SPEEDDRAW

A whole different way to draw on the Hi-Res screen

The Applesoft Hi-Res commands HPLOT and DRAW can create impressive artwork, but drawing accurate curves requires tedious use of the SIN and COS commands, and keeping track of where you are while drawing polygonal figures requires heroic efforts. SpeedDraw changes that by quickly creating smooth curves and sharp angles while taking care of all the trigonometry. Take a little time to learn SpeedDraw's ampersand (&) commands, and you will be rewarded with easy, fast graphics.

USING THE PROGRAM

Think in "polar coordinates" when using SpeedDraw in your Applesoft programs. We are accustomed to rectangular coordinates on the Applesoft Hi-Res screen, but in daily life we often describe a location in polar coordinates; you're more likely to say a spot is "14 feet northwest of the flagpole" than "10 feet north and then 10 feet west," and that's the essential difference between the two coordinate systems.

SpeedDraw uses angles specified in degrees, 360 to a circle, increasing clockwise with zero at the top, as on a map or a compass. You must specify an "origin" or center (XC%,YC%) on the Hi-Res screen. Then, for instance, the command &RAY draws a line of the specified length R%, in the specified direction A1% from XC%,YC%. All of SpeedDraw's inputs must be integer variables.

SpeedDraw has two kinds of commands, a centered kind in which the origin stays put until you move it, and a head-to-tail kind, in which a straight or curved line starts at the origin and then travels to the other end, which becomes the new origin. The head-to-tail routines &VECTR, &TURN, and &LTURN give you the most powerful tools for making circular and polygonal graphics.

Since SpeedDraw has a relocating loader and a link for any previously installed ampersand routine, it is compatible with various external ProDOS commands and ampersand-driven routines. For instance, if you're using DOS 3.3 and the Apple RENUMBER program from the DOS 3.3 master disk has been installed before SpeedDraw, both will be available simultaneously.

To use SpeedDraw in your Applesoft programs, add the following line to your program:

```
10 PRINT CHR$(4)"BRUN SPEEDDRAW":A1%=0:A2%=0:
   XC%=0:YC%=0:R%=0
```

This line does not have to be numbered as line 10, but it must come before any program lines that have any variables in them; the SpeedDraw variables must be the first ones declared in your program. You can BRUN SPEEDDRAW in immediate mode (i.e., at Applesoft's] prompt) if you wish. A1% and A2% are angles (see below for their purpose), XC% and YC% are the X and Y coordinates of the origin on the Hi-Res screen, and R% is the length of a line or the radius of a curve. The inputs can be changed whenever you like in the program, but they must be assigned a value at the start. As usual, X represents the horizontal dimension of the screen (0-279 from left to right) and Y is the vertical dimension (0-191 from top to bottom).

Commands

The examples below can be used as a tutorial. Boot up your Apple, BRUN SPEEDDRAW (assuming you've already entered and saved it), issue the commands HGR and HCOLOR=3, and press Return until you can see the cursor.

&CIRCLE draws a circle centered at XC%,YC% with radius R%. The commands

```
10 A1%=0:A2%=0:XC%=140:YC%=80:R%=78
20 &CIRCLE
```

will make a big circle on the screen when RUN.

&ARC draws an arc centered at XC%,YC% with radius R% from starting angle A1% to finishing angle A2%. Thus the statements

```
10 A1%=270:A2%=360:XC%=30:YC%=30:R%=30
20 &ARC
```

will produce a round corner at the upper left of the screen.

&RAY draws a line from XC%,YC% at an angle A1% with length R%. The statements

```
10 A1%=60:A2%=120:XC%=140:YC%=95
20 R%=80:&RAY
30 &ARC
40 A1%=A2%:&RAY
```

produce a "piece of pie" at the right center of the screen.

&VECTR draws a line from XC%,YC% at an angle A1% with length R% (just like &RAY) and then changes the coordinates XC%,YC% to the new end point of the line. Thus the statements

```
10 A1%=18:A2%=0:XC%=100:YC%=95:R%=40:N=5
20 FOR I=1 TO N
30 &VECTR
40 A1%=A1%-360/N:NEXT
```

produce a pentagon; other values of N make other polygons.

&TURN is a curved version of &VECTR. It starts at XC%,YC%, traveling in the direction A1%, turns right through an angle A2% with radius R%, then resets the values of XC%,YC% to the new end of the line and resets A1% to the new direction of travel. The statements

```
10 A1%=90:A2%=90:XC%=100:YC%=20:R%=30
20 FOR I=1 TO 4
30 &VECTR:&TURN:NEXT
```

produce a box with rounded corners. You can easily approximate an ellipse by concatenating &TURN commands:

```
10 1%=0:A2%=60:XC%=50:YC%=95:R%=0:R=35
20 FOR I=1 TO 6
30 R%=R:IF I=2 OR I=5 THEN R%=3*R
40 &TURN:NEXT
```

Note that the &TURN command takes care of keeping the start of one arc lined up smoothly with the end of the previous one.

&LTURN is a left turn version of &TURN. The statements

```
10 A1%=140:A2%=225:XC%=100:YC%=100
20 R%=20:&LTURN
30 R%=18:&TURN
```

make a letter S.

&MOVE works like &VECTR but without the line. It just moves the origin XC%,YC% to a new location at angle A1% and distance R%, without any plotting.

&SPOT puts one dot at a location R% away from XC%,YC% at angle A1%. It doesn't move the origin.

&DOTLINE is just like &RAY, but makes a broken line (a cycle of 3 dots on, 2 off). A rectangular grid can be made by these statements:

```
10 A1%=0:A2%=0:XC%=0:YC%=0:R%=150
20 FOR I=0 to 5
30 XC%=70:YC%=5+30*I:A1%=90:&DOTLINE
40 YC%=5:XC%=70+30*I:A1%=180:&DOTLINE
50 NEXT
```

&BARC makes a broken-line arc, with the same inputs as &ARC. To make a broken-line circle, set A2% equal to A1%.

Flexibility and Limitations

SpeedDraw is pretty forgiving. Its basic plotting routine DODOT checks coordinates before plotting a point and skips over any that would lie off the limits of the Hi-Res screen. For instance, if you call for a circle that lies partly off-limits, SpeedDraw will plot the part that's on the screen and skip the rest.

The command interpreter looks only at the first letter after the ampersand, so you needn't use the whole word or the same one. I find the words easier to remember.

The input variables must all be within Applesoft's range for integer variables, −32767 to 32767. It's okay for the coordinates of the origin XC%,YC% to lie off the screen. The radius or line length R% should be in the range 0-255; any other values will be ignored. Furthermore, if the radius of a curve is more than about 120, some small gaps will appear. The angles must not be below −14400. None of these limits is particularly confining, considering the size of the screen.

Applesoft's HPLOT and DRAW routines remain available, of course, and can be used well with SpeedDraw. Since SpeedDraw uses the ROM routine HPLOT0, an HPLOT TO command after a SpeedDraw command will pick up the line wherever SpeedDraw left off. The DRAW or XDRAW commands (without the AT) will start wherever SpeedDraw stopped, so you can define a location with &SPOT and then DRAW or XDRAW a shape at that place.

The arc routines run faster at smaller sizes. Speed shifts occur at radii of 80, 40, 20, 10, and 5, with the speed doubling at each shift.

Since R% is limited to 255, it's not possible to cross the screen horizontally or diagonally with a single &RAY, &VECTR or &DOTLINE command. However, two will make it; for instance

```
10 A1%=90:A2%=0:XC%=0:YC%=191:R%=140
20 &DOTLINE:&MOVE:&DOTLINE
```

draws a broken line across the bottom of the screen.

The routines &VECTR, &TURN, and &LTURN give you the most powerful tools for making circular and polygonal graphics.

