# ZOT!

TIPS 'N TECHNIQUES

**Applesoft Hi-Res *shapes offer a number of advantages over HPLOT commands, and you can do a lot with a six-byte shape.***

The most useful shape to DRAW and XDRAW in Apple Hi-Res graphics is also the smallest. I call it a zot. It is one dot long and one dot wide. What it lacks in size, it makes up for in versatility. The zot can be rotated, scaled up, and used to draw a variety of larger shapes and lines. ZOT, a shoot-'em-down UFO game with graphics, sound effects and scoring, shows you how to program with the zot. ZOT is only six lines long; it may give you some ideas about how to write better and shorter graphics routines.

If you have tried to create shapes using shape tables, you probably bogged down somewhere in the *Applesoft II BASIC Programming Reference Manual*, and went out to buy a ready-made graphics utility that makes shape tables for you. Relax — this is not a tutorial on shape tables, and the shape table for ZOT is only six bytes long.

To get the shape table into memory for the purposes of experimentation, enter the Monitor with CALL −151 and type:

```
300:1 0 4 0 4 0 <RETURN>
```

Then type:

```
E8:0 3 <RETURN>
```

and <CTRL>C <RETURN> to return to BASIC.

## DRAWING WITH ZOT

Let's look at a few examples that illustrate zot's versatility. Line 10 sets up the Hi-Res screen and sets a number of constant values. Keep this line for all three examples in this section.

```
10 HGR2:ROT=0:SCALE=1:HCOLOR=3:X=140:Y=96:K=3.984
```

Add the following line:

```
20 DRAW 1 AT X,Y:FOR I=1 TO 64:R=R+16-64*(R=64):
   ROT=R:SCALE=I*2:DRAW 1:NEXT
```

Run it and see what happens as zot grows. If you didn't know already, notice that issuing a DRAW 1 without new x,y coordinates automatically tacks the new shape onto the last shape drawn.

To see how zot can be used as a rotating line of variable length, replace line 20 with:

```
20 HPLOT 0,Y TO 279,Y
30 ROT=PDL(0)/K:SCALE=1+PDL(1)/3:XDRAW 1 AT X,Y:
   XDRAW 1 AT X,Y:GOTO 20
```

Run it and use the paddles to change length and direction. It is good to use two XDRAWs because XDRAW does not erase the background (a DRAW and XDRAW would erase the background). For a slower loop but less flicker, put:

```
FOR I=1 TO 100:NEXT
```

between the two XDRAWs.

Finally, try this crawler, which you control with paddle 1:

```
20 DRAW 1 AT X,Y
30 ROT=PDL(1)/K:DRAW 1:GOTO 30
```

One advantage of using zot to draw a trail around the screen is that, unlike a series of HPLOTs, it does not evoke an ILLEGAL QUANTITY error when it goes off the screen. It just reappears on the opposite side of the screen.

## HOW TO PLAY ZOT

The program ZOT, (not to be confused with the zot shape itself) is a simple shoot-the-UFO type game, designed to demonstrate the versatility of the zot programming technique. It can't be considered a full-fledged game because it doesn't keep score, but it will give you some idea of the many things you can do with zot.

The program requires a paddle or joystick. When you run it, you will see a small UFO which flies back and forth above a gun turret. The turret can be aimed with the paddle or joystick, and pressing the paddle or joystick button will fire the gun. The object, of course, is to hit the UFO.

## ENTERING THE PROGRAM

To key in the ZOT program, type the Applesoft code shown in **Listing 1** and save it with the command:

SAVE ZOT

For help in entering *Nibble* listings, see "A Welcome to New *Nibble* Readers" at the beginning of this issue.

## HOW IT WORKS

Very long program lines cut space and speed up program execution, but they are not easy to understand. **Line 80** of **Listing 1** POKEs the shape table into memory and sets up the shape table vector, just as we did earlier using the Monitor. The variables are defined in this first line for easy reference and quicker program execution. The variables are LE (left edge), RE (right edge), SU (scale of the UFO), SB (scale of the base line), SG (scale of the laser gun), SZ (scale of the laser ray), and so on.

**Line 90** starts the main loop, allowing 10 attempts, with NG being the number of attempts. The screen is cleared, and random factors are set up to change the x and y coordinates of the UFO at varying speeds to give the game some variety and challenge.

**Line 100** draws the base line, draws the UFO in its first position and initiates the loop (FOR I= 0 TO 1) that keeps the UFO flying until you take a shot. After the UFO is drawn, the x and y coordinates are changed by previously calculated amounts to the position for the next plot. If the plot approaches the edge of the screen, the x-change (DX) is reversed to make the UFO fly back.

**Line 110** draws the gun at a rotation determined by the value of PDL(1). The key in this line is the statement I=(PEEK(−16286)>127). This bit of logic leaves I=1 if the PDL(1) button is pressed (when PEEK(−16286)>127 is a true statement) and I=0, otherwise. If I=1, the loop exits at the end of the line because a shot has been fired.

**Line 120** increases the scale of the zot to SZ and draws it along the line of the gun, creating a laser-like beam. If this elongated shape collides with the UFO on the screen, the collision counter (a special zero page location at $EA, decimal 234) holds a value greater than zero. Hence, the PEEK(234). If a hit is recorded, the speaker beeps and an explosion is created by drawing a random series of zots around the impact point. The hit counter (NH) is also incremented. **Line 130** is self-explanatory.

You will notice that throughout this little game no HPLOTs were used, just one shape — our versatile zot.

