SECTION VI

REPETITIVE PROGRAMMING

The Bridgeport CNC provides two different ways of accomplishing repetitive programming without the tedious monotony of writing down the same commands over and over again. These programming techniques are referred to as Looping and Macro (sub-routines).

> LOOPING provides the programmer with the ability to jump back to an earlier part program block and execute the intervening part program blocks a specified number of times.
>
> MACROS provide the programmer with the ability to program a change in the value of an addressed parameter and to execute a subprogram consisting of a group of part program blocks with variable parameters.

The examples included here can only be a few simple cases to illustrate the correct use of the statements. Imaginative programmers have found that repetitive programming is a powerful aid in preparing part programs.

6.1 LOOPING

There will be times when the part programmer is required to write a program in which a sequence of statements is to be repeated. The looping feature of the Bridgeport CNC was designed to relieve the part programmer of this type of tedious task. Looping is a technique that instructs the computer to read and execute a set of statements in the part program as many times as indicated by the programmer. The use of this feature can reduce the programming effort significantly, causing a few statements to do the work of many.

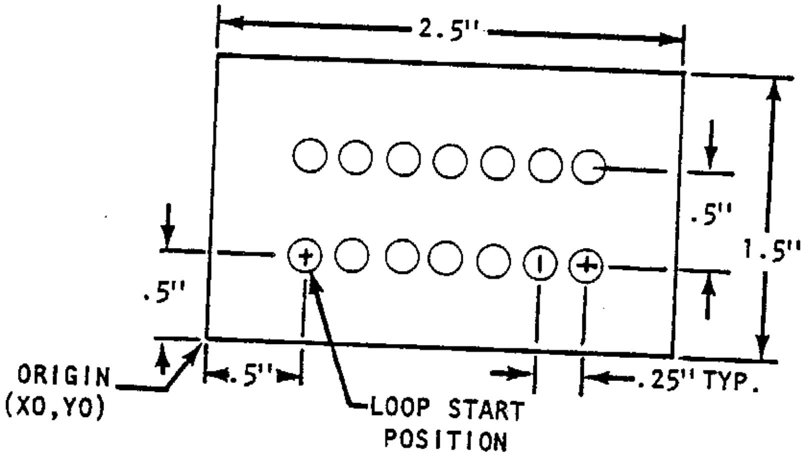
The Looping part program call statement is of the following form:

= Na/b	where	a	is the loop end block sequence number,
		b	is the number of times the loop is to be repeated.

The part program blocks that follow the looping statement up to and including the block with the loop end sequence number is called the range of the loop.

Following a loop call statement, all part program blocks in the range of the loop are executed. After execution of the loop end block, a register containing the number of times the loop is to be repeated is decremented. If this register is greater than zero, the system will branch back to the first executable part program block following the loop call. This process continues until the repeat register is zero; at which time, the system will execute the next part program block following the loop end block.

The part shown below (Example #1) will illustrate one method of using loops to simplify programming.



Example #1

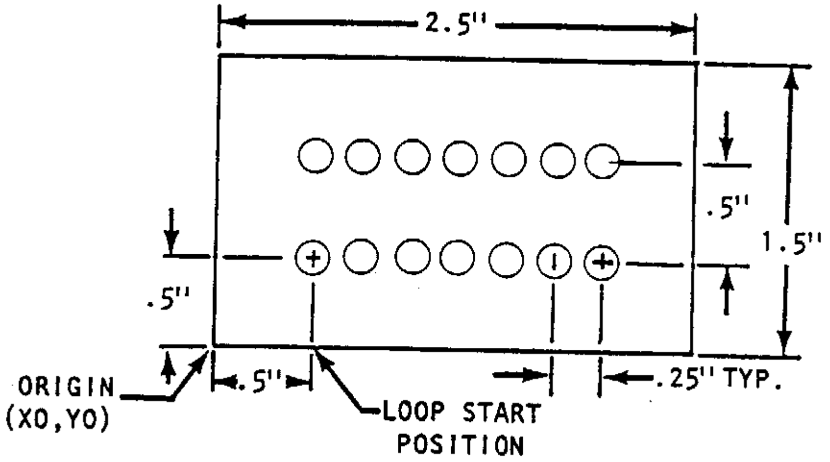
```
%N1G0G90X-3.Y0S3300T1M6
N5X.5Y.5Z.05
N10G91G81Y0Z.625F80
=N15/6
N15X.25
N20Y.5
=N25/6
N25X-.25
N30G0G90X-3.Y0M2
E
```

