CHAPTER 3

PROGRAMMING FORMAT

3.1 OVERVIEW

A part program defines a sequence of NC machining operations. The part program consists of a main routine and can contain up to 40 subroutines. Each type of routine contains a number of data blocks. When the E-CNC runs a part program, it reads the part program data blocks sequentially one at a time, interprets the coded information in the block, decides what function to perform and then does the function.

Some examples of executable functions are:

G175	- Mill a circle
G172	- Mill a pocket
G181	- Drill a bolt circle
G04	- Dwell
M08	- Turn the coolant on

The E-CNC part programming language provides the rules for designating machining functions. This section describes how to use the language to program parts on the E-CNC.

3.2 PART PROGRAM STRUCTURE

3.2.1 Blocks

Each part program data block contains a specific machine executable function. Examples shown in this manual will designate the end-of-block code with a semicolon. An example of some part program blocks:

```
N01G90X1.5;
N02G1Z-.5F30.;
N03M09;
```

NOTES

1. The maximum number of characters allowed in a data block is 132, including CR (carriage return) and LF (line feed).

2. The end-of-block code is CR.

3. Characters entered in a program block after a semicolon and before a CR end-of-block will be ignored by the control. This may be used by the programmer for comments except in blocks containing arithmetic expressions.

3.2.2 Words

A data block is composed of one or more words. A word consists of an address followed by a value. The address is a letter that specifies the meaning of the value contained in the word. The addresses and their meanings as used by the E-CNC are as follows:

NOTE

Some address meanings may vary dependent on the G (preparatory) functions specified in the program.

FUNCTION	ADDRESS	FORMAT	MEANING
Program number	:	5	Program number
Subroutine number	#	2	Subroutine number
Label number	L	1	Label number
Sequence number	N	4	Sequence number
Preparatory function	G	3	System mode (linear, arc, etc.)
Coordinate word	X,Y,Z	+3.4	X,Y,Z axis motion command
	I,J,K	+3.4	Arc center coordinate
	U,V,W	+3.4	Incr. X,Y,Z move
	A	+3.3	Polar angular motion
	B	+3.3	Incremental polar angular motion
	R	3.4	Arc radius, corner R
Feed function	F	3.1	Feedrate
Spindle speed function	S	4	Spindle speed
Tool function	T	2	Tool number
Parameter	P	+3.4	Parameters in canned cycles

For example, a block may be composed of the following words:

N100G0X1.Y1.F10.S2400T1M6;

N___	sequence number
G___	preparatory function
X___Y___	coordinate words
F___	feedrate
S___	spindle speed
T___	tool function
M___;	misc. function

NOTES

1. The format description is: +3.4 for motion words except A and B which are 3.3.

   Number of signed value digits to the left of decimal point
   Decimal point is required
   Number of digits to right of the decimal point

2. All formats shown are for inch input. For metric input all formats shown as 3.4 are 4.3.

3. It is not necessary to use the plus sign for positive values.

4. Decimal points are required (except if the E-CNC is run in the BOSS 4-7 compatibility mode). Zeroes to the left of the decimal point and non-significant zeroes to the right of the decimal point may be omitted.

5. Values less than the least significant input will cause an error. For example: X1.23456 is wrong input

6. The sequence of words used in a block is optional except:

   a. If N, the sequence number is used, it must be the first word in the block.

   b. If multiple defined word addresses are used, they must be in the sequence designated for the programmed function.

Maximum Programmable Dimensions

The following lists the maximum programmable dimensions of each address:

NOTE

> These dimensions give the maximum control limit, not the mechanical limit of the NC machine tool. For example, X may be commanded up to 838.8607 inches but the table travel may only be 17.5 inches.

FUNCTION	ADDRESS	RANGE(INCH)	(METRIC)
Program number	:	1-65536	
Label number	L	1-9	
Sequence number	N	1-9999	
Subroutine number	#	1-40	
Preparatory function	G	1-199	
Coordinate word	X,Y,Z,I,J,K U,V,W,R,P	+/-838.8607	+/-8388.607
	A,B	+/-8388.607	
Feedrate	F	.1-100. ipm	2.-2540. mmpm
Spindle speed	S	1-9999	
Tool function	T	1-24	
Misc. function	M	0-99	

3.2.3 Program Number

The E-CNC can store a number of programs. The program number is used to separate one program from another. The format for the program number is: :___

EXAMPLE: :1234; DEFINE PROGRAM NUMBER 1234

Program numbers can be searched for using the front panel FIND feature.

