOL OFSET MESUR button will enter a value into the offsets to correspond to the use of G43.

I Canned cycle and circular optional data

The **I** address character is used to specify data used for some canned cycles and circular motions. It is either in inches with four fractional positions or mm with three fractional positions. It is followed by a signed number in inches between -838.0000 and 838.0000 for inches or between -8380.000 and 8380.000 for metric.

J Canned cycle and circular optional data

The **J** address character is used to specify data used for some canned cycles and circular motions. It is formatted just like the **I** data.

K Canned cycle and circular optional data

The **K** address character is used to specify data used for some canned cycles and circular motions. It is formatted just like the **I** data.

L Loop count for repeated cycles

The **L** address character is used to specify a repetition count for some canned cycles and auxiliary functions. It is followed by an unsigned number between 0 and 32767.

M M code Miscellaneous Functions

The **M** address character is used to specify an **M** code for a block. These codes are used to control miscellaneous machine functions. Note that only one **M** code is allowed per block of the CNC program and all **M** codes are performed at the end of the block. See Section 9 for a detailed list of **M** codes.

N Number of block

The **N** address character is entirely optional. It can be used to identify or number each block of a program. It is followed by a number between 0 and 99999. The M97 and M98 functions may reference an **N** line number.

O Program number/name

The **O** address character is used to identify a program. It is followed by a number between 0 and 9999. A program saved in memory always has a **Onnnn** identification in the first block; it cannot

be deleted. Altering the **O** in the first block causes the program to be renamed. An **Onnnn** can be placed in other blocks of a program but will have no effect and can be confusing to the reader. A colon (:) may be used in the place of **O**, but is always displayed as "**O**".

P Delay time or program number

The **P** address character is used to enter either a time in seconds or a program number for a subroutine call. If it is used as a time (for a G04 dwell) or a program name (for a M98), the value may be either a positive number without decimal point up to 9999. If it is used as a time, it may be a positive decimal with fraction between 0.001 and 1000.0.

Q Canned cycle optional data

The **Q** address character is used in canned cycles and is always a positive number in inches between 0 and 100.0.

R Canned cycle and circular optional data

The **R** address character is used in canned cycles and circular interpolation. It is either in **Inches** with four fractional positions or **mm** with three fractional positions. It is followed by a signed number in inches between -838.0000 and 838.0000 for inches or between -83800.000 and 8380.000 for metric. It is usually used to define the reference plane for canned cycles.

S Spindle speed command

The **S** address character is used to specify the spindle speed in conjunction with M41 and M42. The **S** is followed by an unsigned number between 1 - 99999. The **S** command does not turn the spindle on or off; it only sets the desired speed. If a gear change is required in order to set the commanded speed, this command will cause a gear change to occur even if the spindle is stopped. If the spindle is running, a gear change operation will occur and the spindle will continue running at the new speed.

T Tool selection code

The **T** address character is used to select the tool for the next tool change. The number following must be a positive number between 1 and the number in Parameter 65. It does not cause the tool change operation to occur. The **Tn** may be placed in the same block that starts the tool change (M6 or M16) or in any previous block.

U Auxiliary external linear axis

The **U** address character is used to specify motion for the optional external linear, **U**, axis. It specifies a position of motion in inches. It is always followed by a signed number and up to four fractional decimal positions. If no decimal point is entered, the last digit is assumed to be 1/10000 inches. The smallest magnitude is 0.0001 inches, the most negative value is -838.0000 inches, and the largest number is 838.0000 inches.

V Auxiliary external linear axis

The **V** address character is used to specify motion for the optional external linear, **V**, axis. It specifies a position of motion in inches. It is always followed by a signed number and up to four fractional decimal positions. If no decimal point is entered, the last digit is assumed to be 1/10000 inches. The smallest magnitude is 0.0001 inches, the most negative value is -838.0000 inches, and the largest number is 838.0000 inches.

W Auxiliary external linear axis

The **W** address character is used to specify motion for the optional external linear, **W**, axis. It specifies a position of motion in inches. It is always followed by a signed number and up to four fractional decimal positions. If no decimal point is entered, the last digit is assumed to be 1/10000 inches. The smallest magnitude is 0.0001 inches, the most negative value is -838.0000 inches, and the largest number is 838.0000 inches.

X Linear X-axis motion

The **X** address character is used to specify motion for the X-axis. It specifies a position or distance along the X-axis. It is either in **inches** with four fractional positions or **mm** with three fractional positions. It is followed by a signed number in inches between -838.0000 and 838.0000 for inches or between -8380.000 and 8380.000 for metric. If no decimal point is entered, the last digit is assumed to be 1/10000 inches or 1/1000 mm.

Y Linear Y-axis motion

The **Y** address character is used to specify motion for the Y-axis. It specifies a position or distance along the Y-axis. It is either in **inches** with four fractional positions or **mm** with three fractional positions. It is followed by a signed number in inches between -838.0000 and 838.0000 for inches or between -8380.000 and 8380.000 for metric. If no decimal point is entered, the last digit is assumed to be 1/10000 inches or 1/1000 mm.

Z Linear Z-axis motion

The **Z** address character is used to specify motion for the Z-axis. It specifies a position or distance along the Z-axis. It is either in **inches** with four fractional positions or **mm** with three fractional positions. It is followed by a signed number in inches between -838.0000 and 838.0000 for inches or between -8380.000 and 8380.000 for metric. If no decimal point is entered, the last digit is assumed to be 1/10000 inches or 1/1000 mm.

3. Preparatory Functions (G Codes)

The following is a **G** codes summary. A " * " indicates the default within each group, if there is one:

Code:	Group:	Function:	Description On Page:
G00	*01	Rapid Motion	51
G01	01	Linear Interpolation Motion	51
G02	01	CW Interpolation Motion	51
G03	01	CCW Interpolation Motion	52
G04	00	Dwell	53
G09	00	Exact Stop	53
G10	00	Programmable Offset Setting	53
G12	00	CW Circular Pock Milling (Yasnac)	54
G13	00	CCW Circular Pock Milling (Yasnac)	54
G17	*02	XY Plane Selection	56
G18	02	ZX Plane Selection	56
G19	02	YZ Plane Selection	56
G20	06	Inch programming selection	88
G21	06	Metric programming selection	88
G28	00	Return To Reference Point	56
G29	00	Set Return Reference Point	56
G31	00	Skip Function	56
G35	00	Automatic Tool Diameter Measurement	57
G36	00	Automatic Work Offset Measurement	57
G37	00	Automatic Tool Length Measurement	57
G40	*07	Cutter Comp Cancel	58
G41	07	Cutter Compensation Left	58
G42	07	Cutter Compensation Right	58
G43	08	Tool Length Compensation +	58
G44	08	Tool Length Compensation -	59
G49	*08	G43/G44 Cancel	59
G50	11	G51 Cancel	59
G51	11	Scaling	59
G52	12	Select work Coordinate G92 System (Yasnac)	63
G52	00	Set Local Coordinate System (Fanuc)	63
G53	00	Non-Modal Machine Coordinate Selection	63
G54	*12	Select Work Coordinate System 1	63
G55	12	Select Work Coordinate System 2	63
G56	12	Select Work Coordinate System 3	63
G57	12	Select Work Coordinate System 4	63
G58	12	Select Work Coordinate System 5	63
G59	12	Select Work Coordinate System 6	63
G60	00	Unidirectional Positioning	63
G61	13	Exact Stop Modal	64
G64	*13	G61 Cancel	64
G65	00	Macro Subroutine Call	112
G68	16	Rotation	60
G69	16	G68 Cancel	63
G70	00	Bolt Hole Circle (Yasnac)	64
G71	00	Bolt Hole Arc (Yasnac)	64
G72	00	Bolt Holes Along an Angle (Yasnac)	64
G73	09	High Speed Peck Drill Canned Cycle	68
G74	09	Reverse Tap Canned Cycle	70
G76	09	Fine Boring Canned Cycle	70
G77	09	Back Bore Canned Cycle	72
G80	*09	Canned Cycle Cancel	73

G81	09	Drill Canned Cycle	73
G82	09	Spot Drill Canned Cycle	74
G83	09	Peck Drill Canned Cycle	75
G84	09	Tapping Canned Cycle	77
G85	09	Boring Canned Cycle	78
G86	09	Bore/Stop Canned Cycle	79
G87	09	Bore/Manual Retract Canned Cycle	80
G88	09	Bore/Dwell Canned Cycle	81
G89	09	Bore Canned Cycle	82
G90	*03	Absolute	83
G91	03	Incremental	83
G92	00	Set Work Coordinates	83
G98	*10	Initial Point Return	83
G99	10	Plane Return	83
G100	00	Disable Mirror Image	83
G101	00	Enable Mirror Image	83
G102	00	Programmable Output To RS-232	84
G103	00	Block Lookahead Limit	129
G110	12	Select Work Coordinate System 7	85
G111	12	Select Work Coordinate System 8	85
G112	12	Select Work Coordinate System 9	85
G113	12	Select Work Coordinate System 10	85
G114	12	Select Work Coordinate System 11	85
G115	12	Select Work Coordinate System 12	85
G116	12	Select Work Coordinate System 13	85
G117	12	Select Work Coordinate System 14	85
G118	12	Select Work Coordinate System 15	85
G119	12	Select Work Coordinate System 16	85
G120	12	Select Work Coordinate System 17	85
G121	12	Select Work Coordinate System 18	85
G122	12	Select Work Coordinate System 19	85
G123	12	Select Work Coordinate System 20	85
G124	12	Select Work Coordinate System 21	85
G125	12	Select Work Coordinate System 22	85
G126	12	Select Work Coordinate System 23	85
G127	12	Select Work Coordinate System 24	85
G128	12	Select Work Coordinate System 25	85
G129	12	Select Work Coordinate System 26	85
G136	00	Automatic Work Offset Center Measurement	57
G150	00	General Purpose Pocket Milling	85

In a control configured with a fifth axis, all G codes that have an option for an **A** axis motion command can also simultaneously command fifth axis, **B**, motion. Of course, since address **B** is modal, it can be entered on any line. The following G codes are users of address **B**:

G00	G03	G29	G73	G77	G83	G86	G89	G101
G01	G10	G31	G74	G81	G84	G87	G92	G102
G02	G28	G36	G76	G82	G85	G88	G100	G136

Each **G** code defined in this control is part of a group of **G** codes. The Group 0 codes are non-modal; that is, they specify a function applicable to this block only and do not affect other blocks. The other groups are modal and the specification of one code in the group cancels the previous code applicable from that group. A modal **G** code applies to all subsequent blocks so those blocks do not need to re-specify the same **G** code.

There is also one case where the Group 01 **G** codes will cancel the Group 9 (canned cycles) codes. If a canned cycle is active (G73 thru G89), the use of G00 or G01 will cancel the canned cycle.

3.1 Rapid Position Commands

G00 Rapid Motion Positioning Group 01

X Optional X-axis motion command
Y Optional Y-axis motion command
Z Optional Z-axis motion command
A Optional A axis motion command

This **G** code is used to cause a rapid traverse of the three or four axes of the machine. The auxiliary axes **B**, **C**, **U**, **V**, and **W** can also be moved with a G00. This **G** code is modal so that a previous block with G00 causes all following blocks to be rapid motions until another Group 01 code is specified. The rapid traverse rate is dependent on the maximum speed possible for each axis independently as modified by the RAPID override operator buttons.

Generally, rapid motions will not be in straight lines. All of the axes specified are moved at the same time but will not necessarily complete their motions at the same time. The block will wait until all motions are complete. Only the axes specified are moved and the incremental or absolute modal conditions (G90 or G91) will change how those values are interpreted. Parameter 57 can change how closely the machine waits for a precise stop before and after a rapid move.

3.2 Interpolation Commands

G01 Linear Interpolation Motion Group 01

F Feed rate in inches (mm) per minute
X Optional X-axis motion command
Y Optional Y-axis motion command
Z Optional Z-axis motion command
A Optional A axis motion command

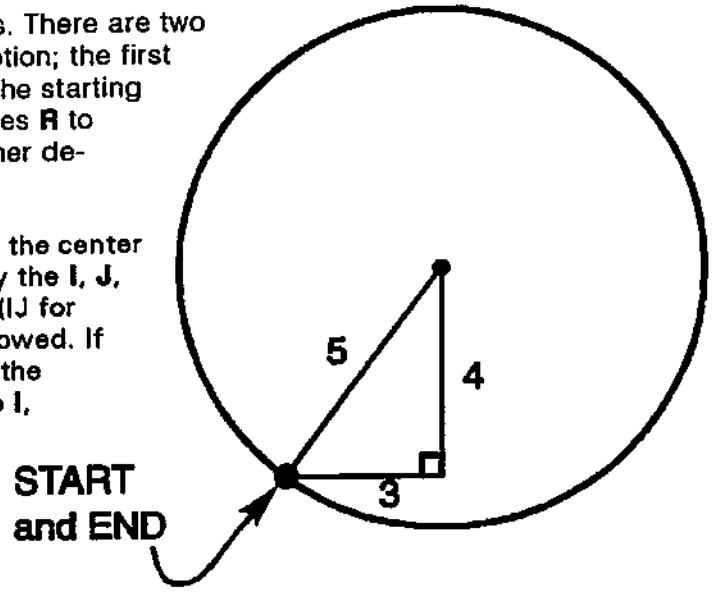
This **G** code provides for straight line (linear) motion from point to point. Motion can occur in 1, 2 or 3 dimensions. All axes will start and finish motion at the same time. The rotary axis may also be commanded and this will provide a helical motion. The speeds of all axes are controlled so that the feed rate specified is achieved along the actual path. Rotary axis speed is dependent on the rotary axis diameter setting (Setting 34) and will provide a helical motion. The **F** command is modal and may be specified in a previous block. Only the axes specified are moved and the incremental or absolute modal conditions (G90 or G91) will change how those values are interpreted. The auxiliary axes **B**, **C**, **U**, **V**, and **W** can also be moved with a G01 but only one axis is moved at a time.

G02 CW Circular Interpolation Motion Group 01

F Feed rate in inches (mm) per minute
I Optional distance along X-axis to center of circle
J Optional distance along Y-axis to center of circle
K Optional distance along Z-axis to center of circle
R Optional radius of circle
X Optional X-axis motion command
Y Optional Y-axis motion command
Z Optional Z-axis motion command
A Optional A axis motion command

This **G** code is used to specify a clockwise circular motion of two of the linear axes. Circular motion is possible in any two of **X**, **Y**, and **Z** axes as selected by G17, G18, and G19. The **X**, **Y**, and **Z** are used to specify the end point of the motion that can use either absolute (G90) or incremental (G91) motion. If any of the **X**, **Y**, or **Z** for the selected plane is not specified, the endpoint of the arc

is the same as the starting point for that axis. There are two ways to specify the center of the circular motion; the first uses **I**, **J**, or **K** to specify the distance from the starting point to the center of the arc; the second uses **R** to specify the radius of the arc. These are further described below:

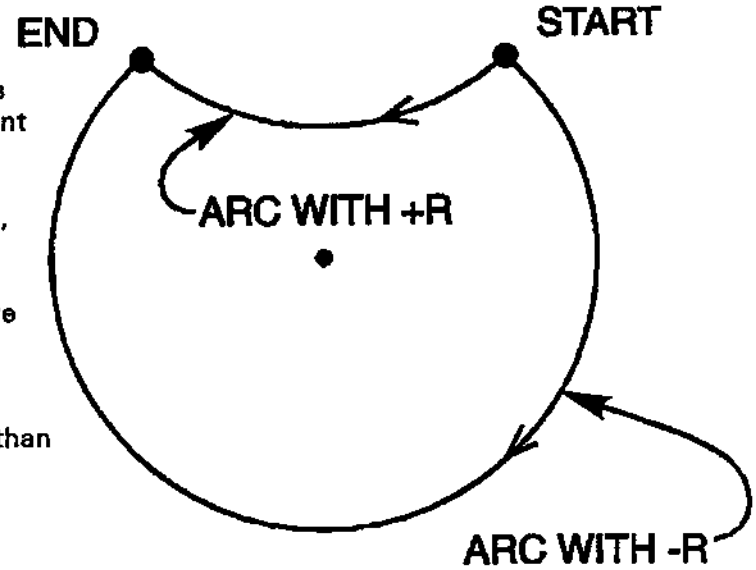


I, J, K: When **I**, **J**, or **K** are used to specify the center of the arc, **R** may not be used. Only the **I**, **J**, or **K** specific to the selected plane (IJ for G17, IK for G18, JK for G19) are allowed. If only one of the **I**, **J**, **K** is specified, the others are assumed to be zero. The **I**, **J**, or **K** is the signed distance from the starting point to the center of the circle. Small errors in these values are tolerated up to 0.0010 inches. Use of **I**, **J**, or **K** is the only way to cut a complete 360 degree arc; in this case, the starting point is the same as the ending point and no **X**, **Y**, or **Z** is needed.

To cut a complete circle of 360 degrees (360°), you do not need to specify an ending point **X**, **Y**, or **Z**; just program **I**, **J**, or **K** to define the center of the circle. The following line will cut a complete circle:

```
G02 I3.0 J4.0        (Assumes G17; XY plane)
```

In cases where you are cutting less than a complete circle, it is much easier to use **R** instead of **I**, **J**, **K**.



R: When **R** is used to specify the center of the circle, a complete 360 degree arc is not possible. **X**, **Y**, or **Z** is required to specify an endpoint different from the starting point. **R** is the distance from the starting point to the center of the circle. With a positive **R**, the control will generate a path of 180 degrees or less; to generate an angle of over 180 degrees, specify a negative **R**. Small errors in this value are tolerated up to 0.0010 inches.

The following line will cut an arc less than 180 degrees (180°):

```
G01 X3.0 Y4.0
G02 X-3.0 R5.0
```

and the following line will cut an arc of more than 180 degrees (180°):

```
G01 X3.0 Y4.0
G02 X-3.0 R-5.0
```

G03 CCW Circular Interpolation Motion

Group 01

G03 will generate counterclockwise circular motion but is otherwise the same as G02.

HELICAL

A helical motion is possible with G02 or G03 by programming the linear axis that is not in the selected plane. This third axis will be interpolated along the specified axis in a linear manner while the other two axes will be moved in the circular motion. The speed of each axis will be controlled so that the helical rate matches the programmed feed rate.

The length of the third axis motion may not be greater than the length of the motion of the two axes for the circular motion. This means that for a complete revolution around a one-inch diameter, the circumference will be 3.1416 and the third axis motion may not be more than 3.1416 inches.

3.3 Miscellaneous G Codes

G04 Dwell Group 00

P The dwell time in seconds or milliseconds

G04 is used to cause a delay or dwell in the program. The block containing G04 will delay for the time specified in the **P** code. If the **P** has no fraction part, the delay is in milliseconds (0.001 seconds); otherwise the delay is in seconds.

G09 Exact Stop Group 00

The G09 code is used to specify exact stop. It is not modal and does not affect the following blocks. Rapid and interpolated moves will decelerate to an exact stop before another block is processed. In exact stop, moves will take a longer time and continuous cutter motion will not occur. This may cause deeper cutting where the tool stops.

3.4 Programmable Offset Setting

G10 Programmable Setting of Tool Offsets Group 00

L Selection of length, length wear, diameter, diameter wear, or work coordinates.
P Selection of offset number.
R Offset value or increment for length and diameter.
X Optional X-axis zero location.
Y Optional Y-axis zero location.
Z Optional Z-axis zero location.
A Optional A-axis zero location.

G10 can be used to change the tool length and work offsets from inside of a program. G10 is non-modal and thus does not affect offset values for subsequent blocks. The following codes are used for selection of offsets:

L2	Work coordinate origin for G52 and G54-G59
L10	Length offset amount (for **H** code)
L1 or L11	Tool wear offset amount (for **H** code)
L12	Diameter offset amount (for **D** code)
L13	Diameter wear offset amount (for **D** code)
L20	Auxiliary work coordinate origin for G110-G129

The **P** code is used to index the appropriate offsets.

P1-P50	Used to reference **D** or **H** code offsets,	L10-L13
P0	G52 references work coordinate	L2

P1-P6	G54-G59 references work coordinates	L2
P1-P20	G110-G129 references auxiliary coordinates	L20

The **R**, **X**, **Y**, **Z**, and **A** codes are signed numbers with fractions in inches (or MM). The **R**, **X**, **Y**, **Z**, and **A** values are absolute or incremental, depending on the current G90/G91 modal value.
G10 Examples:

```
G10 L2  P1  G91 X6.0        {Move coordinate G54 6.0 to the right.};
G10 L20 P2  G90 X10.  Y8.   {Set work coordinate G111 to X10.0 ,Y8.0};
G10 L10 G90 P5  R2.5        {Set offset for Tool #5 to 2.5.};
G10 L12 G90 P5  R.375       {Set diameter for Tool #5 to 3/8ths.};
```

3.5 Circular Pocket Milling

There are two **G** codes that will provide for pocket milling of a circular shape. They are different only in which direction of rotation is used.

G12 Circular Pocket Milling Clockwise

Group 00

D	Tool Radius Or Diameter Selection
I	Radius Of First Circle (Or Finish If No K)
K	Radius Of Finished Circle (If Specified)
L	Loop count for repeating deeper cuts
Q	Radius Increment (Must Be Used With K)
F	Feed Rate in inches (mm) per minute
Z	Z depth of cut or increment

This **G** Code implies the use of G42. The tool must be positioned at the center of the circle either in a previous block or in this block using **X** and **Y**. The cut is performed entirely with circular motions of varying radius. G12 belongs to Group zero and thus is non-modal. If G91 (incremental) is specified and an **L** count is included, the **Z** increment is repeated **L** times at the **F** feed rate. If no **K** is specified, the center of the cut is removed completely.

G13 Circular Pocket Milling Counterclockwise

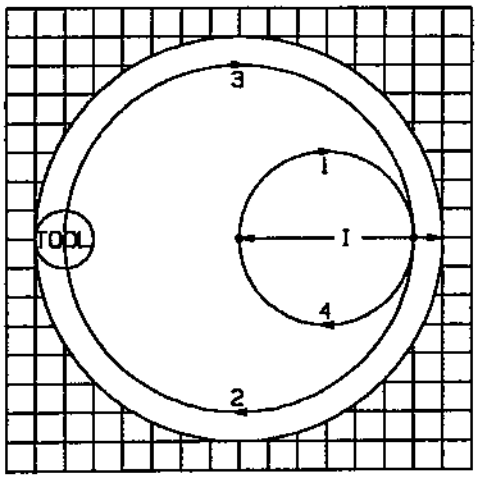
Group 00

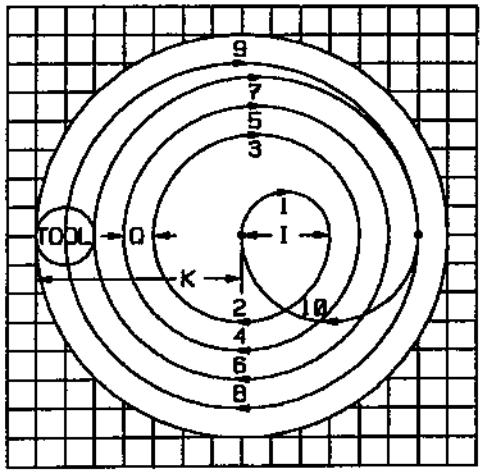
This **G** Code implies the use of G41 and is otherwise similar to G12. G13 belongs to Group zero and thus is non-modal.

```
%
O0100 (SAMPLE G12 AND G13)
(OFFSET D01 SET TO APPROX. TOOL SIZE)
(TOOL MUST BE MORE THAN 0.3 IN DIAM.)
G54 G00 G90 Z-1. X0. Y0.
S2000 M03
G12 I1.5 F10. Z-1.2 D01
G28
G55 Z-1. X0. Y0.
G12 I0.3 K1.5 Q0.3 F10. Z-1.2 D01
G28
G56 Z-1. X0. Y0.
G13 I1.5 F10. Z-1.2 D01
G28
G57 Z-1. X0. Y0.
G13 I0.3 K1.5 Q0.3 F10. Z-1.2 D01
G28 M30
%
```

Circular Pocket Milling

G12

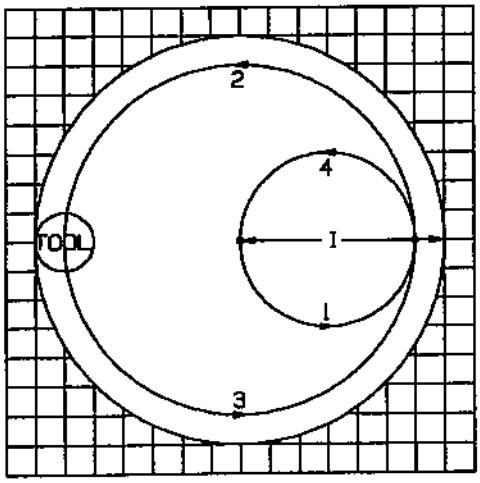


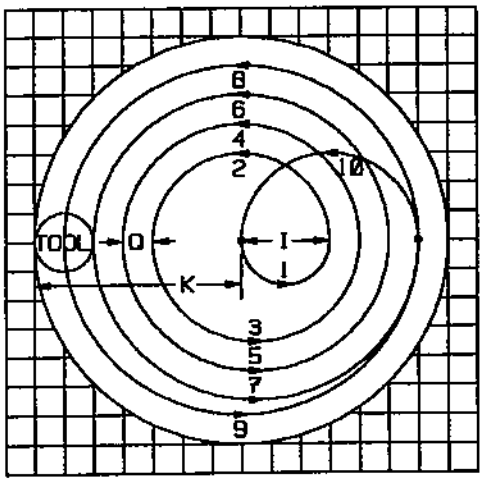


I ONLY

I, K, AND Q

G13





3.6 Circular Plane Selection

G17 XY Plane Selection

Group 02

The G17 code is used to select the XY plane for circular motion. It is modal and applies to all following circular motions until another Group 02 is found.

G18 ZX Plane Selection

Group 02

The G18 code is used to select the ZX plane for circular motion. It is modal and applies to all following circular motions until another Group 02 is found.

G19 YZ Plane Selection

Group 02

The G19 code is used to select the YZ plane for circular motion. It is modal and applies to all following circular motions until another Group 02 is found.

3.7 Reference Point Definition and Return

G28 Return To Reference Point

Group 00

The G28 code is used to return to the machine zero position on all axes. If an **X**, **Y**, **Z**, or **A** code is specified on the same block, only those axes will be moved and they will be moved to the specified positions in the current coordinate system and then they will be moved to machine zero. The intermediate position, if specified, is saved for use in the G29. If no **X**, **Y**, **Z**, or **A** is specified, all axes will be moved directly to machine zero. Any auxiliary axes (**B**, **C**,...) are returned to home after the **X**, **Y**, **Z**, and **A** axes. G28 will also cancel tool length offsets.

G29 Set Return Reference Point

Group 00

The G29 code is used to move the axes to a position via a previously-set reference point. The reference is defined with the G29. This command is normally given with the axes positioned at machine zero. The axes that are selected in this block are moved first to the intermediate reference point and then they are moved to the **X**, **Y**, **Z**, or **A** specified. The positions are interpreted in the current coordinate system.

