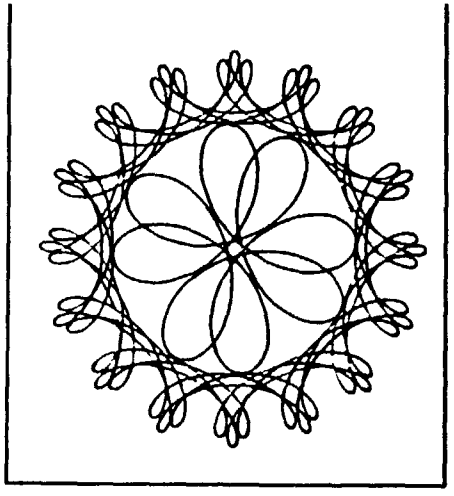
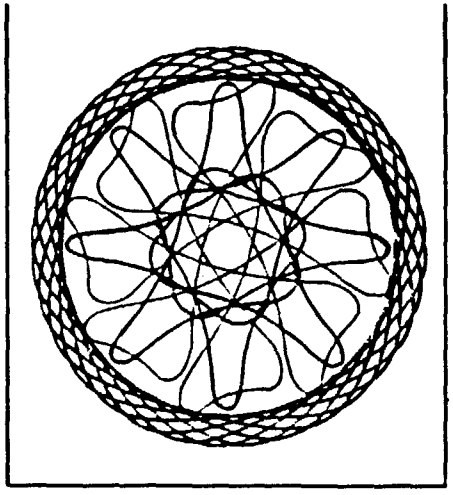
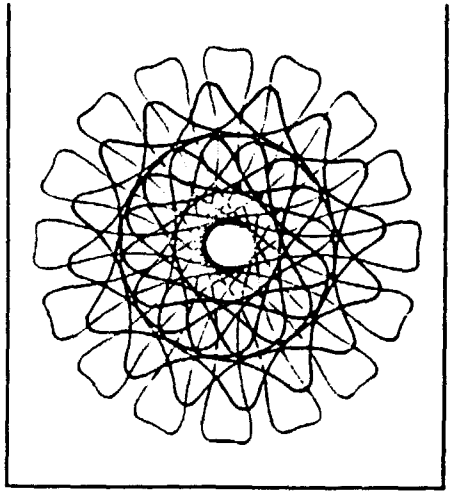
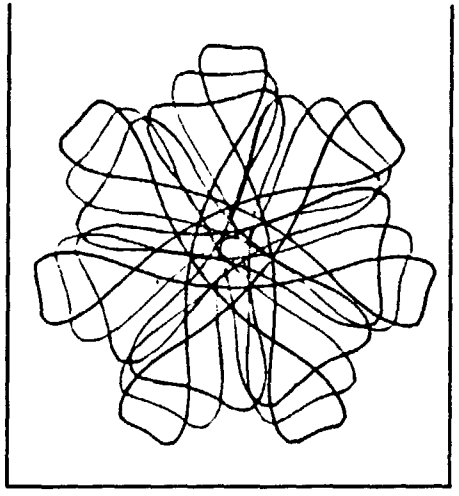




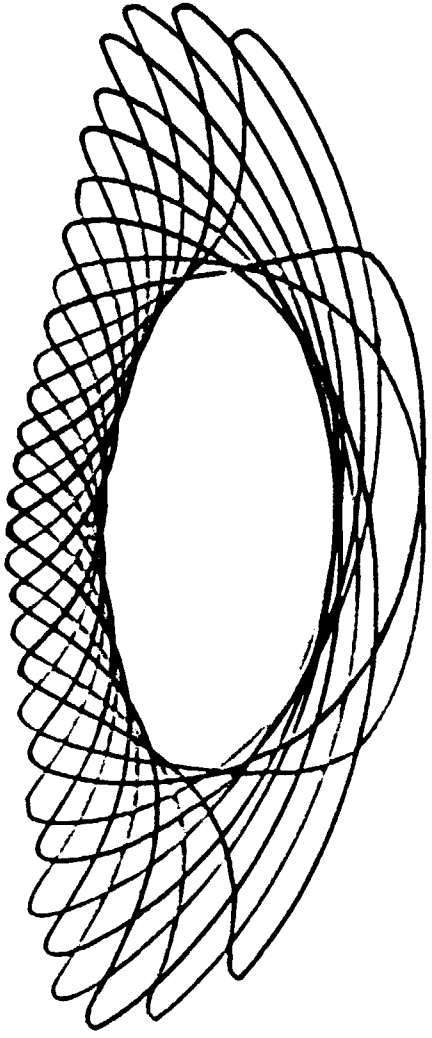


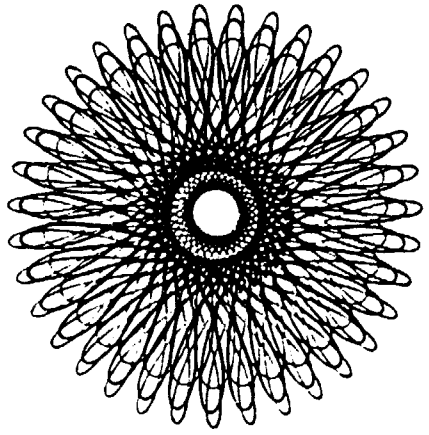


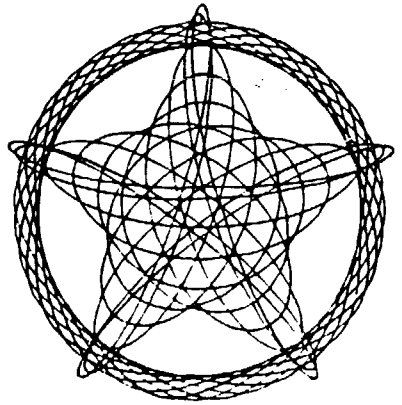
WE REALLY DO CARE!

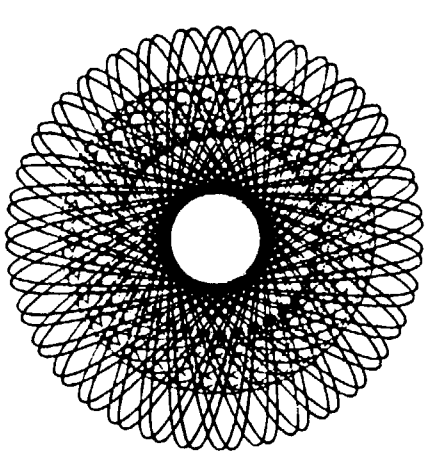
We make each toy with tender loving care. If there is a problem with your toy please let us know. Simply describe the problem and mail that description along with your name and address to:

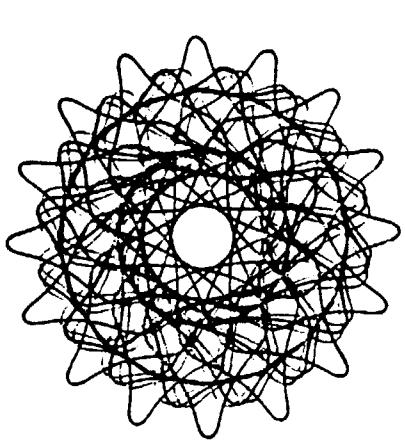
KENNER PRODUCTS
Consumer Relations Dept.
1014 Vine St.
Cincinnati, OH 45202





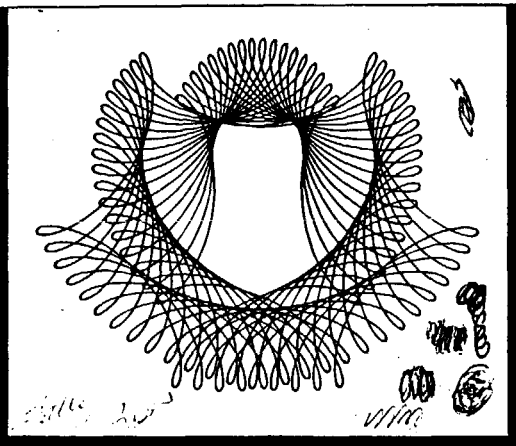




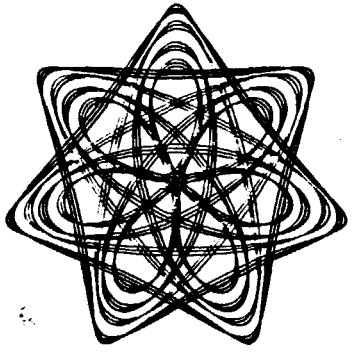


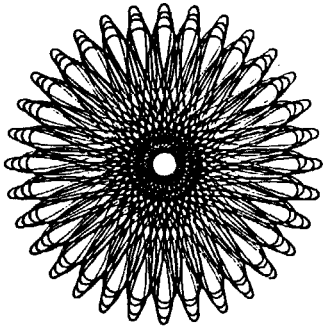
Kenner.

Appendix B: Original Instruction Booklet



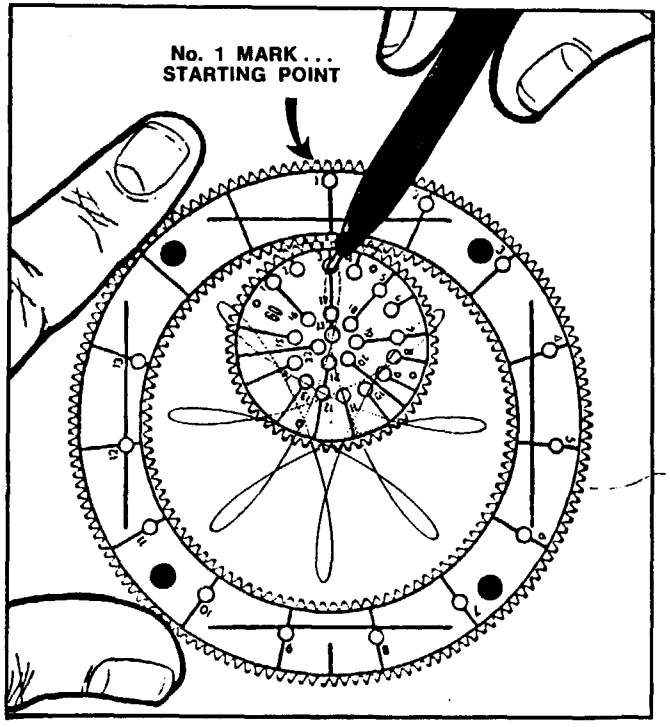
HOW TO DRAW WITH SPIROGRAPH





YOU CAN DRAW PATTERNS IMMEDIATELY!

THIS IS HOW . . . Put a sheet of Spirograph Paper, or any kind of paper, on the white side of the Baseboard and place a **RING** on it. Fasten it down with a pin through each of the four tiny pinholes.



Place a **WHEEL** inside the **Ring.** With a pen in one of the larger holes, (for your first pattern, use one of the holes closer to the center of the wheel), move the **Wheel** carefully around the inside of the Ring, *always keeping the teeth in contact.* Hold the pen upright, and do not press hard on it. Continue until your line meets where it started.

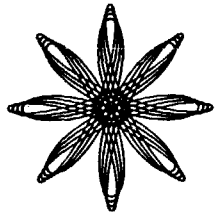
Try this again using a different hole in the **Wheel,** then by starting at a different position in the **Ring.**

The inside of the box lid shows the basic patterns made with the Wheels inside the Rings. By drawing these, you will get to know the pattern each Wheel makes.

Using your ball point pens for the first time, rub the points on a piece of scratch paper until the ink flows smoothly. Your Spirograph parts are precision engineered. Production requires a slightly shorter tooth at hole 1 in each part which does not affect the operation. If some excess plastic is left there, simply scrape it off. The shorter tooth will also assist you in locating the starting point.

Patterns drawn with Wheels inside of Rings

Also see pages 6 through 11



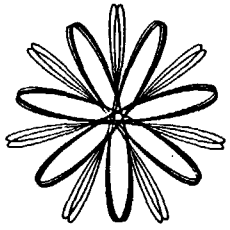
Pin **RING** No. $\frac{144}{96}$ to Paper and Baseboard, the No. 1 mark at the top.

Place **WHEEL** No. 60 so that Hole 1 lines up with the No 1 mark on the **Ring.** With pen in Hole 1 draw the pattern.

Reposition the **Wheel** so that Hole 3 lines up with the No. 1 mark on the **Ring** and over the pattern you have drawn. With pen in Hole 3 draw another pattern.

Repeat, using Holes 5 and 7.

The numbers on the Wheels, Rings and Racks indicate the number of gear teeth. On the Rings, the upper number is for the number of teeth on the outside, the bottom number for the inside.

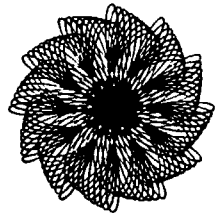


Pin **RING** No. $\frac{150}{105}$ to the Paper and Baseboard. Use **WHEEL** No. 60, Hole 1 at the No. 1 mark on the **Ring.** With Red Pen in Hole 1 draw the pattern.

Move the **Wheel** *one tooth to the right and* with Pen in Hole 1 draw the pattern.

Line up Hole 5 in **Wheel** with the No. 2 mark on **Ring.** With black pen in Hole 5 draw the pattern. Again move **Wheel** one tooth to the right and draw, pen in Hole 5.

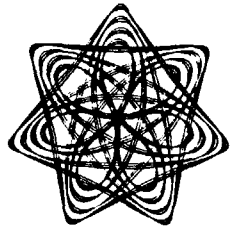
Turn **Wheel** over (numbers underneath) and position Hole 5 between the two patterns you have just drawn. Draw with green pen in Hole 5.



Pin **RING** No. $\frac{144}{96}$ to the Paper and Baseboard. Use **WHEEL** No. 56, Hole 1 at the No. 1 mark of the **Ring.** With pen in Hole 1 draw the pattern.

Reposition the **Wheel** so that *Hole 2* is one tooth to the right of the original pattern. Draw pattern with Pen in Hole 2.

Repeat, using Holes 3, 4, 5, 6, 7, 8 and 9, moving one tooth to the right with each hole.



Pin **RING** No. $\frac{150}{105}$ to the Paper Baseboard. Use **WHEEL** No. 30, Hole 1 at the No. 1 mark of the **Ring.** With the black pen in Hole 1 draw the pattern.

Reposition the **Wheel** so that Hole 2 is at top. Draw pattern, black pen in Hole 2. Repeat with Hole 3.

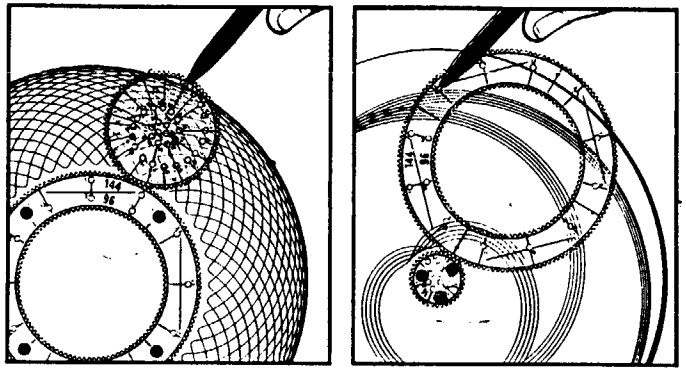
Now use **WHEEL** No. 45, Hole 6 at top. Draw pattern, green pen in Hole 6. Repeat with Holes 7 and 8.

Next, use **WHEEL** No. 60, Hole 11 at top. Draw pattern, red pen in Hole 11. Repeat with Holes 12 and 13.

To finish, use **WHEEL** No. 75, Hole 16 at top. Draw, blue pen in Hole 16. Repeat with Holes 17 and 18.

Giant-Size Patterns and Long Patterns

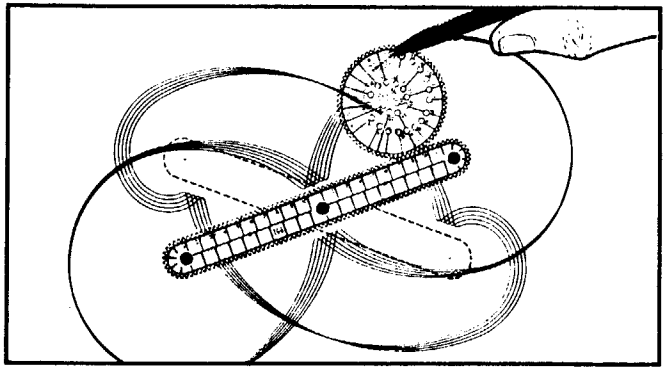
Also see both sides of back cover and page 12



TWO WAYS TO DRAW GIANT-SIZE PATTERNS

1. Pin down a **Ring** and revolve a **Wheel** around the outer edge.
2. Pin down a **Wheel** and revolve a **Ring** around it.

You can draw still more interesting patterns, but slightly smaller, by pinning down a Wheel and rolling another Wheel around it.



TO DRAW LONG PATTERNS

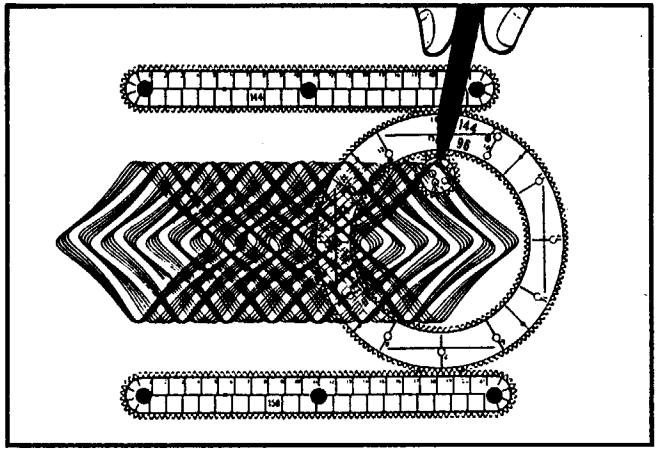
Pin down a **RACK** with pins through three pinholes. Roll a **Wheel** around it. Slow down when you go around the ends of the **Rack** so that the teeth stay in contact.

Beautiful designs can be drawn by changing the position of the **Rack**. Remove the two pins on the ends, leaving the center pin, and rotate the **Rack**. Pin it down again and repeat your pattern.

Another way is to move the **RACK** about a quarter inch up or down without rotating it.

Fascinating Continuous Patterns

Also see pages 13 and 14



IT'S REALLY QUITE SIMPLE!

