PART 3 OF MACHINE AND CONTROLLER

23. PROGRAM CONTROL INSTRUCTIONS

1) SKIP TO KEY
2) SUBROUTINE, SUB RETURN, CALL KEYS
3) REPEAT, REPEAT END KEY

EXAMPLES
1. DRILL PECK
2. X Y HOLE PATTERNS
3. RECT. FRAME
4. RECT. POCKETING
5. ARC POCKETING
6. CUTTING A POLYGON
7. CUTTING A CAM
8. Generation of a pyramid and cone
9. HELICAL THREAD CUTTING

4) PROG REF KEY

24. EXTERNAL CONTROL 1,2,3 CONTROL KEY

25. THE FUNCTION KEY ------ SCALE ON/OFF.

26. THE CANNED FUNCTIONS.

HOW THEY OPERATE

1. MILL KEY
2. RECT. FRAME KEY
3. RECT POCKET KEY
4. CIRCLE POCKET KEY
5. DRILL KEY
6. BOLT CIRCLE KEY
7. ARC FRAME KEY
8. USING PROG REF FOR THE CANNED FUNCTIONS

23. PROGRAM CONTROL INSTRUCTIONS

These instructions are used to 'control' the controller when it is running a program.

1) SKIP TO KEY

When entered it becomes SKIP TO nnn, where nnn is the line number. When the controller comes to this instruction it will jump to this nnn line number and continue running from there. It is useful in endlessly cycling the machine.

2) SUBROUTINE KEY, SUBROUTINE RETURN KEY AND CALL KEY

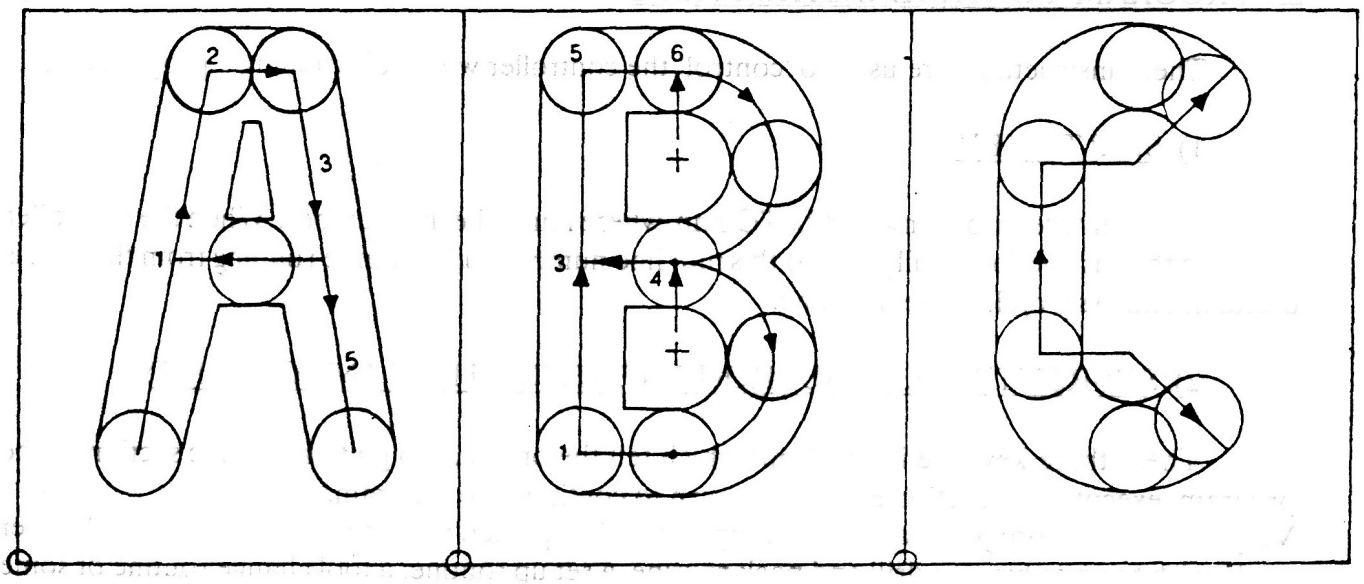
These three keys are instructions which tell the program to go to a specific section of the program, execute that section and then return to continue the program.
Very often in long programs a particular operation is repeated many times. For example, the user might wish to write his own drill and peck routine, a set up routine, a tool change routine or some particular geometry that is to be repeated elsewhere. It is a severe imposition to have to repeat these instructions throughout the program, hence subroutines :
This section is written as:

```
SUB  n n          (00 > n > 99)
- - - -
- - - -          OPERATION
- - - -

SUB RETURN
```

and can be put anywhere in the program space (000-900 line number).
Subroutines are usually placed at the end of the program space. When the user wishes to call this subroutine in his main program he puts in :

```
CALL SUB    nn
```

The controller will branch to this subroutine during execution, execute it, then return to begin executing the next statement of the program.
There is no restriction on the number of nesting levels, i.e., subroutine which can call other subroutines.
Here is an example of how they are used. Suppose the user wanted to mill characters (A,B,C, etc.). Let us use a 1/16" DIA MILL and let us make the characters .34" high by .23" wide. We have scaled up the characters (X10) to illustrated the moves per character. By inspection one can enter the move coordinates.



LOCAL ZERO for A LOCAL ZERO for B LOCAL ZERO for C

SUB 01

```
ZERO XY
GO   X    .04
     Y    .05
GO   Z   -.05
GR   X    .05
     Y    24
GR   X    .03
GR   X    .025
     Y   -.12
GR c X   -.08
GO   X    .13
     Y    .05
Z > C
GO   X    0.0
     Y    0.0
> REF COODS
SUB RETURN
```

SUB 02

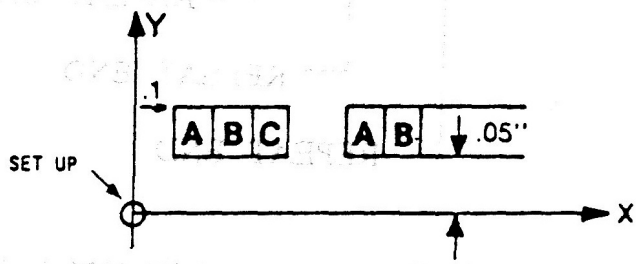
```
ZERO XY
GO   X    .05
     Y    .05
GO   Z   - .05
GR c X    .07
GR   Y    .12
GR c X    .07
GR   Y    .12
GR   X    .07
ZERO AT
     X    .07
     Y    .24
GR   a   - 180
ZERO AT
     Y   -.12
GR   a   - 180
Z > C
GO   X   -.12
     Y   -.11
> REF COODS
SUB RETURN
```

SUB 03

```
ZERO XY
ZERO AT
     X    .12
     Y    .22
GO   r    .07
     a    45
GO   Z   -.05
GR   a    135
GR   Y   -.10
ZERO AT
     Y   -.10
GR   a    135
Z > C
GO   X   -.12
     Y   -.12
> REF COODS
SUB RETURN
```

Each of the above subroutines will mill a specific character. If we now wish to lay out the characters as shown in the example, then the main program would look like this.

```
000   START INS  04
001   TD = .0625
002   FR  XYZ = 10.0
003   SETUP  >  zcxyu
004   GO  X  .10          Go to bottom LH side of
005       Y  .0           1st character
006   CALL SUB  01        Do the "A"
007   GR  X  .23
008   CALL SUB  02        Do the "B"
009   GR  X .23
010   CALL SUB  03        Do the "C"
011   GR  X  .46          SPACE
012   CALL SUB  01        Do the "A"
013   GR  X  .23
014   CALL SUB  02        Do the "B"
015   END
```



Observe that in each subroutine we immediately set a local zero and do all moves with respect to it. When we exit we go to the local zero then switch back to the REF COODS so the main program can locate the characters where it pleases.

3) REPEAT REPEAT END KEYS

These keys are instructions which permit the program to repeat any section of a program a specified number of times, from 1 to 99. This is done by enclosing the section of the program by REPEAT nn and terminating it by REPEAT END. For example:

```
REPEAT 20
-- -- -- -- --
-- -- -- -- --
OPERATION
-- -- -- -- --
-- -- -- -- --
REPEAT END
```

will repeat the operation 20 times.