3.8 Skip Function (G31)

G31 Skip Function

Group 00

F	Feed rate in inches (mm) per minute
X	Optional X-axis motion command
Y	Optional Y-axis motion command
Z	Optional Z-axis motion command
A	Optional A-axis motion command

The skip function is a non-modal operation that causes a linear move to the specified **X**, **Y**, **Z**, and **A** position. It applies only to the block in which G31 is specified. A feed rate must be defined previously or in this block. The specified move is started and continues until the end point or the skip signal. The skip signal is a discrete input that usually indicates that the end of travel has been

reached; this is usually a probe. Cutter compensation may not be active during a skip function. M78 or M79 may be used to test if the skip signal was received.

An M75 can be used to mark the probed point as the reference point for G35 or G136.

3.9 Automatic Tool Measurement (G35, G37)

G35 Automatic Tool Diameter Measurement Group 00
G37 Automatic Tool Length Measurement Group 00

F Feed rate in inches (mm) per minute
D Tool diameter offset number (G35)
H Tool offset number (G37)
Z Required Z-axis offset

The automatic tool length measurement operation (G37) is a non-modal operation that causes a linear move of the Z-axis until the skip signal is received or the end of **Z** travel limits. A nonzero **H** code must be active, G43 or G44 must be active, a **Z** value must be specified, and a feed rate must be defined. No **X**, **Y**, or **A** code is allowed. When the move is terminated, the specified **Z** and the final **Z** positions are used to set the specified (**Hnn**) tool offset. The active coordinate system is taken into account.

The coordinate system (G54..G59, G110..G129) and tool length offset (H01..H50) may be selected in this block or in a previous block. The end point of the **Z** move is controlled only by the maximum travel limits defined for the machine.

The resulting tool offset value is such that a subsequent move to the **Z** value specified in the G37 will move the tool to the position where the skip signal was sensed. The skip signal is a discrete input that usually indicates that the end of travel has been reached; this is sometimes a probe. Cutter compensation may not be active during a skip function. M78 or M79 may be used to test if the skip signal was received. The resulting tool offset is the offset between the work zero and the point where the probe is touched.

The automatic tool diameter measurement function (G35) is used to set the tool diameter (or radius) using two different probe passes; one on each side of the tool. The first point is set with a G31 block using an M75 and the second point is set with the G35 block. The distance between these two points is set into the **Dnn** value active. A nonzero **D** code must be selected. Setting 63 is used to reduce this measurement by the width of the tool probe.

3.10 Automatic Work Offset Measurement (G36,G136)

G36 Automatic Work Offset Measurement Group 00
G136 Automatic Work Offset Center Measurement Group 00

F Feed rate in inches (mm) per minute
I Optional offset distance along X-axis
J Optional offset distance along Y-axis
K Optional offset distance along Z-axis
X Optional X-axis motion command
Y Optional Y-axis motion command
Z Optional Z-axis motion command
A Optional A-axis motion command

The automatic work offset measurement operation is a non-modal operation that causes a linear move of the **X**, **Y**, **Z**, and **A** axes until the skip signal is received or the end of the programmed motion. The **X**, **Y**, **Z**, and **A** axes are moved to the programmed position in a linear move but will stop early if the skip signal is received. Tool offsets must not be active when this function is performed. M78 or M79 may be used to test if the skip signal was received. The currently active work coordinate system is set for each axis that is programmed. The point where the skip signal is received becomes the work zero position. The work coordinate system may be selected in this block or in a previous block.

The points probed are offset by the values set into Settings 59 through 62.

A G36 will set the work coordinates to the point where the probe is hit. The G136 will set the work coordinates to a point at the center of a line between the probed point and the point set with M75. This allows the center of a part to be found using two separated probed points.

Note that the **X**, **Y**, **Z**, or **A** programmed into this block are interpreted in the coordinate system that is about to be set. Thus, the end point of the move will be interpreted in the old work coordinate value. For this reason, it is easier to program these moves as incremental (G91).

If an **I**, **J**, or **K** is specified, the appropriate axis work offset is shifted by the amount in the **I**, **J**, or **K**. This allows the work offset to be shifted some distance away from where the probe actually hits.

3.11 Cutter Compensation

G40 Cutter Comp Cancel — Group 07

G40 will cancel the G41 or G42 cutter compensation. Programming a D00 will also cancel cutter compensation.

G41 Cutter Compensation Left — Group 07

G41 will select cutter compensation left; that is the tool is moved to the left of the programmed path to compensate for the size of the tool. A **Dnn** must also be programmed to select the correct tool size from compensation memory. If compensation memory contains a negative value for cutter size, cutter compensation will operate as though G42 was specified.

G42 Cutter Compensation Right — Group 07

G42 will select cutter compensation right; that is the tool is moved to the right of the programmed path to compensate for the size of the tool. A **Dnn** must also be programmed to select the correct tool size from compensation memory. If compensation memory contains a negative value for cutter size, cutter compensation will operate as though G41 was specified.

See Section 10 for a complete description of cutter compensation operation.

3.12 Tool Length Compensation

G43 Tool Length Compensation + (plus) — Group 08

This code selects tool length compensation in a positive direction. That is; the tool length offsets are added to the commanded axis positions. A nonzero **Hnn** must be programmed to select the correct entry from offsets memory. The automatically entered offsets using the TOOL OFFSET MESUR key assume that G43 is being used.

G44 Tool Length Compensation - (minus) Group 08

This code selects tool length compensation in a negative direction. That is; the tool length offsets are subtracted from the commanded axis positions. A nonzero **Hnn** must be programmed to select the correct entry from offsets memory.

G49 G43/G44 Cancel Group 08

This **G** code cancels tool length compensation. Putting in a H0 will also cancel tool length compensation. G28, M30, and RESET will also cancel tool length compensation.

3.13 Coordinate Rotation and Scaling

G50 Cancel Scaling Group 11

G code 50 cancels scaling on all axes. Any axis scaled by a previous G51 command is no longer in effect.

G51 Scaling Group 11

Programmable scaling can be accomplished with G51. The format is as follows:

G51 [X...] [Y...] [Z...] [P...]

where **X** = optional center of scaling for the X axis.
Y = optional center of scaling for the Y axis.
Z = optional center of scaling for the Z axis.
P = optional scaling factor for all axes. Three-place decimal .001 to 8383.000.

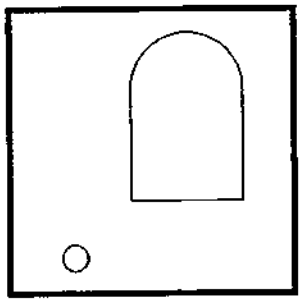
When scaling is invoked, all subsequent **X**, **Y**, **Z**, **I**, **J**, **K**, or **R** values pertaining to machine motion are multiplied by a scaling factor and are offset relative to a scaling center.

G51 is modal and modifies appropriate positional values in the blocks following the G51 command. It does not change or modify values in the block from which it is called. Axes X, Y, and Z are all scaled when the P code is used. If the P code is not used, the scaling factor currently in Setting 71 is used. The default scaling factor in Setting 71 is 1.0. A scaling factor of 1.0 means that no scaling is done. That is, all values are multiplied by 1.0 before being interpreted by the control.

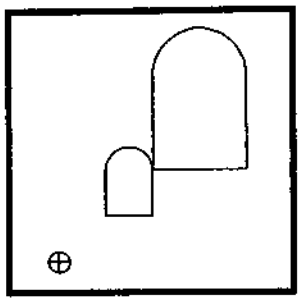
A scaling center is always used by the control in determining the scaled position. If any scaling center is not specified in the G51 command block, then the current work coordinate position is used as the scaling center.

The following programs illustrate how scaling is performed when different scaling centers are used. All three examples call subroutine O0001 which follows.

```
O0001 (GOTHIC WINDOW) ;
F20. S500 ;
G00 X1. Y1. ;
G01 X2. ;
Y2. ;
G03 X1. R0.5        O = Work coordinate origin
G01 Y1. ;           + = Center of scaling
G00 X0 Y0 ;
M99 ;
```

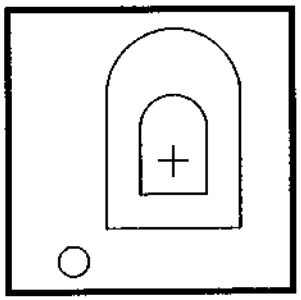


The first example illustrates how the control uses the current work coordinate location as a scaling center. Here, it is X0 Y0 Z0.



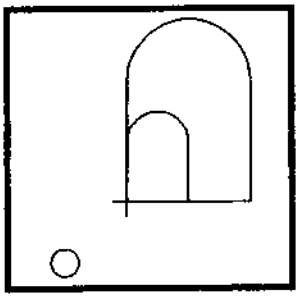
```
O0010 ;
G59 ;
G00 G90 X0 Y0 Z0 ;
M98 P1 ;
G51 P2. (scaling center is X0 Y0 Z0) ;
M98 P1 ;
M30 ;
```

The next example specifies the center of the window as the scaling center.



```
O0011 ;
G59 ;
G00 G90 X0 Y0 Z0 ;
M98 P1 ;
G51 X1.5 Y1.5 P2. ;
M98 P1 ;
M30 ;
```

The last example illustrates how scaling can be placed at the edge of tool paths as if the part was being set against locating pins.



```
O0012 ;
G59 ;
G00 G90 X0 Y0 Z0 ;
M98 P1 ;
G51 X1.0 Y1.0 P2. ;
M98 P1 ;
M30 ;
```

If macros are enabled, G65 arguments are not affected.

Tool offsets and cutter compensation values are not affected by scaling.

The stored program is not changed by G51, so that program lines displayed by the control will not reflect actual machine positions. Position displays WILL reflect the proper scaled values.

Scaling does not affect canned cycle Z axis movements such as clearance planes and incremental values.

The final results of scaling are rounded to the lowest fractional value of the variable being scaled.

G68 Rotation

Group 16

Programmable rotation of a can be accomplished with the G68 command. The format is as follows.

[G17 | G18 | G19] G68 [a...] [b...] [R...] ;

where

G17,G18,G19	optional plane of rotation, default is current.
a	optional first axis rotation center in plane.
b	optional second axis rotation center in plane.
R	optional angle of rotation specified in degrees. Three-place decimal -360.000 to 360.000.

On the previous page, '**a**' and '**b**' correspond to the axes of the current rotation plane. If G17 is the current rotation plane, then '**a**' is X and '**b**' is Y.

When rotation is invoked, all subsequent X, Y, Z, I, J, and K values are rotated through a specified rotation angle **R** using a center of rotation.

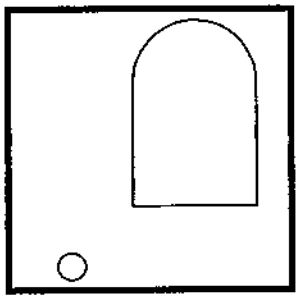
G68 is modal and modifies appropriate positional values in the blocks following the G68 command. Values in the block containing G68 are not rotated. For subsequent blocks, only the values in the plane of rotation are rotated. Thus, if G17 is the current plane of rotation, only X and Y values are affected.

For a positive angle, the rotation is counterclockwise. If the angle of rotation - the R code - is not specified in the G68 command block, then the angle of rotation is taken from Setting 72. The default rotation angle in Setting 72 is 0.0 degrees.

A center of rotation is always used by the control to determine the positional values passed to the control after rotation. If any axis' center of rotation is not specified, then the current location of the work coordinate is used as the center of rotation.

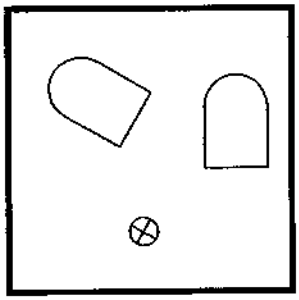
In G90 mode (absolute), the rotation angle takes on the value specified in **R**. When Setting 73 (G68 INCREMENTAL R) is set to ON, then the rotational value can be incremented on each call to G68. In G91 mode (Incremental), the rotation angle is incremented by the value in **R**. Each G68 command block, when in G91 mode, will increment the rotation angle by the value specified in **R**. Angles are modulo 360, so that when an angle is incremented past 360 degrees, the angle will become an equivalent value between 0 and 360 degrees. The rotational angle is set to zero upon cycle start, or it can be set explicitly by using a G68 block in the G90 mode.

The following examples illustrate rotation using G68.



```
O0001 (GOTHIC WINDOW) ;
F20. S500 ;
G00 X1. Y1. ;
G01 X2. ;
Y2. ;
G03 X1. R0.5 ;      O = Work coordinate origin
G01 Y1. ;           + = Center of rotation
M99 ;
```

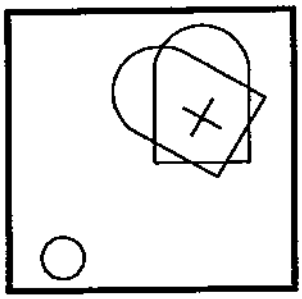
The first example illustrates how the control uses the current work coordinate location as a rotation center. Here, it is X0 Y0 Z0.



```
O0002 ;
G59 ;
G00 G90 X0 Y0 Z0 ;
M98 P1 ;
G90 G00 X0 Y0 ;
G68 R60. ;
M98 P1 ;
G69 G90 G00 X0 Y0 ;
M30 ;
```

The next example specifies the center of the window as the rotation center.

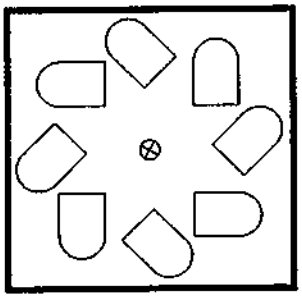
```
O0003 ;
G59 ;
G00 G90 X0 Y0 Z0 ;
M98 P1 ;
G00 G90 X0 Y0 Z0 ;
G68 X1.5 Y1.5 R60. ;
M98 P1 ;
G69 G90 G00 X0 Y0 ;
M30 ;
```



This example shows how the G91 mode can be used to rotate patterns about a center. This is often useful for making parts that are symmetric about a regular polygon.

```
O0004 ;
G59 ;
G00 G90 X0 Y0 Z0 ;
M98 P10 L8 (SUBROUTINE O0010) ;
M30 ;

O0010 ;
G91 G68 R45. ;
G90 M98 P1 ;
G69 ;
G90 G00 X0 Y0 ;
M99 ;
```



Do not change the plane of rotation while G68 is in effect.

ROTATION WITH SCALING

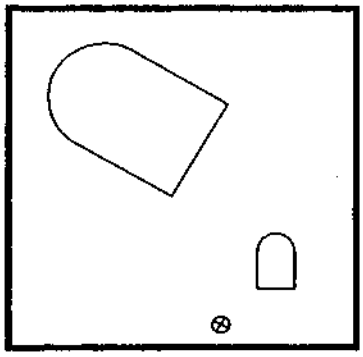
If scaling and rotation is used simultaneously, it is recommended that scaling is turned on prior to rotation, and that separate blocks be used. Use the following template when doing this.

```
G51 ..... (SCALING) ;
...
G68 ..... (ROTATION) ;
.
. program.
.
G69 ..... (ROTATION OFF) ;
...
G50 ..... (SCALING OFF) ;
```

When rotating after scaling, any center specified as the center of rotation will be scaled. Any angle specified in the G68 block is NOT scaled. The control applies scaling and then rotation to any block with motion commands.

Below is an example of a program that has been scaled and rotated.

```
O0004 ;
G59 ;
G00 G90 X0 Y0 Z0 ;
M98 P1 ;
G90 G00 X0 Y0 ;
G51 P3.0 ;
G68 R60. ;
M98 P1 ;
G69 G51 G90 G00 X0 Y0 ;
M30 ;
```



ROTATION WITH CUTTER COMPENSATION

Cutter compensation should be turned on after the rotation and scaling commands are issued. Compensation should also be turned off prior to turning rotation or scaling off.

G69 Cancel G68 Rotation Group 16

G code 69 cancels any rotation specified previously.

3.14 Work Coordinate System Selection

Note: The G52 command works differently depending on the value of Setting 33. That setting selects either FANUC style of coordinates or YASNAC style of coordinates. They are both listed here:

G52 Select work coordinate system G92 YASNAC Group 12

This code selects the work coordinate system that can be set with G92 or from the offsets display. YASNAC compatible.

G52 Set Local Coordinate System FANUC Group 00

This code selects the user coordinate system that can be set by G92. It is non-modal. FANUC compatible.

G53 Non-Modal Machine Coordinate Selection Group 00

This code temporarily cancels work coordinates offset and uses the machine coordinate system. It is non-modal; so the next block will revert to whatever conditions were previously selected.

G54-59 Select Coordinate System #1 - #6 Group 12

These codes select one of the six user coordinate systems stored within the offsets memory. All subsequent references to axes' positions will be interpreted in the new coordinate system. Work coordinate system offsets are entered from the Offsets display page.

3.15 More Miscellaneous G Codes

G60 Uni-Directional Positioning Group 00

This **G** code is used to provide positioning always from the plus direction. In older systems it was used to reduce backlash and is not recommended for use with this control. It is provided only for compatibility. It is non-modal so does not effect the following blocks. Setting 35 controls the distance an axis is positioned past the point prior to reversing for an approach in the plus direction.

G61 Exact Stop Modal Group 13

The G61 code is used to specify exact stop. It is modal and thus affects the following blocks. Rapid and interpolated moves will decelerate to an exact stop before another block is processed. In exact stop, moves will take a longer time and continuous cutter motion will not occur.

This may cause deeper cutting where the tool stops.

G64 G61 Cancel (Select normal cutting mode) Group 13

The G64 code is used to cancel exact stop. It is modal and thus affects the following blocks. Rapid and interpolated moves will not decelerate to an exact stop before another block is processed. Rapid blocks will decelerate to within the distance specified in Parameters 101-104 before another block is processed and interpolated motion will not decelerate at all before the next block is processed.

3.16 Bolt Hole Patterns

There are three **G** codes that provide patterns usually used for bolt holes. These are G70, G71, and G72. They are normally used with one of the Group 09 canned cycles (see 3.12 below).

G70 Bolt Hole Circle Group 00

I Radius (Minus Reverses Direction)
J Starting angle (0 to 360.0 degrees CCW from horizontal)
L Number of holes evenly spaced around the circle

This **G** code must be used with one of the canned cycles G73, G74, G76, G77, Or G81-G89. The tool must be positioned at the center of the circle either in a previous block or in the G70 block. G70 belongs to Group zero and thus is non-modal. For a G70 to work correctly, a canned cycle should be active so that at each of the positions, some type of drill or tap cycle is performed.

G71 Bolt Hole Arc Group 00

I Radius
J Starting angle (Degrees CCW from horizontal)
K Angular spacing of holes (+ and -)
L Number of holes

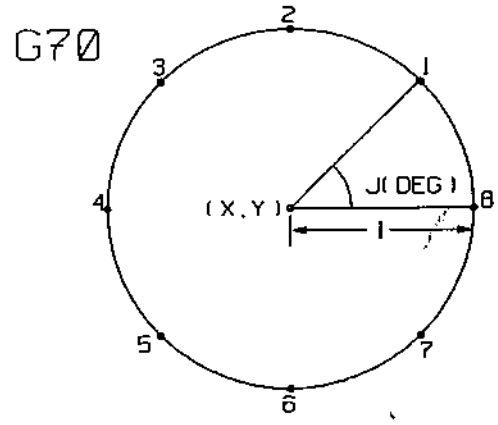
This **G** code is similar to G70 except that it is not limited at one complete circle. G71 belongs to Group zero and thus is non-modal. For a G71 to work correctly, a canned cycle should be active so that at each of the positions, some type of drill or tap cycle is performed.

G72 Bolt Holes Along An Angle Group 00

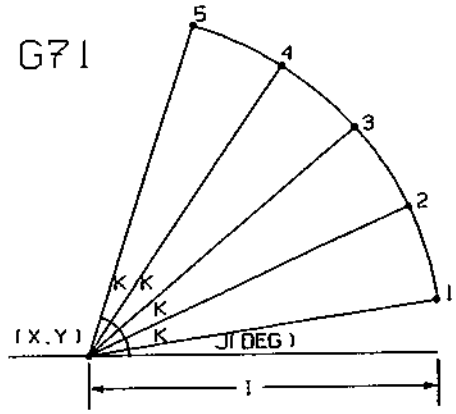
I Distance between holes (Minus will reverse direction)
J Angle of line (Degrees CCW from horizontal)
L Number of holes

This **G** code drills **L** holes in a straight line at the specified angle. It operates similarly to G70 and G71. G72 belongs to Group zero and thus is non-modal. For a G72 to work correctly, a canned cycle should be active so that at each of the positions, some type of drill or tap cycle is performed.

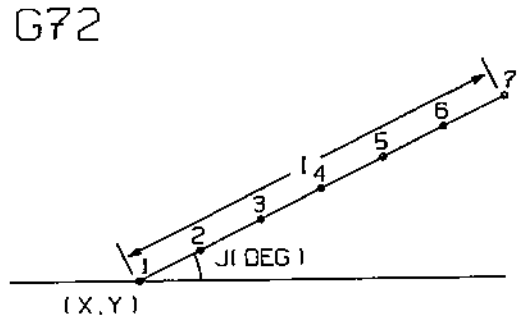
BOLT HOLE CIRCLE



BOLT HOLE ARC



BOLT HOLES ALONG AN ANGLE



3.17 Canned Cycles

A canned cycle is used to simplify programming of a part. Canned cycles are defined for most common Z-axis repetitive operation such as drilling, tapping, and boring. Once selected a canned cycle is active until canceled with G80. When active, the canned cycle is executed every time an **X** or Y-axis motion is programmed. Those Y-Y motions are executed as feed commands (G01) and the canned cycle operation is performed after the X-Y mode.. There are six operations involved in every canned cycle:

1) positioning of **X** and **Y** axes (and optional **A**),
2) rapid traverse to **R** plane,
3) drilling,
4) operation at bottom of hole,
5) retraction to **R** plane,
6) rapid traverse up to initial point.

A canned cycle is presently limited to operations in the Z-axis. That is, only the G17 plane is allowed. This means that the canned cycle will be executed in the Z-axis whenever a new position is selected in the **X** or **Y** axes.

The following is a summary of the canned cycles defined for the VF Series Mill:

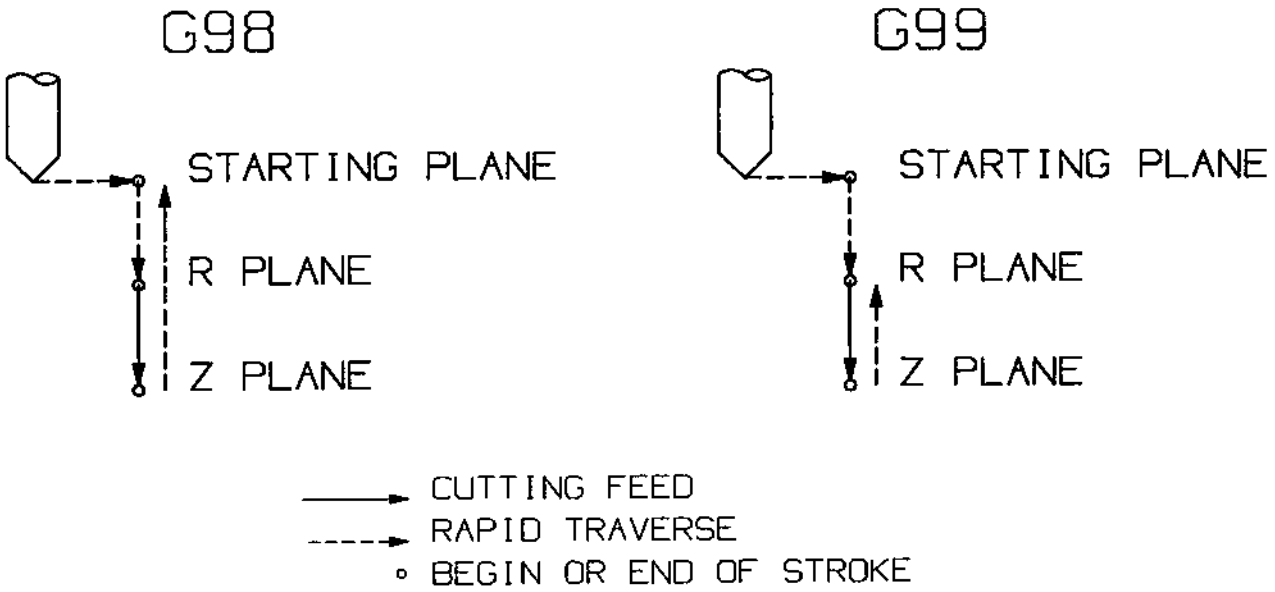
G code	Z Drilling direction	Operation at bottom of hole	Retraction Z direction	Application
G73 feed	intermittent	none	rapid	high speed peck drilling
G74	feed	spindle CW	feed	left hand tapping
G76	feed then stop	orient spindle	rapid	fine boring
G81	feed	none	rapid	spot drilling
G82	feed	dwell	rapid	counter boring
G83	intermittent feed	none	rapid	peck drilling
G84	feed	spindle CCW	feed	tapping cycle
G85	feed	none	feed	boring cycle
G86	feed	spindle stop	rapid	boring cycle
G87	feed	spindle stop	manual/rapid	back boring
G88	feed	dwell, then spindle stop	manual/rapid	boring cycle
G89	feed	dwell	feed	boring cycle

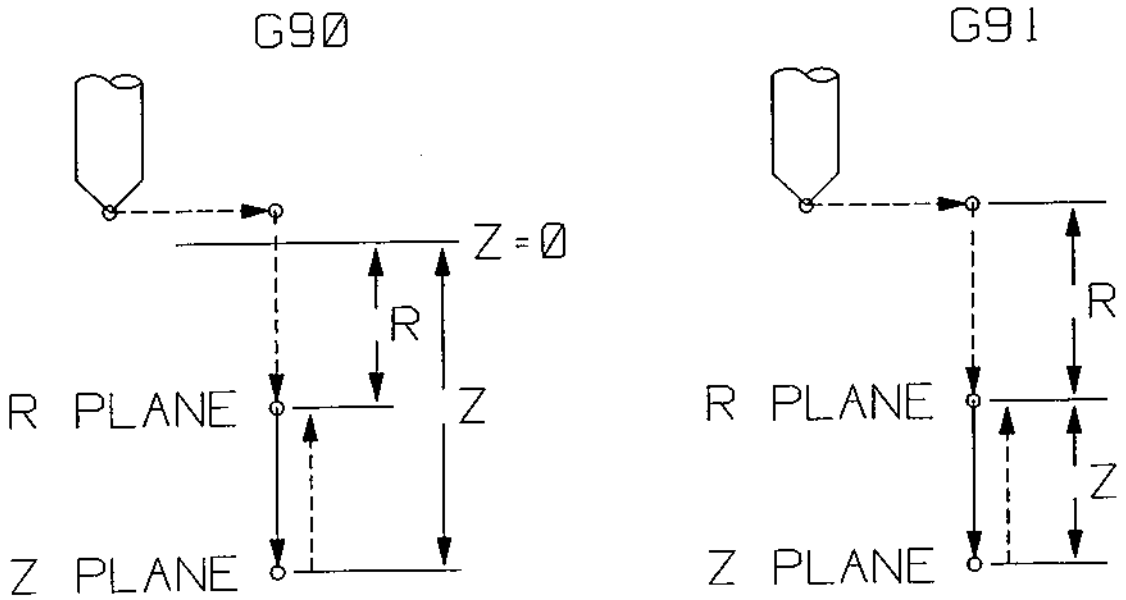
G98 and G99 are modal commands that change the way the canned cycles operate. When G98 is active, the Z-axis will be returned to the same position as at the start of the canned cycle when it completes. When G99 is active, the Z-axis will be returned to the **R** point when the canned cycle completes.

