CS 114

CALGEN-AN INTERACTIVE PICTURE-CALCULUS GENERATION SYSTEM

BY

JAMES E. GEORGE

TECHNICAL REPORT NO. CS 114
DECEMBER 1968

COMPUTER SCIENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERSITY

CALGEN - An Interactive Picture Calculus Generation System*

James E. George

May 16, 1968

Submitted as a C. S. 239 Report
to Professor W. F. Miller

* This work was supported by the National Science Foundation under Contract No. GP-7615.

CONTENTS

FIGURES

Calgen-An Interactive Picture Calculus Generation System

I. Introduction

A sub-set of the Picture Calculus [1,2,3] was implemented on the 360/75 to experiment with the proposed data structure, to study the capability of PL/1 for implementing the Picture Calculus and to evaluate the usefulness of drawing pictures with this formalized language. The system implemented is herein referred to as Calgen.

Like many other drawing programs, Calgen utilizes a graphic display console; however, it differs from previous drawing systems in one major area, namely, Calgen retains structure information. Since the Picture Calculus is highly structured, Calgen retains structure information, and only scope images where convenient; further, these scope images saved may be altered by changing the structure information. The only reason scope images are saved by Calgen is to avoid regeneration of a previously generated picture.

II. PDL Syntax and Semantics

The picture Description Language (PDL) [1,2,3] permits a picture to be described by a linear string. In PDL, pictures have two distinguished points, a tail and a head; pictures may be concatenated to other pictures only at these distinguished points. For the sub-set of PDL implemented, a picture, S, is formally defined by:

1. S → P | S <BINOP> S | <UNOP> S | (S) ;
2. p is a primitive;
3. <BINOP> → + | - | x | * ;
4. <UNOP> → ¬ | # ;

where,

A primitive is a basic unit for generation;

<BINOP> are binary concatenation operators;

<UNOP> are unary operators;

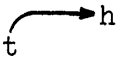
And unary operators have precedence over binary operators.

For a picture formed by concatenation, the following rules determine its tail and head:

1. Tail(S1 <BINOP> S2)=Tail(S1);
2. Head(S1 <BINOP> S2)=Head(S2).

The semantics of the operators are defined by:

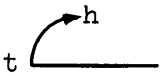
Let S1= 

S2= 

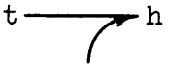
then,

S1+S2= 

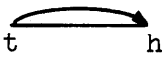
(head to tail)

S1xS2= 

(tail to tail)

S1-S2= 

(head to head)

S1*S2= 

(tail to tail and head to head)

Let S= 

then,

¬ S= 

(the blanking operator)

#S= 

(tail/head reversal)

III. Primitive and Picture Representation

The data structure used for representing primitives in Calgen is a particular instance of the general form proposed by Miller and Shaw [2,3].

PRIMITIVE=(name, basic, string, tail, head, image)

where,

name is a unique alpha-numeric string which identifies a specific primitive

basic is a boolean attribute (true if tail, head and image are present)

string is a well-formed string of PDL from which the tail, head and image can be determined

tail is the tail specification (coordinates)

head is the head specification (coordinates)

image = (number, blank, x, y)

where,

number is the number of coordinates in the image

blank is a boolean array (blank(i) is true if the beam is to be blanked when moving to point(i) else false)

x, y are arrays of the x and y coordinates, respectively.

The same data representation is used to save completed or partially completed pictures. Basic distinguishes between these two cases: if true, then the tail, head and image are available for direct generation; if false, then the picture must be obtained by evaluating its string expression. This feature allows images to be saved if storage is available, otherwise, it allows regeneration. It also provides for formal string manipulation such as replacement and re-evaluation.

For Calgen, pictures are non-basic primitives, hence their name is displayed as a primitive name and then may be selected anywhere a primitive is valid. Unless otherwise noted, by a primitive is meant a basic or non-basic primitive.

IV. Calgen Operators

The Calgen operators consist of the six operators of PDL plus the following control operators:

- = Assigns the evaluation of the first operand to the second operand;
- DIS Displays the graphical representation of the operand;
- RVL Re-evaluates the string expression of the operand (equivalent to operand = operand);
- BSP Pop-up the Result stack;
- DEF Define a primitive (line, line drawing, circle or arc);
- STR Start the program and initialize;
- RST Allow for selection of 1 or 2 scopes;
- DBG Complement the debug switch;
- END Terminate the program;
- ORG Change the translation vector (ORG is selected by typing in ORG and then selecting this with the light pen).

V. Calgen Operation

Initially Calgen defines eight basic primitives (directional north, east, etc.), four non-basic primitives and initializes a translation vector for centering pictures. During this initialization, the user is allowed to rename, set the length and set the translation vector. The default values are:

1. Basic primitives = N, NE, E, SE, S, SW, W, NW;

2. Non-basic primitives = A, B, C, D;

3. Length of primitives = 50 raster units;

4. Translation vector,

x-displacement
y-displacement
> from data cards;

The length specifies the length of N, E, S, W; the lengths of NE, SE, SW, NW are adjusted such that N+E*NE (for example) is valid.

To override the default values for naming, the word "RENAME" is selected with the light pen (to accept the default names depress the end key); each default name is then displayed and the user enters via the keyboard the desired name. At the end of the renaming step the length is requested (the units are raster units); if the default values are to be used simply depress the end key. After the length specification, the displacements are requested (raster units with the origin at lower left); the end key again gives the default values which are input data cards. After the three initialization options, the operators and primitives are displayed and the user is allowed to designate whether an additional scope is to be assigned to function in a slave mode. (See Figure 1.)

The principal cycle of Calgen is:

1. Select an operator with the light pen;

2. Select the necessary operands with the light pen.

Thus, an expression is constructed in pre-fix notation but is converted to normal notation. At each step only legal options are accepted, hence, only well-formed expressions can be constructed. The actual selection is accomplished in the most part using the light pen. When a cycle

has been completed, the resulting expression is displayed as the "RESULT"; "RESULT" actually represents the top of a ten element push-down stack, thereby allowing BSP to be used to delete unwanted construction steps.

For example, to construct the expression N+E*NE the following steps would be executed:

1. Select + with the light pen;
2. " N " " " " ;
3. " E " " " " ;

   At this point RESULT would be updated to N+E;
4. Select DIS with the light pen;
5. " RESULT with the light pen;

   Figure 2 illustrates steps 1 thru 5; T and H designate the tail and head.
6. Select * with the light pen;
7. Select RESULT with the light pen;
8. Select NE with the light pen;

   At this point RESULT would be updated to N+E*NE;
9. Select DIS with the light pen;
10. Select RESULT with the light pen;

    Figure 3 illustrates steps 1-10.

The rules for the selection of operands are:

1. For the binary operators, each operand may be preceded by one unary operator;
2. For = the first operand can be a primitive or RESULT; the second operand must be a primitive;

3. For DIS either primitives or RESULT may be the operand;

4. For BSP, DEF, STR, RST, ORG and END no operands are necessary;

5. For RVL only primitives may be operands.

The rules for inserting parentheses are:

1. If the second operand of a binary operator or the operand for a unary operator is RESULT, then insert parentheses around RESULT;

2. If the RESULT is assigned to a primitive, then parenthesize RESULT before assigning it to "STRING".

Other than initialization and ORG, DEF is the only operator which uses the keyboard; it prompts the user for the information necessary to define a primitive. Lines, line figures, circles or circular arcs may be defined. To define "L" as L' the following steps are necessary:

1. Select DEF with the light pen;

2. Enter name; (See Figure 4)

3. Select line mode by end key;

4. Enter number of points (See Figure 5);

5. Enter tail x-y coordinates (See Figure 6);

6. Enter head x-y coordinates (See Figure 7);

7. Enter coordinates of first point (See Figure 8);

8. " " " second " ;

9. " " " third " ;

10. " " " fourth " (See Figure 9-- B means to blank the beam when moving to this point);

11. " " " fifth " .

"L" is displayed in Figure 10. If the tail, head or any point is not accepted then that item is requested immediately again.

To define ARC as a circular arc the following steps are necessary:

1. Select DEF with the light pen;
2. Select CIRCLE with the light pen;
3. Enter the radius in raster units (See Figure 11);
4. Enter the initial angle in degrees (See Figure 12);
5. Enter the final angle in degrees (See Figure 13);
6. Accept or reject the figure using the light pen; reject returns to step 3 (See Figure 14 for the arc defined).

If an item is not acceptable, then it is changed to zero; by using reject one can repeat the data specification until the desired primitive is obtained.

The rules for defining a circle or circular arc are:

1. Tail at the initial angle a;
2. Head at the final angle b;
3. Number of points generated is min(25, abs(b-a)/5).

Various other constructions are given in Figures 15, 16, and 17. Figure 18 is a house which has been assigned to A. Figure 19 is A+A which has been assigned to B. In Figure 20, A has been modified by inserting a window. Figure 21 shows the result of RVL applied to B; RVL re-evaluates the string of its operand which reflects changes in any basic or non-basic primitive components of the string value of the operand.

VI. Calgen Functional Organization

The Calgen system may be divided into four functional groups:

A. Scope Handling;

B. Initialization;

C. String Handling;

D. PDL Expression Evaluation.

VI.A Scope Handling

Before Calgen was implemented, it was necessary to implement PL/1 and assembly language routines which allow PL/1 to utilize the scope as an I/O device. The routines implemented enable the scope to display strings, plot points or lines and input string information via the keyboard or the light pen[4]. For the output of strings, these routines make the scope appear as a printer with implicit and explicit downspace control; for input, the scope appears as an input device with two input options.

VI.B Initialization

Various parameters, standard messages and devices are initialized when the system first starts execution. These include the assignment of the various scopes, the initialization of tables to be used in parsing PDL expressions and the definition of primitives and items related explicitly to the display on the scope (See section V).

VI.C String Handling

String handling in one of the critical operating functions of the system. PDL expressions are constructed and represented as a linear string. As an expression is formed it is temporarily saved in a push-down stack of ten elements (RESULT); this stack facilitates the BSP

or pop-up operation for the deletion of incorrect steps. When the assign operator (=) is used, the string expression is assigned to the string attribute of the target primitive. By restricting the allowable inputs during the selection cycle, only well-formed PDL expressions can be constructed, hence error recovery in the parser is not so critical.

When a PDL expression is evaluated (by DIS or =) it must be modified so that only basic primitives appear in the expression which is actually evaluated. This is accomplished by a procedure which substitutes the value of the string attribute for non-basic primitives until all primitives in the expression are basic; the parentheses which surround the string attribute are necessary for proper evaluation of the resulting string. Circular substitutions are prevented by monitoring the length of this resultant expression.

VI.D PDL Expression Evaluation

The evaluation of a PDL expression to obtain the graphical representation is accomplished by a simple precedence parser and semantic interpretation system[5]. Several modifications were made in SARPSIS as a result of applying it to the evaluation of PDL expressions. These modifications are:

1. The recognizer for SARPSIS (i.e. for the Parser) scans the input text for the next syntactical unit; it assumes that there are two kinds of syntactical units, numbers and words. These are defined by:

<number> ::= + <num> | - <num> | <num>

<num> ::= <integer> | <integer> . <integer> | <integer> . | . <integer>

<word> ::= if enclosed by QUOTES then [any string of characters not containing a blank or QUOTES and not a <number>] else [any string of characters not containing (a one character reserved word or a blank) and not a <number>]

A separator, thus has the following characteristics:

a. If contained in QUOTES then a blank or QUOTES;

b. Else a blank or a one character reserved word.

2. An additional option was implemented to remove redundant productions (ex. productions which are used to preserve the simple precedence and have a null semantic interpretation). A redundant production is one which satisfies the following restrictions:

a. There are only two symbols in the production (e.g. A::=B);

b.

1. B is <number> or <word>; or

2. The first character of A is "_"; or

3. The first character of B is not "_"; or

4. B is not a basic symbol.

If the redundant option is requested on the option card (by 'REDUNDANT'), then the production number corresponding to redundant productions in the PRTB table is set to a number smaller than the number of productions (in PRTB production numbers are negative).

An additional change was made in the parser to check the

range of the production number returned from PRTB and to only call the semantic routine if it is in the proper range.

The syntax actually used by the parser and semantic system is given in Figure 22. Figure 23 lists the redundant and non-redundant productions of this syntax. Figure 24 shows the semantic procedure calls without the redundant feature; Figure 25 shows the calls with the redundant feature. Thus, for a typical PDL expression, the calls to the semantic routine will be decreased by a factor of two by the redundant feature.

The semantics for the non-redundant productions are (See program listing of SEMANT in the Appendix for details):

1.	statement ::= expr ;	Display the graphic representation and save this in PIC if possible
2.	expr- ::= expr- + term	Translate term according to "+" rule and merge expr- and term
4.	expr- ::= expr- - term	Translate term according to "-" rule and merge
5.	expr- ::= expr- x term	Translate term according to "x" rule and merge
6.	expr- ::= expr- * term	Translate term according to "*" rule; if tails and heads match then merge else error
9.	term ::= ¬ factor	Blank graphic representation of factor
10.	term ::= # factor	Switch tail and head of factor
12.	factor ::= (expr)	Assign the graphic representation for expr to factor

13.	primitive ::= id	Retrieve the graphic structure for id and assign it to primitive
16.	_id ::= _id word	Concatenate right side and assign to left
17.	_id ::= _id number	Concatenate right side and assign to left

VII. Calgen Implementation

With the exception of some of the scope handling routines, Calgen was programmed in PL/1. For the most part (i.e. excluding system bugs for which suitable alternative solutions were found), the string processing was simple and straight-forward. However, list processing or a very flexible storage allocation feature is needed to properly handle the graphic structures; in Calgen, these are handled as arrays which results in individual size constraints. For primitives which require a small amount of storage, the remaining part of the array is not accessible for assignment to other primitives.

Calgen runs on the 360/75 with IBM-2250 graphic displays. The program is overlayed to fit into the 100k scope partition. The overlay structure is given in Figure 26 and consists of a resident root section, three regions and twenty-two branches. For the most part, the overlaying is transparent to the user.

VIII. Program Descriptions

The procedures will be briefly discussed and divided into several functional divisions; listings of all procedures are contained in the Appendix or in the references.

VIII.A Scope Handling

The scope driver routines DISPLAY, CONV, DNSPAC, CONTMO, PLOTL, GINIT, STRSML and GETDAT are discussed in Reference 4.

NAME	ACTION
GINITA	Moves the beam to the lower left corner; used after plotting to separate future string messages from the plot.
TAILHD	Writes T and H at the tail and head, respectively.
BASICP	Plots a primitive and adds T/H using TAILHD.

VIII.B Initialization

NAME	ACTION
OP	Initializes the primitives (N, E, etc.), sets length, calls SCOPINIT.
SCOPINIT	Initializes displacement vector.
SETUP	Initializes and outputs parameters for PARSER using TABREAD and TABPRINT
TABREAD	Reads the data needed by the parser to parse.
TABPRINT	Prints parser data.

VIII.C String Handling

NAME	ACTION
OPA	Fetches operand or unary operator and operand using OPB.
OPB	Fetches operand.
RESD	Update result display and push-down result stack.
EXPA	Expand PDL expression until only basic primitives remain.

VIII.D PDL Expression Evaluation

NAME	ACTION
PARSER	Parses the input string according to the syntax tables, and calls the semantic procedure SEMANT where necessary.
SEMANT	Performs the semantic interpretation according to the syntactical parse; displays and leaves the graphic representation in PIC if possible.

VIII. E Command Functions

These procedures are those which provide the logic and control for the operators selected on the scope.

NAME	ACTION
BINOP	Call for operands and update result for all binary operators using RESD and OPA.
UNOP	Call for operand and update result for all unary operators using RESD and OPB.
BSP	Pop-up result stack and update result using RESD
EQUALS	Obtain the operands using OPB, expand the source string using EXPA, generate the graphic display using PARSER and assign the string and the graphic structure to target if possible using ASSIGN.
RVL	Obtain an operand using OPB, expand the string using EXPA, generate the graphic display using PARSER and assign the graphic structure in PIC to the operand if possible using ASSIGN.
DIS	Obtain the operand using OPB, expand the string using

NAME	ACTION
	EXPA and generate the graphic display using PARSER, or BASICP.
DEF	Define a line or circle type primitive using LINE or CIRCLE and display it using BASICP

VIII.F Primitive Modifying Procedures

NAME	ACTION
ASSIGN	Assigns the graphic structure in PIC to a primitive.
CIRCLE	Obtain the parameters for a circle or arc, generate the graphic structure and assign it to a primitive using CIRCLE_GEN.
CIRCLE_GEN	Generate the graphic structure for a circle or arc and assign it to a primitive.
LINE	Obtain the necessary vectors and assign these to a primitive.

VIII.G Main Control Program

The main control program is GENERAT which initializes some common messages, assigns the scopes and decodes the operations requested; those requested operations are satisfied by calling the appropriate procedures.

IX. Conclusion

As Calgen was implemented, several general items were noticed which will apply to any implementation of an algebraic graphical language. These items were:

A. The language needs a string capability equivalent to that in PL/1;

B. A variable assortment of primitive types is needed such as vectors, circles, arcs, curves and alphabetic;

C. Automatic scaling and translation is needed to fit the area available on the display device;

D. Sub-string replacement and matching is desirable;

E. Powerful storage management is needed for the primitives.

An important item was that this version of PDL is quite cumbersome to use for pictures with a high degree of connectivity, thereby encouraging further investigation into more powerful algebraic type graphical languages.