Appearance Considerations

Since the video screen is a rectangular matrix of dots, anything other than straight vertical, horizontal, or 45 degree lines must have a slightly jagged appearance. Within that limitation, SpeedDraw makes the best possible curved lines since every point is individually calculated from the sine, cosine, and radius. Straight lines are not as bold as HPLOT lines but not as jagged. Broken lines are a bit ragged except at multiples of 45 degrees. The broken arcs made by &BARC look best if the radius is just below a radius where a speed-shift occurs (80, 40, 20, etc.).

Memory Usage

The demo program is not very long, so it can reside at the default location 2048 ($800). However, it's often wise to put a big Applesoft graphics program above the Hi-Res screen where there's plenty of room. To put your Applesoft program above both Hi-Res screens, use this line near the beginning of the program:

```
30 IF PEEK(104) < > 96 THEN POKE 103,1:POKE 104,96:
POKE 24576,0:PRINT CHR$(4);"RUN PROGRAM.NAME"
```

(Use the name of your program, of course.) If you have a IIGS, IIc, or a IIe with 128K of memory, you can do the relocation faster via /RAM under ProDOS:

```
30 IF PEEK(104) <> 96 THEN PRINT CHR$(4);"SAVE /RAM/
   DUMMY" :POKE 103,1:POKE 104,96: POKE 24576,0
   :PRINT CHR$(4);"RUN /RAM/DUMMY"
```

ENTERING THE PROGRAMS

If you have an assembler, enter the source code from **Listing 1** and save the object code as SPEEDDRAW.0. If you don't have an assembler, enter the hex codes from **Listing 2** and save the file with

```
BSAVE SPEEDDRAW.0,A$4069,L$414
```

Enter the Applesoft program in **Listing 3** and save it with

```
SAVE MAKE.SPEEDDRAW
```

After you have saved it, RUN MAKE.SPEEDDRAW, which alters the SPEEDDRAW.0 file to create a file SPEEDDRAW. Enter the Applesoft program from **Listing 4** and save it with

```
SAVE SPEEDDRAW.DEMO
```

Finally, enter the hex codes from **Listing 5** and save the file with

```
BSAVE CLOCKNUM,A$6000,L$FF
```

For help with entering *Nibble* listings, see the Typing Tips section.

HOW THE PROGRAM WORKS

SpeedDraw has a relocating loader, so it's compatible with either DOS 3.3 or ProDOS without wasting memory. It works with most ProDOS external commands and even other ampersand driven commands. SpeedDraw loads into the Hi-Res page 2, checks to see whether it has already been loaded (to avoid tying up another 1K of your memory), and, if not, sets up a link to the address previously specified in the ampersand vector. The "global page" location **$BF00** is checked to see if ProDOS is active, and if so it uses the ProDOS GETBUF routine to make room. Otherwise, HIMEM is moved down four pages, and some pointers are set for the relocation. The relocation routine at **lines 78-100** disassembles each instruction to see how long it is. Any three-byte instructions are checked to see if they specify an internal address, and if so, the address is adjusted to suit the new location. Then the monitor MOVE routine is called to finish the installation. If you make any modifications to the loader, be sure to adjust the ORG location so that INTERP still falls on the page boundary at $4100.

When Applesoft encounters an ampersand instruction, control jumps to the location specified in locations $3F5-7. In this case, it jumps to the command interpreter (**line 130** of the assembly listing). **Lines 135-138** look through the command table for a match of the first letter of the command and hand the command to the old vector if no match is found. If a match is found, **lines 146-147** get the low byte of the address of the selected routine from CMDTBL and push it on the stack for later use. The high byte of all the command routines is always the same; it's pushed on first. Then the loop in **lines 149-155** steps the text pointer over the rest of the command so there won't be an error when returning to Applesoft.

Now the INPUTS routine reads the input variables out of the table where Applesoft put them. The subroutines TWOBYTE and MOVBYTE step through the table. Then DIV90 is used to convert the angles to a special kind of degree, 1024 ($400) to a circle. This is a great convenience in the rest of the program because the high byte is the quadrant of the angle and an INC or DEC of the high byte is a right-angle turn. Also, a smaller unit is needed because 360 dots would not make a complete circle with a radius larger than 57. DIV90 multiplies by 256, then divides by 90. Then an RTS from DIV90 causes a jump to the address that was pushed on the stack.

The first command, RAY, makes a dot at distance RAD by calling DODOT, then repeatedly reduces RAD and makes more dots until RAD is zero.

DOTLINE works like RAY, but calls BDOT rather than DODOT. BDOT counts through a 5-dot cycle, calling DODOT on the first 3, then skipping 2. BARC steps through a series of angles while cycling through BDOT.

VECTOR calls RAY and then goes into MOVE, which finds the coordinates of the end point of the line and puts them in Applesoft's variable table to update XC% and YC%.

CIRCLE starts out by setting the start and finish angles to zero, and then falls through to ARC. SETSPD is called to deal with the need for more dots to make big circles; if the radius is bigger than 80, the angle steps one unit per dot, but for a smaller radius it can move faster, up to 32 units per dot if the radius is 5 or less. After the speed is set, ARC steps through the desired range of angles drawing dots with DODOT.

DODOT gets the X- and Y-coordinates of the dot, checks that the location is within the screen boundary and calls the Applesoft ROM routine HPLOT0 if it's okay; otherwise DODOT returns without plotting. The GETY and GETX routines used by DODOT both use GETCOORD, which is where the trigonometry starts. GETY drops back one quadrant because GETCOORD uses the sine of the angle, and the cosine of an angle is equal to the sine of an angle 90 degrees smaller.

The TURN command first finds the center of the arc, calculates the ending angle, and then calls ARC. Then the "origin" is moved back to the end of the line, ready for the next 'head-to-tail" command. LTURN works the same way but must cope with the fact that ARC only draws clockwise.

GETCOORD first looks to see whether the angle is in an odd or even quadrant to take advantage of symmetry; the sine of an angle equals the sine of 180 degees minus the angle. Then at **line 386** we take advantage of the fact that the sine of an angle larger than $F6 rounds off to $100 at one-byte precision. The result of this maneuvering is that the sines and cosines of 1024 different angles are generated to one-byte accuracy from a table just 245 bytes long.

After **line 390** looks up the sine, **lines 395-407** multiply the sine (or cosine) times the radius to get the horizontal or vertical distance from the origin. For the case of sine=$100, this routine is bypassed and the radius is the answer. Then PLUSMNS adds or subtracts the distance to or from the appropriate coordinate of the origin and we've got the coordinate.

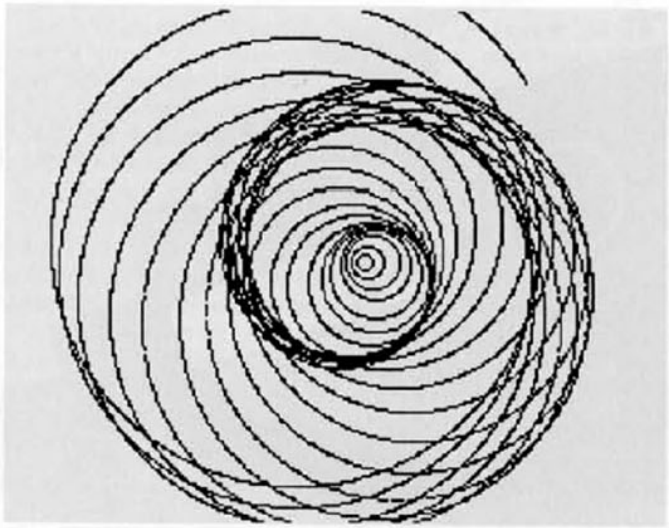

Figure 1: Spirals (Try Doing This with HPLOT)

How the DEMO Program Works

Line 60 sets up SpeedDraw's variable table, and then **lines 70-90** install SpeedDraw if it's not there already. The IF-THEN statement is not really necessary because SpeedDraw has internal protection against multiple installations (which would waste memory), but the IF-THEN avoids unnecessary disk access.