The above program will follow this operational sequence:

Block #	%N1	- Recommended start block format at tool change position (X-3.Y0)
	N5	- Rapid to absolute position (X.5, Y.5) for start of pattern
	N10	- Sets Cycle and Drills first hole
	=N15/6	- Loop call, ending at Block N15, Repeat (6) times
	N15	- Move .25" in X and drill
	N20	- Move .5" in Y and drill
	=N25/6	- Loop call, ending at Block N25, Repeat (6) times
	N25	- Move -.25" in X and drill
	N30	- Recommended end block format at tool change position
	E	- Rewind Tape

Example #1 shows the Loop Start position over the hole, this would be a typical start position for any odd numbers of rows in a pattern of holes. This is further shown in the example Program on page 6-7.

The identical part shown below (Example #1A) will illustrate an alternative method of programming the same pattern. The Loop Start position has been changed to show the typical start position for any even number of rows in a pattern of holes. This basic method will be carried thru to the next two examples.



Example #1A

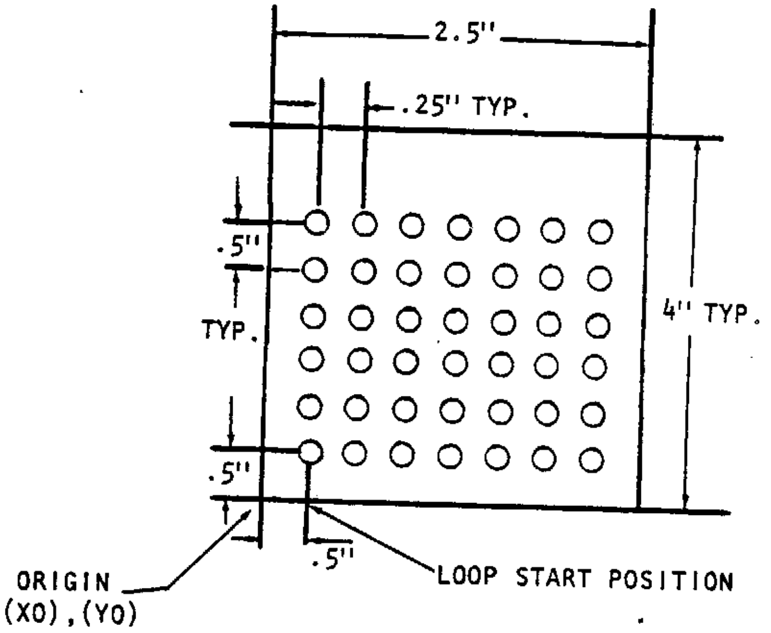
```
%N1G0G90X-3.Y0S3300T1M6
N5X.5Z.05
N10G91G81Y.5Z.625F80
=N15/6
N15X.25
N20Y.5
=N25/6
N25X-.25
N30G0G90X-3.Y0M2
E
```

The above program will follow this operational sequence:

Block # %N1 - Recommended start block format at tool change position (X-3.Y0)
N5 - Rapid to absolute position (X.5, Y.0) for start of pattern
N10 - Sets Cycle and Drills first hole
=N15/6 - Loop call, ending at Block N15, Repeat (6) times
N15 - Move .25" in X and drill
N20 - Move .5" in Y and drill
=N25/6 - Loop call, ending at Block N25, Repeat (6) times
N25 - Move -.25" in X and drill
N30 - Recommended end block format at tool change position
E - Rewind Tape

6.1.1 Nested Looping

Loops may be nested up to 4 levels. The range of an inner nested loop must lie completely within the range of the next outer loop. They may share the same loop end block sequence number. See example #2.