REFERENCES

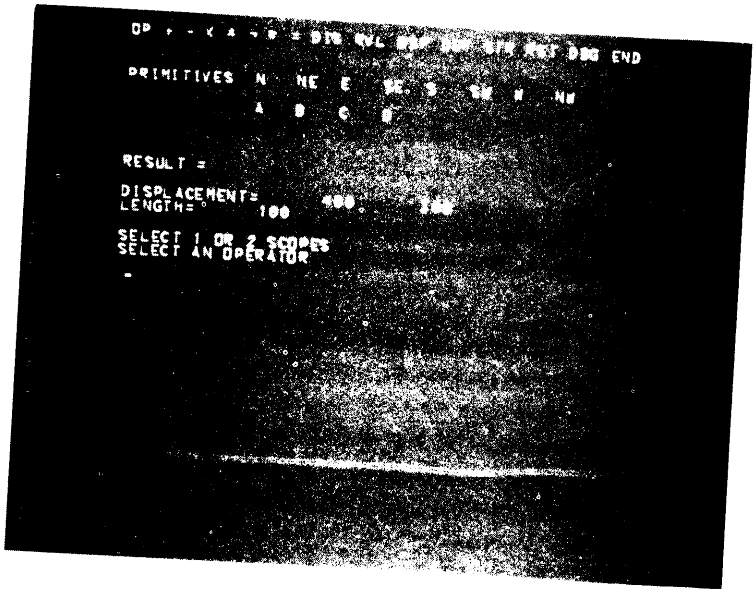
1. Alan C. Shaw, "A Proposed Language for the Formal Description of Pictures", GSG Memo 28, February 1967, SLAC Computation Group.

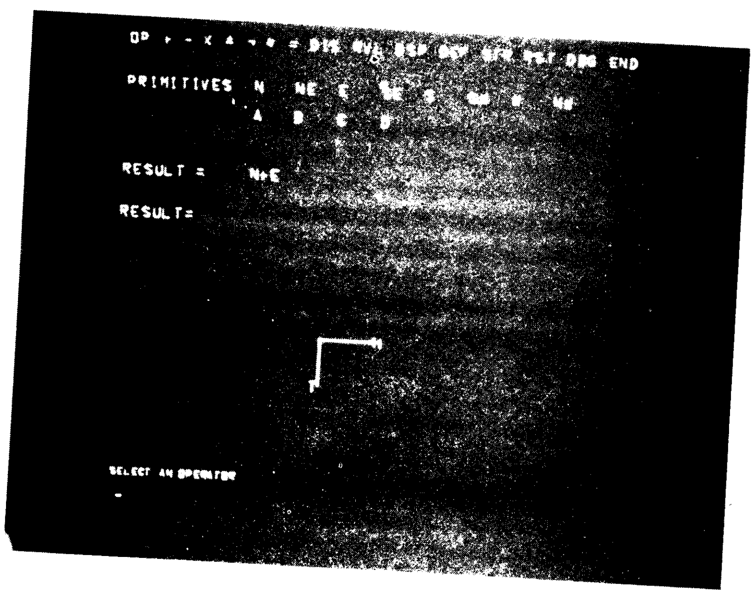
2. W.F. Miller and Alan C. Shaw, "A Picture Calculus", GSG Memo 40, June 21, 1967, SLAC Computation Group.

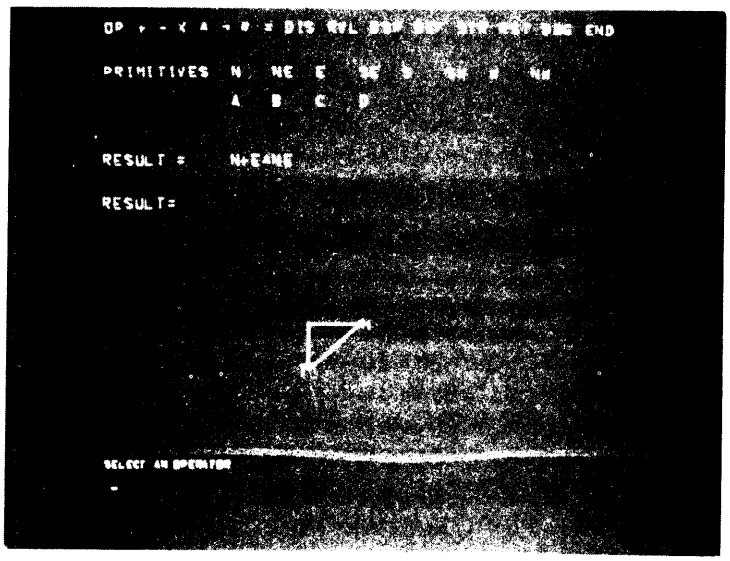
3. Alan C. Shaw, "The Formal Description and Parsing of Pictures", Technical Report No. CS 94, April 5, 1968, Computer Science Department, Stanford University.

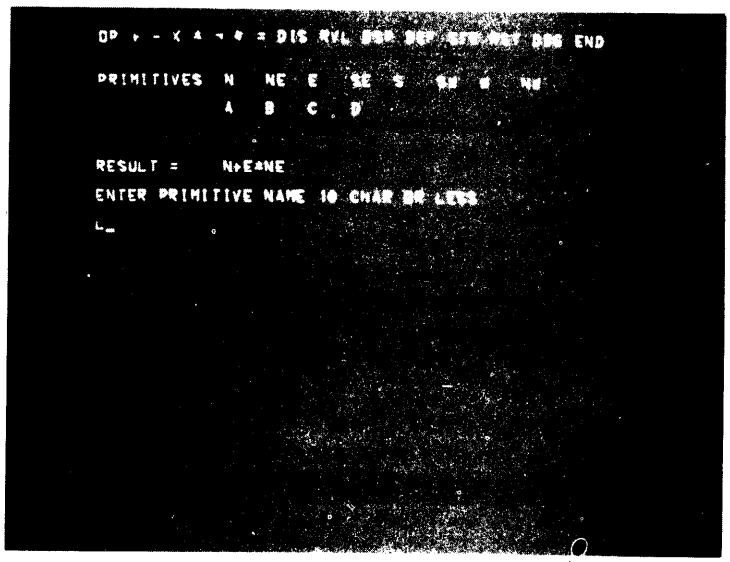
4. J.E. George, "PL/1 Input/Output Procedures for the IBM 2250 Model 2 Scope", SLAC Program Library No. A056; "PL/1 Procedures for Displaying Character Strings and Plotting on the IBM 2250 Model 2 Scope", SLAC Program Library No. A057.

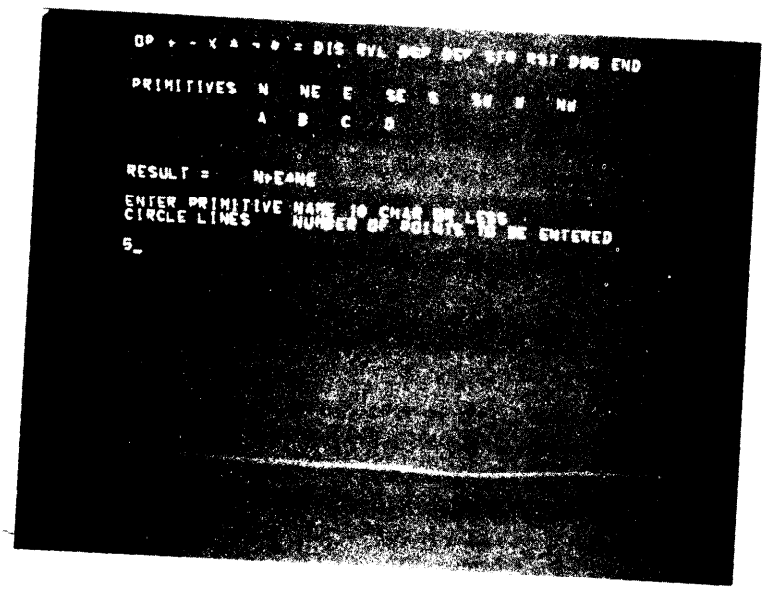
5. J.E. George, "SARPSIS: Syntax Analyzer, Recognizer, Parser and Semantic Interpretation System", CGTM 34, September 29, 1967, SLAC Computation Group.

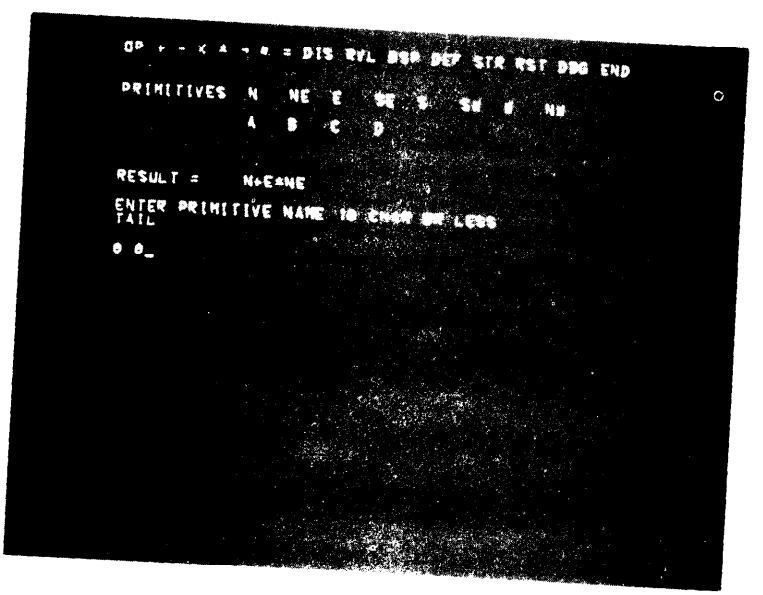

END
PRIMITIVES N NE E
A B C
RESULT =
DISPLACEMENT=
LENGTH=
100
SELECT 1 OR 2 SCOPES
SELECT AN OPERATOR

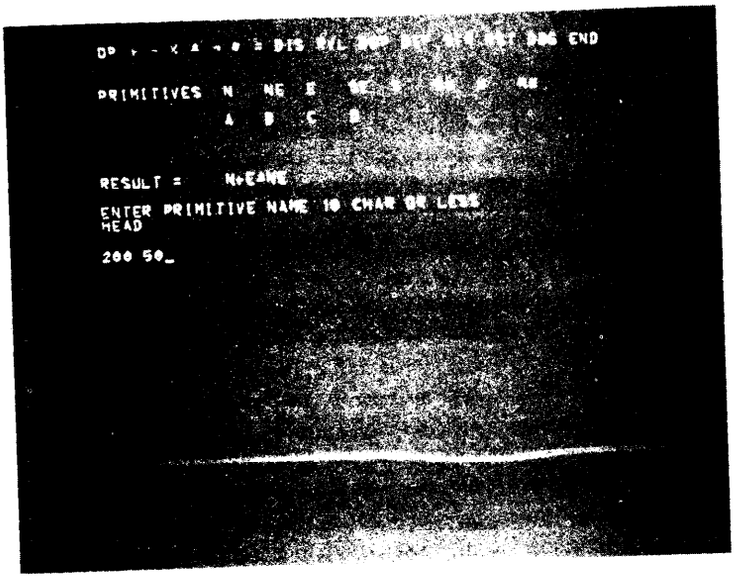

END
PRIMITIVES N NE E
A B
RESULT = N+E
RESULT=
SELECT AN OPERATOR

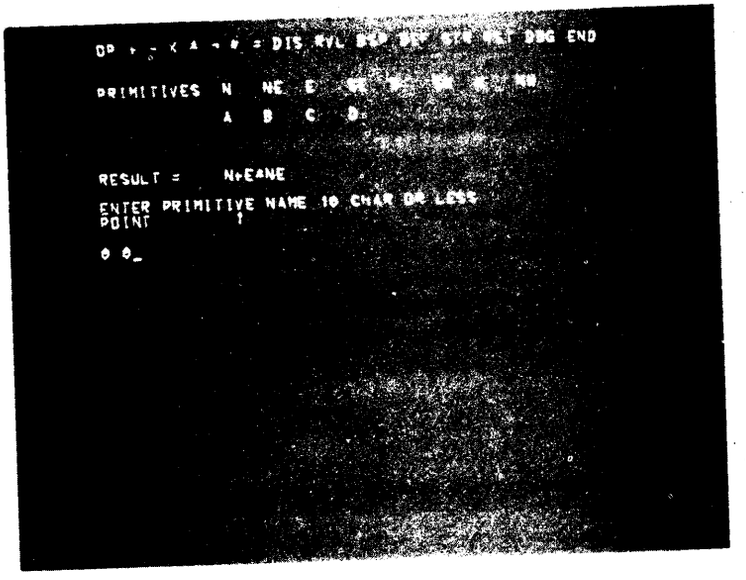

PRIMITIVES N NE E SE S SW W NW
A B C D
RESULT = N+E=NE
RESULT=
SELECT AN OPERATOR

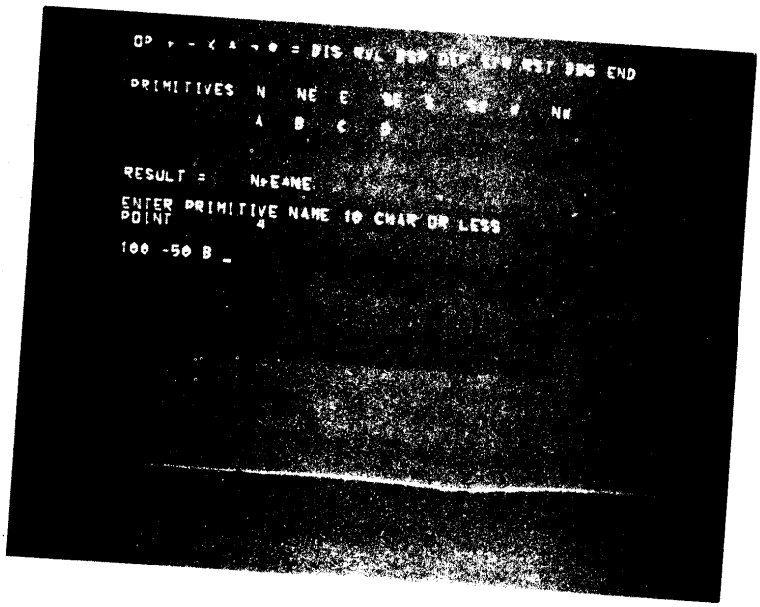

PRIMITIVES N NE E SE S SW W NW
A B C D
RESULT = N+E=NE
ENTER PRIMITIVE NAME 10 CHAR OR LESS

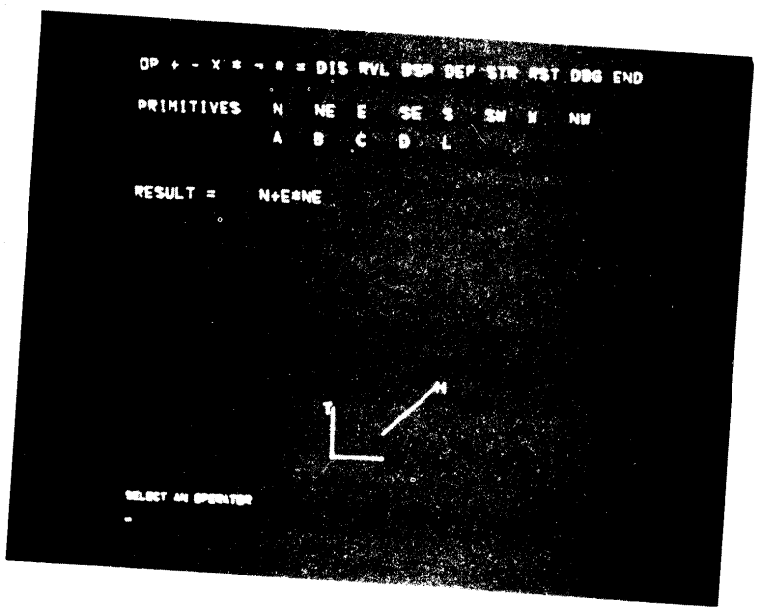

OP + - X A → # = DIS RVL DSP DEF STR RST DBG END
PRIMITIVES N NE E SE S SW W NW
A B C D
RESULT = N+E+NE
ENTER PRIMITIVE NAME TO CHAR OR LESS
CIRCLE LINES NUMBER OF POINTS TO BE ENTERED
5_


OP + - X A → # = DIS RVL DSP DEF STR RST DBG END
PRIMITIVES N NE E SE S SW W NW
A B C D
RESULT = N+E+NE
ENTER PRIMITIVE NAME TO CHAR OR LESS
TAIL
0 0_


DP + - X A → # = DIS RVL
PRIMITIVES N NE E
A B C
RESULT = N+E4NE
ENTER PRIMITIVE NAME 10 CHAR OR LESS
HEAD
200 50_


DP + - X A → # = DIS RVL
PRIMITIVES N NE E
A B C
RESULT = N+E4NE
ENTER PRIMITIVE NAME 10 CHAR OR LESS
POINT
0 0_


OP + - X A → @ = DIS RVL DSP DEF STR RST DBG END
PRIMITIVES N NE E SE S SW W NW
A B C D
RESULT = N+E*NE
ENTER PRIMITIVE NAME 10 CHAR OR LESS
POINT
100 -50 B