If a canned cycle is defined in a block without an **X** or **Y** motion, there are two common actions taken by other controls; some will execute the canned cycle at that time and some will not. With the VF Series Mill, these two options are selectable from Setting 28. In addition to this, if a canned cycle is defined without an **X** or **Y** and a loop count of 0 (L0), the cycle will not be performed initially. The operation of a canned cycle will vary according to whether incremental (G91) or absolute (G90) is active.

Incremental motion in a canned cycle is often useful as a loop (**L**) count can be used to repeat the operation with an incremental **X** or **Y** move between each cycle.

The positioning of the X-Y axis prior to a canned cycle is normally a rapid move and that move does not exact stop prior to plunging the Z-axis to the **R** depth. This may cause a crash with a close tolerance fixture. Setting 57 can be used to select exact stop of these X-Y moves.





The G80 code is used to cancel a canned cycle. In addition to this, a G00 or G01 code will also cancel any active canned cycle. Once a canned cycle is defined, that operation is performed at every X-Y position subsequently listed in a block. Some of the canned cycle numerical values can also be changed after the canned cycle is defined. The most important of these are the **R** plane value and the **Z** depth value.

If these are listed in a block with an X-Y, the X-Y move is done and all subsequent canned cycles are performed with the new **R** or **Z** value.

Changes to the G98/G99 selection can also be made after the canned cycle is active. If changed, the new G98/G99 value will change all subsequent canned cycle.

G73 High Speed Peck Drilling Canned Cycle

Group 09

F	Feed Rate in inches (mm) per minute
I	Optional size of first cutting depth
J	Optional amount to reduce cutting depth each pass
K	Optional minimum depth of cut
L	Number of repeats
Q	The cut-in value, always incremental
R	Position of the R plane
X	Optional X-axis motion command
Y	Optional Y-axis motion command
Z	Position of bottom of hole

This **G** code is modal in that it activates the canned cycle until it is canceled or another canned cycle is selected. Once activated, every motion of **X** or **Y** will cause this canned cycle to be executed. This cycle is a high speed peck cycle where the retract distance is set by Setting 22.

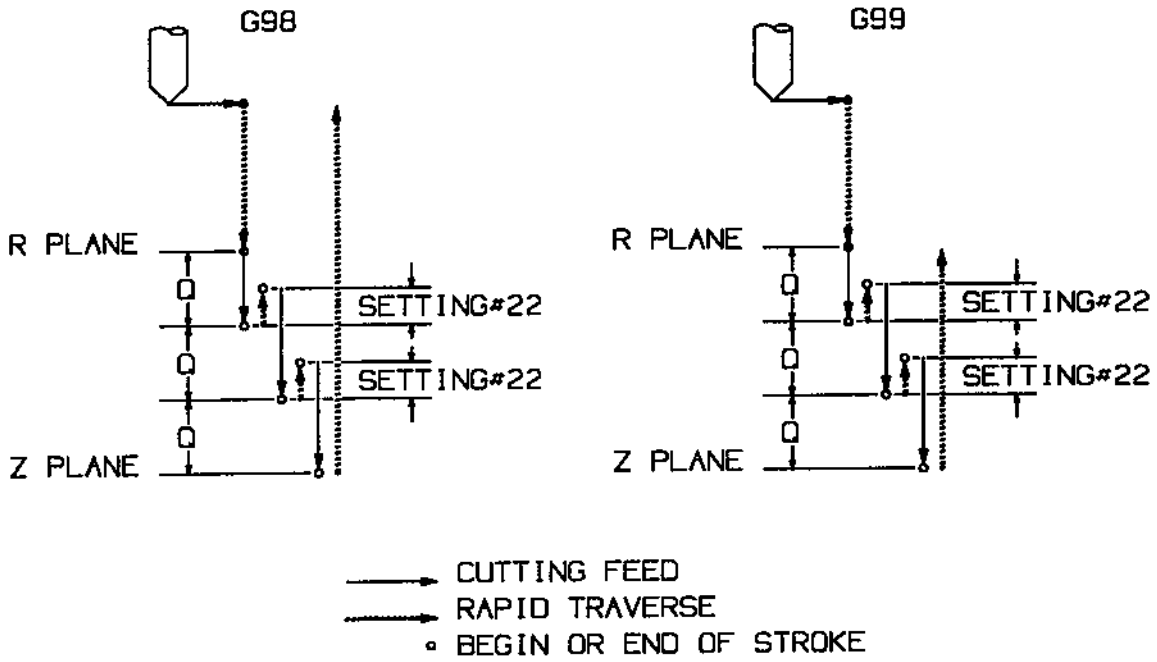
If **I**, **J**, and **K** are specified, a different operating mode is selected. The first pass will cut in by **I**, each succeeding cut will be reduced by amount **J**, and the minimum cutting depth is **K**.

If **K** and **Q** are both specified, a different operating mode is selected for this canned cycle. In this mode, the tool is returned to the **R** plane after a number of passes totals up to the **K** amount. This allows much faster drilling than G83 but still returns to the **R** plane occasionally to clear chips.

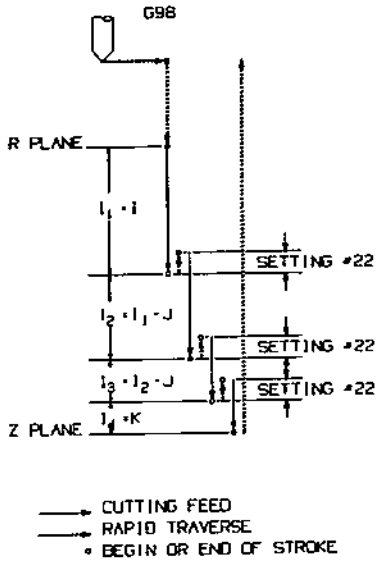
I, **J**, **K**, and **Q** are always positive numbers.

Setting 52 also changes the way G73 works when it returns to the **R** plane. Most programmers set the **R** plane well above the cut to ensure that the chip clear motion actually allows the chips to get out of the hole but this causes a wasted motion when first drilling through this "empty" space. If Setting 52 is set to the distance required to clear chips, the **R** plane can be put much closer to the part being drilled. When the clear move to **R** occurs, the **Z** will be moved above **R** by this setting.

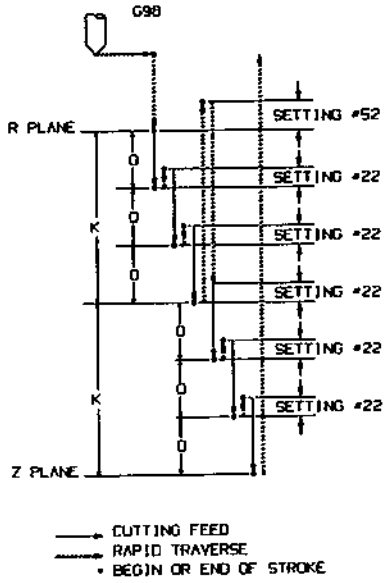
G73 PECK DRILLING CANNED CYCLE



G73 PECK DRILLING CANNED CYCLE
WITH I, J, AND K OPTIONS



G73 PECK DRILLING CANNED CYCLE
WITH K AND Q OPTIONS



G74 Reverse Tap Canned Cycle

Group 09

F	Feed Rate in inches (mm) per minute
L	Number of repeats
R	Position of the R plane
X	Optional X-axis motion command
Y	Optional Y-axis motion command
Z	Position of bottom of hole

This **G** code is modal in that it activates the canned cycle until it is canceled or another canned cycle is selected. Once activated, every motion of **X** or **Y** will cause this canned cycle to be executed. Note that operation of this cycle is different if the rigid tapping option is installed and selected (See Section 7.2). When rigid tapping is used, the ratio between the feed rate and spindle speed must be precisely the thread pitch being cut.

You do not need to start the spindle CCW before this canned cycle. The control does this automatically.

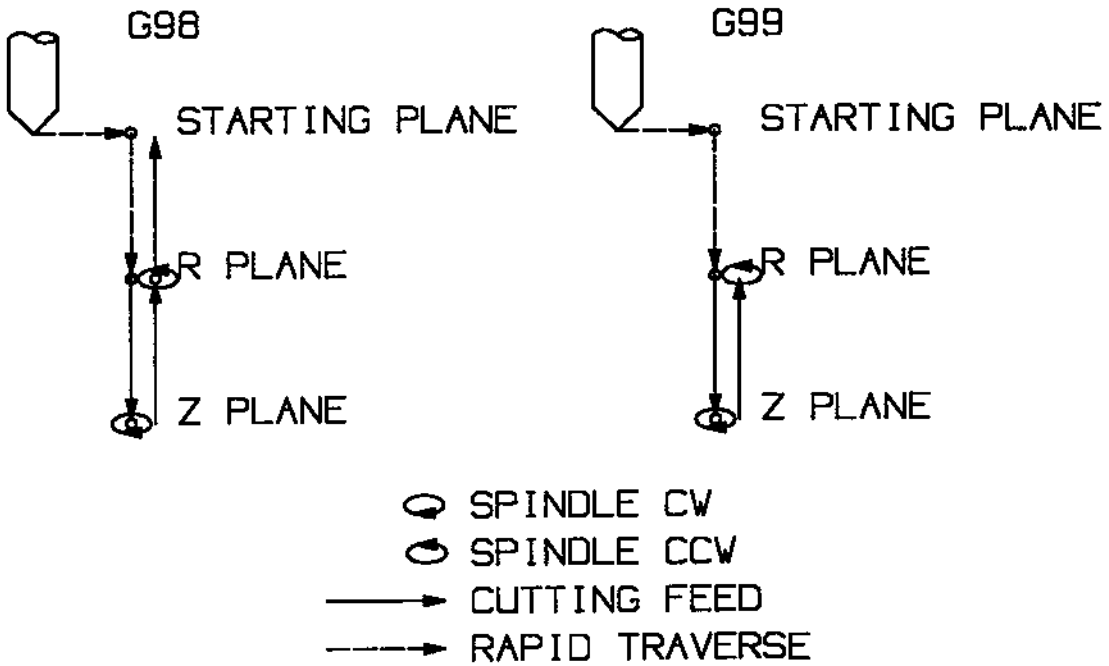
G76 Fine Boring Canned Cycle

Group 09

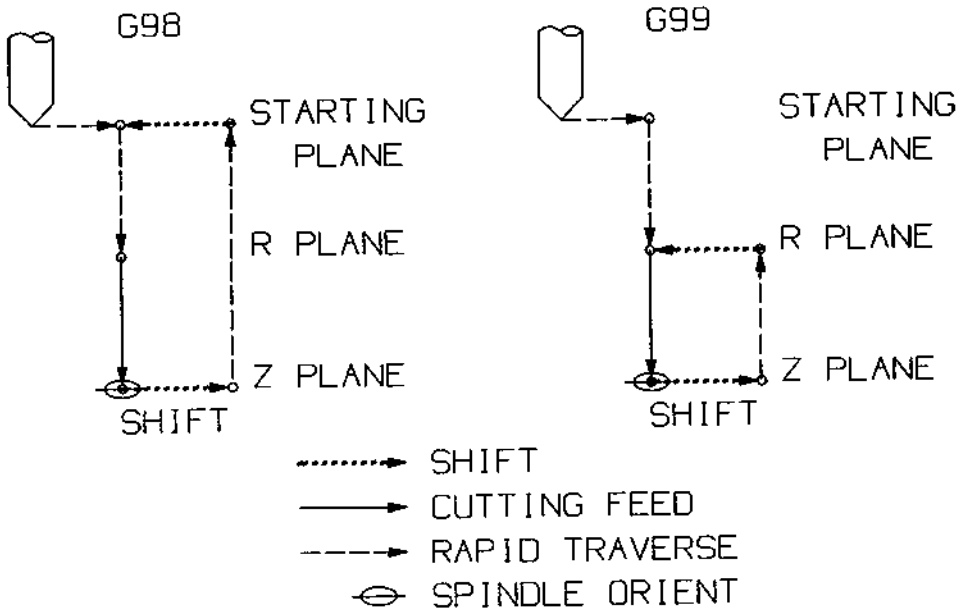
F	Feed Rate in inches (mm) per minute
L	Number of repeats
P	The dwell time at the bottom of the hole
Q	The shift value, always incremental
R	Position of the R plane
X	Optional X-axis motion command
Y	Optional Y-axis motion command
Z	Position of bottom of hole

This **G** code is modal in that it activates the canned cycle until it is canceled or another canned cycle is selected. Once activated, every motion of **X** or **Y** will cause this canned cycle to be executed. This cycle will shift the **X** or **Y**-axis prior to cutting in order to clear the tool while entering the part. This shift direction is set by Setting 27.

G74 REVERSE TAP CANNED CYCLE



G76 FINE BORING CANNED CYCLE

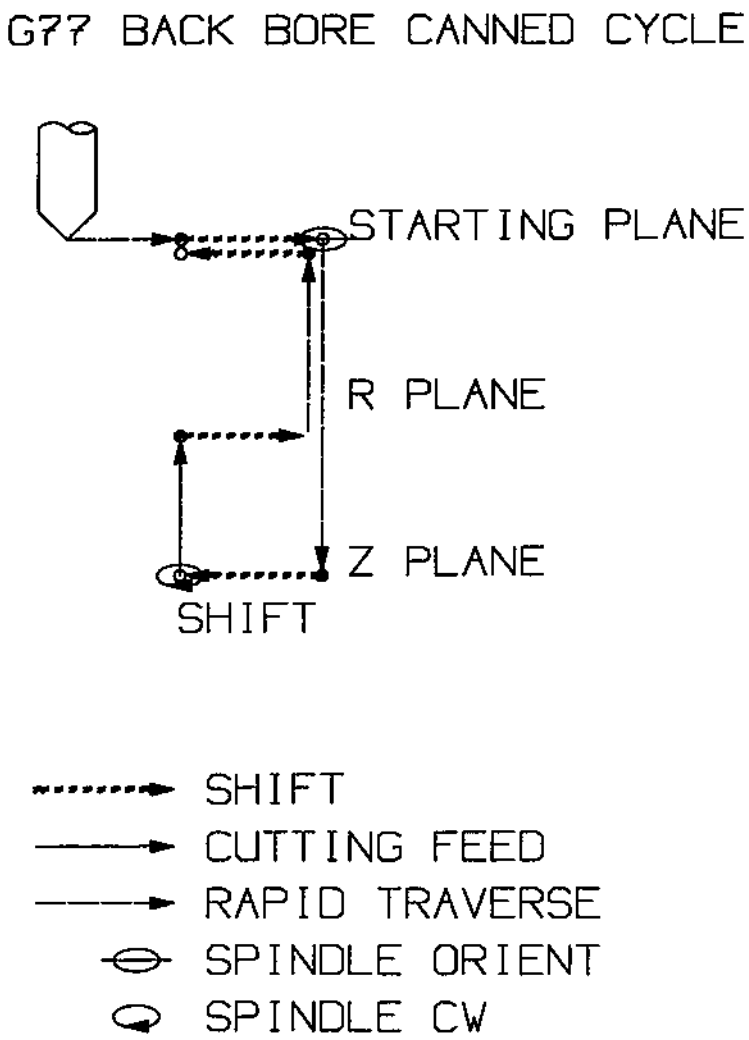


G77 Back Bore Canned Cycle

Group 09

F	Feed Rate in inches (mm) per minute
L	Number of repeats
Q	The shift value, always incremental
R	Position of the R plane
X	Optional X-axis motion command
Y	Optional Y-axis motion command
Z	Position of bottom of hole

This **G** code is modal in that it activates the canned cycle until it is canceled or another canned cycle is selected. Once activated, every motion of **X** or **Y** will cause this canned cycle to be executed.



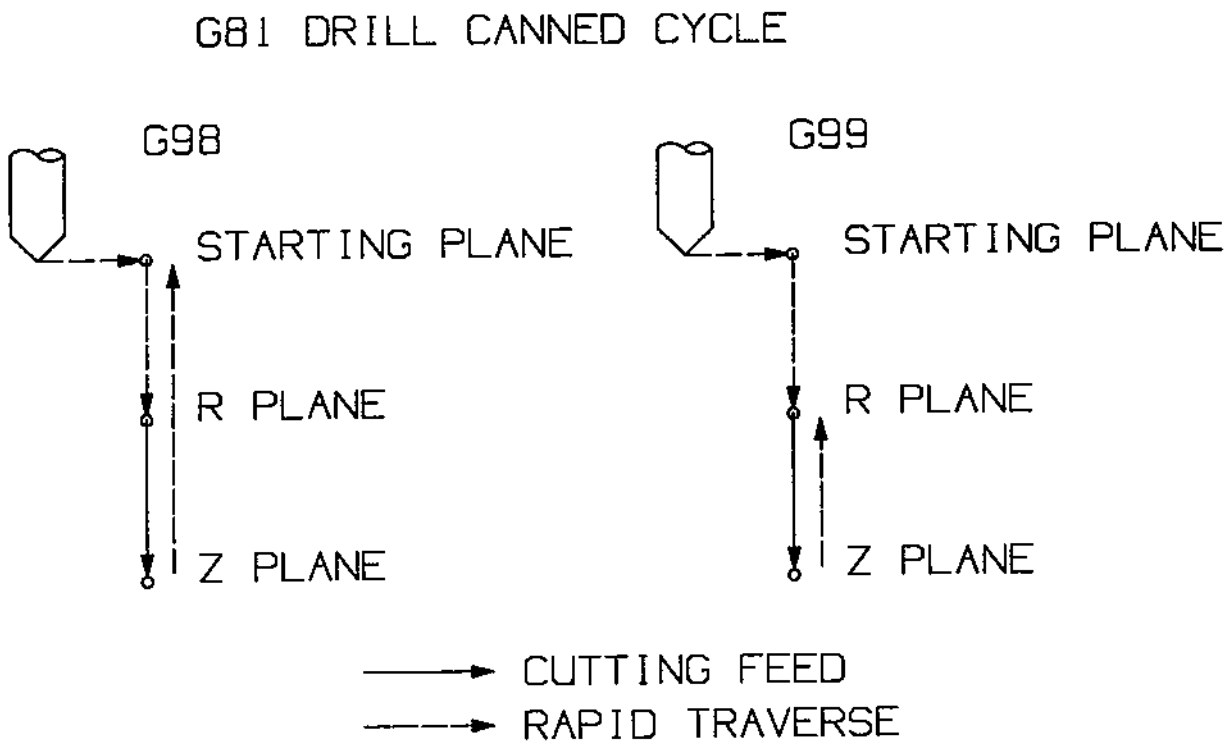
G80 Canned Cycle Cancel Group 09

This **G** code is modal in that it deactivates all canned cycles until a new one is selected. Note that use of G00 or G01 will also cancel a canned cycle.

G81 Drill Canned Cycle Group 09

F	Feed Rate in inches (mm) per minute
L	Number of repeats
R	Position of the R plane
X	Optional X-axis motion command
Y	Optional Y-axis motion command
Z	Position of bottom of hole

This **G** code is modal in that it activates the canned cycle until it is canceled or another canned cycle is selected. Once activated, every motion of **X** or **Y** will cause this canned cycle to be executed.

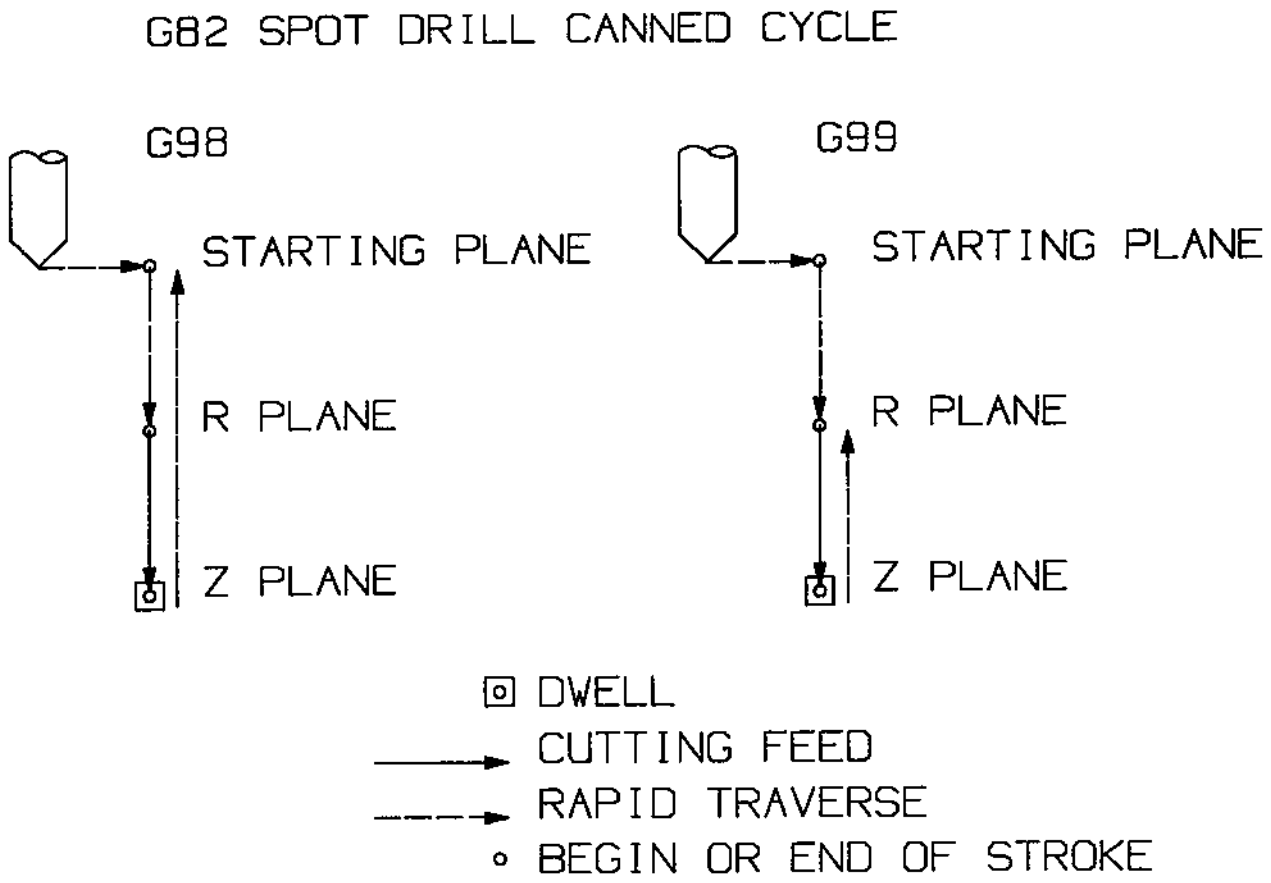


G82 Spot Drill Canned Cycle

Group 09

F	Feed Rate in inches (mm) per minute
L	Number of repeats
P	The dwell time at the bottom of the hole
R	Position of the R plane
X	Optional X-axis motion command
Y	Optional Y-axis motion command
Z	Position of bottom of hole

This **G** code is modal in that it activates the canned cycle until it is canceled or another canned cycle is selected. Once activated, every motion of **X** or **Y** will cause this canned cycle to be executed.



G83 Peck Drilling Canned Cycle

Group 09

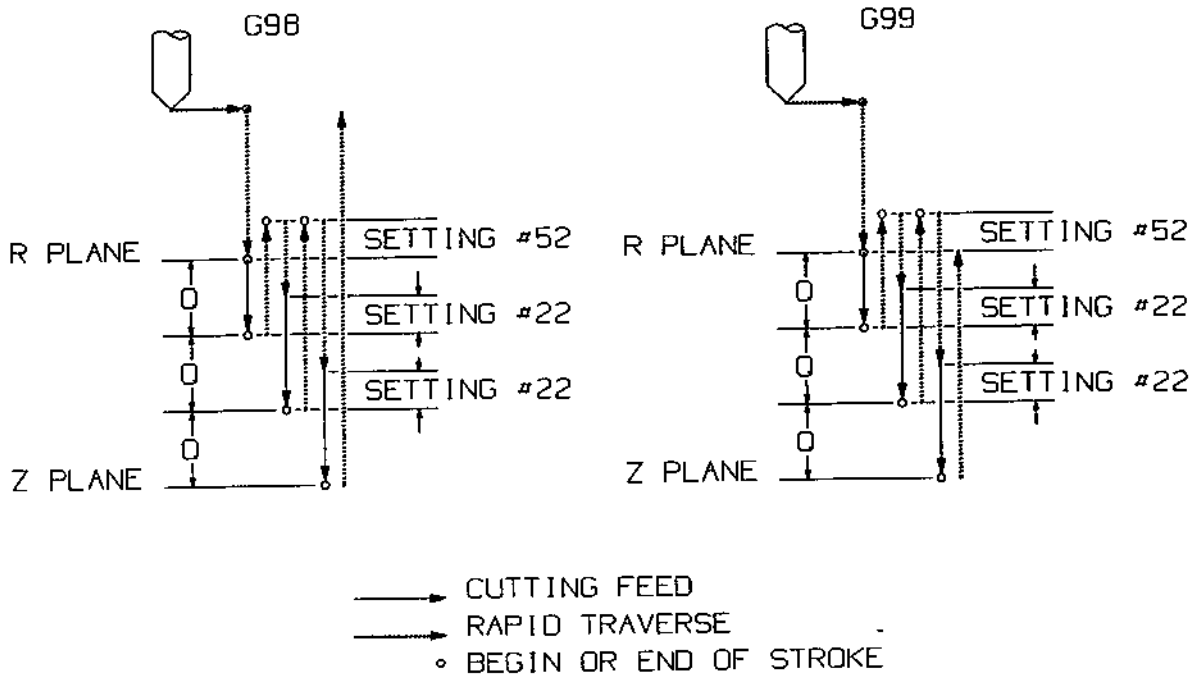
F	Feed Rate in inches (mm) per minute
I	Optional size of first cutting depth
J	Optional amount to reduce cutting depth each pass
K	Optional minimum depth of cut
L	Number of repeats
Q	The cut-in value, always incremental
R	Position of the R plane
X	Optional X-axis motion command
Y	Optional Y-axis motion command
Z	Position of bottom of hole

This **G** code is modal in that it activates the canned cycle until it is canceled or another canned cycle is selected. Once activated, every motion of **X** or **Y** will cause this canned cycle to be executed.