Lines 110-190 are a conventional menu to select an example. The first example, **lines 210-310**, draws a series of three spiral patterns (see **Figure 1**). The first, **line 240**, shows how easily and smoothly the &TURN command concatenates, and all three show how handy it is that the pattern can run off the edges of the screen without causing trouble.

Lines 330-420 make a randomly changing loop pattern. The loops are formed by alternating &TURN commands of two different radii.

The polygon routine in **lines 440-520** consists of three nested FOR-NEXT loops. The outermost one starts at **line 460**; when N is 3, the pyramid is made of triangles and so forth. The middle loop is defined by **line 480** and makes a stack of polygons with gradually increasing side R. Note the R%=R at the end of **line 480.** That is necessary because the index of a FOR-NEXT loop must be a floating-

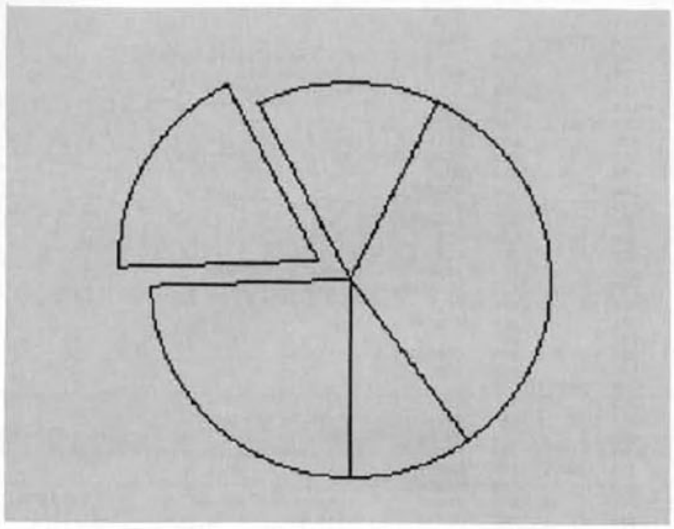

Figure 2: Pie Chart

point variable, while SpeedDraw understands only integer variables. The innermost loop, which actually draws each polygon, starts at **line 500**; it's really very simple.

The clock routine, **lines 540-890**, uses "page flipping" to give a smooth appearance. There are really two faces, one on each Hi-Res page. After inputting the time (note the error traps in **lines 560-570**), the time variables are initialized by **line 580**. The two POKEs in **line 590** tell the Applesoft DRAW command where to find the shape table CLOCKNUM, which is then loaded. Then in **line 610**, the two GOSUBs draw the face on both screens. The POKE in **line 620** is the key to page flipping, because it specifies which page (page 1=32, page 2=64) is to be drawn on, whether it is visible or not.

The first line of the timing loop erases the second hand by calling **line 770**, adds two seconds to the time for the page being worked on, and then draws the second hand again. **Lines 650** and **660** similarly update the minute and hour hands on alternate passes and then **line 670** puts the hub in the middle. Finally the numbers are refreshed by calling **line 710** and the page number index PG is reversed by PG=(PG=1)+1. Then two POKEs flip the page: The first one switches which one we see and the second one calls for drawing on the opposite one. Finally there's some flag-setting to determine whether the hour or the minute hand should be refreshed on the next pass. The loop ends in **line 690** after a little delay loop to adjust the time and a PEEK at the keyboard to see if a key has been pressed.

The subroutine at **lines 700-750** puts the numbers on the face by using the &SPOT command to specify the place, with the HCOLOR set to black. Then the numeral is drawn by **line 730** after POKEing the HCOLOR to white. (The DRAW command doesn't respond to a new HCOLOR without the AT n,n parameter specified.) Then **lines 740** and **750** similarly place a name on the face. You'll note that in this shape table both digits of the numbers 10, 11, and 12 are drawn by the single corresponding shape number, and the whole name is a single shape.

Subroutines at **lines 770, 780** and **790** draw the second, minute and hour hands repectively. The minute and hour hand routines both call the arrow/arrowhead routine in **lines 800-820**. The face subroutine starts at **line 840** by drawing a double circle. Then the FOR-NEXT loop in **lines 850-880** puts tick marks around the perimeter.

The pie chart in **lines 910-1020** is simpler (see **Figure 2**). **Lines 930-970** are an error-trapped input routine for the data. If there is an "S" at the end of a number, the flag FL(N) is set to 14, which is the distance that segment will be displaced. **Line 990** sets up the loop to do the segments and then moves the center point outward the distance FL(N). The segment is drawn by **line 1000**. **Line 1010** updates the starting angle for the next segment and that's it.

Finally, the error handling routine **lines 1040-1100** deals with the two likely disk errors; either the drive is open (or has no disk in it) or the disk in the default drive doesn't contain the specified file.

MODIFICATIONS

A mathematician might prefer that zero degrees be horizontally to the right with angles increasing counterclockwise. It's necessary to reverse the logic in only a few places to do that. For finer plotting of angles, the DIV90 routine could be left out so that the circle would have 1024 degrees, but some intuitive meaning of the angles would be lost.

LISTING 1: SPEEDDRAW.0 Source Code

```
 1 *
 2 * SPEEDDRAW.0 Source Code
 3 * by Jim Savage
 4 * Copyright(c) 1988
 5 * MicroSPARC, Inc.
 6 * Concord, MA 01742
 7 *
 8 * Merlin assembler
 9 *
10 * Equates for installation
11 *
12 DISTANCE EQU   $00
13 LENGTH   EQU   $2F          ;Length of command minus one
14 PCL      EQU   $3A          ;Pointer for disassembly
15 A1L      EQU   $3C
16 A2L      EQU   $3E
17 A4L      EQU   $42
18 STREND   EQU   $6F
19 HIMEM    EQU   $73
20 AMPER    EQU   $3F5
21 INSDS2   EQU   $F88C        ;Monitor disassembly routine
22 MOVE     EQU   $FE2C
23 GETBUF   EQU   $BEF5
24 *
25          ORG   $4069
26 *
27          LDY   #$0A
28          LDA   AMPER+1
29          STA   PCL
30          LDA   AMPER+2
31          STA   PCL+1
32 ]LOOP    LDA   (PCL),Y      ;Look at memory where ampersand points
33          CMP   INTERP,Y     ;See if it's Ampertrig
34          BNE   INSTALL      ;Not same so install
35          DEY                ;Check another byte
36          BNE   ]LOOP        ;Until 10 are OK
37          RTS                ;Don't want to repeat installation
38 *
39 INSTALL  LDA   AMPER+1       ;Get old vector
40          STA   OLDVEC+1     ;  and link
41          LDA   AMPER+2      ;     so it can
42          STA   OLDVEC+2     ;        still be used
43          LDA   #$4C         ;JMP instruction
44          STA   AMPER
45 *
46          LDA   $BF00        ;Check for ProDOS
47          CMP   #$4C
48          BEQ   PRO          ;Yes, use GETBUF
49 *
50          LDA   HIMEM+1      ;Get old HIMEM location
51          SEC
52          SBC   #$04         ;Need 4 pages of memory
53          STA   HIMEM+1      ;Move it down
54          STA   STREND+1     ;Also the string start
55          BNE   SETPTRS      ;Always
56 *
57 PRO      LDA   #04
58          JSR   GETBUF       ;Ask for 4 page buffer
59 *
60 SETPTRS  STA   A4L+1        ;Point MOVE to new home
61          STA   AMPER+2      ;And point ampersand vector at it
62          STA   GOTCMD+1     ;Take care of high byte
63          SEC
64          SBC   #>INTERP     ;Find how far to move
65          STA   DISTANCE
66 *
67          LDA   #>INTERP     ;Start of resident part
68          STA   PCL+1        ;Put in relocation pointer
69          STA   A1L+1        ;  and in MOVE origin
70          LDY   #00          ;Put
```

