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
Appendix B: Original Instruction Booklet
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. 144 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. 150 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. 144 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.

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.

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 244 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.
Always place Ring with mark 1 at top. Line up the Hole you are using with mark 1 unless otherwise instructed.

Ring No. | Wheel No. | Hole No. and Color of Pen
---|---|---
144 | 105 | 144
96 | 105 | 96

1. Place Ring with mark 1 at top. Line up the Hole you are using with mark 1.
2. Starting at mark 1, draw 2, 3, 4, 5, 6, 7, 8, 9, 10, moving one tooth right every hole.
3. Draw 9, moving one tooth right.
4. Turn Wheel over and draw 9, lined up with mark 2.
5. Draw 10, lined up with mark 2.
6. Draw 11, lined up with mark 2.
7. Draw 12, lined up with mark 2.
8. Draw 13, lined up with mark 2.
10. Draw 15, lined up with mark 2.
11. Draw 16, lined up with mark 2.
12. Draw 17, lined up with mark 2.
13. Draw 18, lined up with mark 2.
15. Draw 20, lined up with mark 2.
16. Draw 21, lined up with mark 2.
17. Draw 22, lined up with mark 2.
18. Draw 23, lined up with mark 2.
19. Draw 24, lined up with mark 2.
20. Draw 25, lined up with mark 2.
22. Draw 27, lined up with mark 2.
23. Draw 28, lined up with mark 2.
24. Draw 29, lined up with mark 2.
25. Draw 30, lined up with mark 2.
27. Draw 32, lined up with mark 2.
29. Draw 34, lined up with mark 2.
31. Draw 36, lined up with mark 2.
32. Draw 37, lined up with mark 2.
33. Draw 38, lined up with mark 2.
34. Draw 39, lined up with mark 2.
35. Draw 40, lined up with mark 2.
36. Draw 41, lined up with mark 2.
37. Draw 42, lined up with mark 2.
38. Draw 43, lined up with mark 2.
39. Draw 44, lined up with mark 2.
40. Draw 45, lined up with mark 2.
41. Draw 46, lined up with mark 2.
42. Draw 47, lined up with mark 2.
43. Draw 48, lined up with mark 2.
44. Draw 49, lined up with mark 2.
45. Draw 50, lined up with mark 2.
46. Draw 51, lined up with mark 2.
47. Draw 52, lined up with mark 2.
49. Draw 54, lined up with mark 2.
50. Draw 55, lined up with mark 2.
51. Draw 56, lined up with mark 2.
52. Draw 57, lined up with mark 2.
53. Draw 58, lined up with mark 2.
54. Draw 59, lined up with mark 2.
55. Draw 60, lined up with mark 2.
56. Draw 61, lined up with mark 2.
57. Draw 62, lined up with mark 2.
58. Draw 63, lined up with mark 2.
59. Draw 64, lined up with mark 2.
60. Draw 65, lined up with mark 2.
61. Draw 66, lined up with mark 2.
63. Draw 68, lined up with mark 2.
64. Draw 69, lined up with mark 2.
65. Draw 70, lined up with mark 2.
66. Draw 71, lined up with mark 2.
67. Draw 72, lined up with mark 2.
68. Draw 73, lined up with mark 2.
69. Draw 74, lined up with mark 2.
70. Draw 75, lined up with mark 2.
71. Draw 76, lined up with mark 2.
72. Draw 77, lined up with mark 2.
73. Draw 78, lined up with mark 2.
74. Draw 79, lined up with mark 2.
75. Draw 80, lined up with mark 2.
76. Draw 81, lined up with mark 2.
77. Draw 82, lined up with mark 2.
78. Draw 83, lined up with mark 2.
79. Draw 84, lined up with mark 2.
80. Draw 85, lined up with mark 2.
81. Draw 86, lined up with mark 2.
82. Draw 87, lined up with mark 2.
83. Draw 88, lined up with mark 2.
84. Draw 89, lined up with mark 2.
85. Draw 90, lined up with mark 2.
86. Draw 91, lined up with mark 2.
87. Draw 92, lined up with mark 2.
88. Draw 93, lined up with mark 2.
89. Draw 94, lined up with mark 2.
90. Draw 95, lined up with mark 2.
91. Draw 96, lined up with mark 2.
92. Draw 97, lined up with mark 2.
93. Draw 98, lined up with mark 2.
94. Draw 99, lined up with mark 2.
95. Draw 100, lined up with mark 2.
96. Draw 101, lined up with mark 2.
97. Draw 102, lined up with mark 2.
98. Draw 103, lined up with mark 2.
100. Draw 105, lined up with mark 2.
102. Draw 107, lined up with mark 2.
103. Draw 108, lined up with mark 2.
104. Draw 109, lined up with mark 2.
105. Draw 110, lined up with mark 2.
106. Draw 111, lined up with mark 2.
107. Draw 112, lined up with mark 2.
108. Draw 113, lined up with mark 2.
110. Draw 115, lined up with mark 2.
111. Draw 116, lined up with mark 2.
112. Draw 117, lined up with mark 2.
113. Draw 118, lined up with mark 2.
114. Draw 119, lined up with mark 2.
115. Draw 120, lined up with mark 2.
116. Draw 121, lined up with mark 2.
117. Draw 122, lined up with mark 2.
118. Draw 123, lined up with mark 2.
119. Draw 124, lined up with mark 2.
120. Draw 125, lined up with mark 2.
121. Draw 126, lined up with mark 2.
122. Draw 127, lined up with mark 2.
123. Draw 128, lined up with mark 2.
124. Draw 129, lined up with mark 2.
125. Draw 130, lined up with mark 2.
126. Draw 131, lined up with mark 2.
127. Draw 132, lined up with mark 2.
128. Draw 133, lined up with mark 2.
129. Draw 134, lined up with mark 2.
130. Draw 135, lined up with mark 2.
131. Draw 136, lined up with mark 2.
132. Draw 137, lined up with mark 2.
133. Draw 138, lined up with mark 2.
134. Draw 139, lined up with mark 2.
135. Draw 140, lined up with mark 2.
136. Draw 141, lined up with mark 2.
137. Draw 142, lined up with mark 2.
138. Draw 143, lined up with mark 2.
139. Draw 144, lined up with mark 2.
1 thru 26 moving two teeth right every hole.

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.
Move six teeth right and draw 1-3-5-7.

Line up so that Hole is directly in the center between the blue and green patterns.

1-3-5-7 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 every hole.

1-3-5-7 moving one tooth right every hole.

1-3-5-7 moving one tooth right and left every hole.

24 5 30 11 23, starting at marks 1 and 2

72 29 64 35, starting at marks 1 and 3

1-3-5-7 moving one tooth right and left every hole.

150 105 63

144 96

150 105 63

144 96

144 96

144 96

1-3-5-7 moving one tooth right and left every hole.

1-3-5-7 moving one tooth right and left every hole.

1-3-5-7 moving one tooth right and left every hole.

150 105 63

24 5 30 11 23, starting at marks 1 and 2

72 29 64 35, starting at marks 1 and 3

1-3-5-7 moving one tooth right and left every hole.

150 105 63

144 96

144 96
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 pen.

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.

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

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.
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.

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

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

Outside edge of Ring
1 through 28, 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 (64) on the green pattern you have just drawn. Pin it to the board. Use (53) 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.


If your dealer is temporarily out of kits, we will, as a service send it to you direct. Send 90c (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. SHK, Kenner Products Co., 912 Sycamore Street, Cincinnati, Ohio 45202.
Source Listing
file _STUDY3.CUSER,BTTALEXA,THECIXCYCLOGRAPH.LIST (1976-1986) last revised on 20-MAY-1986 21:52 is a 179 block sequential file owned by UIC (BTTALEXA). The records are variable length with implied (CR) carriage control. The longest record is 132 bytes.

Job CYCLOGRAPH (112S) queued to LPA0 on 10-APR-1976 11:17 by user BTTALEXA, UIC (BTTALEXA) under account C at priority 4, started on printer _VSSCES.LPA0 on 10-APR-1976 11:17 from queue LPA0.

.....
**Program:** Cyclograph

**Author:** Todd A. Alexander

**Purpose:** Thesis / Creative Project required to graduate from the Honors College -- this is part of my submission under the enrollment of 10499.

**Date of Last Revision:** May 17, 1986

**Overview:** This program, along with its subordinates, attempts to emulate the Spirograph(TM) drawing toy developed by Kenner, Inc. Interactive as well as programming commands are used to draw designs at the terminal.