Example #2.

```
%N1G0G90X-3.Y0S3300T1M6
N5X.5Z.05
=N25/3
N10G91G81Y.5Z.625F80
=N15/6
N15X.25                    Loop 1
N20Y.5                                  Loop 3
=N25/6
N25X-.25                   Loop 2
N30G0G90X-3.Y0M2
E
```

The loops within Example #2 above will execute the following:

Loop #1 - Drill (6) holes, on .25" centers, moving positive X direction
Loop #2 - Drill (6) holes, on .25" centers, moving negative X direction
Loop #3 - Repeat Loops 1 and 2, with .5" Y stepover, as pattern to complete 6 rows

6.1.2 Rules

a. The loop end block sequence number must appear later in the part program.

b. All part program blocks following a loop call statement, through the loop end block, must contain a sequence number. NOTE: (Loop calls and macro calls need not have sequence numbers)

c. The maximum number of repeats is 19999.

d. The loop end block must not be the last block of a program.

e. Do not program metric data incrementally. Considerable errors can be accumulated in a looped program (BOSS 4 only).

6.2 MACRO SUB-ROUTINES

The word MACRO is short for macro-command. A macro-command is a single statement which refers to a group of part programming statements. A number is assigned to the macro which is then stored until called for. The macro is not executed at the time of definition.

(REF: The word "Macro" is a prefix meaning combining and is a well accepted term to denote a command which combines a series of operations.)

The Macro definition is of the form:

```
#n
.
.
.      (TEXT)
.
$
```

The #n statement defines the part program commands that follow it up to and including a block containing only a $ sign as a macro sub-routine. Up to 36 macro sub-routines can be designated at any one time within a program.

The range of n is from 1 to 16 (36 for BOSS 5 and 6). Within the MACRO, variable parameters that are to be assigned by the macro call statement, are designated by an asterisk (*), and must be inserted into the macro in the order that they are specified.

The macro call statement is of the form:

=#nF*100 where n is the macro sub-routine to be executed. F*100 is the parameter to be assigned for the unspecified macro variable (F).

6.2.1 Macro Capabilities

1. A macro sub-routine may call another macro sub-routine.

2. Macros may be nested up to 4 levels.

3. A macro sub-routine may include a loop.
NOTE: If a macro sub-routine includes a loop, the ($) macro termination must not be on the loop end block number.

4. A macro sub-routine call may be included within a loop.

5. If a macro variable is the same value in the next call as was previously defined, the value need not be re-entered.

   For example: =#2X*-1.5Y*2.0
   =#2X*Y*

   will call the same values of X-1.5 and Y2.0. Note that this must be the next call.

   Example: =#2X*-1.5Y*2.0
   =#3X*-3.0Y*4.0
   =#2X*Y*

   will call macro#2, but with the wrong values (X-3.0Y4.0).

6. Macros with a given tag may be defined and redefined many times within a program, however, if two macros with the same tag are specified within a program, the call statement will execute the last macro defined.

6.2.2 Macro Programs

Referring back to the part and nested loop program shown in example #2, if it were necessary to perform multiple operations on each hole, the use of a macro sub-routine would substantially reduce the programming effort as shown below.

(Tool 1 90° Spot Drill; Tool 2 - 3/16 Dia. Drill)

Example #3

```
             %#1
             N5X.5Z.05
             =N25/3 ----------------------------------+
             N10G91G*Y.5Z*Z*Z*F*                      |
             =N15/6 -------------------+              |
   Macro     N15X.25 ------------------+ Loop 1       |
 Definition  N20Y.5                                   | Loop 3
             =N25/6 -------------------+              |
             N25X-.25 -----------------+ Loop 2 ------+
             $
```

```
              N1G0G90X-3.Y0S1500T1M6      (Spot Drill)
Macro Call    =#1G*81Z*.150Z*0Z*0F*80
              N100X-3.Y0S3300T2M6         (Drill)
Macro Call    =#1G*87Z*.625Z*.35Z*.25F*100
              N110X-3.Y0M2
              E
```

NOTE: The first macro call of the program above will Spot Drill all the holes, to a diameter of .200. The second macro call will then Drill all the holes using a Peck Drilling cycle.