These REPEATS can be nested to any level, providing there is always a REPEAT END for each REPEAT nn. For example all of these are valid.

```
REPEAT  10
    REPEAT  20
        REPEAT  60
        REPEAT  END
    REPEAT  END
REPEAT  END
```

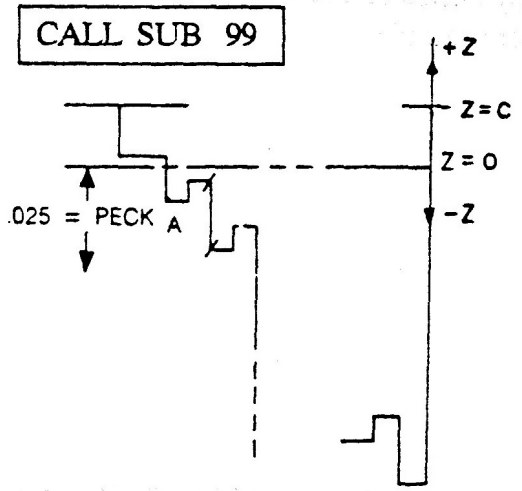
In this example, the operations in the center are repeated 60 x 20 x10 (=12000) times. This instruction is extremely useful in many applications. For example in pocketing and framing, we repeat the operation each time incrementing Z which in turn is repeated in the X & Y axis.

EXAMPLE 1 DRILL PECKING

Suppose the user wishes to write his own drill routine with small pecks. We shall write it as a subroutine with a REPEAT in it. The main program would be...

MAIN PROGRAM

```
CALL SUB 99
```



SUBROUTINE

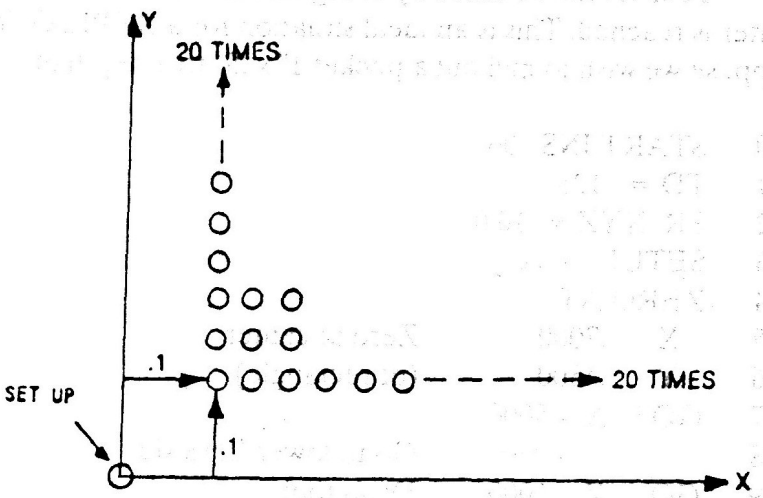
```
SUB       99
GO f  Z   .0050
GO    Z  -.0250
REPEAT  09
GR f  Z  +.0100     Clear drill
GR    Z  -.0100

                    GO up down ten thous
GR    Z  -.0250     drill pecks
REPEAT END          Loop 9 times
Z > C
SUB RETURN
```

In this example the A total depth is .025 + 9 x .025 = .25"

EXAMPLE 2 X,Y HOLE PATTERNS

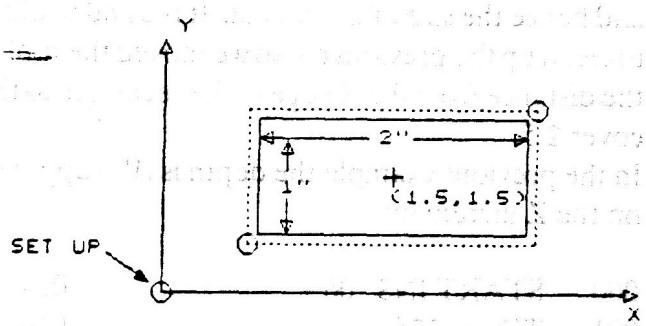
Now suppose we wish to use this in a 20 x 20 hole pattern in the XY plane. The program would be :

```
000   START INS  05
001   TD =       .0625
002   FR  XYZ = 15.0
003   SETUP  >  zcxyu
004   GO    X  .1000
005         Y  .1000
006   REPEAT     20
007   REPEAT     20
008   CALL SUB   99
009   GR    X  .1000
010   REPEAT END
011   GO    X  .1000
012   GR    Y  .1000
013   REPEAT END
014   END
```



EXAMPLE 3 RECTANGULAR FRAMING

Another example is the use of REPEAT / REPEAT END in frame cuts. You cycle round, increasing Z each time on the repeat.
Here is an example of a simple rectangular frame cut. The material is .25" thick and the tool is .125" in diameter. Each pass we lower Z the desired increment.



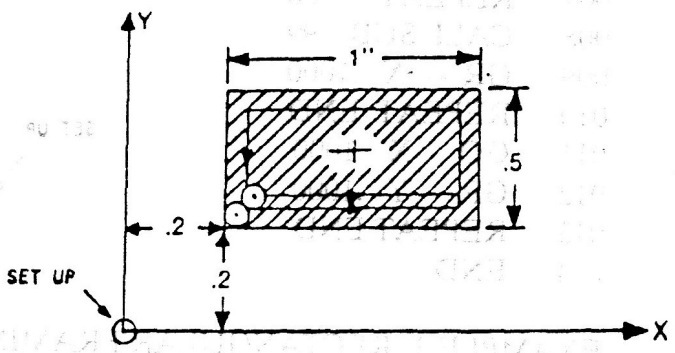

```
000   START INS  06
001   TD = .125
002   FR  XYZ =  10.0
003   SETUP  >  zcxyu
004   ZERO AT                 Zero at center
005     X    1.5000           of rectangle
006     Y    1.5000
007   GO o  X -1.0000
008         Y -0.5000         Go outside of rectangle-Lower LHS
009   GO    Z  0.0000         Lower tool to surface
010   REPEAT     05           Do 5 passes
011   GR    Z  -.0500         each of 50 thousands
012   CYCLE  XY               Cut rectangle
013   REPEAT END
014   END
```

This is exactly how the FRAME function works. For more complex geometries the cycle XY instruction (line 012) is replaced by the move statements necessary to cut the required geometry.

EXAMPLE 4 RECTANGULAR POCKETING

Pockets can be made by doing successive frame cuts. Each frame is made smaller until the center is reached. This is an ideal situation for a REPEAT instruction.
Suppose we wish to mill out a pocket 1" x .5" to a depth of .1" with a .125 dia tool as shown below

```
000   START INS  06
001   TD = .125
002   FR XYZ = 10.0
003   SETUP > zcxyu
004   ZERO AT
005     X    .7000          Zero at center
006     Y    .4500          (of rectangle)
007   GO i  X -.5000
008           Y -.2500       Go to lower LHS side
009   GO    Z -.1000         Drop tool
010   REPEAT   03            Repeat 3 times
011   CYCLE  XY              Cut frame
012   GR    X  .1000         Reduce frame size
013           Y  .1000
014   REPEAT END
015   END
```



How did we set the repeat at 3 ? This hinges on the size of the frame reduction (lines 012 and 013) and hence the size of the XYcut. It is usual to allow some of the tool diameter to be outside the cut to clean up the previous cut so we moved the tool in .1" (not .125). Now the critical dimension is .25, the distance from the X edge of the rectangle to the center. This is the shortest, so 3 frames of .1 will cover 2.5.
In the previous example the depth is .1", suppose we had wanted a depth of .3". We simply repeat on the Z increment.

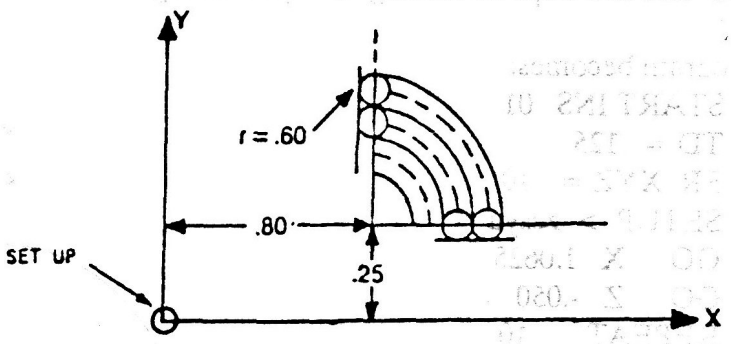
```
000   START INS  06
001   TD = .125
002   FR XYZ = 10.0
003   SETUP > zcxyu
004   XERO AT
005     X    .7000
006     Y    .4500
007   GO i  X -.5000
008           Y -.2500
009   GO i  X -.5000        Drop tool to surface
010   REPEAT    03
011   GR    Z -.1000        Increment Z into wor
012   REPEAT    03          On each pass
013   CYCLE     XY
014   GR    X  .1000        3 passes
015           Y  .1000
016   REPEAT END
017   GO i  X  .5000
018           Y  .2500
019   REPEAT END
020   END
```

The function rect. pocket does essentially the above program. It calculates the repeat n's based on Z% and XY%.