**Modules Called:** Home, Home2

**Modules Used In RAL Library:**
- Clear_Screen
- Init_graphics
- Line
- Move
- Set_window
- Set_color

---

Source Listing

20-May-1986 21:31:02

VAX Pascal V3.2-57

Page 1

00001 C 0 0 (**--------------------------------------------------------------------------------------------------------------------------)**
00002 C 0 0 (* Program: Cyclograph *)
00003 C 0 0 (* Author: Todd A. Alexander *)
00004 C 0 0 (* Purpose: Thesis / Creative Project required to graduate from the Honors College -- this is part of my submission under the enrollment of 10499. *)
00005 C 0 0 (* Date of Last Revision: May 17, 1986 *)
00006 C 0 0 (* Program Overview: This program, along with its subordinates, attempts to emulate the Spirograph(TM) drawing toy developed by Kenner, Inc. Interactive as well as programming commands are used to draw designs at the terminal. *)
00007 C 0 0 (* Modules Called: Home, Home2 *)
00008 C 0 0 (* Modules Used In RAL Library:**
00009 C 0 0 (* Clear_Screen *)
00010 C 0 0 (* Init_graphics *)
00011 C 0 0 (* Line *)
00012 C 0 0 (* Move *)
00013 C 0 0 (* Set_window *)
00014 C 0 0 (* Set_color *)
00015 C 0 0 (**--------------------------------------------------------------------------------------------------------------------------)**
00016 C 0 0 (**--------------------------------------------------------------------------------------------------------------------------)**
00017 C 0 0 (**--------------------------------------------------------------------------------------------------------------------------)**
00018 C 0 0 (**--------------------------------------------------------------------------------------------------------------------------)**
00019 C 0 0 (**--------------------------------------------------------------------------------------------------------------------------)**
00020 C 0 0 (**--------------------------------------------------------------------------------------------------------------------------)**
00021 C 0 0 (**--------------------------------------------------------------------------------------------------------------------------)**
00022 C 0 0 (**--------------------------------------------------------------------------------------------------------------------------)**
00023 C 0 0 (**--------------------------------------------------------------------------------------------------------------------------)**
00024 C 0 0 (**--------------------------------------------------------------------------------------------------------------------------)**
00025 C 0 0 (**--------------------------------------------------------------------------------------------------------------------------)**
00026 C 0 0 (**--------------------------------------------------------------------------------------------------------------------------)**
00027 C 0 0 (**--------------------------------------------------------------------------------------------------------------------------)**
00028 C 0 0 (**--------------------------------------------------------------------------------------------------------------------------)**
00029 C 0 0 (**--------------------------------------------------------------------------------------------------------------------------)**
00030 C 0 0 (**--------------------------------------------------------------------------------------------------------------------------)**
00031 C 0 0 (**--------------------------------------------------------------------------------------------------------------------------)**
00032 C 0 0 (**--------------------------------------------------------------------------------------------------------------------------)**
00033 C 0 0 (**--------------------------------------------------------------------------------------------------------------------------)**
```pascal
PROGRAM Cyclograph(Input,Output,Procfile);

{ Mandatory variables }
Type

SetOfCharacters = Set of Char;

String = Packed Array[1..80] of Char;

Lines = Packed Array[1..80] of Char;

Words = Packed Array[1..10] of Char;

Word_List = Packed Array[1..19] of Words;

Lists = Packed Array[1..200] of Lines;

DoLists = Packed Array[1..200,1..4] of Words;

Var

Cycles : Real;  (* Number calculated to specify number of revolutions of wheel in ring *)
Cosr : Real;  (* Intermediate cosine value used to rotate *)
Cossim : Real;  (* Intermediate cosine value used to rotate *)
Cos : Real;  (* Intermediate cosine value used to rotate *)
LenQt : Real;  (* Length of segment from current point to origin *)
Rotation : Real;  (* Rotation in radians of ring, in steps of "teeth" *)
Sinr : Real;  (* Intermediate sine value used to rotate *)
Simm : Real;  (* Intermediate sine value used to rotate *)
Sin : Real;  (* Intermediate sine value used to rotate *)
Strt : Real;  (* For intermittent graphing - where wheel begins *)
Theta : Real;  (* Angle from x axis to center of wheel *)

Comments : SetOfCharacters;  (* Set of valid one letter commands *)

Heta : Single;  (* Angle of revolution of wheel relative to theta *)
Num : Single;  (* Value of number found within Cline *)
Radius : Single;  (* Combined radius of wheel *)
Ratio : Single;  (* Ring divided by wheel radius *)
a : Single;  (* Radius of ring *)
b : Single;  (* Pasis of wheel *)
}

******************************************************************************

{ Purpose/Explanation }

******************************************************************************

{ Variables used for the immediate mode and graphing }

******************************************************************************

{- LINE-IDC-PL-SL-}
```
<table>
<thead>
<tr>
<th>CYCLOGRAPH 01</th>
<th>Source Listing</th>
<th>20-May-1986 21:31:02</th>
<th>VAX Pascal V3.2-57</th>
</tr>
</thead>
<tbody>
<tr>
<td>00090</td>
<td>0 0 h</td>
<td>Single;</td>
<td>(Hole number)</td>
</tr>
<tr>
<td>00091</td>
<td>0 0 j</td>
<td>Single;</td>
<td>(Distance from center of hole to pen contact)</td>
</tr>
<tr>
<td>00092</td>
<td>0 0 x/y</td>
<td>Single;</td>
<td>(x and y vectors, [1 to axis, plotted from origin)</td>
</tr>
</tbody>
</table>
### Variable Name | Type | Purpose/Explanation
---|---|---
**Counter_stack** | Array[1..20] of Integer | (Stacks number of repetitions)
**PC_stack** | Array[1..20] of Integer | (Stacks line numbers for DO and RUN commands)
**Procline** | Array[1..50] of Integer | (List of line numbers for corresponding procedures in Proclist)
**Error_free** | Boolean | (Flag for errors in program)
**Found** | Boolean | (Flag for searches through lists)
**wordList** | DDLists | (Two dimensional array holding command procedures separated into words)
**A2** | Integer | (Multi-purpose loop index variable)
**Z2** | Integer | (Multi-purpose loop index variable)
**Dos** | Integer | (Number of DC statements found in procedure)
**Errornum** | Integer | (Number of errors found in Proclist)
**K** | Integer | (Multi-purpose loop index variable)
**Linecount** | Integer | (Number of command lines in Proclist)
**Letcount** | Integer | (Number of letters in command words)
**PC** | Integer | (Program line counter - current line number being executed)
**Procs** | Integer | (Total number of defined procedure in Proclist)
**Pointer** | Integer | (Level of nesting of commands or DO loops)
**Runs** | Integer | (Number of RUN statements encountered in Proclist)
**Starts** | Integer | (Number of START statements found in a procedure)
**Wordcount** | Integer | (Current number of words found in command)
**CommLine** | Lines | (Line of text from Proclist)
**CommList** | Lists | (Entire list of lines from Proclist)
**Proclist** | wordList | (List of procedure names)
**Funlist** | wordList | (List of procedures called)
**ValList** | wordList | (Valid words for column 1)
**ValList** | wordList | (Valid words for column 2)
**ValList** | wordList | (Valid words for column 3)
'```pascal
{*************************************************************}
{external Procedures for RAL REGIS Graphics and Cobol External Procedures}
{*****************************************************************************}

Procedure Clear_screen;
{Clears screen of graphics and text, to black}

Procedure Init_graphics;
{Sets default colors, positions for graphics}

Procedure Line(X:Single;Y:Single);
{Draws line from current graphic position to x,y}

Procedure Move(X:Single;Y:Single);
{Moves graphic cursor to x,y}

Procedure Set_window(Left,Yot,Xright,Top:Single);
{Sets x and y bounds}

Procedure Set_color(%STDESC String;Packed array[1..$integer] of Char);
{Changes color of plotting}

Procedure Set_text(%STDESC String;Packed array[1..$integer] of Char;
{Places text on screen}

Procedure Set_numbers(Numbers:Integer);

Procedure Get_text(Packed array[1..$integer] of Char);
{Places text on screen}

Procedure Get_numbers(Integer);

Procedure Home;

Procedure Home2;

Procedure Home3;