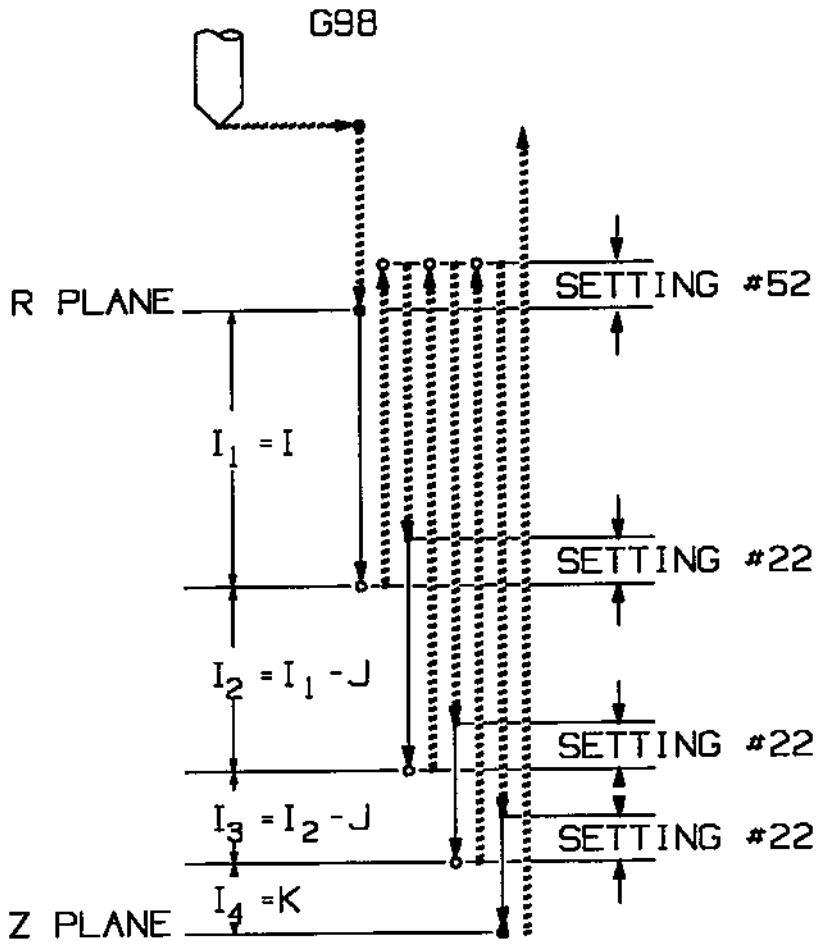
If **I**, **J**, and **K** are specified, a different operating mode is selected. The first pass will cut in by **I**, each succeeding cut will be reduced by amount **J**, and the minimum cutting depth is **K**.

Setting 52 also changes the way G83 works when it returns to the **R** plane. Most programmers set the **R** plane well above the cut to insure that the chip clear motion actually allows the chips to get out of the hole but this causes a wasted motion when first drilling through this "empty" space. If Setting 52 is set to the distance required to clear chips, the **R** plane can be put much closer to the part being drilled. When the clear move to **R** occurs, the **Z** will be moved above **R** by this setting.

G83 PECK DRILLING CANNED CYCLE



G83 PECK DRILLING CANNED CYCLE
WITH I, J, AND K OPTIONS



CUTTING FEED
RAPID TRAVERSE
○ BEGIN OR END OF STROKE

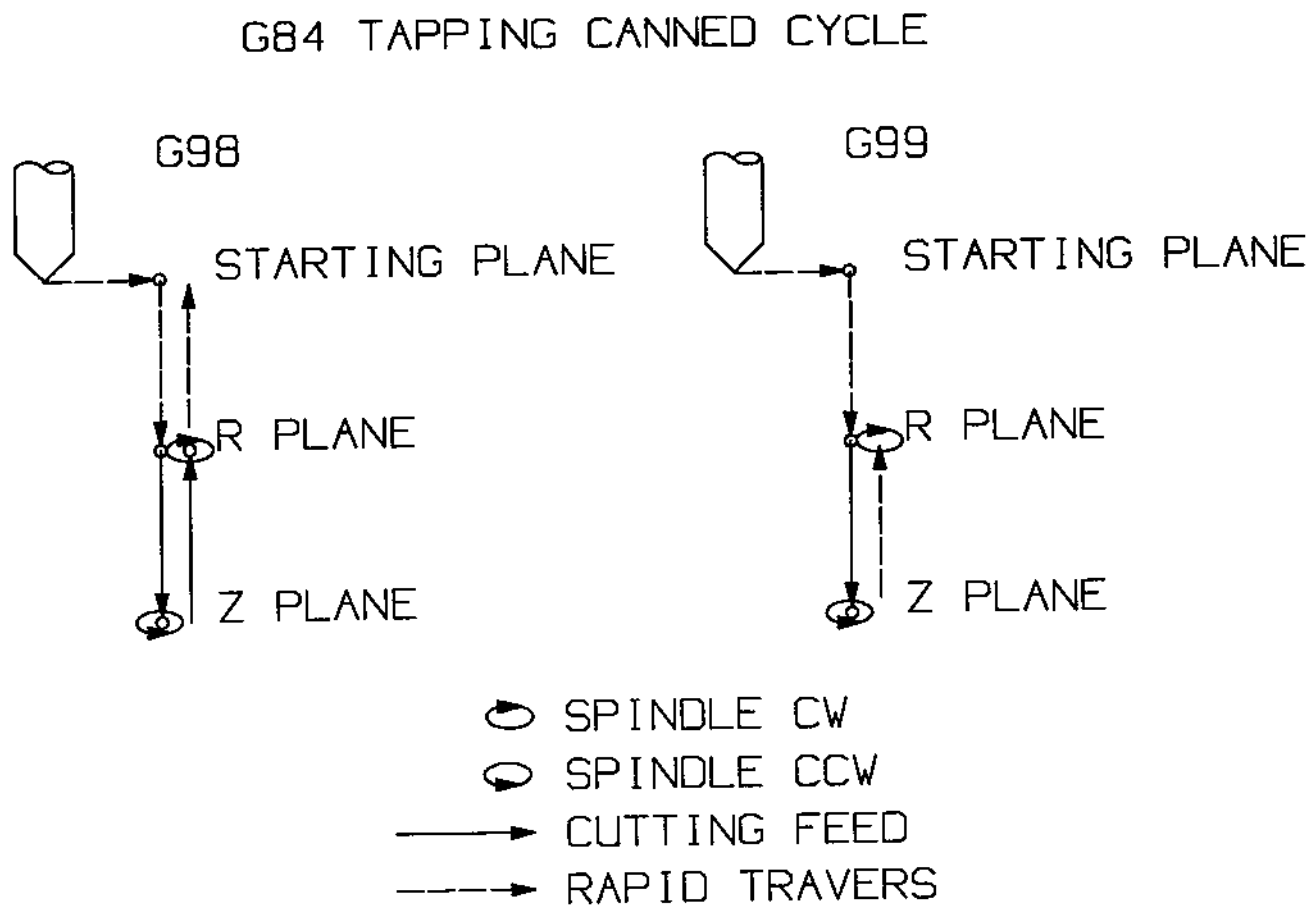
G84 Tapping Canned Cycle

Group 09

F	Feed Rate in inches (mm) per minute
L	Number of repeats
R	Position of the R plane
X	Optional X-axis motion command
Y	Optional Y-axis motion command
Z	Position of bottom of hole

This **G** code is modal in that it activates the canned cycle until it is canceled or another canned cycle is selected. Once activated, every motion of **X** or **Y** will cause this canned cycle to be executed. Note that operation of this cycle is different if the rigid tapping option is installed and selected (See Section 7.2). When rigid tapping is used, the ratio between the feed rate and spindle speed must be precisely the thread pitch being cut.

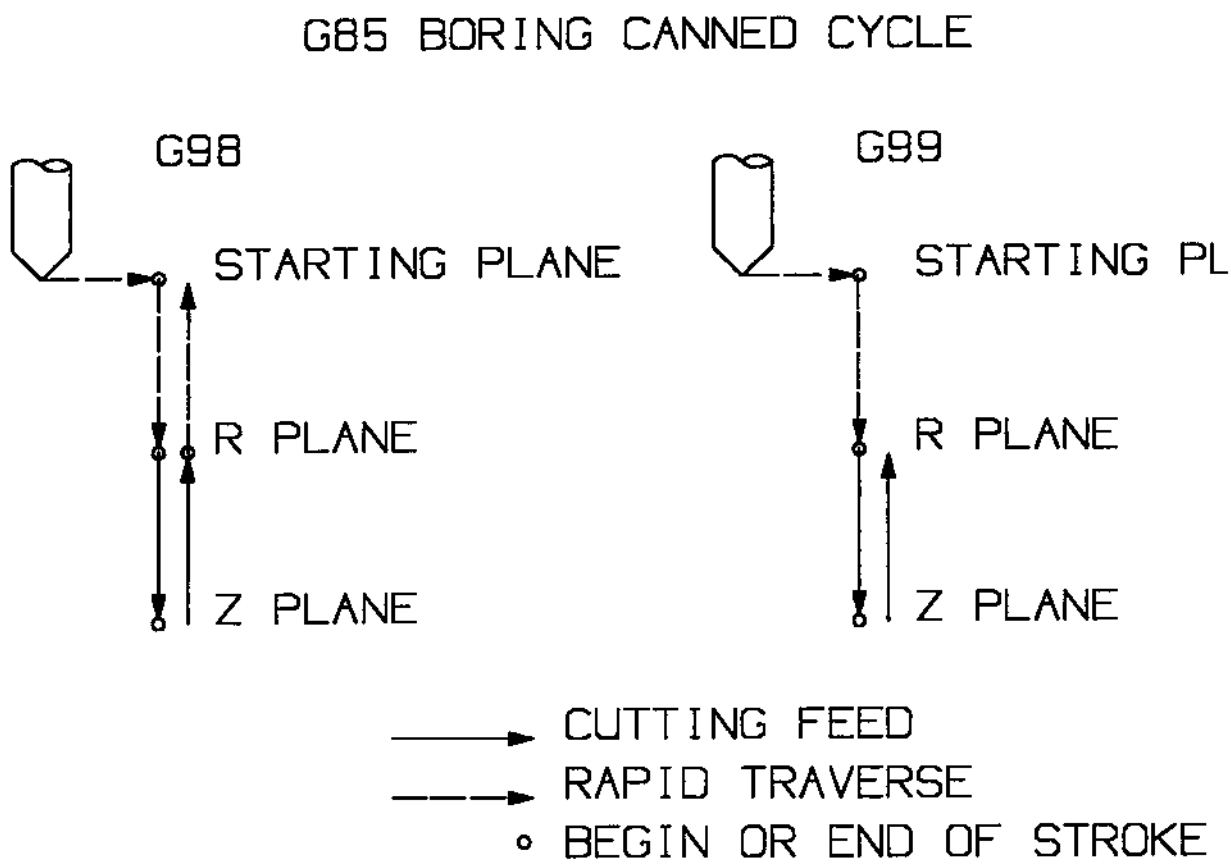
You do not need to start the spindle CW before this canned cycle. The control does this automatically.



G85 Boring Canned Cycle Group 09

F	Feed Rate in inches (mm) per minute
L	Number of repeats
R	Position of the R plane
X	Optional X-axis motion command
Y	Optional Y-axis motion command
Z	Position of bottom of hole

This **G** code is modal in that it activates the canned cycle until it is canceled or another canned cycle is selected. Once activated, every motion of **X** or **Y** will cause this canned cycle to be executed.

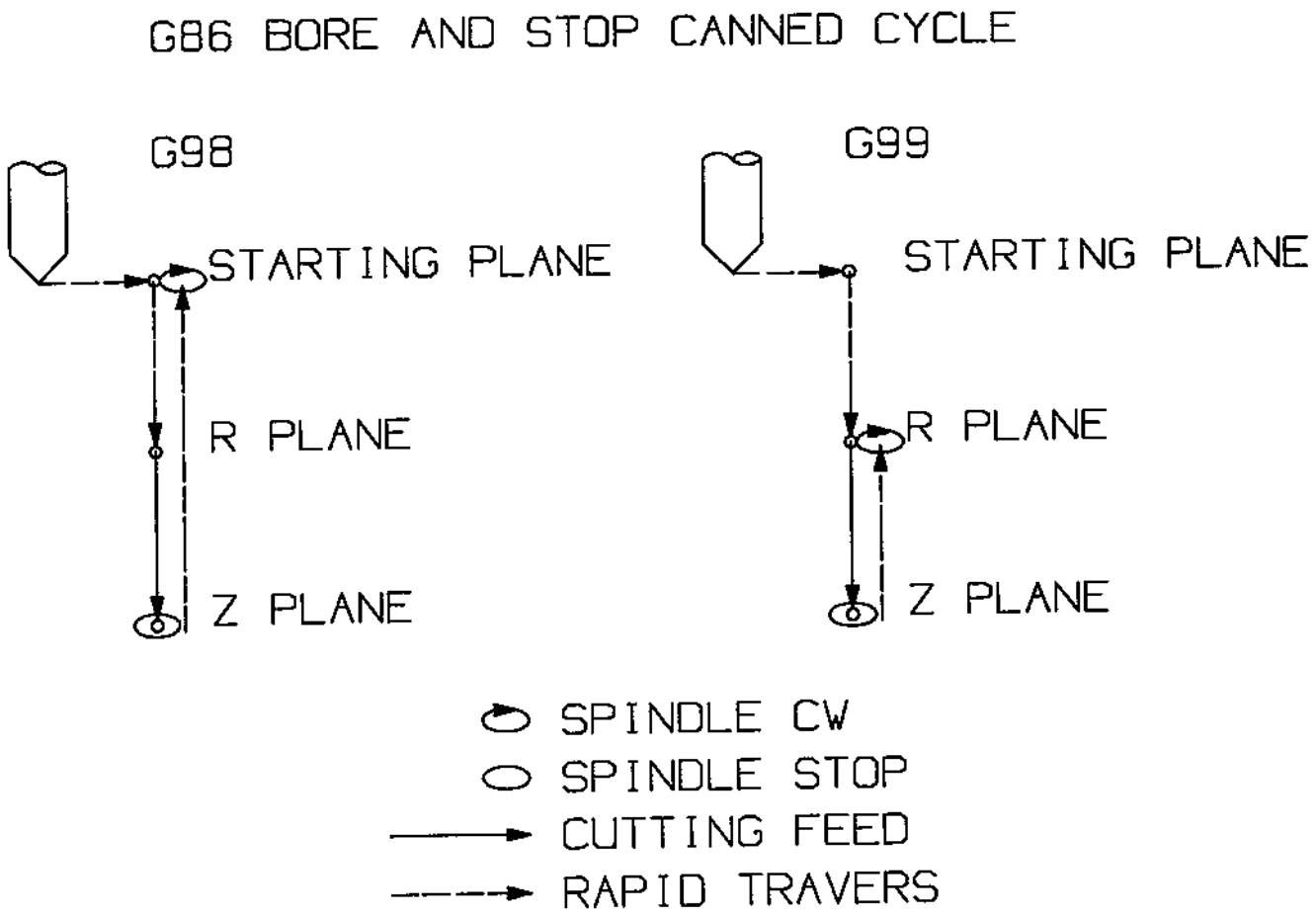


G86 Bore and Stop Canned Cycle

Group 09

F	Feed Rate in inches (mm) per minute
L	Number of repeats
R	Position of the R plane
X	Optional **X**-axis motion command
Y	Optional Y-axis motion command
Z	Position of bottom of hole

This **G** code is modal in that it activates the canned cycle until it is canceled or another canned cycle is selected. Once activated, every motion of **X** or **Y** will cause this canned cycle to be executed.

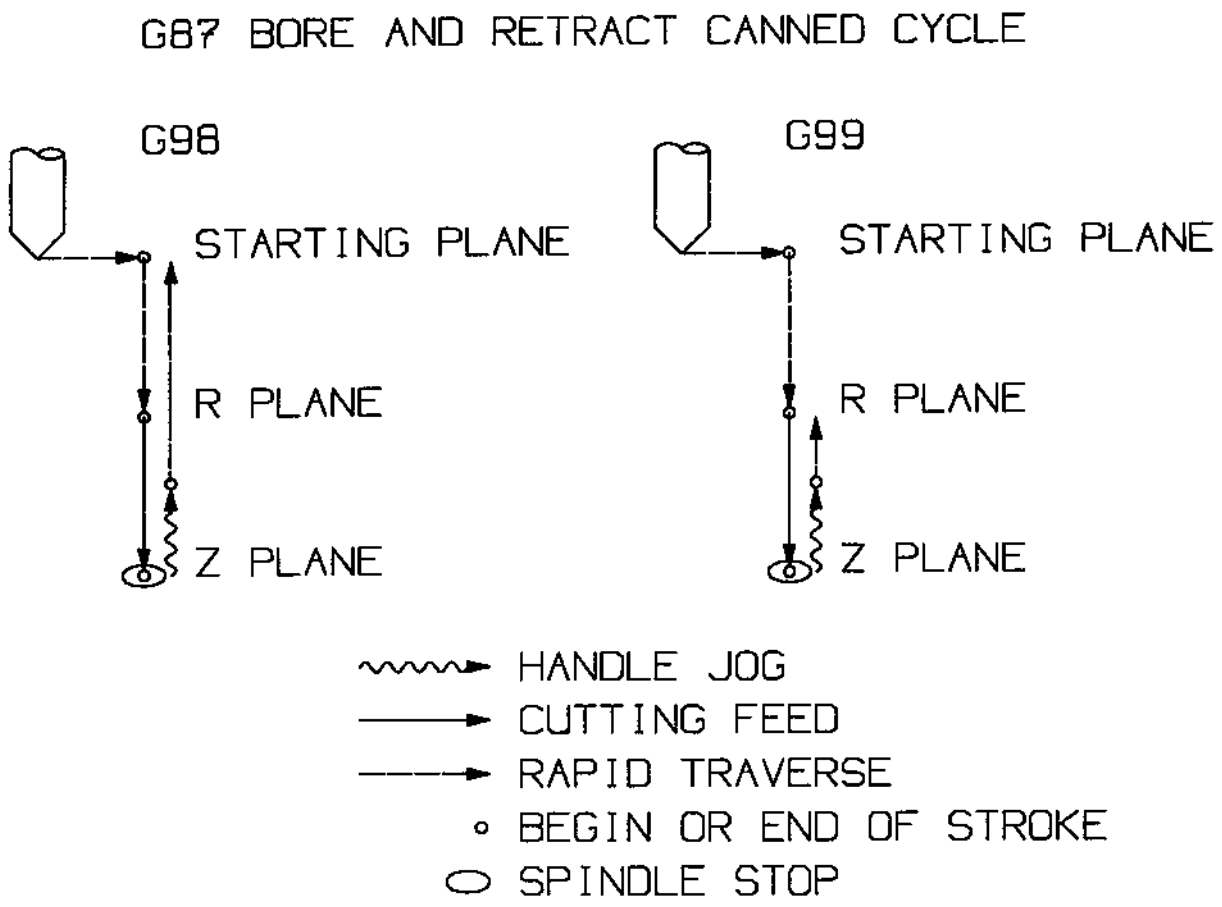


G87 Bore and Manual Retract Canned Cycle

Group 09

F	Feed Rate in inches (mm) per minute
L	Number of repeats
R	Position of the R plane
X	Optional X-axis motion command
Y	Optional Y-axis motion command
Z	Position of bottom of hole

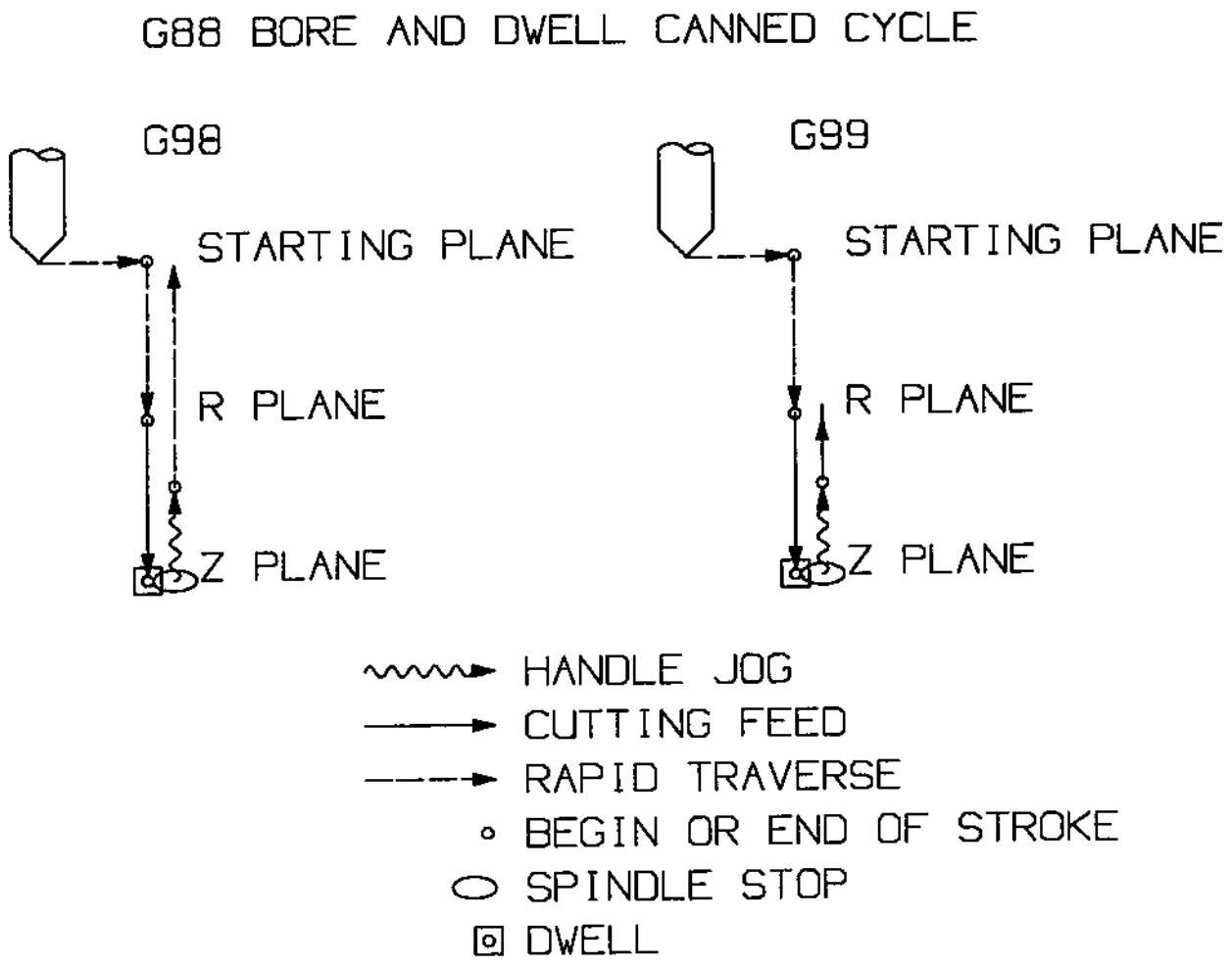
This **G** code is modal in that it activates the canned cycle until it is canceled or another canned cycle is selected. Once activated, every motion of **X** or **Y** will cause this canned cycle to be executed.



G88 Bore and Dwell Canned Cycle Group 09

F	Feed Rate in inches (mm) per minute
L	Number of repeats
P	The dwell time at the bottom of the hole
R	Position of the R plane
X	Optional X-axis motion command
Y	Optional Y-axis motion command
Z	Position of bottom of hole

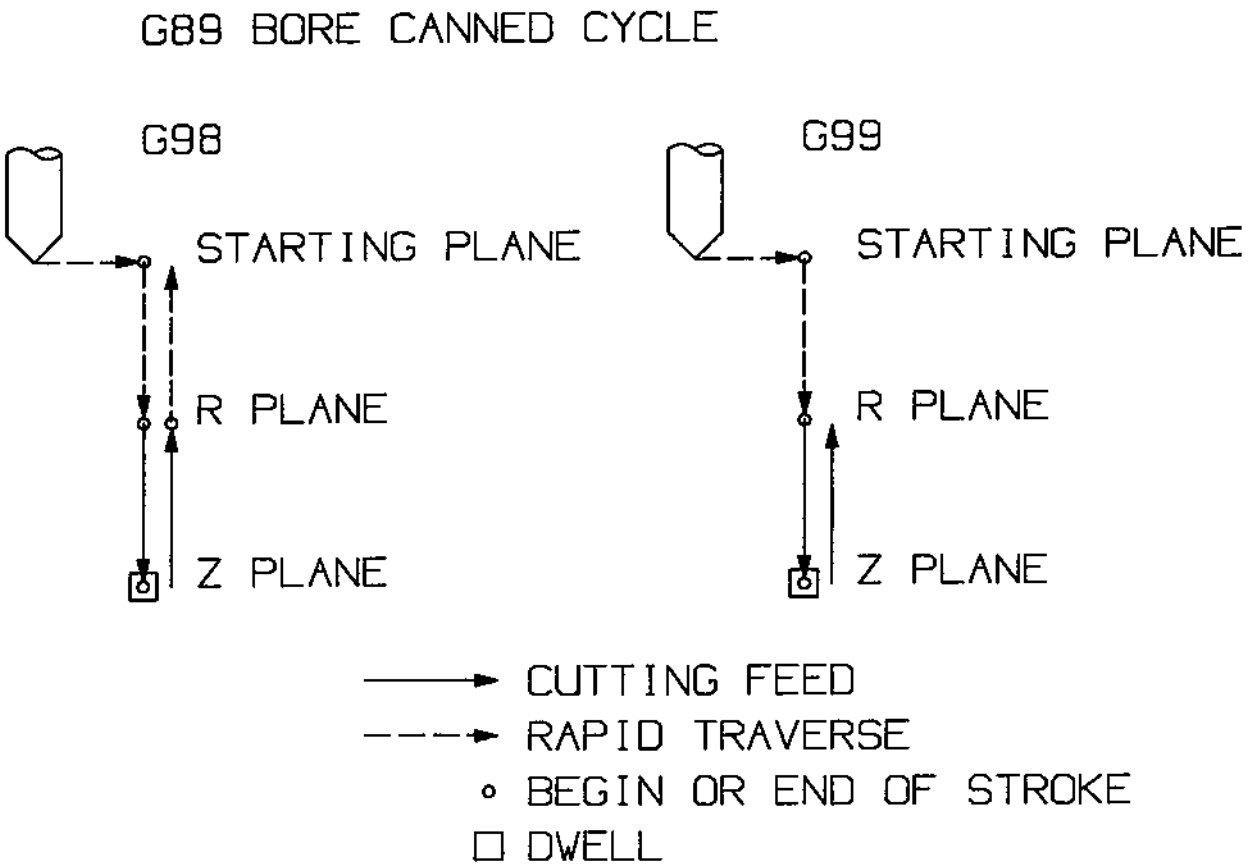
This **G** code is modal in that it activates the canned cycle until it is canceled or another canned cycle is selected. Once activated, every motion of **X** or **Y** will cause this canned cycle to be executed.



G89 Bore Canned Cycle Group 09

F	Feed Rate in inches (mm) per minute
L	Number of repeats
P	The dwell time at the bottom of the hole
R	Position of the **R** plane
X	Optional X-axis motion command
Y	Optional Y-axis motion command
Z	Position of bottom of hole

This **G** code is modal in that it activates the canned cycle until it is canceled or another canned cycle is selected. Once activated, every motion of **X** or **Y** will cause this canned cycle to be executed.



3.18 Absolute/Incremental Selection

G90 Absolute Position Commands Group 03

This code is modal and changes the way axis motion commands are interpreted. G90 makes all subsequent commands absolute positions within the selected user coordinate system. Each axis which is moved will be placed at the position coded in the command block.