OP + - X * → @ = DIS RVL DSP DEF STR RST DBG END
PRIMITIVES N NE E SE S SW W NW
A B C D L
RESULT = N+E*NE
SELECT AN OPERATOR

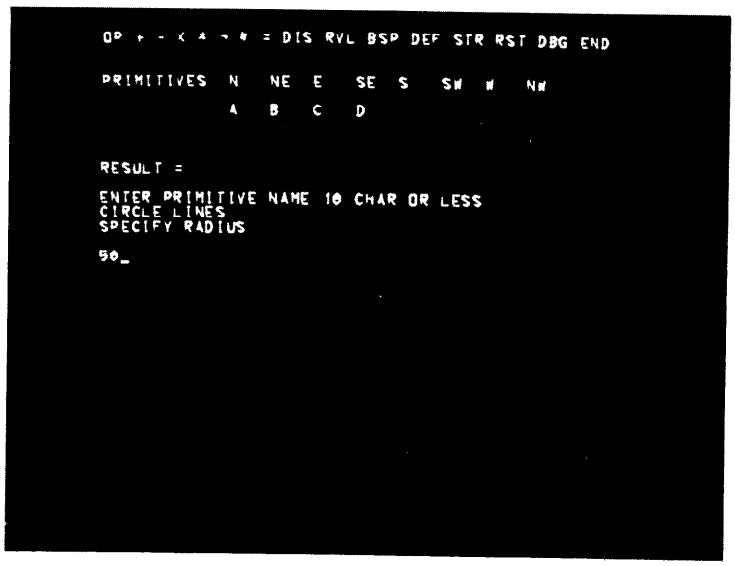


Figure 11

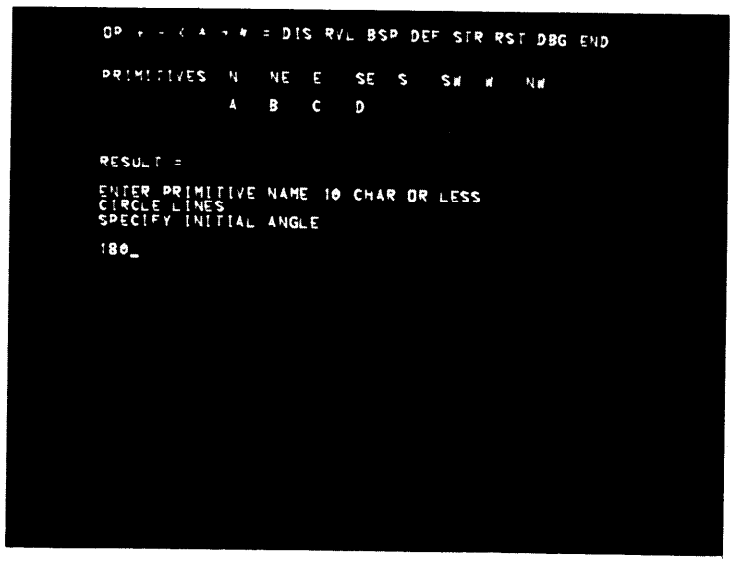
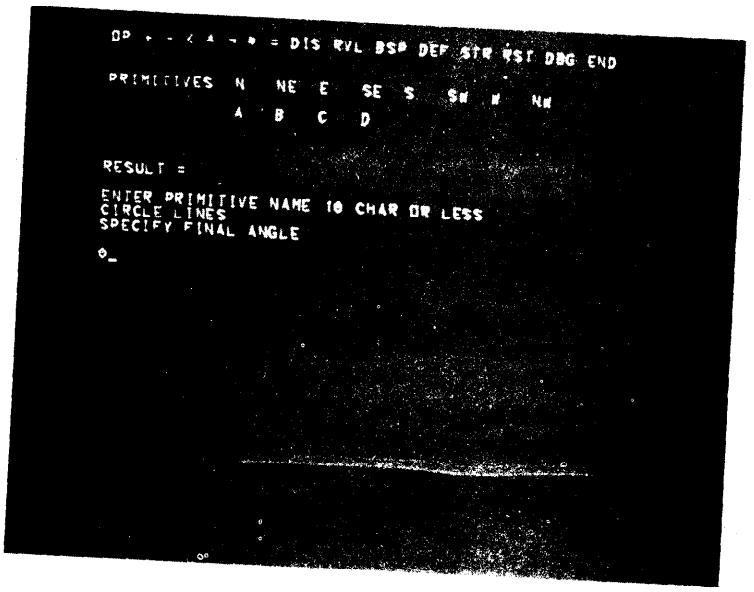
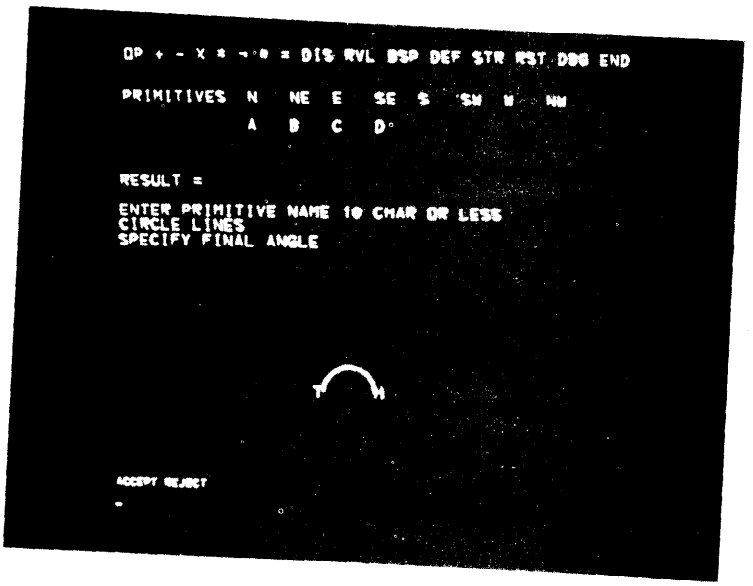
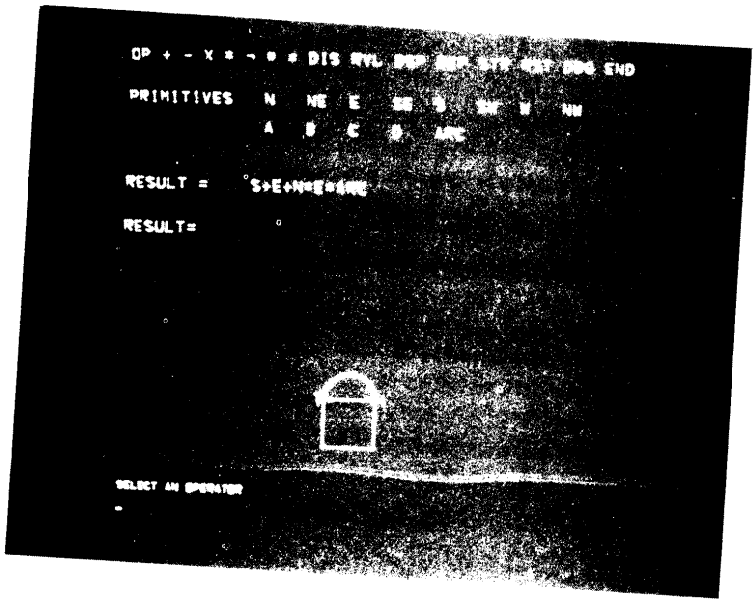
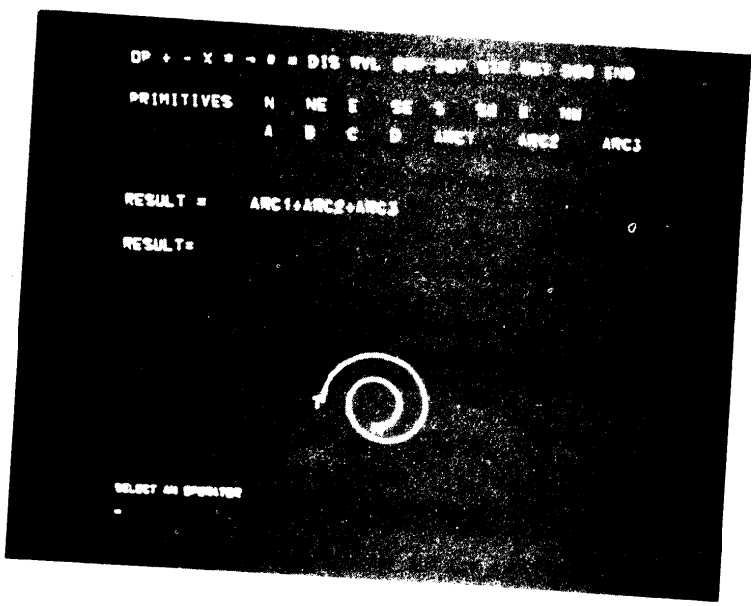


Figure 12


OP + - X * ¬ ↑ = DIS RVL BSP DEF STR RST DBG END
PRIMITIVES N NE E SE S SW W NW
A B C D
RESULT =
ENTER PRIMITIVE NAME 10 CHAR OR LESS
CIRCLE LINES
SPECIFY FINAL ANGLE


OP + - X * ¬ ↑ = DIS RVL BSP DEF STR RST DBG END
PRIMITIVES N NE E SE S SW W NW
A B C D
RESULT =
ENTER PRIMITIVE NAME 10 CHAR OR LESS
CIRCLE LINES
SPECIFY FINAL ANGLE
ACCEPT REJECT


OP + - X ÷ → θ ≠ DIS
PRIMITIVES
RESULT =
RESULT=
SELECT AN OPERATOR


OP + - X ÷ → θ ≠ DIS
PRIMITIVES
ARC1
ARC2
ARC3
RESULT =
RESULT=
SELECT AN OPERATOR

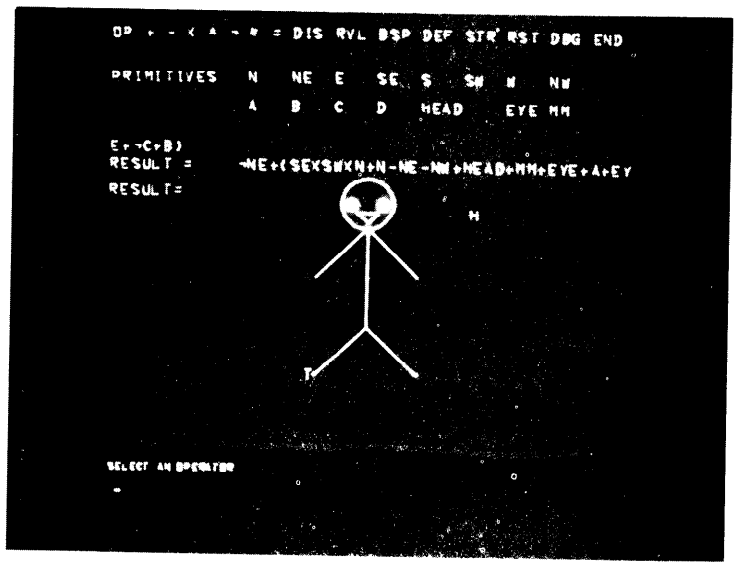


Figure 17

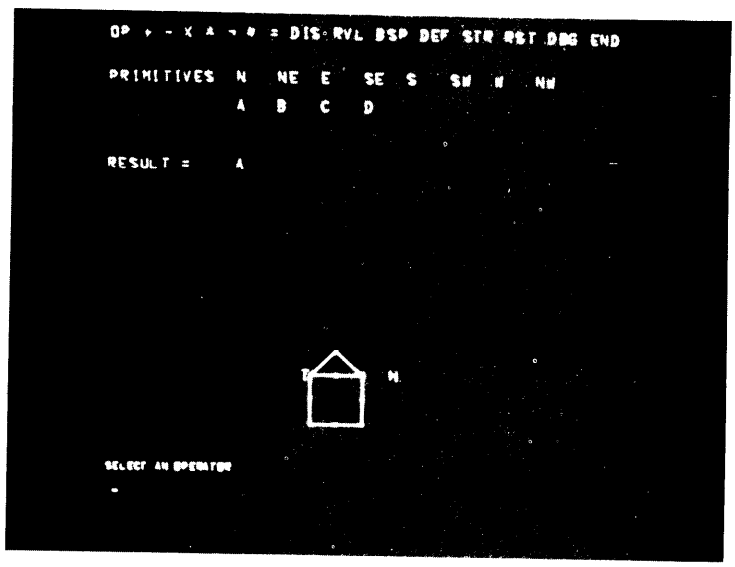
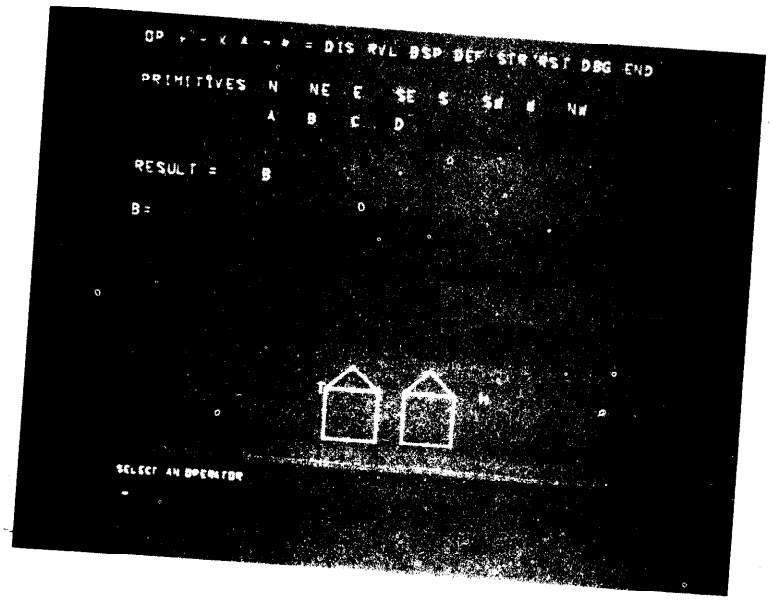
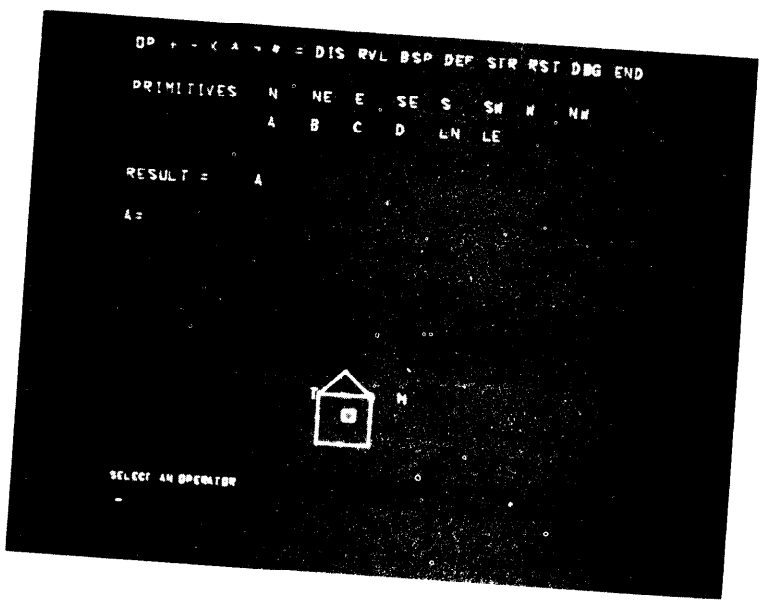


Figure 18


OP + - X A → # = DIS RVL BSP DEF STR RST DBG END
PRIMITIVES N NE E SE S SW W NW
A B C D
RESULT = B
B=
SELECT AN OPERATOR


OP + - X A → # = DIS RVL BSP DEF STR RST DBG END
PRIMITIVES N NE E SE S SW W NW
A B C D LN LE
RESULT = A
A=
SELECT AN OPERATOR

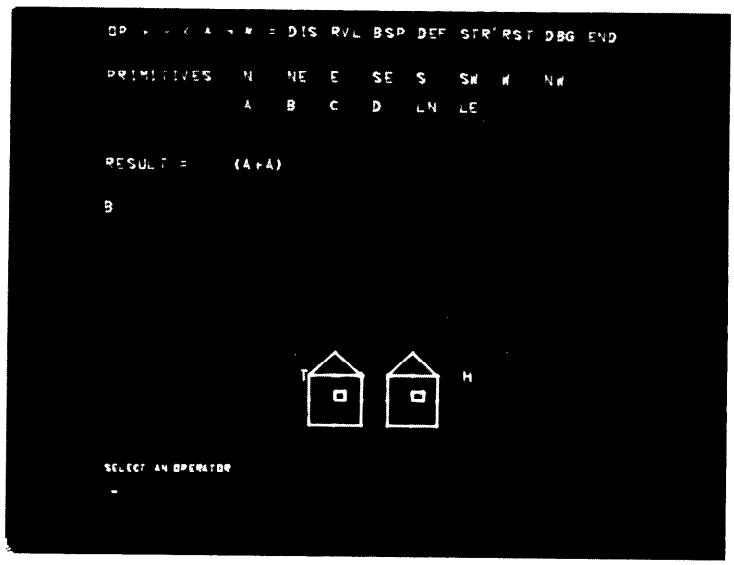

OP + - < A > = DIS RVL BSP DEF STR'RST DBG END
PRIMITIVES N NE E SE S SW W NW
A B C D LN LE
RESULT = (A+A)
B
H
SELECT AN OPERATOR
-