NOTE

> A RESET PROGRAM or M2 (rewind) command resets the part program block pointer to the top of text. If the part program text contains several program numbers, the desired program number must be searched for each time the program is run.

3.2.4 Sequence Number

A sequence number can be specified with up to a 4 digit number (1-9999) following the address N. The order of sequence numbers is arbitrary and need not be consecutive. The same sequence number may be used several times in the program if desired. The sequence number is optional and does not initiate any action from the milling machine. Its main function is for operator convenience and clarity.

Sequence numbers can be searched for using the front panel FIND feature. Sequence numbers can also be used by the operator to establish a BREAK POINT. System operation will stop when a preset breakpoint sequence number is read before its execution; spindle remains ON.

It is recommended that sequence numbers be sequential and used to specify important part program blocks such as a tool change point.

NOTE

The sequence number may contain more than 4 digits. However, only the last 4 digits will be displayed.

3.2.5 Optional Block Delete

When a block has a slash code (/) as the first character and the OPTIONAL/DELETE function has been selected by the operator, all information contained in that data block is ignored by the control. This enables part program blocks to be skipped dependent upon operator interaction.

3.2.6 Definition Blocks

Using a decimal point as the first character in a data block causes the information contained in that block to be executed during a program search. This feature should be used when a system mode is changed during a program so that after the search has been made the system is in the proper mode.

WARNING

BECAUSE DATA CONTAINED IN A DEFINITION BLOCK IS EXECUTED DURING PROGRAM SEARCH, WORDS CAUSING SLIDE MOTION MUST NOT BE PROGRAMMED. PERSONAL INJURY CAN OCCUR.

3.3 PARAMETRIC PART PROGRAMMING LANGUAGE (3PL)

3.3.1 Labels

Labels are used to identify part program blocks such that they may be recalled later in the program. The label number is a single digit number 1-9. If used, the label must be the first word used in the block. The label call will cause part program execution to jump back to the block prefixed by the designated label.

```
EXAMPLE:   L1G0G90X5.;   LABEL DEFINITION
           X0.
           =L1;   LABEL CALL (ENDLELOOP)
```

Labels may be reused as required in the program. The milling and drilling canned cycles are defined as system routines and have labels L8 and L9 inserted in them. Avoid using these label calls when looping back over blocks containing milling and drilling canned cycles.

3.3.2 Arithmetic Expressions

An expression evaluator assigns values for variables that can be used within a part program block instead of a specifically defined value. The format for a variable assignment is:

A___ = expression

Where A is the alphabetic character defining one of the allowable variables used in the system and ___ is any number from 1-9.

The allowable variables are P, X, U, Y, V, Z, W, A, B, R, I, J, K, F. The range for A, B, R, F is 1-4, the range for all others is 1-9.

NOTES

1. R1 is set by the system equal to the current tool radius.

2. Milling and drilling cycles have been canned using arithmetic expressions. When arithmetic expressions are used they must be redefined after each canned cycle.

3. The third character in an expression block must be an "=".

Expressions are a combination of arithmetic operators and elements. The arithmetic operators are as follows:

OPERATOR	MEANING	EXAMPLE
-	unitary minus	-2 meaning - 1*2
*	multiplication	X1*.995
/	division	X1/X2
+	addition	A1+.001
-	subtraction	A2-A1

Additional computations can be specified by using function names. The following functions are available:

SIN___	Sine, ___ is expressed in degrees
COS___	Cosine, ___ is expressed in degrees
SQR___	Square root
TAN___	Tangent, ___ is expressed in degrees
ATN___	Arc tangent, will return expression in degrees
ABS___	Absolute value of an expression
DTF___,___	Evaluates SQR (A1*A1+B1*B1)
#PI___	Equal to 3.1415927

NOTES

1. The result of ATN is a value between -90 and +90 degrees.

2. An arithmetic expression may consist of a single element.

```
EXAMPLE:     -X4
             22.75
```

Or a combination of arithmetic operators, functions and/or values.

```
EXAMPLE:     *1+R1*COS(A1/2.)
```

Arithmetic expressions should be enclosed in parenthesis and considered a basic element to be evaluated before use in the remainder of an expression. This is recommended practice. Within a basic element the order of operator evaluation is unary minus, multiplication, division, addition and subtraction.

```
EXAMPLE:     -P1*SQR2+R1*COS(A1/2.)
             P4-P1/P2+P3
             ((P4-P1)/P2)+P3
             (P4-P1)/(P2+P3)
             P4-(P1/P2)+P3
             P4-(P1/(P2+P3))
```

3.3.3 Variable Substitution

Variables can be substituted for words within a data block. The character defining the variable is used for the word address in the block.

```
EXAMPLE:     X1=1.
             N1X1Y1.0
```

The value equated to X1 will be used as the value for the X word.