EXAMPLE 5 ARC POCKETING

Arc pocketing is done similarly. Suppose we wished to pocket an arc 90 degrees on a radius of .60", with a 1/8 inch dia mill, to a depth of .1". We shall move the tool along the radii to clean up the tool cuts.

```
000   START INS  07
001   TD =  .125
002   FR  XYZ = 10.0
003   SETUP  >  zcxyu
004   GO    X  .8000
005         Y  .2500
006   ZERO  XY
007   GO    Z  -.1000
008   GO i  r  .6000
009         a  90.000
010   GO    r  0.0000
011   GO    r  .5000
012   REPEAT     02
013   GR    a  90.000
014   GR    r  -.1000
015   GO    a  0.0000
```




```
016   GR    r  -.2000
017   REPEAT END
018   END
```

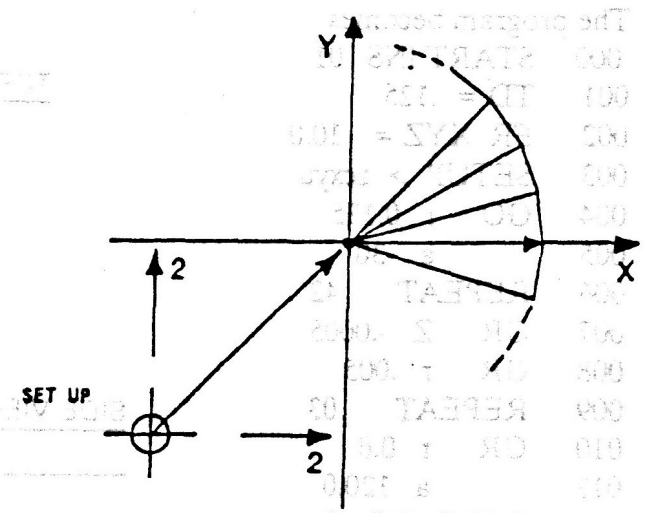
EXAMPLE 6 CUTTING A POLYGON

The object is to cut a polygon with 17 sides and radius 1 inch. It illustrates the use of the instruction.

GR	r
	a

The program becomes:

```
000   START INS  01
001   TD =  .125
002   FR    XY =  10
003   FR    Z  =  6
004   SETUP  >  zcxyu
005   GO    X  2          Zero at center 2,2
006         Y  2          from SETUP point
007   ZERO  XY
008   GO    X  1          Go to start of
009   GO    Z  -.050      polygon drop tool
010   REPEAT     17
011   GR    r  0.000
012         a  21.176
013   REPEAT END
014   Z  >  C
015   END
```



EXAMPLE 7 CUTTING A CAM

The object is to cut a cam. The radius of the cam is 1 inch at 0 degrees going to 0.9 inches at 180 degrees. Therefore the radius is reduced by 0.1 inch over 180 degrees. We have to decide on the decrement for r. Fix it at 0.0004 and find the angle.

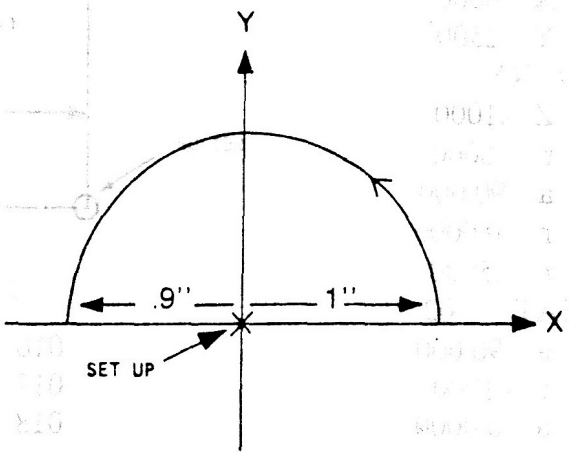
No. of steps is (.1) / (0.0004) = 250

So the angle increment is 180 / 250 = 0.72 degrees

Thus we take 250 steps increasing "a" by 0.72 degrees and decreasing r by .0004 each time.

The program becomes:

```
 000   START INS  01
001    TD = .125
002    FR XYZ =  10
003    SETUP > zcxyu
004    GO   X  1.0625
005    GO   Z  -.050
006    REPEAT    10
007    REPEAT    25
008    GR   r  -.0004
009         a  .72
010    REPEAT END
011    REPEAT END
012    Z > C
013    ENDN
```



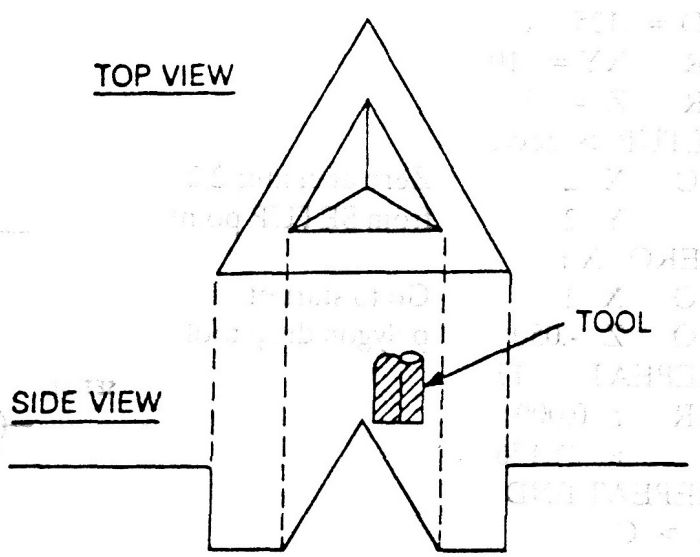
EXAMPLE 8 GENERATION OF A PYRAMID AND CONE

Here are two example that illustrate the generation of 3D shapes by little vector moves.

1. To generate a Pyramid

The program becomes:

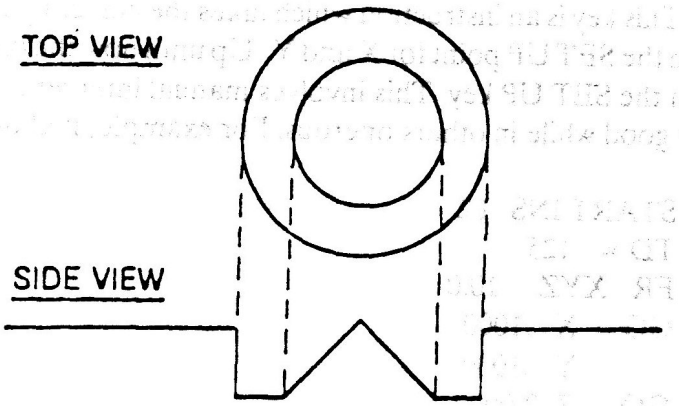
```
 000   START INS  01
001    TD = .125
002    FR XYZ =  10.0
003    SETUP > zcxyu
004    GO   r  0.135
005         a  -30
006    REPEAT    42
007    GR   Z  -.0005
008    GR   r  .005
009    REPEAT    03
010    GR   r  0.0
011         a  120.0
012    REPEAT END
013    REPEAT END
014    END
```



2. To generate a Cone

The program is:

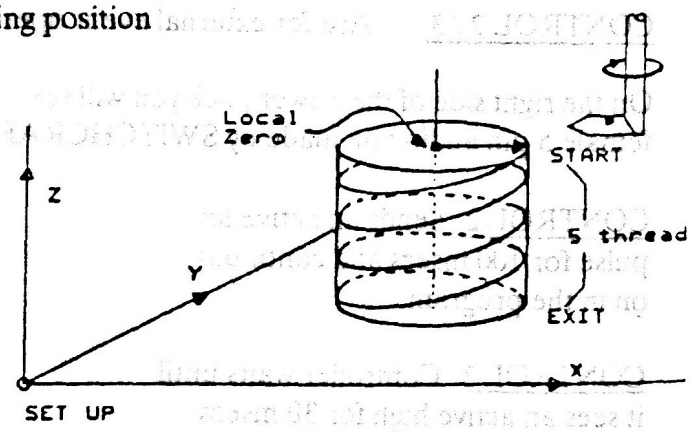
```
000   START INS  01
001   TD = .125
002   FR XYZ =  6.0
003   SETUP  >  zcxyu
004   GO    X  0.0625
005   GO    Z  0.0
006   REPEAT     42
007   GR    Z  -0.0005
008         X  0.005
009   GO    a  360.00
010   REPEAT END
011   END
```