To draw the design shown here, follow these steps:

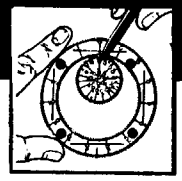
1. Pin down **Rack** 150 with three pins.
2. With **Ring** $\frac{144}{96}$ between the two Racks, pin down Rack 144 parallel to **Rack** 150.
3. Place the **Ring** at the *left end* of the two Racks, with *mark 1 at top.*
4. Use **Wheel** 24. With the pen in Hole 1 draw pattern. *Always line up Hole 1 with mark 1 in the Ring before drawing a pattern.*
5. Lift the top of the **Ring** and move *one tooth to the right along the upper Rack,* keeping the teeth engaged on the lower Rack; draw a pattern. Repeat until you have drawn six patterns.
6. Move the **Ring** two teeth to right along *both Racks;* draw a pattern.
7. Now move the **Ring** one tooth to right along the *lower Rack,* keeping teeth engaged in upper Rack; draw a pattern. Repeat until you have drawn six patterns.
8. Again move the **Ring** two teeth to right along *both Racks.* Keep repeating steps 4 through 8 until you reach the end of the Racks.

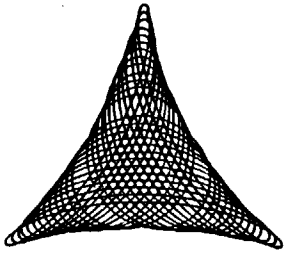
Try other continuous designs using different Wheels and different Holes. For other design ideas using two Racks see pages 13 and 14.

HOW TO USE THE EASY GUIDES THAT FOLLOW

In the guides for the patterns on the following pages, the RING to use is shown first. Then, the number of the WHEEL is shown in a Circle. The numbers thereafter indicate the HOLE in which to insert the pen. The color of this number tells you the color of the pen to use.

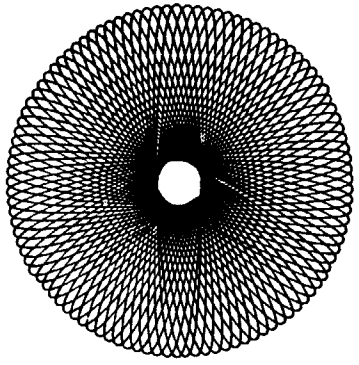
Have fun! Create your own patterns, too. Change the combination and position of the Wheels, Rings and Racks, using different holes and pens.



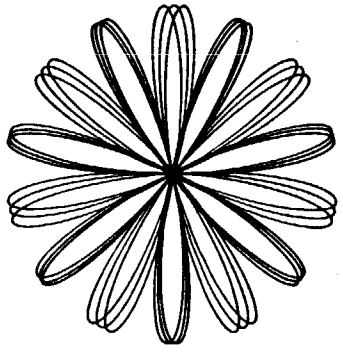


Ring No.	Wheel No.	Hole No. and Color of Pen
144/96	(64)	1-3-5-7-9-11-13-15-17 19-21-23-25

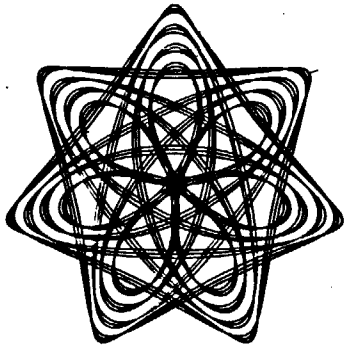
Always place Ring with mark 1 at top. Line up the Hole you are using with mark 1 unless otherwise instructed.



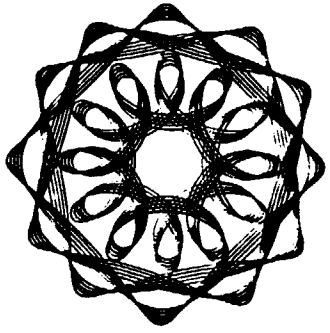
150/105 • (52) • 1 (18 loops)
1 (18 loops)
1 (to finish)



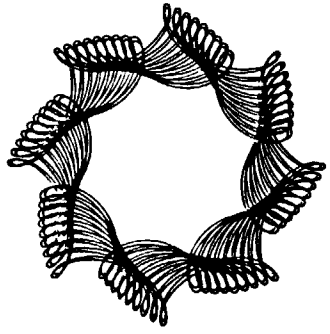
150/105 • (60) • 3-3 one tooth to right; 3 one tooth left. Now line up with mark 2 on Ring and draw 3-3 one tooth to right; turn Wheel over and draw 3, lined up with mark 2.



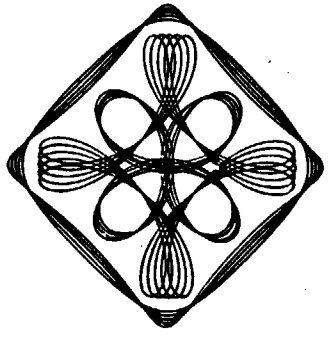
150/105 • (30) • 1-2-3 (45) • 6-7-8
(60) • 11-12-13 (75) • 16-17-18



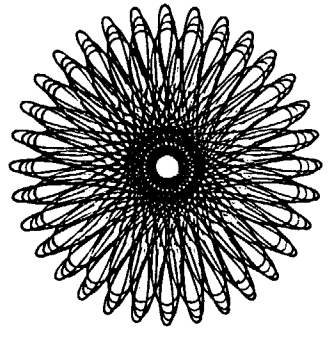
144/96 • (24) • 1-2-3-4-5
1-2-3-4-5, lined up at mark 2 on the Ring
1-2-3-4-5, lined up at mark 3
(80) • 15-16-17-18-19
15-16-17-18-19 lined up at mark 2



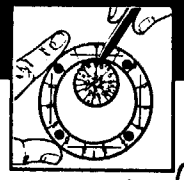
144/96 • (84) • 1-2-3-4-5-6-7-8-9, moving one tooth to the right every hole. Starting again at mark 1, draw 2-3-4-5-6-7-8-9-10, again moving one tooth right every hole.

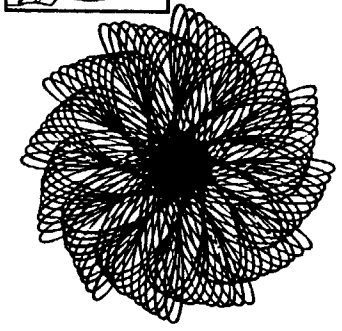


144/96 • (24) • 1-2-3-4-5
(72) • 9 at mark 1; 9 one tooth right; 9 two teeth right; 9 one tooth left of mark 1; 9 two teeth left. 18-19-20-21 lined up with the mark half-way between marks 2 and 3 on the Ring.

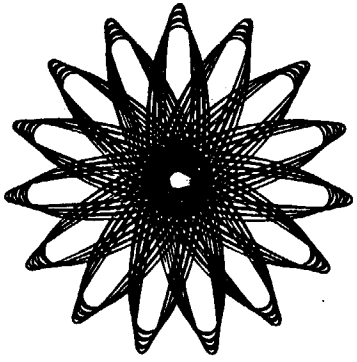


144/96 • (63) • 1-3-5-7

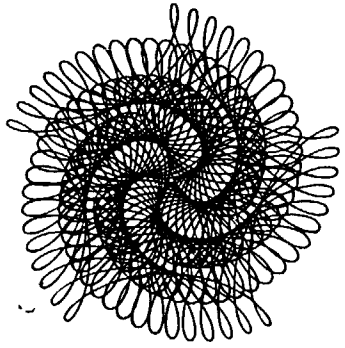




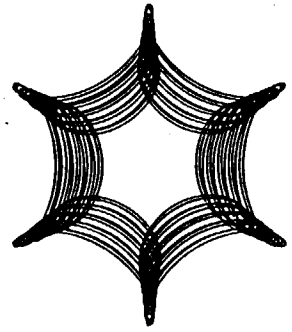
144/96 • (56) • 1-2-3-4-5-6-7-8-9-10 moving one tooth right every hole.



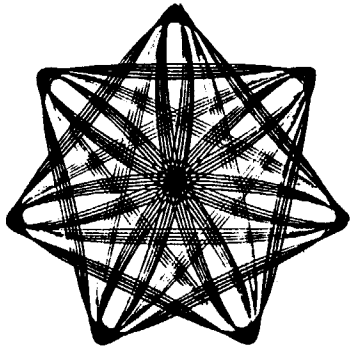
150/105 • (56) • 1-2-3-4-5-6-7



150/105 • (84) • 1 thru 26 moving two teeth right every hole.



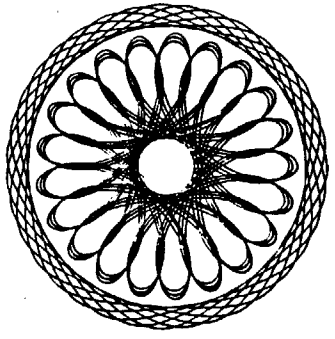
144/96 • (80) • 1-2-3-4-5-6-7-8-9-10-11-12-13-14



150/105 • (30) • 1-2-3-4-5

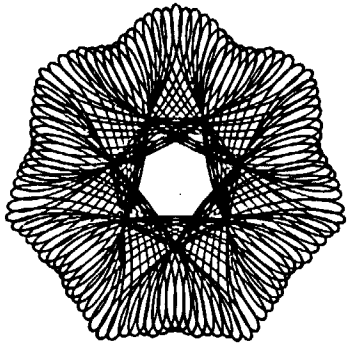
(45) • 1-2-3-4-5

(60) • 1-2-3-4-5



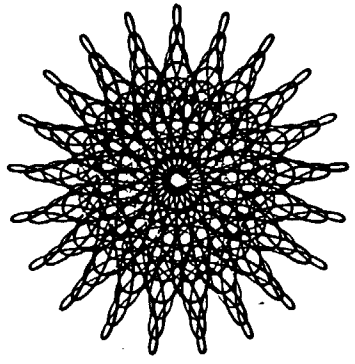
150/105 • (24) • 5

(80) • 13-14-15

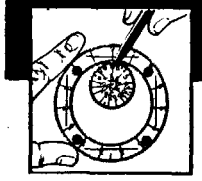


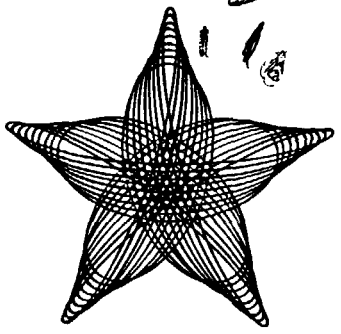
150/105 • (75) • 1-2-2-3-3-4-4-5-5-6-6-7-7-8-8

Draw with Hole 1 lined up at mark 1. Next, draw with Hole 2 one tooth to right, then one tooth to left of mark 1. Now Hole 3 one tooth farther to right, then one tooth farther left. Continue moving one tooth farther right and left with each hole.

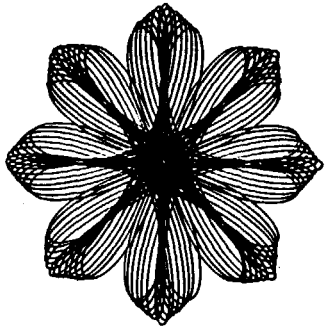


150/105 • (80) • 1-6-11-16-21

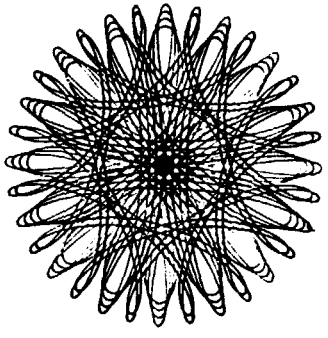




$\frac{150}{105}$ • (63) • 1-3-5-7-9-11-13-15-17

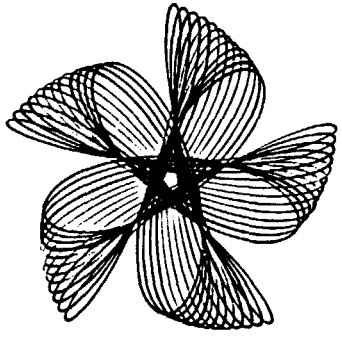


$\frac{144}{96}$ • (60) • 1-3-3-5-5-7-7 moving one tooth right and left every hole.
Turn Wheel over and draw 2-2-4-4-6-6 moving one tooth right and left of mark 1 every hole.

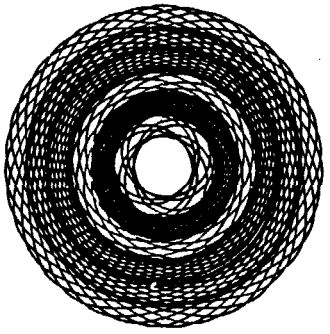


$\frac{144}{96}$ • (60) • 1-3-5-7, Move six teeth right and draw 1-3-5-7

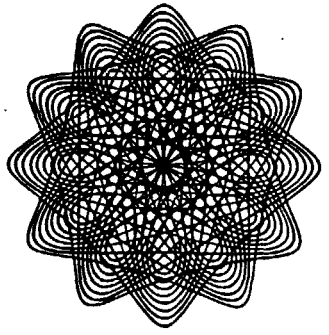
(72) • 1-3 Line up so that Hole is directly in the center between the blue and green patterns.



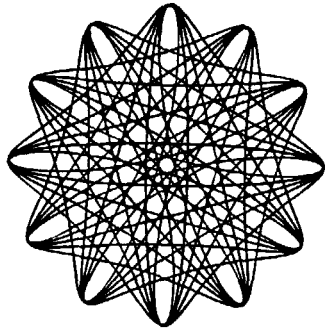
$\frac{150}{105}$ • (63) • 1-2-3-4-5-6-7-8-9-10-11-12 moving one tooth right every hole.



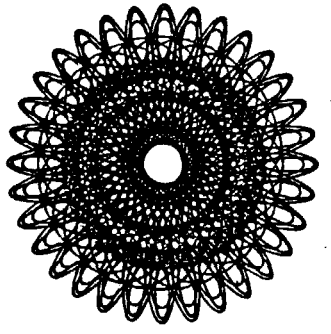
$\frac{150}{105}$ • (24) • 5 (36) • 11
(48) • 17 (60) • 23, starting at marks 1 and 2
(72) • 29 (84) • 35, starting at marks 1 and 3



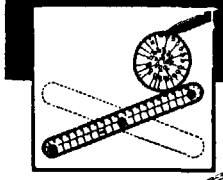
$\frac{144}{96}$ • (40) • 5 (56) • 9
(80) • 15, starting at marks 1 and 2
(72) • 13 and (24) • 1, both starting at marks 1, 2 and 3
(32) • 3 and (64) • 11, both starting at marks 1, 2, 3 and 4
(48) • 7, starting at marks 1 thru 6

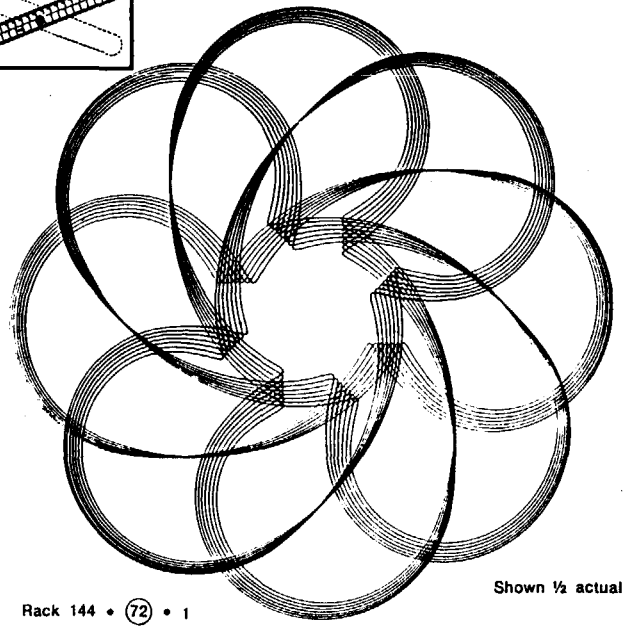


$\frac{144}{96}$ • (40) • 1 (56) • 1
(24) • 1, starting at marks 1, 2 and 3
(32) • 1, starting at marks 1, 2, 3 and 4
(48) • 1, starting at marks 1 thru 6



$\frac{144}{96}$ • (45) • 1-10-16 (63) • 3-5

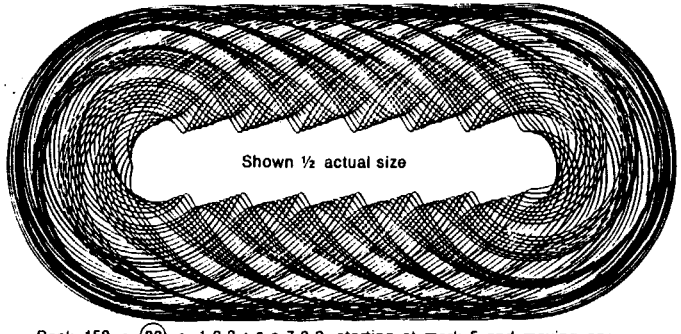




Shown ½ actual size

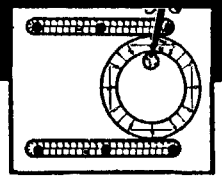
Rack 144 • (72) • 1

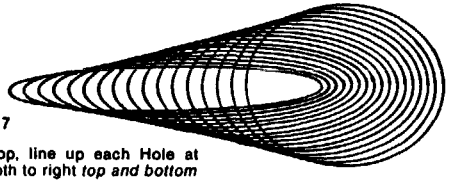
With Rack horizontal, start at mark 5 on Rack and draw seven times, moving one tooth right each time. Remove the two end pins and pivot Rack so that it is vertical and repeat with red pen. Again pivot so that Rack is between the two designs and draw with black pen, then pivot and draw with blue



Shown ½ actual size

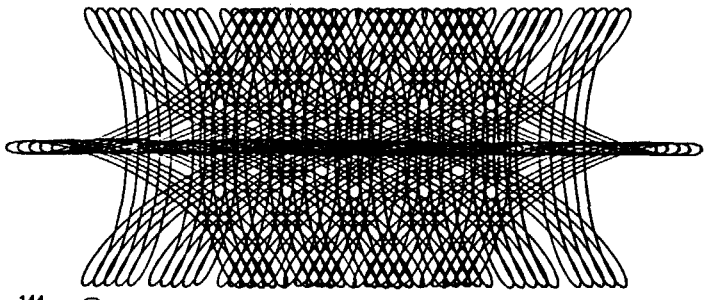
Rack 150 • (80) • 1-2-3-4-5-6-7-8-9, starting at mark 5 and moving one tooth right every hole.





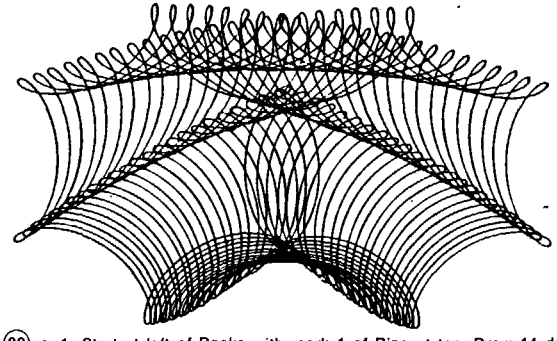
144/96 • (48) • 1 through 17

With mark 1 of Ring at top, line up each Hole at mark 4 and move Ring 1 tooth to right *top and bottom* for each pattern.