NOTES

1. P, X, U, Y, V, Z, W, A, B, R, I, J, K, F words without a decimal point are assumed to be variables. Thus, part programs input for previous versions of BOSS that do not contain decimal points for these

data fields must be run under the BOSS 4-7 compatibility mode.

2. If the range of the variable is greater than 9 an error condition will be flagged.

3.3.4 Conditional Part Program Execution

An IF type statement provides the ability to conditionally execute part program blocks depending on the truth of a conditional expression.

Relational operators are used to compare the value of a variable with another variable or value. The relational operators used are:

Relational Operator	Conditional Test
<GE>	Greater than or equal to
<GT>	Greater than
<LE>	Less than or equal to
<LT>	Less than
<EQ>	Equal to
<NE>	Not equal to

The format of the conditional test data block is:

```
? A___ <relational operator> B___
```

Where A___ is the variable to be tested against B___ and B___ is either the second variable or a value. If the condition is not true, then execution of the following part program blocks up to a block terminated by a ! character will not occur.

Conditionals may be nested as many times as required.

EXAMPLE:

```
X1=.1;          SET X1 EQUAL TO .1
L1G0X1;         MOVE RAPID TO THE VALUE OF THE VARIABLE X1
X1=X1+.01;      INCREMENT X1 BY .01
?X1<LE>1.;      IF X1 LESS OR EQUAL TO 1
=L1!;           JUMP BACK TO THE BLOCK LABELED L1
                ELSE DO NEXT BLOCK
```

NOTE

The expression evaluator handles data in floating point format. The floating point format is precise to 7 decimal digits. For example, multiplying 1000. by .0001 is not precisely equal to .1. Care should be taken when testing for the <EQ> condition.

3.4 MACRO SUBROUTINES

A subroutine is a sequence of part program blocks that define a specific function that may be used several times within a program. The subroutine is called by a single statement within the part program text. Variables may be inserted in the subroutine definition. These variables are assigned in the subroutine call statement. The subroutine definition is called a "macro". The subroutine is executed via a "macro" call command.

3.4.1 Macro Definition

The definition is of the form:

```
#___; START MACRO
part program text
$;END MACRO
```

Where ___ is the macro number with a range of 1 to 40.

Within the macro variable parameters that are to be assigned in the macro call command are designated by an * character.

EXAMPLE:

```
#3;PECK CYCLE
G1Z*F*
G0Z*
Z*
G1Z*F*
Z*
$
```

3.4.2 Macro Call Command

The macro call command is of the form: = ___A*A*....

Where ___ is the called macro number and A* is the variable list in the order that they were designated in the macro.

EXAMPLE: = 3Z*-.5F*10.Z*.5Z*-.5Z*-.2F*20.Z*.7;

NOTES:

1. A macro call may call another macro subroutine.
2. Macros may be nested up to 4 deep.
3. A macro subroutine may include a loop. The macro termination must not be on the loop end block number.
4. A macro call may be included in a loop.
5. Macros with a specified name may be defined and redefined many times within a part program. If two macros with the same tag are specified within a program, the last macro defined will be the one called.
6. A macro <u>CANNOT</u> be defined within another macro.
7. The number of unspecified variables within the macro and the number of assignments in the call command <u>MUST</u> be the same.
8. The macro call statement can contain up to 132 characters. The total number of active macro variables (including nested macro variables) is 12.
9. The maximum input value for a macro variable is determined by the word addressed, refer to Maximum Programmable Dimensions in Section 3.2.2.