G91 Incremental Position Commands Group 03

This code is modal and changes the way axis motion commands are interpreted. G91 makes all subsequent commands incremental. Each axis which is moved will be moved by the amount coded in the command block.

3.19 More Work Coordinate Selection

G92 Set Work Coordinates Group 00

This command works differently depending on the value of Setting 33. That setting selects either FANUC style of coordinates or YASNAC style of coordinates. This command does not move any of the axis; it only changes the values stored as user work offsets.

FANUC: The FANUC style of coordinates uses the G92 to modify all of the user work coordinates. All subsequent references to the user coordinates will be modified by the values of **X**, **Y**, **Z**, and **A** in the block.

YASNAC: The YASNAC style of coordinates uses the G92 to set the G52 work coordinates. All of the values of **X**, **Y**, **Z**, and **A** in this block are stored into the offsets entry for G52.

3.20 Canned Cycle Auxiliary Functions

G98 Canned cycle Initial Point Return Group 10

This **G** code is modal and changes the way canned cycles operate. With G98, the canned cycle will return to the initial starting point of the canned cycle when it completes.

G99 Canned cycle R Plane Return Group 10

This **G** code is modal and changes the way canned cycles operate. With G99, the canned cycle will return to the **R** plane when the canned cycle completes.

3.21 Programmable Mirror Image (G100, G101)

G100 Disable Mirror Image Group 00
G101 Enable Mirror Image Group 00

X Optional X-axis command
Y Optional Y-axis command
Z Optional Z-axis command
A Optional A-axis command

Programmable mirror image can be turned on or off individually for any of the four axes. The two **G** codes (G100 and G101) are non-modal but the mirror image status of each axis is modal. The bottom of the CRT will indicate when an axis is mirrored. These **G** codes should be used in a command block without any other **G** codes and will not cause any axis motion. G101 will turn on mirror image for any axis listed in that block. G100 will turn off mirror image for any axis listed in the block. The actual value given for the **X**, **Y**, **Z**, or **A** code has no effect.

When using Cutter Compensation with Mirror Imaging, follow this guideline: After turning Mirror Imaging ON or OFF with a G100 or G101, the next motion block should be to a different work coordinate position than the first one. The following code is an example:

INCORRECT:	CORRECT:
G41 X1.0 Y1.0	G41 X1.0 Y1.0
G01 X2.0 Y2.0	G01 X2.0 Y2.0
G101 X0	G101 X0
G00 Z1.0	G00 Z1.0
G00 X2.0 Y2.0	G00 X1.0
	G00 X2.0 Y2.0
G40	G40

The mirror function can change the direction of motion along any of the axes. If any one of these are selected, the display will show the status. Mirror image will reflect programmed motion around your work coordinate zero point. Be careful that mirror of only one of **X** or **Y** will cause the cutter to move along the opposite side of a cut. In addition, if mirror is selected for only one axis of a circular motion plane, circular motion G02 and G03 are reversed and left side and right side cutter compensation G41 and G42 are reversed. Settings 45 through 48 are used to select mirror image.

3.22 Programmable Output to RS-232 (G102)

G102 Programmable Output to RS-232 Group 00

X Optional X-axis command
Y Optional Y-axis command
Z Optional Z-axis command
A Optional A-axis command

Programmable output to the RS-232 port allows the current work coordinates of the four axes to be output. This **G** code (G102) is non-modal so only effects the block in which it is programmed. This **G** code should be used in a command block without any other **G** codes and will not cause any axis motion. The actual value given for the **X**, **Y**, **Z**, or **A** code has no effect. One complete line of text is sent to the first RS-232 port (same one used for upload, download, and DNC). Each axis listed in the G102 command block is output to the RS-232 port in the same format as values are displayed in a program.

Optional spaces (Setting 41) and EOB control (Setting 25) are applied. The values sent out are always the current axes positions referenced to the current work coordinate system.

Digitizing of a part is possible using this **G** code and a program which steps over a part in X-Y and probes downward in **Z** with a G31. When the probe hits, the next block could be a G102 to send the **X**, **Y**, **Z** position out to a computer which could store the coordinates as a digitized part.

3.23 More Work Coordinate Selection

G110-G129 Coordinate system #7-26 Group 12

These codes select one of the additional 20 user coordinate systems stored within the offsets memory. All subsequent references to axes positions will be interpreted in the new coordinate system. Operation of G110 to G129 are the same as G54 to G59.

3.24 General Purpose Pocket Milling Function

G150 General Purpose Pocket Milling Group 00

D	Cutter size selection
F	Feed rate
I	X-axis cut increment
J	Y-axis cut increment
K	Finishing cut allowance
L	Optional repetition count
P	Subroutine number defining outside of shape
Q	Incremental Z-axis cut depth per pass
R	R plane position
S	Optional spindle speed
X	X position of starting hole
Y	Y position of starting hole
Z	Final depth of the pocket

This **G** code provides for general purpose pocket milling. The shape of the pocket to be cut must be defined by a series of motions within a subroutine. A series of motions in either the X or Y-axis will be used to cut out the specified shape followed by a finishing pass to clean up the outer edge. One of either **I** or **J** must be specified. If **I** is used, the pocket is cut from a series of strokes in the X-axis. If **J** is used, the pocket is cut from a series of strokes in the Y-axis. **I** and **J** must be positive numbers. The finishing pass is of width **K** and **K** must be a positive number. There is no finishing pass in the **Z** depth.

Multiple passes over the area can be selected to control the depth of the cut. At least one pass is made over the pocket and multiple passes are made after feeding down by **Q** amount until the **Z** depth is reached. **Q** must be positive. If an **L** count is specified, the entire block is repeated and an incremental **X** or **Y** (G91) will reposition the pocket.

The subroutine must define a closed area by a series of G01, G02, or G03 motions in **X** and **Y** and must end with an M99. **G** codes G90 and G91 can also be used in the subroutine to select absolute or incremental. Any codes other than **G**, **I**, **J**, **R**, **X**, or **Y** are ignored in the subroutine. This subroutine must consist of less than 20 strokes.

Pocket milling should begin from a hole which has been previously drilled to the **Z** depth in order to clear the tool on entry to the pocket. The G150 block must specify this hole location with **X** and **Y**. The first motion in the subroutine should move from this clear hole to the starting point of the block shape. The final motion in the subroutine should return to the starting point of the shape.

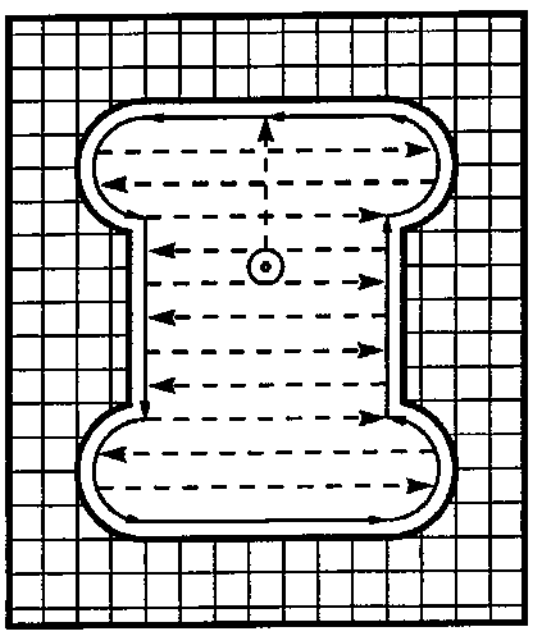
If a **K** is specified, the finishing pass is taken along the outside edge but is done at the full pocket depth and the previous cuts will cut inside the programmed pocket size by **K**.

```
O0100 (G150 POCKET EXAMPLE)
G58 G00 G90 X3.25 Y4.5 (STARTING HOLE POSITION)
T1 M06 (T1 CUTS ENTRY FOR END MILL)
```

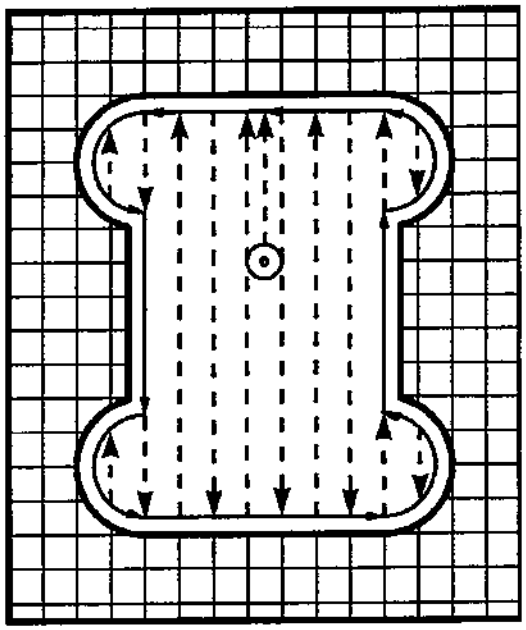
```
G83 R-1. Q0.5 Z-2. F20.
T2 M06 (END MILL T2 CUTS POCKET)
(0.4 DIA CUTTER, TWO PASSES TO Z DEPTH)
(LEAVE 0.1 FOR FINISH PASS)
G150 G41 F15. D01 J0.35 K0.1 Q0.5 R-1. X3.25 Y4.5 Z-2. P200
G40 G28
M30

O0200 (G150 POCKET SUBROUTINE)
G01 Y7.
X1.5
G03 Y5.25 R0.875
G01 Y2.25
G03 Y0.5 R0.875
G01 X5.
G03 Y2.25 R0.875
G01 Y5.25
G03 Y7. R0.875
G01 X3.25
M99 (RETURN FROM SUBROUTINE)
```

G150 ADVANCED POCKET MILLING



J>0



I>0

⊙ = STARTING HOLE

4. Circular Plane Selection

The plane used for circular motions must be comprised of two of **X**, **Y**, or **Z**. There are three **G** codes used to select the plane; G17 for XY, G18 for XZ, and G19 for YZ.

Plane selection applies also to G12 and G13 circular pocket milling which must always be in the X-Y plane (G17).

The default plane selection when the machine is powered on is G17 for the X-Y plane. This means that a circular motion in the plane of the X-Y table may be programmed without first selecting G17. In this plane, circular motion is defined as clockwise for the operator looking down onto the X-Y table from above. This is the motion of the tool relative to the table.

In the X-Z plane (G18), circular motion is defined as clockwise for the operator looking towards the rear of the machine from the front control panel.

In the Y-Z plane (G19), circular motion is defined as clockwise for the operator looking across the table from the side of the machine the control panel is mounted.

A helical motion is possible with G02 or G03 by programming the linear axis which is not in the selected plane. This third axis will be interpolated along the specified axis in a linear manner while the other two axes will be moved in the circular motion. The speed of each axis will be controlled so that the helical rate matches the programmed feed rate.

If cutter radius compensation is selected (G41 or G42), you may only use the X-Y plane for circular motions (G17). Cutter radius compensation is only available in the **X** and **Y** axes.

5. Inch / Metric Selection (G20, G21)

Selection between inch and metric programming can only be done from the Setting page, Setting 9. Inch programming allows displacements up to 838 inches and a resolution of 0.0001 inches. Metric programming uses millimeter units with a maximum displacement of 8380 mm and a resolution of 0.001 mm.

When in metric, the feed rate is also defined as millimeters per minute with a range of 1000. to 0.001 mm/min.

When jogging in metric, the speeds and units on the keypad are interpreted as mm/min but the value used is ten times larger than shown on the keypad.

The optional fourth or fifth axis programming is not effected by the selection of metric. It is always programmed in degrees. The auxiliary **C** axis is also always in degrees.

Changing the setting from inches to metric or back again will change the content of any programs already stored in memory. You must reload your programs with metric values after changing this setting.

The standard **G** codes G20 and G21 are sometimes used to select between inch and metric BUT, in this control, the G20 (inch) and G21 (mm) codes can only be used to insure that the inch/metric setting is set correctly for that program.

6. Work Coordinate System

These machines have three linear axes named **X**, **Y**, and **Z**. The X-axis moves the table left and right, the Y-axis moves it to and from the operator and the **Z** moves the milling head up and down. The machine zero position is the upper right corner of the mill table. All moves from this point are in a negative machine direction. If a rotary table is connected, an additional A-axis work offset is provided.

The work offset display is found on the offset display by pushing the PAGE UP key. You can display and manually enter work offsets from here. The work coordinate systems on a control with a fifth axis have all been expanded to accommodate **B**, the fifth axis. Work coordinate offsets can be set for the **B** axis in the offset display. Note that the auxiliary axes **C**, **U**, **V**, and **W** do not have any offsets; they are always programmed in machine coordinates.

The Home or Machine-Zero position is X0, Y0, Z0. All motion from machine zero is in a negative direction. Travel of these axes is limited in the negative direction by stored stroke limits defined in the parameters. Travel in the positive direction for the **X** and **Y** axes is limited simply to values less than zero. Positive travel for the Z-axis is limited to the highest position used for tool changing (about Z4.5). In addition, positive travel for all axes is limited by the home switch which acts as a limit switch.

Before a tool can machine your part, the control must know where your part is. The work coordinate system tells the control the distance from the work zero point of your part to the machine zero position. The work zero point of the part is decided by the programmer and usually is the common point where all print dimensions are referenced from. The machine zero position is fixed by the machine on power up and does not change. The operator must determine this distance and enter the value.

There are two work coordinate systems that can be used. A G92 code followed by an **X** and **Y** value and a Z0 on the first line of the program is the most common. This first line then tells the control the distance from part zero to machine home. A better way is to use the G54 through G59 or G110 through G129 work offset. They do the same thing that a G92 code does except that the G54 value is located outside the program in the offset display.

The advantages of the G54 are: 1) on power up, the machine automatically establishes its coordinate system; 2) the operator does not need to edit the program; 3) a crash cannot occur when the program is restarted at a point other than home as can in a G92 which establishes a new reference point each time it is read; 4) there are twenty six offsets available; 5) you can load and save offsets just as you load and save programs.

This control automatically chooses the G54 system on power up. If you do not wish to use this system, zero out the values in the G54 **X**, **Y**, and **Z** or select another work offset.

The G54 through G59 or G110 through G129 offsets can be set by using the PART ZERO SET key. Position the axes to the work zero point of your part. Using the cursor, select the proper axis and work number. Press the PART ZERO SET key and the current machine position will be automatically stored in that address. This will work with only the work zero offsets display selected. Note that entering a nonzero **Z** work offset will interfere with the operation of an automatically entered tool length offset.

Work coordinate numbers are usually entered as positive numbers. Normally the Z-axis work coordinate is zero because the offset is supplied by the tool offset number (**T** command). An exception to this would be if you wish to raise the tool above **Z** zero for program setup. By putting a Z-1.0 into the **Z** work coordinate that tool will start one inch above **Z** zero on the part. If you wish to raise all the tools, enter a Z-1.0 into the G52 work coordinate. The G52 work coordinate shifts all work coordinates by the amount entered but applies only when using the Fanuc type of coordinate system (Setting 33).

Work coordinates are entered into the table as a number only, that is, to enter an X value of X2.00 into G54 you would cursor over to the column and enter the number 2.0 only.

The mirror function can change the direction of motion along any of the axes. If any one of these are selected, the display will show the status. Mirror image will reflect programmed motion around your work coordinate zero point. Be careful that mirror of only one of **X** or **Y** will cause the cutter to move along the opposite side of a cut. In addition, if mirror is selected for only one axis of a circular motion plane, circular motion G02 and G03 are reversed and left side and right side cutter compensation G41 and G42 are reversed. Settings 45 through 48 are used to select mirror image.

Offsets can be sent and received with the RS-232 port. See Chapter III, Section 13 for a description of how to do this.

7. Spindle Speed Functions

7.1 Spindle Speed Commands

Spindle speed functions are controlled primarily by the **S** address code. The **S** address specifies RPM in integer values from 1 to maximum spindle speed (Parameter 131). NOT TO BE CHANGED BY USER!

Speeds from S1 to the Parameter 142 value (usually 1200) will automatically select low gear and speeds above Parameter 142 will select high gear. Two **M** codes, M41 and M42 can be used to override the gear selection. M41 for low gear and M42 for high gear. Low gear operation above S1250 is not recommended. High gear operation below S100 may lack torque or speed accuracy.

If there is no gear box in your machine (VF-0) the gear box is disabled by parameters, it is always in high gear, and M41 and M42 commands are ignored.

Three **M** codes are used to start and stop the spindle. M03 starts the spindle clockwise, M04 starts the spindle counterclockwise, and M05 stops the spindle.

Note that only one **M** code is allowed in a block. This means that if you wish to override the gear with M41 or M42, you must put the **Snnnn** and M41 (or M42) in one block and the M03 (or M04) in the next block. The **Snnn** should always be in the same block as the M41 or M42 as an unneeded double gear change might otherwise be performed.

7.2 Rigid Tapping Control Of Spindle

Rigid tapping is an option which can be installed on this machine at the factory. It is enabled with the Parameter 57 "Rigid Tap" flag. When installed and enabled, it changes the way G74 and G84 work and a floating tap holder is not needed for these **G** codes. In addition, if the "REPT RIG TAP" flag in Parameter 57 is set, every repetition of a tapping operation will control the orientation of the spindle so that the tapping is repeatable.

Rigid tapping allows the use of a tap without a floating tap holder. Pitch control is within 0.0005 inch. Bottom depth control is +/-0.020 inch and repeatability is +/-0.005 inch. Rigid tapping will operate from 100 to 2000 RPM and up to 100 inches per minute feed. Bottom depth control is better at lower speeds and low gear. Thread pitch can be from 5 to 100 TPI. Feeds above 50 inches per minute work better with a change to Parameters 103 and 115.

The pitch of a tapped hole is defined by the ratio between the feed rate and spindle speed. When rigid tapping is selected, these two must be set exactly. An encoder mounted with the spindle tracks the position of the spindle and the Z-axis is moved precisely to match the pitch of the thread. If the repeatable option is selected, a position pulse from the encoder is used to synchronize the starting of the **Z** motion with the position of the spindle.

Note that with G74 and G84, you do not ever need to use M03, M04, or M05. These canned cycles start and stop the spindle automatically. This applies to using normal or rigid tapping.

The rigid tapping option requires additional hardware be added to the spindle head assembly and a different electronics card put into the control. When this option is installed, the second page of diagnostic data will show the actual spindle speed. See Section 14.1 for further details.

8. Tool Functions (Tn)

The **Tnn** code is used to select the next tool to be placed in the spindle from the tool changer. The **T** address does not start the tool change operation; it only selects which tool will be used next. M06 and M16 are used to start a tool change operation. The **Tnn** does not need to be in a block prior to the M06 or M16; it can be in the same block.

Note: there is no **X** or **Y** motion required prior to performing a tool change and it would waste time in most cases to return **X** or **Y** to the home position. However, if your work piece or fixture is quite large, you may need to position **X** or **Y** prior to a tool change in order to prevent a crash between the tools and your fixture.

Other controls may require the programmer to position the Z-axis to machine zero prior to a tool change but this is not required with this control. You may command a tool change with **X**, **Y**, and **Z** in any position and the control will bring the **Z** up to the machine zero position prior to starting the tool change. The control will move the **Z** to a position above machine zero during a tool change but will never move below machine zero. At the end of a tool change, the Z-axis will be at machine zero.

The tool changer is an all electric fixed shuttle type. Tools are always loaded through the spindle and should never be installed directly in the carousel in order to avoid crashes. The pocket open to the spindle must always be empty in the retracted position.

The tool holders used are #40 taper, V flange, commonly called "CT 40". The tool changer is manufactured to hold either BT40 or CAT40 tools. They are NOT interchangeable.

For machines equipped for CAT40 tools, use a "45 degree, P40T type 1, inch threads" pull stud built to JMTBA standard "MAS 403-1982". This pull stud is characterized by a long shaft and a 45 degree shoulder under the head. Do not use the short shaft or pull studs with a sharp right angle (90 degree) head as they will not work and will cause serious damage.

For machines equipped for BT40 tools, use only HAAS pull studs (PN: 22-7165).

Tool holders and pull studs must be in good condition and tighten together with wrenches or they may stick in the spindle taper. Clean the tool tapers with a lightly oiled rag to leave a film to prevent rusting. Tools that make a loud bang when being released indicate a problem and should be checked before serious damage to the shuttle occurs. When the TOOL RELEASE button is pressed the tool should be pushed out of the spindle by a small amount (approximately .07 Inch). This is an indication that the pull stud is correctly touching the release mechanism.

Low air pressure or insufficient volume will reduce the pressure applied to the tool unclamp piston and will slow down tool change time or will not release the tool.

After AUTO POWER UP and ZERO RET, the control will insure that the tool changer is in a normal position. To load a new tool, select MDI mode, put the tool in the spindle with the TOOL RELEASE button and then push the ATC FWD or ATC REV and the machine will put the tool in the carousel. Use the CURNT COMDS display to see what tool is currently in the spindle.

To manually select another tool, use the "ATC FWD" or "ATC REV" key while in the MDI mode. To select a tool other than an adjacent one, enter the "**T**" number first. That is: "T8" "ATC FWD" will place tool 8 in the spindle.

If the shuttle should become jammed, the control will automatically come to an alarm state. To correct this, push the EMERGENCY STOP button and remove the cause of the jam. Push the RESET key to clear any alarms. Push the ZERO RET and the AUTO ALL AXES keys to reset the Z-axis and tool changer. Never put your hands near the tool changer when powered unless the EMERGENCY STOP button is pressed first.

The tool changer is protected by fuse FU5, located on the POWER PCB. It might be blown by an overload or jam of the tool changer. Operation of the tool changer can also be interrupted by problems with the tool clamp/unclamp and the spindle orientation mechanism.

There are some other **M** codes which will also cause tool operations to occur:

M19 Will orient the tool for special user functions
M39 Will rotate the tool turret without changing tools
M82 Will unclamp the tool (be careful, it will fall!)
M86 Will clamp the tool

9. Miscellaneous Functions (M Functions)

9.1 M Code Summary

Only one **M** code may be programmed per block of a program. All **M** codes are effective or cause an action to occur at the end of the block and only one **M** code is allowed in each block.

M00	Stop Program
M01	Optional Program Stop
M02	Program End
M03	Spindle Forward
M04	Spindle Reverse
M05	Spindle Stop
M06	Tool Change
M08	Coolant On
M09	Coolant Off
M10	Engage 4th Axis Brake
M11	Release 4th Axis Brake
M16	Tool Change (same as M06)
M19	Orient Spindle
M21-M28	Optional Pulsed User M Function with Fin
M27	Apply fifth axis brake, wait until M-fin signal is received through the secondary RS-232 port.
M30	Prog End and Rewind
M39	Rotate Tool Turret
M41	Low Gear Override
M42	High Gear Override
M51-M58	Optional User M turn ON
M57	Apply fifth axis brake, continue with program.
M61-M68	Optional User M turn OFF
M67	Release fifth axis brake, continue with program.
M75	Set G35 or G136 reference point
M76	Disable Displays
M77	Enable Displays
M78	Alarm if skip signal found
M79	Alarm if skip signal not found
M82	Tool Unclamp
M86	Tool Clamp
M97	Local Sub-Program Call
M98	Sub Program Call
M99	Sub Program Return Or Loop

9.2 M Code Detailed Description

M00 Stop Program

The M00 code is used to stop a program. It also stops the spindle and turns off the coolant. The program pointer will advance to the next block and stop. A cycle start will continue program operation from the next block.

M01 Optional Program Stop

The M01 code is identical to M00 except that it only stops if OPTIONAL STOP is turned on from the front panel. A cycle start will continue program operation from the next block.

M02 Program End

The M02 code will stop program operation the same as M00 but does not advance the program pointer to the next block.

M03 Spindle Forward

The M03 code will start the spindle moving is a clockwise direction at whatever speed was previously set. The block will delay until the spindle reaches about 90% of commanded speed.

M04 Spindle Reverse

The M04 code will start the spindle moving is a counterclockwise direction at whatever speed was previously set. The block will delay until the spindle reaches about 90% of commanded speed.

M05 Spindle Stop

The M05 code is used to stop the spindle. The block is delayed until the spindle slows below 10 RPM.

M06 Tool Change

The M06 code is used to initiate a tool change. The previously selected tool (**Tn**) is put into the spindle. If the spindle was running, it will be stopped. No previous axis commands are required before the tool change unless there is a problem with tool/part/fixture clearance. The Z-axis will automatically move up to the machine zero position and the selected tool will be put into the spindle. The Z-axis is left at machine zero. The spindle will not be started again after the tool change but the **Snnnn** speed and gear will be unchanged. The **Tnn** must be in the same block or in a previous block. The coolant pump will be turned off during a tool change.

M08 Coolant On

The M08 code will turn on the coolant supply. Note that the **M** code is performed at the end of a block; so that if a motion is commanded in the same block, the coolant is turned on after the motion. The low coolant status is only checked at the start of a program so a low coolant condition will not stop a program which is already running.

M09 Coolant Off

The M09 code will turn off the coolant supply.

M10 Engage 4th Axis Brake

The M10 code is used to apply the optional brake to the 4th axis. It is only used when M11 is used to release the brake.

M11 Release 4th Axis Brake

The M11 code will "pre-release" the 4th axis brake. This is useful to prevent the delay otherwise occurring when a 4th axis is used with a brake and a motion is commanded in that axis. It is not required but, without a prior M11, there will be a delay in motion in order to release the air.

M16 Tool Change

The M16 code is used to initiate a tool change. In the present machine configuration, M16 works exactly like M06.

M19 Orient Spindle

The M19 code is used to orient the spindle to a fixed position. This command leaves the spindle in that position and locked by a pin. The next spindle motion command (**Snnnn**, M3, M4, M41, or M42) will release the pin and unlock the spindle.

M21-M28 Optional User M

The M21 thru M28 codes are optional for user interfaces. They will activate one of relays 17 through 24, wait for the M-fin signal, release the relay, and wait for the M-fin signal to cease. The RESET button will terminate any operation that is hung-up waiting for M-fin.

M27 Release fifth axis Brake, Wait For M-fin signal.

This code activates the fifth axis brake relay, which must be connected to relay M27 on the I/O board. It activates the relay, waits for the M-fin signal, releases the relay upon receipt, and waits for the M-fin signal to cease. The RESET key will terminate any operation that is hung-up waiting for M-fin.

M30 Prog End and Rewind

The M30 code is used to stop a program. It also stops the spindle and turns off the coolant. The program pointer will be reset to the first block of the program and stop. The parts counters displayed on the Current Commands display are also incremented. M30 will also cancel tool length offsets.

M39 Rotate Tool Turret

The M39 code is used to rotate the tool turret without performing a tool change. This is not normally required but is useful for diagnostic purposes or to recover from a tool changer crash. Remember that the pocket facing the spindle must always be empty for a tool change. This **M** code may be useful to move an empty pocket to face the spindle.

M41 Low Gear Override

The M41 code is used to override the spindle gear implied by the **Snnn** command. With M41, the spindle gear will always be low. If the speed commanded is above the low gear limit, the spindle speed will be the low gear limit. This **M** code does not turn the spindle on or off. If the spindle was turning before this command, it will be started again. If it was stopped before this command it will be left off. M41 is ignored if there is no gear box.