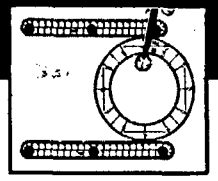


144/96 • (64) • 1 Begin at mark 10 on Ring and draw 5 blue patterns, moving Ring to the right one tooth top and bottom every time. Move two teeth, repeat in green; do this three times in each color.

After the sixth pattern, reverse, starting at mark 4 on the Ring and moving left.

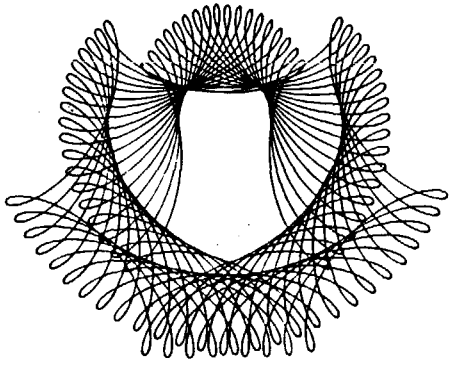


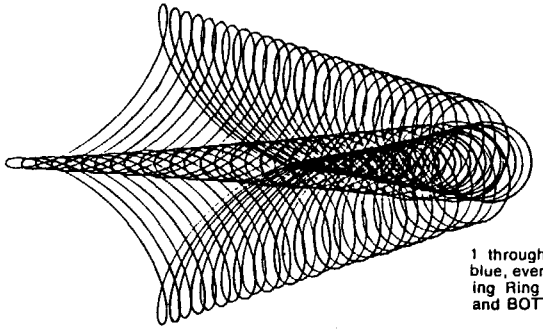
144/96 • (80) • 1 Start at left of Racks with mark 1 of Ring at top. Draw 14 designs, alternating red and blue, moving Ring two teeth right on TOP Rack every time. Always line up Hole 1 in the Wheel with mark 1 on the Ring.



$\frac{150}{105}$ • (84) • 1

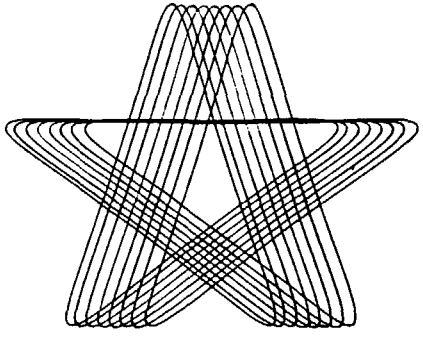
With mark 1 of Ring at top, begin with Hole 1 of Wheel at mark 3. Draw 18 patterns, each time moving Ring 1 tooth to right along *upper* Rack (keep teeth engaged in lower Rack) and each time move the Wheel 2 teeth to left inside the Ring.





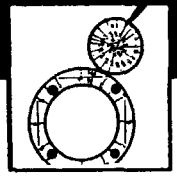
$\frac{144}{96}$ • (72)

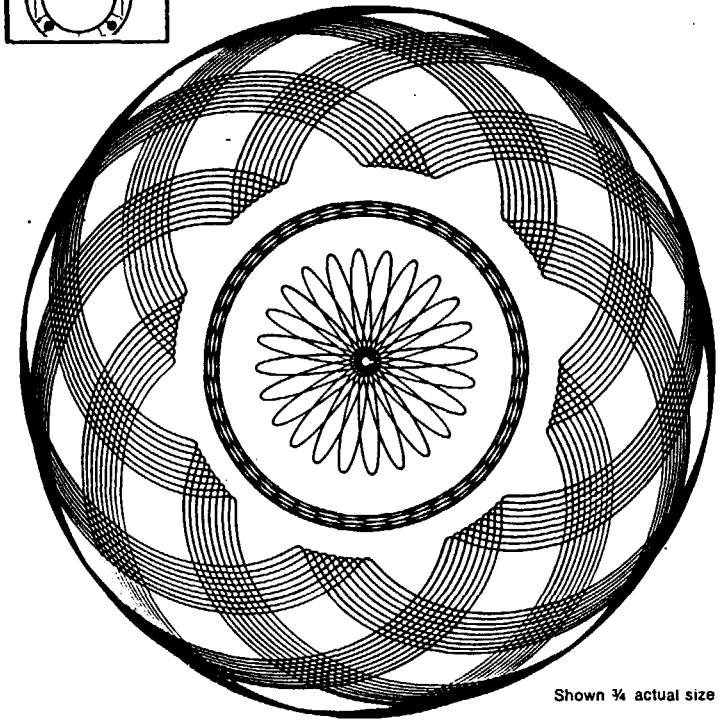
1 through 28, odd numbers in blue, even numbers in red, moving Ring one tooth right TOP and BOTTOM for each pattern.



$\frac{150}{105}$ • (42)

1-1-1-1-1-1-1-1, move Ring one tooth to right TOP and BOTTOM for each pattern.





Shown ¾ actual size

OUTSIDE EDGE OF RING

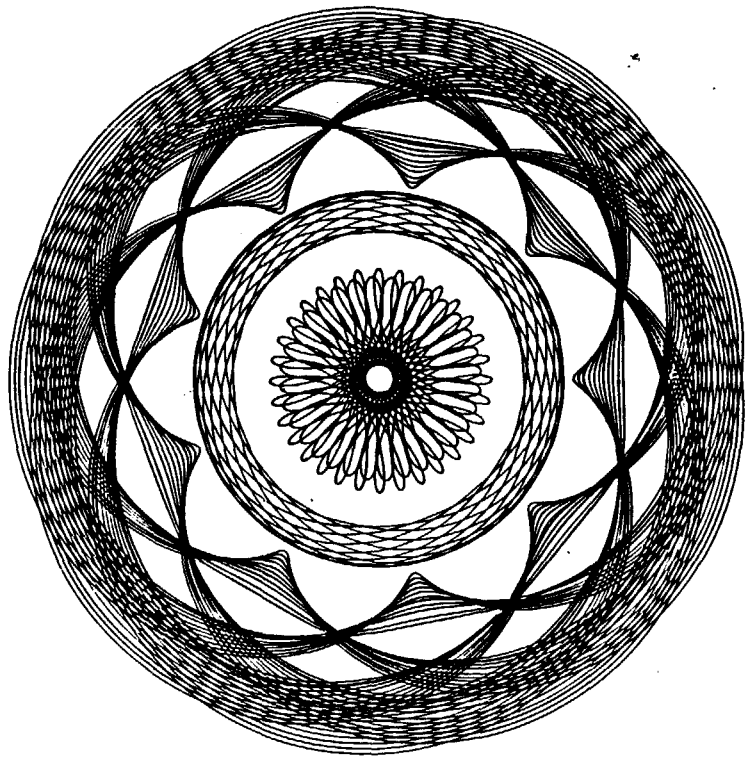
$\frac{144}{96}$ • (72) • 1; nine times, moving one tooth right every time; do the same in red, starting at the mark between marks 2 and 3; same in green, starting at mark 4; and in black starting at the mark between marks 5 and 6.

Leave Ring in place and draw INSIDE: (52) • 1

Now remove Ring and center Wheel (84) on the green pattern you have just drawn. Pin it to the board.

Use (40) • 13 around the Wheel.

Pattern on back cover is drawn in a similar manner . . . try it yourself.



Spirograph is fun. It is easy, yet challenging. A young child or an adult can draw beautiful designs at once. Teaches coordination of hand and eye, stimulates the imagination and develops creativity.

Based on mathematical principles and precision engineered, Spirograph is an award winning toy that provides endless fascination and enjoyment.

Use Spirograph to create designs on materials for embroidering, to decorate stationery, greeting cards, trading cards, lampshades, textiles and many more.

Make your own album of "SPIROGRAPHICS." Show it to your family and friends . . . compare theirs with yours.

Printed in U.S.A.

Now get Kenner's New SPIROGRAPH REFILL KIT No. 402, available at your favorite dealer. Kit contains simple guides for 70 more beautiful Spirograph patterns, special Drawing baseboard and Spirograph Paper.

If your dealer is temporarily out of kits, we will, as a service send it to you direct. Send 60c (check or money order, sorry no C.O.D.'s. Ohio residents add 4% Sales Tax). Include your name, address and the stock number and mail to: Dept. SRK,

Kenner Products Co., 912 Sycamore Street, Cincinnati, Ohio 45202

Source Listing

file _1DUA1:[USER.JOTAALEXA.THESIS]CYCLOGRAPH.LIS;1 (15743,41,0), last revised on 20-MAY-1986 21:32, is a 179 block sequential file owned by UIC [OOTAALEXANDE]. The records are variable length with implied (CR) carriage control. The longest record is 132 bytes.