EXAMPLE 9 HELICAL THREAD CUTTING

Any thread can be cut, externally or internally. You need a carbide insert that matches the required thread, locked in a boring tool. Multiple thread cutting tools are also available and this reduces the number of turns required to cut the helix. Essentially the procedure is to spin the boring tool and move it radially round in a circle in small angular increments dropping Z as well in small increments such that one turn is equivalent to the pitch of the thread. The tool chips out the thread cross section. Thus a pitch of .1 means Z is dropped .1 in 360 degrees, or .0025 every 9. This becomes the inner repeat loop. So you have :

```
Zero at
  X     2.0          Zero at the top of the cylinder
  Y     2.0
  Z     .6
GO    Z  .65         Move tool to starting position
GO    r  0.5
      a  0.0
GR    r  -.02
REPEAT 5             Make 5 threads
REPEAT 40            Cut 1 thread 360
GR    a  9
GR    z  -.0025
REPEAT END
REPEAT END
GR    r  +.2         Move tool out
GO    Z  0.0
```



4) PROG REF. KEY

This key is an instruction which takes the current position of the tool center and makes that position the SET UP point for X and Y. Up until now the only mechanism to set this point has been through the SET UP key. This involves manual intervention by the operator which in some cases may be good while in others onerous. For example, it allows the user to do this:

```
000   START INS  02
001   TD = .125
002   FR  XYZ   10.0
003   GO    X  .1000
004         Y  .1000
005   GO    Z 2.0000
006   HALT
007   PROG REF          set SET UP point at (.1,.1) in space
008 .....
```

The user MOVES the part to this location point for his SET UP, then clamps it in the jig. The halt can be removed later. The next time the program is run, it will go non-stop. This pre-supposes tool length etc. remain constant.
It also allows the user global program repeats. Just before the end tool is moved to the new SET UP POINT and is set by PROG REF, we then REPEAT the whole program.

24. THE CONTROL KEY

Pressing the control key, the display will show control n. Selecting : -

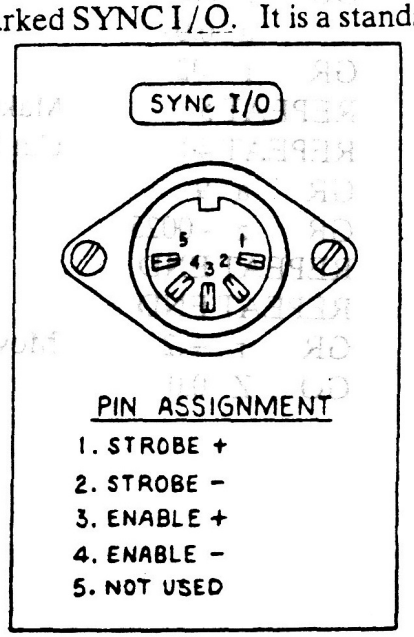
CONTROL 1 Not Used

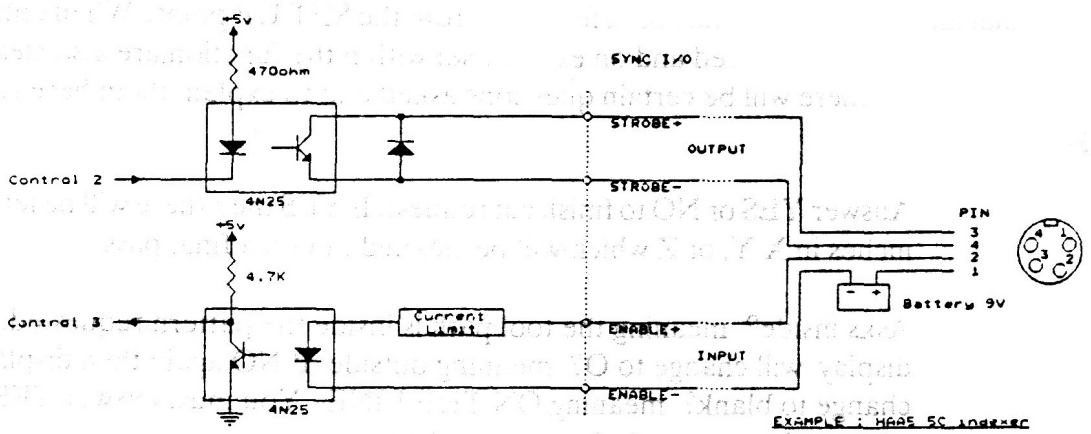
CONTROL 2 / 3 Are for external event operation for example in an indexer or cim cell.

On the right side of the power pack you will see a connector marked SYNC I/O. It is a standard female 5 pin audio one made by SWITCHCRAFT.

CONTROL 2 Sends an active low pulse for 100 msecs and continues on in the program.

CONTROL 3 Controller waits until it sees an active high for 30 msecs then continues on in the program.
Internally both signals are opto isolated like so : -





It is usually neccessary for the user to supply an external power supply 5v to 24v to activate the signals. Do not use the machine's internal ones.

25. THE FUNCTION KEY : SCALE ON /OFF

Touch the FUNCTION KEY and enter 00. Scale is function 00. It is a linear scaling on X,Y,Z or any combination of them either up or down and is used as a switch like SPINDLE ON/ OFF. All move statements between SCALE ON and SCALE OFF which involve the scaled axis or axes will be scaled.

To set SCALE

SCALE ON
(X x.xxxx)
(Y y.yyyy)
(Z z.zzzz)

( ) = OPTIONAL

For example scaling XYZ = 1.5 will produce a part 1.5 times the programmed dimensions. Where the values entered are the scaling factors. If fractional then the axis is scaled down, if greater than 1 then the scale is up.

Caution - The scale function must be turned off at the end of a program. Enter Function 00, answer No to Scale on, Yes to Scale off then enter which axis to Scale off.

* Obviously you can only scale up to the extent that the maximum move on that axis is not exceeded.
* When using SCALE ON, SCALE OFF, SCALE ON, SCALE OFF... in a program sequentially
* it is necessary to return the tool to the SCALE ON LOCATION before turning SCALE OFF to
* allow the controller to re establish its position.

26. THE CANNED FUNCTIONS

These are pre programmed to do certain operations like a circle pocket for example that greatly simplify program entry. They can be located anywhere in the program and the only caution in their use is that all the geometry must be referenced from the SET UP point. When entered all previously set local zeroes are cleared and on exit any set within the function are also cleared. When the key is pressed there will be certain questions asked and to explain them here is a short summary.

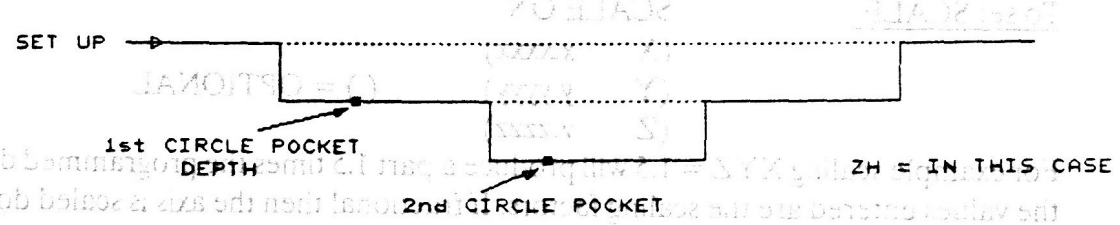
F ? Answer YES or NO to finish cut request. If YES then there will be left 0.0064 inches in X Y, or Z which will be cleaned out on a final pass.

i ? Asks inside? meaning the tool path is inside the pattern requested. If NO, display will change to O? meaning outside. If NO again then display will change to blank? meaning ON THE LINE. You must answer YES to one of these 3 questions before proceeding.

Z% nnn The distance Z is dropped on each pass expressed asa percentage of the tool diameter. Thus 050 means 50% , 100 means 100% , 200 means 200% . If the tool diameter is 0.25 then Z% 050 means the tool will be dropped 0.125 on each pass.

Zd = ? Total depth of cut in the Z axis relative to Z = 0 the SET UP point. Requires a numeric entry.

ZH = ? Normally zero. If however you have a circle pocket (for example) within a circle pocket then you would like to start this circle pocket a depth ZH from the origin. This offset ZH corresponds to Zd in the first pocket.
ZH is - below surface (Z = 0).
+ above the surface.



XY CUT % nnn To the amount of horizontal cut in X and Y on each pass of a pocket mill and is always expressd as a percentage of the tool diameter. For example 050 means that 50% or half the tool will be in the work in the XY plane.

X1 = ?
Y1 = ? These ask for the distances in X and Y from or of a specific geometry. Always refer to the function description following.
XA= ?
YB= ?
etc.

ELSEWHERE? This question appears at the end of a function. It permits the replication of the geometry defined by the funcion above it in another location on the work piece. A YES response to it results in additional questions for the coordinates of the new location.