Figure 21

PRODUCTIONS

```
 1    STATMENT      ::=  EXPR        ;
 2    EXPR          ::=  EXPR-
 3    EXPR-         ::=  EXPR-       +       TERM
 4                  ::=  EXPR-       -       TERM
 5                  ::=  EXPR-       X       TERM
 6                  ::=  EXPR-       *       TERM
 7                  ::=  TERM
 8    TERM          ::=  FACTOR
 9                  ::=  ¬           FACTOR
10                  ::=  #           FACTOR
11    FACTOR        ::=  PRIMITIVE
12                  ::=  (           EXPR    )
13    PRIMITIVE     ::=  _ID
14    _ID           ::=  WORD
15                  ::=  NUMBER
16                  ::=  _ID         WORD
17                  ::=  _ID         NUMBER
```

Figure 22

Syntax

REDUNDANT PRODUCTIONS

```
 2  EXPR       ::=  EXPR-
 7  EXPR-      ::=  TERM
 8  TERM       ::=  FACTOR
11  FACTOR     ::=  PRIMITIVE
14  _ID        ::=  WORD
15  _ID        ::=  NUMBER
```

NON-REDUNDANT PRODUCTIONS

```
 1  STATMENT   ::=  EXPR      ;
 3  EXPR-      ::=  EXPR-     +      TERM
 4  EXPR-      ::=  EXPR-     -      TERM
 5  EXPR-      ::=  EXPR-     X      TERM
 6  EXPR-      ::=  EXPR-     *      TERM
13  PRIMITIVE  ::=  _ID
16  _ID        ::=  _ID       WORD
17  _ID        ::=  _ID       NUMBER
 9  TERM       ::=  ¬         FACTOR
10  TERM       ::=  #         FACTOR
12  FACTOR     ::=  (         EXPR       )
```

Figure 23

```
START OF PARSING


                                        A+B-CXD*F↑¬G-#F; END
FORMULA     14  <                   >
FORMULA     13  <                   >
FORMULA     11  <                   >
FORMULA      8  <                   >
FORMULA      7  <                   >
FORMULA     14  <                   >
FORMULA     13  <                   >
FORMULA     11  <                   >
FORMULA      8  <                   >
FORMULA      3  <                   >
FORMULA     14  <                   >
FORMULA     13  <                   >
FORMULA     11  <                   >
FORMULA      8  <                   >
FORMULA      4  <                   >
FORMULA     14  <                   >
FORMULA     13  <                   >
FORMULA     11  <                   >
FORMULA      8  <                   >
FORMULA      5  <                   >
FORMULA     14  <                   >
FORMULA     13  <                   >
FORMULA     11  <                   >
 CRMULA      8  <                   >
 CRMULA      6  <                   >
FORMULA     14  <                   >
FORMULA     13  <                   >
FORMULA     11  <                   >
FORMULA      9  <                   >
FORMULA      3  <                   >
FORMULA     14  <                   >
FORMULA     13  <                   >
FORMULA     11  <                   >
FORMULA     10  <                   >
FORMULA      4  <                   >
FORMULA      2  <                   >
FORMULA      1  <                   >
```

Figure 24

```
START OF PARSING

                                                A+B-(XD*F+¬G-#F; END
FORMULA     13  <                                    >
FORMULA     13  <                                    >
FORMULA      3  <                                    >
FORMULA     13  <                                    >
FORMULA      4  <                                    >
FORMULA     13  <                                    >
FORMULA      5  <                                    >
FORMULA     13  <                                    >
FORMULA      6  <                                    >
FORMULA     13  <                                    >
FORMULA      9  <                                    >
FORMULA      3  <                                    >
FORMULA     13  <                                    >
FORMULA     10  <                                    >
FORMULA      4  <                                    >
FORMULA      1  <                                    >
```

Figure 25

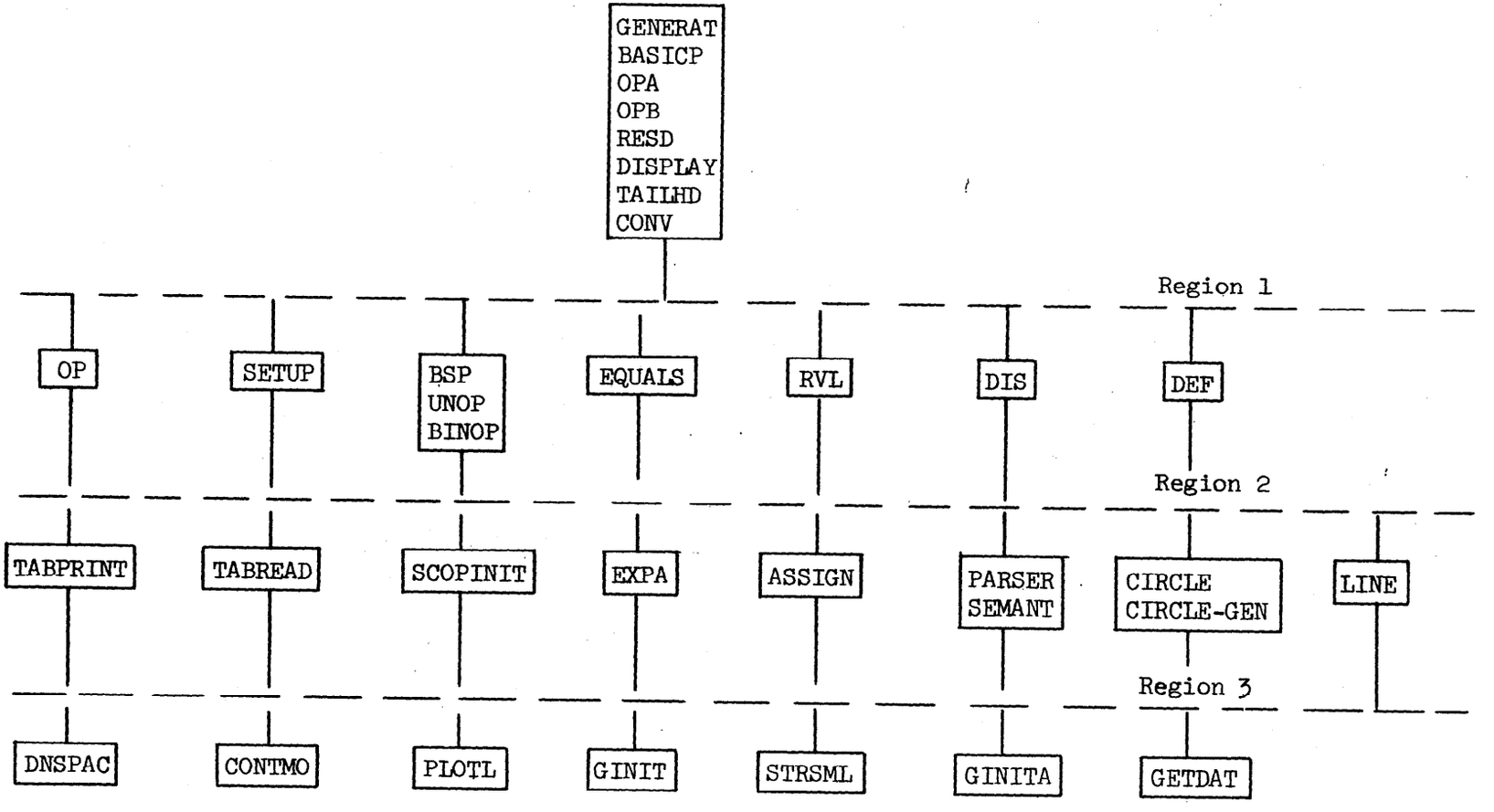


Figure 26. Overlay Structure

APPENDIX

PROGRAM GENERAT

```
GENERAT:   PROC OPTIONS(MAIN);
        /*  MAIN PROGRAM---DECODES SELECTED OPERATORS                    */
        DCL N FIXED BINARY EXT;
        DCL (MS,MM,XI,YI,PI,PNTI,STI,MX,XWORD,XNUM,XSEQ) FIXED BIN EXT;
            GET LIST(N,MS,MM,PI,PNTI,STI,XI,YI);
            PUT LIST(N,MS,MM,PI,PNTI,STI,XI,YI) SKIP;
        BEGIN;
        DCL (A1,A2,A3,A4,A5,A6,A7) CHAR(1) EXT;
        DCL M1 CHAR(72) VAR EXT;
        DCL QUOTES CHAR(12) EXT;
        DCL 1 PIC,
            2 CNT FIXED BINARY,
            2 TX FIXED BINARY,
            2 TY FIXED BINARY,
            2 HX FIXED BINARY,
            2 HY FIXED BINARY,
            2 BL(PNTI) BIT(1),
            2 X(PNTI) FIXED BINARY,
            2 Y(PNTI) FIXED BINARY;
        DCL (BASVAL(MS),PRTB(0:5*N),KEY(0:MS),G(0:MS),F(0:MS))
              FIXED BINARY;
        DCL BASSYM(MS) CHAR(12);
        DCL 1 PRIM,
            2 NUM FIXED BINARY,
            2 BASIC(PI) BIT(1),
            2 NAM(PI) CHAR(10) VAR,
            2 VAL(PI) CHAR(72) VAR,
            2 POINTS(PI),
                3 COUNT FIXED BINARY,
                3 TAILX FIXED BINARY,
                3 TAILY FIXED BINARY,
                3 HEADX FIXED BINARY,
                3 HEADY FIXED BINARY,
                3 COOD   ,
                    4 BLANK(PNTI) BIT(1),
                    4 X(PNTI) FIXED BINARY,
                    4 Y(PNTI) FIXED BINARY;
        DCL (M2,M3,M4,M5,M7,M8) CHAR(72) VAR EXT;
        DCL S BIT(1) EXT;
        DCL DBG BIT(1) EXT;
        DCL B CHAR(500) VAR EXT;
        DCL (I,LOC,PRILOC,RESLOC,J) FIXED BINARY EXT;
        DCL (RES(10),M,BUFA,BUFB) CHAR(72) VAR EXT;
        DCL XX CHAR(1);
            DCL XA CHAR(3);
        DCL (UNITA,UNITB,NBYTE,ISTR) FIXED BINARY(31,0);
            DBG='0'B;
     PUT LIST('START OF PIC CALCULUS.................') SKIP;
            GET LIST (A1,A2,A3,A4,A5,A6,A7);
            M1='OP '||A1||' '||A2||' '||A3||' '||A4||' '||A5||' '||A6
               ||' '||A7||' DIS RVL BSP DEF STR RST DBG END   ';
            M2='PRIMITIVES';
            M3='RESULT = ';
            M4='SELECT OPERAND';
            M5='SELECT OPERAND OR UNARY OP';
            M7='SELECTION ERROR';
            M8='SELECT AN OPERATOR';
            CALL SETUP(F,G,KEY,PRTB,BASVAL,BASSYM);
```

```
                CALL WAITR;
                UNITA=2;  UNITB=1;  NBYTE=5000; ISTR=0;
                CALL GASBUF(UNITA,NBYTE);
                    CALL OP(PRIM);
RST:
                I=1;  CALL DNSPAC(I,LOC);
                M='SELECT 1 OR 2 SCOPES';
                CALL STRLAR(M,LOC);
                CALL GETDAT(M,LOC,S);
                IF M='2' THEN DO;
                    CALL GASBUF(UNITB,ISTR);
                    CALL GASBUF(UNITA,ISTR);
                    CALL GDUPLI(UNITB,UNITA,ISTR);
                    END;
                I=1;  CALL DNSPAC(I,LOC);
LOOP:           CALL CONTMO(M8,LOC);
                CALL GETDAT(M,LOC,S);
                IF S THEN GO TO LOOP;
                XX=SUBSTR(M,1,1);  XA=SUBSTR(M,1,3);
                IF XX=A1|XX=A2|XX=A3|XX=A4 THEN CALL BINOP(XX,PRIM);
                   ELSE IF XX=A5|XX=A6 THEN CALL UNOP(XX,PRIM);
                    ELSE IF XX=A7 THEN CALL EQUALS(PIC,PRIM,F,G,KEY,
                          PRTB,BASVAL,BASSYM);
                ELSE IF XA='DEF' THEN CALL DEF(PRIM);
                    ELSE IF XA='DIS' THEN CALL DIS(PIC,PRIM,F,G,KEY,PRTB,
                            BASVAL,BASSYM);
                ELSE IF XA='BSP' THEN CALL BSP;
                ELSE IF XA='STR' THEN CALL OP(PRIM);
                ELSE IF XA='RST' THEN GO TO RST;
                ELSE IF XA='DBG' THEN DBG=¬DBG;
                    ELSE IF XA='RVL' THEN CALL RVL(PIC,PRIM,F,G,KEY,PRTB,
                    BASVAL,BASSYM);
                ELSE IF XA='END' THEN GO TO EXIT;
                    ELSE IF XA='ORG' THEN CALL SCOPINIT;
                I=1;  CALL DNSPAC(I,LOC);
                GO TO LOOP;
EXIT:           CALL GCLOSE;
                CALL GASBUF(UNITB,ISTR);
                CALL GCLOSE;
                END;    /*  MATCHES BEGIN     */
                END GENERAT;
```

PROGRAM BASICP

```
BASICP: PROC(I,LOC,PRIM);
        /*  PLOTS A BASIC PRIMITIVE                                    */
        DCL (XI,YI,PI,PNTI,STI) FIXED BIN EXT;
        DCL 1 PRIM,
            2 NUM FIXED BINARY,
            2 BASIC(PI) BIT(1),
            2 NAM(PI) CHAR(10) VAR,
            2 VAL(PI) CHAR(72) VAR,
            2 POINTS(PI),
                3 COUNT FIXED BINARY,
                3 TAILX FIXED BINARY,
                3 TAILY FIXED BINARY,
                3 HEADX FIXED BINARY,
                3 HEADY FIXED BINARY,
                3 COOD    ,
                    4 BLANK(PNTI) BIT(1),
                    4 X(PNTI) FIXED BINARY,
                    4 Y(PNTI) FIXED BINARY;
        DCL (I,LOC) FIXED BINARY;
        DCL (XT,YT,XH,YH,J) FIXED BINARY STATIC;
        DCL 1 D,
            2 BL (PNTI) BIT(1),
            2 Z(PNTI) FIXED BINARY,
            2 W(PNTI) FIXED BINARY;
            XT=TAILX(I)+XI;
            YT=TAILY(I)+YI;
            XH=HEADX(I)+XI;
            YH=HEADY(I)+YI;
            DO J=1 TO COUNT(I);
                BL(J)=POINTS(I).BLANK(J);
                Z(J)=POINTS(I).X(J)+XI;
                W(J)=POINTS(I).Y(J)+YI;
                END;
            J=PNTI;
            CALL PLOTL(D,COUNT(I),J,LOC);
            CALL TAILHD(XT,YT,XH,YH,LOC);
            END BASICP;
```

PROGRAM OPA

```
OPA:    PROC(A,I,PRIM);
        /*FETCHES OPERAND OR UNARY OP AND OPERAND LEAVES IN A          */
        DCL (XI,YI,PI,PNTI,STI) FIXED BINARY EXT;
        DCL (LOC,PRILOC,RESLOC) FIXED BINARY EXT;
        DCL (A1,A2,A3,A4,A5,A6,A7) CHAR(1) EXT;
        DCL 1 PRIM,
            2 NUM FIXED BINARY,
            2 BASIC(PI) BIT(1),
            2 NAM(PI) CHAR(10) VAR,
            2 VAL(PI) CHAR(72) VAR,
            2 POINTS(PI),
                3 COUNT FIXED BINARY,
                3 TAILX FIXED BINARY,
                3 TAILY FIXED BINARY,
                3 HEADX FIXED BINARY,
                3 HEADY FIXED BINARY,
                3 COOD    ,
                    4 BLANK(PNTI) BIT(1),
                    4 X(PNTI) FIXED BINARY,
                    4 Y(PNTI) FIXED BINARY;
        DCL RES(10) CHAR(72) VAR EXT;
        DCL (M2,M3,M4,M5,M7,M8,M) CHAR(72) VAR EXT;
        DCL S BIT(1) EXT;
        DCL A CHAR(72) VAR;
        DCL X CHAR(1) STATIC;
        DCL I FIXED BINARY;
        DCL TMP CHAR(72) VAR;
LOOP:       I=1;  CALL DNSPAC(I,LOC);
            CALL CONTMO(M5,LOC);
            CALL GETDAT(M,LOC,S);
            IF S THEN GO TO ERR;
            X=SUBSTR(M,1,1);
            IF X=A5|X=A6 THEN DO;
                CALL OPB(A,I,PRIM);
                IF I=0 THEN TMP=X||'('||A||')'; ELSE TMP=X||A;
                A=TMP;
                I=1;
                RETURN;
                END;
            IF M='RESULT' THEN DO;
                I=0;
                TMP=RES(10);   A=TMP;   RETURN;
                END;
            DO I=1 TO NUM;
                IF M=NAM(I) THEN DO;  A=NAM(I);  RETURN; END;
                END;
ERR:        I=1;  CALL DNSPAC(I,LOC);
            CALL CONTMO(M7,LOC);   GO TO LOOP;
            END OPA;
```