**LISTING 1: ZOT**

```
10  REM ***********************
20  REM *         ZOT         *
30  REM *  BY TIM KENDRICK    *
40  REM * COPYRIGHT (C) 1986 *
50  REM * BY MICROSPARC, INC *
60  REM * CONCORD, MA  01742 *
70  REM ***********************
80  HOME : CLEAR : POKE 768,1: POKE 769,0: POKE
     770,4: POKE 771,0: POKE 772,4: POKE 773,
     0: POKE 232,0: POKE 233,3:K = .1255:DY =
     .97:B = 180: HCOLOR= 3:GX = 140:GY = 190
     :LE = 0:RE = 279:BE = 191:SU = 6:SG = 20
     :SZ = 190:SB = 255:RB = 16:RS = 48
90  FOR NG = 1 TO 10: HGR2 :X = 267:RN =  RND
     (1) * 20 - 10:DX = RN + 1 *  SGN (RN) *
     (RN < 1):Y = 1 +  RND (1) * 170:UY = Y:K
     Y = ( RND (1) * 2 < 1): IF DX > 0 THEN X
      = 1
100  ROT= RB: SCALE= SB: DRAW 1 AT LE + 12,BE
     : SCALE= SU: DRAW 1 AT X,UY: FOR I = 0 TO
     1:Z =  PEEK ( - 16336): ROT= RB: SCALE=
     SU: XDRAW 1 AT X,UY:Y = Y * DY:UY = Y +
     KY * (B - 2 * Y):X = X + DX: IF ((X + 6)
      > RE) OR (X < LE) THEN DX =  - DX:X = X
      + DX
110 RG = RS +  PDL (1) * K: ROT= RG: SCALE= S
     G: XDRAW 1 AT GX,GY:I = ( PEEK ( - 16286
     ) > 127): SCALE= SU: ROT= RB: DRAW 1 AT
     X,UY: SCALE= SG: ROT= RG: XDRAW 1 AT GX,
     GY:Z =  PEEK ( - 16336): NEXT
120  SCALE= SZ: DRAW 1 AT GX,GY: IF  PEEK (23
     4) > 0 THEN  PRINT  CHR$ (7):NH = NH + 1
     : FOR I = 4 TO 64 STEP 4: ROT= I: SCALE=
     1 +  RND (1) * 25: DRAW 1 AT X,UY:Z =  PEEK
     ( - 16336): NEXT
130  NEXT : TEXT : HOME : VTAB 10: PRINT "YOU
      HIT "NH" OUT OF TEN": PRINT " PRESS <RE
     TURN> FOR ANOTHER GO": GET Z$: PRINT : GOTO
     80
```

END OF LISTING 1

# WINDOW SHOW

SECOND FEATURE

**With five *ampersand commands, you can control shadowed text windows on the Hi-Res screen.***

When Apple Computer introduced the Lisa and Macintosh computers, it debuted an innovative user interface that has rapidly become an industry standard. Windows and pull-down menus can now be found in all types of software for all types of computers. If the folks at Apple had had a crystal ball, they surely would have implemented a standard user interface for the Apple // series of computers, as well.

The Apple // has windowing capabilities built into the System Monitor, but these routines are very limited. Neither pull-down menus nor dialog boxes are supported. Furthermore, the Apple text screen is pallid compared to the graphic and font capabilities of the Macintosh window interface. It offers a challenge: to create a user interface for the Apple // series that allows for true windowing capabilities.

Window Show does just that. While it lacks the elegance of the Macintosh interface, it does the programmer's dirty work by providing a standard set of subroutines to perform windowing tasks.

## USING WINDOW SHOW

In Applesoft, all calls to Window Show are made through the Applesoft ampersand (&) command, so incorporating Window Show into your own programs is easy. For a summary of commands, see Table 1. First, a command to start up the window is required. This is accomplished by:

```
&NEW A$,L,W,T,B
```

where A$ is the title line, and L, W, T and B are equivalent to the Monitor window specifications Window Left, Window Width, Window Top and Window Bottom, respectively.

For example, to dress up a graphic program with some windows, you first need to decide what you want for the title line. This is the line that appears at the top of the screen, and can be used to implement pull-

TABLE 1: Command Summary

| Command | Function |
|---|---|
| &NEW sexpr,aexpr1, aexpr2,aexpr3,aexpr4* | Clears the screen, sets up pointers, and draws a base window. |
| &DRAW aexpr1, aexpr2,aexpr3,aexpr4* | Clears the new output window. |
| &STORE aexpr1, aexpr2,aexpr3,aexpr4* | Saves a window on the stack and clears the new output window. |
| &RESTORE | Restores the last window &STOREd. |
| &HOME | Clears the current output window. |

*sexpr = title line; aexpr1 = window left; aexpr2 = window width; aexpr3 = window top; aexpr4 = window bottom.

down menus or just to display information. It could be just about anything, but let's say you need the words Colors, Brushes, Shapes, Printer and File. You would place these words in a string and include it as the first parameter of the &NEW command.

Next, a window size must be chosen. For graphics you need a large window. Let's use a 38 × 20 line window starting at line 2. Note that Window Show can only handle a window with a minimum left and top value of 1, a width of up to column 38 and a bottom value of no more than 23. Putting all of these things together, the maximum size window can be set up with:

```
&NEW " Colors Brushes Shapes
Printer File", 1, 38, 2, 22
```

That's all there is to it!

## &STORE, &DRAW and &RESTORE

You may be thinking, "Sure that looks pretty, but how useful is it?" The real power of Window Show is in the &STORE and &DRAW commands. Once the base window is open, new windows can be opened by the commands:

```
&STORE L,W,T,B
```

and

```
&DRAW L,W,T,B
```

where the parameters L, W, T and B are the same as for the &NEW command. Both of these commands clear a graphic window, open a text window at the same location, and home the cursor in this new window. The only difference between them is that &STORE creates a window that can be restored by &RESTORE, and &DRAW merely draws the window on the screen.