REPEAT ?
REPEAT X nn
REPEAT Y nn

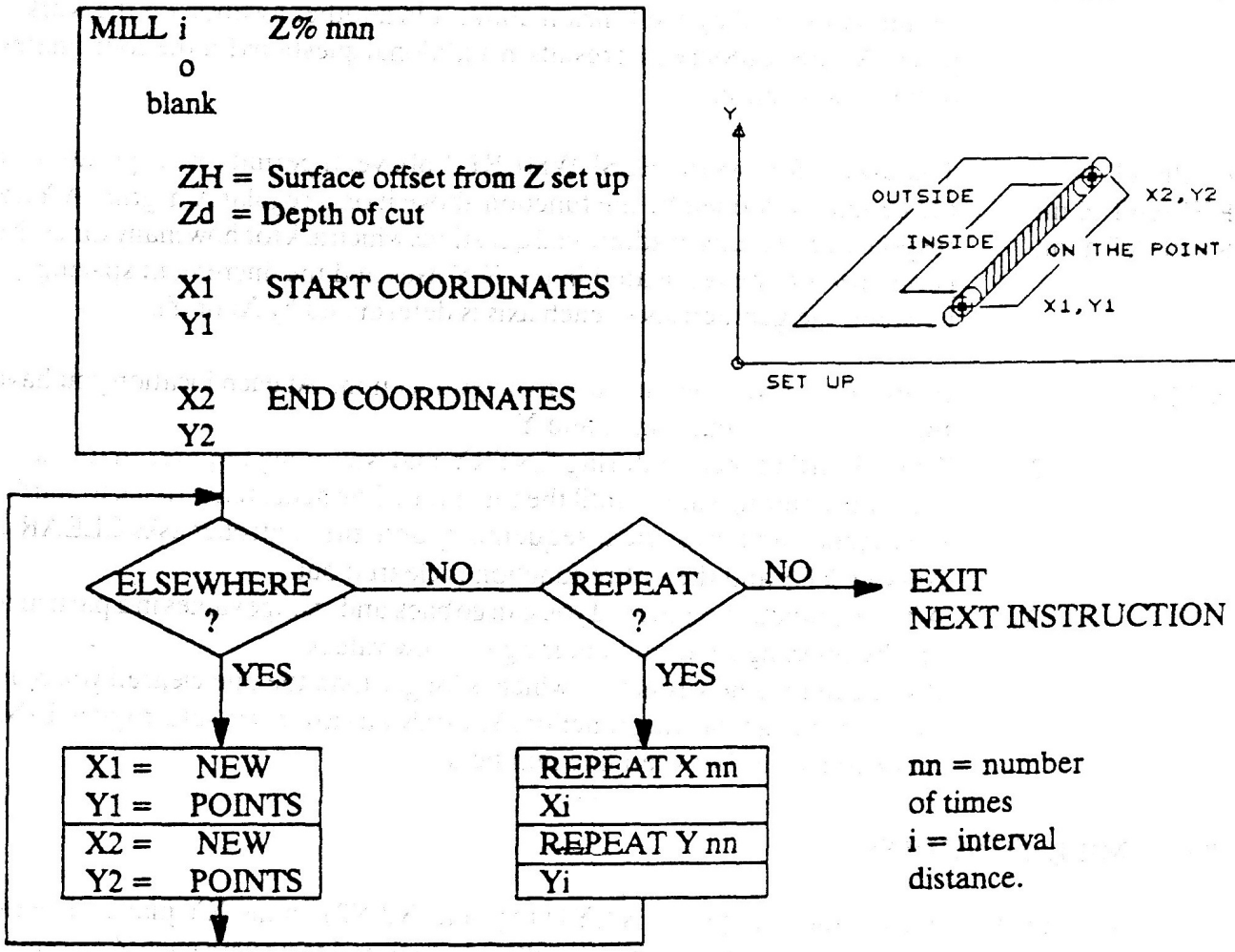
This always follows the ELSEWHERE? above. It permits the replication of the geometry defined by the function above it on a regular X Y grid. A YES response to it results in additional questions which ask for how many times the geometry is to be replicated in the X Y axes and the increment spacing between the geometries in each axis is determined by Xi or Yi.

NOTE :
1) Elsewhere moves the function to a new location. At each location you have the option to repeat it in X and Y.
2) To CLEAR a function during ENTRY mid way along requires that you continue entering values until the function is finished, then go back to start of the function and press clear sequentially until the controller asks CLEAR ?. Answer YES and the entire function is cleared out.
3) After the function is entered you can go back and change values in a particular line by pressing clear and entering the new values.
4) If you call up a new function which is longer than the one cleared you may over-run into a following function. The only way to clear it out is to go to LINE mode and clear between line numbers.

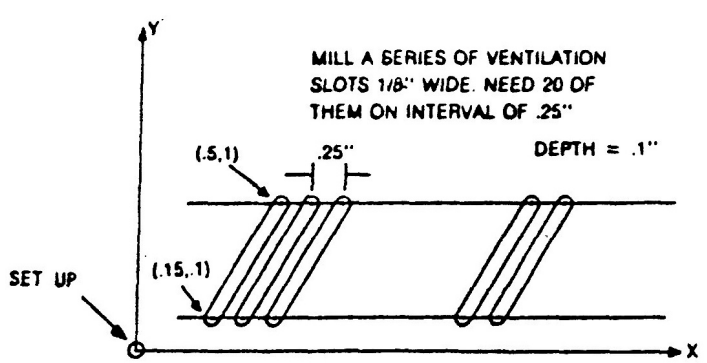
26.1 MILL FUNCTION

This function mills a slot from point (X1,Y1) to point (X2,Y2) on the XY plane. The slot can be inside the points, outside or on the points. It will cycle back and forth with each Z increment a percentage (Z% nnn) of the tool diameter until Zd is reached. The width is the tool diameter.

The ENTRY FORMAT is: -



EXAMPLE OF MILL



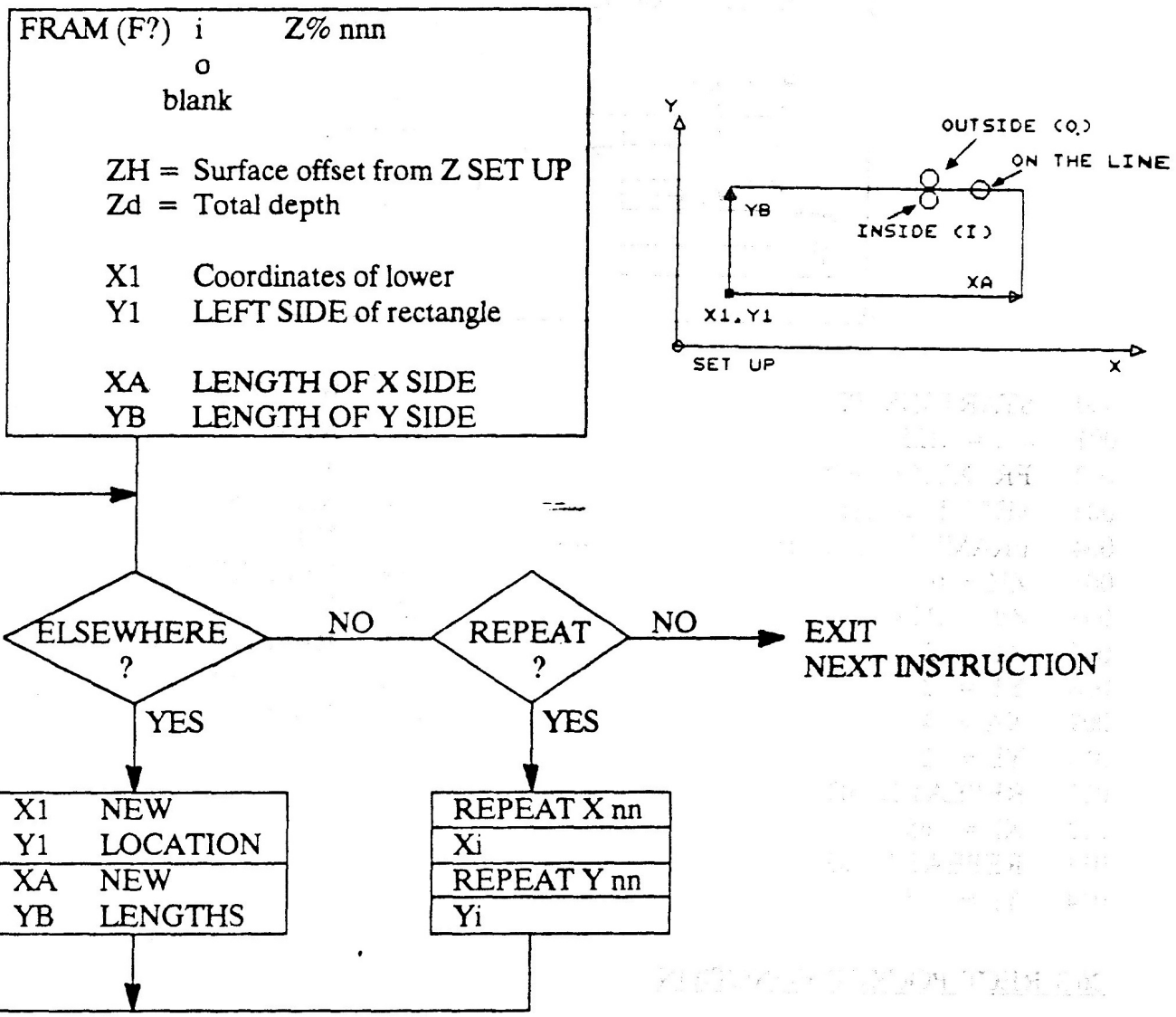
The program using MILL would appear as:

```
000   START INS  01
001   TD  = .125
002   FR XYZ =  10.0
003   SET UP  > zcxyu
004   MILL  Z%  050
005   ZH = 0
006   Zd =  .1
007   X1 =  .15
008   Y1 =  .1
009   X2 =  .5
010   X2 =  1.0
011   REPEAT  X  20
012   Xi =  .25
013   REPEAT  Y  00
014   Yi =  0
015   END
```