PROGRAM OPB

```
OPB:    PROC(A,I,PRIM);
        DCL (LOC,PRILOC,RESLOC) FIXED BINARY EXT;
        DCL (XI,YI,PI,PNTI,STI) FIXED BINARY EXT;
        DCL (M,M1,M2,M3,M4,M5,M7,M8) CHAR(72) VAR EXT;
        DCL 1 PRIM,
            2 NUM FIXED BINARY,
            2 BASIC(PI) BIT(1),
            2 NAM(PI) CHAR(10) VAR,
            2 VAL(PI) CHAR(72) VAR,
            2 POINTS(PI),
                3 COUNT FIXED BINARY,
                3 TAILX FIXED BINARY,
                3 TAILY FIXED BINARY,
                3 HEADX FIXED BINARY,
                3 HEADY FIXED BINARY,
                3 COOD   ,
                    4 BLANK(PNTI) BIT(1),
                    4 X(PNTI) FIXED BINARY,
                    4 Y(PNTI) FIXED BINARY;
        DCL RES(10) CHAR(72) VAR EXT;
        DCL S BIT(1) EXT;
        /*FETCHES AN OPERAND LEAVES IN A  */
        DCL A CHAR(72) VAR;
        DCL I FIXED BINARY;
        DCL TMP CHAR(72) VAR;
LOOP:       I=1;  CALL DNSPAC(I,LOC);
            CALL CONTMO(M4,LOC);
            CALL GETDAT(M,LOC,S);
            IF S THEN GO TO ERR;
            IF M='RESULT' THEN DO;
                I=0;
                TMP=RES(10);   A=TMP;  RETURN;
                END;
            DO I=1 TO NUM;
                IF M=NAM(I) THEN DO;  A=NAM(I);  RETURN; END;
                END;
ERR:        I=1;   CALL DNSPAC(I,LOC);
            CALL CONTMO(M7,LOC);   GO TO LOOP;
            END OPB;
```

PROGRAM RESD

```
RESD:   PROC(M);
        /*UPDATES RESULT AND RES STACK   */
        DCL (LOC,PRILOC,RESLOC) FIXED BINARY EXT;
        DCL RES(10) CHAR(72) VAR EXT;
        DCL I FIXED BINARY STATIC;
        DCL M CHAR(72) VAR;
        DCL TMP CHAR(72) VAR;
            LOC=RESLOC;
            CALL CONTMO(M,LOC);
            I=1;  CALL DNSPAC(I,LOC);
            DO I=1 TO 9;  TMP=RES(I+1); RES(I)=TMP;  END;
            TMP=M;      RES(10)=TMP;
            END RESD;
```

PROGRAM TAILHEAD

```
TAILHD:     PROC(XT,YT,XH,YH,LOC);
/* WRITES TAIL AND HEAD AT CO-OD XT,YT,SH,YH DISPLACES BY 10 */
DCL (XT,YT,XH,YH) FIXED BINARY;
DCL LOC FIXED BINARY;
DCL (XXH,XXT,I) FIXED BINARY STATIC;
DCL M CHAR(72) VAR STATIC;
DCL 1 A STATIC,
    2 BLANK(1) BIT(1),
    2 X(1) FIXED BINARY,
    2 Y(1) FIXED BINARY;
    XXH=XH+10;    XXT=XT-10;
    X(1)=XXT; Y(1)=YT; BLANK(1)='1'B;  I=1;  M='T';
    CALL PLOTL(A,I,I,LOC);
    CALL STRSML(M,LOC);
    CALL PLOTL(A,I,I,LOC);
    CALL STRSML(M,LOC);
    X(1)=XXH; Y(1)=YH;  M='H';
    CALL PLOTL(A,I,I,LOC);
    CALL STRSML(M,LOC);
    CALL PLOTL(A,I,I,LOC);
    CALL STRSML(M,LOC);
    CALL GINITA(LOC);
    END TAILHD;
```

```
PROGRAM                    OP

 OP:      PROC(PRIM);
          /*INITIALIZES AND STARTS DISPLAY                                  */
          DCL M1 CHAR(72) VAR EXT;
          DCL RES(10) CHAR(72) VAR EXT;
          DCL (M2,M3,M4,M5,M7,M8,M) CHAR(72) VAR EXT;
          DCL S BIT(1) EXT;
          DCL (LOC,PRILOC,RESLOC) FIXED BINARY EXT;
          DCL DBG BIT(1) EXT;
          DCL (XI,YI,PI,PNTI,STI) FIXED BINARY EXT;
          DCL 1 PRIM,
               2 NUM FIXED BINARY,
               2 BASIC(PI) BIT(1),
               2 NAM(PI) CHAR(10) VAR,
               2 VAL(PI) CHAR(72) VAR,
               2 POINTS(PI),
                    3 COUNT FIXED BINARY,
                    3 TAILX FIXED BINARY,
                    3 TAILY FIXED BINARY,
                    3 HEADX FIXED BINARY,
                    3 HEADY FIXED BINARY,
                    3 COOD    ,
                       4 BLANK(PNTI) BIT(1),
                       4 X(PNTI) FIXED BINARY,
                       4 Y(PNTI) FIXED BINARY;
          DCL (I,J,K,LN) FIXED BINARY STATIC;
               ON CONVERSION ONSOURCE='0';
               DBG='0'B;
               DO I=1 TO PI;   NAM(I)=''; VAL(I)=' ';  BASIC(I)='0'B;
                       COUNT(I)=0;  END;
               DO I=1 TO 10; RES(I)='';   END;
               DO I=1 TO 8;
                   BASIC(I)='1'B;   COUNT(I)=2;
                   POINTS(I).BLANK(1)='1'B;
                   POINTS(I).BLANK(2)='0'B;
                   POINTS( I ).X(1),POINTS( I ).Y(1)=0;
                   END;
               CALL GINIT(LOC);
               M='RENAME ';
               CALL STRSML(M,LOC);
               CALL GETDAT(M,LOC,S);
                       NAM(1)='N';  NAM(2)='NE'; NAM(3)='E'; NAM(4)='SE';
                       NAM(5)='S';  NAM(6)='SW'; NAM(7)='W';  NAM(8)='NW';
                   NAM(9)='A'; NAM(10)='B'; NAM(11)='C';  NAM(12)='D';
               IF M='RENAME' THEN DO;
                            J=1;  CALL DNSPAC(J,LOC);
                            I=LOC;
                            DO J=1 TO 12;
                                 M=NAM(J);  LOC=I;   CALL CONTMO(M,LOC);
                                 CALL GETDAT(M,LOC,S);
                                 K=INDEX(M,' ');
                                 IF K¬=0 THEN M=SUBSTR(M,1,K-1);
                                 NAM(J)=M;
                                 END;
                            END;
               J=1;   CALL DNSPAC(J,LOC);
               M='LENGTH';      CALL CONTMO(M,LOC);
               CALL GETDAT(M,LOC,S);
               IF M='' THEN LN=50; ELSE LN=M;
```

```
DO I=1,5;  POINTS(I).X(2)=0;  END;
DO I=3,7;  POINTS(I).Y(2)=0;  END;
DO I=1,2,8;  POINTS(I).Y(2)=LN;  END;
DO I=2,3,4;  POINTS(I).X(2)=LN;  END;
DO I=4,5,6;   POINTS(I).Y(2)=-LN;  END;
DO I=6,7,8;  POINTS(I).X(2)=-LN;  END;
DO I=1 TO 8;
    POINTS(I).TAILX=POINTS(I).X(1);
    POINTS(I).TAILY=POINTS(I).Y(1);
    POINTS(I).HEADX=POINTS(I).X(2);
    POINTS(I).HEADY=POINTS(I).Y(2);
    END;
NUM=12;
CALL GINIT(LOC);
CALL STRSML(M1,LOC);
I=2;  CALL DNSPAC(I,LOC);
PRILOC=LOC;
CALL STRSML(M2,LOC);
I=1;
DO WHILE(I<=NUM);
    DO J=I TO MIN(NUM,I+7);
        M=NAM(J)||' ';
        CALL CONTMO(M,LOC);
        END;
    M='    ';   CALL CONTMO(M,LOC);
    J=2;   CALL DNSPAC(J,LOC);
    M=(10)' ';
    CALL CONTMO(M,LOC);
    I=I+8;
    END;
M='    ';   CALL CONTMO(M,LOC);
I=2;  CALL DNSPAC(I,LOC);
CALL CONTMO(M3,LOC);
RESLOC=LOC;
CALL SCOPINIT;
I=1;  CALL DNSPAC(I,LOC);
M='LENGTH='||LN;
CALL CONTMO(M,LOC);
CALL DNSPAC(I,LOC);
END OP;
```

PROGRAM SETUP

```
SETUP:  PROC(F,G,KEY,PRTB,BASVAL,BASSYM);
          /*  SETUP PARSING TABLES    IF FIRST CARD=ANALYZE THEN  */
        /*  INPUT IS BNF SYNTAX    IF TABLE THEN INPUT TABLES FROM  */
          /*  DATA CARDS      */
        DCL N FIXED BIN EXT;
        DCL MS FIXED BIN EXT;
        DCL MM FIXED BINARY EXT;
        DCL COM CHAR(100) VAR STATIC;
        DCL BASSYM(MS) CHAR(12);
        DCL (BASVAL(MS),PRTB(0:5*N),KEY(0:MS),G(0:MS),F(0:MS))
            FIXED BINARY;
          PUT LIST('SETUP') PAGE;
          GET LIST(COM);
          PUT LIST(COM) SKIP;
          IF COM='TABLE' THEN DO;
              CALL TABREAD(F,G,KEY,PRTB,BASVAL,BASSYM);
             CALL TABPRINT(F,G,KEY,PRTB,BASVAL,BASSYM);
              RETURN;   END;
          ELSE IF COM='ANALYZE' THEN DO;
              PUT LIST('ANALYZE NOT IN THIS VERSION')SKIP;
              END;
          ELSE PUT EDIT('INPUT FORMAT ERROR')(SKIP,A);
          END SETUP;
```

PROGRAM BSP

```
BSP:    PROC;
        /*POP UP RESULT STACK   */
        DCL RES(10) CHAR(72) VAR EXT;
        DCL (BUFA,BUFB) CHAR(72) VAR EXT;
        DCL I FIXED BINARY STATIC;
        DCL TMP CHAR(72) VAR;
            BUFA=RES(9);
            DO I=10 TO 3 BY -1; TMP =RES(I-2); RES(I)=TMP; END;
            RES(2)='';
            CALL RESD(BUFA);
            END BSP;
```

PROGRAM UNOP

```
UNOP:   PROC(A,PRIM);
        /*UPDATES RES FOR ALL UNARY OPERATORS                              */
        DCL (XI,YI,PI,PNTI,STI) FIXED BINARY EXT;
        DCL 1 PRIM,
            2 NUM FIXED BINARY,
            2 BASIC(PI) BIT(1),
            2 NAM(PI) CHAR(10) VAR,
            2 VAL(PI) CHAR(72) VAR,
            2 POINTS(PI),
                3 COUNT FIXED BINARY,
                3 TAILX FIXED BINARY,
                3 TAILY FIXED BINARY,
                3 HEADX FIXED BINARY,
                3 HEADY FIXED BINARY,
                3 COOD    ,
                    4 BLANK(PNTI) BIT(1),
                    4 X(PNTI) FIXED BINARY,
                    4 Y(PNTI) FIXED BINARY;
        DCL (M2,M3,M4,M5,M7,M8,M) CHAR(72) VAR EXT;
        DCL (BUFA,BUFB) CHAR(72) VAR EXT;
        DCL A CHAR(1);
        DCL I FIXED BINARY STATIC;
            CALL OPB(BUFA,I,PRIM);
            IF I=0 THEN M=A||'('||BUFA||')';
            ELSE M=A||BUFA;
            CALL RESD(M);
            END UNOP;
```

PROGRAM BINOP

```
BINOP:  PROC(A,PRIM);
        /*UPDATES RES FOR ALL BINARY OPERATORS                          */
        DCL (XI,YI,PI,PNTI,STI) FIXED BINARY EXT;
        DCL 1 PRIM,
            2 NUM FIXED BINARY,
            2 BASIC(PI) BIT(1),
            2 NAM(PI) CHAR(10) VAR,
            2 VAL(PI) CHAR(72) VAR,
            2 POINTS(PI),
                3 COUNT FIXED BINARY,
                3 TAILX FIXED BINARY,
                3 TAILY FIXED BINARY,
                3 HEADX FIXED BINARY,
                3 HEADY FIXED BINARY,
                3 COOD    ,
                    4 BLANK(PNTI) BIT(1),
                    4 X(PNTI) FIXED BINARY,
                    4 Y(PNTI) FIXED BINARY;
        DCL (BUFA,BUFB) CHAR(72) VAR EXT;
        DCL (M2,M3,M4,M5,M7,M8,M) CHAR(72) VAR EXT;
        DCL A CHAR(1);
        DCL I FIXED BINARY STATIC;
            CALL OPA(BUFA,I,PRIM);
            CALL OPA(BUFB,I,PRIM);
            IF I=0 THEN M=BUFA||A||'('||BUFB||')';
            ELSE M=BUFA||A||BUFB;
            CALL RESD(M);
            END BINOP;
```

PROGRAM EQUALS

```
EQUALS: PROC(PIC,PRIM,F,G,KEY,PRTB,BASVAL,BASSYM);
        /*STORES ARG1 INTO ARG2    */
        /* AND STORES PIC IN PRIM IF POSSIBLE   */
        DCL N FIXED BIN EXT;
        DCL MS FIXED BIN EXT;
        DCL MM FIXED BINARY EXT;
        DCL (XI,YI,PI,PNTI,STI) FIXED BIN EXT;
        DCL (A1,A2,A3,A4,A5,A6,A7) CHAR(1) EXT;
        DCL M1 CHAR(72) VAR EXT;
        DCL (XWORD,XNUM,XSEQ) FIXED BIN EXT;
        DCL QUOTES CHAR(12) EXT;
        DCL BASSYM(MS) CHAR(12);
        DCL (BASVAL(MS),PRTB(0:5*N),KEY(0:MS),G(0:MS),F(0:MS))
             FIXED BINARY;
        DCL 1 PRIM,
            2 NUM FIXED BINARY,
            2 BASIC(PI) BIT(1),
            2 NAM(PI) CHAR(10) VAR,
            2 VAL(PI) CHAR(72) VAR,
            2 POINTS(PI),
                3 COUNT FIXED BINARY,
                3 TAILX FIXED BINARY,
                3 TAILY FIXED BINARY,
                3 HEADX FIXED BINARY,
                3 HEADY FIXED BINARY,
                3 COOD   ,
                    4 BLANK(PNTI) BIT(1),
                    4 X(PNTI) FIXED BINARY,
                    4 Y(PNTI) FIXED BINARY;
        DCL 1 PIC,
            2 CNT FIXED BINARY,
            2 TX FIXED BINARY,
            2 TY FIXED BINARY,
            2 HX FIXED BINARY,
            2 HY FIXED BINARY,
            2 BL(PNTI) BIT(1),
            2 X(PNTI) FIXED BINARY,
            2 Y(PNTI) FIXED BINARY;
        DCL RES(10) CHAR(72) VAR EXT;
        DCL (M2,M3,M4,M5,M7,M8,M) CHAR(72) VAR EXT;
        DCL (LOC,PRILOC,RESLOC) FIXED BINARY EXT;
        DCL S BIT(1) EXT;
        DCL (BUFA,BUFB) CHAR(72) VAR EXT;
        DCL DBG BIT(1) EXT;
        DCL B CHAR(500) VAR EXT;
        DCL I FIXED BINARY STATIC;
            IF NUM<=0 THEN RETURN;
                CALL UPB(BUFB,I,PRIM);
            IF I¬=0 THEN DO;
                BUFB=VAL(I);
                IF LENGTH(BUFB)>2 THEN BUFB=SUBSTR(BUFB,2,LENGTH(BUFB)
                                -2);
                END;
            CALL RESD(BUFB);
                CALL UPB(BUFB,I,PRIM);
            IF I=0 THEN RETURN;
                    VAL(I)='('||RES(10)||')';
            M=' ';
```

```
        CALL EXPA(RES(10),M,PRIM);
        CALL PARSER(N,MS,MM,XWORD,XNUM,XSEQ,QUOTES,F,G,KEY,
            PRTB,BASVAL,BASSYM,LOC,B,PRIM,OBG,PIC,XI,YI,PI,
            PNTI,STI);
        CALL ASSIGN(I,PIC,PRIM);
    END EQUALS;
```