Why use &DRAW when &STORE is just as functional? The answer is that when &STORE creates a window that can be &RESTOREd, it utilizes a large portion of memory to save what is under it. Since in many programs there is a large main window that will never be removed, it makes sense not to waste valuable memory saving what is under it.

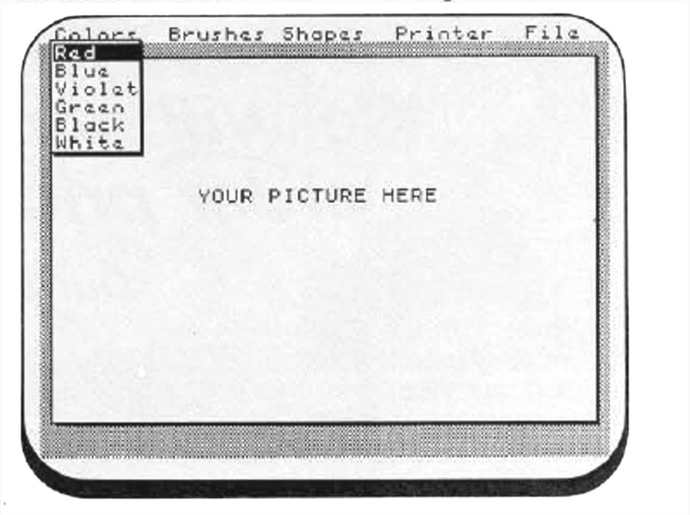
*Note:* If you attempt to &STORE a window, and there is not enough memory available to save the background, the window will not be opened. Window Show does not generate an error message in this case. Try a smaller window, or close a window first.

In the graphics program example, let's say that you want to open up a window below the word Colors and display a menu of available colors: Red, Blue, Violet, Green, Black and White (see **Figure 1**). Since the longest of these words is Violet (six letters) and there are six choices, a 6 × 6 window is adequate. A command to open such a window would be:

```
&STORE 1,6,1,7
```

Since the text window is open over the same area, Applesoft PRINT statements are all

***FIGURE 1: Simulated Pull-down Menu Using Window Show***



```
Colors  Brushes Shapes  Printer  File
Red
Blue
Violet
Green
Black
White

          YOUR PICTURE HERE
```

that are needed to display the color names in the window.

After a window is created with &STORE, it is possible to erase it with the command &RESTORE. When a window is &RESTOREd, not only is the screen area under the window restored, but the old cursor position and old window parameters are restored with them, as if the window had never existed.

## &HOME

&HOME is identical to the HOME command in Applesoft, except that it clears the Hi-Res portion of the text window along with the text screen.

A word of warning: Do not use TEXT, HOME, GR, HGR, HGR2, PR# or IN# in programs that use Window Show. Some of these commands will do strange things to Window Show and there is no guarantee that you can recover.

In order to incorporate the Window Show routines into your Applesoft programs, you must relocate your program above the Hi-Res page. The following line at the beginning of your program will accomplish this:

```
10 IF PEEK (104) <> 64 THEN POKE
   103,1: POKE 104,64 : POKE
   16384,0: PRINT CHR$(4) "RUN
   EXAMPLE"
```

where EXAMPLE is the name of your program. To set up all of the pointers correctly, you must also **BLOAD CHAR.SET** and **BRUN SHOW** (see SHOW.DEMO in **Listing 3**).

## MACHINE LANGUAGE CONTROL

All commands except for &NEW can be called from their respective starting addresses. Instead of being stored in an Applesoft statement, the parameters must be stored in locations $0800-$0803 in the same order as the Applesoft parameters. However, there is no easy way to call &NEW, so it would be best to write your own routine for this purpose.

## ENTERING THE PROGRAM

If you do not have an assembler, enter the Monitor with CALL −151 and type in the code to the left of the line numbers in **Listing 1**. Save it to disk with the command:

**BSAVE SHOW,A$804,L$4B9**

If you do have an assembler, enter the source code to the right of the line numbers in **Listing 1**. If you are not using the Lisa 2.5 assembler, make sure you cross-reference the pseudo-opcodes (HEX, EPZ, EQU, etc.) to ensure compatibility.

Next, the Hi-Res character set must be entered. If you have DOS Tool Kit or a program with compatible fonts, you can simply load one of the fonts into memory by typing:

**BLOAD CHAR.SET,A$D00**

Otherwise, a sample font is provided in **Listing 2**. Type this in from the Monitor and save it to disk with the command:

**BSAVE CHAR.SET,A$D00,L$300**

Finally, type in the Applesoft program in **Listing 3** and save it to disk with **SAVE SHOW.DEMO.** Since SHOW.DEMO relocates itself, be sure to save it before you run it.

For help with entering *Nibble* programs, see "A Welcome to New *Nibble* Readers" at the beginning of this issue.