LISTING 1: SPEEDDRAW.0 Source Code

```
71          STY   PCL          ;  zero
72          STY   AMPER+1      ;     in
73          STY   A1L          ;       various
74          STY   A4L          ;         places
75          STY   STREND       ;
76          STY   HIMEM        ;
77 *
78 RELOC8    LDX   #00
79          JSR   INSDS2       ;Disassemble an instruction
80          LDY   LENGTH       ;How long?
81          CPY   #02          ;A 3-byter?
82          BNE   NXTINST
83          LDA   (PCL),Y      ;Get the 3rd byte
84          CMP   #>INSTALL    ;Below our code?
85          BMI   NXTINST      ;Yes, don't adjust
86          CMP   #>END+$100   ;Above our code?
87          BPL   NXTINST      ;Yes, don't adjust
88          CLC
89          ADC   DISTANCE     ;Adjust the address
90          STA   (PCL),Y      ;Put it back
91 NXTINST  INY
92          TYA
93          CLC
94          ADC   PCL          ;Advance the pointer
95          STA   PCL
96          BCC   NOPAGE
97          INC   PCL+1
98 NOPAGE   LDY   #00
99          LDA   (PCL),Y      ;Look at next byte
100         BNE   RELOC8       ;If not zero, keep on
101 *
102         LDA   #<END-1
103         STA   A2L          ;Finish preparing for MOVE
104         LDA   #>END-1
105         STA   A2L+1
106         JMP   MOVE         ;Exit via the move
107 *
108 *  Equates for resident routine
109 *
110 ANG1     EQU   $00
111 QUAD1    EQU   $01
112 ANG2     EQU   $02
113 QUAD2    EQU   $03
114 XCH      EQU   $04
115 XCL      EQU   $05
116 YCH      EQU   $06
117 YCL      EQU   $07
118 RAD      EQU   $08
119 VARTAB   EQU   $69
120 TXTPTR   EQU   $B8
121 SINE     EQU   $2F0
122 SINRAD   EQU   $2F1
123 LOCENT   EQU   $2F2
124 HICENT   EQU   $2F3
125 SPEED    EQU   $2F4
126 COUNT    EQU   $2F6
127 HPLOT0   EQU   $F457
128 *
129 *
130 INTERP   LDY   #00
131         LDX   #09          ;End of command table
132         LDA   (TXTPTR),Y   ;Get command character
133         CMP   #'&'         ;Second ampersand?
134         BEQ   TRYOLD       ;Yes, so not an Ampertrig command
135 CMDLOOP  CMP   CMDTBL,X     ;Look for match
136         BEQ   GOTCMD       ;Found
137         DEX
138         BPL   CMDLOOP      ;No, next
139         BMI   OLDVEC
140 TRYOLD   INC   TXTPTR       ;Step beyond ampersand
141         BNE   OLDVEC
142         INC   TXTPTR+1
143 OLDVEC   JMP   $0000         ;Send to address formerly in AMPER
144 GOTCMD   LDA   #$00          ;Loader puts hi byte of cmds here
145         PHA
146         LDA   CALLTBL,X    ;Set up the address
147         PHA                ;  for the "funny jump"
148 *
149 ]LOOP    INC   TXTPTR       ;Advance
150         BNE   NOCARI2
151         INC   TXTPTR+1
152 NOCARI2  LDA   (TXTPTR),Y   ;  until
153         BEQ   INPUTS       ;     zero
154         CMP   #$3A         ;       or colon
155         BNE   ]LOOP        ;Keep looking
156 *
157 INPUTS   LDX   #00          ;Index to input targets
158         LDY   #$02         ;Index to variable table
159         JSR   TWOBYTE      ;Move A1%
160         JSR   TWOBYTE      ;Move A2%
161         JSR   TWOBYTE      ;Move XC%
162         JSR   TWOBYTE      ;Move YC%
163         INY                ;Just a low byte
164         JSR   MOVBYTE      ;Move R%
165         LDX   #00          ;Index to ANG1
166         JSR   DIV90        ;Convert degrees to Hex
167         LDX   #02          ;Index to ANG2
168         JMP   DIV90        ;Again, then funny jump to command
169 *
170 RAY      JSR   DODOT
171         LDA   RAD          ;Check if done
172         BEQ   DONEVEC      ;Yes
173         DEC   RAD          ;Next one
174         JMP   RAY
175 *
176 DOTLIN   LDA   #03          ;Start in mid-dash
177         STA   COUNT
178 ]LOOP    JSR   BDOT         ;Draw a dot or not
179         LDA   RAD          ;Check if done
180         BEQ   DONEVEC      ;Yes
181         DEC   RAD
182         JMP   ]LOOP        ;Next
183 *
184 VECTOR   LDA   RAD
185         PHA                ;Put radius aside
186         JSR   RAY          ;Draw the line
187         PLA                ;Retrieve radius
188         STA   RAD
189 MOVEC    JSR   GETX
190         LDY   #$11         ;Index to XC% low
191         STA   (VARTAB),Y   ;Put away
192         STA   XCL
193         LDA   HICENT
194         DEY                ;XC% high
195         STA   (VARTAB),Y   ;  away
196         STA   XCH
197         JSR   GETY
198         LDY   #$18         ;Index to YC% low
199         STA   (VARTAB),Y   ;  away
200         STA   YCL
201         LDA   HICENT
202         DEY
203         STA   (VARTAB),Y   ;YC% high
204         STA   YCH
205 DONEVEC  RTS
206 *
207 BARC     LDA   #03          ;Start in mid-dash
208         STA   COUNT
209         JSR   SETSPD       ;Set speed appropriate to radius
210 ]LOOP    JSR   BDOT
211         JSR   NXTANGL
212         BNE   ]LOOP        ;Another dot or not
213         JMP   BDOT         ;Last one
214 *
215 CIRCLE   LDA   #00          ;Zero the
216         STA   ANG1         ;  start
217         STA   ANG2         ;    and
218         STA   QUAD1        ;      finish
219         STA   QUAD2        ;        angles
220 *
221 ARC      JSR   SETSPD       ;Set speed appropriate to radius
222 ]LOOP    JSR   DODOT
223         JSR   NXTANGL
224         BNE   ]LOOP        ;Another dot, else do last one
225 *
226 DODOT    JSR   GETY
227         SEC
228         CMP   #192         ;Legal Y?
229         BCS   RTS1         ;Too big, punt
230         LDY   HICENT       ;Hi byte <> 0?
231         BNE   RTS1         ;Return without plotting
232         PHA                ;Put Y aside
233         JSR   GETX
234         TAX                ;Put XL where HPLOT0 wants it
235         PLA                ;Retrieve Y-coordinate
236         LDY   HICENT       ;Put XH where HPLOT0 wants it
237         BMI   RTS1         ;Can't have negative X coord
238         CPY   #01          ;Check for oversize X coord
239         BMI   PLOTIT       ;O.K.
240         BEQ   CHEKLO       ;Look at lo byte
241 RTS1     RTS                ;Hi byte 2 or more, too big.
242 CHEKLO   SEC
243         CPX   #$18         ;X bigger than 279?
244         BCS   RTS1         ;Yes, return without plotting
245 PLOTIT   JMP   HPLOT0       ;Dot done!
246 *
247 LTURN1   JMP   LTURN        ;Just so all commands are on 1 page
248 *
249 TURN     INC   QUAD1        ;Right turn
250         JSR   MOVEC        ;Set center
251         DEC   QUAD1        ;Face
252         DEC   QUAD1        ;  outward
253         CLC
254         LDY   #$0A
255         LDA   (VARTAB),Y   ;Get A2% lo
256         LDY   #03
257         ADC   (VARTAB),Y   ;Add A1% lo
258         STA   (VARTAB),Y   ;Update A1% lo
259         STA   QUAD2        ;Set finish angle
260         LDY   #09
261         LDA   (VARTAB),Y   ;Likewise the hi bytes
262         LDY   #02
263         ADC   (VARTAB),Y
264         STA   (VARTAB),Y   ;Put in A1%
265         STA   ANG2
266         LDX   #02
267         JSR   DIV90        ;Convert to hex degrees
268         DEC   QUAD2        ;Face outward
269         JSR   ARC
270         JMP   MOVEC        ;Set origin at end & exit
271 *
272 LTURN    SEC
273         LDY   #03
274         LDA   (VARTAB),Y   ;Get A1% low
275         LDY   #$0A
276         SBC   (VARTAB),Y   ;Subtract A2% low
277         LDY   #03
278         STA   (VARTAB),Y   ;New angle in A1% low
279         DEY
280         LDA   (VARTAB),Y   ;Likewise hi byte
281         LDY   #09
282         SBC   (VARTAB),Y
283         LDY   #02
284         STA   (VARTAB),Y
285         LDA   ANG1         ;Move 1
286         STA   ANG2         ;  to 2
287         LDA   QUAD1
288         STA   QUAD2
289         DEC   QUAD1        ;Face in
290         JSR   MOVEC        ;Set center
291         JSR   GETA1        ;Get the (new) input A1%
292         INC   QUAD2        ;Face out
293         JSR   ARC          ;Do turn
294         JSR   GETA1        ;Restore (new) input A1%
295         JMP   MOVEC        ;Put center at end: Done
296 *
```