Job CYCLOGRAPH (1205) queued to LPA0 on 20-MAY-1986 21:47 by user JOTAALEXANDE, UIC [OOTAALEXANDE], under account C at priority 4, started on printer _$ISCE$LPA0: on 20-MAY-1986 21:47 from queue LPA0.

```
-LINE-IDC-PL-SL-
-
 00001    C  0  0 {*****************************************************************************}
 00002    C  0  0 {*                                                                           *}
 00003    C  0  0 {* Program:                  Cyclograph                                     *}
 00004    C  0  0 {*                                                                           *}
 00005    C  0  0 {* Author:                   Todd A. Alexander                              *}
 00006    C  0  0 {*                                                                           *}
 00007    C  0  0 {* Purpose:                  Thesis / Creative Project required to graduate *}
 00008    C  0  0 {*                           from the Honors College -- this is part of my  *}
 00009    C  0  0 {*                           submission under the enrollment of ID 499.     *}
 00010    C  0  0 {*                                                                           *}
 00011    C  0  0 {* Date of Last Revision:    May 17, 1986                                   *}
 00012    C  0  0 {*                                                                           *}
 00013    C  0  0 {* Program Overview:         This program, along with its subordinates,     *}
 00014    C  0  0 {*                           attempts to emulate the Spirograph(TM) drawing *}
 00015    C  0  0 {*                           toy developed by Kenner, Inc.  Interactive as  *}
 00016    C  0  0 {*                           well as programming commands are used to draw  *}
 00017    C  0  0 {*                           designs at the terminal.                       *}
 00018    C  0  0 {*                                                                           *}
 00019    C  0  0 {* Modules Called:           Home                                           *}
 00020    C  0  0 {*                           Home2                                          *}
 00021    C  0  0 {*                                                                           *}
 00022    C  0  0 {* Modules Used In RAL Library:                                             *}
 00023    C  0  0 {*                                                                           *}
 00024    C  0  0 {*                           Clear_Screen                                   *}
 00025    C  0  0 {*                           Init_graphics                                  *}
 00026    C  0  0 {*                           Line                                           *}
 00027    C  0  0 {*                           Move                                           *}
 00028    C  0  0 {*                           Set_window                                     *}
 00029    C  0  0 {*                           Set_color                                      *}
 00030    C  0  0 {*                                                                           *}
 00031    C  0  0 {*****************************************************************************}
 00032       0  0
 00033       0  0
-
```

```
-LINE-IDC-PL-SL-

00035      0  0 Program Cyclograph(Input,Output,Procfile);
00036      0  0
00037      0  0  Type
00038      0  0    SetOfCharacters = Set of Char;
00039      0  0    String          = Packed Array[1..80] of Char;
00040      0  0    Lines           = Packed Array[1..80] of Char;
00041      0  0    Words           = Packed Array[1..10] of Char;
00042      0  0    Word_List       = Packed Array[1..19] Of Words;
00043      0  0    Lists           = Packed Array[1..200] of Lines;
00044      0  0    DDLists         = Packed Array[1..200,1..4] of Words;
00045      0  0
00046      0  0  Var
00047      0  0
00048   C  0  0 {*******************************************************************************}
00049   C  0  0 {* Variables used for the immediate mode and graphing                           *}
00050   C  0  0 {*******************************************************************************}
00051      0  0
00052   C  0  0 {  VARIABLE NAME    TYPE          PURPOSE/EXPLANATION                          }
00053      0  0
00054      0  0    Immediate_mode : Boolean;     {False=program mode}
00055      0  0
00056      0  0    Command        : Char;        {Single character command}
00057      0  0
00058      0  0    L              : Integer;     {Multi-purpose loop index variable}
00059      0  0    Lcount         : Integer;     {Number of letters in words in Cline}
00060      0  0    M              : Integer;     {Multi-purpose loop index variable}
00061      0  0    Numhole        : Integer;     {Number of hole on wheel}
00062      0  0    Theta2         : Integer;     {Loop index for real Theta}
00063      0  0
00064      0  0    OldColor       : Packed Array[1..80] of Char;  {Old color, used in NEXT command}
00065      0  0    Cline          : Packed Array[1..80] of Char;  {Immediate command line}
00066      0  0    Word           : Packed Array[1..80] of Char;  {Contiguous string of characters after single character command }
00067      0  0
00068      0  0    Cycles         : Real;        {Number calculated to specify number of revolutions of wheel in ring}
00069      0  0    Cosr           : Real;        {Intermediate cosine value, used to rotate}
00070      0  0    Cossum         : Real;        {Intermediate cosine value, used to rotate}
00071      0  0    Cosl           : Real;        {Intermediate cosine value, used to rotate}
00072      0  0    Length         : Real;        {Length of segment from current point to origin}
00073      0  0    Rotation       : Real;        {Rotation in radians of ring, in steps of "teeth"}
00074      0  0    Sinsum         : Real;        {Intermediate sine value, used to rotate}
00075      0  0    Stc            : Real;        {For intermittent graphing - where wheel left off}
00076      0  0    Sinr           : Real;        {Intermediate sine value, used to rotate}
00077      0  0    Sinl           : Real;        {Intermediate sine value, used to rotate}
00078      0  0    Strt           : Real;        {For intermittent graphing - where wheel begins}
00079      0  0    Theta          : Real;        {Angle from x axis to center of wheel}
00080      0  0
00081      0  0    Commands       : SetOfCharacters;       {Set of valid one letter commands}
00082      0  0
00083      0  0    Beta           : Single;      {Angle of revolution of wheel relative to theta}
00084      0  0    Num            : Single;      {Value of number found within Cline}
00085      0  0    Radius         : Single;      {Adjusted radius of wheel}
00086      0  0    Ratio          : Single;      {Ring divided by wheel radius}
00087      0  0    a              : Single;      {Radius of ring}
00088      0  0    b              : Single;      {Radius of wheel}
00089      0  0    e              : Single;      {Length to hole in wheel}
```

```
-LINE-IDC-PL-SL-

 00090      0  0    h             : Single;     {Hole number}
 00091      0  0    j             : Single;     {Distance from center of hole to pen contact}
 00092      0  0    x,y           : Single;     {x and y vectors, || to axis, plotted from origin}
 00093      0  0
```

```
-LINE-IDC-PL-SL-

00095   C  0  0 {*****************************************************************************}
00096   C  0  0 {* Variables used for the intepreter                                          *}
00097   C  0  0 {*****************************************************************************}
00098      0  0
00099   C  0  0 {  VARIABLE NAME    TYPE          PURPOSE/EXPLANATION                        }
00100      0  0
00101      0  0    Counter_stack  : Array[1..20] of Integer;   {Stacks number of repetions}
00102      0  0    Pc_stack       : Array[1..20] of Integer;   {Stacks line numbers for DO and RUN commands}
00103      0  0    Procline       : Array[1..50] of Integer;   {List of line numbers for corresponding procedures in Proclist}
00104      0  0
00105      0  0    Error_free     : Boolean;     {Flag for errors in program}
00106      0  0    Found          : Boolean;     {Flag for searches through lists}
00107      0  0
00108      0  0    WordList       : DDLists;     {2 dimensional array holding command procedures seperated into words}
00109      0  0
00110      0  0    A2             : Integer;     {Multi-purpose loop index variable}
00111      0  0    B2             : Integer;     {Multi-purpose loop index variable}
00112      0  0    Dos            : Integer;     {Number of DO statements found in procedure}
00113      0  0    Errornum       : Integer;     {Number of errors found in Procfile}
00114      0  0    K              : Integer;     {Multi-purpose loop index variable}
00115      0  0    Linecount      : Integer;     {Number of command lines in Procfile}
00116      0  0    Letcount       : Integer;     {Number of letters in command words}
00117      0  0    PC             : Integer;     {Program line counter - current line number being executed}
00118      0  0    Procs          : Integer;     {Total number of defined procedure in procfile}
00119      0  0    Pointer        : Integer;     {Level of nesting of procedures or DO loops}
00120      0  0    Runs           : Integer;     {Number of RUN statements encountered in Procfile}
00121      0  0    Starts         : Integer;     {Number of START statements found in a procedure}
00122      0  0    Wordcount      : Integer;     {Current number of words found in command}
00123      0  0
00124      0  0    CommLine       : Lines;       {Line of text from Procfile}
00125      0  0
00126      0  0    CommList       : Lists;       {Entire list of lines from Procfile}
00127      0  0
00128      0  0    Digits         : Set of char; {Valid set of characters found in integers}
00129      0  0    RealDigits     : Set of char; {Valid set of characters found in real numbers}
00130      0  0
00131      0  0    Procfile       : TEXT;        {Text file which holds program procedures}
00132      0  0
00133      0  0    Proclist       : Word_List;   {List of procedure names}
00134      0  0    Runlist        : Word_List;   {List of procedures called}
00135      0  0    Val1List       : Word_List;   {Valid words for column 1}
00136      0  0    Val2List       : Word_List;   {Valid words for column 2}
00137      0  0    Val3List       : Word_List;   {Valid words for column 3}
00138      0  0
```

```
-LINE-IDC-PL-SL-

00140   C  0  0 {*****************************************************************************}
00141   C  0  0 {* External Procedures for RAL ReGIS Graphics and Cobol External Procedures  *}
00142   C  0  0 {*****************************************************************************}
00143      0  0
00144      1  0 Procedure Home;
00145   C  1  0  {Clears screen and places prompt}
00146      0  0  Extern;
00147      0  0
00148      1  0 Procedure Home2;
00149   C  1  0  {Clears screen and places second prompt}
00150      0  0  Extern;
00151      0  0
00152      1  0 Procedure Clear_screen;
00153   C  1  0  {Clears screen of graphics and text, to black}
00154      0  0  Fortran;
00155      0  0
00156      1  0 Procedure Init_graphics;
00157   C  1  0  {Sets default colors, positions for graphics}
00158      0  0  Fortran;
00159      0  0
00160      1  0 Procedure Line(X:Single;Y:Single);
00161   C  1  0  {Draws line from current graphic position to x,y}
00162      0  0  Fortran;
00163      0  0
00164      1  0 Procedure Move(X:Single;Y:Single);
00165   C  1  0  {Moves graphic cursor to x,y}
00166      0  0  Fortran;
00167      0  0
00168      1  0 Procedure Set_window(Xleft,Ybot,Xright,Ytop:Single);
00169   C  1  0  {Sets x and y bounds}
00170      0  0  Fortran;
00171      0  0
00172      1  0 Procedure Txt(%Stdescr Message: Packed Array[a..b:integer] Of Char);
00173   C  1  0  {Places text on screen}
00174      0  0  Fortran;
00175      0  0
00176      1  0 Procedure Set_color(%STDESCR String:Packed Array[a..b:integer] Of Char;
00177      1  0                         Number: Integer);
00178   C  1  0  {Changes color of plotting}
00179      0  0  Fortran;
00180      0  0
```

```
-LINE-IDC-PL-SL-

00182   C  0  0 {*****************************************************************************}
00183   C  0  0 {* Internally Defined Functions                                               *}
00184   C  0  0 {*****************************************************************************}
00185      0  0
00186      1  0 Function Power(x:real; n:integer) :real;
00187   C  1  0    {Returns x raised to the n}
00188      1  1    Begin
00189      1  1     If n=0 then Power:=1 else Power:=x * Power(x,n-1)
00190      0  0    End;
00191      0  0
00192      1  0 Function Value(X:char) :integer;
00193   C  1  0    {Returns value of a number character}
00194      1  1    Begin
00195      1  1      Value := Ord(X) - Ord('0');
00196      0  0    End;
00197      0  0
00198      1  0 Function Value_Real(x:words) :real;
00199   C  1  0  {Returns value of real number which was in character format}
00200      1  0  Var
00201      1  0   M,Num: Integer;
00202      1  0   Num2 : real;
00203      1  0   Point_found, found_again: boolean;
00204      1  1  Begin
00205      1  1      Num := 0;
00206      1  1      Num2 := 0;
00207      1  1      m := 1;
00208      1  1      Point_found := false;
00209      1  1      while (m < 11) and not(point_found) Do
00210      1  2       Begin
00211      1  2        If x[m] = '.' then Point_found := true
00212      1  2         else if x[m] <> ' ' then  Num := NUM*10 + VALUE(X[M]);
00213      1  2        m := m+1;
00214      1  1       End;
00215      1  1      M := 10;
00216      1  1      While Point_found and Not(Found_again) Do
00217      1  2       Begin
00218      1  2        If x[m] = '.' then found_again := true;
00219      1  2        If (x[m] <> ' ') and (x[m] <> '.')
00220      1  2              then  Num2 := NUM2/10 + VALUE(X[M])/10;
00221      1  2        m := m-1;
00222      1  1       End;
00223      1  1      Value_Real := num + num2;
00224      0  0  End;
00225      0  0
00226      1  0 Function Value_Integer(x:words) :integer;
00227   C  1  0  {Returns value of an integer which was in character format}
00228      1  0  Var
00229      1  0   M,Num : Integer;
00230      1  1  Begin
00231      1  1      Num := 0;
00232      1  1      m := 1;
00233      1  1      while (m < 11) Do
00234      1  2       Begin
00235      1  2        If x[m] <> ' ' then  Num := Num*10 + Value(x[m]);
00236      1  2        m := m+1;
```

```
-LINE-IDC-PL-SL-

00237      1  1      End;
00238      1  1      Value_Integer := num;
00239      0  0  End;
00240      0  0
00241      1  0 Function Column2_Not_Null(X:DDLists; k:integer) :boolean;
00242   C  1  0    {Returns true if something appears in column 2}
00243      1  1    Begin
00244      1  1      Column2_Not_Null := false;
00245      1  1      If (x[k,2]<>'|          ') then Column2_not_null := true;
00246      0  0    End;
00247      0  0
00248      1  0 Function Column3_Not_Null(X:DDLists; k:integer) :boolean;
00249   C  1  0    {Returns true if something appears in column 3}
00250      1  1    Begin
00251      1  1      Column3_Not_Null := false;
00252      1  1      If (x[k,3]<>'|          ') then Column3_not_null := true;
00253      0  0    End;
00254      0  0
00255      1  0 Function Column4_Not_Null(X:DDLists; k:integer) :boolean;
00256   C  1  0    {Returns true if something appears in column 4}
00257      1  1    Begin
00258      1  1      Column4_Not_Null := false;
00259      1  1      If (x[k,4]<>'|          ') then Column4_not_null := true;
00260      0  0    End;
00261      0  0
00262      1  0 Function Valid_Integer(x:words) :boolean;
00263   C  1  0    {Returns true if string is in valid integer format}
00264      1  0    Var
00265      1  0      a : integer;
00266      1  0      valid_bool:boolean;
00267      1  1    Begin
00268      1  1      Valid_Integer := true;
00269      1  1      valid_bool := true;
00270      1  1      a := 1;
00271      1  1      while( (valid_bool) and (a<11) and (x[a]<>' ') ) do
00272      1  1          If not (x[a] in Digits) then
00273      1  2           Begin Valid_Integer := false;
00274      1  2                 Valid_bool := false;
00275      1  2           End
00276      1  1          Else
00277      1  1           a := a + 1;
00278      0  0    End;
00279      0  0
00280      1  0 Function Valid_real(x:words) :boolean;
00281   C  1  0    {Returns true if string is in valid real format}
00282      1  0    Var
00283      1  0      a,pointcount : integer;
00284      1  0      valid_bool:boolean;
00285      1  1    Begin
00286      1  1      Valid_real := true;
00287      1  1      valid_bool := true;
00288      1  1      a := 1;
00289      1  1      pointcount := 0;
00290      1  1      while( (valid_bool) and (a<11) and (x[a]<>' ') ) do
00291      1  1       Begin
```

```
-LINE-IDC-PL-SL-

00292      1  2          If x[a] = '.' then pointcount := pointcount + 1;
00293      1  2          If (not (x[a] in RealDigits)) or (Pointcount > 1) then
00294      1  3           Begin Valid_real := false;
00295      1  3                 Valid_bool := false;
00296      1  3           End
00297      1  2          Else a := a + 1;
00298      1  1       End;
00299      0  0    End;
00300      0  0
00301   C  0  0 {*****************************************************************************}
00302   C  0  0 {* Internally Defined Procedures                                             *}
00303   C  0  0 {*****************************************************************************}
00304      1  0 Procedure Find_closure_cycles;
00305   C  1  0    {Calculates number of cycles necessary to complete a design}
00306      1  1    Begin
00307      1  1      Ratio := a/o;
00308      1  1      Cycles := 0;
00309      1  1      m := 1;
00310      1  1      While (Cycles = 0) Do
00311      1  2        Begin
00312      1  2          If abs(trunc(m * Ratio) - (m * Ratio)) < 0.0001 then Cycles := m;
00313      1  2          m := m + 1;
00314      1  1        End;
00315      0  0 End;
00316      0  0
00317      0  0
00318      0  0
00319      1  0 Procedure Find_word;
00320   C  1  0    {Locates the first contiguous string of characters following command}
00321      1  1    Begin
00322      1  1       word := '                                                          ';
00323      1  1       Lcount := 1;
00324      1  1       L := 0;
00325      1  1       While (Cline[L] <> ' ') Do
00326      1  2          Begin
00327      1  2          word[Lcount] := Cline[L];
00328      1  2          Lcount := Lcount + 1;
00329      1  2          L := L + 1;
00330      1  1          end;
00331      0  0    End;
00332      0  0
00333      0  0
00334      1  0 Procedure Find_number;
00335   C  1  0    {Assigns NUM the value of the number found within a command}
00336      1  1    Begin
00337      1  1      Find_word;
00338      1  1      Num := 0;
00339      1  2    If word <> ' ' then Begin
00340      1  2      m := 11;
00341      1  2      Lcount := 0;
00342      1  2      while (m > 0) Do
00343      1  3      Begin
00344      1  3        If word[m] <> ' ' then
00345      1  4          Begin
00346      1  4            Num := Value(Word[m]) * Power(10,Lcount) + Num;
```

```
-LINE-IDC-PL-SL-
 00347      1  4            lcount := lcount + 1;
 00348      1  3          End;
 00349      1  3        m := m-1;
 00350      1  2      End;
 00351      1  1    End;
 00352      0  0 end;
 00353      0  0
 00354      0  0
 00355      1  0 Procedure Find_loop_cycles;
 00356    C 1  0    {Calculates number of cycles according to loop number given by user}
 00357      1  1    Begin
 00358      1  1      Cycles := 2 * Num * b / a;
 00359      0  0 End;
 00360      0  0
 00361      1  0 Procedure Go_graph;
 00362    C 1  0   {Graphs wheel within a ring, according to cycles, a, and b}
 00363      1  1   Begin
 00364      1  1      Radius :=  0.95 * b;
 00365      1  1      Numhole := Trunc(b/2 - 7 + 0.5);
 00366      1  1      If numhole < 5 then numhole := 5;
 00367      1  1      e := (Radius/Numhole) * h;
 00368      1  1      j :=  3;
 00369      1  1      If Immediate_mode then Find_number;
 00370      1  2      If Num = 0 then Begin
 00371      1  2            Find_closure_cycles;
 00372      1  2            Cycles := Cycles * 2;
 00373      1  2            Strt := 0;
 00374      1  2            Stp := Cycles;
 00375      1  2            End
 00376      1  2        else Begin
 00377      1  2            Find_loop_cycles;
 00378      1  2            Cycles := Cycles * 2;
 00379      1  2            Stp := Cycles + Strt;
 00380      1  1            End;
 00381      1  1      For Theta2 := Trunc(36 * Strt) to Trunc(36 * Stp)
 00382      1  2      Do Begin
 00383      1  2        Theta := Theta2 * 5;
 00384      1  2        Theta := Theta * (3.1415/180);
 00385      1  2        Beta := Theta*(a-b)/b - 3.1415/2;
 00386      1  2        x := (a-b+j)*Cos(Theta) + b*Cos(Theta*(a-b)/b) + e*Sin(Beta);
 00387      1  2        y := (a-b+j)*Sin(Theta) - b*Sin(Theta*(a-b)/b) + e*Cos(Beta);
 00388      1  2        Length := Sqr(x*x + y*y);
 00389      1  2        cos1 := (x/length);
 00390      1  2        sin1 := (y/length);
 00391      1  2        Sinr := sin(rotation);
 00392      1  2        Cosr := cos(rotation);
 00393      1  2        cossum := Cosr*cos1 - Sinr*sin1;
 00394      1  2        sinsum := Sinr*cos1 + Cosr*sin1;
 00395      1  2        x := - length * sinsum * 2.9 + 300;
 00396      1  2        y := length * cossum * 2.9 + 300;
 00397      1  2        If Theta2 <> 0 then Line(x,y);
 00398      1  2        Move(x,y)
 00399      1  1      End;
 00400      1  1      Strt := Stp;
 00401      0  0 End;
```

```
-LINE-IbC-PL-SL-

00402      0  0
00403      1  0 Procedure Get_cline;
00404    C 1  0   {Reads in a command line - immediate mode}
00405      1  1   Begin
00406      1  1      Home;
00407      1  1      Readln(Cline);
00408      1  1      If Not (Cline[1] in Commands) then
00409      1  2        Begin;
00410      1  2          Home;
00411      1  2          Writeln('                          ');
00412      1  2          Write('Invalid Command');
00413      1  2          Home;
00414      1  2          Writeln('');
00415      1  1        End;
00416      1  1      Immediate_mode := true;
00417      0  0   End;
00418      0  0
00419      1  0 Procedure Get_cline2;
00420    C 1  0   {Reads in a command line in program mode}
00421      1  1   Begin
00422      1  1      Home;
00423      1  1      Readln(Cline);
00424      1  1      Clear_screen;
00425      1  1      Found := false;
00426      1  1      k := 1;
00427      1  1      while (k<=Procs) and not(found) and (Cline[1] <> '*') do
00428      1  1               If cline = proclist[k] then found := true   else k := k + 1;
00429      1  1      If (Cline[1] <> '*') and not found then
00430      1  2        Begin
00431      1  2          writeln('Not a valid procedure name.  Valid procedures follow:');
00432      1  2          writeln(' ');
00433      1  2          for l:= 1 to procs do Writeln(Proclist[l]);
00434      1  1        End;
00435      1  1      Immediate_mode := false;
00436      0  0   End;
00437      0  0
00438      1  0 Procedure Move_teeth;
00439    C 1  0   {Rotate design in increments of "Teeth"}
00440      1  1   Begin
00441      1  1     Find_number;
00442      1  1     Rotation := Rotation + Num * (2*3.1415/a);
00443      0  0   End;
00444      0  0
00445      1  0 Procedure Specify_wheel;
00446      1  1   Begin
00447      1  1     Find_number;
00448      1  1     b := Num;
00449      0  0   End;
00450      0  0
00451      1  0 Procedure Specify_ring;
00452      1  1   Begin
00453      1  1     Find_number;
00454      1  1     a := Num;
00455      0  0   End;
00456      0  0
```

```
-LINE-IDC-PL-SL-

00457      1  0 Procedure Specify_hole;
00458      1  1   Begin
00459      1  1     Find_number;
00460      1  1     h := Num;
00461      0  0   End;
00462      0  0
00463      1  0 Procedure Specify_color;
00464      1  1   Begin
00465      1  1     Find_word;
00466      1  1     Set_color(word,1);
00467      0  0   End;
00468      0  0
00469      1  0 Procedure Reset_wheel;
00470      1  1  Begin
00471      1  1    Strt := 0;
00472      1  1    Rotation := 0;
00473      0  0  End;
00474      0  0
00475      0  0
00476      0  0
00477      1  0 Procedure Execute_command;
00478      1  1   Begin
00479      1  1      If Cline[1] = 'C' then Clear_screen
00480      1  1       Else if Cline[1] = 'G' then Go_graph
00481      1  1       Else if Cline[1] = 'W' then Specify_wheel
00482      1  1       Else if Cline[1] = 'M' then Move_teeth
00483      1  1       Else if Cline[1] = 'H' then Specify_hole
00484      1  1       Else if Cline[1] = 'R' then Specify_ring
00485      1  1       Else if Cline[1] = 'P' then Specify_color
00486      1  1       Else if Cline[1] = 'Z' then Reset_wheel
00487      0  0   End;
00488      0  0
00489      0  0
00490      0  0
00491      1  0 Procedure Fill_Valid_Lists;
00492      1  1   Begin
00493      1  1    Digits := ['0'..'9'];
00494      1  1    RealDigits := ['0'..'9','.'];
00495      1  1    Val1List[1] := 'OVAL';
00496      1  1    Val1List[2] := 'QUAD';
00497      1  1    Val1List[3] := 'TRIANGLE';
00498      1  1    Val1List[4] := 'RESET';
00499      1  1    Val1List[5] := 'END';
00500      1  1    Val1List[6] := 'STOP';
00501      1  1    Val1List[7] := 'GRAPH';
00502      1  1    Val1List[8] := 'RING';
00503      1  1    Val1List[9] := 'HOLE';
00504      1  1    Val1List[10] := 'WHEEL';
00505      1  1    Val1List[11] := 'START';
00506      1  1    Val1List[12] := 'PEN';
00507      1  1    Val1List[13] := 'MOVE';
00508      1  1    Val1List[14] := '+';
00509      1  1    Val1List[15] := '-';
00510      1  1    Val1List[16] := '*';
00511      1  1    Val1List[17] := '/';
```

```
-LINE-IDC-PL-SL-

00512      1  1    Val1List[18] := 'RUN';
00513      1  1    Val1List[19] := 'DO';
00514      1  1    Val2List[1] := 'BLUE';
00515      1  1    Val2List[2] := 'RED';
00516      1  1    Val2List[3] := 'GREEN';
00517      1  1    Val3List[1] := 'LOOPS';
00518      1  1    Val3List[2] := 'LEFT';
00519      1  1    Val3List[3] := 'RIGHT';
00520      1  1    Val3List[4] := 'HOLE';
00521      1  1    Val3List[5] := 'WHEEL';
00522      1  1    Val3List[6] := 'RING';
00523      0  0   End;
00524      0  0
00525      1  0 Procedure Read_file;
00526      1  1   Begin
00527      1  1      Reset(ProcFile);
00528      1  1      k := 1;
00529      1  1      while not eof(Procfile) and (k < 201) Do
00530      1  2      Begin
00531      1  2        Readln(Procfile,CommLine);
00532      1  2        If commline <> ' ' then
00533      1  3        Begin
00534      1  3          CommList[k] := CommLine;
00535      1  3          k := k + 1;
00536      1  2        End;
00537      1  1      End;
00538      0  0 End;
00539      0  0
00540      0  0
00541      1  0 Procedure Build_Word_Lists;
00542      1  0
00543   C  1  0  {Seperates command procedures into words and places them in }
00544   C  1  0  {lists according the number of the word.  First words are   }
00545   C  1  0  {stored in the first list.  The second word of each line is }
00546   C  1  0  {store in the second list, etc.                              }
00547      1  0
00548      1  0  Var
00549      1  0   a,o : integer;
00550      1  0
00551      1  1  Begin
00552      1  1   linecount := k - 1;
00553      1  1
00554   C  1  1   {Fill word lists with the symbol | }
00555      1  1   For k := 1 to linecount do for l := 1 to 4 do
00556      1  1      wordlist[k,l] := '|          ';
00557      1  1
00558      1  1   For k := 1 to linecount do