26.2 RECT FRAME FUNCTION

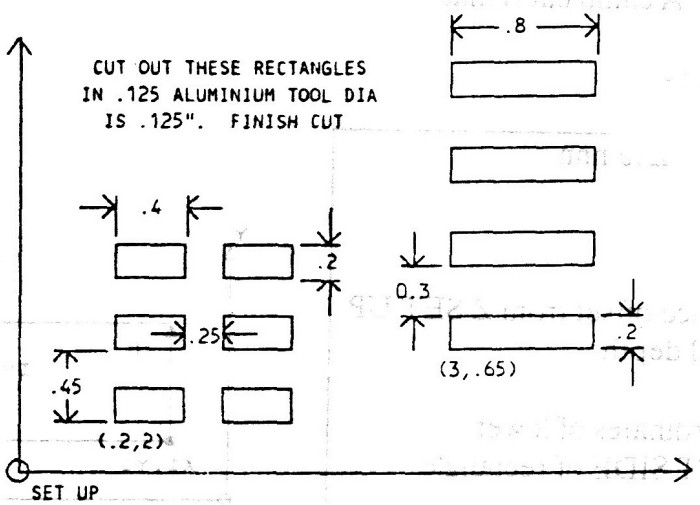
This will cut a rectangular frame in the XY plane. The cutting can be Inside (i), Outside (o) or on the frame (blank). If a finish cut (F) is specified then a remainder is left of 0.0064" which is cleared out by a final pass. A climb cut is made inside (counterclockwise) or outside (clockwise).

The ENTRY FORMAT is : -



EXAMPLE OF RECT FRAME

The program using the RECT FRAME would be :



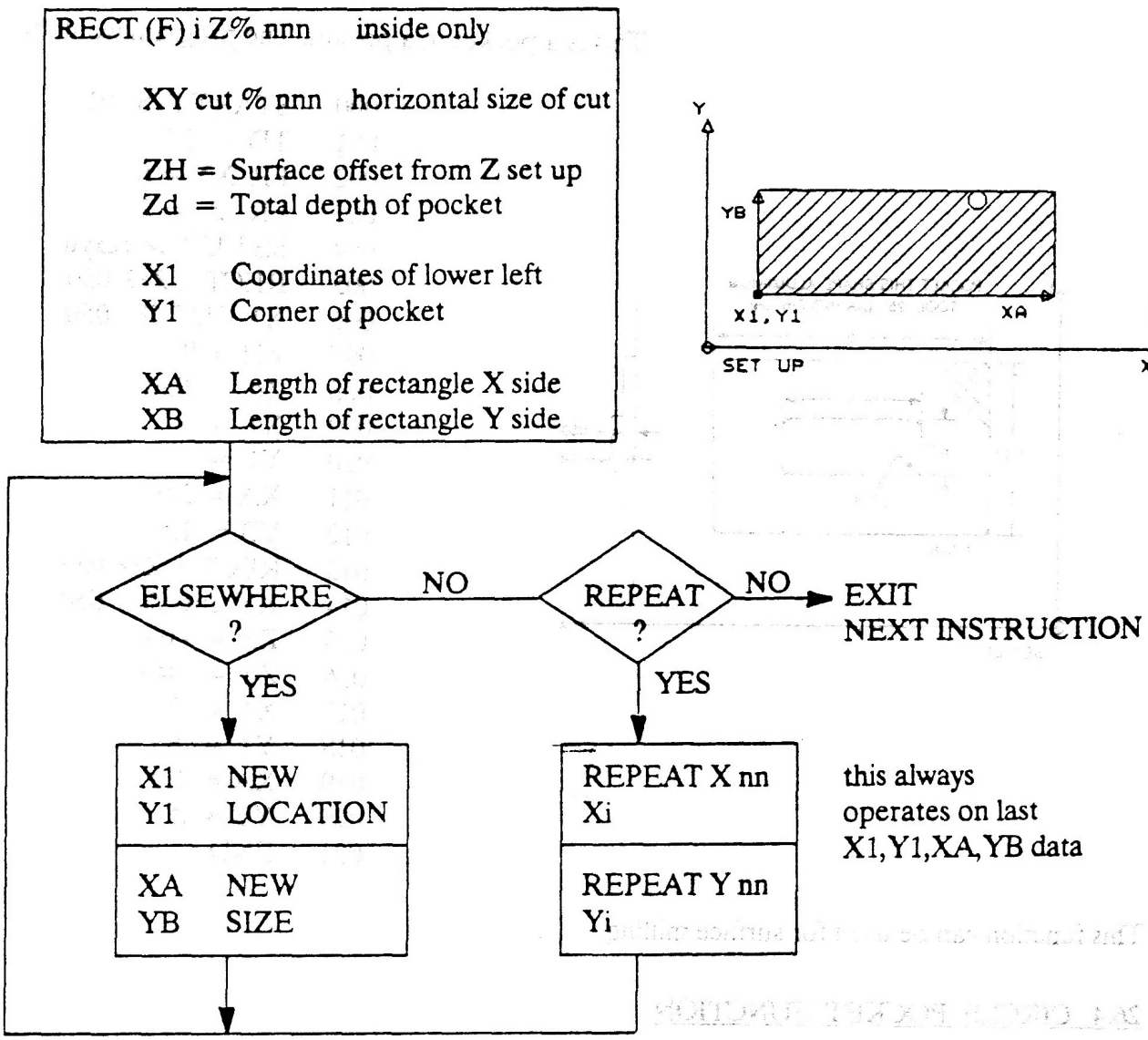

```
000   START INS  02
001   TD = .125
002   FR  XYZ =  8.0
003   SET UP  >  zcxyu
004   FRAME F  i  Z% 050
005   ZH = 0
006   Zd  = .128
007   X1 = .2
008   Y1 = .2
009   XA = .4
010   YB = .2
011   REPEAT X  02
012   Xi  =  .65
013   REPEAT Y  03
014   Yi  =  .45
(ELSEWHERE) YES
015   X1 = 3.0
016   Y1 = .65
017   XA = .8
018   YB = .2
019   REPEAT X  01
020   Xi  =  0.0
021   REPEAT Y  04
022   Yi  =  0.5
023   END
```

26.3 RECT POCKET FUNCTION

This function generates a rectangular pocket in the XY plane. If a finish cut (F) is requested the pocket is made smaller by 0.0064 ins. on each dimension and a final pass clears this out.

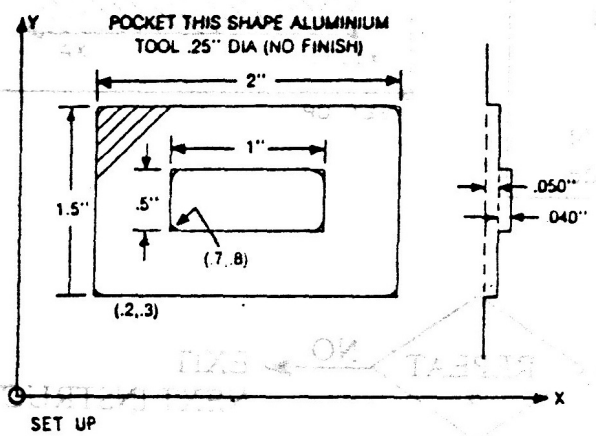
There is only Inside (i) allowed.