LISTING 1: SPEEDDRAW.O Source Code

```
297 GETA1    LDX   #00
298          LDY   #02
299          JSR   TWOBYTE     ;Move input A1% to ANG1/QUAD1
300          LDX   #00
301          JSR   DIV90
302          INC   QUAD1
303          RTS
304 *
305 SETSPD   LDA   QUAD2
306          AND   #03         ;Always work in 1st 4 quadrants
307          STA   QUAD2
308          LDA   #$20        ;Speed=32 if RAD<=5
309          STA   SPEED
310          LDA   #05         ;Test criterion
311 RADTEST  SEC
312          CMP   RAD
313          BCS   SET         ;Speed found
314          ASL   A           ;Double criterion
315          LSR   SPEED       ;Halve speed
316          BNE   RADTEST
317          INC   SPEED       ;Must be at least one
318 SET      SEC
319          LDA   #00
320          SBC   SPEED       ;Make mask for rounding angles
321          PHA               ;Put clone aside
322          AND   ANG1
323          STA   ANG1
324          PLA               ;Retrieve clone
325          AND   ANG2
326          STA   ANG2
327          RTS
328 *
329 NXTANGL  CLC
330          LDA   SPEED
331          ADC   ANG1        ;Increment angle by 1,2,4 etc
332          STA   ANG1
333          BCC   CHECK
334          INC   QUAD1       ;If last quad done
335 CHECK    LDA   QUAD1
336          AND   #03         ;Stay in 4 quadrants
337          CMP   QUAD2       ;In final quad?
338          BNE   RTSA        ;No, keep on
339          LDA   ANG1
340          CMP   ANG2        ;Same?
341 RTSA     RTS               ;Return w/ Z-flag set if done
342 *
343 BDOT     LDA   COUNT       ;Where are we?
344          BNE   NOTZED      ;Still counting
345          LDA   #04         ;Reset
346          STA   COUNT
347          RTS
348 NOTZED   CMP   #02         ;In plot part of cycle?
349          BMI   SKPDOT      ;No, skip
350          JSR   DODOT       ;Make your point
351 SKPDOT   DEC   COUNT
352          RTS
353 *
354 GETY     LDA   YCL         ;Prepare to calc Y-coordinate
355          STA   LOCENT
356          LDA   YCH         ;Allow for center off screen
357          STA   HICENT
358          DEC   QUAD1       ;Get ready for cosine
359          JSR   GETCOORD    ;Go get Y-coordinate
360          INC   QUAD1       ;Now for sine
361          RTS
362 *
363 GETX     LDA   XCH         ;Get ready for X
364          STA   HICENT
365          LDA   XCL
366          STA   LOCENT
367          JSR   GETCOORD    ;Go get x-coordinate
368          RTS
369 *
370 * Generate sine * radius
371 *
372 GETCOORD LDA   QUAD1
373          AND   #$01        ;Odd or even?
374          BEQ   EVEN
375          LDA   ANG1
376          BEQ   SINONE      ;Special case, sine=1
377          LDA   #$00
378          SEC
379          SBC   ANG1        ;Get complement of angle
380          CLC
381          BCC   EVEN+2
382 EVEN     LDA   ANG1        ;Don't complement
383          SEC
384          CMP   #$F6        ;Big enough that sine=1?
385          BCC   LOOKUP      ;No, look up sine
386 SINONE   LDA   RAD         ;Yes, so the leg = the radius
387          STA   SINRAD
388          JMP   PLUSMNS
389 LOOKUP   TAY               ;Angle indexes the table
390          LDA   SINTBL,Y    ;Got the sine!
391          STA   SINE
392 *
393 * Multiply radius by (co)sine *
394 *
395          LDA   #$00
396          STA   SINRAD
397          LDX   #$08        ;Set bit index
398 ]LOOP    ASL   A           ;Shift
399          ROL   SINRAD      ;  three
400          ASL   SINE        ;    bytes
401          BCC   NOCARI      ;No carry, no add
402          CLC
403          ADC   RAD
404          BCC   NOCARI
405          INC   SINRAD
406 NOCARI   DEX               ;Next bit
407          BNE   ]LOOP
408          ASL   A           ;Round off to closest hi bit
409          BCC   PLUSMNS
410          INC   SINRAD
411 PLUSMNS  LDA   QUAD1
412          AND   #02         ;In neg half of sine wave?
413          BNE   SUBTRACT    ;Yes
414          LDA   LOCENT      ;Get lo byte of center
415          CLC
416          ADC   SINRAD      ;Add (sine * radius)
417          BCC   GOTIT
418          INC   HICENT
419 GOTIT    RTS
420 SUBTRACT LDA   LOCENT      ;Get lo byte of center
421          SEC
422          SBC   SINRAD      ;Subtract (sine * radius)
423          BCS   GOTIT
424          DEC   HICENT
425          RTS
426 *
427 TWOBYTE  JSR   MOVBYTE     ;Move high byte
428          JSR   MOVBYTE     ;Move low byte
429          INY               ;5 bumps
430          INY               ; to get to
431          INY               ; the next
432          INY               ; variable
433          INY               ; in the table
434          RTS
435 *
436 MOVBYTE  LDA   (VARTAB),Y  ;Get byte from variable table
437          STA   $00,X       ;Put on zero page
438          INY               ;Bump
439          INX               ;  indices
440          RTS
441 *
442 DIV90    LDA   00,X        ;Look at hi byte (bytes rev!)
443          BPL   DODIV       ;If positive, proceed
444          CLC
445          ADC   #$5A        ;Add 360x64 to it
446          STA   00,X
447 DODIV    LDY   #00         ;Begin division
448          STY   $00,X
449          LDY   #$10        ;Bit index
450 ]LOOP    ASL   $00,X
451          ROL   $01,X
452          ROL   A
453          CMP   #$5A        ;Bigger than 90?
454          BCC   TOOBIG      ;No
455          SBC   #$5A
456          INC   $00,X
457 TOOBIG   DEY
458          BNE   ]LOOP
459          RTS               ;Return with bytes reversed
460 *
461 SINTBL   DS    $F6         ;Leave $F6 bytes for SIN table
462 *
463 CMDTBL   ASC   'RDVMBCASLT'
464 CALLTBL  DFB   #<RAY-1,#<DOTLIN-1,#<VECTOR-1
465          DFB   #<MOVEC-1,#<BARC-1
466          DFB   #<CIRCLE-1,#<ARC-1,#<DODOT-1
467          DFB   #<LTURN1-1,#<TURN-1
468 END      EQU   *
469          LST   OFF
```