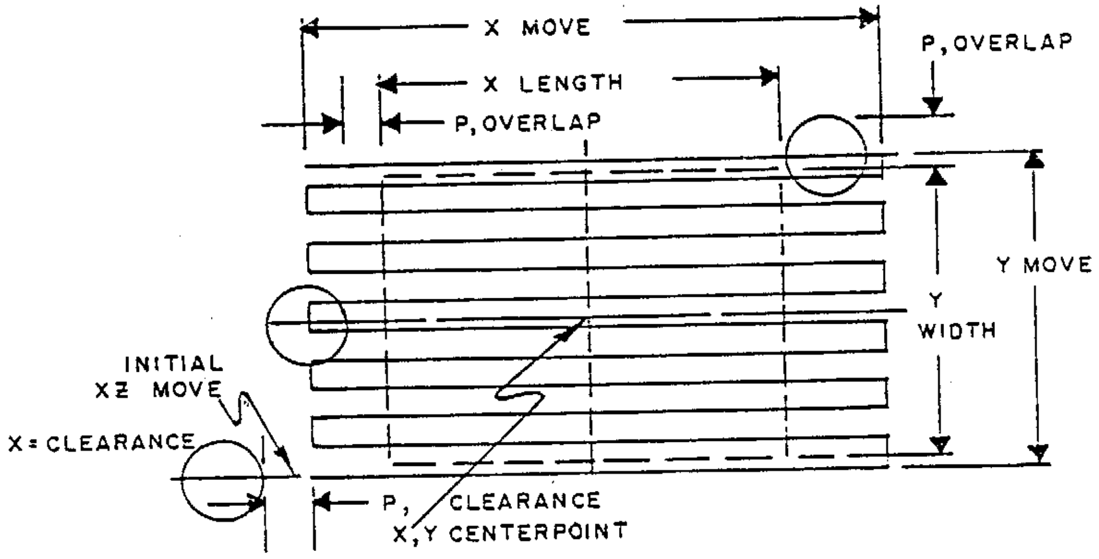
The following parameters may not be used as macro variables:

Macro definition, macro call
Loop call
Cutter compensation values
Tool offset values

Arithmetic expressions and variable substitutions may be used within a macro.

3.5 EXAMPLE FOR THE USE OF THE EXPRESSION EVALUATOR

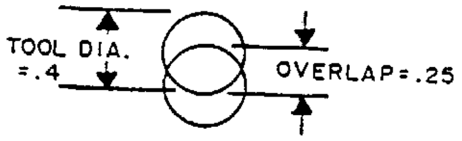
```
;OUTSIDE FACE
;G173X1Y1Z1X2Y2Z2Z3P1P2F1F2
;     X1Y1Z1     ABS START PT
;     X2Y2       XLG, YLG
;     Z2Z3       Z DEPTH, Z STEP
;     R1         TOOL RAD (IMPLICIT VALUE)
;     P1P2       STEPOVER, CLEARANCE
;     F1F2       MILL FEED, PLUNGE FEED

   N2G173X3.Y4.Z-4.X2.Y1.5Z.5Z.5P.25P.2F20.
```



NOTE

Overlap is the distance two consecutive passes overcuts. The center to center tool move = tool diameter overlap.

For the example shown the tool diameter was .4". The center to center move was .15".