## HOW THE PROGRAM WORKS

Window Show makes extensive use of the Hi-Res page; consequently, this area must be reserved for program use. This program also requires a large area of memory to be used as a stack. Starting the program at $800 (where Applesoft programs usually start) and reserving memory through the end of Hi-Res page 1 ($3FFF) fulfills these requirements. It also helps to solve a problem common to many utility programs: the lack of ProDOS compatibility. In the past, many programs written for DOS 3.3 utilized the space immediately below the DOS buffers

**TABLE 2: Supported Control Characters**

| Control Character | Function |
|---|---|
| <CTRL>A | Normal character output |
| <CTRL>J | Linefeed |
| <CTRL>L | Clears the screen and homes the cursor |
| <CTRL>M | Carriage return |
| <CTRL>Z | Forces lower-case (on the II Plus) |

**TABLE 3: Supported Escape Codes**

| Escape Code | Function |
|---|---|
| <ESC>@ | Clears the screen and homes the cursor |
| <ESC>A | Cursor right |
| <ESC>B | Cursor left |
| <ESC>C | Cursor down |
| <ESC>D | Cursor up |
| <ESC>I or up arrow | Cursor up (escape mode active) |
| <ESC>J or left arrow | Cursor left (escape mode active) |
| <ESC>K or right arrow | Cursor right (escape mode active) |
| <ESC>M or down arrow | Cursor down (escape mode active) |

and set HIMEM to protect this space. However, when ProDOS came along, this area became unreliable, at best. (The exact reason is beyond the scope of this article; for more information, see the *ProDOS Technical Reference Manual*.)

The first part of the program simply sets up the ampersand vector and returns to BASIC to wait for an &NEW command. When an &NEW command is encountered, a check is made to see which operating system (DOS 3.3 or ProDOS) is in place, and a branch is made to the appropriate routine to set up the character output and keyboard input vectors. These vectors are pointed to the routines COUT2 and KEYIN2, respectively.

COUT2 and KEYIN2 are preprocessing routines for the normal Monitor COUT1 and KEYIN1 routines. This is the same way the disk operating system handles input and output. COUT2 is a routine that echoes what would appear on the text screen, to the Hi-Res screen. It also checks for certain control codes (see **Table 2**) and handles screen scrolling; thus it is very similar to COUT in the Monitor.

KEYIN2, on the other hand, is very different. Since there is no flashing Hi-Res cursor, one must be simulated. KEYIN2 flashes this pseudo-cursor until a key is detected and then jumps to KEYIN. To conserve memory, escape codes are not fully supported. (See **Table 3** for escape codes that are supported.)

The screen is cleared to the checkered background in **line 500**. The program uses a few Applesoft subroutines to get the string for the title line and print it on the top line of the screen. If the string is shorter than 40 characters, the program fills the rest of the line with spaces; if the string is longer than 40 characters, the program truncates it at 40.

The next task is to load in the window parameters. Since three of the commands need a routine to accomplish this, a common subroutine called GETPARMS is used. GETPARMS (with some help from Applesoft) gets the window positions and checks for out-of-bounds conditions. For convenience, the resulting parameters are stored at locations $800-$803. &NEW now jumps to the &DRAW handling code and returns.

Window Show waits until the ampersand is called and checks for one of the valid commands. Input is analyzed by calling the Applesoft subroutine CHARGET, and checking for a keyword. If one is found, the program branches to the appropriate routine, and execution continues. Otherwise, the program exits through the SYNTAX ERROR routine.

Now let's examine the heart of the program — windowing. A program is often classified by its data structures. There are many types of data structures (arrays, lists, linked lists, trees, queues, sets, stacks, etc.), and specific types of programs use different data structures. (Simulations use queues, for instance.) Windowing, however, uses mainly stacks.

What is a stack? A stack is a LIFO (last in, first out) data structure that keeps track of subroutines. Think of a stack as a desk piled with papers. Unless you sort the papers, the stuff on the top gets done first. Windows function basically the same way. The top of the pile of papers is like the top window, and the pile of papers underneath is like the stack.

There are two functions associated with stacks: push and pull (sometimes called pop). These functions do exactly what their names imply. Push puts something on the stack and pull takes something off. Implementing a stack-oriented program on the Apple is not as easy as on other computers, as the 6502 has only one stack, which is limited in size to 256 bytes in a fixed location. However, it is possible to simulate a stack that is larger. Window Show uses a stack consisting of 4,096 bytes.

The subroutine OPEN (when called by &STORE) saves the current window parameters to the stack, along with the current cursor position. It then gets the new window parameters and calculates the boundaries of the portion of the Hi-Res screen that it must save. It stores the actual Hi-Res data followed by these values on the stack. Execution of this routine falls through to DRAWWIND (the entry point for &DRAW). This routine draws a window on the screen as described by the new window parameters stored at $800-$803.

This brings us to the CLOSE routine. When called by &RESTORE, CLOSE does the opposite of the OPEN routine, thus effectively restoring the screen to its condition before the call. Cursor position and window parameters are also restored.

The HOME routine clears the current window to white and jumps to the Monitor's HOME routine to complete the job.

The rest of the subroutines are fairly common and serve mostly to support the above routines.

## CUSTOMIZATION

Although Window Show's stack is adequate for most applications, a larger one could be accommodated with only slight modifications if it were placed in a different area of memory. The stack can also be shortened to make room for any improvements. The most obvious would be full escape code support, but keep in mind the memory limitations.

Another possible modification would be to use double Hi-Res graphics to make an 80-column display with windows. This would not be too hard, but would require that you rework many of the routines to switch banks of memory between character positions. Also, the cursor horizontal value is stored in a different memory location if 80 columns are active.

Use your imagination — many other improvements are possible. Next time those Mac owners show off their fancy windows, you'll have something to show *them*.