The ENTRY FORMAT is : -



EXAMPLE OF RECT POCKET

This is a pocket in a pocket. The program would be:



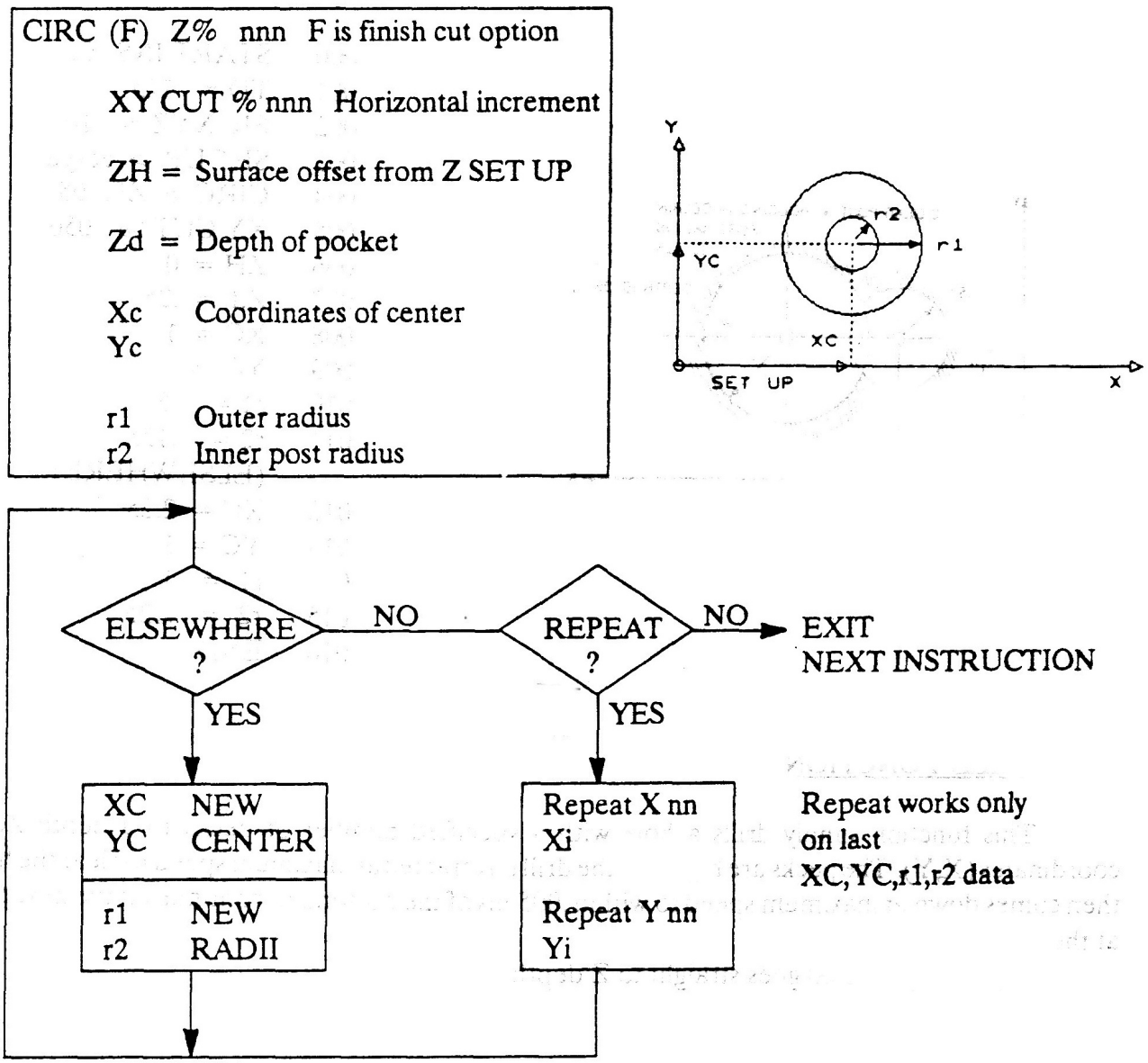

```
000   START INS  01
001   TD = .250
002   FR XY = 8
003   FR  Z  = 2
004   SET UP > zcxyu
005   RECT  i Z% 050
006   XY CUT %   050
007   ZH = 0
008   Zd = .050
009   X1 = .2
010   Y1 = .3
011   XA = 2.0
012   YB = 1.5
013   RECT i Z% 050
014   XY cut %   050
015   ZH = -.05
016   Zd = .040
017   X1 = .7
018   Y1 = .8
019   XA = 1.0
020   YB = .5
021   END
```

This function can be used for surface milling.

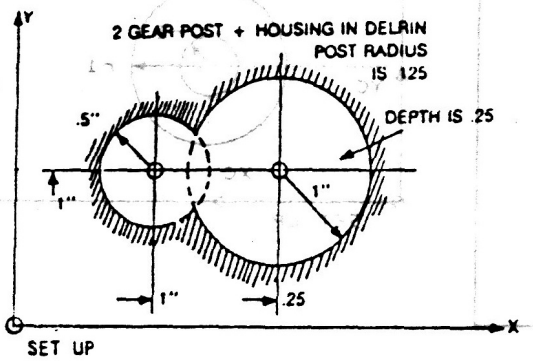
26.4 CIRCLE POCKET FUNCTION

This function generates a circular pocket of radius r1 with a center post of radius r2 on the inside. The tool is automatically compensated for on the inside of r1 and the outside of r2. If r2 = 0 then the post disappears. To cut a frame, make r2 = r1 - TOOL DIAMETER with no finish cut.

The ENTRY FORMAT is :



EXAMPLE OF CIRCLE POCKET



The program would be :

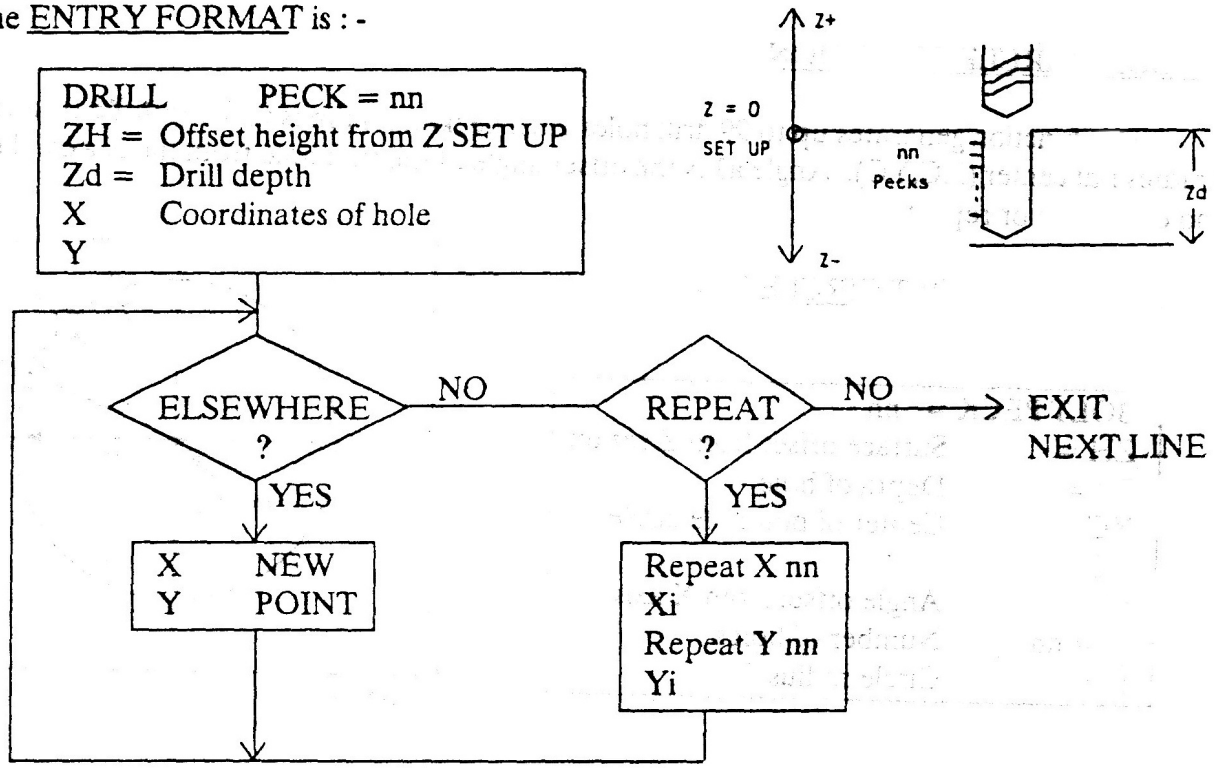
```
000   START INS  01
001   TD =  .125
002   FR XYZ =   10
003   SET UP  >  zcxyu
004   CIRC F Z%  050
005   XY CUT %   050
006   ZH = 0
007   Zd  =  .25
008   XC =  1
009   YC  = 1
010   r1 =   .5
011   r2 =   .125
      (ELSEWHERE)
012   XC =  2.25
013   YC =  1
014   r1 =   1
015   r2 =   .125
016   END
```

26.5 DRILL FUNCTION

This function simply drills a hole with a specified number of pecks to a depth Zd at coordinates (X,Y). The pecks are big ones, the drill is retracted at maximum speed to clear the hole, then comes down at maximum speed to within .005 ins of the bottom, to continue on the next peck at the programmed speed.