Procedure Home4;
```

\begin{verbatim}
1 20-May-1986 21:31:02 VAX Pascal V3.2-57
Source Listing 20-May-1986 19:59:49
CUTALEXAHESISCYCLOGRAPH.PAS:34 (5)

--- LINE-LOC-P3-SL ---
00132  C 0 0 {***********************************************
00133  C 0 0 (* Internally Defined Functions *)
00134  C 0 0 {***********************************************
00135  0015  0 0
00136  0016  0 0 Function Power(x:real; n:integer):real;
00137  0017  1 0 (Returns x raised to the n)
00138  0018  1 1 Begin
00139  0019  1 1 If n=0 then Power:=1 else Power:=x * Power(x,n-1)
00140  001A  0 0 End;
00141  001B  0 0
00142  001C  0 0 Function Value(X:char):integer;
00143  001D  1 0 (Returns value of a number character)
00144  001E  1 1 Begin
00145  001F  1 1 Value := Ord(X) - Ord('0');
00146  0020  0 0 End;
00147  0021  0 0
00148  0022  0 0 Function Value_Real(x:words):real;
00149  0023  1 0 (Returns value of real number which was in character format)
00150  0024  1 0 Var
00151  0025  1 0 Num: integer;
00152  0026  1 0 Num2: real;
00153  0027  1 0 Point_found, found_again: boolean;
00154  0028  1 1 Begin
00155  0029  1 1 num := 0;
00156  002A  1 1 num2 := 0;
00157  002B  1 1 m := 1;
00158  002C  1 1 Point_found := false;
00159  002D  1 1 While (m < 11) and (not(Point_found)) Do
00160  002E  1 2 Begin
00161  002F  1 2 If x[m] <> ' ' then Point_found := true
00162  0030  1 2 Else if x[m] <> ' ' then Num := Num*10 + VALUE(X[M]);
00163  0031  1 2 m := m+1;
00164  0032  1 1 End;
00165  0033  1 1 Num := Num + Num2;
00166  0034  0 0 End;
00167  0035  0 0
00168  0036  0 0 Function Value_Integer(x:words):integer;
00169  0037  1 0 (Returns value of an integer which was in character format)
00170  0038  1 0 Var
00171  0039  1 0 Num: integer;
00172  003A  1 1 Begin
00173  003B  1 1 num := 0;
00174  003C  1 1 m := 1;
00175  003D  1 1 While (m < 11) Do
00176  003E  1 2 Begin
00177  003F  1 2 If x[m] <> ' ' then Num := Num*10 + Value(x[M]);
00178  0040  1 2 m := m+1;
00179  0041  0 0
\end{verbatim}
Function Column2_Not_Null(X:UDLists; k:integer) :boolean;
  begin
  Column2_Not_Null := false;
  if (x(k,2)<>""") then Column2_not_null := true;
  end;

Function Column3_Not_Null(X:UDLists; k:integer) :boolean;
  begin
  Column3_Not_Null := false;
  if (x(k,3)<>"") then Column3_not_null := true;
  end;

Function Column4_Not_Null(X:UDLists; k:integer) :boolean;
  begin
  Column4_Not_Null := false;
  if (x(k,4)<>"") then Column4_not_null := true;
  end;

Function Valid_Integer(x:words) :boolean;
begin
  Var
  a : integer;
  valid_pool:boolean;
  begin
  valid_integer := true;
  valid_pool := true;
  a := 1;
  while( (valid_pool) and (a<>1) and (x[i]<>' ') ) do
    if not (x[i] in Digits) then
      begin valid_integer := false;
      valid_pool := false;
      end
    a := a + 1;
  end;

Function Valid_real(x:words) :boolean;
begin
  Var
  a:pointcount : integer;
  valid_pool:boolean;
  begin
  valid_real := true;
  valid_pool := true;
  a := 1;
  while( (valid_pool) and (a<11) and (x[i]<>' ') ) do
    begin
    end
  end;
If \( x[a] = \cdot \) then pointcount := pointcount + 1;
If (not (x[a] in RealDigits)) or (Pointcount > 1) then
  begin
    Valid_real := false;
    Valid_bool := false;
  end;
End;

If (not (x[a] in RealDigits) or (Pointcount > 1)) then
  begin
    valid_real := false;
    Valid_bool := false;
  end;
else a := a + 1;
End;

Procedure Find_closure_cycles;
{Calculates number of cycles necessary to complete a design}.
procedure Find_closure_cycles;
{Locates the first contiguous string of characters following command}.
procedure Find_words;
{Locates the first contiguous string of characters following command}.
procedure Find_number;
{Assigns num the value of the number found within a command}.
begin
  Find_words;
  Num := \( \cdot \).
end;
begin
  if word <> ' ' then begin
    m := 11;
  end;
  while (m > 0) do
    begin
      if word[a] <> ' ' then
        begin
          Num := Value(wor[n]) * Power(10, Lcount) + Num;
        end;
end;