**LISTING 1: SHOW**

```
0800    ;*********************
0800    ;*                   *
0800    ;*       SHOW        *
0800    ;*  BY STEPHEN LEW   *
0800    ;* COPYRIGHT  1986   *
0800    ;*BY MICROSPARC, INC*
0800    ;*CONCORD, MA 01742 *
0800    ;*                   *
0800    ;*LISA 2.5 ASSEMBLER*
0800    ;*                   *
0800    ;*********************
0800    ;
0804    START     ORG $0804
0804              OBJ $0804
0804    ;
0804    ; ZERO PAGE LOCATIONS
0804    ;
0006    PNT       EPZ $06
0008    CAPS      EPZ $08
0019    YSAV      EPZ $19
001A    XSAV      EPZ $1A
001B    CHAR      EPZ $1B
001C    GBASL     EPZ $1C
001E    GBASH     EPZ $1E
0020    WNDLFT    EPZ $20
0021    WNDWDTH   EPZ $21
0022    WNDTOP    EPZ $22
0023    WNDBTM    EPZ $23
0024    CH        EPZ $24
0025    CV        EPZ $25
0028    BASL      EPZ $28
0029    BASH      EPZ $29
0032    INVFLG    EPZ $32
004E    RNDLO     EPZ $4E
004F    RNDHI     EPZ $4F
00A0    STRPNT    EPZ $A0
00B1    CHARGET   EPZ $B1
00B7    CHARGOT   EPZ $B7
00E6    GPAGE     EPZ $E6
00FC    TEMP      EPZ $FC
0804    ;
0804    ; INTERNAL EQUATES
```

```
0804                 ;
0800                 PAR       EQU $0800
1000                 MIN       EQU $1000
2000                 MAX       EQU $2000
0804                 ;
0804                 ; SOFT SWITCHES
0804                 ;
C000                 KEYBOARD  EQU $C000
C010                 KBDSTRB   EQU $C010
C050                 GRAPHICS  EQU $C050
C052                 FULLSCRN  EQU $C052
C057                 HIRES     EQU $C057
0804                 ;
0804                 ; ROM ROUTINES
0804                 ;
DD7B                 FRMEVL    EQU $DD7B
DEBE                 CHKCOM    EQU $DEBE
DEC9                 SYNERR    EQU $DEC9
E199                 ILLEGAL   EQU $E199
E6F8                 GETBYTE1  EQU $E6F8
E74C                 GETBYTE   EQU $E74C
FB2F                 INITTEXT  EQU $FB2F
FC22                 VTAB      EQU $FC22
FC58                 HOME      EQU $FC58
FD1B                 KEYIN     EQU $FD1B
FDF0                 COUT1     EQU $FDF0
0804                 ;
0804                 ;*****************
0804                 ;*               *
0804                 ;*  WINDOW SYSTEM  *
0804                 ;*               *
0804                 ;*****************
0804                 ;
0804 A9 4C           BEGIN     LDA #$4C
0806 8D F5 03                  STA $03F5
0809 A9 AF                     LDA #FIRST
080B 8D F6 03                  STA $03F6
080E A9 0B                     LDA /FIRST
0810 8D F7 03                  STA $03F7
0813 60                        RTS
0814                 ;
0814                 ; KEYIN2 VECTOR
0814                 ;
0814 91 28           KEYIN2    STA (BASL),Y
0816 86 1A                     STX XSAV
0818 A4 24           KEYIN3    LDY CH
081A A5 28                     LDA BASL
081C 85 1C                     STA GBASL
081E A5 29                     LDA BASH
0820 09 3C                     ORA #$3C           ; CALC GRAPHBASE
0822 85 1D                     STA GBASL+1
0824 A2 01                     LDX #$01
0826 B1 1C                     LDA (GBASL),Y
0828 48                        PHA
0829 E6 4E           RNDCNT    INC RNDLO          ; INC RND NUMBER
082B D0 0B                     BNE NOFLIP
082D E6 4F                     INC RNDHI
082F CA                        DEX
0830 D0 06                     BNE NOFLIP
0832 49 7F                     EOR #$7F           ; FLIP CURSOR
0834 91 1C                     STA (GBASL),Y
0836 A2 50                     LDX #$50
0838 2C 00 C0        NOFLIP    BIT KEYBOARD       ; KEY DOWN?
083B 10 EC                     BPL RNDCNT
083D 68                        PLA
083E 91 1C                     STA (GBASL),Y
0840 AD 00 C0                  LDA KEYBOARD
0843 C9 9B                     CMP #$9B           ; IS IT ESCAPE?
0845 D0 09                     BNE NOTESC
0847 2C 10 C0                  BIT KBDSTRB
084A 20 94 0C                  JSR ESCAPE
084D 4C 18 08                  JMP KEYIN3
0850 B1 28           NOTESC    LDA (BASL),Y
0852 A6 1A                     LDX XSAV
0854 4C 1B FD                  JMP KEYIN
0857 18              DISPLAY   CLC                ; DISPLAY CHAR
0858 A5 28                     LDA BASL           ; ON SCREEN
085A 65 24                     ADC CH
085C 85 1C                     STA GBASL
085E A5 29                     LDA BASH           ; CALC BASADDR
0860 69 1C                     ADC #$1C
0862 85 1D                     STA GBASL+1
0864 A5 1B                     LDA CHAR
0866 29 40                     AND #$40
0868 F0 10                     BEQ NUMB+1
086A A5 08                     LDA CAPS           ; FORCE LCASE?