M42 High Gear Override

The M42 code is used to override the spindle gear implied by the **Snnn** command. With M42, the spindle gear will always be high. Note that this may reduce the torque at the tool. This **M** code does not turn the spindle on or off. If the spindle was turning before this command, it will be started again. If it was stopped before this command it will be left off. M42 is ignored if there is no gear box.

M51-M58 Optional User M ON

The M51 thru M58 codes are optional for user interfaces. They will activate one of relays 17 through 24 and leave it active. These are the same relays used for M21-M28. Use M61-M68 to turn these off. The RESET button will turn off all of these relays.

M57 Release fifth axis Brake.

This will activate relay M27 (fifth axis brake) and leave it active. Use M67 to engage the fifth axis brake.

M61-M68 Optional User M OFF

The M61 thru M68 codes are optional for user interfaces. They will deactivate one of relays 17 through 24. These are the same relays used for M21-M28.

M67 Engage fifth axis Brake.

This will deactivate relay M27 (fifth axis brake). Use M57 to release the fifth axis brake.

M75 Set G35 or G136 Reference Point

This code is used to set the reference point used for G35 and G136. It must be used after a motion which is terminated with the skip function.

M76 Disable Displays

This code is used to disable the updating of the screen displays during high speed machining.

M77 Enable Displays

This code is used to enable the updating of the screen displays at the end of high speed machining.

M78 Alarm if skip signal found

This code is used to generate an alarm if the previous skip function actually got the skip signal. This is usually used when a skip signal is not expected and may indicate a probe crash. This code can be placed in a block with the skip function or in any subsequent block. The skip functions are G31, G36, and G37.

M79 Alarm if skip signal not found

This code is used to generate an alarm if the previous skip function did not actually get the skip signal. This is usually done when the absence of the skip signal means a positioning error of a probe. This code can be placed in a block with the skip function or in any subsequent block. The skip functions are G31, G36, and G37.

M82 Tool Unclamp

This code is used to release the tool from the spindle. It is not normally needed as tool change operations do this automatically and a manual TOOL RELEASE button is available to the operator. THIS M CODE IS NOT RECOMMENDED FOR USE AS THE TOOL WILL BE DROPPED FROM THE SPINDLE AND MAY DAMAGE THE TOOL, THE MACHINE, OR YOUR SETUP.

M86 Tool Clamp

This code will clamp a tool into the spindle. It is not normally needed as tool change operations do this automatically and a manual TOOL RELEASE button is available to the operator.

M97 Local Sub-Program Call

This code is used to call a subroutine referenced by a line **N** number within the same program. A **Pnnnn** code is required and must match a line number within the same program. This is useful for simple subroutines within a program and does not require the complication of a separate program. The subroutine must still be ended with an M99. An **L** count on the M97 block will repeat the subroutine call that number of times.

M98 Sub Program Call

This code is used to call a subroutine. The **Pnnnn** code is the number of the program being called. The **Pnnnn** code must be in the same block. The program by the same number must already be loaded into memory and it must contain an M99 to return to the main program. An **L** count can be put on the line containing the M98 and will cause the subroutine to be called **L** times before continuing to the next block.

M99 Sub Program Return Or Loop

This code is used to return to the main program from a subroutine or macro. It will also cause the main program to loop back to the beginning without stopping if it is used in other than a subprogram without a **P** code. If an M99 **Pnnnn** is used, it will cause a jump to the line containing **Nnnnn** of the same number.

10. Cutter Compensation Description

Cutter compensation is a method of shifting the tool path so that the actual finished cut is moved to either the left or right of the programmed path. Normally cutter compensation is programmed to shift by exactly the radius of the tool so that the finished cut matches the programmed path. The Offset display page is used to enter the amount for the tool to be shifted. The offset can be entered as either diameter or radius for both a geometry and a wear value. The effective value is the sum of the geometry and wear value. Setting 40 is used to select either diameter or radius. If diameter is specified, the shift amount is half of the value entered. Cutter radius compensation is only available in the X-Y-axis (G17).

10.1 General Description of Cutter Compensation

G41 will select cutter compensation left; that is, the tool is moved to the left of the programmed path to compensate for the size of the tool. A **Dnn** must also be programmed to select the correct tool size from compensation memory. If compensation memory contains a negative value for cutter size, cutter compensation will operate as though G42 was specified. Cutter path compensation in this machine applies only to motion in the **X** and **Y** axes.

G42 will select cutter compensation right; that is, the tool is moved to the right of the programmed path to compensate for the size of the tool. A **Dnn** must also be programmed to select the correct tool size from compensation memory. If compensation memory contains a negative value for cutter size, cutter compensation will operate as though G41 was specified.

The code G40 will cancel cutter compensation and is the default condition when a machine is powered-on. When canceled, the programmed path is the same as the center of the cutter path. You may not end a program (M30, M00, M01, or M02) with cutter compensation active.

If cutter radius compensation is selected (G41 or G42), you may only use the X-Y plane for circular motions (G17). Cutter radius compensation is only available in the **X** and **Y** axes.

There is a simple rule about cutter compensation which helps to understand the motions the control uses to compensate for tool size. The control operates on one motion block at a time. It will look ahead, however, to check the next two blocks containing **X** or **Y** motions. The interference checks are performed on these three motions. Setting 58 controls how this part of cutter compensation works. It can be set to Fanuc or Yasnac:

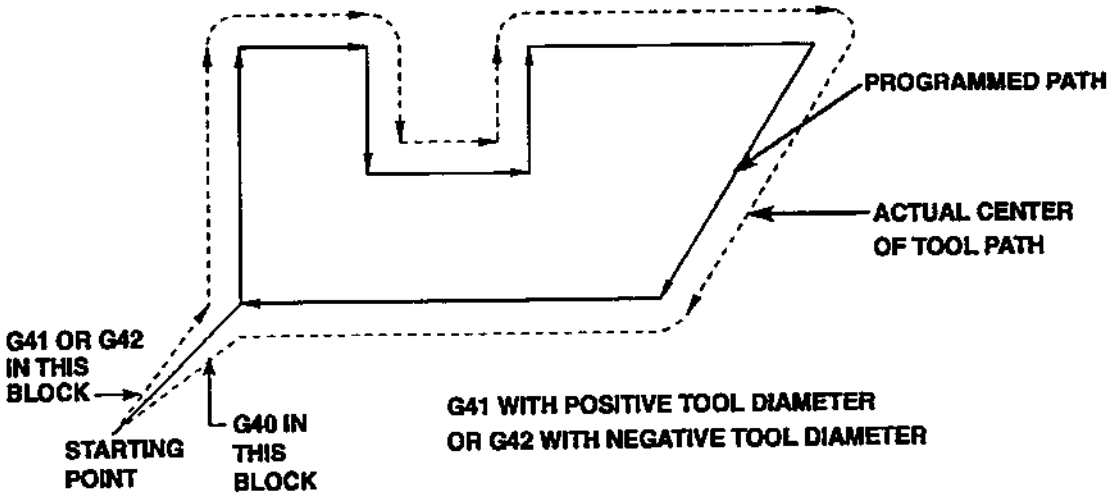
When Setting 58 is set to Yasnac, the control must be able to position the tool edge along all of the programmed cuts without overcutting the next two motions. All outside angles are joined by a circular motion.

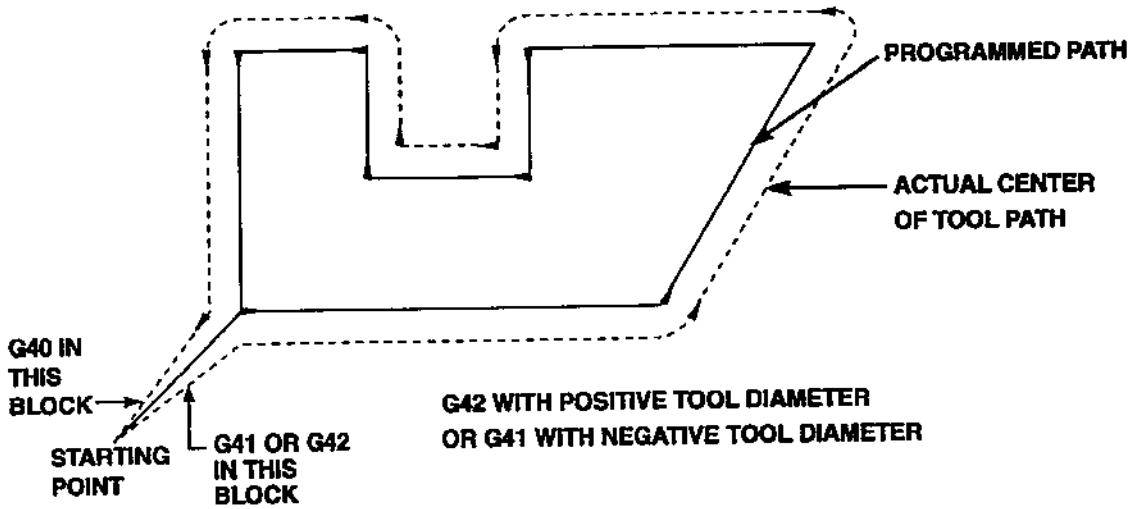
When Setting 58 is set to Fanuc, the control does not require that the tool cutting edge be placed along all programmed cuts. Overcutting, however, is still prevented and, if overcutting cannot be prevented, an alarm will still occur. Outside angles less than or equal to 270 degrees are joined by a square corner and outside angles of more than 270 degrees are joined by an extra linear motion.

The following two diagrams show how cutter compensation works for the two possible values of Setting 58. Note that a small cut of less than tool radius and at right angle to the previous motion will only work with the Fanuc setting.

CUTTER COMPENSATION

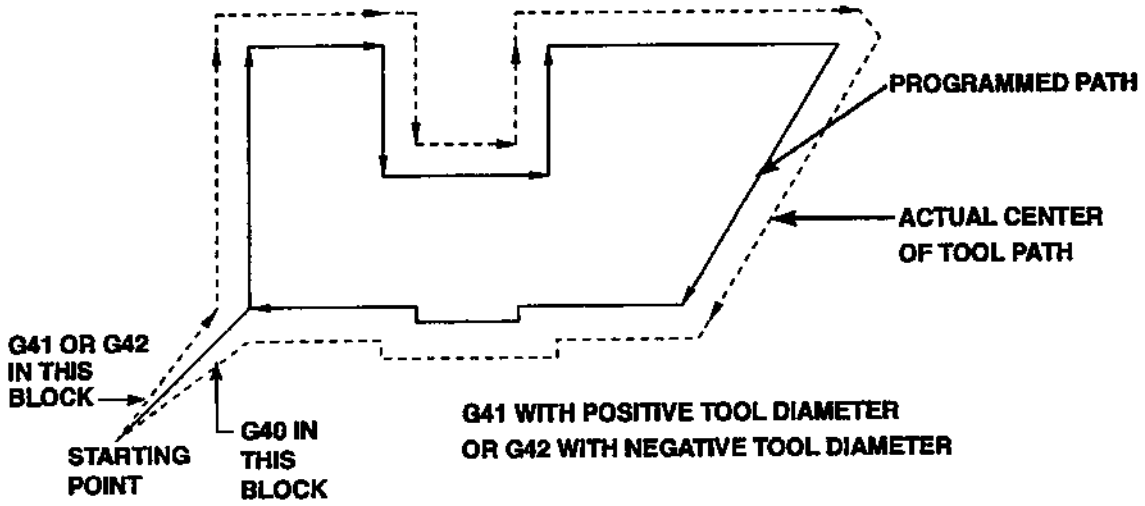
(Yasnac style)

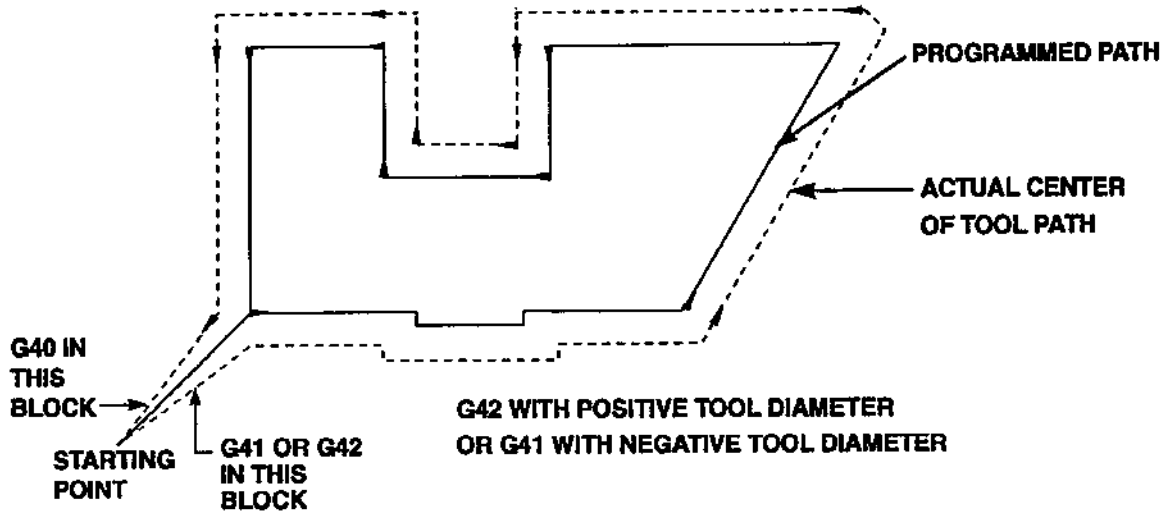




CUTTER COMPENSATION

(Fanuc style)





10.2 Entry and Exit From Cutter Compensation

When entering and exiting cutter compensation or when changing from left side to right side compensation, there are special considerations to be aware of. Cutting should not be performed during any of these three type of moves. In a block that turns on cutter compensation, the starting position of the move is the same as the programmed position but the ending position of the move will be offset by the cutter compensation size. In a block that turns off cutter compensation, the starting point is offset and the ending point is not offset. Similarly, when a block changes from left to right compensation, the starting point is shifted in one direction and the ending point is shifted in the other direction. The result of all of this is that the tool is moved through a path that may not be the same as the intended path or direction.

If cutter compensation is turned on or off in a block without any X-Y move, there is no change made to cutter compensation until the next **X** or **Y** move is encountered. To enter cutter compensation, a nonzero **D** code must be specified and either G41 or G42 specified. To exit from cutter compensation, you may either specify D0 or G40 or both.

You should always turn off cutter compensation in a move which clears the tool away from the part being cut. If a program is terminated with cutter compensation still active, an alarm is generated. In addition, you cannot turn on or off cutter compensation during a circular move (G02 or G03); otherwise an alarm will be generated.

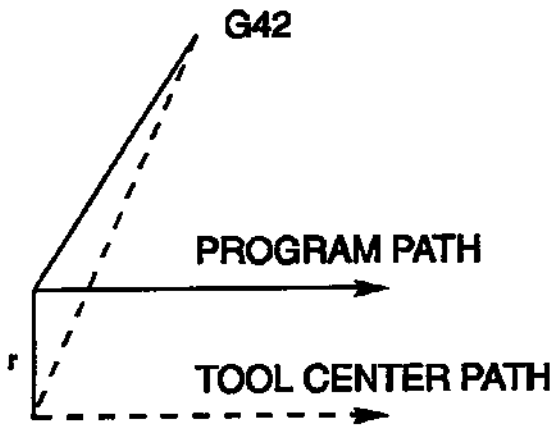
An offset selection of D00 will use zero as the offset size and have the same effect as turning off cutter compensation. If a new value from offset memory is selected while cutter compensation is active, the starting point of a move will reflect the old value and the ending point will reflect the new value. This will also have the effect of shifting the motion to something other than what was intended by the programmer. You cannot change the offset code or side during a circular motion block.

When turning on cutter compensation in a move followed by a second move at an angle of less than 90 degrees, there are two common ways of computing the first motion. They are called cutter compensation type A and B. Type A will not stay on the programmed side of the first cut but will go directly to the starting point for the second cut. Type B will remain clear of the first line and follow it with the same motions as described in 10.1 to position for the second cut. Types A and B are selected with Setting 43.

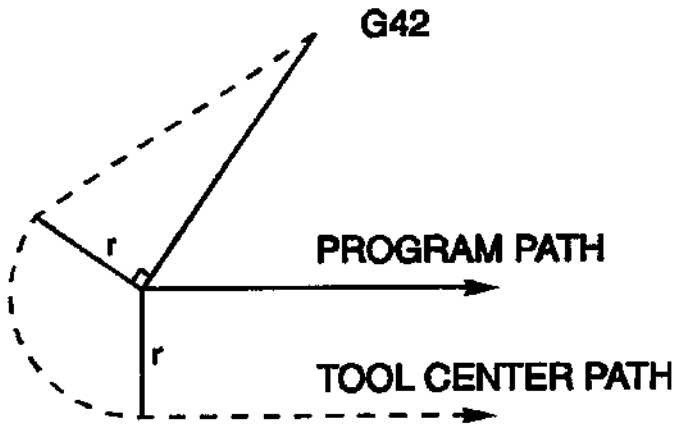
Setting 58 also changes the way the entry and exit to cutter compensation works. There is still a type A or B but the type of moves used to clear the tool from the beginning of the cut change as described in the previous section. The following two diagrams describe how this works.

CUTTER COMPENSATION ENTRY
(Yasnac style)

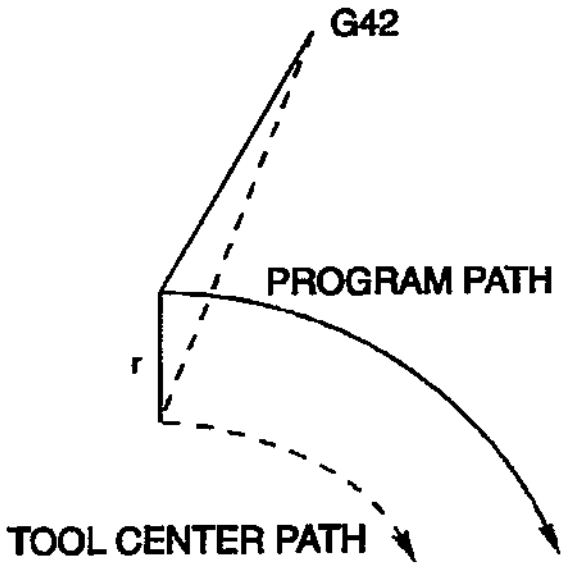
TYPE A

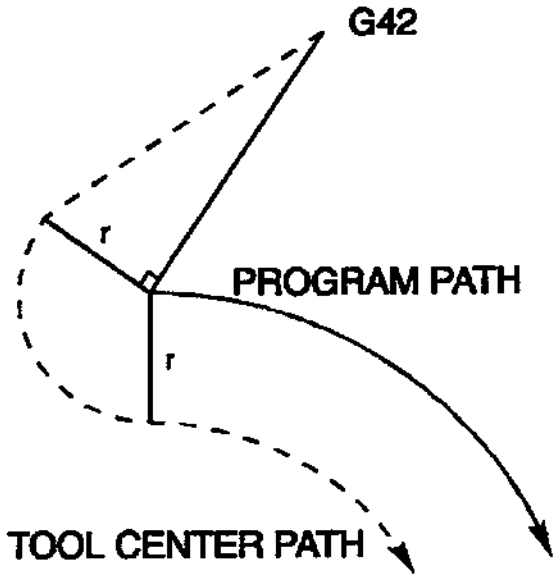


TYPE B

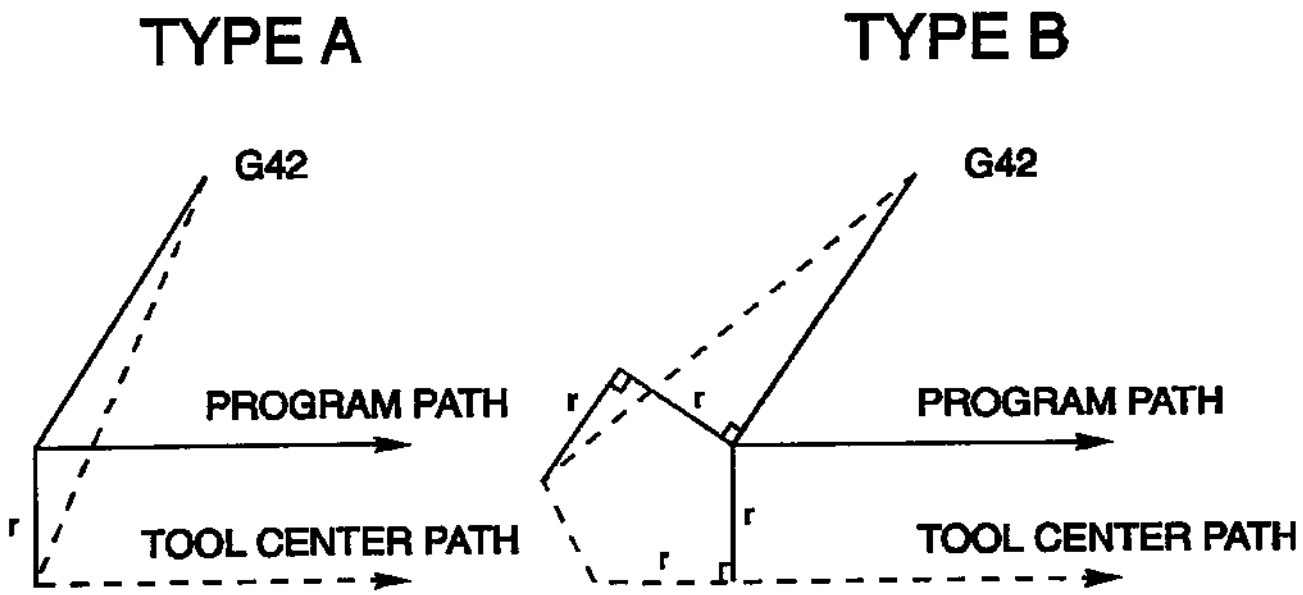


r = TOOL RADIUS

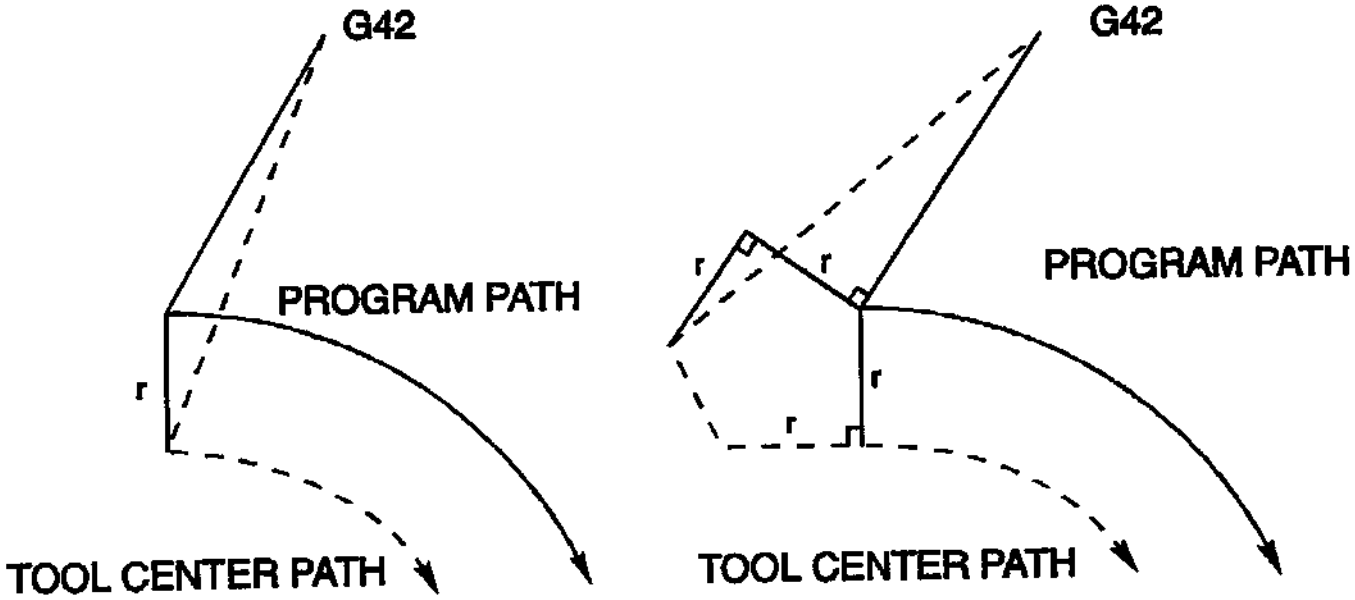




CUTTER COMPENSATION ENTRY
(Fanuc style)



r = TOOL RADIUS



10.3 Feed Adjustments In Cutter Compensations

When using cutter compensation in circular moves, there is the possibility of speed adjustments to what has been programmed. If the intended finish cut is on the outside of a circular motion, the tool should be slowed down to ensure that the surface feed does not exceed what was intended by the programmer. There are problems, however, when the speed is slowed by too much. For this reason, Setting 44 is used to limit the amount by which the feed is adjusted in this case. It can be set between 1% and 100%. If set to 100%, there will be no speed changes. If set to 1% the speed can be slowed to 1% of the programmed feed.

When the cut is on the inside of circular motion, there is no speedup adjustment made to the feed rate.

11. Automatic Acceleration/Deceleration

This machine is not capable of instantly changing speed; it takes some nonzero time to accelerate and decelerate. Acceleration and deceleration in this machine have both a constant accel/decel mode and an exponential mode. Constant acceleration is used at the beginning of a rapid move and at the end of any move whose speed exceeds the exponential accel/decel time constant.

11.1 Constant Acceleration

Constant acceleration is a type of motion when the amount of speed change over time is constant. This constant is set by Parameters 7, 21, 35, and 49. It has units of encoder increments per second per second.

Constant acceleration applies to the beginning of a rapid move so that the minimum time is spent getting up to rapid speed. It also applies to the end of rapid moves until the speed drops below the exponential accel/decel time constant. That change occurs at about 100 inches per minute in the following example.

11.2 Exponential Acceleration

Exponential acceleration/deceleration is a type of motion where the speed is proportional to the distance remaining in a programmed travel. The exponential accel/decel time constant is set by Parameters 113, 114, 115, and 116. It has units of 0.0001 seconds. The speed limit at which exponential accel/decel is not available is defined by the relationship between Parameters 7 and 113 (for the X-axis). Thus, if Parameter 7 is 1500000 steps/sec/sec and Parameter 113 is 375 (0.0375 seconds); the maximum velocity for exponential accel/decel would be:

1500000 x 0.0375 = 56250 steps/second
For a 2000 line encoder and 6 mm screw, this would be:
60 x 56250 / 33867 = 100 inches/minute

11.3 Acceleration in Feed Motions

In the normal feed cutting mode, with G64 active, giving continuous cutter motion, deceleration of the axes in motion begins at some distance away from the end point. If lookahead has buffered another motion, the acceleration for that motion will begin at the same instant. This means that two motions, at right angles to each other, will not produce a perfectly square corner. The corner will be rounded. In addition, two motions which smoothly blend one into the other will not cause the tool to pause.

If you use cutter compensation to cut an outside corner, there will be no rounding if the cutter compensation amount is close to the actual tool size. This is because the tool is moved beyond the end of the first programmed stroke before it is moved to the beginning of the second stroke. Note that in this machine, using the default parameter settings, rapid and feed moves will both be blended to provide continuous cutter path and rounded corners. Unless you specify exact stop, the following rapid or feed block will be started slightly before the completion of the previous block.