```
01 Source Listing
  20-May-1986 21:31:02 VAX Pascal V3.2-S7

   -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 0 0
   00356 0 0
   00357 1 1  Cycles := 2 * Num + a + b;
   00358 1 1  End;
   00359 0 0
   00360 0 0
   00361 1 0
   00362 1 0
   00363 1 1  Begin
   00364 1 1  Radius := 0.75 + 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) * m;
   00368 1 1  j := 3;
   00369 1 1  If immediate_node then Find_number;
   00370 1 1  If num = 0 then Begin
   00371 1 1  Fidc_nole_cycles;
   00372 1 1  Cycles := Cycles + 2;
   00373 1 1  Strt := 0;
   00374 1 1  Sto := Cycles;
   00375 1 1  End;
   00376 1 1  else Begin
   00377 1 1  Find_loop_cycles;
   00378 1 1  Cycles := Cycles + 2;
   00379 1 1  Sto := Cycles + Strt;
   00380 1 1  End;
   00381 1 1  For Theta2 := Trunc(36 + Strt) to Trunc(36 + Sto)
   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-n)/b = 1.1415/2;
   00386 1 2  x := (a-o+B)*Cos(Theta) + b*Cos(Theta*(a-b)/b) + e*Sin(Beta);
   00387 1 2  y := (a-o+B)*Sin(Theta) - b*Sin(Theta*(a-b)/b) + e*Cos(Beta);
   00388 1 2  Length := Sqr(x*x + y*y);
   00389 1 2  cost := C(a/Length);
   00390 1 2  sinl := y/Length;
   00391 1 2  linr := sin(rotation);
   00392 1 2  Coss := cos(rotation);
   00393 1 2  cossum := cos*cost - sin*linr;
   00394 1 2  sinsum := sin*cost + cos*linr;
   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  End;
   00399 1 2  End;
   00400 1 1  strl := Sto;
   00401 0 0  end;
```

Procedure Get_cline;
Begin;
Readln(Cline);
If Not (Line[1] in Commands) then
Begin;
Writeln("Invalid Command");
End;
If (Line[1] <> ' ') then
Writeln('Invalid Code. ');
Halt;
End;

Procedure Get_cline2;
Begin;
Readln(Cline);
Clear_screen;
found := false;
for k := 2 to Procs do
WriteLn('Procedures follow:
Proclist[1]);
c := c + 1;
End;

Procedure move_teeth;
Begin;
find_number;
rotation := rotation + num * (2*3.1415/3);
End;

Procedure specifies wheels;
Begin;
find_number;
End;

Procedure specify_ring;
Begin;
find_number;
End;

Procedure specify_eel;
Begin;
End;

Procedure move_teeth;
Begin;
End;

Procedure Get_cline;
Begin;
Readln(Cline);
begin;
Halt;
end;

Procedure Get_cline2;
Begin;
Readln(Cline);
Clear_screen;
found := false;
for k := 2 to Procs do
WriteLn('Procedures follow:
Proclist[1]);
c := c + 1;
End;

Procedure move_teeth;
Begin;
find_number;
rotation := rotation + num * (2*3.1415/3);
End;

Procedure specifies wheels;
Begin;
find_number;
End;

Procedure specify_ring;
Begin;
find_number;
End;

Procedure specify_eel;
Begin;
End;
Procedure Specify_hole;  
begin  
h := Find_number;  
end;  

Procedure Specify_color;  
begin  
Find_word;  
Set_color(word, 1);  
end;  

Procedure Reset_wheel;  
begin  
Start := 3;  
Rotation := 3;  
end;  

Procedure Execute_command;  
begin  
if Cline[1] = 'C' then Clear_screen  
else if Cline[1] = 'G' then Go_prong  
else if Cline[1] = 'W' then Specify_wheels  
else if Cline[1] = 'H' then Move_teeth  
else if Cline[1] = 'I' then Specify_hole  
else if Cline[1] = 'L' then Specify_ring  
else if Cline[1] = 'F' then Specify_color  
else if Cline[1] = 'Z' then Reset_wheel  
end;  

Procedure Fill_valid_lists;  
begin  
Nums := ['A', ..., 'Z'];  
Real_digits := ['G', ..., '9', '1'];  
ValList[8] := 'HOLE';  
ValList[10] := 'START';  
ValList[12] := 'MOVE';  
ValList[14] := 'TON';  
ValList[16] := 'TON';  
ValList[17] := 'TON';
```pascal
Source Listing

```
RE'gin

While (Co ••

line(a[ ]
<>

I I) and

(a
<
81) and (b
<
11)

do

Begin

WordList(k,wordcount,b
]:=

Comm[ a];

b := b+1;

Letcount := Letcount + 1;

End;

End;

While (Callilline(al = I I)

and

(3
<
30)

do a

:=

a+1; (start of nllt vrd>

If

(Letcount

>:>

J)

and

(Wordcount

>:>

(J)

then

Wordcount := Wordcount + 1;

Letcount := 0;

End;

End;

1 1

End;

End;

End;

End;

Procedure Signal;

Error prompt)

Begin

Write('""""""""""""""""""""""""""""""

**

**

"""""""""""

**

**');

Errornum := Errornum + 1;

End;

End;

Procedure Syntax_check;

Checks word lists for illegal commands or contracts)

Begin

Starts := 0;

10 := 1;

Proc := 1;

Errornum := 1;

For k := 1 to Linecount do

Begin

If wordlist(k,2) = "START" then

Found := true;

l := 1;

While (not found) and (l <= Proc) do

20012

If wordlist[k+1] = "START" then)

Found := true;

Else l := l + 12

20012

If found then

Begin

Signal:

Write('wordlist[k+1],"has already been defined as a STARTed procedu."

20012

Write('"");

20012

Write('"");

20012

End;

20012

If wordlist[k+2] = "" then

Write('"START command must be followed by a valid procedure name.");

20012

Proc := Proc + 1;

20012

ProcList[Proc] := wordlist[k+2];
Source Listing

1  Procline[procs] := k+1;
2  If Dos <> 0 then Dos := 0;
3  Starts := Starts + 1;
4  If Starts > 1 then
5      Begin
6      Signal;
7      writeln("Missing STOP statement for procedure ",Proclist[procs-1]);
8      End;
9  Start := Start + 1;
10  If wordlist[k,1] = "STOP" then
11      Begin
12      If Dos > 0 then
13          Begin
14          Signal;
15          writeln("Missing END statement in procedure ",Proclist[procs]);
16          writeln("");
17          End;
18      If Dos < 0 then
19          Begin
20          Signal;
21          writeln("Missing DO statement in procedure ",Proclist[procs]);
22          writeln("");
23          End;
24      Starts := Starts - 1;
25      If Starts < 0 then
26          Begin
27          Signal;
28          writeln("Missing START statement.");
29          writeln("");
30          Starts := 1;
31          End;
32      If wordlist[k,1] = "END" then Dos := Dos + 1;
33      If wordlist[k,1] = "END" then Dos := Dos - 1;
34      If wordlist[k,1] = "RUN" then
35          Begin
36          Runs := Runs + 1;
37          wordlist[Runs] := wordlist[k,1];
38          End;
39      l := l + 1;
40      l := 1;
41      l := 1;
42      l := 1;
43      l := 1;
44      End;
45      l := 1;
46      l := 1;
47      l := 1;
48      l := 1;
49      l := 1;
50      l := 1;
51      l := 1;
52      l := 1;
53      l := 1;
54      l := 1;
55      l := 1;
56      l := 1;
57      l := 1;
58      l := 1;
59      l := 1;
60      l := 1;
61      l := 1;
62      l := 1;
63      l := 1;
64      l := 1;
65      l := 1;
66      l := 1;
67      l := 1;
68      l := 1;
69      l := 1;
70      l := 1;
71      l := 1;
72      l := 1;
73      l := 1;
74      l := 1;
75      l := 1;
76      l := 1;
77      l := 1;
78      l := 1;
79      l := 1;
80      l := 1;
81      l := 1;
82      l := 1;
83      l := 1;
84      l := 1;
85      l := 1;
86      l := 1;
87      l := 1;
88      l := 1;
89      l := 1;
90      l := 1;
91      End;
```pascal
Sourc", Listing [00TAALEXA.THESISJCYCLOGRAPH.PAS;34

Signal2;
writef('Above line begins with unknown command ",Wordlist[k,1],");
writef('"');
End;
Else
BEGIN
If (ordlist[k,1] = 'OVAL') or (Wordlist[k,1] = 'QUAD') or
(wordlist[k,1] = 'TRIANGLE') or (Wordlist[k,1] = 'RESET') or
(wordlist[k,1] = 'END') or (wordlist[k,1] = 'STOP') then
BEGIN
If Column2_Not_Null(Wordlist,k) then
BEGIN
Signal;
writeln('Wordlist[k,1] requires no operands. Unexpected characters found.");
writeln('"');
END;
Else If Column3_Not_Null(Wordlist,k) then
BEGIN
Signal;
writeln('Unexcected characters found after integer value.");
writeln('"');
END;
END;
END;
END.
ELSE If Column3_not_null(Wordlist,k) then
BEGIN
Signal;
writeln('Unexcected characters found after integer value.");
writeln('"');
END;
END;
END;
END;
END;
END.
ELSE If Column3_not_null(Wordlist,k) then
BEGIN
Signal;
writeln('Unexcected characters found after integer value.");
writeln('"');
END;
END;
END;
END.
END.

If (wordlist[k,1] = 'RING') or (wordlist[k,1] = 'HOLE') or
(wordlist[k,1] = 'WHEEL') then
BEGIN
If (wordlist[k,1] = 'RING') or (wordlist[k,1] = 'HOLE') or
(wordlist[k,1] = 'WHEEL') then
BEGIN
If (Column2_not_null(Wordlist,k)) then
BEGIN
Signal;
writeln('Wordlist[k,1] requires an integer value. No value found.");
writeln('"');
END;
END;
END.
END.
```

```pascal
00732   1 5                      Writeln('Unexpected characters found after pen color.');
00733   1 5                      Writeln('');
          End;
          End;
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('');
          End;
          End;
00744   1 4                      End;
00745   1 3                      End;
00746   1 3                      End;
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('');
          End;
          End;
00757   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('');
          End;
00763   1 5                      Else If Column4_not_null(Wordlist[k]) then
00764   1 5                      Begin;
00765   1 5                      Signal;
00766   1 5                      Writeln('Unexpected characters found after LOOPS.');
00767   1 5                      Writeln('');
          End;
          End;
00769   1 5                      End;
00770   1 4                      End;
00771   1 3                      End;
00772   1 3                      End;
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('');
          End;
          End;
00783   1 5                      Else If (wordlist[k,1] <> 'RIGHT') and
00784   1 5                      (wordlist[k,1] <> 'LEFT') then
          Re
```pascal
Signal;
Writeln('Keyword LEFT or RIGHT expected after integer value.');
End;
Else If Column4_not_null(wordlist[k]) then
Begin
Signal;
Writeln('Unexpected characters found after direction.');
End;
End;
End;
If (wordlist[k,1] = '"') Or
(wordlist[k,1] = '<') Or
(wordlist[k,1] = '>') then
Begin
If Not(Valid_Real(wordlist[k,2])) then
Signal;
Writeln('Wordlist[k,1] may be used to alter the values of 
"HOLE", "WHEEL", or "RING only."');
End;
Else If Column4_not_null(wordlist[k]) then
Begin
Signal;
Writeln('Unexpected characters found after end of command.');
End;
If Not(Valid_Real(wordlist[k,2])) then
Signal;
Writeln('Missing END statement in procedure "/Proclist[procs]"');
End;
If Not(Valid_Real(wordlist[k,2])) then
Signal;
Writeln('Missing DO statement in procedure "/Proclist[procs]"');
```
End;  
If Starts > 0 then  
Begin  
Signal;  
Writeln('Missing STOP statement for procedure ' + Proclist[procs]);  
Writeln('');  
End;  