00559      1  2   Begin
00560      1  2     Commline := CommList[k];
00561      1  2     Wordcount := 1;
00562      1  2     a := 1;
00563      1  2     o := 1;
00564      1  2     Letcount := 0;
00565      1  2     If Commline[a] = ' ' then wordcount := wordcount - 1;
00566      1  2     while (wordcount < 5) and (a < 81) Do
```

```
-LINE-IDC-PL-SL-

00567      1  3       Begin
00568      1  3         While (Commline[a] <> ' ') and (a < 81) and (b < 11) do
00569      1  4          Begin
00570      1  4            Wordlist[k,wordcount,b] := Commline[a];
00571      1  4            a := a+1;
00572      1  4            b := b+1;
00573      1  4            letcount := letcount + 1;
00574      1  3          End;
00575      1  3         While (Commline[a] = ' ') and (a < 80) do a := a+1; {start of nxt wrd}
00576      1  3         If (letcount > 0) and (wordcount > 0)
00577      1  3                                then for l := letcount+1 to 10 do
00578      1  3                                Wordlist[k,wordcount,l] := ' '; {Pad}
00579      1  3         Wordcount := Wordcount + 1;
00580      1  3         b := 1;
00581      1  3         Letcount := 0;
00582      1  2       End;
00583      1  1   End;
00584      0  0 End;
00585      0  0
00586      1  0 Procedure Signal;
00587   C  1  0   {Error prompt}
00588      1  1   Begin
00589      1  1    writeln('>>>>> ********************************************** <<<<<');
00590      1  1    Errornum := Errornum + 1;
00591      0  0   End;
00592      0  0
00593      1  0 Procedure Syntax_check;
00594   C  1  0   {Checks word lists for illegal commands or contructs}
00595      1  1   Begin
00596      1  1   Runs := 0;
00597      1  1   Starts := 0;
00598      1  1   Dos := 0;
00599      1  1   Procs := 0;
00600      1  1   Errornum := 0;
00601      1  1     For k := 1 to Linecount do
00602      1  2      Begin
00603      1  2        writeln(Commlist[k]);
00604   C  1  2        {Verify Starts/Stops and Do/Ends}
00605      1  2        If Wordlist[k,1] = 'START' then
00606      1  3         Begin
00607      1  3           Found := false;
00608      1  3           l := 1;
00609      1  3           while (not found) and (l <= procs) do
00610      1  3             If Wordlist[k,2] = Proclist[l] then Found := true
00611      1  3             Else l := l + 1;
00612      1  3           If found then
00613      1  4             Begin
00614      1  4               Signal;
00615      1  4               Writeln(wordlist[k,2],'has already been defined as a STARTed procedure.');
00616      1  4               Writeln(' ');
00617      1  3             End;
00618      1  3           If wordlist[k,2] = ' ' then
00619      1  3             writeln('START command must be followed by a valid procedure name.');
00620      1  3           Procs := Procs + 1;
00621      1  3           Proclist[procs] := wordlist[k,2];
```

```
-LINE-IDC-PL-SL-

00622       1  3          Procline[procs] := k+1;
00623       1  3          If Dos <> 0 then Dos := 0;
00624       1  3          Starts := Starts + 1;
00625       1  3          If Starts > 1 then
00626       1  4          Begin
00627       1  4            Signal;
00628       1  4            Writeln('Missing STOP statement for procedure ',Proclist[procs-1]);
00629       1  4            Writeln(' ');
00630       1  4            Starts := 1;
00631       1  3          End;
00632       1  2        End;
00633       1  2
00634       1  2        If Wordlist[k,1] = 'STOP' then
00635       1  3         Begin
00636       1  3           If Dos > 0 then
00637       1  4             Begin
00638       1  4               Signal;
00639       1  4               Writeln('Missing END statement in procedure ',Proclist[procs]);
00640       1  4               Writeln(' ');
00641       1  3             End;
00642       1  3           If Dos < 0 then
00643       1  4             Begin
00644       1  4               Signal;
00645       1  4               Writeln('Missing DO statement in procedure ',Proclist[procs]);
00646       1  4               Writeln(' ');
00647       1  3             End;
00648       1  3           Starts := Starts - 1;
00649       1  3           If Starts < 0 then
00650       1  4             Begin
00651       1  4               Signal;
00652       1  4               Writeln('Missing START statement.');
00653       1  4               Writeln(' ');
00654       1  3             End;
00655       1  3           Starts := 0;
00656       1  2         End;
00657       1  2
00658       1  2        If Wordlist[k,1] = 'DO' then Dos := Dos + 1;
00659       1  2
00660       1  2        If Wordlist[k,1] = 'END' then Dos := Dos - 1;
00661       1  2
00662       1  2        If Wordlist[k,1] = 'RUN' then
00663       1  3        Begin
00664       1  3             Runs := Runs + 1;
00665       1  3             Runlist[Runs] := Wordlist[k,2];
00666       1  2        End;
00667       1  2
00668       1  2        L := 1;
00669       1  2        Found := false;
00670       1  2        While (L<20) and not found do
00671       1  3         Begin
00672       1  3           If ValidList[L] = Wordlist[k,1] then Found := true;
00673       1  3           L := L + 1;
00674       1  2         End;
00675       1  2        If not found then
00676       1  3          Begin
```

```
-LINE-IDC-PL-SL-

00677      1  3          Signal;
00678      1  3          Writeln('Above line begins with unknown command ',Wordlist[k,1],'.');
00679      1  3          Writeln(' ');
00680      1  3        End
00681      1  2      Else
00682      1  3        Begin
00683      1  3         If (Wordlist[k,1] = 'OVAL') or (Wordlist[k,1] = 'QUAD') or
00684      1  3            (Wordlist[k,1] = 'TRIANGLE') or (Wordlist[k,1] = 'RESET') or
00685      1  3            (Wordlist[k,1] = 'END') or (Wordlist[k,1] = 'STOP') then
00686      1  4           Begin
00687      1  4            If Column2_Not_Null(Wordlist,k) then
00688      1  5            Begin
00689      1  5              Signal;
00690      1  5              Writeln(Wordlist[k,1],' requires no operands.  Unexpected characters found.');
00691      1  5              Writeln(' ');
00692      1  4            End;
00693      1  3           End;
00694      1  3
00695      1  3      If (Wordlist[k,1] = 'RING') or (Wordlist[k,1] = 'HOLE') or
00696      1  3         (Wordlist[k,1] = 'WHEEL') then
00697      1  4        Begin
00698      1  4         If Not(Column2_not_null(Wordlist,k) ) then
00699      1  5          Begin
00700      1  5            Signal;
00701      1  5            Writeln(Wordlist[k,1],
00702      1  5                'requires an integer value.  No value found.');
00703      1  5            Writeln(' ');
00704      1  5          End
00705      1  4         Else If Not(Valid_Integer(Wordlist[k,2])) then
00706      1  5          Begin
00707      1  5            Signal;
00708      1  5            Writeln(Wordlist[k,1],
00709      1  5            'requires an integer.  The value specified is not in integer form.');
00710      1  5            Writeln(' ');
00711      1  5          End
00712      1  4         Else If Column3_not_null(Wordlist,k) then
00713      1  5          Begin
00714      1  5            Signal;
00715      1  5            Writeln('Unexpected characters found after integer value.');
00716      1  5            Writeln(' ');
00717      1  4          End;
00718      1  3        End;
00719      1  3
00720      1  3      If Wordlist[k,1] = 'PEN' then
00721      1  4        Begin
00722      1  4         If (Wordlist[k,2] <> 'RED' ) and (Wordlist[k,2] <> 'BLUE' ) and
00723      1  4            (Wordlist[k,2] <> 'GREEN' ) and (Wordlist[k,2] <> 'NEXT') then
00724      1  5          Begin
00725      1  5            Signal;
00726      1  5            Writeln('Proper pen choices are BLUE, RED, GREEN, or NEXT.');
00727      1  5            Writeln(' ');
00728      1  5          End
00729      1  4         Else If Column3_not_null(Wordlist,k) then
00730      1  5          Begin
00731      1  5            Signal;
```

```
-LINE-IDC-PL-SL-

00732      1  5              Writeln('Unexpected characters found after pen color.');
00733      1  5              Writeln(' ');
00734      1  4            End;
00735      1  3          End;
00736      1  3
00737      1  3        If (Wordlist[k,1] = 'START') or (Wordlist[k,1] = 'RUN') then
00738      1  4          Begin
00739      1  4            If Column3_not_null(Wordlist,k) then
00740      1  5             Begin
00741      1  5               Signal;
00742      1  5               Writeln('Unexpected characters found after procedure name.');
00743      1  5               Writeln(' ');
00744      1  4             End;
00745      1  3          End;
00746      1  3
00747      1  3        If Wordlist[k,1] = 'GRAPH' then
00748      1  4          Begin
00749      1  4            If Column2_not_null(Wordlist,k) then
00750      1  5             Begin
00751      1  5              If Not(Valid_Integer(Wordlist[k,2])) then
00752      1  6                Begin
00753      1  6                  Signal;
00754      1  6                  Writeln(Wordlist[k,1],
00755      1  6                  'requires an integer.  The value specified is not in integer form.');
00756      1  6                  Writeln(' ');
00757      1  6                End
00758      1  5              Else If Wordlist[k,3] <> 'LOOPS' then
00759      1  6                Begin
00760      1  6                  Signal;
00761      1  6                  Writeln('Keyword LOOPS expected after integer value.');
00762      1  6                  Writeln(' ');
00763      1  6                End
00764      1  5              Else If Column4_not_null(Wordlist,k) then
00765      1  6                Begin
00766      1  6                  Signal;
00767      1  6                  Writeln('Unexpected characters found after LOOPS.');
00768      1  6                  Writeln(' ');
00769      1  5                End;
00770      1  4            End;
00771      1  3          End;
00772      1  3
00773      1  3        If Wordlist[k,1] = 'MOVE' then
00774      1  4          Begin
00775      1  4            If Column2_not_null(Wordlist,k) then
00776      1  5             Begin
00777      1  5              If Not(Valid_Integer(Wordlist[k,2])) then
00778      1  6                Begin
00779      1  6                  Signal;
00780      1  6                  Writeln(Wordlist[k,1],
00781      1  6                  'requires an integer.  The value specified is not in integer form.');
00782      1  6                  Writeln(' ');
00783      1  6                End
00784      1  5              Else If (Wordlist[k,2] <> 'RIGHT') and
00785      1  5                      (Wordlist[k,2] <> 'LEFT')  then
00786      1  6                Begin
```

```
-LINE-IDC-PL-SL-

00787      1  6                Signal;
00788      1  6                Writeln('Keyword LEFT or RIGHT expected after integer value.');
00789      1  6                Writeln(' ');
00790      1  6              End
00791      1  5             Else If Column4_not_null(Wordlist,k) then
00792      1  6              Begin
00793      1  6                Signal;
00794      1  6                Writeln('Unexpected characters found after direction.');
00795      1  6                Writeln(' ');
00796      1  5              End;
00797      1  4            End;
00798      1  3          End;
00799      1  3
00800      1  3        If (Wordlist[k,1] = '+') or
00801      1  3           (Wordlist[k,1] = '-') or
00802      1  3           (Wordlist[k,1] = '*') or
00803      1  3           (Wordlist[k,1] = '/') then
00804      1  4          Begin
00805      1  4            If Not(Valid_Real(Wordlist[k,2])) then
00806      1  5              Begin
00807      1  5                Signal;
00808      1  5                Writeln(Wordlist[k,1],
00809      1  5                ' requires a real number.  The value specified is not in real form.');
00810      1  5                Writeln(' ');
00811      1  5              End
00812      1  4            Else If (Wordlist[k,1] <> 'HOLE') and
00813      1  4                    (Wordlist[k,3] <> 'WHEEL') and
00814      1  4                    (Wordlist[k,3] <> 'RING')  then
00815      1  5              Begin
00816      1  5                Signal;
00817      1  5                Writeln(Wordlist[k,1],' may be used to alter the values of',
00818      1  5                  ' HOLE, WHEEL, or RING only.');
00819      1  5                Writeln(' ');
00820      1  5              End
00821      1  4             Else If Column4_not_null(Wordlist,k) then
00822      1  5              Begin
00823      1  5                Signal;
00824      1  5                Writeln('Unexpected characters found after end of command.');
00825      1  5                Writeln(' ');
00826      1  4              End;
00827      1  3          End;
00828      1  2      End;
00829      1  1    End;
00830      1  1
00831      1  1    If Dos > 0 then
00832      1  2       Begin
00833      1  2         Signal;
00834      1  2         Writeln('Missing END statement in procedure ',Proclist[procs]);
00835      1  2         Writeln(' ');
00836      1  1       End;
00837      1  1    If Dos < 0 then
00838      1  2       Begin
00839      1  2         Signal;
00840      1  2         Writeln('Missing DO statement in procedure ',Proclist[procs]);
00841      1  2         Writeln(' ');
```

```
-LINE-IDC-PL-SL-

00842      1  1         End;
00843      1  1      If Starts > 0 then
00844      1  2         Begin
00845      1  2           Signal;
00846      1  2           Writeln('Missing STOP statement for procedure ',Proclist[procs]);
00847      1  2           Writeln(' ');
00848      1  1         End;
00849      0  0    End;
00850      0  0
00851      1  0 Procedure Verify_runs;
00852    C 1  0    {Check to see if all procedures called are defined in the proclist}
00853      1  1    Begin
00854      1  1      For l := 1 to Runs do
00855      1  2      Begin
00856      1  2        Found := false;
00857      1  2        k := 1;
00858      1  2        While (not found) and (k <= procs) do
00859      1  2           If Runlist[l] = Proclist[k] then Found := true
00860      1  2             Else k := k + 1;
00861      1  2        If not found then Writeln(Runlist[l],'not defined as a STARTed procedure.');
00862      1  1      End;
00863      0  0    End;
00864      0  0
00865      1  0 Procedure Error_report;
00866      1  1    Begin
00867      1  1      Error_free := false;
00868      1  1      Writeln(' ');
00869      1  1      If errornum= 0 then
00870      1  2         Begin
00871      1  2           Writeln('No errors found.');
00872      1  2           Error_free := true;
00873      1  2         End
00874      1  1         else Writeln(errornum :3,' errors found.');
00875      0  0    End;
00876      0  0
00877      0  0
00878      1  0 Procedure Load_file_and_syntax_check;
00879      1  1   Begin
00880      1  1     Fill_Valid_Lists;
00881      1  1     Read_file;
00882      1  1     Build_Word_Lists;
00883      1  1     Syntax_check;
00884      1  1     Verify_Runs;
00885      1  1     Error_report;
00886      0  0   End;
00887      0  0
00888      1  0 Procedure Clear_stacks;
00889      1  1    Begin
00890      1  1      For pointer := 1 to 20 do
00891      1  2        begin
00892      1  2          PC_Stack[pointer] := 0;
00893      1  2          Counter_Stack[pointer] := 0;
00894      1  1        end;
00895      1  1      Pointer := 0;
00896      0  0    End;
```

```
-LINE-IDC-PL-SL-

00897       0  0
00898       1  0 Procedure Push_PC_and_count;
00899    C  1  0   {Place return line number and times to repeat on stack}
00900       1  1   Begin
00901       1  1     Pointer := Pointer + 1;
00902       1  1     PC_Stack[pointer] := PC + 1;
00903       1  1     If Wordlist[PC,2] = 'RUN' then Counter_Stack[pointer] := 0
00904       1  1      Else Counter_Stack[pointer] := Value_Integer(Wordlist[PC,2]) - 1;
00905       1  1     If Pointer > 20 then Writeln('Fatal Error:  Maximum nesting level exceeded.');
00906       0  0   End;
00907       0  0
00908       1  0 Procedure Pop_PC_and_count;
00909    C  1  0   {Remove items from stack}
00910       1  1   Begin
00911       1  1     Pointer := Pointer - 1;
00912       0  0   End;
00913       0  0
00914       1  0 Procedure Execute_program_command;
00915       1  1   Begin
00916       1  1     If Wordlist[PC,1] = 'RESET' then Reset_wheel;
00917       1  1
00918       1  1     If (Wordlist[PC,1] = 'GRAPH') and (Wordlist[PC,3] = 'LOOPS') then
00919       1  2       Begin
00920       1  2         Num := Value_integer(Wordlist[PC,2]);
00921       1  2         Go_graph;
00922       1  2       End
00923       1  1     Else If (Wordlist[PC,1] = 'GRAPH') then
00924       1  2       Begin
00925       1  2         Num := 0;
00926       1  2         Go_graph;
00927       1  1       End;
00928       1  1
00929       1  1     If Wordlist[PC,1] = 'PEN' then
00930       1  2        Begin
00931       1  2          Word := Wordlist[PC,2];
00932       1  2          If Word = 'NEXT' then
00933       1  3            Begin
00934       1  3              If OldColor = 'BLUE' then Word := 'GREEN'
00935       1  3              Else If OldColor = 'GREEN' then Word := 'RED'
00936       1  3              Else If OldColor = 'RED' then Word := 'BLUE'
00937       1  2            End;
00938       1  2          OldColor := Word;
00939       1  2          Set_color(Word,1);
00940       1  1        End;
00941       1  1
00942       1  1     If Wordlist[PC,1] = 'RING' then a := Value_Integer(Wordlist[PC,2]);
00943       1  1
00944       1  1     If Wordlist[PC,1] = 'WHEEL' then b := Value_Integer(Wordlist[PC,2]);
00945       1  1
00946       1  1     If Wordlist[PC,1] = 'HOLE' then h := Value_Integer(Wordlist[PC,2]);
00947       1  1
00948       1  1     If (Wordlist[PC,1] = 'MOVE') and (Wordlist[PC,1] = 'LEFT') then
00949       1  1         Rotation := Rotation + Value_Integer(Wordlist[PC,2]) * (2*3.1415/a)
00950       1  1     Else If (Wordlist[PC,1] = 'MOVE') then
00951       1  1         Rotation := Rotation - Value_Integer(Wordlist[PC,2]) * (2*3.1415/a);
```

```
-LINE-IDC-PL-SL-

00952       1  1
00953       1  1     If (Wordlist[PC,1] = '+') then
00954       1  2        Begin
00955       1  2          If (Wordlist[PC,3] = 'HOLE') then h := h + Value_Real(Wordlist[PC,2])
00956       1  2          Else If (Wordlist[PC,3] = 'WHEEL') then
00957       1  2                  b := b + Value_Real(Wordlist[PC,2])
00958       1  2          Else If (Wordlist[PC,3] = 'RING') then
00959       1  2                  a := a + Value_Real(Wordlist[PC,2]);
00960       1  1        End;
00961       1  1
00962       1  1     If (Wordlist[PC,1] = '-') then
00963       1  2        Begin
00964       1  2          If (Wordlist[PC,3] = 'HOLE') then h := h - Value_Real(Wordlist[PC,2])
00965       1  2          Else If (Wordlist[PC,3] = 'WHEEL') then
00966       1  2                  b := b - Value_Real(Wordlist[PC,2])
00967       1  2          Else If (Wordlist[PC,3] = 'RING') then
00968       1  2                  a := a - Value_Real(Wordlist[PC,2]);
00969       1  1        End;
00970       1  1
00971       1  1     If (Wordlist[PC,1] = '*') then
00972       1  2        Begin
00973       1  2          If (Wordlist[PC,3] = 'HOLE') then h := h * Value_Real(Wordlist[PC,2])
00974       1  2          Else If (Wordlist[PC,3] = 'WHEEL') then
00975       1  2                  b := b * Value_Real(Wordlist[PC,2])
00976       1  2          Else If (Wordlist[PC,3] = 'RING') then
00977       1  2                  a := a * Value_Real(Wordlist[PC,2]);
00978       1  1        End;
00979       1  1
00980       1  1     If (Wordlist[PC,1] = '/') then
00981       1  2        Begin
00982       1  2          If (Wordlist[PC,3] = 'HOLE') then h := h / Value_Real(Wordlist[PC,2])
00983       1  2          Else If (Wordlist[PC,3] = 'WHEEL') then
00984       1  2                  b := b / Value_Real(Wordlist[PC,2])
00985       1  2          Else If (Wordlist[PC,3] = 'RING') then
00986       1  2                  a := a / Value_Real(Wordlist[PC,2]);
00987       1  1        End;
00988       1  1
00989       0  0   End;
00990       0  0
00991       1  0 Procedure Execute_program;
00992       1  1   Begin
00993       1  1     PC := Procline[k];
00994       1  1     While not( (pointer = 0) and (Wordlist[PC,1] = 'STOP') ) do
00995       1  2       Begin
00996       1  2         If Wordlist[PC,1] = 'DO' then
00997       1  3           Begin
00998   C   1  3            {Place return line and repeat number on stack}
00999       1  3            Push_PC_and_Count;
01000       1  3           End
01001       1  2         Else if Wordlist[PC,1] = 'END' then
01002       1  3           Begin
01003       1  3            If Counter_Stack[pointer] > 0 then
01004       1  4              Begin
01005   C   1  4                {Decrement number to repeat and go back}
01006       1  4                Counter_Stack[pointer] := Counter_Stack[pointer] - 1;
```

```
-LINE-IDC-PL-SL-

01007      1  4                PC := PC_Stack[pointer] - 1;
01008      1  4              End
01009      1  3            Else If Counter_Stack[pointer] = 0 then Pop_PC_and_Count;
01010      1  3          End
01011      1  2        Else if Wordlist[PC,1] = 'RUN' then
01012      1  3          Begin
01013      1  3            Found := false;
01014      1  3            k := 1;
01015      1  3            while (not found) and (k <= procs) do
01016      1  3               If (Wordlist[PC,2] = Proclist[k]) then Found := true
01017      1  3               Else k := k + 1;
01018      1  3            PC := PC - 1;
01019      1  3            Push_PC_and_Count;
01020      1  3            PC := Procline[k] - 1;
01021      1  3          End
01022      1  2        Else if Wordlist[PC,1] = 'STOP' then
01023      1  3          Begin
01024      1  3            PC := PC_Stack[pointer];
01025      1  3            Pop_PC_and_Count;
01026      1  2          End;
01027      1  2        Execute_Program_command;
01028      1  2        PC := PC + 1;
01029      1  1      End; {While}
01030      0  0   End;
01031      0  0
01032      1  0 Procedure Do_Programs;
01033      1  1  Begin
01034      1  1    Load_file_and_syntax_check;
01035      1  1    If error_free then
01036      1  2    Begin
01037      1  2       Clear_stacks;
01038      1  2       Get_cline2;
01039      1  2       while (Cline[1] <> '*') do
01040      1  *         Begin
01041      1  3            If found then Execute_program;
01042      1  3            Get_cline2;
01043      1  2         End;
01044      1  1    End;
01045      0  0  End;
01046      0  0
01047      0  1 Begin {Main}
01048   C  0  1      {Set up graphics and graphics window}
01049      0  1        Init_graphics;
01050      0  1        Clear_screen;
01051      0  1        Set_window(0.0,0.0,1000.0,625.0);
01052      0  1
01053   C  0  1      {Defaults}
01054      0  1        Oldcolor := 'GREEN';
01055      0  1        Rotation := 0;
01056      0  1        Strt := 0;
01057      0  1        a := 96;
01058      0  1        b := 24;
01059      0  1        h := 1;
01060      0  1
01061      0  1      Error_free := true;
```

```
-LINE-IDC-PL-SL-

01062       0  1      Commands := ['*','O','M','C','W','H','R','X','E','G','P','Z'];
01063       0  1      While (Cline[1] <> 'E') Do
01064       0  2      Begin;
01065       0  2        While Not (Cline[1] in Commands) Do Get_cline;
01066       0  2        If (Cline[1] = '*') and (error_free)
01067       0  2              then Do_Programs else
01068       0  2           If (Cline[1] = '*') then Writeln('Program contains errors.');
01069       0  2        If (cline[1] <> '*') then Execute_command;
01070       0  2        Home;
01071       0  2        If Cline[1] <> 'E' then Cline := '';
01072       0  1      End;
01073       0  0 End.
```

```
A                       VAR   [PSECT(SLOCAL)] SINGLE ( IN PROGRAM CYCLOGRAPH )
                               87      307     307     358     385     386     386     387     387     442
                              454 =    942 =   949     951     959 =   959     959     968 =   968     968
                              977 =    977     977     986 =   986     986     1057 =
A                       BOUND INTEGER ( IN PROCEDURE TXT )
                              172
A                       BOUND INTEGER ( IN PROCEDURE SET_COLOR )
                              176
A                       VAR   INTEGER ( IN FUNCTION VALID_INTEGER )
                              265     270 =   271     271     272     277 =   277     277
A                       VAR   INTEGER ( IN FUNCTION VALID_REAL )
                              283     288 =   290     290     292     293     297 =   297     297
A                       VAR   INTEGER ( IN PROCEDURE BUILD_WORD_LISTS )
                              549     562 =   565     566     568     568     570     571 =   571     571
                              575     575     575 =   575     575
A2                      VAR   INTEGER ( IN PROGRAM CYCLOGRAPH )
                              110
ABS                     FUNC  ( BUILTIN ) : INTEGER
                                *     312
B                       VAR   [PSECT(SLOCAL)] SINGLE ( IN PROGRAM CYCLOGRAPH )
                               88     307     358     364     365     385     385     386     386     386
                              386     387     387     387     387     448 =   944 =   957 =   957     957
                              966 =   966     966     975 =   975     975     984 =   984     984     1058 =
B                       BOUND INTEGER ( IN PROCEDURE TXT )
                              172
B                       BOUND INTEGER ( IN PROCEDURE SET_COLOR )
                              176
B                       VAR   INTEGER ( IN PROCEDURE BUILD_WORD_LISTS )
                              549     563 =   568     570     572 =   572     572     580 =
B2                      VAR   INTEGER ( IN PROGRAM CYCLOGRAPH )
                              111
BETA                    VAR   [PSECT(SLOCAL)] SINGLE ( IN PROGRAM CYCLOGRAPH )
                               83     385 =   386     387
BOOLEAN                 TYPE
                                *      54      105     106     203     241     248     255     262     266
                              280     284
BUILD_WORD_LISTS        PROC  [PSECT(SCODE), UNBOUND]  ( IN PROGRAM CYCLOGRAPH )
                              541     582
CHAR                    TYPE
                                *      38      39      40      41      56      64      65      66      128
                              129     172     176     192
CLEAR_SCREEN            PROC  [EXTERNAL, UNBOUND]  ( IN PROGRAM CYCLOGRAPH )
                              152     424     479     1050
CLEAR_STACKS            PROC  [PSECT(SCODE), UNBOUND]  ( IN PROGRAM CYCLOGRAPH )
                              888     1037
CLINE                   VAR   [PSECT(SLOCAL)] ARRAY OF CHAR ( IN PROGRAM CYCLOGRAPH )
                               65     325     327     327     407 R   408     423 R   427     428     429
                              479     480     481     482     483     484     485     486     1039    1063
                             1065    1066    1068    1069    1071    1071 =
COLUMN2_NOT_NULL        FUNC  [PSECT(SCODE), UNBOUND] : BOOLEAN ( IN PROGRAM CYCLOGRAPH )
                              241     244 =   245 =   687     698     749     775
COLUMN3_NOT_NULL        FUNC  [PSECT(SCODE), UNBOUND] : BOOLEAN ( IN PROGRAM CYCLOGRAPH )
                              248     251 =   252 =   712     729     739
COLUMN4_NOT_NULL        FUNC  [PSECT(SCODE), UNBOUND] : BOOLEAN ( IN PROGRAM CYCLOGRAPH )
                              255     258 =   259 =   764     791     821
COMMAND                 VAR   CHAR ( IN PROGRAM CYCLOGRAPH )
                               56
```

```
COMMANDS                VAR   [PSECT($LOCAL)] SETOFCHARACTERS ( IN PROGRAM CYCLOGRAPH )