NOTE

X move = X length + tool dia + (2*overlap)
Y move = Y width + (2*overlap) - tool dia

```
$29;                       ZIGZAG MACRO
U5=U3                      SET INITIAL ZIGZAG X MOVE
N1G91G1U5F1;               1ST X ZIGZAG MOVE
Y7=0;                      Y STEP SUM (DISTANCE MOVED)
Y6=V3;                     Y RM DIST
L9;                        BEGIN XY ZIGZAG LOOP
?Y6<LT>Y4;                 CHECK FOR REM Y DIST
Y4=Y6!;                    IF Y REM LT Y STEP, SET Y STEP=Y REM
N2Y4;                      MOVE Y STEP
U5=-U5;                    NEGATE X
N3U5;                      MAKE -X ZIGZAG MOVE
Y7=Y7+Y4;                  SET SUM OF Y STEPS
Y6=V3-Y7;                  CALCULATE Y REM DIST
?Y6<GT>0;                  CHECK IF THERE IS STILL Y DIS LEFT
=L9!;                      IF THERE IS REPEAT XY ZIGZAG
$;                         ELSE EXIT
```

```
#4;                     G173 OUTSIDE FACE
P3=R1+R1;               TOOL DIA
U3=X2+P3+P1+P1;         TOTAL X ZIGZAG MOVE
V3=Y3+P1+P1-P3;         TOTAL Y ZIGZAG MOVE
UR=P2+R1;               CLEARANCE MOVE
X3=X1+(-U3/2.)-U4;      X START PT
Y3=Y1+(-V3/2.);         Y START PT
Z1=Z1+.02;              Z CLEARANCE
Y5=P3-P1;               Y STEP
?Y5<LE>0;               CHECK FOR NEG Y STEP
Y5=P1!;                 SET DEFAULT Y STEP=STEPOVER
N10G90G0X3Y3Z1;         GOTO START PT
Z5=0;                   TOTAL Z MOVED (INITIAL Z5=0)
Z4=Z2;                  Z REM DIST (INITIAL Z4=TOTAL Z)
L8;                     BEGIN Z LOOP
?Z4<LT>Z3;              CHECK FOR Z REM
Z3=Z4!;                 IF Z REM LT Z STEP, SET Z STEP=Z REM
N15G91G1Z-.02F2;        PLUNGE Z -.02
W1=-Z3;                 SET INCR Z= -Z STEP
N20U4W1;                PLUNGE Z STEP, X CLEARANCE DIS
Y4=Y5;                  Y4=Y STEP (INITIAL ZIGZAG)
=#29;                   CALL ZIGZAG MACRO
N25Z.02;                MOVE Z UP .02
Z5=Z5+Z3;               CALCULATE TOTAL Z MOVED
Z4=Z2-Z5;               CALCULATE Z REM DIST
?Z4<GT>0;               CHECK FOR Z REM DIST
N28G0G90X3Y3;           IF REM DIST GT 0, RETURN TO START
=L8!;                   AND REPEAT Z LOOP
N30G0G90X1Y1Z1;         ELSE EXIT ZIGZAG ROUTINE
$
```

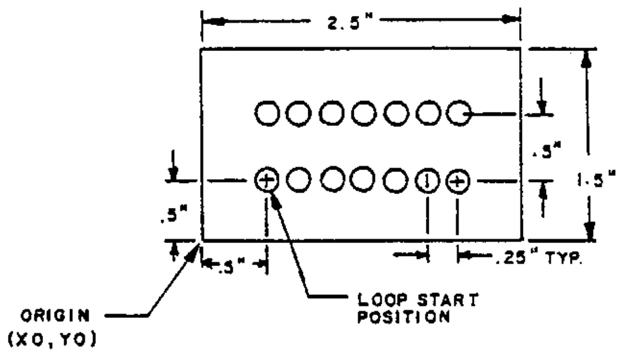
3.6 REPETITIVE PROGRAMMING

Looping enables the programmer to repeat a specified segment of the part program a designated number of times. The loop start command is of the following format: =N___/___

Where N___ is the sequence number of the loop end data block and /___ is the number of times the loop is to be repeated, from 1 to 16,384.

Following a loop start command all part program blocks up to and including the loop end block are executed. After execution of the loop end block, a register initially set with the number of repeats input in the loops start command is decremented by one. If the repeat register is greater than zero, the part program execution will loop back to the part program block following the loop start command. If the repeat register is zero, the looping will end and the part program block following the loop end block will be executed.

EXAMPLE #1, Looping Without Nesting:

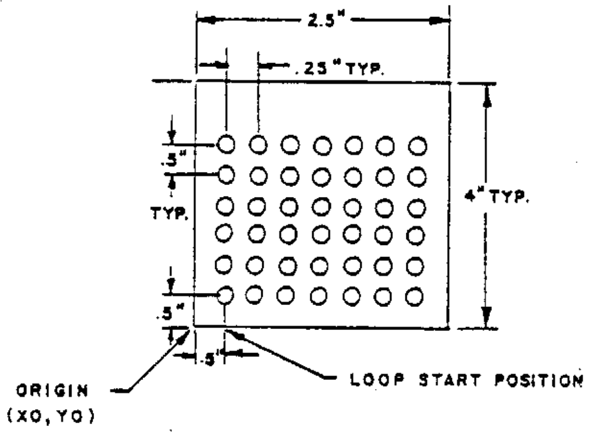


```
N1G0G90X-3.Y0S3300T1M6
N5X.5Y.5Z.05
N10G91G81Y0Z.625F80.
=N15/6
N15X.25
N20Y.5
=N25/6
N25X-.25
N30G0G90X-3.Y0M2
```

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

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.

EXAMPLE #2, Nested Looping:



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

NOTES

1. The loop end block sequence number MUST appear later in the part program.

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