Procedure Verify_runs;  
{Check to see if all procedures called are defined in the proclist}  
Begin  
For l := 1 to Runs do  
Begin  
Found := false;  
k := 1;  
while (not found) and (k <= proc) do  
If Runlist(l) = Proclist[k] then Found := true  
Else k := k + 1;  
If not found then Writeln(Runlist(l), 'not defined as a STARTed procedure.');  
End;  
End;  

Procedure Error_report;  
Begin  
errorFree := false;  
writeln('');  
If errorNum = then  
Begin  
writeln('No errors found.');  
errorFree := true;  
End;  
else writeln(errorNum, ' errors found.');  
End;  

Procedure Load_file_and_syntax_check;  
Begin  
Fill_valid_lists;  
Read_files;  
Build_word_lists;  
Syntax_check;  
Verify_runs;  
Error_Report;  
End;  

Procedure Clear_stacks;  
Begin  
For pointer := 1 to 20 do  
Begin  
PC_stack[pointer] := 0;  
Counter_stack[pointer] := 0;  
End;  
End;
Procedure Push_PC_and_count;
{Place return line number and times to repeat on stack}

Begin
Pointer := Pointer + 12
PC_Stack[Pointer] := PC + 12;

If wordlist[PC,2] = 'RUN' then Counter_Stack[pointer] := 0
Else Counter_Stack[pointer] := Value_Integer(wordlist[PC,2]) - 12

If Pointer > 20 then writeln('Fatal Error: Maximum nesting level exceeded.');

End;

Procedure Pop_PC_and_count;
{Remove items from stack}

Begin
Pointer := Pointer - 1:
End;

Procedure execute_program_command;

Begin
If wordlist[PC,1] = 'RESET' then Reset_wheel;

If (wordlist[PC,1] = 'GRAPH') and (wordlist[PC,3] = 'LOOPS') then
Begin
Num := Value_Integer(wordlist[PC,2]);
Go_to(PC);
End;

Else If (wordlist[PC,1] = 'GRAPH') then
Begin
Num := 3;
Go_to(PC);
End;

Else If (wordlist[PC,1] = 'HELS') then
Begin
Num := Value_Integer(wordlist[PC,2]);
End;

Else If (wordlist[PC,1] = 'HEL') then
Begin
Num := Value_Integer(wordlist[PC,2]);
End;

Else If (wordlist[PC,1] = 'MOVE') then
Begin
Num := Value_Integer(wordlist[PC,2]);
End;

Else If (wordlist[PC,1] = 'LEFT') then
Begin
Num := Value_Integer(wordlist[PC,2]);
End;

Else If (wordlist[PC,1] = 'RIGHT') then
Begin
Num := Value_Integer(wordlist[PC,2]);
End;

End;

Else If (wordlist[PC,1] = 'RING' then a := Value_Integer(wordlist[PC,2]);

Else If (wordlist[PC,1] = 'WHEEL' then b := Value_Integer(wordlist[PC,2]);

Else If (wordlist[PC,1] = 'WOLE' then h := Value_Integer(wordlist[PC,2]);

End;

End;
If (Wordlist[PC,1] = "***") then
begin
  If (Wordlist[PC,3] = "HOLE") then h := h + Value_Real(Wordlist[PC,2])
  else if (Wordlist[PC,3] = "WHEEL") then
    b := b + Value_Real(Wordlist[PC,2])
  else if (Wordlist[PC,3] = "RING") then
    a := a + Value_Real(Wordlist[PC,2])
  end;
end;

If (Wordlist[PC,1] = "+") then
begin
  If (Wordlist[PC,3] = "HOLE") then h := h - Value_Real(Wordlist[PC,2])
  else if (Wordlist[PC,3] = "WHEEL") then
    b := b - Value_Real(Wordlist[PC,2])
  else if (Wordlist[PC,3] = "RING") then
    a := a - Value_Real(Wordlist[PC,2])
  end;
end;

Procedure execute_program;
begin
  PC := procline[k];
  while not (pointer = 0) and (Wordlist[PC,1] = "STOP") do
  begin
    if Wordlist[PC,1] = "DO" then begin
      {place return line and repeat number on stack}
      push_PC_and_Count;
      end;
  end;
  else if Wordlist[PC,1] = "END" then begin
    {Increment number to repeat and go back}
    counter_stack[pointer] := counter_stack[pointer] - 1;
  end;
end;
PC := PC_Stack[pointer] - 1;
End

Else if Counter_Stack[pointer] = 0 then Pop_PC_and_Count;
End

Else if Wordlist(PC,1) = 'RUN' then
    Begin
        found := false;
        k := 12;
        While (not found) and (k <= procs) do
            If (Wordlist(PC,k) = Proclist[k]) then found := true
            Else k := k + 12;

        PC := PC - 12;
        Push_PC_and_Count;

        PC := Proclist[k] - 12;
        End

    Else if Wordlist(PC,1) = 'STOP' then
        Begin
            pc := PC_Stack[pointer];
            Pop_PC_and_Count;

            End;

    Execute_Program_command;

End

End (while)

End

Procedure load_programs;

Begin
    Load_file_and_syntax_check;

    If error_free then
        Clear_stack;

        Set_line;

        While (line[1] <> '***') do
            Begin
                If found then Execute_program;

                End;

        Inc;

        End;

End (Main);

(Set up graphics and graphics window)

Int_graphics;

Clear_screen;

Set_window(0,0,300,80,100,0,625,3);

(Defaults)

Old_color := 'GREEN';

Rotation := 0;

Str := 0;

a := 0;

b := 242;

n := 17;

Error_free := true;
Program input:

```

While (Cline[1] <> 'E') Do
  Begin
    While Not (Cline[11 in Commands) Do Get_cline;
    If (Cline[1] = '.') and (error_free) then Do_Program
    else If (Cline[1] = '.') then writeln('Program contains errors.');
    If (Cline[1] <> '.') then execute_command;
    Home;
  End;
End;
```

End.
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td><img src="image.png" alt="Image" /></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**VAR**

<table>
<thead>
<tr>
<th>SECTION</th>
<th>SINGLE (IN PROGRAM CYCLOGRAPH)</th>
<th>87</th>
<th>107</th>
<th>356</th>
<th>385</th>
<th>386</th>
<th>387</th>
<th>387</th>
<th>442</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>454 = 942 = 949 = 951 = 959 = 959 = 968 = 968 = 986 = 986 = 1057 =</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>BOUND INTEGER (IN PROCEDURE TXT)</td>
<td>172</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>BOUND INTEGER (IN PROCEDURE SET_COLOR)</td>
<td>176</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>INTEGER (IN FUNCTION VALID_INTEGER)</td>
<td>265</td>
<td>270 = 271</td>
<td>272 = 277 = 277</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>INTEGER (IN FUNCTION VALID_REAL)</td>
<td>233</td>
<td>253 = 290 = 290 = 292 = 293 = 297 = 297</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>INTEGER (IN PROCEDURE BUILD_WORD_LISTS)</td>
<td>549</td>
<td>592 = 565 = 566 = 568 = 570 = 571 = 571</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A2</td>
<td>INTEGER (IN PROGRAM CYCLOGRAPH)</td>
<td>110</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>485</td>
<td>FUNC (MULTIN) : INTEGER</td>
<td>312</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>INTEGER (IN PROGRAM CYCLOGRAPH)</td>
<td>48</td>
<td>307</td>
<td>353</td>
<td>364</td>
<td>367</td>
<td>385</td>
<td>386</td>
<td>386</td>
</tr>
<tr>
<td>B</td>
<td>INTEGER (IN FUNCTION VALID_INTEGER)</td>
<td>386</td>
<td>387</td>
<td>387</td>
<td>444 = 944 = 957 = 957 = 957</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>INTEGER (IN PROCEDURE SET_COLOR)</td>
<td>170</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>INTEGER (IN PROCEDURE BUILD_WORD_LISTS)</td>
<td>549</td>
<td>593 = 563 = 570 = 572 = 572</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>INTEGER (IN PROGRAM CYCLOGRAPH)</td>
<td>111</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>BOUND INTEGER (IN PROCEDURE TXT)</td>
<td>172</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>BOUND INTEGER (IN PROCEDURE SET_COLOR)</td>
<td>176</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B2</td>
<td>INTEGER (IN PROCEDURE BUILD_WORD_LISTS)</td>
<td>549</td>
<td>593 = 563 = 570 = 572 = 572 = 560</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>INTEGER (IN PROGRAM CYCLOGRAPH)</td>
<td>111</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>INTEGER (IN PROGRAM CYCLOGRAPH)</td>
<td>35</td>
<td>325 = 326 = 337</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BOOLEAN</td>
<td>TYPE</td>
<td>54 = 105 = 106 = 203 = 241 = 248 = 255 = 262 = 266</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BUILD_WORD_LISTS</td>
<td>PROC (ENTRY) : UNOUND</td>
<td>347</td>
<td>162</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHAR</td>
<td>TYPE</td>
<td>54 = 33 = 39 = 40 = 41 = 56 = 64 = 65 = 66 = 123</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLEAR_SCREEN</td>
<td>PROC (ENTRY) : UNOUND</td>
<td>129</td>
<td>172 = 174 = 176</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLEAR_STACKS</td>
<td>PROC (ENTRY) : UNOUND</td>
<td>152</td>
<td>479 = 1050</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLINE</td>
<td>VAR (ENTRY) : ARRAY OF CHAR (IN PROGRAM CYCLOGRAPH)</td>
<td>153</td>
<td>1537</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLUMN2_NOT_NULL</td>
<td>FUNC (ENTRY) : UNOUND</td>
<td>241</td>
<td>244 = 245 = 607 = 698 = 749 = 775</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLUMN3_NOT_NULL</td>
<td>FUNC (ENTRY) : UNOUND</td>
<td>241</td>
<td>244 = 245 = 607 = 698 = 749 = 775</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COLUMN4_NOT_NULL</td>
<td>FUNC (ENTRY) : UNOUND</td>
<td>255</td>
<td>254 = 764 = 791 = 821</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMMAND</td>
<td>VAR</td>
<td>CHAR (IN PROGRAM CYCLOGRAPH)</td>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
CYCLOGRAPH