6.2.3 Rules

a. A macro sub routine must not be defined within another macro sub routine.

b. Variables must be inserted into the macro in the order that they are specified. The number of unspecified variables within the macro and the number of specified variables in the call statement must be the same.

c. The macro call can contain no more than 47 characters. This is the limiting factor as to the number of variable parameters that can be used within a macro. The total number of active macro variables (*) is equal to 12 whether one macro or a group of nexted macros is considered. If the call statement contains more than 47 characters (due to the number of variables) the original macro must be split into two or more macros so that each call statement will not exceed 47 characters.

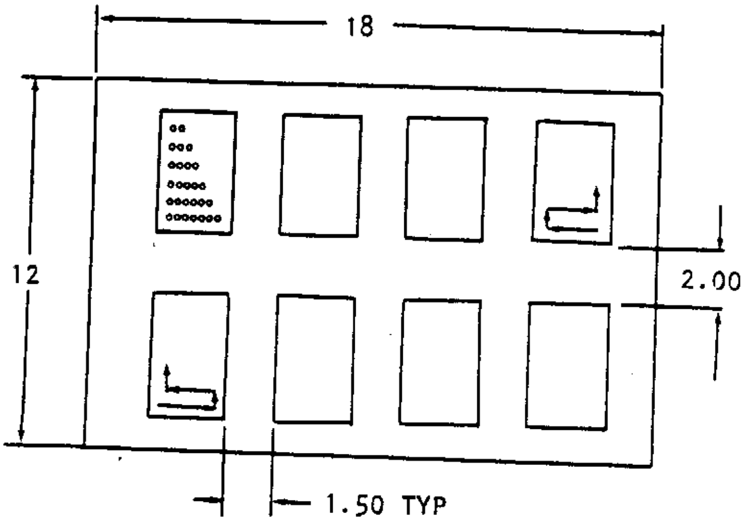
d. The largest number that may be entered as a variable parameter value is ±16.0000" (160 mm in metric mode). e.g. "A" may be used as a variable however, its value is limited to 16. degrees.

e. The following parameters may not be used as macro variables:

Macro Call
Loop Call
Scale Factors
+ or - signs
Cutter Compensation Values
Tool Offset Value

Example #4

To further exemplify the usefulness of loops and macros, consider eight parts located on a fixture plate as shown below.



Notice the Macro, with an "X" variable, is looped in conjunction with a G92 block, to shift the origin for each part. The "X" variable allows positioning <u>left to right</u> for the bottom four parts, then <u>right to left</u> for the top four parts, saving rapid positioning time.

```
%#1
N10G*G91X0Z*Z*Z*F*
=N35/3
N15
=N20/6
N20X*
N25Y.5
=N30/6
N30X*
N35Y.5
=N40/6
N40X*
$
N1G0G90X-1.5Y0S1500T1M6 (SPOT DRILL)
.N5G4/2
=N60/4
N55X.5Y.5Z.05
=#1G*82Z*.119Z*0Z*0F*80X*.25X*-.25X*.25
N60G0G90G92X-2.
```