086C D0 09                     BNE LCASE+1
086E A5 1B                     LDA CHAR
0870 29 20                     AND #$20
0872 D0 03                     BNE LCASE+1
0874 A9 0E                     LDA #$0E
0876 2C A9 0F        LCASE     BIT $0FA9
0879 2C A9 0D        NUMB      BIT $0DA9
087C 85 1F                     STA GBASH+1
087E A5 1B                     LDA CHAR
0880 0A                        ASL
0881 0A                        ASL
0882 0A                        ASL
0883 85 1E                     STA GBASH
0885 A2 07                     LDX #$07
0887 A0 00                     LDY #$00
0889 B1 1E           NLIN      LDA (GBASH),Y      ; GET LINE
088B 45 FC                     EOR TEMP           ; INVERT
088D 91 1C                     STA (GBASL),Y      ; STORE LINE
088F E6 1E                     INC GBASH
0891 18                        CLC
0892 A5 1D                     LDA GBASL+1
0894 18                        CLC
0895 69 04                     ADC #$04
0897 85 1D                     STA GBASL+1
0899 CA                        DEX
089A 10 ED                     BPL NLIN
089C 60                        RTS
089D                 ;
089D                 ; COUT VECTOR
089D                 ;
089D 85 1B           COUT2     STA CHAR           ; SAVE REG
089F 84 19                     STY YSAV
08A1 86 1A                     STX XSAV
08A3 A4 32                     LDY INVFLG         ; SET INVERSE
08A5 C0 FF                     CPY #$FF
08A7 F0 02                     BEQ FLIP
08A9 A0 00                     LDY #$00
08AB 84 FC           FLIP      STY TEMP
08AD A5 1B                     LDA CHAR
08AF 29 7F                     AND #$7F
08B1 C9 20                     CMP #$20           ; PRINT IF NOT
08B3 B0 28                     BCS STORE          ; CTRL CHAR
08B5 C9 01                     CMP #$01           ; CTRL-A
08B7 D0 04                     BNE NOTOG
08B9 A9 00                     LDA #$00           ; SET CAPS
08BB 85 08                     STA CAPS
08BD C9 1A           NOTOG     CMP #$1A           ; CTRL-Z
08BF D0 04                     BNE NOTOG2
08C1 A9 FF                     LDA #$FF           ; SET LCASE
08C3 85 08                     STA CAPS
08C5 C9 0D           NOTOG2    CMP #$0D           ; RETURN
08C7 F0 1E                     BEQ CR
08C9 C9 0A                     CMP #$0A           ; LINE FEED
08CB F0 1A                     BEQ CR
08CD C9 0C                     CMP #$0C           ; CTRL-L
08CF D0 03                     BNE OUT
08D1 20 53 0A                  JSR HOME1
08D4 A4 19           OUT       LDY YSAV           ; RESTORE REG
08D6 A6 1A                     LDX XSAV           ; AND EXIT
08D8 A5 1B                     LDA CHAR           ; THROUGH COUT1
08DA 4C F0 FD                  JMP COUT1
08DD 20 57 08        STORE     JSR DISPLAY        ; DISPLAY CHAR
08E0 A4 24                     LDY CH
08E2 C8                        INY                ; NEXT WILL BE
08E3 C4 21                     CPY WNDWDTH        ; ON NEXT LINE?
08E5 90 ED                     BCC OUT
08E7 A4 25           CR        LDY CV
08E9 C8                        INY                ; ON BOTTOM OF
08EA C4 23                     CPY WNDBTM         ; SCREEN?
08EC 90 E6                     BCC OUT            ; NO, THEN OUT
08EE A5 20           SCROLL    LDA WNDLFT         ; YES, THEN
08F0 85 FC                     STA TEMP           ; SCROLL SCREEN
08F2 A5 23                     LDA WNDBTM         ; SET UP PARAMS
08F4 0A                        ASL                ; FOR SCROLL
08F5 0A                        ASL
08F6 0A                        ASL
08F7 85 FF                     STA TEMP+3
08F9 A5 22                     LDA WNDTOP
08FB 0A                        ASL
08FC 0A                        ASL
08FD 0A                        ASL
08FE 85 FE                     STA TEMP+2
0900 69 08                     ADC #08
0902 85 FD                     STA TEMP+1
0904 20 A1 0A        SCRL1     JSR CVTOBAS        ; CALC BASADDR
0907 A5 1C                     LDA GBASL          ; SWITCH TO
0909 85 1E                     STA GBASH          ; STORE ADDR
090B A5 1D                     LDA GBASL+1
090D 85 1F                     STA GBASH+1
090F A4 FE                     LDY TEMP+2
0911 A5 FD                     LDA TEMP+1
0913 85 FE                     STA TEMP+2
0915 20 A1 0A                  JSR CVTOBAS        ; CALC BASADDR
0918 A5 FE                     LDA TEMP+2
091A 85 FD                     STA TEMP+1
091C 84 FE                     STY TEMP+2
091E A0 00                     LDY #$00
0920 B1 1C           SCRL2     LDA (GBASL),Y      ; SCROLL UP
0922 91 1E                     STA (GBASH),Y
0924 C8                        INY