                               81      408     1062 =  1065
COMMLINE                VAR   [PSECT($LOCAL)] LINES ( IN PROGRAM CYCLOGRAPH )
                              124      531 R    532     534     534     560 =   565     568     570     570
                              575
COMMLIST                VAR   [PSECT($LOCAL)] LISTS ( IN PROGRAM CYCLOGRAPH )
                              126      534 =    560     560     603
COS                     FUNC  ( BUILTIN ) : SINGLE
                                *      386      386     387     392     392
COS1                    VAR   [PSECT($LOCAL)] SINGLE ( IN PROGRAM CYCLOGRAPH )
                               71      389 =    393     394
COSR                    VAR   [PSECT($LOCAL)] SINGLE ( IN PROGRAM CYCLOGRAPH )
                               69      392 =    393     393     394
COSSUM                  VAR   [PSECT($LOCAL)] SINGLE ( IN PROGRAM CYCLOGRAPH )
                               70      393 =    396
COUNTER_STACK           VAR   [PSECT($LOCAL)] ARRAY OF INTEGER ( IN PROGRAM CYCLOGRAPH )
                              101      893 =    903 =   904 =   1003    1006 =  1006    1006    1009
CYCLES                  VAR   [PSECT($LOCAL)] SINGLE ( IN PROGRAM CYCLOGRAPH )
                               68      308 =    310     312 =   358 =   372 =   372     372     374     374
                              378 =    378      378     379     379
CYCLOGRAPH              PROG  [PSECT($CODE)]
                               35
DDLISTS                 TYPE  ARRAY ( IN PROGRAM CYCLOGRAPH )
                               44      108      241     248     255
DIGITS                  VAR   [PSECT($LOCAL)] SET OF CHAR ( IN PROGRAM CYCLOGRAPH )
                              128      272      493 =
DOS                     VAR   [PSECT($LOCAL)] INTEGER ( IN PROGRAM CYCLOGRAPH )
                              112      598 =    623     623 =   636     642     658 =   658     658     660 =
                              660      660      831     837
DO_PROGRAMS             PROC  [PSECT($CODE), UNBOUND]  ( IN PROGRAM CYCLOGRAPH )
                             1032     1067
E                       VAR   [PSECT($LOCAL)] SINGLE ( IN PROGRAM CYCLOGRAPH )
                               89      367 =    386     387
EOF                     FUNC  ( BUILTIN ) : BOOLEAN
                                *      529
ERRORNUM                VAR   [PSECT($LOCAL)] INTEGER ( IN PROGRAM CYCLOGRAPH )
                              113      590 =    590     59?     603 =   869     874
ERROR_FREE              VAR   [PSECT($LOCAL)] BOOLEAN ( IN PROGRAM CYCLOGRAPH )
                              105      867 =    872 =   1035    1061 =  1066
ERROR_REPORT            PROC  [PSECT($CODE), UNBOUND]  ( IN PROGRAM CYCLOGRAPH )
                              865      885
EXECUTE_COMMAND         PROC  [PSECT($CODE), UNBOUND]  ( IN PROGRAM CYCLOGRAPH )
                              477     1069
EXECUTE_PROGRAM         PROC  [PSECT($CODE), UNBOUND]  ( IN PROGRAM CYCLOGRAPH )
                              991     1041
EXECUTE_PROGRAM_COMMAND PROC  [PSECT($CODE), UNBOUND]  ( IN PROGRAM CYCLOGRAPH )
                              914     1027
FALSE                   CONST 'FALSE = 0'
                                *      208      208     244     244     251     251     258     258     273
                              273      274      274     294     294     295     295     425     425     435
                              435      607      607     669     669     856     856     867     867    1013
                             1013
FILL_VALID_LISTS        PROC  [PSECT($CODE), UNBOUND]  ( IN PROGRAM CYCLOGRAPH )
                              491      530
FIND_CLOSURE_CYCLES     PROC  [PSECT($CODE), UNBOUND]  ( IN PROGRAM CYCLOGRAPH )
                              304      371
FIND_LOOP_CYCLES        PROC  [PSECT($CODE), UNBOUND]  ( IN PROGRAM CYCLOGRAPH )
```

```
                        355     377
FIND_NUMBER             PROC    [PSECT($CODE), UNBOUND]  { IN PROGRAM CYCLOGRAPH }
                        334     369     441     447     453     459
FIND_WORD               PROC    [PSECT($CODE), UNBOUND]  { IN PROGRAM CYCLOGRAPH }
                        319     337     465
FOUND                   VAR     [PSECT($LOCAL)] BOOLEAN { IN PROGRAM CYCLOGRAPH }
                        106     425 =   427     428 =   429     607 =   609     610 =   612     669 =
                        670     672 =   675     856 =   858     859 =   861     1013 =  1015    1016 =
                       1041
FOUND_AGAIN             VAR     BOOLEAN { IN FUNCTION VALUE_REAL }
                        203     216     213 =
GET_CLINE               PROC    [PSECT($CODE), UNBOUND]  { IN PROGRAM CYCLOGRAPH }
                        403    1065
GET_CLINE2              PROC    [PSECT($CODE), UNBOUND]  { IN PROGRAM CYCLOGRAPH }
                        419    1038    1042
GO_GRAPH                PROC    [PSECT($CODE), UNBOUND]  { IN PROGRAM CYCLOGRAPH }
                        361     480     921     926
H                       VAR     [PSECT($LOCAL)] SINGLE { IN PROGRAM CYCLOGRAPH }
                         90     367     460 =   946 =   955 =   955     955     964 =   964     964
                        973 =   973     973     982 =   982     982    1059 =
HOME                    PROC    [EXTERNAL, UNBOUND]  { IN PROGRAM CYCLOGRAPH }
                        144     406     410     413    1070
HOME2                   PROC    [EXTERNAL, UNBOUND]  { IN PROGRAM CYCLOGRAPH }
                        148     422
IMMEDIATE_MODE          VAR     [PSECT($LOCAL)] BOOLEAN { IN PROGRAM CYCLOGRAPH }
                         54     369     416 =   435 =
INIT_GRAPHICS           PROC    [EXTERNAL, UNBOUND]  { IN PROGRAM CYCLOGRAPH }
                        156    1049
INPUT                   VAR     [EXTERNAL(PAS$FV_INPUT)] TEXT { IN PROGRAM CYCLOGRAPH }
                          *      35     407     423
INTEGER                 TYPE
                          *      58      59      60      61      62     101     102     103     110
                        111     112     113     114     115     116     117     118     119     120
                        121     122     172     176     177     186     192     201     226     229
                        241     248     255     265     283     549
J                       VAR     [PSECT($LOCAL)] SINGLE { IN PROGRAM CYCLOGRAPH }
                         91     368 =   386     387
K                       VAR     [PSECT($LOCAL)] INTEGER { IN PROGRAM CYCLOGRAPH }
                        114     426 =   427     428     428 =   428     428     528 =   529     534
                        535 =   535     535     552     552     555 =   556     558 =   560     570
                        578     601 =   603     605     610     615     618     621     622     622
                        634     658     660     662     665     672     678     683     683     684
                        684     685     685     687     690     695     695     696     698     701
                        705     708     712     720     722     722     723     723     729     737
                        737     739     747     749     751     754     758     764     773     775
                        777     780     784     785     791     800     801     802     803     805
                        808     812     813     814     817     821     857 =   858     859     860 =
                        860     860     993    1014 =  1015    1016    1017 =  1017    1017    1020
K                       PARM    INTEGER { IN FUNCTION COLUMN2_NOT_NULL }
                        241     245
K                       PARM    INTEGER { IN FUNCTION COLUMN3_NOT_NULL }
                        248     252
K                       PARM    INTEGER { IN FUNCTION COLUMN4_NOT_NULL }
                        255     259
L                       VAR     [PSECT($LOCAL)] INTEGER { IN PROGRAM CYCLOGRAPH }
                         88     324 =   325     327     329 =   329     329     433 =   433     555 =
                        556     577 =   578     603 =   609     610     611 =   611     611     668 =
```

```
                          670      672 -    673 =    673      673      854 =    859      861
LCOUNT                    VAR   [PSECT($LOCAL)] INTEGER ( IN PROGRAM CYCLOGRAPH )
                           59      323 =    327      328 =    328      328      341 =    346      347 =    347
                          347
LENGTH                    VAR   [PSECT($LOCAL)] SINGLE ( IN PROGRAM CYCLOGRAPH )
                           72      388 =    389      390      395      396      396
LETCOUNT                  VAR   [PSECT($LOCAL)] INTEGER ( IN PROGRAM CYCLOGRAPH )
                          116      564 =    573 =    573      573      576      577      581 =
LINE                      PROC  [EXTERNAL, UNBOUND]  ( IN PROGRAM CYCLOGRAPH )
                          160      397
LINECOUNT                 VAR   [PSECT($LOCAL)] INTEGER ( IN PROGRAM CYCLOGRAPH )
                          115      552 =    555      558      601
LINES                     TYPE  ARRAY OF CHAR ( IN PROGRAM CYCLOGRAPH )
                           40       43      124
LISTS                     TYPE  ARRAY OF LINES ( IN PROGRAM CYCLOGRAPH )
                           43      126
LOAD_FILE_AND_SYNTAX_CHECK PROC [PSECT($CODE), UNBOUND]  ( IN PROGRAM CYCLOGRAPH )
                          373     1034
M                         VAR   [PSECT($LOCAL)] INTEGER ( IN PROGRAM CYCLOGRAPH )
                           60      309 =    311      312      312      312      313 =    313      313      340 =
                          342      344      346      349 =    349      349
M                         VAR   INTEGER ( IN FUNCTION VALUE_REAL )
                          201      207 =    209      211      212      212      213 =    213      213      215 =
                          218      219      219      220      221 =    221      221
M                         VAR   INTEGER ( IN FUNCTION VALUE_INTEGER )
                          229      232 =    233      235      235      236 =    236      236
MESSAGE                   PARM  CONFORMANT ARRAY OF CHAR ( IN PROCEDURE TXT )
                          172
MOVE                      PROC  [EXTERNAL, UNBOUND]  ( IN PROGRAM CYCLOGRAPH )
                          164      398
MOVE_TEETH                PROC  [PSECT($CODE), UNBOUND]  ( IN PROGRAM CYCLOGRAPH )
                          433      482
N                         PARM  INTEGER ( IN FUNCTION POWER )
                          186      189      189
NUM                       VAR   [PSECT($LOCAL)] SINGLE ( IN PROGRAM CYCLOGRAPH )
                           54      332 =    346 =    346      358      370      442      448      448      454
                          454      460      463      920 =    925 =
NUM                       VAR   INTEGER ( IN FUNCTION VALUE_REAL )
                          201      205 =    211 =    212      212      223      223
NUM                       VAR   INTEGER ( IN FUNCTION VALUE_INTEGER )
                          229      231 =    235 =    235      235      238      238
NUM2                      VAR   SINGLE ( IN FUNCTION VALUE_REAL )
                          202      206 =    220 =    220      220      223
NUMBER                    PARM  INTEGER ( IN PROCEDURE SET_COLOR )
                          177
NUMHOLE                   VAR   [PSECT($LOCAL)] INTEGER ( IN PROGRAM CYCLOGRAPH )
                           61      365 =    366      366 =    367
OLDCOLOR                  VAR   [PSECT($LOCAL)] ARRAY OF CHAR ( IN PROGRAM CYCLOGRAPH )
                           54      934      935      936      938 =   1054 =
ORD                       FUNC  ( BUILTIN ) : INTEGER
                            *      195      195      195
OUTPUT                    VAR   [EXTERNAL(PASSFV_OUTPUT)] TEXT ( IN PROGRAM CYCLOGRAPH )
                            *       35      411      412      414      431      432      433      589      603
                          615      616      619      628      629      639      640      645      646      652
                          653      678      679      690      691      701      703      708      710      715
                          719      726      727      732      733      742      743      754      756      761
                          762      767      768      780      782      783      789      794      795      808
```

```
                        810     817     819     824     825     834     835     840     841     846
                        847     861     868     871     874     905    1068
PC                  VAR [PSECT($LOCAL)] INTEGER ( IN PROGRAM CYCLOGRAPH )
                        117     902     902     903     904     916     918     918     920     923
                        929     931     942     942     944     944     946     946     948     948
                        949     950     951     953     955     955     956     957     958     959
                        962     964     964     965     966     967     968     971     973     973
                        974     975     976     977     980     982     982     983     984     985
                        986     993 =   994     996    1001    1007 =  1011    1016    1018 =  1018
                       1018    1020 =  1022    1024 =  1028 =  1028    1028
PC_STACK            VAR [PSECT($LOCAL)] ARRAY OF INTEGER ( IN PROGRAM CYCLOGRAPH )
                        102     892 =   902 =  1007    1007    1024    1024
POINTCOUNT          VAR INTEGER ( IN FUNCTION VALID_REAL )
                        283     289 =   292 =   292     292     293
POINTER             VAR [PSECT($LOCAL)] INTEGER ( IN PROGRAM CYCLOGRAPH )
                        119     890 =   892     893     895 =   901 =   901     901     902     903
                        904     905     911 =   911     911     994    1003    1006    1006    1007
                       1009    1024
POINT_FOUND         VAR BOOLEAN ( IN FUNCTION VALUE_REAL )
                        203     208 =   209     211 =   216
POP_PC_AND_COUNT    PROC [PSECT($CODE), UNBOUND]  ( IN PROGRAM CYCLOGRAPH )
                        903    1009    1025
POWER               FUNC [PSECT($CODE), UNBOUND] : SINGLE ( IN PROGRAM CYCLOGRAPH )
                        186     189 =   189 =   189     346
PROCFILE            VAR [PSECT($LOCAL)] TEXT ( IN PROGRAM CYCLOGRAPH )
                        131      35     527     529     531
PROCLINE            VAR [PSECT($LOCAL)] ARRAY OF INTEGER ( IN PROGRAM CYCLOGRAPH )
                        103     822 =   993     993    1020    1020
PROCLIST            VAR [PSECT($LOCAL)] WORD_LIST ( IN PROGRAM CYCLOGRAPH )
                        133     423     433     610     621 =   628     639     645     834     840
                        846     859    1016
PROCS               VAR [PSECT($LOCAL)] INTEGER ( IN PROGRAM CYCLOGRAPH )
                        113     427     433     599 =   609     620 =   620     620     621     622
                        638     639     645     834     840     846     858    1015
PUSH_PC_AND_COUNT   PROC [PSECT($CODE), UNBOUND]  ( IN PROGRAM CYCLOGRAPH )
                        998     999    1019
RADIUS              VAR [PSECT($LOCAL)] SINGLE ( IN PROGRAM CYCLOGRAPH )
                         35     364 =   367
RATIO               VAR [PSECT($LOCAL)] SINGLE ( IN PROGRAM CYCLOGRAPH )
                         36     307 =   312     312
READLN              PROC ( BUILTIN )
                          *     407     423     531
READ_FILE           PROC [PSECT($CODE), UNBOUND]  ( IN PROGRAM CYCLOGRAPH )
                        535     881
REAL                TYPE SINGLE
                          *      68      69      70      71      72      73      74      75      76
                         77      78      79     186     186     198     202
REALDIGITS          VAR [PSECT($LOCAL)] SET OF CHAR ( IN PROGRAM CYCLOGRAPH )
                        129     293     494 =
RESET               PROC ( BUILTIN )
                          *     527
RESET_WHEEL         PROC [PSECT($CODE), UNBOUND]  ( IN PROGRAM CYCLOGRAPH )
                        469     486     916
ROTATION            VAR [PSECT($LOCAL)] SINGLE ( IN PROGRAM CYCLOGRAPH )
                         73     391     392     442 =   442     442     472 =   949 =   949     949
                        951 =   951     951    1055 =
RUNLIST             VAR [PSECT($LOCAL)] WORD_LIST ( IN PROGRAM CYCLOGRAPH )
```

```
                        134     665 =   859     861
RUNS                    VAR   [PSECT($LOCAL)] INTEGER { IN PROGRAM CYCLOGRAPH }
                        120     596 =   664 =   664     664     665     854
SETOFCHARACTERS         TYPE  SET OF CHAR { IN PROGRAM CYCLOGRAPH }
                         38      81
SET_COLOR               PROC  [EXTERNAL, UNBOUND]  { IN PROGRAM CYCLOGRAPH }
                        176     466     939
SET_WINDOW              PROC  [EXTERNAL, UNBOUND]  { IN PROGRAM CYCLOGRAPH }
                        166    1051
SIGNAL                  PROC  [PSECT($CODE), UNBOUND]  { IN PROGRAM CYCLOGRAPH }
                        586     614     627     638     644     651     677     689     700     707
                        714     725     731     741     753     760     766     779     787     793
                        807     816     823     833     839     845
SIN                     FUNC  { BUILTIN } : SINGLE
                          *     386     387     387     391     391
SIN1                    VAR   [PSECT($LOCAL)] SINGLE { IN PROGRAM CYCLOGRAPH }
                         77     390 =   393     394
SINGLE                  TYPE  F_FLOATING
                          *      83      84      85      86      87      88      89      90      91
                         92     160     160     164     164     168
SINR                    VAR   [PSECT($LOCAL)] SINGLE { IN PROGRAM CYCLOGRAPH }
                         76     391 =   393     394     394
SINSUM                  VAR   [PSECT($LOCAL)] SINGLE { IN PROGRAM CYCLOGRAPH }
                         74     394 =   395
SPECIFY_COLOR           PROC  [PSECT($CODE), UNBOUND]  { IN PROGRAM CYCLOGRAPH }
                        453     485
SPECIFY_HOLE            PROC  [PSECT($CODE), UNBOUND]  { IN PROGRAM CYCLOGRAPH }
                        457     483
SPECIFY_RING            PROC  [PSECT($CODE), UNBOUND]  { IN PROGRAM CYCLOGRAPH }
                        451     484
SPECIFY_WHEEL           PROC  [PSECT($CODE), UNBOUND]  { IN PROGRAM CYCLOGRAPH }
                        445     481
SQR                     FUNC  { BUILTIN } : SINGLE
                          *     388     388
STARTS                  VAR   [PSECT($LOCAL)] INTEGER { IN PROGRAM CYCLOGRAPH }
                        121     597 =   624 =   624     624     625     630 =   648 =   648     648
                        649     655 =   343
STP                     VAR   [PSECT($LOCAL)] SINGLE { IN PROGRAM CYCLOGRAPH }
                         75     374 =   379 =   381     400     400
STRING                  TYPE  ARRAY OF CHAR { IN PROGRAM CYCLOGRAPH }
                         39
STRING                  PARM  CONFORMANT ARRAY OF CHAR { IN PROCEDURE SET_COLOR }
                        176
STRT                    VAR   [PSECT($LOCAL)] SINGLE { IN PROGRAM CYCLOGRAPH }
                         73     373 =   379     381     400 =   471 =   1056 =
SYNTAX_CHECK            PROC  [PSECT($CODE), UNBOUND]  { IN PROGRAM CYCLOGRAPH }
                        593     383
TEXT                    TYPE
                          *     131
THETA                   VAR   [PSECT($LOCAL)] SINGLE { IN PROGRAM CYCLOGRAPH }
                         79     383 =   384 =   384     384     385     385     386     386     387
                        387
THETA2                  VAR   [PSECT($LOCAL)] INTEGER { IN PROGRAM CYCLOGRAPH }
                         82     381 =   387     384     397
TRUE                    CONST 'TRUE = 1'
                          *     211     211     218     218     245     245     252     252     259
                        259     268     269     269     269     236     236     237     237     414
```

```
                          416      428 -    428      610      610      672      672      859      859      872
                          872     1016     1016     1061     1061
TRUNC                FUNC  ( BUILTIN ) : INTEGER
                            *      312      365      365      381      381
TXT                  PROC  [EXTERNAL, UNBOUND]  ( IN PROGRAM CYCLOGRAPH )
                          172
VAL1LIST             VAR   [PSECT($LOCAL)] WORD_LIST ( IN PROGRAM CYCLOGRAPH ) -
                          135      495 =    496 =    497 =    498 =    499 =    500 =    501 =    502 =    503 =
                          504 =    505 =    506 =    507 =    508 =    509 =    510 =    511 =    512 =    513 =
                          672
VAL2LIST             VAR   [PSECT($LOCAL)] WORD_LIST ( IN PROGRAM CYCLOGRAPH )
                          136      514 =    515 =    516 =
VAL3LIST             VAR   [PSECT($LOCAL)] WORD_LIST ( IN PROGRAM CYCLOGRAPH )
                          137      517 =    518 =    519 =    520 =    521 =    522 =
VALID_BOOL           VAR   BOOLEAN ( IN FUNCTION VALID_INTEGER )
                          266      269 =    271      274 =
VALID_BOOL           VAR   BOOLEAN ( IN FUNCTION VALID_REAL )
                          284      287 =    290      295 =
VALID_INTEGER        FUNC  [PSECT($CODE), UNBOUND] : BOOLEAN ( IN PROGRAM CYCLOGRAPH )
                          262      268 =    273 =    705      751      777
VALID_REAL           FUNC  [PSECT($CODE), UNBOUND] : BOOLEAN ( IN PROGRAM CYCLOGRAPH )
                          280      286 =    294 =    805
VALUE                FUNC  [PSECT($CODE), UNBOUND] : INTEGER ( IN PROGRAM CYCLOGRAPH )
                          192      195 =    212      220      235      346      346
VALUE_INTEGER        FUNC  [PSECT($CODE), UNBOUND] : INTEGER ( IN PROGRAM CYCLOGRAPH )
                          226      238 =    904      904      920      920      942      942      944      944
                          946      946      949      951
VALUE_REAL           FUNC  [PSECT($CODE), UNBOUND] : SINGLE ( IN PROGRAM CYCLOGRAPH )
                          198      223 =    955      957      959      964      966      968      973      975
                          977      982      984      986
VERIFY_RUNS          PROC  [PSECT($CODE), UNBOUND]  ( IN PROGRAM CYCLOGRAPH )
                          851      884
WORD                 VAR   [PSECT($LOCAL)] ARRAY OF CHAR ( IN PROGRAM CYCLOGRAPH )
                           99      322 =    327 =    339      344      346      466 P    931 =    932      934 =
                          935 =    936 =    938      938      939 P
WORDCOUNT            VAR   [PSECT($LOCAL)] INTEGER ( IN PROGRAM CYCLOGRAPH )
                          122      561 =    565 =    565      565      566      570      576      578      579 =
                          579      579
WORDLIST             VAR   [PSECT($LOCAL)] WORDLISTS ( IN PROGRAM CYCLOGRAPH )
                          103      556 =    570 =    578 =    605      610      615      618      621      621
                          634      658      660      662      665      665      672      678      683      683
                          684      684      685      685      687      690      695      695      696      698
                          701      705      708      712      720      722      722      723      723      729
                          737      737      739      747      749      751      754      758      764      773
                          775      777      780      794      785      791      800      801      802      803
                          805      808      812      813      814      817      821      903      904      916
                          918      918      920      923      929      931      931      942      942      944
                          944      946      946      948      948      949      950      951      953      955
                          955      956      957      958      959      962      964      964      965      966
                          967      968      971      973      973      974      975      976      977      980
                          982      982      983      984      985      986      994      996     1001     1011
                         1016     1022
WORDS                TYPE  ARRAY OF CHAR ( IN PROGRAM CYCLOGRAPH )
                           41       42       44      193      226      262      230
WORD_LIST            TYPE  ARRAY OF WORDS ( IN PROGRAM CYCLOGRAPH )
                           42      133      134      135      136      137
WRITE                PROC  ( BUILTIN )
```

```
                         *      412
WRITELN                  PROC   ( BUILTIN )
                                *      411      414      431      432      433      589      603      615      616
                              619      628      629      639      640      645      646      652      653      678
                              679      690      691      701      703      708      710      715      716      726
                              727      732      733      742      743      754      756      761      762      767
                              768      780      782      788      789      794      795      808      810      817
                              819      324      825      834      835      840      841      846      847      861
                              868      371      874      905     1068
X                        VAR    [PSECT($LOCAL)] SINGLE ( IN PROGRAM CYCLOGRAPH )
                               92      386 =    388      388      389      395 =    397      398
X                        PARM   SINGLE ( IN PROCEDURE LINE )
                              160
X                        PARM   SINGLE ( IN PROCEDURE MOVE )
                              164
X                        PARM   SINGLE ( IN FUNCTION POWER )
                              186      189      189      189
X                        PARM   CHAR ( IN FUNCTION VALUE )
                              192      195
X                        PARM   WORDS ( IN FUNCTION VALUE_REAL )
                              198      211      212      212      218      219      219      220
X                        PARM   WORDS ( IN FUNCTION VALUE_INTEGER )
                              226      235      235
X                        PARM   DDLISTS ( IN FUNCTION COLUMN2_NOT_NULL )
                              241      245
X                        PARM   DDLISTS ( IN FUNCTION COLUMN3_NOT_NULL )
                              248      252
X                        PARM   DDLISTS ( IN FUNCTION COLUMN4_NOT_NULL )
                              255      259
X                        PARM   WORDS ( IN FUNCTION VALID_INTEGER )
                              262      271      272
X                        PARM   WORDS ( IN FUNCTION VALID_REAL )
                              280      290      292      293
XLEFT                    PARM   SINGLE ( IN PROCEDURE SET_WINDOW )
                              166
XRIGHT                   PARM   SINGLE ( IN PROCEDURE SET_WINDOW )
                              163
Y                        VAR    [PSECT($LOCAL)] SINGLE ( IN PROGRAM CYCLOGRAPH )
                               92      387 =    388      388      390      396 =    397      398
Y                        PARM   SINGLE ( IN PROCEDURE LINE )
                              160
Y                        PARM   SINGLE ( IN PROCEDURE MOVE )
                              164
YBOT                     PARM   SINGLE ( IN PROCEDURE SET_WINDOW )
                              168
YTOP                     PARM   SINGLE ( IN PROCEDURE SET_WINDOW )
                              163
```