The end of a feed move is delayed until the following error is below an amount set in Parameters 101...104. If this is set to 2500 (corresponding to about 0.0725 inches), with Parameter 113 set to 375 (0.0375 seconds), this means that the highest feed rate which will give continuous cutter motion is:

(2500/33867)*60/0.0375 = 110 inches per minute.

11.4 Acceleration in Rapid Moves

Rapid moves have a slightly different operation when continuous cutter mode is active. Acceleration for the next motion is started when the axes being moved are all within the "In Position Limit" Parameters 101, 102, 103, and 104. These parameters have units of encoder steps. Rapid moves will also decelerate at the constant accel/decel limit until the speed drops below that for exponential accel/decel (see example above giving 159 inches per minute). An example of the "In position limit" values follows. If Parameter 101 (for **X**) is 2000 and Parameter 5 is 33867, a rapid move of will proceed to the next block when the X-axis is within a distance of:

1000 / 33867 = 0.0295 Inches

To prevent the rounding of corners, you can specify exact stop either with G09 (non-modal) or with G61 (modal). When either of these is active in a motion, all of the axes are brought to an exact stop, at zero speed, before the next motion is started.

Note that in this machine, using the default parameter settings, rapid and feed moves will both be blended to provide continuous cutter path and rounded corners. Unless you specify exact stop, the following rapid or feed block will be started slightly before the completion of the previous block.

11.5 Acceleration/Deceleration in Circular Moves

The tool path in a circular move (G02 or G03) is not changed by the exponential acceleration/deceleration so that there is no error introduced in the radius of the cut unless the speed exceeds that for exponential accel/decel (see example above giving 100 inches per minute). However, the actual radius of a circular move will always be slightly smaller than the programmed value. The amount of change can be computed by the following equation:

Ra = SQRT(R*R - L*L)

Where **Ra** is the actual radius,
R is the programmed radius, and
L is the accel/decel lag in feed motion.

The lag amount is computed by:

L = (Par. 113) * (feed in/min) / 600000

As an example; if Par 113 is 375 (0.0375 sec) and the feed is 30 inches per minute and the programmed diameter is two inches, the actual radius will be:

L = 375 * 30 / 600000 = 0.0187 inches
and
Ra = SQRT(1-0.000351) = 0.999824

or an error of 176 millionth's of an inch. This is an upper bound on the accuracy of this cut and many other factors could contribute additional errors.

Note that in this machine, using the default parameter settings, rapid and feed moves will both be blended to provide continuous cutter path and rounded corners. Unless you specify exact stop, the following rapid or feed block will be started slightly before the completion of the previous block.

11.6 Fanuc 6M, 10M, and 15M Compatibility

Parameter 57 may be used to change the rapid accel/decel mode to one closer to that of the 10M and 15M controls. This is done with the flag called "EX ST MD CHG". This means "exact stop in mode change" and, if this flag is set to 1, will cause an exact stop at both the beginning and end of any rapid move. Thus continuous cutter motion is provided only for a feed motion followed by another feed motion. When this flag is set, the exact stop codes G09 and G61 will still provide an exact stop between two feed motions.

Setting 33 controls how the G52 and G92 codes work. These are different between Fanuc class controls and Yasnac class controls. To operate like a Fanuc control, Setting 33 should be set to FANUC.

Setting 58 controls how cutter compensation goes around outside corners. This motion is different between Fanuc class controls and Yasnac class controls. To operate like a Fanuc control, Setting 58 should be set to FANUC.

Note: the Haas CNC Control is **compatible** with many other controls; it is not **identical** in performance to any single control.

12. High Speed Machining

High speed machining in the VF Series CNC Mill is those operations which require more than 100 inches per minute feed rate or which require more than 20 blocks per second execution rate. In order to achieve these rates, the control is provided with a function that turns off the displays and provides more execution time for servo control.

M76 is used to turn off the displays and enable high speed machining; M77 is used to turn the displays back on at the end of high speed machining.

This function is typically needed in applications that have a high feed rate and relatively short travel per block. If the feed rate were 100 inches per minute and the block lengths were as short as 0.018 inches, the number of blocks per second would be 93. This would require the high speed machining function. A similar example would be machining at 30 inches per minute with blocks of 0.005 inches; this would require 100 blocks per second.

When M76 is active, the maximum number of blocks per second is more than 150 and the maximum feed rate is 200 inches per minute.

13. Subroutines

One of the more important programming features of a CNC is called subroutines. Subroutines allow the CNC programmer to define a series of commands which might be repeated several times in a program and, instead of repeating them many times, they can be "called". A subroutine call is done with M97 or M98 and a **Pnnnn**. The **P** code is the same as the **O** number of the subroutine to be called.

It is important to note that there is little difference between the main program and the subroutines. In the LIST PROG display, they all appear as numbered programs. When starting execution of a program, the LIST PROG display is used to select the MAIN program and any subroutines used are called from within the main program.

Local subroutines can be used with the M97. This can be even easier to use than the M98 because the subroutine is part of a single main program without the need to define a different **Onnnn** program. With local subroutines, you can code an M30 for the end of your main program followed by a line number and a subroutine that ends with an M99.

The subroutine call causes the blocks in the subroutine to be executed just as if they were included in the main program. In order to return control to the main program, subroutines must end with an M99.

Another very important feature of subroutines is that the M98 "call" block may also include an **L** or repeat count. If there is an **L**, the subroutine call is repeated that number of times before the main program continues with the next block.

The most common use of subroutines is in the definition of a series of holes which must be first center drilled, peck drilled, tapped, and chamfered. If a subroutine is defined that consists only of the X-Y position of the holes, the main program can call that subroutine after defining a canned cycle to do each of the operations. Thus, the X-Y positions can be used several times and need not be repeated for each tool. An example follows:

```
O0100 (MAIN PROGRAM FOR EXAMPLE OF SUBROUTINES) ;
G54 G00 G90 X0. Y0. ;
T01 M06 (CENTER DRILL) ;
G81 R0.2 Z-0.1 F20. L0 (NO OPERATION HERE, JUST DEFINE CANNED CYCLE) ;
S2000 M03 ;
M98 P0200 (CENTER DRILL EACH HOLE) ;
T02 M06 (PECK DRILL) ;
G83 R0.2 Z-1. F10. L0 (NO OPERATION HERE, JUST DEFINE CANNED CYCLE) ;
S1000 M03 ;
M98 P0200 (PECK DRILL EACH HOLE) ;
T03 M06 (TAP IN FLOATING HOLDER OR HARD TAP) ;
G84 R0.2 Z-1. F10. L0 (NO OPERATION HERE, JUST DEFINE CANNED CYCLE) ;
S200 (1/4-20) ;
M98 P0200 (TAP EACH HOLE) ;
T04 M06 (CHAMFER) ;
G81 R0.2 Z-0.1 F20. L0 (NO OPERATION HERE, JUST DEFINE CANNED CYCLE) ;
S2000 M03 ;
M98 P0200 (CHAMFER EACH HOLE) ;
G28 M30 (END OF MAIN PROGRAM) ;
```

```
O0200 (SUBROUTINE EXAMPLE LISTING ALL HOLE POSITIONS) ;
X0. Y0. ;
X1. Y0. ;
X2. Y0. ;
X0. Y1. ;
X1. Y1. ;
X2. Y1. ;
X0. Y2. ;
X1. Y2. ;
X2. Y2. ;
M99 (END OF SUBROUTINE) ;

O0300 (EXAMPLE USING A LOCAL SUBROUTINE)
G54 G00 G90 X0. Y0.;
G81 R0.2 Z-0.1 F20 L0 (NO OPERATION HERE, JUST DEFINE CANNED CYCLE);
S2000 M03;
M97 P0500 (CENTER DRILL EACH HOLE);
T02 M06 (PECK DRILL);
G83 R0.2 Z-1. F10. L0 (NO OPERATION HERE, JUST DEFINE CANNED CYCLE);
S1000 M03;
M97 P0500 (PECK DRILL EACH HOLE);
G28 M30 (END OF MAIN PROGRAM);

N0500 (LOCAL SUBROUTINE EXAMPLE LISTING ALL HOLE POSITIONS);
X0. Y0,;
X1. Y0.;
X2. Y0.
X0. Y1.
X1. Y1.
X2. Y1.
X0. Y2.
X1. Y2.
X2. Y2.
M99 (END OF SUBROUTINE);
```

14. Macros

14.1 Introduction

This is an introduction to macros as implemented on the HAAS CNC controls. MACROS adds capabilities and flexibility to standard G-code programming that allow the programmer to better define a tool path in a quicker and more natural way. With few exceptions, MACROS, as implemented on the HAAS controls, is compatible with FANUC 10M and 15M controls. Macro features not included in the current release are listed at the end of the manual. Programmers already familiar with macro programming will want to review this section in order to avoid unnecessary work.

In traditional CNC programming, a program consists of subroutines that CANNOT be changed or altered except by editing individual values with an editor. MACROS allows the capability to program subroutines where the tool path or location of the tool path is changed, depending on the values contained within variables set by the programmer. These variables can be passed to the subroutine as parameters, or the values can reside in what are called *global variables.*

What this all means is that a programmer can create a collection of subroutines that have been fully debugged. These programs can be used as high level tools that can enhance programmer and machinist productivity. MACROS is not intended to replace modern CAD/CAM software, but it can and has improved machine productivity for those who use it.

Here are a few examples of the applications for MACROS. Rather than give macro code here, we will outline the general applications that MACROS can be used for.

• Tools For Immediate, On-Table Fixturing

Many setup procedures can be semi-automated to assist the machinist. Tools can be reserved for immediate situations that were not anticipated during tool design. For instance, suppose a company uses a standard clamp with a standard bolt hole pattern. If it is discovered, after setup, that a fixture will need an additional clamp and if macro subroutine 2000 has been programmed for drilling the bolt pattern of the clamp, then the following two-step procedure is all that is needed for adding the clamp to the fixture.

1) Determine X, Y, and Z coordinates and angle where the clamp is to be placed by jogging the machine to the proposed clamp position and reading the position coordinates from the machine display.

2) Execute the following command in MDI mode.

G65 P2000 X??? Y??? Z??? A??? ;

where **???** are the values determined in step 1.

Here, macro 2000 (not shown) takes care of all the work since it was designed to drill the clamp bolt hole pattern at the specified angle of **A**. Essentially, the machinist has created his own custom canned cycle.

• Simple Patterns That Are Repeated Over And Over Again In The Shop

Patterns that recur over and over again can be parameterized and kept around for easy, immediate use. For example:

1) Bolt hole patterns.
2) Slotting.
3) Angular patterns, 5 holes at 30 degrees 1 inch apart.
4) Specialty milling such as soft jaws.
5) Matrix Patterns, 12 across and 15 down.
6) Flycutting a surface, 12 inches by 5 inches using a 3 inch fly cutter.

• Automatic Offset Setting Based On The Program

With macros, coordinate offsets can be set in each program so that setup procedures become easier and less error-prone.

• Probing

Probing enhances the capabilities of the machine in many ways. Below is just a hint of the possibilities.

1) Profiling of a part to determine unknown dimensions for later machining.
2) Tool calibration for offset and wear values.
3) Inspection prior to machining to determine material allowance on castings.
4) Inspection after machining to determine parallelism and flatness values as well as location.

Macros allow less experienced personnel to operate the machine. Conditions can be detected and custom operator messages or alarms can be displayed on the console to notify the operator.

14.2 Macro Subroutine Call (G65)

G65 is the command that calls a subroutine with the ability to pass arguments to it. The format follows.

[N#####] G65 P##### [L####] [arguments] ;

Anything enclosed in brackets is optional. This should not be confused with expression brackets that are explained below. The G65 command requires a **P** address parameter corresponding to any program number currently in memory. When the optional **L** address is used the macro call is repeated the specified number of times.

In Example 1, subroutine 1000 is called once with no parameters passed to the routine. G65 calls are similar to, but not the same as, M98 calls. Up to four G65 calls can be made at the same time, (Nesting four deep).

```
Example 1:  G65 P1000 ;  (Call subroutine 1000 as a macro)
    M30 ;        (Program stop)

    O1000 ;      (Macro Subroutine)
      ...
    M99 ;        (Return from Macro Subroutine)
```

In Example 2, subroutine 9010 is designed to drill a sequence of holes along a line whose slope is determined by the X and Y arguments that are passed to it in the G65 command line. The Z drill depth is passed as **Z**, the feed rate is passed as **F**, and the number of holes to be drilled is passed as **T**. The line of holes is drilled starting from the current tool position when the macro subroutine is called.

Example 2:

```
G00 G90 X1.0 Y1.0 Z.05 ;          (Position tool)
G65 P9010 X.5 Y.25 Z.05 F10. T10 ; (Call 9010)
G28 M30 ;
O9010 ;                           (Diagonal hole pattern)
F#9 ;                             (F=Feed rate)
WHILE [#20>0] DO1                 (Repeat T times)
G91 G81 Z#26 ;                    (Drill To Z depth)
#20=#20-1 ;                       (Decrement counter)
IF [#20 EQ 0] GOTO5 ;             (All holes drilled)
G00 X#24 Y#25 ;                   (Move along slope)
N5 END1 ;
M99 ;                             (Return to caller)
```

14.3 Aliasing

Aliasing is a means of assigning a G code to a G65 P##### sequence. For instance, in Example 2 it would be easier if one could write:

G93 X.5 Y.25 Z.05 F10. T10 ;

Here, we have substituted G93, an unused G code, for G65 P9010. In order for the above block to work we must set the parameter associated with subroutine 9010 to 93.

Program numbers 9010 through 9019 are reserved for G code aliasing. The following table lists which HAAS parameters are reserved for macro subroutine aliasing.

HAAS Parameter	O Code
91	9010
92	9011
93	9012
94	9013
95	9014
96	9015
97	9016
98	9017
99	9018
100	9019

G00, G65, G66, and G67 can not be aliased. All other codes between 1 and 255 can be used for aliasing.

Setting an aliasing parameter to 0 disables aliasing for the associated subroutine. If an aliasing parameter is set to a G-code and the associated subroutine is not in memory, then an alarm will be given.

14.4 Macro Arguments

The arguments in a G65 statement are a means of sending values to and setting the local variables of a called macro subroutine.

In Example 2 above, the arguments X and Y are passed to the macro subroutine local variables. Local variable #24 is associated with X and is set to 0.5. Similarly, Local variable #25 is associated with Y and is set to 0.25.

The following two tables indicate the mapping of the alphabetic address variables to the numeric variables used in a macro subroutine.

Alphabetic addressing.

Address:	A	B	C	D	E	F	G	H	I	J	K
Variable:	1	2	3	7	8	9	-	11	4	5	6
Address:	L	M	N	O	P	Q	R	S	T	U	V
Variable:	-	13	-	-	-	17	18	19	20	21	22
Address:	W	X	Y	Z							
Variable:	23	24	25	26							

Alternate Alphabetic addressing.

Address:	A	B	C	I	J	K	I	J	K	I	J
Variable:	1	2	3	4	5	6	7	8	9	10	11
Address:	K	I	J	K	I	J	K	I	J	K	I
Variable:	12	13	14	15	16	17	18	19	20	21	22
Address:	J	K	I	J	K	I	J	K	I	J	K
Variable:	23	24	25	26	27	28	29	30	31	32	33

Arguments accept any floating point value to four decimal places. If you are in metric, the control will assume thousandths (.000). In Example 3 below, local variable #7 will receive .0004.

If a decimal is <u>not</u> included in an argument value, such as:

G65 P9910 A1 B2 C3

The values are passed to macro subroutines according to the following table:

Integer Argument Passing (no decimal point)

Address:	A	B	C	D	E	F	G	H	I	J	K
Variable:	.001	.001	.001	1.	1.	1.	-	1.	.0001	.0001	.0001
Address:	L	M	N	O	P	Q	R	S	T	U	V
Variable:	1.	1.	-	-	-	.0001	.0001	1.	1.	.0001	.0001
Address:	W	X	Y	Z							
Variable:	.0001	.0001	.0001	.0001							

All 33 local macro variables can be assigned values with arguments by using the alternate addressing method. The following example shows how one could send two sets of coordinate locations to a macro subroutine. Local variables #4 through #9 would be set to .0001 through .0006 respectively.

Example 3: G65 P2000 I1 J2 K3 I4 J5 K6 ;

The following letters cannot be used to pass parameters to a macro subroutine: G, L, N, O or P.

14.5 Macro Constants

Constants are floating point values placed in a macro expression. They can be combined with addresses A...Z or they can stand alone when used within an expression. Examples of constants are .0001, 5.3 or -10.

14.6 Macro Variables

There are three categories of macro variables: *system* variables, *global* variables, and *local* variables.

• Variable Usage

All variables are referenced with a number sign (#) followed by a positive number. Examples are: #1, #101, and #501.

Variables are decimal values that are represented as floating point numbers. If a variable has never been used, it can take on a special "undefined" value. This indicates that it has not been used. A variable can be set to undefined with the special variable #0. #0 has the value of undefined or 0.0 depending on the context it is used in. More about this later. Indirect references to variables can be accomplished by enclosing the variable number in brackets.

#[<expression>]

The expression is evaluated and the result becomes the variable accessed. For example:

#1=3;
#[#1]=3.5 + #1;

This sets the variable #3 to the value 6.5.

Variables can be used in place of G-code address constants where "address" refers to the letters A...Z.
The block

N1 G0 G90 X1.0 Y0 ;

can be replaced by,

N1 G#7 G#11 X#1 Y#2 ;

providing that the variables take on the values:

#7=0;
#11=90;
#1=1.0;
#2=0.0;

Here, the values in the variables at runtime are used as the address values.

• Local Variables

Local variables range between #1 and #33. A set of local variables is available at all times. When a call to a subroutine with a G65 command is executed, the local variables are saved and a new set is available for use. This is called "nesting" of the local variables. During a G65 call, all of the new local variables are cleared to undefined values and any local variables that have corre-

sponding address variables in the G65 line are set to the G65 line values. Below is a table of the local variables along with the address variable arguments that change them.

Local Variables and corresponding address.

Variable:	1	2	3	4	5	6	7	8	9	10	11
Address:	A	B	C	I	J	K	D	E	F	H	
Alternate:						I	J	K		I	J
Variable:	12	13	14	15	16	17	18	19	20	21	22
Address:		M				Q	R	S	T	U	V
Alternate:	K	I	J	K	I	J	K	I	J	K	I
Variable:	23	24	25	26	27	28	29	30	31	32	33
Address:	W	X	Y	Z							
Alternate:	J	K	I	J	K	I	J	K	I	J	K

Note that variables 10, 12, 14...16 and 27...33 do not have corresponding address arguments. They can be set if a sufficient number of I, J and K arguments are used as indicated above in the section about arguments.

Once in the macro subroutine, the local variables can be read and modified by referencing the variable numbers 1...33.

When the **L** argument is used to do multiple repetitions of a macro subroutine, the arguments are set only on the first repetition. This means that if local variables 1...33 are modified in the first repetition, then the next repetition will have access only to the modified values. Local values are retained from repetition to repetition when the **L** address is greater than 1.

Calling a subroutine via an M98 does not nest the local variables. Any local variables referenced in a subroutine called by an M98 are the same variables and values that existed prior to the M98 call.

• Global Variables

Global variables are variables that are accessible at all times. There is only one copy of each global variable. Global variables occur in two ranges: 100...199 and 500...599. The global variables remain in memory when power is turned off. They are not cleared as in the FANUC controls.

• System Variables

System variables give the programmer the ability to interact with a variety of control parameters and settings. By setting a system variable, the function of the control can be modified or altered. By reading a system variable, a program can modify its behavior based on the value in the variable. Some system variables have a READ ONLY status. This means that they can not be modified by the programmer. A brief table of currently implemented system variables follows with an explanation of their use.

VARIABLES	USAGE
#1000-#1031	32 x 1-BIT DISCRETE INPUTS
#2000-#2999	TOOL OFFSETS
#3000	PROGRAMMABLE ALARM WITH MESSAGE
#3001	MILLISECOND TIMER
#3002	HOUR TIMER
#3006	PROGRAMMABLE STOP WITH MESSAGE

#4001-#4021	LAST BLOCK GROUP CODES
#4101-#4126	LAST BLOCK ADDRESS DATA
#5001-#5004	LAST BLOCK TARGET POSITION
#5021-#5024	CURRENT MACHINE COORD POSITION
#5041-#5044	CURRENT WORK COORD POSITION
#5061-#5064	CURRENT SKIP SIGNAL POSITION
#5081-#5084	TOOL LENGTH COMPENSATION
#5221-#5224	G54 OFFSET VALUES
#5241-#5244	G55 " "
#5261-#5264	G56 " "
#5281-#5284	G57 " "
#5301-#5304	G58 " "
#5321-#5324	G59 " "
#7001-#7004	G110 ADDITIONAL OFFSET VALUES
#7021-#7024	G111 " " "
#7381-#7384	G129 " " "

14.7 System Variables In-Depth

This section fully describes system variables.

• 1-Bit Discrete Inputs

For a complete description of discrete inputs, refer to the Service Manual. Inputs designated as "spare" can be connected to external devices and used by the programmer.

#1000-#1020	Reserved for HAAS Controller use.
#1021	Spare
#1022	Spare
#1023	Spare
#1024-#1028	Reserved for HAAS Controller use.
#1029	Skip signal
#1030	Spare
#1031	Spare

• 1-Bit Discrete Outputs

#1124 Some HAAS controls have eight discrete outputs designated as SPARE. These extra relays usually must be specially-ordered. The user can actuate these spare output relays by reading or writing to variables #1124-#1131. An assignment of "1" sets the relay, whereas a reference reads the relay. For example:

```
#1124=1;                                  (Turns #1124 relay on)
#101=#3001+1000;                          (101 is second from now)
WHILE #101 GT #3001 AND #1125 EQ 03 D01
END1                                      (Wait here 1 second or until relay #1125 goes high)
#1124=0;                                  (Turns #1124 relay off)
```

• Tool Offsets

HAAS macros have been implemented with FANUC control memory C option in mind. This means that each tool offset has a length (H) and radius (D) along with associated wear values.

#2001-#2050	H geometry offsets (1-50) for length.
#2200-#2250	H geometry wear (1-50) for length.
#2401-#2450	D geometry offsets (1-50) for diameter.
#2601-#2650	D geometry wear (1-50) for diameter.

• Programmable Messages

#3000 ALARMS can be programmed. A programmable alarm will act just like HAAS internal alarms. An alarm is generated by setting the macro variable #3000 to a number between 1 and 999.

#3000= 15 (MESSAGE PLACED INTO ALARM LIST) ;

When this is done, ALARM flashes in the lower right hand corner of the display and the text in the next comment is placed into the alarm list. The alarm number (in this example, 15) is added to 1000 and used as an alarm number. If an alarm is generated in this manner all motion stops and the program must be reset to continue. Programmable alarms can always be identified in alarm history because the alarm numbers range between 1000 and 1999.

The first 34 characters of the comment will be used for the alarm message.

• Timers

HAAS macros supports access to two timers. These timers can be set to a value by assigning a number to the respective variable. A program can then later read the variable and determine the time passed since the timer was set. Timers can be used to emulate dwell cycles, determine part to part time or wherever time dependent behavior is desired.

#3001 MILLISECOND TIMER - The millisecond timer is updated every 20 milliseconds and thus activities can be timed with an accuracy of only 20 milliseconds. At POWER ON, the millisecond timer is reset. The timer has a limit of 497 days. The whole number returned after accessing #3001 represents the number of milliseconds.

#3002 HOUR TIMER - The hour timer is similar to the millisecond timer except that the number returned after accessing #3002 is in hours. The hour and millisecond timers are independent of each other and can be set separately.

• System Overrides

#3004 Variable #3004 is a bitmapped variable that overrides specific control features during runtime.

The first bit disallows FEED HOLD from the keypad. If you do not want feed hold to be executed during any section of code, then bracket that code with assignments to variable #3004. Assigning "1" to #3004 disables the console's feed hold button. Assigning "0" to #3004 re-enables the FEED HOLD button. For example:

Approach code	(FEED HOLD allowed)
#3004=1;	(Disables FEED HOLD button)
Non-stoppable code	(FEED HOLD not allowed)
#3004=0;	(Enables FEED HOLD button)
Depart code	(FEED HOLD allowed)

The following is a map of variable #3004 bits and the associated overrides. E=Enabled D=Disabled

#3004	MAP	FEED HOLD
0	000	E
1	001	D
2	010	E
3	011	D
4	100	E
5	101	D
6	110	E
7	111	D

• Programmable Stop

#3006 Stops can be programmed. A programmable stop acts like an M00. In the following example, when the assignment statement is executed, the first 15 characters of the comment are displayed in the messaging area on the lower left part of the screen above the command input line. The control stops and waits for a cycle start from the operator. Upon cycle start, operation continues with the next block after the assignment statement.

F [#1 EQ #0] THEN #3006=101(ARG.A REQUIRED);

In addition to displaying a message and stopping, the first 34 characters of the comment will be placed on the last line of the operator notes page. Lines 2 through 16 of operator notes are scrolled up and the first line is lost. Important information should not be placed in operator notes if programmable stops are to be used.

• Last Block (MODAL) Group Codes

#4001-#4021 The grouping of G codes permits more efficient processing. G codes with similar functions are usually under the same group. For instance, G90 and G91 are under group 3. Variables have been set aside to store the last or default G code issued for any of 21 groups. By reading the group code, a macro program can change its behavior based on the contents of the group code. If 4003 contains 91, then a macro program could determine that all moves should be incremental rather than absolute. There is no associated variable for group zero, group zero G codes are NON-modal.

• Last Block (MODAL) Address Data

#4101-#4126 Address codes A...Z (excluding G) are also maintained as modal values. The modal information represented by the last block interpreted by the lookahead process is contained in variables 4101 through 4126. The numeric mapping of variable numbers to alphabetic addresses corresponds to the mapping under alphabetic addresses. For instance, the value of the previously interpreted **D** address is found in #4107 and the last interpreted **J** value is #4104.

• Last Target Position

#5001-#5004 The final programmed point, target position, for the most recent motion block can be accessed through variables #5001-#5004, X, Y, Z and A respectively. Values are given in the current work coordinate system and can be used while the machine is in motion.

• Current Machine Coord Position

#5021-#5024 The current position in machine coordinates can be obtained through #5021-#5024, X, Y, Z, and A respectively. The values CANNOT be read while the machine is in motion. #5023 (Z) represents the value after tool length compensation has been applied.

• Current Work Coord Position

#5041-#5044 The current position in the current work coordinates can be obtained through #5041, 5044, X, Y, Z and A respectively. The values can NOT be read while the machine is in motion. #5043 (Z) represents the value after tool length compensation has been applied.

• Current Skip Signal Position

#5061-#5064 The position where the last skip signal was triggered can be obtained through #5061 #5064, X, Y, Z and A respectively. Values are given in the current work coordinate system and can be used while the machine is in motion. #5063 (Z) represents the value after tool length compensation has been applied.

• Tool Length Compensation

#5081-#5084 The current total tool length compensation that is being applied to the tool is returned. This includes tool length offset referenced by the current modal value set in H (#4008) plus the wear value.