01

COMMANDS
VAR [PSECT(SL0CAL)] SETOFCHARACTERS (IN PROGRAM CYCLOGRAPH)
  87 408 1062 = 1065
COMMLINE
VAR [PSECT(SL0CAL)] LINES (IN PROGRAM CYCLOGRAPH)
  126 531 = 534 534 560 = 565 568 570 570
COMMLIST
VAR [PSECT(SL0CAL)] LISTS (IN PROGRAM CYCLOGRAPH)
  126 534 = 560 560 603
COS
FUNC (BUILDIN) : SINGLE
* - 186 386 387 392 392
COST
VAR [PSECT(SL0CAL)] SINGLE (IN PROGRAM CYCLOGRAPH)
  69 392 = 393 393 394
COSR
VAR [PSECT(SL0CAL)] SINGLE (IN PROGRAM CYCLOGRAPH)
  70 393 = 396
COUNTER_STACK
VAR [PSECT(SL0CAL)] ARRAY OF INTEGER (IN PROGRAM CYCLOGRAPH)
  104 903 = 904 = 1003 1006 = 1006 1009
CYCLES
VAR [PSECT(SL0CAL)] SINGLE (IN PROGRAM CYCLOGRAPH)
  n8 310 = 317 317 = 358 372 = 372 374 374
  374 374 374 374 374
CYCLOGRAPH
PROC [PSECT(SCODE)]
  35
DDLISTS
TYPE ARRAY (IN PROGRAM CYCLOGRAPH)
  103 241 243 = 255
DIGITS
VAR [PSECT(SL0CAL)] SET OF CHAR (IN PROGRAM CYCLOGRAPH)
  126 372 = 493
DOS
VAR [PSECT(SL0CAL)] INTEGER (IN PROGRAM CYCLOGRAPH)
  112 398 = 523 625 636 658 = 659 658 660 = 660
DO_PROGRAMS
PROC [PSECT(SCODE), UNOUNCD] (IN PROGRAM CYCLOGRAPH)
  1032 1357
E
VAR [PSECT(SL0CAL)] SINGLE (IN PROGRAM CYCLOGRAPH)
  129 367 = 386 387
EOF
FUNC (BUILDIN) : IOCLEAN
  522
ERROR
VAR [PSECT(SL0CAL)] INTEGER (IN PROGRAM CYCLOGRAPH)
  113 190 = 193 57 = 502 = 369 874
ERROR_FREI
VAR [PSECT(SL0CAL)] IOCLEAN (IN PROGRAM CYCLOGRAPH)
  134 377 = 472 = 1035 1054 = 1066
ERROR_REPORT
PROC [PSECT(SCODE), UNOUNCD] (IN PROGRAM CYCLOGRAPH)
  160 185
EXECUTE_COMMAND
PROC [PSECT(SCODE), UNOUNCD] (IN PROGRAM CYCLOGRAPH)
  477 1069
EXECUTE_PROGRAM
PROC [PSECT(SCODE), UNOUNCD] (IN PROGRAM CYCLOGRAPH)
  1241
EXECUTE_PROGRAM_COMMAND
PROC [PSECT(SCODE), UNOUNCD] (IN PROGRAM CYCLOGRAPH)
  914 1027
FALSE
CONST 'FALSE' = 0
  208 203 244 244 251 251 258 258 258 273
  274 274 294 294 295 295 425 425 425 435
  435 607 607 669 669 856 856 867 867 1013
FILL_VALID_lists
PROC [PSECT(SCODE), UNOUNCD] (IN PROGRAM CYCLOGRAPH)
  433 80
FIND_CLOSURE_CYCLES
PROC [PSECT(SCODE), UNOUNCD] (IN PROGRAM CYCLOGRAPH)
  771
FIND_LOOP_CYCLES
PROC [PSECT(SCODE), UNOUNCD] (IN PROGRAM CYCLOGRAPH)
  133
<table>
<thead>
<tr>
<th>Procedure</th>
<th>Lines</th>
<th>Columns</th>
<th>Start Position</th>
<th>End Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOAD_FILE_AND_SYNTAX_CHECK</td>
<td>64</td>
<td>398</td>
<td>21:31:02</td>
<td>26:01:01</td>
</tr>
<tr>
<td>MOVE</td>
<td>433</td>
<td>482</td>
<td>21:31:02</td>
<td>26:01:01</td>
</tr>
<tr>
<td>MOVE_TEETH</td>
<td>434</td>
<td>483</td>
<td>21:31:02</td>
<td>26:01:01</td>
</tr>
<tr>
<td>NUMBER</td>
<td>370</td>
<td>444</td>
<td>21:31:02</td>
<td>26:01:01</td>
</tr>
<tr>
<td>NUMMEN2</td>
<td>370</td>
<td>444</td>
<td>21:31:02</td>
<td>26:01:01</td>
</tr>
<tr>
<td>NUMMENL</td>
<td>370</td>
<td>444</td>
<td>21:31:02</td>
<td>26:01:01</td>
</tr>
<tr>
<td>NOTECOLOR</td>
<td>57</td>
<td>360</td>
<td>21:31:02</td>
<td>26:01:01</td>
</tr>
<tr>
<td>ORDER</td>
<td>143</td>
<td>197</td>
<td>21:31:02</td>
<td>26:01:01</td>
</tr>
<tr>
<td>OUTPUT</td>
<td>195</td>
<td>195</td>
<td>21:31:02</td>
<td>26:01:01</td>
</tr>
</tbody>
</table>

**LCOUNT**

- **VAR**: (PSECT(SLOCAL)) INTEGER (IN PROGRAM CYCLOGRAPH)
  - 59 = 323 = 327 = 328 = 328 = 328 = 341 = 346 = 347 = 347

**LENGTH**

- **VAR**: (PSECT(SLOCAL)) SINGLE (IN PROGRAM CYCLOGRAPH)
  - 72 = 398 = 398 = 390 = 395 = 396

**LETCOUNT**

- **VAR**: (PSECT(SLOCAL)) INTEGER (IN PROGRAM CYCLOGRAPH)
  - 116 = 572 = 572 = 576 = 576

**LINE**

- **PROC**: (EXTERNAL, UNBOUND) (IN PROGRAM CYCLOGRAPH)
  - 163 = 397

**LINECOUNT**

- **VAR**: (PSECT(SLOCAL)) INTEGER (IN PROGRAM CYCLOGRAPH)
  - 115 = 552 = 555 = 558 = 601

**LINES**

- **TYPE**: ARRAY OF CHAR (IN PROGRAM CYCLOGRAPH)