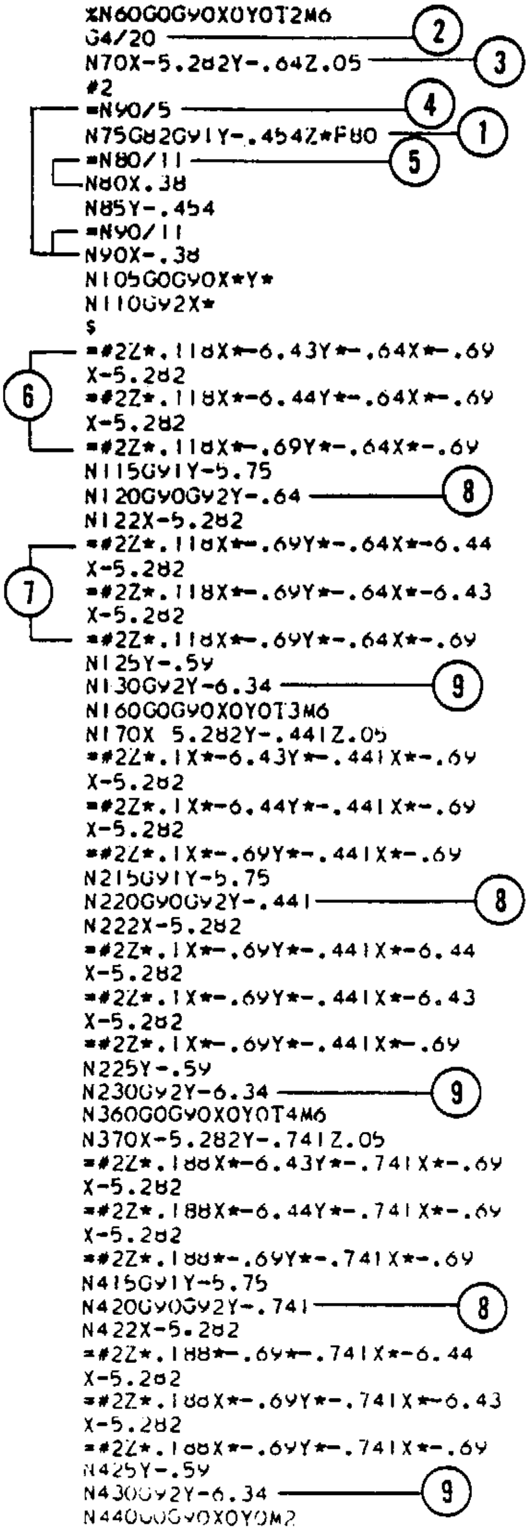
```
N65G92X2.Y-2.5
=N75/4
N70X2.Y.5
=#1G*82Z*.119Z*0Z*0F*80X*-.25X*.25X*-.25
N75G0G90G92X4.5
N80G92X.5Y9.5
N85G0G90X-1.5Y0S3300T2M6        (DRILL)
=N95/4
N90X.5Y.5Z.05
=#1G*87Z*.625Z*.35Z*.275F*100X*.25X*-.25X*.25
N95G0G90G92X-2.
N100G92X2.Y-2.5
=N110/4
N105X2.Y.5
=#1G*87Z*.6275Z*.35Z*.25F*100X*-.25X*.25X*-.25
N110G0G90G92X4.5
N115G92X.5Y9.5
N120G0G90X-1.5Y9.M2
E
```

6.3 ADDITIONAL REPETITIVE PROGRAMMING EXAMPLES

Three examples follow showing specific difficult parts beings machined in the field. The tape image data in each case has been marked to reference notation by way of explanation:

Figure 6-1. Nested Loop Example

Figure 6-2. Macro Call Within A Loop, Within A Loop, Within A Macro.



```
%N60G0G90X0Y0T2M6
G4/20
N70X-5.282Y-.64Z.05
#2
*N90/5
N75G82G91Y-.454Z*F80
*N80/11
N80X.38
N85Y-.454
*N90/11
N90X-.38
N105G0G90X*Y*
N110G92X*
$
*#2Z*.118X*-6.43Y*-.64X*-.69
X-5.282
*#2Z*.118X*-6.44Y*-.64X*-.69
X-5.282
*#2Z*.118X*-.69Y*-.64X*-.69
N115G91Y-5.75
N120G90G92Y-.64
N122X-5.282
*#2Z*.118X*-.69Y*-.64X*-6.44
X-5.282
*#2Z*.118X*-.69Y*-.64X*-6.43
X-5.282
*#2Z*.118X*-.69Y*-.64X*-.69
N125Y-.59
N130G92Y-6.34
N160G0G90X0Y0T3M6
N170X 5.282Y-.441Z.05
*#2Z*.1X*-6.43Y*-.441X*-.69
X-5.282
*#2Z*.1X*-6.44Y*-.441X*-.69
X-5.282
*#2Z*.1X*-.69Y*-.441X*-.69
N215G91Y-5.75
N220G90G92Y-.441
N222X-5.282
*#2Z*.1X*-.69Y*-.441X*-6.44
X-5.282
*#2Z*.1X*-.69Y*-.441X*-6.43
X-5.282
*#2Z*.1X*-.69Y*-.441X*-.69
N225Y-.59
N230G92Y-6.34
N360G0G90X0Y0T4M6
N370X-5.282Y-.741Z.05
*#2Z*.188X*-6.43Y*-.741X*-.69
X-5.282
*#2Z*.188X*-6.44Y*-.741X*-.69
X-5.282
*#2Z*.188*-.69Y*-.741X*-.69
N415G91Y-5.75
N420G90G92Y-.741
N422X-5.282
*#2Z*.188*-.69*-.741X*-6.44
X-5.282
*#2Z*.188X*-.69Y*-.741X*-6.43
X-5.282
*#2Z*.188X*-.69Y*-.741X*-.69
N425Y-.59
N430G92Y-6.34
N440G0G90X0Y0M2
```