KEY TO REFERENCE FLAGS:

*	predeclared
=	modified
A	address of
F	forward declared procedure or function
L	label defining reference
P	passed as a parameter, possibly modified or called
R	READ operation done
W	used in a WITH statement

PSECT SUMMARY

Name	Bytes	Attributes
$CODE	15907	NOVEC,NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)
$LOCAL	25918	NOVEC, WRT, RD,NOEXE,NOSHR, LCL, REL, CON, PIC,ALIGN(2)

COMMAND QUALIFIERS

```
  PASCAL/NOOPTIMIZE/LIS/CROSS CYCLOGRAPH

  /CHECK=(BOUNDS,NOCASE_SELECTORS,NOOVERFLOW,NOPOINTERS,NOSUBRANGE)
  /DEBUG=(NOSYMBOLS,TRACEBACK)
  /SHOW=(DICTIONARY,INCLUDE,NOINLINE,HEADER,SOURCE,STATISTICS)
  /NOOPTIMIZE
  /NOENVIRONMENT
  /LIST=BSU$USER_2:[OOTAALEXA.THESIS]CYCLOGRAPH.LIS;1
  /OBJECT=BSU$USER_2:[OOTAALEXA.THESIS]CYCLOGRAPH.OBJ;1
  /CROSS_REFERENCE /ERROR_LIMIT=30 /NOG_FLOATING /NOMACHINE_CODE /NOOLD_VERSION /NOSTANDARD /WARNINGS
```

COMPILER INTERNAL TIMING

Phase	Faults	CPU Time	Elapsed Time
Initialization	176	00:00.3	00:02.1
Source Analysis	529	00:03.4	00:10.6
Source Listing	50	00:02.4	00:10.9
Tree Construction	714	00:02.4	00:04.8
Flow Analysis	0	00:00.0	00:00.0
Value Propagation	0	00:00.0	00:00.0
Profit Analysis	0	00:00.0	00:00.0
Context Analysis	795	00:12.0	00:28.3
Name Packing	10	00:00.6	00:02.7
Code Selection	121	00:02.6	00:05.2
Final	114	00:03.7	00:06.5
TOTAL	2155	01:07.5	01:11.4

COMPILATION STATISTICS

```
  CPU Time:       00:07.5        (7333 Lines/Minute)
  Elapsed Time:   01:11.4
  Page Faults:    2155
  Compilation Complete
```

```
                          EEEEE  N    N  DDDD          000   FFFFF          FFFFF   III   L       EEEEE
                          E      N    N  D   D        0   0  F              F        I    L       E
TTTTTTTTTT   2222222222222 E      NN   N  D   D        0   0  F              F        I    L       E       22222222222222   TTTTTTTTTT
TTTTTTTTTT   2222222222222 EEEE   N N  N  D   D        0   0  FFFF           FFFF     I    L       EEEE    22222222222222   TTTTTTTTTT
TTTTTTTTTT   2222222222222 E      N  NN  D   D        0   0  F              F        I    L       E       22222222222222   TTTTTTTTTT
                          E      N   N  D   D        0   0  F              F        I    L       E
                          EEEEE  N    N  DDDD          000   F              F       III   LLLLL   EEEEE

                       000    000   TTTTT   AAA    AAA   L      EEEEE  X   X   AAA   N   N  DDDD   EEEEE
                      0   0  0   0    T    A   A  A   A  L      E      X   X  A   A  N   N  D   D  E
                      0  00  0  00    T    A   A  A   A  L      E       X X   A   A  NN  N  D   D  E
                      0 0 0  0 0 0    T    A   A  A   A  L      EEEE     X    A   A  N N N  D   D  EEEE
                      00  0  00  0    T    AAAAA  AAAAA  L      E       X X   AAAAA  N  NN  D   D  E
                      0   0  0   0    T    A   A  A   A  L      E      X   X  A   A  N   N  D   D  E
                       000    000     T    A   A  A   A  LLLLL  EEEEE  X   X  A   A  N   N  DDDD   EEEEE

                 CCCC  Y   Y  CCCC  L       000    GGGG  RRRR    AAA   PPPP   H   H         L       III    SSSS    ;;     1
                C      Y   Y C      L      0   0  G      R   R  A   A  P   P  H   H         L        I    S        ;;    11
                C       Y Y  C      L      0   0  G      R   R  A   A  P   P  H   H         L        I    S             1
                C        Y   C      L      0   0  G      RRRR   A   A  PPPP   HHHHH         L        I     SSS     ;;     1
                C        Y   C      L      0   0  G GGG  R R    AAAAA  P      H   H         L        I        S    ;;     1
                C        Y   C      L      0   0  G   G  R  R   A   A  P      H   H    ..   L        I        S    ;      1
                 CCCC    Y    CCCC  LLLLL   000    GGG   R   R  A   A  P      H   H    ..   LLLLL   III.  SSSS    ;      111
```

Qualifiers: /FEED /FLAG /FORM=DEFAULT /LENGTH=66 /LIBRARY=SYS$LIBRARY:SYSDEVCTL.TLB /MARGIN=(BOTTOM=0) /TRAILER /WIDTH=132 /WRAP

File _1DUA2:[USER.OOTAALEXA.THESIS]CYCLOGRAPH.LIS;1 (16743,41,0), last revised on 20-MAY-1986 21:32, is a 179 block sequential file owned by UIC [OOTAALEXANDE]. The records are variable length with implied (CR) carriage control. The longest record is 132 bytes.