PROGRAM RVL

```
RVL:    PROC(PIC,PRIM,F,G,KEY,PRTB,BASVAL,BASSYM);
        /* REEVALUATES A PRIMITIVE    */
        DCL N FIXED BIN EXT;
        DCL MS FIXED BIN EXT;
        DCL MM FIXED BIN EXT;
        DCL (XI,YI,PI,PNTI,STI) FIXED BIN EXT;
        DCL (A1,A2,A3,A4,A5,A6,A7) CHAR(1) EXT;
        DCL M1 CHAR(72) VAR EXT;
        DCL (XWORD,XNUM,XSEQ) FIXED BIN EXT;
        DCL QUOTES CHAR(12) EXT;
        DCL BASSYM(MS) CHAR(12);
        DCL (BASVAL(MS),PRTB(0:5*N),KEY(0:MS),G(0:MS),F(0:MS))
            FIXED BINARY;
        DCL 1 PRIM,
            2 NUM FIXED BINARY,
            2 BASIC(PI) BIT(1),
            2 NAM(PI) CHAR(10) VAR,
            2 VAL(PI) CHAR(72) VAR,
            2 POINTS(PI),
                3 COUNT FIXED BINARY,
                3 TAILX FIXED BINARY,
                3 TAILY FIXED BINARY,
                3 HEADX FIXED BINARY,
                3 HEADY FIXED BINARY,
                3 COOD   ,
                    4 BLANK(PNTI) BIT(1),
                    4 X(PNTI) FIXED BINARY,
                    4 Y(PNTI) FIXED BINARY;
        DCL 1 PIC,
            2 CNT FIXED BINARY,
            2 TX FIXED BINARY,
            2 TY FIXED BINARY,
            2 HX FIXED BINARY,
            2 HY FIXED BINARY,
            2 BL(PNTI) BIT(1),
            2 X(PNTI) FIXED BINARY,
            2 Y(PNTI) FIXED BINARY;
        DCL RES(10) CHAR(72) VAR EXT;
        DCL (M2,M3,M4,M5,M7,M8,M) CHAR(72) VAR EXT;
        DCL (LOC,PRILOC,RESLOC) FIXED BINARY EXT;
        DCL S BIT(1) EXT;
        DCL (BUFA,BUFB) CHAR(72) VAR EXT;
        DCL DBG BIT(1) EXT;
        DCL B CHAR(500) VAR EXT;
        DCL I FIXED BINARY STATIC;
            IF NUM<=0 THEN RETURN;
                CALL OPB(BUFB,I,PRIM);
            CALL RESD(VAL(I));
            IF I=0 THEN RETURN;
                    CALL EXPA(VAL(I),M,PRIM);
                CALL PARSER(N,MS,MM,XWORD,XNUM,XSEQ,QUOTES,F,G,KEY,
                    PRTB,BASVAL,BASSYM,LOC,B,PRIM,DBG,PIC,XI,YI,PI,
                    PNTI,STI);
                CALL ASSIGN(I,PIC,PRIM);
            END RVL;
```

PROGRAM DIS

```
DIS:    PROC(PIC,PRIM,F,G,KEY,PRTB,BASVAL,BASSYM);
        /*DISPLAYS SELECTED OPERAND                                    */
        DCL N FIXED BIN EXT;
        DCL MS FIXED BIN EXT;
        DCL MM FIXED BIN EXT;
        DCL (XI,YI,PI,PNTI,STI) FIXED BIN EXT;
        DCL (A1,A2,A3,A4,A5,A6,A7) CHAR(1) EXT;
        DCL M1 CHAR(72) VAR EXT;
        DCL (XWORD,XNUM,XSEQ) FIXED BIN EXT;
        DCL QUOTES CHAR(12) EXT;
        DCL BASSYM(MS) CHAR(12);
        DCL (BASVAL(MS),PRTB(0:5*N),KEY(0:MS),G(0:MS),F(0:MS))
              FIXED BINARY;
        DCL 1 PIC,
            2 CNT FIXED BINARY,
            2 TX FIXED BINARY,
            2 TY FIXED BINARY,
            2 HX FIXED BINARY,
            2 HY FIXED BINARY,
            2 BL(PNTI) BIT(1),
            2 X(PNTI) FIXED BINARY,
            2 Y(PNTI) FIXED BINARY;
        DCL 1 PRIM,
            2 NUM FIXED BINARY,
            2 BASIC(PI) BIT(1),
            2 NAM(PI) CHAR(10) VAR,
            2 VAL(PI) CHAR(72) VAR,
            2 POINTS(PI),
                3 COUNT FIXED BINARY,
                3 TAILX FIXED BINARY,
                3 TAILY FIXED BINARY,
                3 HEADX FIXED BINARY,
                3 HEADY FIXED BINARY,
                3 COOD    ,
                  4 BLANK(PNTI) BIT(1),
                  4 X(PNTI) FIXED BINARY,
                  4 Y(PNTI) FIXED BINARY;
        DCL RES(10) CHAR(72) VAR EXT;
        DCL (M2,M3,M4,M5,M7,M8,M) CHAR(72) VAR EXT;
        DCL (LOC,PRILOC,RESLOC) FIXED BINARY EXT;
        DCL S BIT(1) EXT;
        DCL (BUFA,BUFB) CHAR(72) VAR EXT;
        DCL DBG BIT(1) EXT;
        DCL B CHAR(500) VAR EXT;
        DCL I FIXED BINARY STATIC;
        DCL J FIXED BINARY STATIC;
            CALL OPB(BUFA,I,PRIM);
            CALL RESD(BUFA);
            IF I=0 THEN DO;
                BUFA=RES(10);  BUFB='RESULT=';
                GO TO DISP;
                END;
            ELSE DO;
                BUFB=NAM(I)||'=';  BUFA=VAL(I);   END;
            IF BASIC(I) THEN DO;
                CALL BASICP(I,LOC,PRIM);
                RETURN;
                END;
```

```
DISP:
        CALL EXPA(BUFA,BUFB,PRIM);
        CALL PARSER(N,MS,MM,XWORD,XNUM,XSEQ,QUOTES,F,G,KEY,
            PRTB,BASVAL,BASSYM,LOC,B,PRIM,DBG,PIC,XI,YI,PI,
            PNTI,STII;
    END DIS;
```

```
PROGRAM                    DEF

 DEF:    PROC(PRIM);
         /*DEFINES A NEW PRIMITIVE  */
         /*CIRCLE ARC LINE OR LINE DRAWING                                  */
         DCL 1 PRIM,
             2 NUM FIXED BINARY,
             2 BASIC(PI) BIT(1),
             2 NAM(PI) CHAR(10) VAR,
             2 VAL(PI) CHAR(72) VAR,
             2 POINTS(PI),
                 3 COUNT FIXED BINARY,
                 3 TAILX FIXED BINARY,
                 3 TAILY FIXED BINARY,
                 3 HEADX FIXED BINARY,
                 3 HEADY FIXED BINARY,
                 3 COOD    ,
                     4 BLANK(PNTI) BIT(1),
                     4 X(PNTI) FIXED BINARY,
                     4 Y(PNTI) FIXED BINARY;
         DCL (LOC,PRILOC,RESLOC) FIXED BINARY EXT;
         DCL (M2,M3,M4,M5,M7,M8,M) CHAR(72) VAR EXT;
         DCL (XI,YI,PI,PNTI,STI) FIXED BIN EXT;
         DCL RES(10) CHAR(72) VAR EXT;
         DCL S BIT(1) EXT;
         DCL (I,J,K) FIXED BINARY STATIC;
         DCL NU FIXED BINARY STATIC;
         DCL MM CHAR(10) VAR STATIC;
             M=RES(10);   CALL RESD(M);
 LOOP:       I=1;  CALL DNSPAC(I,LOC);
             M='ENTER PRIMITIVE NAME 10 CHAR OR LESS';
             CALL CONTMO(M,LOC);
             CALL GETDAT(M,LOC,S);
             IF ¬S THEN GO TO ERR;
             DO WHILE(LENGTH(M)>0&SUBSTR(M,1,1)=' ');
                 M=SUBSTR(M,2);   END;
             M=M||' ';
             I=INDEX(M,' ');
             I=MIN(I,11);
             M=SUBSTR(M,1,I-1);
             DO I=1 TO NUM;   IF M=NAM(I) THEN DO;
             NU=I;  GO TO XAB;
                 END;   END;
             NUM=NUM+1;
             IF NUM>PI THEN DO;
                 NUM=PI;
                 I=1;  CALL DNSPAC(I,LOC);
                 M='PRIMITIVES EXHAUSTED';
                 CALL CONTMO(M,LOC);  RETURN;
                 END;
             NU=NUM;
             NAM(NUM)=M;
 XAB:
             VAL(NU )='';
             BASIC(NU )='1'B;
             J=1;  CALL DNSPAC(J,LOC);
             I=LOC;
             M='CIRCLE LINES ';
             CALL CONTMO(M,LOC);
             CALL GETDAT(M,LOC,S);
```

```
                IF M='CIRCLE' THEN CALL CIRCLE(NU,PRIM);
                        ELSE CALL LINE(NU,I,PRIM);
                LOC=PRILOC;
                CALL STRSML(M2,LOC);
                I=1;
                DO WHILE(I<=NUM);
                    DO J=I TO MIN(NUM,I+7);
                        M=' '||NAM(J);
                        CALL CONTMO(M,LOC);
                        END;
                        M='    ';   CALL CONTMO(M,LOC);
                    J=2;   CALL DNSPAC(J,LOC);
                    M=(10)' ';
                    CALL CONTMO(M,LOC);
                    I=I+8;
                    END;
                M='    ';  CALL CONTMO (M,LOC);
                I=2;CALL DNSPAC(I,LOC);
                CALL CONTMO(M3,LOC);
                RESLOC=LOC;
                M=RES(10);
                CALL CONTMO(M,LOC);
                I=4;   CALL DNSPAC(I,LOC);
                        CALL BASICP(NU,LOC,PRIM);
                RETURN;
ERR:            I=1;  CALL DNSPAC(I,LOC);
                CALL CONTMO(M7,LOC);
                GO TO LOOP;
                END DEF;
```

PROGRAM TABPRINT

```
TABPRINT: PROC(F,G,KEY,PRTB,BASVAL,BASSYM);
            /*  PRINT PARSING TABLES   */
          DCL J FIXED BINARY EXT;
          DCL I FIXED BINARY EXT;
          DCL N FIXED BIN EXT;
          DCL MS FIXED BIN EXT;
          DCL MM FIXED BIN EXT;
          DCL (XWORD,XNUM,XSEQ) FIXED BIN EXT;
          DCL QUOTES CHAR(12) EXT;
          DCL MX FIXED BINARY EXT;
          DCL BASSYM(MS) CHAR(12);
          DCL (BASVAL(MS),PRTB(0:5*N),KEY(0:MS),G(0:MS),F(0:MS))
                FIXED BINARY;
              PUT EDIT('TABLE ENTRIES')(PAGE,A);
              PUT EDIT('M=',MX,'N=',N,'MM=',MM)(3(X(6),A,F(6)));
              PUT EDIT('XWORD=',XWORD,'XNUM=',XNUM,
                   'XSEQ=',XSEQ,'QUOTES=',QUOTES)(SKIP,3(X(3),A,F(4)),
                        X(3),2 A);
              PUT EDIT('F')(SKIP,A);
              PUT EDIT((F(J) DO J=0 TO MX+1))
                   (100(SKIP,10(X(3),F(4))));
              PUT EDIT('G')(SKIP,A);
              PUT EDIT((G(J) DO J=0 TO MX+1))
                   (100(SKIP,10(X(3),F(4))));
              PUT EDIT('KEY')(SKIP,A) ;
              PUT EDIT((KEY(J) DO J=0 TO MX+1))
                   (100(SKIP,10(X(3),F(4))));
              PUT EDIT('PRTB')(SKIP,A);
              PUT EDIT((PRTB(J) DO J=0 TO KEY(MX+1)))
                   (100(SKIP,10(X(3),F(4))));
              PUT EDIT('BASVAL')(SKIP,A);
              PUT EDIT((BASVAL(J) DO J=1 TO MX-1-MM))
                   (100(SKIP,10(X(3),F(4))));
              PUT EDIT('BASSYM')(SKIP,A);
            PUT EDIT((BASSYM(J) DO J=1 TO MX-1-MM))(100(SKIP,6(X(5),A)));
              RETURN;
              END TABPRINT;
```

PROGRAM TABREAD

```
TABREAD:  PROC(F,G,KEY,PRTB,BASVAL,BASSYM);
            /*   READ PARSING TABLES    */
          DCL N FIXED BIN EXT;
          DCL MS FIXED BIN EXT;
          DCL MM FIXED BIN EXT;
          DCL (XWORD,XNUM,XSEQ) FIXED BIN EXT;
          DCL QUOTES CHAR(12) EXT;
          DCL BASSYM(MS) CHAR(12);
          DCL (BASVAL(MS),PRTB(0:5*N),KEY(0:MS),G(0:MS),F(0:MS))
                FIXED BINARY;
          DCL MX FIXED BINARY EXT;
          DCL (I,J) FIXED BINARY STATIC;
          DCL COM CHAR(100) VAR STATIC;
              PUT LIST('TABREAD') SKIP;
LOOP:         GET LIST(COM);
              PUT LIST(COM) SKIP;
              IF COM='END' THEN RETURN;
                   IF COM='MNMM' THEN GET LIST(MX,N,MM);
              IF COM='XNUM-XWORD-XSEQ' THEN GET LIST(XNUM,XWORD,XSEQ);
                   IF COM='QUOTES' THEN GET LIST(QUOTES);
                   IF COM='BASSYM' THEN GET LIST((BASSYM(J) DO J=1 TO
                        MX-1-MM));
                   IF COM='BASVAL' THEN GET LIST((BASVAL(J) DO J=1 TO
                    MX-1-MM));
                   IF COM='KEY' THEN GET LIST((KEY(J) DO J=0 TO MX+1));
                   IF COM='PRTB' THEN GET LIST((PRTB(J) DO J=0 TO
                  KEY(MX+1)));
                   IF COM='F' THEN GET LIST((F(J) DO J=0 TO MX+1));
                   IF COM='G' THEN GET LIST((G(J) DO J=0 TO MX+1));
              GO TO LOOP;
              END TABREAD;
```

PROGRAM SCOPINIT

```
SCOPINIT:   PROC;
        /*INITIALIZE DISPLACEMENT VECTOR                                    */
        DCL (LOC,PRILOC,RESLOC) FIXED BINARY EXT;
        DCL RES(10) CHAR(72) VAR EXT;
        DCL (XI,YI,PI,PNTI,SII) FIXED BIN EXT;
        DCL (M,MM) CHAR(72) VAR STATIC;
        DCL J FIXED BINARY STATIC; DCL S BIT(1) STATIC;
            ON CONVERSION BEGIN;
                MM='ERROR IN INPUT ZERO SUBSTITUTED';
                J=1;  CALL DNSPAC(J,LOC);  CALL CONTMO(MM,LOC);
                ONSOURCE='0';
                END;
            J=1;
            M='SPECIFY X DISPLACEMENT';
            CALL DNSPAC(J,LOC);
            CALL CONTMO(M,LOC);
            CALL GETDAT(M,LOC,S);
            IF M¬='' THEN XI=SUBSTR(M,1,4);
            M='SPECIFY Y DISPLACEMENT';
            CALL DNSPAC(J,LOC);
            CALL CONTMO(M,LOC);
            CALL GETDAT(M,LOC,S);
            IF M¬='' THEN YI=SUBSTR(M,1,4);
            M=RES(10);  CALL RESO(M);
            M='DISPLACEMENT='||XI||YI;
            CALL DNSPAC(J,LOC);
            CALL CONTMO(M,LOC);
            END SCOPINIT;
```

PROGRAM EXPA

```
EXPA:   PROC(BUFB,M,PRIM);
        /*EXPANDS BUFB LEAVES ANSWER IN B GLOBAL                    */
        DCL (A1,A2,A3,A4,A5,A6,A7) CHAR(1) EXT;
        DCL 1 PRIM,
            2 NUM FIXED BINARY,
            2 BASIC(PI) BIT(1),
            2 NAM(PI) CHAR(10) VAR,
            2 VAL(PI) CHAR(72) VAR,
            2 POINTS(PI),
                3 COUNT FIXED BINARY,
                3 TAILX FIXED BINARY,
                3 TAILY FIXED BINARY,
                3 HEADX FIXED BINARY,
                3 HEADY FIXED BINARY,
                3 COOD  ,
                    4 BLANK(PNTI) BIT(1),
                    4 X(PNTI) FIXED BINARY,
                    4 Y(PNTI) FIXED BINARY;
        DCL (LOC,PRILOC,RESLOC) FIXED BINARY EXT;
        DCL DBG BIT(1) EXT;
        DCL B CHAR(500) VAR EXT;
        DCL (BUFB,M) CHAR(72) VAR;
        DCL (I,J,K) FIXED BINARY;
        DCL BUFA CHAR(500) VAR ;        DCL (BUFFB CHAR(72))VAR ;
        DCL X CHAR(1) ;
        DCL TMP CHAR(500) VAR;
MINX:   PROC(I,J);
        DCL(I,J) FIXED BINARY;
            IF J=0 THEN RETURN;
            IF J<I THEN I=J;      RETURN;
            END MINX;
            BUFA=BUFB;
            B='';
LOOP:       IF BUFA='' THEN GO TO FINIS;
            X=SUBSTR(BUFA,1,1);
            DO WHILE((X='('|X=')'|X=A1|X=A2|X=A3|X=A4|X=A5|X=A6)
                &LENGTH(BUFA)>0);
                TMP=B||X;   B=TMP;
                IF LENGTH(BUFA)=1 THEN GO TO FINIS;
                BUFA=SUBSTR(BUFA,2);
                X=SUBSTR(BUFA,1,1);
                END;
            I=1000;
            J=INDEX(BUFA,'(');
            CALL MINX(I,J);
            J=INDEX(BUFA,')');
            CALL MINX(I,J);
            J=INDEX(BUFA,A1);
            CALL MINX(I,J);
            J=INDEX(BUFA,A2);
            CALL MINX(I,J);
            J=INDEX(BUFA,A3);
            CALL MINX(I,J);
            J=INDEX(BUFA,A4);
            CALL MINX(I,J);
            J=INDEX(BUFA,A5);
            CALL MINX(I,J);
            J=INDEX(BUFA,A6);
```

```
                CALL MINX(I,J);
                IF I=1000 THEN DO;  BUFFB=BUFA;  BUFA='';  END;
                    ELSE DO;
                        BUFFB=SUBSTR(BUFA,1,I-1);  BUFA=SUBSTR(BUFA,I);
                        END;
                DO J=1 TO NUM;  IF BUFFB=NAM(J) THEN GO TO LL;
                    END;
LL:             IF ¬BASIC(J) THEN DO;
                    IF LENGTH(VAL(J))+LENGTH(BUFA)>500 THEN DO;
                        M='EXPAND BUFFER OVERFLOW';
                        I=1;   CALL DNSPAC(I,LOC);
                        CALL CONTMO(M,LOC);
                        CALL DNSPAC(I,LOC);
                        RETURN;   END;
                    BUFA=VAL(J)||BUFA;   GO TO LOOP;
                    END;
                TMP=B||BUFFB;     B=TMP;
                GO TO LOOP;
FINIS:
                I=2;  CALL DNSPAC(I,LOC);
                CALL CONTMO(M,LOC);
                I=1;
                IF DBG THEN DO;
                    DO J=1 TO LENGTH(B) BY 60;
                    K=MIN(60,LENGTH(B)-J+1);
                    M=SUBSTR(B,J,K);
                    IF M¬='' THEN DO;
                    CALL CONTMO(M,LOC);
                    CALL DNSPAC(I,LOC);
                    END;
                    END;
                    END;
                CALL DNSPAC(I,LOC);
                B=B||';END    ';
                RETURN;
                END EXPA;
```