END OF LISTING 1

LISTING 2: SPEEDDRAW.0

Start: 4069 Length: 414

```
56|4069:A0 0A AD F6 03 85 3A
7A|4070:AD F7 03 85 3B B1 3A D9
65|4078:00 41 D0 04 88 D0 F6 60
F3|4080:AD F6 03 8D 1B 41 AD F7
38|4088:03 8D 1C 41 A9 4C 8D F5
2D|4090:03 AD 00 BF C9 4C F0 0B
02|4098:A5 74 38 E9 04 85 74 85
27|40A0:70 D0 05 A9 04 20 F5 BE
AC|40A8:85 43 8D F7 03 8D 1E 41
AC|40B0:38 E9 41 85 00 A9 41 85
43|40B8:3B 85 3D A0 00 84 3A 8C
D8|40C0:F6 03 84 3C 84 42 84 6F
11|40C8:84 73 A2 00 20 8C F8 A4
CE|40D0:2F C0 02 D0 0F B1 3A C9
72|40D8:40 30 09 C9 45 10 05 18
7B|40E0:65 00 91 3A C8 98 18 65
EB|40E8:3A 85 3A 90 02 E6 3B A0
07|40F0:00 B1 3A D0 D5 A9 7C 85
7B|40F8:3E A9 44 85 3F 4C 2C FE
08|4100:A0 00 A2 09 B1 B8 C9 26
8E|4108:F0 0A DD 69 44 F0 0E CA
02|4110:10 F8 30 06 E6 B8 D0 02
AD|4118:E6 B9 4C 00 00 A9 00 48
50|4120:BD 73 44 48 E6 B8 D0 02
8D|4128:E6 B9 B1 B8 F0 04 C9 3A
56|4130:D0 F2 A2 00 A0 02 20 40
26|4138:43 20 40 43 20 40 43 20
23|4140:40 43 C8 20 4C 43 A2 00
90|4148:20 53 43 A2 02 4C 53 43
14|4150:20 C1 41 A5 08 F0 41 C6
77|4158:08 4C 50 41 A9 03 8D F6
45|4160:02 20 A1 42 A5 08 F0 30
90|4168:C6 08 4C 61 41 A5 08 48
94|4170:20 50 41 68 85 08 20 C9
B9|4178:42 A0 11 91 69 85 05 AD
A9|4180:F3 02 88 91 69 85 04 20
A3|4188:B7 42 A0 18 91 69 85 07
E0|4190:AD F3 02 88 91 69 85 06
62|4198:60 A9 03 8D F6 02 20 5C
BF|41A0:42 20 A1 42 20 88 42 D0
5E|41A8:F8 4C A1 42 A9 00 85 00
63|41B0:85 02 85 01 85 03 20 5C
```

LISTING 2: SPEEDDRAW.O

continued from page 78

```
5E | 41B8:42 20 C1 41 20 88 42 D0
5E | 41C0:F8 20 B7 42 38 C9 C0 B0
6A | 41C8:16 AC F3 02 D0 11 48 20
FA | 41D0:C9 42 AA 68 AC F3 02 30
68 | 41D8:06 C0 01 30 08 F0 01 60
5D | 41E0:38 E0 18 B0 FA 4C 57 F4
D6 | 41E8:4C 1A 42 E6 01 20 76 41
FB | 41F0:C6 01 C6 01 18 A0 0A B1
C5 | 41F8:69 A0 03 71 69 91 69 85
A5 | 4200:03 A0 09 B1 69 A0 02 71
22 | 4208:69 91 69 85 02 A2 02 20
29 | 4210:53 43 C6 03 20 B6 41 4C
6A | 4218:76 41 38 A0 03 B1 69 A0
F9 | 4220:0A F1 69 A0 03 91 69 88
2B | 4228:B1 69 A0 09 F1 69 A0 02
F9 | 4230:91 69 A5 00 85 02 A5 01
25 | 4238:85 03 C6 01 20 76 41 20
61 | 4240:4D 42 E6 03 20 B6 41 20
39 | 4248:4D 42 4C 76 41 A2 00 A0
D2 | 4250:02 20 40 43 A2 00 20 53
6A | 4258:43 E6 01 60 A5 03 29 03
05 | 4260:85 03 A9 20 8D F4 02 A9
80 | 4268:05 38 C5 08 B0 09 0A 4E
97 | 4270:F4 02 D0 F5 EE F4 02 38
F3 | 4278:A9 00 ED F4 02 48 25 00
0F | 4280:85 00 68 25 02 85 02 60
5C | 4288:18 AD F4 02 65 00 85 00
25 | 4290:90 02 E6 01 A5 01 29 03
75 | 4298:C5 03 D0 04 A5 00 C5 02
CA | 42A0:60 AD F6 02 D0 06 A9 04
DA | 42A8:8D F6 02 60 C9 02 30 03
64 | 42B0:20 C1 41 CE F6 02 60 A5

BA | 42B8:07 8D F2 02 A5 06 8D F3
08 | 42C0:02 C6 01 20 D7 42 E6 01
10 | 42C8:60 A5 04 8D F3 02 A5 05
61 | 42D0:8D F2 02 20 D7 42 60 A5
C5 | 42D8:01 29 01 F0 0C A5 00 F0
5F | 42E0:0F A9 00 38 E5 00 18 90
42 | 42E8:02 A5 00 38 C9 F6 90 08
65 | 42F0:A5 08 8D F1 02 4C 20 43
20 | 42F8:A8 B9 73 43 8D F0 02 A9
F6 | 4300:00 8D F1 02 A2 08 0A 2E
FE | 4308:F1 02 0E F0 02 90 08 18
9C | 4310:65 08 90 03 EE F1 02 CA
7D | 4318:D0 EC 0A 90 03 EE F1 02
07 | 4320:A5 01 29 02 D0 0D AD F2
5D | 4328:02 18 6D F1 02 90 03 EE
56 | 4330:F3 02 60 AD F2 02 38 ED
AA | 4338:F1 02 B0 F6 CE F3 02 60
49 | 4340:20 4C 43 20 4C 43 C8 C8
F5 | 4348:C8 C8 C8 60 B1 69 95 00
F9 | 4350:C8 E8 60 B5 00 10 05 18
97 | 4358:69 5A 95 00 A0 00 94 00
70 | 4360:A0 10 16 00 36 01 2A C9
41 | 4368:5A 90 04 E9 5A F6 00 88
A0 | 4370:D0 F0 60 00 00 00 00 00
BD | 4378:00 00 00 00 00 00 00 00
9D | 4380:00 00 00 00 00 00 00 00
C0 | 4388:00 00 00 00 00 00 00 00
B7 | 4390:00 00 00 00 00 00 00 00
F4 | 4398:00 00 00 00 00 00 00 00
FD | 43A0:00 00 00 00 00 00 00 00
EC | 43A8:00 00 00 00 00 00 00 00
33 | 43B0:00 00 00 00 00 00 00 00

30 | 43B8:00 00 00 00 00 00 00 00
59 | 43C0:00 00 00 00 00 00 00 00
3C | 43C8:00 00 00 00 00 00 00 00
E7 | 43D0:00 00 00 00 00 00 00 00
B8 | 43D8:00 00 00 00 00 00 00 00
B5 | 43E0:00 00 00 00 00 00 00 00
A4 | 43E8:00 00 00 00 00 00 00 00
83 | 43F0:00 00 00 00 00 00 00 00
60 | 43F8:00 00 00 00 00 00 00 00
89 | 4400:00 00 00 00 00 00 00 00
8C | 4408:00 00 00 00 00 00 00 00
63 | 4410:00 00 00 00 00 00 00 00
80 | 4418:00 00 00 00 00 00 00 00
AD | 4420:00 00 00 00 00 00 00 00
BC | 4428:00 00 00 00 00 00 00 00
B7 | 4430:00 00 00 00 00 00 00 00
E8 | 4438:00 00 00 00 00 00 00 00
41 | 4440:00 00 00 00 00 00 00 00
64 | 4448:00 00 00 00 00 00 00 00
33 | 4450:00 00 00 00 00 00 00 00
30 | 4458:00 00 00 00 00 00 00 00
E5 | 4460:00 00 00 00 00 00 00 00
16 | 4468:00 52 44 56 4D 42 43 41
A1 | 4470:53 4C 54 4F 5B 6C 75 98
16 | 4478:AB B5 C0 E7 EA

TOTAL: 0F23

END OF LISTING 2
```