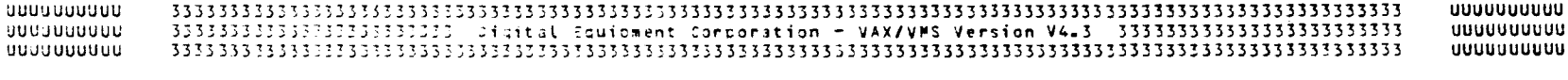
Job CYCLOGRAPH (1265) queued to LPA0 on 20-MAY-1986 21:47 by user OOTAALEXANDE, UIC [OOTAALEXANDE], under account C at priority 4, completed on printer _PISCES$LPA0: on 20-MAY-1986 21:47 from queue LPA0.

```
TTTTTTTTTT    22222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222    TTTTTTTTTT
TTTTTTTTTT    22222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222    TTTTTTTTTT
TTTTTTTTTT    22222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222    TTTTTTTTTT
```

```
        1         2         3         4         5         6         7         8         9         0         1         2         3
12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012
```

File _1DUA2:[USER.OOTAALEXA.THESIS]HOME.LIS;1 (51264,*,0), last revised on 20-MAY-1986 21:49, is a 8 block sequential file owned by UIC [OOTAALEXANDE]. The records are variable length with implied (CR) carriage control. The longest record is 132 bytes.

Job HOME (1269) queued to LPA0 on 20-MAY-1986 21:50 by user OOTAALEXANDE, UIC [OOTAALEXANDE], under account C at priority 4, started on printer _PISCES$LPA0: on 20-MAY-1986 21:50 from queue LPA0.



```
     1        ****************************************************************
     2        *                                                              *
     3        *   HOME is a module, written in COBOL with language extensions,     *
     4        *  that clears the command line area and produces the prompt       *
     5        *  " i> " which tells the user that he or she is in the interactive  *
     6        *  mode.                                                       *
     7        *                                                              *
     8        ****************************************************************
     9
    10         IDENTIFICATION DIVISION.
    11          PROGRAM-ID.      HOME.
    12          AUTHOR.          TODD ALEXANDER.
    13
    14         ENVIRONMENT DIVISION.
    15          CONFIGURATION SECTION.
    16            SOURCE-COMPUTER. VAX-11.
    17            OBJECT-COMPUTER. VAX-11.
    18
    19         DATA DIVISION.
    20
    21          WORKING-STORAGE SECTION.
    22
    23          LINKAGE SECTION.
    24
    25        ****************************************************************
    26        *                       PROCEDURE DIVISION                     *
    27        ****************************************************************
    28
    29         PROCEDURE DIVISION.
    30         0000-MAIN.
    31              DISPLAY '
    32                      LINE 1 COLUMN 1.
    33              DISPLAY 'i>' LINE 1 COLUMN 1.
    34              DISPLAY '' LINE 1 COLUMN 3.
    35              EXIT PROGRAM.
    36
```

```
0000-MAIN                       30#
HOME                            11#
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	60	PIC CON REL LCL SHR EXE RD NOWRT Align(2)
2	$PDATA	192	PIC CON REL LCL SHR NOEXE RD NOWRT Align(2)
3	COB$NAMES_____2	24	PIC CON REL LCL SHR NOEXE RD NOWRT Align(2)
4	COB$NAMES_____4	5	PIC CON REL LCL SHR NOEXE RD NOWRT Align(2)

DIAGNOSTICS

Informational: 7 (suppressed by command qualifier)

COMMAND QUALIFIERS

```
COBOL /LIS/CROSS HOME,HOME2

/NOCOPY_LIST  /NOMACHINE_CODE  /CROSS_REFERENCE=ALPHABETICAL
/NOANSI_FORMAT  /NOSEQUENCE_CHECK  /NOMAP
/NOTRUNCATE  /NOAUDIT  /NOCONDITIONALS
/CHECK=(NOPERFORM,NOBOUNDS)  /DEBUG=(NOSYMBOLS,TRACEBACK)
/WARNINGS=(NOSTANDARD,OTHER,NOINFORMATION)
/STANDARD=(NOSYNTAX,NOPDP11)  /NOFIPS
```

STATISTICS

Run Time:	0.73 seconds
Elapsed Time:	2.07 seconds
Page Faults:	333
Dynamic Memory:	477 pages

```
                                        EEEEE  N   N  DDDD        000   FFFFF        FFFFF  III  L      EEEEE
                                        E      N   N  D   D      O   O  F            F       I   L      E
UUUUUUUUUU   3333333333333              E      NN  N  D   D      O   O  F            F       I   L      E       3333333333333   UUUUUUUUUU
UUUUUUUUUU   3333333333333              EEEE   N N N  D   D      O   O  FFFF         FFFF    I   L      EEEE    3333333333333   UUUUUUUUUU
UUUUUUUUUU   3333333333333              E      N  NN  D   D      O   O  F            F       I   L      E       3333333333333   UUUUUUUUUU
                                        E      N   N  D   D      O   O  F            F       I   L      E
                                        EEEEE  N   N  DDDD        000   F            F      III  LLLLL  EEEEE

                                      000    000   TTTTT   AAA    AAA   L      EEEEE  X   X   AAA   N   N  DDDD   EEEEE
                                     0   0  0   0    T    A   A  A   A  L      E      X   X  A   A  N   N  D   D  E
                                     0  00  0  00    T    A   A  A   A  L      E       X X   A   A  NN  N  D   D  E
                                     0 0 0  0 0 0    T    A   A  A   A  L      EEEE     X    A   A  N N N  D   D  EEEE
                                     00  0  00  0    T    AAAAA  AAAAA  L      E       X X   AAAAA  N  NN  D   D  E
                                     0   0  0   0    T    A   A  A   A  L      E      X   X  A   A  N   N  D   D  E
                                      000    000     T    A   A  A   A  LLLLL  EEEEE  X   X  A   A  N   N  DDDD   EEEEE

                                        H   H   000   M   M  EEEEE        L      III   SSSS   ;;    1
                                        H   H  O   O  MM MM  E            L       I   S       ;;   11
                                        H   H  O   O  M M M  E            L       I   S             1
                                        HHHHH  O   O  M   M  EEEE         L       I    SSS    ;;    1
                                        H   H  O   O  M   M  E            L       I       S   ;;    1
                                        H   H  O   O  M   M  E       ..   L       I       S    ;    1
                                        H   H   000   M   M  EEEEE   ..   LLLLL  III  SSSS     ;   111
```

Qualifiers: /FEED /FLAG /FORM=DEFAULT /LENGTH=66 /LIBRARY=SYS$LIBRARY:SYSDEVCTL.TLB /MARGIN=(BOTTOM=6) /TRAILER /WIDTH=132 /WRAP

File _1DUA2:[USER.OOTAALEXA.THESIS]HOME.LIS;1 (51264,8,0), last revised on 20-MAY-1986 21:49, is a 8 block sequential file owned by UIC [OOTAALEXANDE]. The records are variable length with implied (CR) carriage control. The longest record is 132 bytes.

Job HOME (1269) queued to LPA0 on 20-MAY-1986 21:50 by user OOTAALEXANDE, UIC [OOTAALEXANDE], under account C at priority 4, completed on printer _PISCES$LPA0: on 20-MAY-1986 21:50 from queue LPA0.

```
UUUUUUUUUU   3333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333   UUUUUUUUUU
UUUUUUUUUU   33333333333333333333333333  Digital Equipment Corporation - VAX/VMS Version V4.3  3333333333333333333333333   UUUUUUUUUU
UUUUUUUUUU   3333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333   UUUUUUUUUU

         1         2         3         4         5         6         7         8         9         0         1         2         3
123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012
```

```
AAAAAAAAAA    5555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555    AAAAAAAAAA
AAAAAAAAAA    555555555555555555555555555555  Ball State University VAXcluster VMS 4.3  555555555555555555555555555555    AAAAAAAAAA
AAAAAAAAAA    5555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555    AAAAAAAAAA

 OOO     OOO    TTTTT   AAA    AAA   L      EEEEE  X   X   AAA   N   N  DDDD   EEEEE
O   O   O   O     T    A   A  A   A  L      E      X   X  A   A  N   N  D   D  E
O  OO   O  OO     T    A   A  A   A  L      E       X X   A   A  NN  N  D   D  E
O O O   O O O     T    A   A  A   A  L      EEEE     X    A   A  N N N  D   D  EEEE
OO  O   OO  O     T    AAAAA  AAAAA  L      E       X X   AAAAA  N  NN  D   D  E
O   O   O   O     T    A   A  A   A  L      E      X   X  A   A  N   N  D   D  E
 OOO     OOO      T    A   A  A   A  LLLLL  EEEEE  X   X  A   A  N   N  DDDD   EEEEE

HH        HH     OOOOOO     MM        MM  EEEEEEEEEE    222222
HH        HH     OOOOOO     MM        MM  EEEEEEEEEE    222222
HH        HH   OO      OO   MMMM    MMMM  EE          22      22
HH        HH   OO      OO   MMMM    MMMM  EE          22      22
HH        HH   OO      OO   MM  MM  MM    EE                  22
HH        HH   OO      OO   MM  MM  MM    EE                  22
HHHHHHHHHHHH   OO      OO   MM        MM  EEEEEEEE          22
HHHHHHHHHHHH   OO      OO   MM        MM  EEEEEEEE          22
HH        HH   OO      OO   MM        MM  EE              22
HH        HH   OO      OO   MM        MM  EE              22
HH        HH   OO      OO   MM        MM  EE            22
HH        HH   OO      OO   MM        MM  EE            22
HH        HH     OOOOOO     MM        MM  EEEEEEEEEE  2222222222
HH        HH     OOOOOO     MM        MM  EEEEEEEEEE  2222222222

         LL              IIIIII      SSSSSSSS      ;;;;       222222
         LL              IIIIII      SSSSSSSS      ;;;;       222222
         LL                II      SS              ;;;;     22      22
         LL                II      SS              ;;;;     22      22
         LL                II      SS                               22
         LL                II      SS                               22
         LL                II        SSSSSS        ;;;;           22
         LL                II        SSSSSS        ;;;;           22
         LL                II              SS      ;;;;         22
         LL                II              SS      ;;;;         22
....     LL                II              SS        ;;       22
....     LL                II              SS        ;;       22
....     LLLLLLLLLL      IIIIII    SSSSSSSS          ;;     2222222222
....     LLLLLLLLLL      IIIIII    SSSSSSSS          ;;     2222222222
```

File _B1CDUA2:[USER.OOTAAL[illegible]]HOME2.LIS;2 (21310,102,0), last revised on 10-MAY-1986 15:30, is a 7 block sequential file owned by UIC [OOTAALEXANDE]. The records are variable length with implied (CR) carriage control. The longest record is 132 bytes.

Job HOME2 (1583) queued to LPA[illegible] on 17-MAY-1986 11:[illegible] by user OOTAALEXANDE, UIC [OOTAALEXANDE], under account C at priority 4, started on printer _[illegible]LPA[illegible]: on 17-MAY-1986 11:[illegible] from queue LPA0.

```
AAAAAAAAAA    5555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555    AAAAAAAAAA
AAAAAAAAAA    [illegible] Digital Equipment Corporation - VAX/VMS Version V4.3  555555555555555555555555    AAAAAAAAAA
AAAAAAAAAA    5555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555    AAAAAAAAAA
```

```
     1        *****************************************************************
     2        *                                                               *
     3        *   HOME2 is a module, written in COBOL with language extensions,    *
     4        *  that clears the command line area and produces the prompt        *
     5        *  " p> " which tells the user that he or she is in the programming  *
     6        *  mode.                                                        *
     7        *                                                               *
     8        *****************************************************************
     9
    10
    11         IDENTIFICATION DIVISION.
    12          PROGRAM-ID.      HOME2.
    13          AUTHOR.          TODD ALEXANDER.
    14
    15         ENVIRONMENT DIVISION.
    16          CONFIGURATION SECTION.
    17            SOURCE-COMPUTER. VAX-11.
    18            OBJECT-COMPUTER. VAX-11.
    19
    20         DATA DIVISION.
    21
    22          WORKING-STORAGE SECTION.
    23
    24          LINKAGE SECTION.
    25
    26        *****************************************************************
    27        *                       PROCEDURE DIVISION                      *
    28        *****************************************************************
    29
    30         PROCEDURE DIVISION.
    31         0000-MAIN.
    32              DISPLAY '
    33                      LINE 1 COLUMN 1.
    34              DISPLAY ' p>' LINE 1 COLUMN 1.
    35              DISPLAY ' ' LINE 1 COLUMN 1.
    36              EXIT PROGRAM.
    37
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	60	PIC CON REL LCL SHR EXE RD NOWRT Align(2)
2	$PDATA	192	PIC CON REL LCL SHR NOEXE RD NOWRT Align(2)
3	COB$NAMES_____2	24	PIC CON REL LCL SHR NOEXE RD NOWRT Align(2)
4	COB$NAMES_____4	0	PIC CON REL LCL SHR NOEXE RD NOWRT Align(2)

DIAGNOSTICS

Informational: 7 (suppressed by command qualifier)

COMMAND QUALIFIERS

COBOL /LIST HOME/HOME2

/NOCOPY_LIST /NOMACHINE_CODE /NOCROSS_REFERENCE
/NOANSI_FORMAT /NOSEQUENCE_CHECK /NOMAP
/NOTRUNCATE /NOAUDIT /NOCONDITIONALS
/CHECK=(NOPERFORM,NOBOUNDS) /DEBUG=(NOSYMBOLS,TRACEBACK)
/WARNINGS=(NOSTANDARD,OTHER,NOINFORMATION)
/STANDARD=(NOSYNTAX,NOPS211) /NOFIPS

STATISTICS

Run Time: 3.20 seconds
Elapsed Time: 1.11 seconds
Page Faults: 81
Dynamic Memory: [illegible] bytes

```
                                       EEEEE  N   N  DDDD          OOO   FFFFF        FFFFF  III  L      EEEEE
                                       E      N   N  D   D        O   O  F            F       I   L      E
AAAAAAAAAA   5555555555555   E      NN  N  D   D        O   O  F            F       I   L      E      5555555555555   AAAAAAAAAA
AAAAAAAAAA   5555555555555   EEEE   N N N  D   D        O   O  FFFF         FFFF    I   L      EEEE   5555555555555   AAAAAAAAAA
AAAAAAAAAA   5555555555555   E      N  NN  D   D        O   O  F            F       I   L      E      5555555555555   AAAAAAAAAA
                                       E      N   N  D   D        O   O  F            F       I   L      E
                                       EEEEE  N   N  DDDD          OOO   F            F      III  LLLLL  EEEEE

                                    OOO    OOO   TTTTT   AAA    AAA   L      EEEEE  X   X   AAA   N   N  DDDD   EEEEE
                                   O   O  O   O    T    A   A  A   A  L      E      X   X  A   A  N   N  D   D  E
                                   O  OO  O  OO    T    A   A  A   A  L      E       X X   A   A  NN  N  D   D  E
                                   O O O  O O O    T    A   A  A   A  L      EEEE     X    A   A  N N N  D   D  EEEE
                                   OO  O  OO  O    T    AAAAA  AAAAA  L      E       X X   AAAAA  N  NN  D   D  E
                                   O   O  O   O    T    A   A  A   A  L      E      X   X  A   A  N   N  D   D  E
                                    OOO    OOO     T    A   A  A   A  LLLLL  EEEEE  X   X  A   A  N   N  DDDD   EEEEE

                                     H   H   OOO   M   M  EEEEE   222          L      III   SSSS   ;;   222
                                     H   H  O   O  MM MM  E      2   2         L       I   S       ;;  2   2
                                     H   H  O   O  M M M  E          2         L       I   S               2
                                     HHHHH  O   O  M   M  EEEE      2          L       I    SSS    ;;     2
                                     H   H  O   O  M   M  E        2           L       I       S   ;;    2
                                     H   H  O   O  M   M  E       2       ..   L       I       S    ;   2
                                     H   H   OOO   M   M  EEEEE  22222    ..   LLLLL  III  SSSS     ;   22222
```

Qualifiers: /FEED /FLAG /FORM=DEFAULT /LENGTH=66 /LIBRARY=SYS$LIBRARY:SYSDEVCTL.TLB /MARGIN=(BOTTOM=6) /TRAILER /WIDTH=132 /WRAP

File _1DUA2:[USER.OOTAALEXANDE.THESIS]HOME2.LIS;2 (21310,102,0), last revised on 10-MAY-1986 15:30, is a 7 block sequential file owned by UIC [OOTAALEXANDE]. The records are variable length with implied (CR) carriage control. The longest record is 132 bytes.

Job HOME2 (1063) queued to LPA0 on 17-MAY-1986 11:22 by user OOTAALEXANDE, UIC [OOTAALEXANDE], under account C at priority 4, completed on printer _1LPA0: on 17-MAY-1986 11:23 from queue LPA0.

```
AAAAAAAAAA   55555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555   AAAAAAAAAA
AAAAAAAAAA   5555555555555555555555555  Digital Equipment Corporation - VAX/VMS Version V4.3  5555555555555555555555555   AAAAAAAAAA
AAAAAAAAAA   55555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555555   AAAAAAAAAA

         1         2         3         4         5         6         7         8         9         0         1         2         3
123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012
```