**LISTS**

- **TYPE**: ARRAY OF LINES (IN PROGRAM CYCLOGRAPH)
  - 43 = 126

**LOAD_FILE_AND_SYNTAX_CHECK**

- **PROC**: (PSECT(SCODE), UNBOUND) (IN PROGRAM CYCLOGRAPH)
  - 377 = 1334

**MOVE**

- **PROC**: (PSECT(SCODE), UNBOUND) (IN PROGRAM CYCLOGRAPH)
  - 424 = 482

**MOVE_TEETH**

- **PROC**: (PSECT(SCODE), UNBOUND) (IN PROGRAM CYCLOGRAPH)
  - 433 = 483

**NUM**

- **VAR**: (PSECT(SLOCAL)) SINGLE (IN PROGRAM CYCLOGRAPH)
  - 137 = 340 = 349 = 349

**NUM2**

- **VAR**: (PSECT(SLOCAL)) SINGLE (IN FUNCTION VALUE_REAL)
  - 229 = 233 = 235 = 235 = 238 = 238

**NUMBER**

- **PARM**: (IN PROGRAM SET_COLOR)

**NUMMENL**

- **VAR**: (PSECT(SLOCAL)) INTEGER (IN PROGRAM CYCLOGRAPH)
  - 57 = 360 = 360 = 367

**ORD**

- **FUNC**: (BUILTIN) : INTEGER
  - 195 = 195 = 195

**OUTPUT**

- **VAR**: (EXTERNAL(PASFY_OUTPUT)) TEXT (IN PROGRAM CYCLOGRAPH)
  - 195 = 35 = 411 = 412 = 414 = 431 = 432 = 433 = 589 = 603
  - 513 = 516 = 617 = 628 = 629 = 630 = 639 = 640 = 645 = 655
  - 71 = 573 = 575 = 577 = 579 = 691 = 721 = 723 = 733 = 743 = 754 = 761
  - 792 = 795 = 808
Cyclograph

Cross Reference Listing

20-May-1986 21:51:02
20-May-1986 19:59:49

VAX Pascal v3.2-57

[DOITALEX.THE8IS]CYCLOGRAPH.PAS/34 (S)

Page 29

416  428  -  428  610  610  672  672  859  859  872
872 1016 1016 1061 1061

TRUNC

FUNC ( BUILTIN ) : INTEGER

* 512 = 365 365 381 381

TXT

PROC (EXTERNAL, UNBOUND) : IN PROGRAM CYCLOGRAPH

172

VAULST

VAR [PSELECT(STOLOCAL)] WORD_LIST ( IN PROGRAM CYCLOGRAPH ) :

135 695 = 696 = 497 = 498 = 499 = 500 = 501 = 502 = 503 =
504 = 505 = 506 = 507 = 508 = 509 = 510 = 511 = 512 = 513 =
672

VAULST

VAR [PSELECT(STOLOCAL)] WORD_LIST ( IN PROGRAM CYCLOGRAPH ) :

136 514 = 515 = 516 =

VALID_BOOL

VAR [PSELECT(STOLOCAL)] WORD_LIST ( IN PROGRAM CYCLOGRAPH ) :

266 269 = 271 274 =

VALID_BOOL

VAR [PSELECT(STOLOCAL)] WORD_LIST ( IN PROGRAM CYCLOGRAPH ) :

264 237 = 290 295 =

VALID_INTEGER

VAR [PSELECT(STOLOCAL)] WORD_LIST ( IN PROGRAM CYCLOGRAPH ) :

262 263 = 273 = 705 751 777

VALID_REAL

VAR [PSELECT(STOLOCAL)] WORD_LIST ( IN PROGRAM CYCLOGRAPH ) :

250 256 = 294 = 805

VALUE

VAR [PSELECT(STOLOCAL)] WORD_LIST ( IN PROGRAM CYCLOGRAPH ) :

192 195 = 212 220 = 235 236 = 346

VALUE_INTEGER

VAR [PSELECT(STOLOCAL)] WORD_LIST ( IN PROGRAM CYCLOGRAPH ) :

226 233 = 904 904 920 920 942 = 942 944 944

VALUE_REAL

VAR [PSELECT(STOLOCAL)] WORD_LIST ( IN PROGRAM CYCLOGRAPH ) :

198 223 = 935 957 959 964 976 973 975

VERIFY_RUNS

PROC [PSELECT(STOLOCAL)] UNBOUND : IN PROGRAM CYCLOGRAPH

977 982 984 986

WORD

VAR [PSELECT(STOLOCAL)] ARRAY OF CHAR ( IN PROGRAM CYCLOGRAPH ) :

35 322 = 327 = 339 = 344 346

WORDCOUNT

VAR [PSELECT(STOLOCAL)] INTEGER ( IN PROGRAM CYCLOGRAPH ) :

122 541 = 555 = 565 586 570 576 578 579 =

WORDLIST

VAR [PSELECT(STOLOCAL)] PROGRAM ( IN PROGRAM CYCLOGRAPH ) :

16 556 = 570 = 573 = 605 610 615 618 621 621

634 559 660 662 665 665 672 678 683 683

564 584 685 695 687 690 695 696 698

701 705 708 712 720 722 722 723 723 729

737 737 739 747 749 751 754 753 764 773

775 777 783 794 785 791 800 801 802 805

801 803 812 813 814 817 821 903 904 916

913 918 920 923 929 931 931 947 942 944

944 946 946 = 948 949 950 951 953 955

955 968 957 958 959 962 964 964 966 966

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 )

-1 -2 -64 199 226 262 230

WORDLIST

TYPE ARRAY OF WORDS ( IN PROGRAM CYCLOGRAPH )

-2 -123 124 135 136 137

WHITE

PROC ( BUILTIN )
**Cyclograph**

Cross Reference Listing

<table>
<thead>
<tr>
<th>Page</th>
<th>Line Numbers</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>412</td>
<td>WRITELN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PROC (BUILTIN)</td>
</tr>
<tr>
<td></td>
<td>413</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VAR [PSEC(SLOCAL)] SINGLE (IN PROGRAM CYCLOGRAPH)</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>PARM SINGLE (IN PROCEDURE LINE)</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>PARM SINGLE (IN PROCEDURE MOVE)</td>
</tr>
<tr>
<td></td>
<td>104</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>PARM SINGLE (IN FUNCTION POWER)</td>
</tr>
<tr>
<td></td>
<td>180</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>PARM CHAR (IN FUNCTION VALUE)</td>
</tr>
<tr>
<td></td>
<td>189</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>PARM WORDS (IN FUNCTION VALUE_REAL)</td>
</tr>
<tr>
<td></td>
<td>192</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>PARM WORDS (IN FUNCTION VALUE_INTEGER)</td>
</tr>
<tr>
<td></td>
<td>218</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>PARM ODLISTS (IN FUNCTION COLUMN2_NOT_NULL)</td>
</tr>
<tr>
<td></td>
<td>219</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>PARM ODLISTS (IN FUNCTION COLUMN3_NOT_NULL)</td>
</tr>
<tr>
<td></td>
<td>220</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>PARM ODLISTS (IN FUNCTION COLUMN4_NOT_NULL)</td>
</tr>
<tr>
<td></td>
<td>221</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>PARM WORDS (IN FUNCTION VALID_INTEGER)</td>
</tr>
<tr>
<td></td>
<td>270</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>PARM WORDS (IN FUNCTION VALID_REAL)</td>
</tr>
<tr>
<td></td>
<td>295</td>
<td></td>
</tr>
<tr>
<td>XLEFT</td>
<td></td>
<td>PARM SINGLE (IN PROCEDURE SET_WINDOW)</td>
</tr>
<tr>
<td></td>
<td>397</td>
<td></td>
</tr>
<tr>
<td>KLEFT</td>
<td></td>
<td>PARM SINGLE (IN PROCEDURE SET_WINDOW)</td>
</tr>
<tr>
<td></td>
<td>398</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>PARM SINGLE (IN PROCEDURE MOVE)</td>
</tr>
<tr>
<td></td>
<td>194</td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td></td>
<td>PARM SINGLE (IN PROCEDURE SET_WINDOW)</td>
</tr>
<tr>
<td></td>
<td>198</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>PARM SINGLE (IN PROCEDURE LINE)</td>
</tr>
<tr>
<td></td>
<td>194</td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td></td>
<td>PARM SINGLE (IN PROCEDURE MOVE)</td>
</tr>
<tr>
<td></td>
<td>198</td>
<td></td>
</tr>
<tr>
<td>YTOP</td>
<td></td>
<td>PARM SINGLE (IN PROCEDURE SET_WINDOW)</td>
</tr>
<tr>
<td></td>
<td>198</td>
<td></td>
</tr>
<tr>
<td>YTOP</td>
<td></td>
<td>PARM SINGLE (IN PROCEDURE SET_WINDOW)</td>
</tr>
<tr>
<td></td>
<td>198</td>
<td></td>
</tr>
</tbody>
</table>
Pascal Compilation Statistics