Example #5

1. Nest variable (loop inside macro).

2. Set Dwell time.

3. Position away from first hole (in "Y").

4. (10) Rows of holes.

5. (12) columns of holes.

6. Top row of parts.

7. Bottom row of parts.

8. Set"Y" to bottom row of parts.

9. Reset"Y"back to top row of parts.

```
  6.34
+  .59
  5.75  Incremental Y
        Move Between
        Rows of Parts

  6.43
 -.74
  5.69
 -.408
  5.282 Abs X Coordinate
        of Last Column
        (Each Part)
```

Figure 6-1 (Sheet 1 of 2) Nested Loop Example

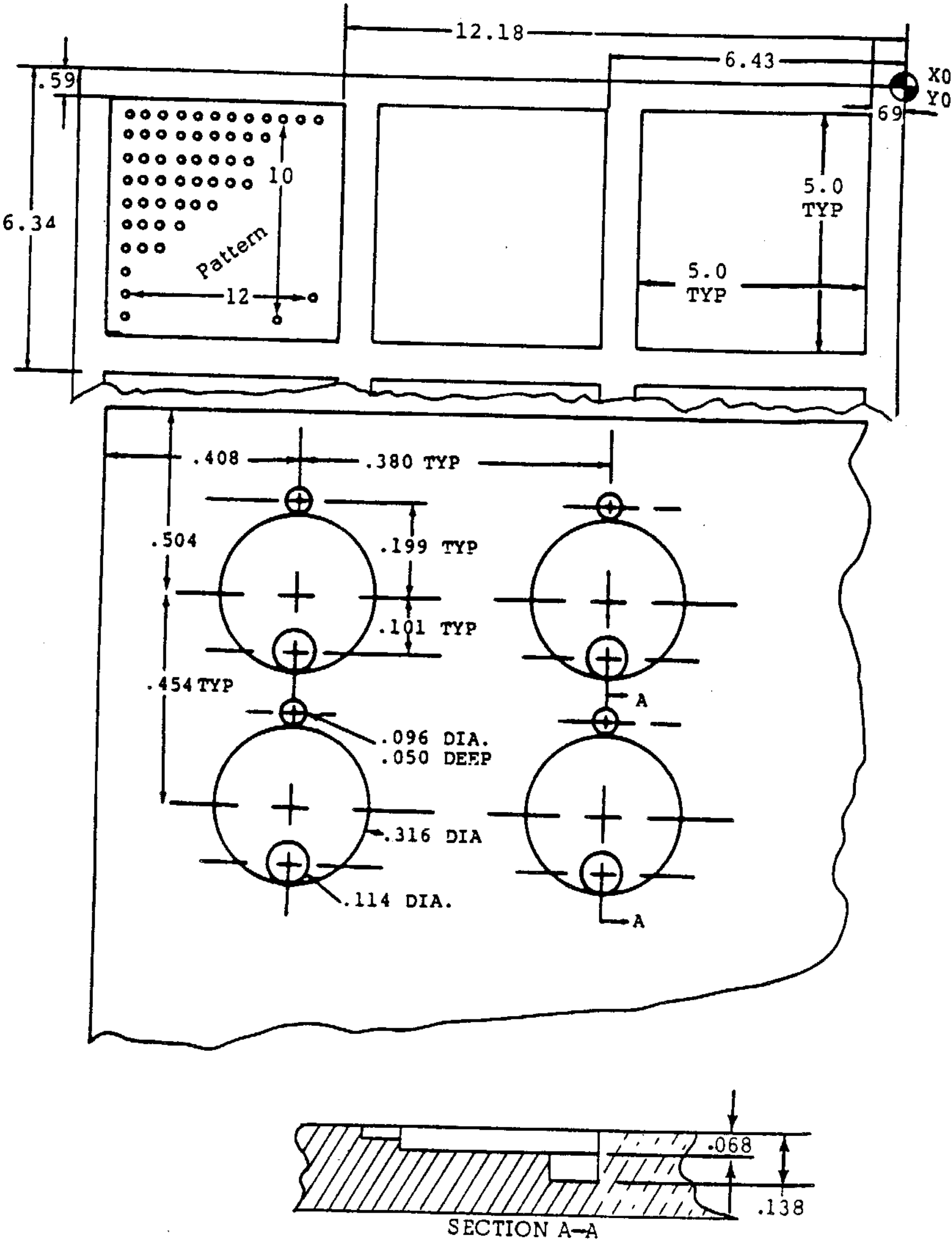
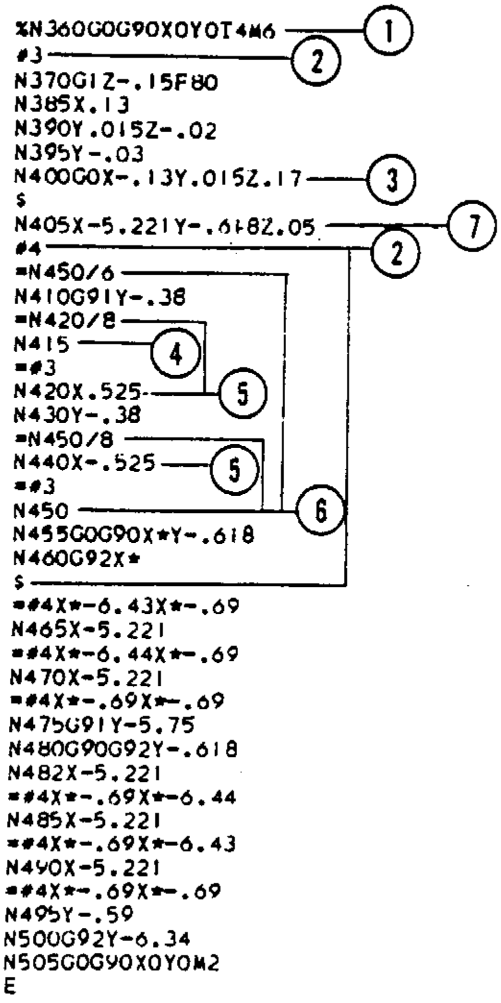