0925 C4 21                     CPY WNDWDTH        ; END OF LINE?
0927 90 F7                     BCC SCRL2
0929 E6 FE                     INC TEMP+2
092B E6 FD                     INC TEMP+1
092D A5 FD                     LDA TEMP+1
092F C5 FF                     CMP TEMP+3         ; LAST LINE?
```

```
0931 B0 02              BCS SCRL3
0933 90 CF              BCC SCRL1
0935 20 A1 0A  SCRL3    JSR CVTOBAS        ; CLEAR LAST
0938 A9 7F              LDA #$7F           ; LINE
093A A0 00              LDY #$00
093C 91 1C     SCRL4    STA (GBASL),Y
093E C8                 INY
093F C4 21              CPY WNDWDTH
0941 90 F9              BCC SCRL4
0943 E6 FE              INC TEMP+2
0945 A5 FE              LDA TEMP+2
0947 C5 FF              CMP TEMP+3
0949 90 EA              BCC SCRL3
094B 4C D4 08           JMP OUT
094E           ;
094E           ; CLOSE OUTPUT WINDOW
094E           ;
094E A5 07     CLOSE    LDA PNT+1
0950 C9 1F              CMP /MAX-1
0952 D0 07              BNE CLOSE1
0954 A5 06              LDA PNT
0956 C9 FF              CMP #$FF
0958 D0 01              BNE CLOSE1
095A 60                 RTS
095B A2 00     CLOSE1   LDX #$00           ; PULL PARAMS
095D 20 EA 0A  LOOPA1   JSR PULL           ; OFF OF STACK
0960 9D 00 08           STA PAR,X
0963 E8                 INX
0964 E0 04              CPX #$04
0966 D0 F5              BNE LOOPA1
0968 A2 00              LDX #$00
096A 20 EA 0A  LOOPA2   JSR PULL
096D 95 20              STA WNDLFT,X
096F E8                 INX
0970 E0 0A              CPX #$0A
0972 D0 F6              BNE LOOPA2
0974 20 7D 0A           JSR CONVERT
0977 A5 FE              LDA TEMP+2
0979 A4 FF              LDY TEMP+3
097B A6 FD              LDX TEMP+1
097D CA                 DEX
097E 86 FD              STX TEMP+1
0980 88                 DEY
0981 85 FF              STA TEMP+3
0983 84 FE              STY TEMP+2
0985 20 A1 0A  AGAINA   JSR CVTOBAS        ; PULL SAVED
0988 A4 FD              LDY TEMP+1         ; SCREEN OFF
098A CA                 DEX                ; STACK
098B 20 EA 0A  LOOPB    JSR PULL
098E 91 1C              STA (GBASL),Y
0990 88                 DEY
0991 C0 FF              CPY #$FF
0993 D0 F6              BNE LOOPB
0995 C6 FE              DEC TEMP+2
0997 A5 FE              LDA TEMP+2
0999 C5 FF              CMP TEMP+3
099B B0 E8              BCS AGAINA
099D 60                 RTS
099E           ;
099E           ; OPEN OUTPUT WINDOW
099E           ;
099E A5 06     OPEN     LDA PNT
09A0 8D FC 0A           STA ERRLOC+1
09A3 A5 07              LDA PNT+1
09A5 8D 00 0B           STA ERRLOC+5
09A8 20 7D 0A  OPEN1    JSR CONVERT        ; SAVE SCREEN TO
09AB 20 A1 0A  AGAINB   JSR CVTOBAS        ; STACK
09AE A0 00              LDY #$00
09B0 B1 1C     LOOPC    LDA (GBASL),Y
09B2 20 D1 0A           JSR PUSH
09B5 C8                 INY
09B6 C4 FD              CPY TEMP+1
09B8 90 F6              BCC LOOPC
09BA E6 FE              INC TEMP+2
09BC A5 FE              LDA TEMP+2
09BE C5 FF              CMP TEMP+3
09C0 90 E9              BCC AGAINB
09C2 A2 09              LDX #$09
09C4 B5 20     LOOPD2   LDA WNDLFT,X       ; SAVE PARAMS TO
09C6 20 D1 0A           JSR PUSH           ; STACK
09C9 CA                 DEX
09CA 10 F8              BPL LOOPD2
09CC A2 03              LDX #$03
09CE BD 00 08  LOOPD1   LDA PAR,X
09D1 20 D1 0A           JSR PUSH
09D4 CA                 DEX
09D5 10 F7              BPL LOOPD1
09D7           ;
09D7           ; CLEAR OUTPUT WINDOW
09D7           ;
09D7 A2 03     CLEAR    LDX #$03           ; SET FOR WINDOW
09D9 BD 00 08  LOOPE    LDA PAR,X
09DC 95 20              STA WNDLFT,X
09DE CA                 DEX
09DF 10 F8              BPL LOOPE
09E1 20 7D 0A           JSR CONVERT
09E4 C6 FD              DEC TEMP+1         ; FIX PARAMS
09E6 C6 FF              DEC TEMP+3
09E8 C6 FF              DEC TEMP+3
09EA 20 A1 0A           JSR CVTOBAS        ; TOP LINE
09ED A0 00              LDY #$00
09EF B1 1C              LDA (GBASL),Y
09F1 29 1F              AND #$1F
09F3 91 1C              STA (GBASL),Y
09F5 98                 TYA
09F6 C8                 INY
09F7 91 1C     LOOPF    STA (GBASL),Y
09F9 C8                 INY
09FA C4 FD              CPY TEMP+1
09FC 90 F9              BCC LOOPF
09FE B1 1C              LDA (GBASL),Y
0A00 29 7E              AND #$7E
0A02 91 1C              STA (GBASL),Y
0A04 E6 FE              INC TEMP+2
0A06 20 A1 0A  AGAINC   JSR CVTOBAS        ; MIDDLE LINES
0A09 A0 00              LDY #$00
0A0B B1 1C              LDA (GBASL),Y
0A0D 29 1F              AND #$1F
0A0F 09 40              ORA #$40
0A11 91 1C              STA (GBASL),Y
0A13 C8                 INY
0A14 A9 7F              LDA #$7F
0A16 91 1C     LOOPG    STA (GBASL),Y