PROGRAM ASSIGN

```
ASSIGN: PROC(I,PIC,PRIM);
        /*ASSIGN PIC TO PRIM(I) IF PRIM AREA LARGE ENOUGH                */
        DCL (XI,YI,PI,PNTI,STI) FIXED BINARY EXT;
        DCL 1 PIC,
             2 CNT FIXED BINARY,
             2 TX FIXED BINARY,
             2 TY FIXED BINARY,
             2 HX FIXED BINARY,
             2 HY FIXED BINARY,
             2 BL(PNTI) BIT(1),
             2 X(PNTI) FIXED BINARY,
             2 Y(PNTI) FIXED BINARY;
        DCL 1 PRIM,
             2 NUM FIXED BINARY,
             2 BASIC(PI) BIT(1),
             2 NAM(PI) CHAR(10) VAR,
             2 VAL(PI) CHAR(72) VAR,
             2 POINTS(PI),
                3 COUNT FIXED BINARY,
                3 TAILX FIXED BINARY,
                3 TAILY FIXED BINARY,
                3 HEADX FIXED BINARY,
                3 HEADY FIXED BINARY,
                3 COOD   ,
                   4 BLANK(PNTI) BIT(1),
                   4 X(PNTI) FIXED BINARY,
                   4 Y(PNTI) FIXED BINARY;
        DCL K FIXED BINARY STATIC;
        DCL I FIXED BINARY;
             BASIC(I)='0'B;
             IF PIC.CNT<=PNTI THEN DO;
                  BASIC(I)='1'B;
                  POINTS(I).COUNT=PIC.CNT;
                  POINTS(I).TAILX=PIC.TX;
                  POINTS(I).TAILY=PIC.TY;
                  POINTS(I).HEADX=PIC.HX;
                  POINTS(I).HEADY=PIC.HY;
                  DO K=1 TO PIC.CNT;
                       COOD(I).BLANK(K)=PIC.BL(K);
                       COOD(I).X(K)=PIC.X(K);
                       COOD(I).Y(K)=PIC.Y(K);
                       END;
                  END;
             END ASSIGN;
```

PROGRAM CIRCLE

```
CIRCLE: PROC(I,PRIM);
        /*OBTAINS PARAMETERS FOR CIRCLE CR ARC FROM KEYBOARD          */
        DCL 1 PRIM,
            2 NUM FIXED BINARY,
            2 BASIC(PI) BIT(1),
            2 NAM(PI) CHAR(10) VAR,
            2 VAL(PI) CHAR(72) VAR,
            2 POINTS(PI),
                3 COUNT FIXED BINARY,
                3 TAILX FIXED BINARY,
                3 TAILY FIXED BINARY,
                3 HEADX FIXED BINARY,
                3 HEADY FIXED BINARY,
                3 COOD    ,
                    4 BLANK(PNTI) BIT(1),
                    4 X(PNTI) FIXED BINARY,
                    4 Y(PNTI) FIXED BINARY;
        DCL (XI,YI,PI,PNTI,STI) FIXED BIN EXT;
        DCL (LOC,PRILOC,RESLOC) FIXED BINARY EXT;
        DCL I FIXED BINARY;
        DCL (  J,LOCA,N) FIXED BINARY STATIC;
        DCL (RAD,ALPHA,BETA) FLOAT BINARY STATIC;
        DCL (M,MM) CHAR(72) VAR STATIC; DCL S BIT(1) STATIC;
            ON CONVERSION BEGIN;
                MM='ERROR IN INPUT ZERO SUBSTITUTED';
                J=1;  CALL DNSPAC(J,LOC);  CALL CONTMO(MM,LOC);
                ONCHAR='0';
                END;
            J=1;   CALL DNSPAC(J,LOC);
            LOCA=LOC;
X2:         M='SPECIFY RADIUS';
            CALL CONTMO(M,LOC);
            CALL GETDAT(M,LOC,S);
            IF ¬S THEN GO TO X2;
            RAD=SUBSTR(M,1,4);
            LOC=LOCA;
X3:         M='SPECIFY INITIAL ANGLE';
            CALL CONTMO(M,LOC);
            CALL GETDAT(M,LOC,S);
            IF ¬S THEN GO TO X3;
            ALPHA=SUBSTR(M,1,4);
            LOC=LOCA;
X4:         M='SPECIFY FINAL ANGLE';
            CALL CONTMO(M,LOC);
            CALL GETDAT(M,LOC,S);
            IF ¬S THEN GO TO X3;
            BETA=SUBSTR(M,1,4);
            N=ABS((BETA-ALPHA)/5);
            N=MIN(N,25);
            IF N<=1 THEN N=2;
            CALL CIRCLE_GEN(RAD,ALPHA,BETA,N,I,PRIM);
            CALL BASICP(I,LOC,PRIM);
            J=1;   CALL DNSPAC(J,LOC);
            M='ACCEPT REJECT ';
            CALL CONTMO(M,LOC);
            CALL GETDAT(M,LOC,S);
            IF M='ACCEPT' THEN RETURN;
            LOC=LOCA;
```

```
GO TO X2;
END CIRCLE;
```

PROGRAM CIRCLE_GEN

```
CIRCLE_GEN: PROC(RAD,ALPHA,BETA,N,I,PRIM);
        /*CALCULATES CIRCLE PTS FOR PRIM(I) FOR N-1 ARCS FROM ALPHA
                TO BETA     */
        DCL (XI,YI,PI,PNTI,STI) FIXED BINARY EXT;
        DCL 1 PRIM,
            2 NUM FIXED BINARY,
            2 BASIC(PI) BIT(1),
            2 NAM(PI) CHAR(10) VAR,
            2 VAL(PI) CHAR(72) VAR,
            2 POINTS(PI),
                3 COUNT FIXED BINARY,
                3 TAILX FIXED BINARY,
                3 TAILY FIXED BINARY,
                3 HEADX FIXED BINARY,
                3 HEADY FIXED BINARY,
                3 COOD    ,
                    4 BLANK(PNTI) BIT(1),
                    4 X(PNTI) FIXED BINARY,
                    4 Y(PNTI) FIXED BINARY;
        DCL (RAD,ALPHA,BETA,DELTA) FLOAT BINARY;
        DCL (N,I) FIXED BINARY;
        DCL J FIXED BINARY STATIC;
            DELTA=(BETA-ALPHA)/(N-1);
            POINTS(I).BLANK(1)='1'B;
            POINTS(I).X(1)=RAD*COSD(ALPHA);
            POINTS(I).Y(1)=RAD*SIND(ALPHA);
            DO J=2 TO N;
                POINTS(I).BLANK(J)='0'B;
                POINTS(I).X(J)=RAD*COSD(ALPHA+(J-1)*DELTA)
                                -POINTS(I).X(1);
                POINTS(I).Y(J)=RAD*SIND(ALPHA+(J-1)*DELTA)
                                -POINTS(I).Y(1);
                END;
            COUNT(I)=N;
            TAILX(I),POINTS(I).X(1)=0;
            TAILY(I),POINTS(I).Y(1)=0;
            HEADX(I)=POINTS(I).X(N);
            HEADY(I)=POINTS(I).Y(N);
            END CIRCLE_GEN;
```

PROGRAM LINE

```
LINE:   PROC(NU,I,PRIM);
        /*OBTAINS POINTS FOR LINE SEGMENTS FROM KEYBOARD                */
        DCL (XI,YI,PI,PNTI,STI) FIXED BIN EXT;
        DCL M1 CHAR(72) VAR EXT;
        DCL 1 PRIM,
            2 NUM FIXED BINARY,
            2 BASIC(PI) BIT(1),
            2 NAM(PI) CHAR(10) VAR,
            2 VAL(PI) CHAR(72) VAR,
            2 POINTS(PI),
                3 COUNT FIXED BINARY,
                3 TAILX FIXED BINARY,
                3 TAILY FIXED BINARY,
                3 HEADX FIXED BINARY,
                3 HEADY FIXED BINARY,
                3 COOD    ,
                    4 BLANK(PNTI) BIT(1),
                    4 X(PNTI) FIXED BINARY,
                    4 Y(PNTI) FIXED BINARY;
        DCL (M2,M3,M4,M5,M7,M8,M) CHAR(72) VAR EXT;
        DCL (LOC,PRILOC,RESLOC) FIXED BINARY EXT;
        DCL S BIT(1) EXT;
        DCL NU FIXED BINARY;
        DCL I FIXED BINARY;
        DCL (J,K) FIXED BINARY STATIC;
        DCL MM CHAR(10) VAR STATIC;
        DCL DBLANK INTERNAL ENTRY (CHAR(72) VAR) RETURNS (CHAR(72)VAR);
        DCL NUMBER INTERNAL ENTRY(CHAR(10) VAR) RETURNS(BIT(1));
NUMBER: PROC(A) BIT(1);
        /* TRUE IF A IS NUMBER   */
        DCL A CHAR(10) VAR;
        DCL J FIXED BINARY STATIC;
            IF A='' | A=' ' THEN RETURN('0'B);
            J=1;
            DO WHILE(SUBSTR(A,J,1)=' ' & J<LENGTH(A));
                J=J+1;
                END;
            IF SUBSTR(A,J,1)>='0' | SUBSTR(A,J,1)='-' | SUBSTR(A,J,1)=
                              '+' THEN DO;
                 J=J+1;
                DO WHILE(J<= LENGTH(A));
                    IF SUBSTR(A,J,1)<'0' THEN RETURN('0'B);
                    J=J+1;
                    END;
                RETURN('1'B);
                END;
            RETURN('0'B);
            END NUMBER;
CBLANK: PROC(A) CHAR(72) VAR;
        /*DELETES LEADING BLANKS     */
        DCL A CHAR(72) VAR;
        DCL B CHAR(72) VAR STATIC;
            B=A;
            IF B='' | B=' ' THEN RETURN(B);
            DO WHILE(SUBSTR(B,1,1)=' ');
                B=SUBSTR(B,2);
                END;
            RETURN(B);
```

```
                END DBLANK;
X11:            M='NUMBER OF POINTS TO BE ENTERED';
                CALL CONTMO(M,LOC);
X1:             CALL GETDAT(M,LOC,S);
                IF ¬S THEN GO TO X1;
                M=M||' ';
                J=INDEX(M,' ');
                       MM=SUBSTR(M,1,J-1);
                         IF ¬NUMBER(MM) THEN GO TO    X11;
                COUNT(NU )=M;
                IF COUNT(NU)>PNTI THEN GO TO X11;
X2:             M='TAIL';
                LOC=I;
                CALL CONTMO(M,LOC);
                CALL GETDAT(M,LOC,S);
                IF ¬S THEN GO TO X2;
                M=DBLANK(M);
                M=M||'     ';
                J=INDEX(M,' ');
                MM=SUBSTR(M,1,J-1);
                        IF ¬NUMBER(MM) THEN GO TO X2;
                TAILX(NU )=SUBSTR(M,1,J);
                M=SUBSTR(M,J+1);
                M=M||'     ';
                J=INDEX(M,' ');
                 MM=SUBSTR(M,1,J-1);
                    IF ¬NUMBER(MM) THEN GO TO     X2;
                TAILY(NU )=SUBSTR(M,1,J);
X3:             M='HEAD';
                LOC=I;
                CALL CONTMO(M,LOC);
                CALL GETDAT(M,LOC,S);
                IF ¬S THEN GO TO X3;
                M=DBLANK(M);
                M=M||'     ';
                J=INDEX(M,' ');
                M=M||'     ';
                  MM=SUBSTR(M,1,J-1);
                     IF ¬NUMBER(MM) THEN GO TO   X3;
                HEADX(NU )=SUBSTR(M,1,J);
                M=SUBSTR(M,J+1);
                M=DBLANK(M);
                M=M||' ';
                J=INDEX(M,' ');
                   MM=SUBSTR(M,1,J-1);
                        IF ¬NUMBER(MM) THEN GO TO X3;
                HEADY(NU )=SUBSTR(M,1,J);
                DO K=1 TO COUNT(NU );
X4:                  M='POINT'||K;
                     LOC=I;
                     CALL CONTMO(M,LOC);
                     CALL GETDAT(M,LOC,S);
                     IF ¬S THEN GO TO X4;
                     M=DBLANK(M);
                     M=M||'     ';
                     J=INDEX(M,' ');
                     MM=SUBSTR(M,1,J-1);
                       IF ¬NUMBER(MM) THEN GO TO  X4;
                     POINTS(NU ).X(K)=SUBSTR(M,1,J);
                     M=SUBSTR(M,J+1);
```

```
     M=DBLANK(M);
M=M||'     ';
     J=INDEX(M,' ');
      MM=SUBSTR(M,1,J-1);
         IF ¬NUMBER(MM) THEN GO TO X4;
     POINTS(NU ).Y(K)=SUBSTR(M,1,J);
     M=SUBSTR(M,J+1);
     IF M¬=' ' THEN POINTS(NU ).BLANK(K)='1'B;
         ELSE POINTS(NU ).BLANK(K)='0'B;
     END;
POINTS(NU).BLANK(1)='1'B;
END LINE;
```