Figure 6-1 (Sheet 2 of 2) Nested Loop Example

Example #6



```
%N360G0G90X0Y0T4M6
#3
N370G1Z-.15F80
N385X.13
N390Y.015Z-.02
N395Y-.03
N400G0X-.13Y.015Z.17
$
N405X-5.221Y-.618Z.05
#4
=N450/6
N410G91Y-.38
=N420/8
N415
=#3
N420X.525
N430Y-.38
=N450/8
N440X-.525
=#3
N450
N455G0G90X*Y-.618
N460G92X*
$
=#4X*-6.43X*-.69
N465X-5.221
=#4X*-6.44X*-.69
N470X-5.221
=#4X*-.69X*-.69
N475G91Y-5.75
N480G90G92Y-.618
N482X-5.221
=#4X*-.69X*-6.44
N485X-5.221
=#4X*-.69X*-6.43
N490X-5.221
=#4X*-.69X*-.69
N495Y-.59
N500G92Y-6.34
N505G0G90X0Y0M2
E
```

1. Rewind stop with first line of data.

2. Macro definition nest inside program

3. Rapid Z on block with X & Y. Projection in the part complete

4. Block after loop must have sequence no. **(Rule b Section 6.1.2)**

5. Position in "X" so as to finish first column.

6. Blank block to end loop on.

7. First positioning move

```
 -.59
 -.408
 -.998
 +.380
 -.618   Y Abs (1 space above
                  first row)
-5.690
  +.318
-5.372
  +.151
-5.221   X Abs (last row)
```

Figure 6-2 (Sheet 1 of 2) Macro Call #3 Within a Loop, Within a Loop, Within a Macro #4.