• Offsets

All tool work offsets can be read and set within a macro expression. This allows the programmer to preset coordinates to approximate locations, or to set coordinates to values based upon the results of skip signal locations and calculations.

#5201-#5204	G52 X, Y, Z, A OFFSET VALUES
#5221-#5224	G54 " " " " " "
#5241-#5244	G55 " " " " " "
#5261-#5264	G56 " " " " " "
#5281-#5284	G57 " " " " " "
#5301-#5304	G58 " " " " " "
#5321-#5324	G59 " " " " " "
#7001-#7004	ADDITIONAL X, Y, Z, A OFFSET VALUES
#7021-#7024	" " " " " " "
"	" " " " " " "
#7941-#7944	ADDITIONAL X, Y, Z, A OFFSET VALUES

14.8 Address Constant Substitution

The usual method of setting control addresses A...Z is by appending a constant to the address. For instance,

G01 X1.5 Y3.7 F20. ;

sets addresses G, X, Y and F to 1, 1.5, 3.7 and 20.0 respectively and thus instructs the control to move linearly, G01, to position X=1.5 Y=3.7 at a feed rate of 20 inches per minute. Macro syntax allows the constants to be replaced with any variable or expression in any section of code (i.e., you do not have to be in a macro subroutine).

The previous statement can be replaced by the following code:

```
#1=1;
#2=.5;
#3=3.7;
#4=20;
G#1 X[#1+#2] Y#3 F#4 ;
```

The permissible syntax on addresses A...Z (exclude N or O) is as follows:

<address><-><variable>	A-#101
<address>[<expression>]	Y[#5041+3.5]
<address><->[<expression>]	Z-[SIN[#1]]

If the value of the variable does not agree with the range of the address, then the usual control alarm will result. For instance, the following code would result in a range error alarm because tool diameter numbers range from 0...50.

```
#1=75;
D#1;
```

When a variable or expression is used in place of an address constant, then the floating point value is rounded to the least significant digit. If #1=.123456, then G1X#1 would move the machine tool to .1235 on the X axis. If the control is in the metric mode, the machine would be moved to .123 on the X axis.

When an UNDEFINED variable is used to replace an address constant, then that address reference is ignored. For example, if #1 is undefined then the block

G00 X1.0 Y#1 ;

becomes

G00 X1.0.

No Y movement takes place.

14.9 Macro Statements

Macro statements are lines of code that allow the programmer to manipulate the control with features similar to any standard programming language. Included are functions, operators, conditional and arithmetic expressions, assignment statements, and control statements.

Functions and operators are used in expressions to modify variables or values. The operators are essential to expressions while functions make the programmer's job easier.

A Functions

Functions are built-in routines that the programmer has available to use. All functions have the form **<function_name>[argument]**. Functions can be passed any expression as arguments. Functions return floating point decimal values. The function provided with the HAAS control are as follows:

FUNCTION	ARGUMENT	RETURNS	NOTES
SIN[ ]	Degrees	Decimal	Sine
COS[ ]	Degrees	Decimal	Cosine
TAN[ ]	Degrees	Decimal	Tangent
ATAN[ ]	Decimal	Degrees	Arctangent Same as FANUC ATAN[ ]/[1]
SQRT[ ]	Decimal	Decimal	Square root
ABS[ ]	Decimal	Decimal	Absolute value
ROUND[ ]	Decimal	Decimal	Round off a decimal
FIX[ ]	Decimal	Integer	Truncate fraction
ACOS[ ]	Degrees	Decimal	Arccosine
ASIN[ ]	Degrees	Decimal	Arcsine
#[ ]	Integer	Integer	Variable Indirection
DPRNT[ ]	ASCII text		External Output

• Notes on Functions

The function round works differently depending on the context that it is used. When used in arithmetic expressions, the round function works as one would expect. That is, any number with a fractional part greater than or equal to .5 is rounded up to the next whole integer; otherwise, the fractional part is truncated from the number.

```
#1= 1.714 ;
#2= ROUND[#1] ;    (#2 is set to 2.0)
#1= 3.1416 ;
#2= ROUND[#1] ;    (#2 is set to 3.0)
```

When round is used in an address expression, then the argument of round is rounded to the addresses significant precision. For *metric* and *angle* dimensions, three-place precision is the default. For *inch*, four-place precision is the default. Integral addresses such as D, T and H are rounded normally.

```
#1= 1.00333 ;
G0 X[ #1 + #1 ] ;
     (Table moves to 1.0067) ;
G0 X[ ROUND[ #1 ] + ROUND[ #1 ] ] ;
     (Table moves to 1.0066) ;
G0 A[ #1 + #1 ] ;
     (Axis moves to 1.007) ;
G0 A[ ROUND[ #1 ] + ROUND[ #1 ] ] ;
     (Axis moves to 1.006) ;
D[1.67]      (Diameter 2 is made current) ;
```

B Operators

Operators can be classified into three categories: Arithmetic operators, logical operators and Boolean operators.

• Arithmetic Operators

Arithmetic operators consist of the usual unary and binary operators. They are:

```
  +      - Unary plus   +1.23
  -      - Unary minus -[COS[30]]
  +      - Binary addition      #1=#1+5
  -      - Binary subtraction   #1=#1-1
  *      - Multiplication       #1=#2*#3
  /      - Division      #1=#2/4
MOD      - Remainder   #1=27 MOD 20
                (#1 contains 7)
```

• Logical Operators

Logical operators are operators that work on binary bit values. Macro variables are floating point numbers. When logical operators are used on macro variables, only the integer portion of the floating point number is used. The logical operators are:

OR - logically OR two values together
XOR - Exclusively OR two values together
AND - Logically AND two values together

Examples:

#1=1.0;	0000 0001	Here the variable #3
#2=2.0;	0000 0010	will contain 3.0 after
#3=#1 OR #2	0000 0011	the OR operation.
#1=5.0;		Here control will
#2=3.0;		transfer to block 1
IF [#1 GT 3.0 AND #2 LT 10]		GOTO1 because #1 GT 3.0 evaluates to 1.0 and #2 LT 10 evaluates to 1.0, thus 1.0 AND 1.0 is 1.0 (TRUE) and the GOTO occurs.

As can be seen from the previous examples, CARE must be taken when using logical operators so that the desired result is achieved.

• Boolean Operators

Boolean operators always evaluate to 1.0 (TRUE) or 0.0 (FALSE). There are six Boolean operators. These operators are not restricted to conditional expressions, but they most often are used in conditional expressions. They are:

EQ - Equal to
NE - Not Equal to
GT - Greater Than
LT - Less Than
GE - Greater than or Equal to
LE - Less Than or Equal to

The following are examples of how Boolean and Logical operators can be used:

Example

IF [#1 EQ 0.0] GOTO100;

WHILE [#101 LT 10] DO1;
#1=[1.0 LT 5.0];

IF [#1 AND #2 EQ #3] GOTO1 Description

Jump to block 100 if variable #1 equals 0.0.

While variable #101 is less than 10 then repeat loop DO1...END1.
Variable #1 is set to 1.0 (TRUE).

If variable #1 logically ANDed with variable #2 is equal to the value in #3 then control is transferred to block 1.

C Expressions

Expressions are defined as any sequence of variables and operators surrounded by the square brackets "[" and "]". There are two uses for expressions: conditional expressions or arithmetic expressions. Conditional expressions return FALSE (0.0) or TRUE (any non zero) values. Arithmetic expressions use arithmetic operators along with functions to determine a value.

• Conditional Expressions

In the HAAS control, ALL expressions set a conditional value. The value is either 0.0 (FALSE) or the value is nonzero (TRUE). The context in which the expression is used determines if the expression is a conditional expression. Conditional expressions are used in the IF and WHILE statements and in the M99 command. Conditional expressions can make use of Boolean operators to help evaluate a TRUE or FALSE condition.

The M99 conditional construct is unique to the HAAS control. Without macros, M99 in the HAAS control has the ability to branch unconditionally to any line in the current subroutine by placing a P code on the same line. For example:

N50 M99 P10 ;

branches to line N10. It does not return control to the calling subroutine. With macros enabled, M99 can be used with a conditional expression to branch conditionally. To branch when variable #100 is less than 10 we could code the above line as follows.

N50 [#100 LT 10] M99 P10 ;

In this case, the branch occurs only when #100 is less than 10, otherwise processing continues with the next program line in sequence. In the above, the conditional M99 can be replaced with

N50 IF [#100 LT 10] GOTO10 ;.

• Arithmetic Expressions

An arithmetic expression is any expression using constants, variables, operators, or functions. An arithmetic expression returns a value. Arithmetic expressions are usually used in assignment statements, but are not restricted to them.

Examples of Arithmetic expressions: #101=#145*#30;
#1=#1+1;
X[#105+COS[#101]];
#[#2000+#13]=0;

D Assignment Statements

Assignment statements allow the programmer to modify variables. The format of the assignment statement is:
<expression>=<expression>.

The expression on the left of the equal sign must always refer to a macro variable, whether directly or indirectly. The following macro initializes a sequence of variables to any value. Here both direct and indirect assignments are used.

```
O0300 (Initialize an array of variables) ;
N1 IF [#2 NE #0] GOTO2  (B=base variable) ;
#3000=1(BASE VARIABLE NOT GIVEN) ;
N2 IF [#19 NE #0] GOTO3 (S=size of array);
#3000=2(SIZE OF ARRAY NOT GIVEN) ;
WHILE [#19 GT 0] DO1 ;
#19=#19-1  (DECREMENT COUNT) ;
#[#2+#19]=#22        (V=value to set array to) ;
END1 ;
M99 ;
```

The above macro could be used to initialize three sets of variables as follows.

```
G65 P300 B101. S20   (INIT 101...120 TO #0) ;
G65 P300 B501. S5 V1 (INIT 501...505 TO 1.0) ;
G65 P300 B550. S5 V0 (INIT 550...554 TO 0.0) ;
```

The decimal point in B101., etc. would be required.

E Control Statements

Control statements allow the programmer to branch, both conditionally and unconditionally. They also provide the ability to iterate a section of code based on a condition.

• Unconditional Branch (GOTOnnn and M99 Pnnnn)

In the HAAS control, there are two methods of branching unconditionally. An unconditional branch will always branch to a specified block. M99 P15 will branch unconditionally to block number 15. The M99 can be used whether or not macros is installed and is the traditional method for branching unconditionally in the HAAS control. GOTO15 does the same as M99 P15. In the HAAS control, a GOTO command can be used on the same line as other G coding. The GOTO is executed after any other control commands as are traditional M codes.

• Conditional Branch (IF and M99 Pnnnn)

Conditional branching allows the program to transfer control to another section of code within the same subroutine. Conditional branching can only be used when macros are enabled. The HAAS control allows two similar methods for accomplishing conditional branching.
IF [<conditional expression>] GOTOn

Here, as discussed above, <conditional expression> is any expression that uses the six Boolean operators EQ, NE, GT, LT, GE, or LE. The brackets surrounding the expression are mandatory. In the HAAS control, it is not necessary to include these operators. For example:

IF [#1 NE 0.0] GOTO5;

could also be:
IF [#1] GOTO5 ;

In this statement, if the variable #1 contains anything but 0.0, or the undefined value #0, then branching to block 5 will occur; otherwise, the next block will be executed. If portability to a control other than HAAS is desired, then it is recommended that the conditional operators be used.

In the HAAS control, a conditional expression can also be used with the M99 Pnnnn format, providing that macros have been enabled. An example is as follows.

G0 X0 Y0 [#1EQ#2] M99 P5;

Here, the conditional is for the M99 portion of the statement only. The machine tool is instructed to X0, Y0 whether or not the expression evaluates to TRUE or FALSE. Only the branch, M99, is executed based on the value of the expression. It is recommended that the IF GOTO version is used if portability is desired.

• Conditional Execution (IF THEN)

Execution of control statements can also be achieved by using the IF THEN construct. The format is:

IF [<conditional expression>] THEN <statement> ;

This format is traditionally used for conditional assignment statements such as:

IF [#590 GT 100] THEN #590=0.0 ;

Here, variable #590 is set to zero when the value of #590 exceeds 100.0. In the HAAS control, if a conditional evaluates to FALSE (0.0), then the remainder of the IF block is ignored. This means that control statements can also be conditioned so that we could write something like:
IF [#1 NE #0] THEN G1 X#24 Y#26 F#9 ;

This executes a linear motion only if variable #1 has been assigned a value. You might try something like this:

IF [#1 GE 180] THEN #101=0.0 M99 ;

This says that if variable #1 (address A) is greater than or equal to 180, then set variable #101 to zero and return from the subroutine.

Here is an example of an IF statement that branches if a variable has been initialized to contain any value. Otherwise, processing will continue and an alarm will be generated. Remember, when an alarm is generated, program execution is halted.

N1 IF [#9NE#0] GOTO3 (TEST FOR VALUE IN F) ;
N2 #3000=11(NO FEED RATE) ;
N3 (CONTINUE) ;

Iteration/Looping (WHILE DO END):

Essential to all programming languages is the ability to execute a sequence of statements a given number of times or to loop through a sequence of statements until a condition is met. Traditional G coding allows this with the use of the **L** address. A subroutine can be executed any number of times by using the **L** address.
M98 P2000 L5 ;

This is limited since you can not terminate execution of the subroutine on condition. Macros allows more flexibility with the WHILE-DO-END construct. The syntax is as follows:

```
WHILE [<conditional expression>] DOn ;
<statements> ;
ENDn ;
```

This executes the statements between DOn and ENDn as long as the conditional expression evaluates to TRUE. The brackets in the expression are necessary. If the expression evaluates to FALSE, then the block after ENDn is executed next. WHILE can be abbreviated to WH. The DOn-ENDn portion of the statement is a matched pair. The value of n is 1...3. This means that there can be no more than three nested loops per subroutine. A nest is basically a loop within a loop. A good example of how nesting of WHILE loops can be used is in defining a matrix.

```
#101= 3 ;
#102= 4 ;
G0 X#101 Y4. ;
F2.5 ;
WH [ #101 GT 0 ] DO1 ;
 #102= 4 ;
WH [ #102 GT 0 ] DO2 ;
G81 X#101 Y#102 Z-0.5 ;
#102= #102 - 1 ;
END2 ;
#101= #101 - 1 ;
END1 ;
;
M30 ;
```

The previous program drills a 3 x 4 matrix hole pattern.

Although nesting of WHILE statements can only be nested to three levels, there really is no limit since each subroutine can have up to three levels of nesting. If there ever is a need to nest to a level greater than 3, then the segment containing the three lowest levels of nesting can be made into a subroutine thus overcoming the limitation.

If two separate WHILE loops are in a subroutine, they can use the same nesting index. For example:

```
#3001=0 (WAIT 500 MILLISECONDS) ;
WH [#3001 LT 500] DO1 ;
END1 ;

<other statements>

#3001=0 (WAIT 300 MILLISECONDS) ;
WH [#3001 LT 150] DO1 ;
END1 ;
```

This is valid code.

You can use GOTO to jump out of a region encompassed by a DO-END, but you can not use a GOTO to jump into it. Jumping around inside a DO-END region using a GOTO is allowed.

An infinite loop can be executed by eliminating the WHILE and expression. Thus,

```
DO1 ;
<statements>
END1 ;
```

executes until the RESET key is pressed.

CAUTION! The following code can be confusing:

```
WH [ #1 ] D01 ;
END1 ;
```

In the above, an alarm results indicating no "then" was found, here "then" refers to the D01. Change D01 (zero) to DO1 (letter O).

14.10 Communication With External Devices - DPRNT[ ]

Macros allow additional capabilities to communicate with peripheral devices. One can do digitizing of parts, provide runtime inspection reports, or synchronize controls with user provided devices. The commands provided for this are POPEN, DPRNT[ ] and PCLOS.

• Communication preparatory commands

POPEN and PCLOS are not required on the HAAS mill. It has been included so that programs from different controls can be ported to the HAAS. On some controls POPEN is required prior to using a DPRNT statement. POPEN prepares the device on the serial port by sending it a DC2 code. PCLOS terminates communication with external devices by sending it a DC4 code.

• Formatted output

The DPRNT statement allows the programmer to send formatted text to the serial port. Any text and any variable can be printed to the serial port. Variables can be formatted. The form of the DPRNT statement is as follows.

DPRNT[<text> <#nnnn[wf]>...] ;

DPRNT must be the only command in the block. In the above, <text> is any character from A to Z or the letters (+,-,/,*, and the space). When an asterisk is output, it is converted to a space. The <#nnnn[wf]> is a variable followed by a format. The variable number can be any legal macro variable. The format [wf] is required and consists of two digits within square brackets. Remember that macro variables are real numbers with a whole part and a fractional part. The first digit in the format designates the total places reserved in the output for the whole part. The second digit designates the total places reserved for the fractional part. The total places reserved for output cannot be equal to zero or greater that eight. Thus the following formats are illegal:

[00] [54] [45] [36] /* not legal formats */

A decimal point is printed out between the whole part and the fractional part. The fractional part is rounded to the least significant place. When zero places are reserved for the fractional part, then no decimal point is printed out. Trailing zeros are printed as necessary if there is a fractional part. At least one place is reserved for the whole part, even when a zero is used there. If the value of the whole part has fewer digits than have been reserved, then leading spaces are output. If the value of the whole part has more digits than has been reserved, then the field is expanded so that these numbers are printed.

A carriage return is sent out after every DPRNT block.

• DPRNT[ ] Examples

```
#1= 1.5436 ;
DPRNT[X#1[44]*Z#1[03]*T#1[40]] ;
          outputs:    X1.5436 Z 1.544 T   1
```

```
DPRNT[***MEASURED*INSIDE*DIAMETER***] ;
   outputs:      MEASURED INSIDE DIAMETER

DPRNT[] ;
   outputs:      (no text, only a carriage return)

#1=123.456789 ;
DPRNT[X-#1[25]] ;
   outputs:      X-123.45679 ;
```

14.11 Runtime Execution

DPRNT statements are executed at block interpretation time. This means that the programmer must be careful about where the DPRNT statements appear in the program, particularly if the intent is to print out positional information. Generally, a program is interpreted many blocks ahead in order to prevent the machine from pausing between movements.

G103 is useful for limiting lookahead. If you wanted to limit lookahead interpretation to one block, you would include the following command at the beginning of your program: (This actually results in a two block lookahead.)

```
G103 P1 ;
```

To cancel the lookahead limit, then issue a G103 P0 ;. G103 can not be used when cutter compensation is active.

14.12 Operation Notes

This section explains the additional screens and operator actions that come with macros.

• Variable Display Page

The macro variables are displayed and can be modified through the current commands display. The variable display is located after the operation timers display. To get to this page, press CURNT COMNDS and use the page up/down key.

As the control interprets a program, the variable changes are displayed on the variable display page and results can be viewed.

Pages contain up to 32 variables and the display can be "paged" by pressing the left/right arrow keys.

Setting of a variable is accomplished by entering a value and then pressing the WRITE key. The variable that is highlighted on the screen is the variable that is affected.

Searching for a variable can be done by entering the variable number and pressing the up/down arrow. The page will change to the one that contains that variable and the entered variable will become the highlighted item.

The variables displayed represent the values of the variables at program interpretation time. At times, this may be up to 15 blocks ahead of the actual machine activity. Debugging of programs can be made easier by inserting a G103 at the beginning of a program to limit block buffering and then removing the G103 block after debugging is completed.

• Editing

For the most part, the editing of macro programs from the control is the same as before. There are a few peculiarities to be aware of.

Editing macro statements is more open than previously. For instance, it is possible to place a floating point constant within a standard G-code block, but it doesn't make much sense, and the control will raise an alarm at runtime. For all instances of improperly structured or improperly placed macro statements, the control will raise an appropriate alarm. Most of these alarms have been put off until runtime so that operator editing can be more flexible. Be careful when editing expressions. Brackets must be balanced and you will not receive an alarm until runtime.

The DPRNT[ ] function can be edited much like a comment. You can delete it or move it as a whole item, or you can edit individual items within the brackets. Variable references and format expressions must be altered as a whole entity. If you wanted to change [24] to [44], place the cursor so that [24] is highlighted, enter [44] and press the write key. Remember, you can use the crank handle to maneuver through long DPRNT[ ] expressions.

Addresses with expressions can be somewhat confusing. In this case, the alphabetic address stands alone. For instance, the following block contains an address expression in X:

```
G1 G90 X [ COS[ 90 ] ] Y3.0 (CORRECT) ;
```

Here, the **X** and brackets stand alone and are individually editable items. It is possible, through editing, to delete the entire expression and replace it with a floating point constant.

```
G1 G90 X 0 Y3.0 (!!! WRONG !!!) ;
```

The above block will result in an alarm at runtime. The correct form looks as follows:

```
G1 G90 X0 Y3.0 (CORRECT) ;
```

Note that the zero is attached to **X**. REMEMBER when you see an alpha character standing alone it is an address expression.

14.13 Fanuc-Style Macro Features Not Included In Haas CNC Control

This section lists the FANUC macro features that have not been implemented as of this release.

```
M ALIASING          REPLACE G65 Pnnnn WITH Mnn PROGS 9020-9029.
P ADDRESS IN G65    NEED TO ALLOW P ADDRESS
G66  MODAL          CALL IN EVERY MOTION BLOCK
G66.1 MODAL         CALL IN EVERY BLOCK
G67  MODAL CANCEL
M98    ALIASING, T CODE      PROG  9000, VAR #149, ENABLE BIT
M98    ALIASING, S CODE      PROG  9029, VAR #147, ENABLE BIT
M98    ALIASING, B CODE      PROG  9028, VAR #146, ENABLE BIT
SKIP   /N N=1...9

#3003  SINGLE BLOCK SUPPRESSION FLAG
#3004  FEED HOLD, OVERRIDE, AND EXACT STOP OVERRIDE.
#3007  MIRROR IMAGE ON FLAG, EACH AXIS
#3011  YEAR/MONTH/DAY
#3012  HOUR/MINUTE/SECOND
#3901          TOTAL NUMBER OF PARTS
```

#3902	REQUIRED NUMBER OF PARTS
#4201-#4320	CURRENT BLOCK MODAL DATA
#5101-#5106	CURRENT SERVO DEVIATION
ADDITIONAL OFFSETS	G54.1P## FORMAT
NAMES FOR VARIABLES	FOR DISPLAY PURPOSES
ATAN [ ]/[ ]	ARCTANGENT, FANUC VERSION
BIN [ ]	CONVERSION FROM BCD TO BIN
BCD [ ]	CONVERSION FROM BIN TO BCD
FUP [ ]	TRUNCATE FRACTION CEILING
LN [ ]	NATURAL LOGARITHM
EXP [ ]	BASE E EXPONENTIATION
ADP [ ]	RE-SCALE VAR TO WHOLE NUMBER
BPRNT [ ]	

GOTO VAR	GOTO #N GOTO BLOCK REF BY VAR
GOTO EXPRESSION	GOTO [EXPRESSION] CALCULATED GOTO

15. Tapping With The VF Series CNC Mill

Making tapped holes with the VF Series CNC Mill can be made with several devices. Threads may be generated with a tap held in a rigid tool holder (called rigid tapping), a floating tap holder, a reversing tapping head, or helical thread milling. Each method has distinct advantages.

Tapping is done using canned cycles. You must select the tapping RPM and, using the pitch (threads per inch), calculate the feed rate that is entered in the **F** command. The HELP/CALC page will compute these numbers for you.

15.1 Rigid Tapping

Rigid Tapping eliminates the cost of special tap holders since taps can be held in drill collet holders. The spindle is accurately synchronized with the Z-axis feed, thereby producing threads as accurately as a lead screw tapper. No side forces are generated on the flanks of the threads and tighter thread tolerances are produced. Rigid tapping also eliminates the pullout and distortion of the first thread that occurs on all spring compression/tension devices and tapping heads. While this is not usually a problem on medium to coarse threads, small diameter, fine pitch or soft material tapped holes can have their last thread damaged when the tap pops out of the hole. You can also re-tap a hole without cross-threading provided the tap and **Z** depths have not been changed. Rigid tapping is used with canned cycles G74 and G84 and is an extra cost factory installed option. Example:

N100 G84 Z-1. R.3 F37.5 (for a 20 pitch thread at 750 RPM)

A word of caution on Rigid Tapping: As the term implies, the tap is rigidly held in place. This requires that runout be less than .001 TIR or the tap will generate an oversized thread. This problem can be minimized by using a small diameter drill extension to provide flex, a radial floating tool holder, or specially-designed chucks for holding taps, because tap shanks are not common collet sizes.

15.2 Using Floating Tap Holders

Floating tap holders are probably the most common method of tapping holes. The tap is held in a quick change holder that can float up and down slightly. This is done to allow the tap to follow the hole it is tapping and compensate for differences in the accel and decel of the spindle versus the feed of the Z-axis. Upon reaching the bottom of the hole, the feed stops and the spindle reverses, if you watch closely, you will see the tap pull the floating holder out slightly. Upon reversal, the tap will be pushed back into the holder.

If the holder is pulled out or pushed in to it's mechanical limits while tapping, you can break the tap, damage the threaded part, or pull the tap completely out of the holder. Carefully watch for this condition when setting up a job because it usually becomes a problem after the job has been running a while. Also tapping of diameters less than 5/16 of an inch while below 1201 rpm (low gear shift point par 142) should be done in high gear. Spindle reversal is quicker in high gear and will minimize tap pullout. This is done by putting an M42 code with the speed command such as: M42 S900. Tapping is done with G74 and G84 cycles which automatically reverse the spindle at **Z** depth. The feed rate can be calculated by using the HELP display and paging down to the tapping calculator and inputting your speed and tap pitch into the control to obtain your feed rate that is then input to the **F** command of the cycle. Example:

N100 G84 Z-1.0 R.3 F46.875 (for 32 pitch tap at 1250 RPM)

15.3 Autoreversing Tapping Heads

Auto reversing tapping heads eliminate the need for the spindle to reverse at bottom and provide for high production rates. The reversing function of the tapping head requires an arm to prevent the body from rotating. This must be considered when changing tools so as not to interfere with operation. The tool block on the VF Series CNC Mill will accommodate the Tapmatic series of heads. Several sizes are available and should be chosen dependent on tap size. Choose the head specifically for NC use as they have a 1:1 feed rate. Manual types have a faster withdraw rate that leads to clatter on the up stroke. A disadvantage to these types of heads is that when the inevitable crash occurs, you can destroy an expensive device.

Use the G85 or G89 (dwells at bottom) cycle when using a tapping head. Example:

N100 G98 G85 Z-1.0 R0.25 F46.875

15.4 Thread Milling

Thread milling uses a cutter formed with the pitch of the thread to mill the thread. The cutters are solid carbide, fragile and expensive. Some companies sell replaceable insert holders that are more economical. Internal holes smaller than 3/8 inch may not be possible or practical. It does allow for making thread diameter compensation and external threads. For large threads, port threads, blind hole threads, thread milling can be the most economical method.

Thread milling is accomplished with Helical milling. Use a standard G02 or G03 move to create the circular move in X-Y and then insert a **Z** move on the same block corresponding to the thread pitch. The feed rate is selected as in standard milling practice. This will generate one turn of the thread. The multiple teeth of the cutter will generate the rest. A typical line would be as follows:

N100 G02 I-1.0 Z-.05 F5. (generates a one inch radius for 20 pitch thread)

INDEX

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Z