LISTING 3: MAKE.SPEEDDRAW

```
37 | 10  REM *
C0 | 20  REM * MAKE.SPEEDDRAW
B9 | 30  REM * BY JIM SAVAGE
AE | 40  REM * COPYRIGHT(C) 1988
CB | 50  REM * MICROSPARC, INC.
24 | 60  REM * CONCORD, MA 01742
45 | 70  REM *
A3 | 80 D$ =  CHR$ (4):P = 3.1416 / 512
27 | 90  PRINT D$"BLOAD SPEEDDRAW.0"
2E | 100  FOR I = 0 TO 245:SN =  SIN (I * P)
63 | 110 SN =  INT ((SN * 256) + .5): POKE 17267 + I
       ,SN
4D | 120  NEXT
0A | 130  PRINT D$"BSAVE SPEEDDRAW,A$4069,L$414"

TOTAL: 6210

END OF LISTING 3
```

LISTING 4: SPEEDDRAW.DEMO

```
CB | 1  REM *
47 | 2  REM * SPEEDDRAW.DEMO
47 | 3  REM * BY JIM SAVAGE
3F | 4  REM * COPYRIGHT(C) 1988
3B | 5  REM * MICROSPARC, INC.
BF | 6  REM * CONCORD, MA 01742
BF | 7  REM *
CB | 60 A1% = 0:A2% = 0:XC% = 0:YC% = 0:R% = 0: REM
       PUT VARIABLES WHERE SPEEDDRAW LOOKS FOR
       THEM
63 | 70  ONERR  GOTO 1050
A7 | 80 AMP = 256 *  PEEK (1015): IF  PEEK (AMP + 7)
       < > 38 OR  PEEK (AMP + 9) < > 10 THEN FL
       $ = "SPEEDDRAW": PRINT  CHR$ (4);"BRUN";FL$

B6 | 90  POKE 216,0
23 | 100 :
D7 | 110  REM  MENU
4B | 120  READ NN: FOR I = 1 TO NN: READ MN$(I): NEX
       T
B1 | 130  DATA  6,SPIRALS,LOOPS,POLYGONS,CLOCK,"PIE
       CHART",QUIT

91 | 140  TEXT : HOME : PRINT  CHR$ (21);: HTAB 14:
       PRINT "SPEEDDRAW DEMO": PRINT : HTAB 4:
       PRINT "COPYRIGHT(C) 1988 MICROSPARC, INC."
       : FOR I = 1 TO 40: PRINT "=";: NEXT : PRINT
       : PRINT
84 | 150  FOR I = 1 TO NN: PRINT : HTAB 14: PRINT I;
       " - ";MN$(I)
F6 | 160  NEXT
35 | 170  VTAB 20: HTAB 14: PRINT "CHOOSE 1-";NN;:
       GET A$:A =  VAL (A$): IF A < 1 OR A > NN
       THEN 170
6F | 180  PRINT : ON A GOTO 220,340,450,550,920
63 | 190  HOME : VTAB 23: END
9F | 200 :
A4 | 210  REM  SPIRALS
F5 | 220 A1% = 0:A2% = 45:XC% = 140:YC% = 95:R% = 4
55 | 230  HCOLOR= 3: HGR2
D2 | 240  FOR I = 1 TO 80: & TURN:R% = R% + 2: NEXT

56 | 250  HGR2 :XC% = 140: FOR I = 1 TO 25
FC | 260 R% = I:A1% = 24 * I: & MOVE
13 | 270 R% = 3.4 * I: & CIRCLE: NEXT
35 | 280  HGR2 : FOR I = 1 TO 40
38 | 290 XC% = 140:YC% = 95:R% = I:A1% = 24 * I:
       & MOVE
E7 | 300 R% = 3.4 * I: & CIRCLE: NEXT
76 | 310  GOTO 140
01 | 320 :
60 | 330  REM  LOOPS
72 | 340 A1% = 180:A2% = 0:XC% = 0:YC% = 0:R% = 0
C0 | 350 A2 = 100 +  RND (1) * 160
C7 | 360 A2% = A2:A1% = (180 + A2) / 2:XC% = 150:YC%
       = 26
5B | 370  HGR2 : HCOLOR= 3
B2 | 380  FOR I = 1 TO 24
DA | 390 R% = 130:A2% = 50: & TURN
85 | 400 R% = 16:A2% = A2: & TURN
97 | 410  NEXT : IF  PEEK ( - 16384) > 127 THEN  POK
       E  - 16368,0: GOTO 140
A9 | 420  GOTO 340
D6 | 430 :
B7 | 440  REM  POLYGONS
34 | 450 A1% = 0:A2% = 0:XC% = 40:YC% = 50:R% = 0:
       HCOLOR= 3
13 | 460  FOR N = 3 TO 8: HGR2
B2 | 470 NP = 360 / N:D = 0
12 | 480  FOR R = 35 TO 75 STEP 5:R% = R
58 | 490 D = D + 1:XC% = 110 - D:YC% = 30 - D
28 | 500  FOR A = 90 TO 450 STEP NP:A1% = A: & VECTR
```

```
35 | 510  NEXT A,R,N
75 | 520  GOTO 140
34 | 530 :
F5 | 540  REM  CLOCK
34 | 550 A1% = 0:A2% = 0:XC% = 140:YC% = 95:R% = 80:
       PG = 1
25 | 560  HOME : VTAB 4: PRINT "ENTER CORRECT TIME--
       -": PRINT : INPUT "   HOUR (1-12): ";A$:HR
       =  VAL (A$): IF HR < 1 OR HR > 12 THEN 560

0A | 570  VTAB 8: INPUT "   MINUTE (0-59): ";A$:MIN
       =  VAL (A$): IF (MIN < 1 AND A$ <  > "0")
       OR MIN > 59 THEN 570
45 | 580 SEC(1) = 60 * MIN + 3600 * HR:SEC(2) = SEC(
       1) + 1:HR(1) = SEC(1) / 3600:HR(2) = SEC(2)
       / 3600
71 | 590  HGR : HCOLOR= 3: ROT= 0: SCALE= 1: POKE 23
       2,0: POKE 233,96: ONERR  GOTO 1050
CC | 600 FL$ = "CLOCKNUM": PRINT : PRINT  CHR$ (4);"
       BLOAD";FL$
A0 | 610  POKE 216,0: GOSUB 840: HGR2 : GOSUB 840
3C | 620  HCOLOR= 3: GOSUB 780: POKE 230,32: GOSUB 7
       80
D7 | 630  REM  TIME LOOP
55 | 640 XC% = 140:YC% = 95: HCOLOR= 0: GOSUB 770:SE
       C(PG) = SEC(PG) + 2: HCOLOR= 3: GOSUB 770
63 | 650  IF FL THEN  HCOLOR= 0: GOSUB 780:HR(PG) =
       SEC(PG) / 3600: HCOLOR= 3: GOSUB 780: GOTO
       670
6F | 660  HCOLOR= 0: GOSUB 790:MIN(PG) = SEC(PG) / 6