A peck equal to 00 goes straight to Z depth.

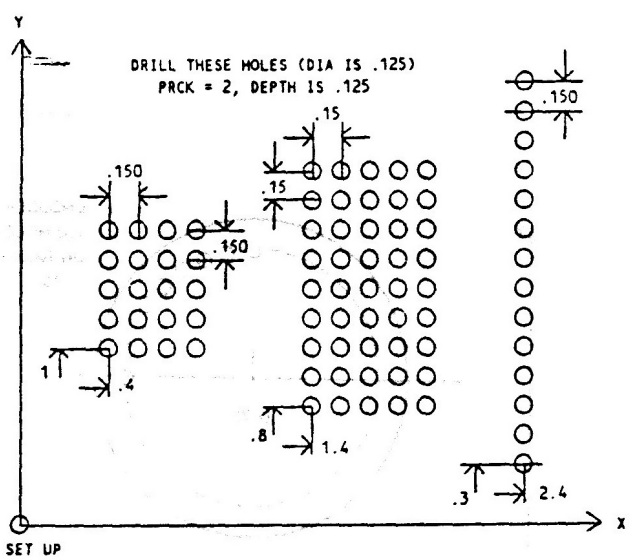
The ENTRY FORMAT is : -



EXAMPLE OF DRILL

The program would be :

```
000   START INS  01
001   FR  XY = 16
002   FR    Z = 10
003   SET UP ---> zcxyu
004   DRILL PECK = 02
005   ZH = 0
006   Zd = .25
007   X  = .4
008   Y  = 1
009   REPEAT X  04
010   Xi = .150
011   REPEAT Y  05
012   Yi = .150
  (ELSEWHERE)
013   X  = 1.4
014   Y  = .8
015   REPEAT X  05
016   Xi = .150
017   REPEAT Y  09
018   Yi = .150
```



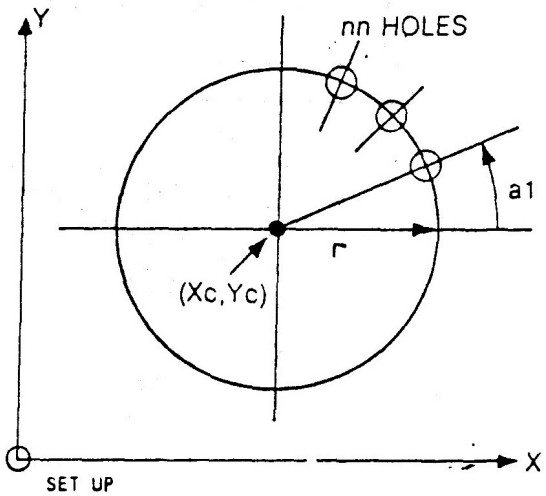

```
  (ELSEWHERE)
019   X  = 2.4
020   Y  = .3
021   REPEAT X  01
022   Xi = 0
023   REPEAT Y  14
024   Yi = .150
025   END
```

26.6 BOLT CIRCLE FUNCTION

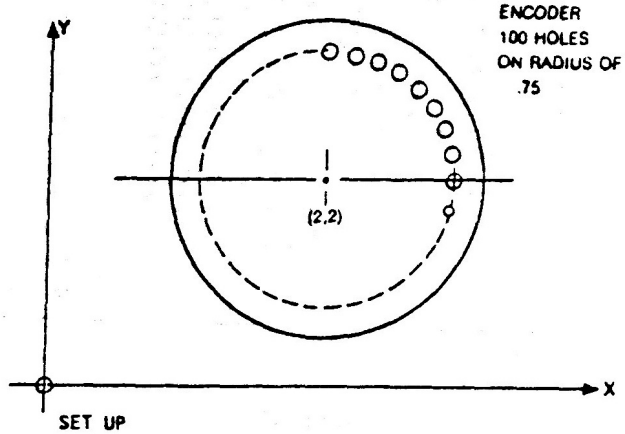
This function generates up to 99 drill holes each pecked nn times to a depth Zd on a circle of radius r at center (XC,YC). Angle a1 is the offset angle of the first hole from the X axis. There is no elsewhere or repeat.

The ENTRY FOR BOLT CIRCLE is : -

BOLT PECK = nn	
ZH =	Surface offset from Z set up
Zd =	Depth of hole
XC	Center of bolt hole circle
YC	
a1	Angle offset from X axis
N = nn	Number of holes
r =	Circle radius



EXAMPLE OF BOLT CIRCLE



The program would be :

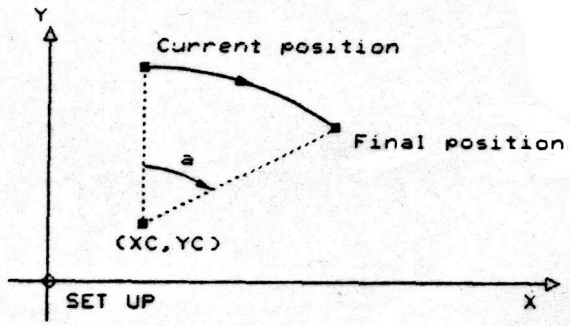
```
000   START INS  01
001   FR XY =    16
002   FR Z =     8
003   SET UP > zcxyu
004   BOLT PECK = 0
005   ZH = 0
006   Zd = .1
007   XC = 2.0
008   XY = 2.0
009   a1 =  0 deg
010   N =  50
011   r =  .75
012   BOLT PECK = 0
013   ZH = 0
014   Zd = .1
015   XC = 2.0
016   YC = 2.0
017   a1 =  3.6
018   N =  50
019   r =  .75
020   END
```

26.7 ARC FRAME FUNCTION

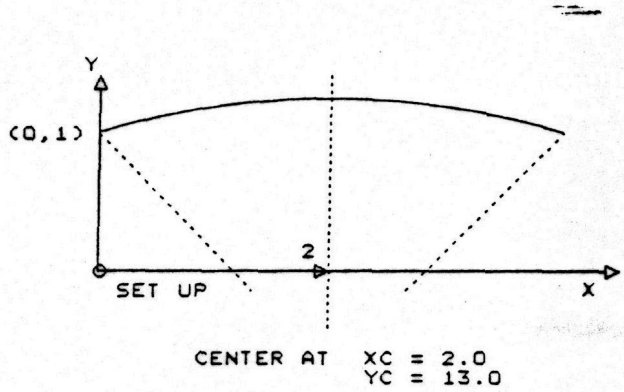
This function simply moves the tool from its current position through an arc centered at XC,YC by an angle a to its final position. The main purpose of this function is to allow a zero at off the table such that the radius can be up to 36". Thus it can cut a very shallow arc.
There is no elsewhere or repeat

The ENTRY FORMAT FOR ARC FRAME:

ARC	
XC	Coordinates of ARC
YC	Center
a	Turn angle (degrees) + for counterclockwise - for clockwise



EXAMPLE OF ARC



The program would be :-

```
000   START INS  01
001   TD =  .125
002   FR XYZ =   10.0
003   SET UP  >  zcxyu
004   GO    Y      1.0
005   GO    Z      -.1
006   ARC
007   XC    =  2.0
008   YC    =  -13.0
009         a     -16.426
010   END
```

26.8 USING PROG REF FOR THE CANNED FUNCTIONS

The elsewhere and repeat on the canned functions allows the same function to be done at random or periodically across the XY plane. However it is sometimes necessary to do a sequence of canned functions at one location then move to a new location and repeat this sequence of canned functions. The best way to this is with the PROG REF instruction. Enclose the sequence of canned functions in a subroutine headed by PROG REF.
Move the tool to the required "new set up" point and call the subroutine. Rember that the next "new set up" point must be referenced to the previous one.