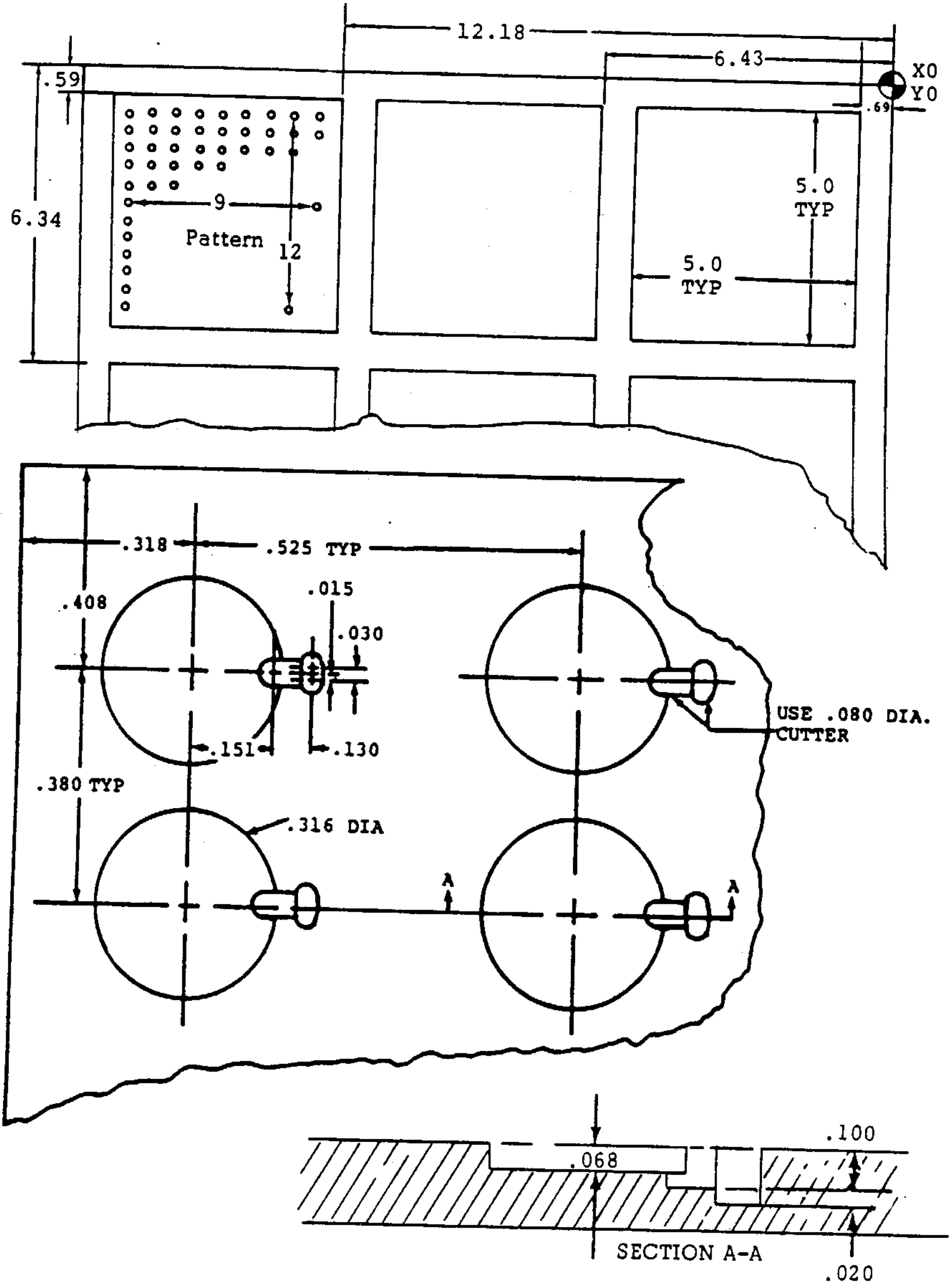


Figure 6-2 (Sheet 2 of 2) Macro Call #3 Within a Loop, Within a Loop, Within a Macro #4.