       0: HCOLOR= 3: GOSUB 790
F0 | 670 XC% = 140:YC% = 95: FOR R = 1 TO 3:R% = R:
       & CIRCLE: NEXT
F3 | 680  GOSUB 710:PG = (PG = 1) + 1: POKE  - 16298
       - PG,0: POKE 230,PG * 32:A =  PEEK ( - 163
       36) +  PEEK ( - 16336): IF PG = 1 THEN FL =
       FL = 0
9A | 690  FOR I = 1 TO 15: NEXT : ON  PEEK ( - 16384
       ) < 128 GOTO 640: POKE  - 16368,0: GOTO 140

8E | 700  REM  ALPHA
27 | 710 R% = 64: FOR A = 1 TO 12:A1% = 30 * A:XC% =
       140:YC% = 95
22 | 720  HCOLOR= 0: & SPOT
C2 | 730  POKE 28,127: DRAW A: NEXT
E3 | 740 R% = 34:A1% = 0: HCOLOR= 0: & SPOT
73 | 750  POKE 28,127: DRAW 13: RETURN
80 | 760  REM  HANDS
BC | 770 XC% = 140:YC% = 95:A1% = (SEC(PG) - 3600 *
       INT (HR(PG))) * 6:R% = 70: & RAY: RETURN
84 | 780 A1% = HR(PG) * 30:R% = 40: GOSUB 800: RETUR
       N
D9 | 790 A1% = MIN(PG) * 6:R% = 64: GOSUB 800: RETUR
       N
EB | 800 XC% = 140:YC% = 95: & VECTR
E7 | 810 A1% = A1% + 160:R% = 5: & RAY
D5 | 820 A1% = A1% + 40: & RAY: RETURN
5F | 830  REM  FACE
87 | 840 XC% = 140:YC% = 95: FOR R = 80 TO 79 STEP
       - 1:R% = R: & CIRCLE: NEXT
8C | 850  FOR A = 1 TO 60:A1% = 6 * A
C1 | 860 XC% = 140:YC% = 95
E7 | 870 R1 = 75: IF A / 5 =  INT (A / 5) THEN R1 =
       71
A9 | 880 R% = R1: & MOVE:R% = 79 - R1: & RAY: NEXT
4C | 890  RETURN
41 | 900 :
9B | 910  REM  PIE CHART
8D | 920 A1% = 0:A2% = 0:XC% = 140:YC% = 95:R% = 80:
       SUM = 0: HCOLOR= 3:A1 = 0
8B | 930  TEXT : HOME : VTAB 5: INPUT "ENTER NUMBER
       OF PIECES OF DATA (2-9) ";N$:N =  VAL (N$):
       IF N < 2 OR N > 9 THEN 930
BE | 940  PRINT : PRINT "ENTER THE DATA. PLACE AN 'S
       ' AFTER      ANY SEGMENT THAT SHOULD BE SEP
       ARATED.": FOR I = 1 TO N
30 | 950  VTAB 12 + I: PRINT "SEGMENT ";I;: INPUT "
       - ";DA$:DA(I) =  VAL (DA$): IF DA(I) = 0
       THEN 950
28 | 960 FL(I) = 0: IF  RIGHT$ (DA$,1) = "S" THEN FL
       (I) = 14
15 | 970 SUM = SUM + DA(I): NEXT
8F | 980  HGR
3E | 990  FOR I = 1 TO N:XC% = 140:YC% = 75:R% = FL(
       I):DD = 360 * DA(I) / SUM:A1% = A1 + DD / 2
       : & MOVE
3C | 1000 R% = 70:A1% = A1: & RAY:A2% = A1 + DD:
       & ARC:A1% = A2%: & RAY
08 | 1010 A1 = A1 + DD: NEXT
1F | 1020  VTAB 22: PRINT : PRINT "PRESS RETURN TO C
       ONTINUE ";: POKE  - 16368,0: GET A$: GOTO 1
       40
D0 | 1030 :
17 | 1040  REM  DISK ERROR HANDLER
F2 | 1050  POKE 216,0:ER =  PEEK (222): TEXT
0C | 1060  HOME : VTAB 10: PRINT "DISK ERROR"
7E | 1070  PRINT : PRINT : IF ER = 6 THEN  PRINT "FI
       LE ";FL$;" MUST BE ON THIS DISK": IF  PEEK
       (218) + 256 *  PEEK (219) = 80 THEN  END
5B | 1080  IF ER = 8 THEN  PRINT "CHECK DRIVE"
F6 | 1090  PRINT : PRINT "PRESS <RETURN> TO TRY AGAI
       N ";: GET A$: PRINT : IF FL$ = "CLOCKNUM"
       THEN 140
E9 | 1100  GOTO 70
```

TOTAL: 4712

END OF LISTING 4

LISTING 5: CLOCKNUM

Start: 6000 Length: FF

```
86 | 6000:0D 00 1C 00 25 00 33 00
F1 | 6008:40 00 4C 00 59 00 66 00
C4 | 6010:73 00 80 00 8D 00 A2 00
74 | 6018:B2 00 C6 00 24 BC 96 31
43 | 6020:17 2D 5D 58 00 4E 12 3F
A7 | 6028:3F 2C 68 28 20 1C 3F 17
82 | 6030:5E 58 00 DE 72 2D 5D 20
3D | 6038:5F 20 0C 3C 3F 3F 58 00
83 | 6040:A9 D6 24 3F 27 0C 0C 0C
5E | 6048:36 5D 58 00 98 72 2D 5D
B9 | 6050:20 E4 3F 27 2C 2D 2D 58
4D | 6058:00 AD F6 3F 5F 20 2C 1C
2E | 6060:0C 0C 2D 5D 58 00 17 36
E4 | 6068:0D 58 58 58 28 20 3F 3F
F4 | 6070:5F 58 00 BF 76 2D 5D 20
41 | 6078:5F 60 E4 3F 17 36 58 00
F2 | 6080:3F 20 0C 2D 15 36 77 1E
BE | 6088:1E 3F 5C 58 00 36 DE 3B
23 | 6090:67 24 24 67 4D 09 2D DE
9A | 6098:33 95 12 2D 5D 20 24 24
63 | 60A0:58 00 92 3B 67 24 24 67
8A | 60A8:4D 09 3E 0E 36 BE 2D 5D
79 | 60B0:58 00 58 60 2D 15 F6 BF
B0 | 60B8:17 4D 11 3F 3F DF 3F 0C
93 | 60C0:24 24 67 5D 58 00 D8 3F
5D | 60C8:27 FC B3 FF 3B 1F 2E 1E
EF | 60D0:36 4D 21 24 4D 36 17 2D
E6 | 60D8:0D 24 AC 12 2D 5D 20 24
DB | 60E0:40 31 36 36 2E 2D 5D 20
F2 | 60E8:FC 5B 28 6D 40 31 36 36
E8 | 60F0:17 2D 0D 2D 25 20 1C 3F
34 | 60F8:17 36 5D 28 35 58 00
```

TOTAL: A42B

END OF LISTING 5