PROGRAM PARSER

```
PARSER: PROC(N,M,MM,XWORD,XNUM,XSEQ,QUOTES,F,G,KEY,PRTB,BASVAL,BASSYM,
        /*PARSES AN EXPRESSION USING TABLES AND CALLS SEMANTIC        */
             LOCS,A,PRIM,DBG,PIC,XI,YI,PI,PNTI,STI);
       DCL N FIXED BINARY;                /*NUMBER OF PRODUCTIONS*/
        DCL M FIXED BINARY;                /* NUMBER OF SYMBOLS*/
        DCL MM FIXED BINARY;               /*NO. NON-BASIC SYMBOLS*/
        DCL (XWORD,XNUM,XSEQ) FIXED BINARY;
        DCL QUOTES CHAR(12);
            DCL F(0:M   ) FIXED BINARY;
            DCL G(0:M   ) FIXED BINARY;
            DCL KEY(0:M   ) FIXED BINARY;
            DCL PRTB(0:5*N) FIXED BINARY;
            DCL BASVAL(M   ) FIXED BINARY;
            DCL BASSYM (M   ) CHAR(12);
        DCL A CHAR(500) VAR;
        DCL DBG BIT(1);
        DCL (I,J,K,L,KK) FIXED BINARY;
        DCL (I1,I2) FIXED BINARY;
        DCL I3 FIXED BINARY;
        DCL XN FIXED BINARY;
            DCL VS(0:25 ) CHAR(10 ) VAR;     /*  VALUE STACK   */
        DCL S(0:25 ) FIXED BINARY;        /*  PARSING STACK    */
        DCL QUOTE BIT(1);     /*BOOLEAN FOR QUOTING BASIC SYMBOLS */
        DCL SYM FIXED BINARY;
        DCL SYMS CHAR(80)  VARYING;
        DCL ERROR BIT(1);
        DCL ANS  FIXED BINARY;
        DCL XXX BIT(1);
            DCL OS CHAR(80) VARYING;
        DCL TEMP CHAR(72) VAR;
        DCL (LOCS,LOCA) FIXED BINARY;
        DCL 1 PRIM,
            2 NUMM FIXED BINARY,
            2 BASIC(PI) BIT(1),
            2 NAM(PI) CHAR(10) VAR,
            2 VAL(PI) CHAR(72) VAR,
            2 POINTS(PI),
                3 COUNT FIXED BINARY,
                3 TAILX FIXED BINARY,
                3 TAILY FIXED BINARY,
                3 HEADX FIXED BINARY,
                3 HEADY FIXED BINARY,
                3 COOD   ,
                    4 BLANK(PNTI) BIT(1),
                    4 X(PNTI) FIXED BINARY,
                    4 Y(PNTI) FIXED BINARY;
        DCL 1 PIC,
            2 CNT FIXED BINARY,
            2 TAILX FIXED BINARY,
            2 TAILY FIXED BINARY,
            2 HEADX FIXED BINARY,
            2 HEADY FIXED BINARY,
            2 BL(PNTI) BIT(1),
            2 X(PNTI) FIXED BINARY,
            2 Y(PNTI) FIXED BINARY;
        DCL XI FIXED BINARY;  /* X SHIFT  */
        DCL YI FIXED BINARY;  /* Y SHIFT  */
        DCL PI FIXED BINARY;   /* NUMBER OF PRIMITIVES  */
```

```
          DCL PNTI FIXED BINARY;   /* NUMBER OF POINTS PER PRIMITIVE */
          DCL STI FIXED BINARY;   /* NUMBER OF STACK POINTS   */
          DCL 1 STK(5),
              2 NUM FIXED BINARY,
              2 TX FIXED BINARY,
              2 TY FIXED BINARY,
              2 HX FIXED BINARY,
              2 HY FIXED BINARY,
              2 BL(STI) BIT(1),
              2 U(STI) FIXED BINARY,
              2 V(STI) FIXED BINARY;
SINIT:    PROC;
          /*INITIALIZES SCOPE   */
              IF DBG THEN DO;
                  TEMP='PARSER';     CALL STRSML(TEMP,LOCS);
                  END;
                  LOCA=LOCS;
              END SINIT;
LOOK:     PROCEDURE(OS,A,I,T);
          /* FREE FIELD READ PROCEDURE   T IS FALSE IF NUMBER ELSE TRUE
              STRING RETURNED IN OS   */
          /*NUMBER::=INTEFER |+INTEGER | -INTEGER | -INTEGER. |+INTEGER.
              | INTEGER. | +INTEGER.INTEGER | -INTEGER.INTEGER |
                  INTEGER.INTEGER       */
          /*SEPARATOR IS ALWAYS BLANK  IF NOT A QUOTED STRING THEN A
              SEPARATOR IS ANY SINGLE CHARACTER IN THE SYNTAX     */
NEXT:     PROC CHAR(1);
          /*GETS NEXT CHAR SCOPE MODEL   */
          DCL K FIXED BINARY;
              IF I>LENGTH(A) THEN GO TO FINIS;
              RETURN(SUBSTR(A,I,1));
              END NEXT;
CON:      PROCEDURE;
              OS=OS||SYM;   I=I+1;
              END CON;
SPEC:     PROC(A,B) BIT(1);
           /*  TRUE  IF A IS NOT A SEPARATING CHARACTER   */
          DCL A CHAR(1);
          DCL B BIT(1);
          DCL J FIXED BINARY;
              IF A=' ' | A=QUOTES THEN RETURN('0'B);
              IF B THEN RETURN('1'B);
              DO J=1 TO M-1-MM;   IF A=BASSYM(J) THEN RETURN('0'B); END;
              RETURN('1'B);
              END SPEC;
              DCL SYM CHAR(1);
              DCL T BIT(1);
              DCL A CHAR(500) VAR;   /*READ BUFFER  */
              DCL I FIXED BINARY;  /* READ BUFFER POINTER   */
              DCL OS CHAR(80) VARYING;  /* OUTPUT STRING  */
          DCL CS CHAR(3) VAR;
          DCL J FIXED BINARY;
          DCL SPEC INTERNAL ENTRY (CHAR(1),BIT(1)) RETURNS(BIT(1));
              DCL NEXT INTERNAL ENTRY RETURNS (CHAR(1));
              DCL CON INTERNAL ENTRY;
              SYM=NEXT;  OS='';
              DO WHILE (SYM=' ');
                  I=I+1;  SYM=NEXT;   END;
              IF ¬SPEC(SYM,QUOTE) THEN DO;
                  CALL CON;   T='1'B;  RETURN;    END;
```

```
            CS='+-.';
            J=INDEX(CS,SYM);
            IF J=3 THEN CS='';  ELSE CS='.';
            IF (J¬=0 & NEXT>'Z')|(SYM>'Z' & (NEXT=CS |NEXT>'Z'))
                THEN DO;
                DO WHILE (NEXT>'Z' |NEXT=CS);
                CALL CON;
                    SYM=NEXT;
                    IF SYM=CS THEN CS='';
                    IF ¬SPEC(SYM,QUOTE) THEN DO;
                        T='0'B;  RETURN;   END;
                    END;
                END;
            DO WHILE (SPEC(SYM,QUOTE));
                CALL CON;
                SYM=NEXT;
                END;
            T='1'B;   RETURN;
            END LOOK;
ASSIGN: PROCEDURE (QUOTE,OS,V) RECURSIVE;
            /*ASSIGNS A NUMERICAL VALUE TO CURRENT INPUT STRING */
            DCL QUOTE BIT (1);
            DCL OS CHAR(80) VARYING;
            DCL (V,J) FIXED BINARY;
            DCL T BIT(1);
            IF QUOTE THEN DO;
                CALL LOOK(OS,A,I,T);
              IF OS=QUOTES THEN  DO;
                    QUOTE='0'B;  CALL ASSIGN(QUOTE,OS,V); RETURN;
                    END;
                    V=XWORD;   RETURN;
                END;
            CALL LOOK(OS,A,I,T);
            IF T THEN DO;
                IF OS=QUOTES THEN DO;
                    QUOTE='1'B; CALL ASSIGN(QUOTE,OS,V);  RETURN; END;
                DO J=1 TO M-1-MM;
                    IF OS= BASSYM(J) THEN DO;
                        V=BASVAL(J); RETURN;
                        END;
                    END;
                    V=XWORD;   RETURN;
                END;
            V=XNUM;  RETURN;
            END ASSIGN;
        XN=N;
            CALL SINIT;
            /*  PARSING SECTION    */
PROCES:
        ANS=0;  ERROR='0'B;
        DO J=0 TO 25;       S(J)=0;  END;
            J=0;   I=1;     QUOTE='0'B;
        CALL ASSIGN(QUOTE,SYMS,SYM);
        DO WHILE (SYM>0);
            J=J+1;
            IF J>25 THEN DO;
                TEMP='STACK OVERFLOW';
                CALL STRLAR(TEMP,LOCS);
                I1=1;  CALL DNSPAC(I1,LOCS);
                GO TO FINIS;
```

```
            END;
                K=J;  S(J)=SYM; VS(J)=SYMS;
            CALL ASSIGN(QUOTE,SYMS,SYM);
        DO WHILE (F(S(J))>G(SYM));
            IF S(J)=XSEQ THEN GO TO FINIS;
            DO WHILE ((F(S(J-1))=G(S(J)))&(J>1));
                J=J-1;  END;
            L=KEY(S(J));
            DO WHILE (PRTB(L)¬=0);
                KK=J+1;
                DO WHILE ((KK<=K)&(S(KK)=PRTB(L)));
                    KK=KK+1;  L=L+1;  END;
                IF ((KK>K)&(PRTB(L)<0)) THEN DO;
                    I1=J;  I2=K;
                        LOCS=LOCA;
                    I3=-PRTB(L);
                    IF I3<=N THEN
                        CALL SEMANT(I3,VS,I1,I2,ANS,ERROR,XN
                        ,PRIM,STK,LOCS,DBG,PIC,XI,YI,PI,PNTI,STI);
                    IF ERROR THEN GO TO FINIS;
                    IF I1¬=J|I2¬=K THEN PUT LIST('POINTER ERROR') SKIP;
                    S(J)=PRTB(L+1);  L=0;  END;
                ELSE DO;
                    DO WHILE (PRTB(L)>0);
                        L=L+1;  END;
                    L=L+2;    END;
                END;
                IF L¬=0 THEN DO;
                      L=0;   J=0;
                I1=1;  CALL DNSPAC(I1,LOCS);
            TEMP='PARSER ERROR';  CALL STRLAR(TEMP,LOCS);
            GO TO FINIS;
                END;
                K=J;
            END;
        END;
FINIS:
            END PARSER;
```

PROGRAM SEMANT

```
SEMANT: PROC(N,VS,J,K,   ANS,SWITCH,UU,PRIM,STK,LOCS,DBG,PIC,
        /*SEMANTIC ROUTINE FOR PDL                                    */
            XI,YI,PI,PNTI,STI);
            DCL UU FIXED BINARY;  /* UPPERBOUND FOR EQ  */
            DCL N FIXED BINARY;  /* FORMULA NUMBER */
            DCL VS(0:25 ) CHAR(10 ) VAR;    /*  VALUE STACK  */
            DCL J FIXED BINARY;  /* LEFT HAND STACK POINTER */
            DCL K FIXED BINARY;  /* RIGHT HAND STACK POINTER */
            DCL ANS FIXED BINARY; /*SET TO 0 USED ONLY IN SEMANTICS*/
            DCL SWITCH BIT(1);/* SET TO FALSE USED ONLY BY SEMANTICS*/
            DCL LOCS FIXED BINARY;
            DCL DBG BIT(1);
        DCL 1 PRIM,
            2 NUMM FIXED BINARY,
            2 BASIC(PI) BIT(1),
            2 NAM(PI) CHAR(10) VAR,
            2 VAL(PI) CHAR(72) VAR,
            2 POINTS(PI),
                3 COUNT FIXED BINARY,
                3 TAILX FIXED BINARY,
                3 TAILY FIXED BINARY,
                3 HEADX FIXED BINARY,
                3 HEADY FIXED BINARY,
                3 COOD    ,
                    4 BLANK(PNTI) BIT(1),
                    4 X(PNTI) FIXED BINARY,
                    4 Y(PNTI) FIXED BINARY;
        DCL 1 PIC,
            2 CNT FIXED BINARY,
            2 TAILX FIXED BINARY,
            2 TAILY FIXED BINARY,
            2 HEADX FIXED BINARY,
            2 HEADY FIXED BINARY,
            2 BL(PNTI) BIT(1),
            2 X(PNTI) FIXED BINARY,
            2 Y(PNTI) FIXED BINARY;
        DCL XI FIXED BINARY;  /* X SHIFT  */
        DCL YI FIXED BINARY;  /* Y SHIFT  */
        DCL PI FIXED BINARY;   /* NUMBER OF PRIMITIVES  */
        DCL PNTI FIXED BINARY;   /* NUMBER OF POINTS PER PRIMITIVE */
        DCL STI FIXED BINARY;   /* NUMBER OF STACK POINTS   */
        DCL 1 STK(5),
            2 NUM FIXED BINARY,
            2 TX FIXED BINARY,
            2 TY FIXED BINARY,
            2 HX FIXED BINARY,
            2 HY FIXED BINARY,
            2 BL(STI) BIT(1),
            2 U(STI) FIXED BINARY,
            2 V(STI) FIXED BINARY;
            DCL TEMP CHAR(72) VAR;
            DCL P FIXED BINARY;
            DCL Q FIXED BINARY;
MERGE:  PROC(I,J,K,M);
        /* MERGES STK(J) INTO STK(I) WITH X MOD K AND Y MOD M*/
        DCL (I,J,K,M,N,L) FIXED BINARY;
            IF NUM(I)+NUM(J)>STI THEN DO;  SWITCH='1'B; RETURN;  END;
            L=NUM(I);
```

```
          NUM(I)=NUM(I)+NUM(J);
          DO N=1 TO NUM(J);
               STK(I).BL(L+N)=STK(J).BL(N);
               STK(I).U(L+N)=STK(J).U(N)+K;
               STK(I).V(L+N)=STK(J).V(N)+M;
               END;
          STK(I).BL(L+1)='1'B;
          END MERGE;
PLUS:     PROC(I,J);
          /* DOES STK(I)+STK(J) LEAVES ANS IN I  */
          DCL (I,J,K,M) FIXED BINARY;
               K=HX(I)-TX(J);
               M=HY(I)-TY(J);
               HX(I)=HX(J)+K;
               HY(I)=HY(J)+M;
               CALL MERGE(I,J,K,M);
               END PLUS;
MINUS:    PROC(I,J);
          /* STK(I)-STK(J);      */
          DCL (I,J,K,M) FIXED BINARY;
               K=HX(I)-HX(J);
               M=HY(I)-HY(J);
               CALL MERGE(I,J,K,M);
               END MINUS;
STAR:     PROC(I,J);
          /*STK(I)*STK(J)'  */
          DCL (I,J,K,MM) FIXED BINARY;
          DCL M CHAR(72) VAR;
               IF TX(I)=TX(J) & TY(I)=TY(J) & HX(I)=HX(J) & HY(I)=HY(J)
                    THEN DO;
                    K=0; MM=0;  CALL MERGE(I,J,K,MM);
                    END;
               ELSE DO;
                    M='** ERROR';
                    CALL STRLAR(M,LOCS);   SWITCH='1'B;
                    END;
               END STAR;
CROSS:    PROC(I,J);
          /* STK(I) X STK(J)   */
          DCL (I,J,K,M) FIXED BINARY;
               K=TX(I)-TX(J);
               M=TY(I)-TY(J);
               HX(I)=HX(J)+K;
               HY(I)=HY(J)+M;
               CALL MERGE(I,J,K,M);
               END CROSS;
NEG:      PROC(I);
          /*   #  INTERCHANGES TAIL AND HEAD OF STK(I)   */
          DCL (I,J,K) FIXED BINARY;
               J=TX(I);
               K=TY(I);
               TX(I)=HX(I);
               TY(I)=HY(I);
               HX(I)=J;
               HY(I)=K;
               END NEG;
NCT:      PROC(I);
          /* BLANKS STK(I)   */
          DCL (I,J) FIXED BINARY;
              DO J=1 TO NUM(I);
```

```
                STK(I).BL(J)='1'B;
                END;
            END NOT;
PLOTS:  PROC(P);
        /* PLOTS A STRUCTURE    */
        DCL (P,J) FIXED BINARY;
        DCL 1 E,
            2 B(STI) BIT(1),
            2 X(STI) FIXED BINARY,
            2 Y(STI) FIXED BINARY;
            DO J=1 TO STK(P).NUM;
                B(J)=STK(P).BL(J);
                X(J)=STK(P).U(J);
                Y(J)=STK(P).V(J);
                END;
            J=STI;
            CALL PLOTL(E,STK(P).NUM,J,LOCS);
            CALL TAILHD(TX(P),TY(P),HX(P),HY(P),LOCS);
            END PLOTS;
            IF DBG THEN DO;
            TEMP=N;   CALL STRSML(TEMP,LOCS);
            END;
            IF SWITCH THEN GO TO EXIT;
            IF N=1 THEN DO;
                P=VS(J);    CALL PLOTS(P);
                IF STK(P).NUM<=PNTI THEN DO;
                    PIC.CNT=STK(P).NUM;
                    PIC.TAILX=STK(P).TX-XI;
                    PIC.TAILY=STK(P).TY-YI;
                    PIC.HEADX=STK(P).HX-XI;
                    PIC.HEADY=STK(P).HY-YI;
                    DO Q=1 TO STK(P).NUM;
                        PIC.BL(Q)=STK(P).BL(Q);
                        PIC.X(Q)=STK(P).U(Q)-XI;
                        PIC.Y(Q)=STK(P).V(Q)-YI;
                        END;
                    END;
                ELSE PIC.CNT=PNTI+10;
                END;
            IF N=3 THEN DO;
                P=VS(J);  Q=VS(K);      CALL PLUS(P,Q);
                ANS=P;
                END;
            IF N=4 THEN DO;
                P=VS(J);  Q=VS(K);   CALL MINUS(P,Q);   ANS=P;
                END;
            IF N=5 THEN DO;
                P=VS(J);  Q=VS(K);  CALL CROSS(P,Q);  ANS=P;
                END;
            IF N=6 THEN DO;
                P=VS(J);Q=VS(K);  CALL STAR(P,Q);  ANS=P;
                IF SWITCH THEN GO TO EXIT;
                END;
            IF N=9 THEN DO;
                P=VS(K);   CALL NOT(P);  VS(J)=P;
                END;
            IF N=10 THEN DO;
                P=VS(K);  CALL NEG(P);  VS(J)=P;
                END;
            IF N=12 THEN VS(J)=VS(J+1);
```

```
                IF N=13 THEN DO;
                    DO P=1 TO PRIM.NUMM;
                        IF VS(J)=NAM(P) & BASIC(P) THEN DO;
                            ANS=ANS+1;  VS(J)=ANS;
                            IF ANS>5   THEN DO;
                                SWITCH='1'B;   TEMP='STK OVERFLOW';
                                CALL STRLAR(TEMP,LOCS);  GO TO EXIT;
                                END;
                            STK(ANS).NUM=COUNT(P);
                            TX(ANS)=PRIM.TAILX(P)+XI;
                            TY(ANS)=PRIM.TAILY(P)+YI;
                            HX(ANS)=PRIM.HEADX(P)+XI;
                            HY(ANS)=PRIM.HEADY(P)+YI;
                            DO Q=1 TO COUNT(P);
                                STK(ANS).BL(Q)=POINTS(P).BLANK(Q);
                                STK(ANS).U(Q)=POINTS(P).X(Q)+XI;
                                STK(ANS).V(Q)=POINTS(P).Y(Q)+YI;
                                END;
                                GO TO EXIT;
                            END;
                        END;
                        SWITCH='1'B;   TEMP='NAME ERROR '||VS(J);
                        CALL STRLAR(TEMP,LOCS);
                    END;
                IF N=16|N=17 THEN VS(J)=VS(J)||' '||VS(K);
 EXIT:
                END SEMANT;
TERMINATE

IHE140I FILE SYSIN - END OF FILE ENCOUNTERED AT OFFSET +0006C FROM ENTRY POINT PR
```