COMMAND QUALIFIERS

PASCAL/NOOPTIMIZE/LIS/CROSS CYCLOGRAPH

/CHECK=(BOUNDS,NOCASE SELECTORS,NOOVERFLOW,NOPOINTERS,NOSUBRANGE)
/DEBUG=(NOSYMBOLOLS,TRACEBACK)
/SHOW=(DICTIONARY,INCLUDE,NOINLINE,HEADER,SOURCE,STATISTICS)
/NOOPTIMIZE
/NOENVIRONMENT
/LIST=COUSEUSE_2:CODHALEXA.THEISI/TCYCLOGRAPH.LIS;1
/OBJECT=SUUSUSER_2:CODHALEXA.THEISI/TYCLOGRAPH,OJ;1
/CROSS_REFERENCE /ERROR_LIMIT=5 /NOG_FLOATING /NOMACHINE_CODE /NOOLD_VERSION /NOSTDARAD /WARNINGS

COMPILER INTERNAL TIMING

<table>
<thead>
<tr>
<th>Phase</th>
<th>Faults</th>
<th>CPU Time</th>
<th>Elapsed Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialization</td>
<td>175</td>
<td>00:11:2</td>
<td>00:11:8</td>
</tr>
<tr>
<td>Source Analysis</td>
<td>321</td>
<td>00:11:3</td>
<td>00:11:8</td>
</tr>
<tr>
<td>Source Listing</td>
<td>70</td>
<td>00:11:4</td>
<td>00:11:8</td>
</tr>
<tr>
<td>Tree Construction</td>
<td>174</td>
<td>00:11:5</td>
<td>00:11:8</td>
</tr>
<tr>
<td>Flow Analysis</td>
<td>7</td>
<td>00:11:6</td>
<td>00:11:8</td>
</tr>
<tr>
<td>Value Propagation</td>
<td>5</td>
<td>00:11:7</td>
<td>00:11:8</td>
</tr>
<tr>
<td>Profit Analysis</td>
<td>9</td>
<td>00:11:8</td>
<td>00:11:8</td>
</tr>
<tr>
<td>Context Analysis</td>
<td>795</td>
<td>00:11:9</td>
<td>00:11:8</td>
</tr>
<tr>
<td>Name Packing</td>
<td>11</td>
<td>00:11:10</td>
<td>00:11:8</td>
</tr>
<tr>
<td>Code Selection</td>
<td>121</td>
<td>00:11:11</td>
<td>00:11:8</td>
</tr>
<tr>
<td>Final</td>
<td>14</td>
<td>00:11:12</td>
<td>00:11:8</td>
</tr>
<tr>
<td>TOTAL</td>
<td>137</td>
<td>00:11:13</td>
<td>00:11:8</td>
</tr>
</tbody>
</table>

COMPILATION STATISTICS:

CPU Time: 00:11:7
Elapsed Time: 00:11:8
Page Faults: 1558
Compilation Complete
Ball State University VAXcluster VMS 4.3

File: UNIVUSER.UOTAAL.EXACTSCIENCEHOME.LIST71 (91256/-53), last revised on 20-MAY-1986 21:49, is a 8 block sequential file owned by UIC (COTAAL). The records are variable length with implied (CR) carriage control. The longest record is 132 bytes.

Joo HOME (1269) queued to LPAU on 20-MAY-1986 21:50 by user COTAAL, under account C at priority 4, started on printer _P175(3LPAU) on 20-MAY-1986 21:50 from queue LPAU.
HOME is a module, written in COBOL with language extensions, that clears the command line area and produces the prompt "**" which tells the user that he or she is in the interactive mode.

IDENTIFICATION DIVISION.
PROGRAM-ID. HOME.
AUTHOR. TODD ALEXANDER.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. VAX-11.
OBJECT-COMPUTER. VAX-11.
DATA DIVISION.
WORKING-STORAGE SECTION.
LINKAGE SECTION.

PROCEDURE DIVISION.
PROCEDURE DIVISION.
ENTRY-MAIN.

DISPLAY ** LINE 1 COLUMN 1.
DISPLAY "**" LINE 1 COLUMN 1.
DISPLAY ** LINE 1 COLUMN 1.
EXIT PROGRAM.
## Compilation Summary

### Program Sections

<table>
<thead>
<tr>
<th>Name</th>
<th>Bytes</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$CODE</td>
<td>60</td>
<td>PIC CON REL LCL SHR EXE RD NOWRT Align(2)</td>
</tr>
<tr>
<td>2 SPODATA</td>
<td>192</td>
<td>PIC CON REL LCL SHR NOEXE RD NOWRT Align(2)</td>
</tr>
<tr>
<td>3 COBSNAMES</td>
<td>24</td>
<td>PIC CON REL LCL SHR NOEXE RD NOWRT Align(2)</td>
</tr>
<tr>
<td>4 COBSNAMES</td>
<td>5</td>
<td>PIC CON REL LCL SHR NOEXE RD NOWRT Align(2)</td>
</tr>
</tbody>
</table>

### 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 /NOCONDITIONS
/CHECK=(NOPERFORM,NOERRORS) /DEBUG=(NOSTANDARD,TRACEBACK)
/WARNINGS=(NOSTANDARD,OTHER,NOINFORMATION)
/STANDARD=(NOSTANDARD,NCP) /NOFIPS

### Statistics

- Run Time: 0.73 seconds
- Elapsed Time: 1.77 seconds
- Page Faults: 112
- Dynamic Memory: 477 pages
HOMEZ Source Listing

10-May-1986 15:30:32
10-May-1986 15:30:20

VAX COBOL V3.2-42
BSUSUSER_2:[CDTALEXA.THERESIS]HOMEZ.COB27

***************

HOME2 is a module written in COBOL with language extensions,
that clears the command line area and produces the prompt
"p> " which tells the user that he or she is in the programming
mode.

IDENTIFICATION DIVISION.
PROGRAM-ID. HOME2.
AUTHOR. TCOO ALEXANDER.

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPILER. VAX-11.
OBJECT-COMPILER. VAX-11.

DATA DIVISION.
WORKING-STORAGE SECTION.
LINEAGE SECTION.

***************

PROCEDURE DIVISION.
COMMON.
DISPLAY ".
DISPLAY "p> " WITH NO-CR.
DISPLAY "p> " WITH NO-CR.
EXIT NOW.

***************
Compilation Summary

PROGRAM SECTIONS

<table>
<thead>
<tr>
<th>Name</th>
<th>Cycles</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 SCORE</td>
<td>0</td>
<td>PIC CON REL LCL SHR EXE RD NOWRT Align(2)</td>
</tr>
<tr>
<td>2 SPODATA</td>
<td>122</td>
<td>PIC CON REL LCL SHR NOEKE RD NOWRT Align(2)</td>
</tr>
<tr>
<td>3 CONSAMES...</td>
<td>24</td>
<td>PIC CON REL LCL SHR NOEKE RD NOWRT Align(2)</td>
</tr>
<tr>
<td>4 CONSAMES...</td>
<td>0</td>
<td>PIC CON REL LCL SHR NOEKE RD NOWRT Align(2)</td>
</tr>
</tbody>
</table>

DIAGNOSTICS

Informational: 7 (suppressed by command qualifier)

COMMAND QUALIFIERS

<table>
<thead>
<tr>
<th>Command Name</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>COBOL LIST</td>
<td>HOME2.2</td>
</tr>
</tbody>
</table>
| /NOTCOPY_LIST| /NOAMBIGUOUS /NOTMACHINE_CODE /NOTRECURSIVE /NOTRECURSIV
| /NOAMBIGUOUS | /NOTMACHINE_CODE /NOTRECURSIVE /NOTRECURSIV
| /NOTMACHINE_CODE | /NOTRECURSIVE /NOTRECURSIV
| /NOTRECURSIVE | /NOTRECURSIVE /NOTRECURSIV
| /NOTRECURSIV | /NOTRECURSIVE /NOTRECURSIV
| /NOTRECURSIVE | /NOTRECURSIVE /NOTRECURSIV
| /NOTRECURSIVE | /NOTRECURSIVE /NOTRECURSIV

STATISTICS

Run Time: 0.00 seconds
Elapsed Time: 1.00 seconds
Page Faults: 31
Dynamic Memory: 7128 bytes