0A18 C8                 INY
0A19 C4 FD              CPY TEMP+1
0A1B 90 F9              BCC LOOPG
0A1D B1 1C              LDA (GBASL),Y
0A1F 29 7C              AND #$7C
0A21 91 1C              STA (GBASL),Y
0A23 E6 FE              INC TEMP+2
0A25 A5 FE              LDA TEMP+2
0A27 C5 FF              CMP TEMP+3
0A29 90 DB              BCC AGAINC
0A2B A2 01              LDX #$01
0A2D 86 FF              STX TEMP+3
0A2F 20 A1 0A  AGAIND   JSR CVTOBAS        ; BOTTOM LINES
0A32 A0 00              LDY #$00
0A34 A5 FF              LDA TEMP+3
0A36 F0 06              BEQ SKIP
0A38 B1 1C              LDA (GBASL),Y
0A3A 29 1F              AND #$1F
0A3C 91 1C              STA (GBASL),Y
0A3E 98        SKIP     TYA
0A3F C8                 INY
0A40 91 1C     LOOPH    STA (GBASL),Y
0A42 C8                 INY
0A43 C4 FD              CPY TEMP+1
0A45 90 F9              BCC LOOPH
0A47 B1 1C              LDA (GBASL),Y
0A49 29 7C              AND #$7C
0A4B 91 1C              STA (GBASL),Y
0A4D E6 FE              INC TEMP+2
0A4F C6 FF              DEC TEMP+3
0A51 10 DC              BPL AGAIND
0A53           ;
0A53           ; CLEAR WINDOW
0A53           ;
0A53 A5 20     HOME1    LDA WNDLFT
0A55 85 FC              STA TEMP
0A57 A5 22              LDA WNDTOP
0A59 0A                 ASL
0A5A 0A                 ASL
0A5B 0A                 ASL
0A5C 85 FE              STA TEMP+2
0A5E A5 23              LDA WNDBTM
0A60 0A                 ASL
0A61 0A                 ASL
0A62 0A                 ASL
0A63 85 FF              STA TEMP+3
0A65 20 A1 0A  NLN1     JSR CVTOBAS
0A68 A9 7F              LDA #$7F
0A6A A4 21              LDY WNDWDTH
0A6C 88                 DEY
0A6D 91 1C     NLN2     STA (GBASL),Y
0A6F 88                 DEY
0A70 10 FB              BPL NLN2
0A72 E6 FE              INC TEMP+2
0A74 A5 FE              LDA TEMP+2
0A76 C5 FF              CMP TEMP+3
0A78 90 EB              BCC NLN1
0A7A 4C 58 FC           JMP HOME
0A7D AE 00 08  CONVERT  LDX PAR            ; CONVERT PARAMS
0A80 CA                 DEX                ; TO GRAPHICS
0A81 86 FC              STX TEMP           ; LOCATIONS, AND
0A83 AE 01 08           LDX PAR+1          ; SET FOR BORDER
0A86 E8                 INX
0A87 E8                 INX
0A88 86 FD              STX TEMP+1
0A8A AD 02 08           LDA PAR+2
0A8D 0A                 ASL
0A8E 0A                 ASL
0A8F 0A                 ASL
```

```
0A90 AA                 TAX
0A91 CA                 DEX
0A92 CA                 DEX
0A93 86 FE              STX TEMP+2
0A95 AD 03 08           LDA PAR+3
0A98 0A                 ASL
0A99 0A                 ASL
0A9A 0A                 ASL
0A9B AA                 TAX
0A9C E8                 INX
0A9D E8                 INX
0A9E 86 FF              STX TEMP+3
0AA0 60                 RTS
0AA1 A5 FE     CVTOBAS  LDA TEMP+2         ; CALCULATE BASE
0AA3 0A                 ASL                ; LOCATIONS FOR
0AA4 0A                 ASL                ; HIRES GRAPHICS
0AA5 29 1C              AND #$1C
0AA7 85 1D              STA GBASL+1
0AA9 A5 FE              LDA TEMP+2
0AAB 4A                 LSR
0AAC 4A                 LSR
0AAD 4A                 LSR
0AAE 4A                 LSR
0AAF 29 03              AND #$03
0AB1 05 1D              ORA GBASL+1
0AB3 05 E6              ORA GPAGE
0AB5 85 1D              STA GBASL+1
0AB7 A9 00              LDA #$00
0AB9 90 02              BCC SKIP2
0ABB 69 7F              ADC #$7F
0ABD 85 4E     SKIP2    STA RNDLO
0ABF A5 FE              LDA TEMP+2
0AC1 4A                 LSR
0AC2 29 60              AND #$60
0AC4 85 1C              STA GBASL
0AC6 4A                 LSR
0AC7 4A                 LSR
0AC8 05 4E              ORA RNDLO
0ACA 05 1C              ORA GBASL
0ACC 65 FC              ADC TEMP
0ACE 85 1C              STA GBASL
0AD0 60                 RTS
0AD1 84 19     PUSH     STY YSAV           ; PUSH VALUE TO
0AD3 A0 00              LDY #$00           ; STACK
0AD5 91 06              STA (PNT),Y
0AD7 C6 06              DEC PNT
0AD9 A5 06              LDA PNT
0ADB C9 FF              CMP #$FF
0ADD D0 08              BNE SKIP3
0ADF C6 07              DEC PNT+1
0AE1 A5 07              LDA PNT+1
0AE3 C9 10              CMP /MIN
0AE5 F0 12              BEQ ERROR
0AE7 A4 19     SKIP3    LDY YSAV
0AE9 60                 RTS
0AEA 84 19     PULL     STY YSAV           ; PULL VALUE
0AEC E6 06              INC PNT            ; FROM STACK
0AEE D0 02              BNE SKIP4
0AF0 E6 07              INC PNT+1
0AF2 A0 00     SKIP4    LDY #$00
0AF4 B1 06              LDA (PNT),Y
0AF6 A4 19              LDY YSAV
0AF8 60                 RTS
0AF9 68        ERROR    PLA                ; STACK ERROR!
0AFA 68                 PLA
0AFB A9 00     ERRLOC   LDA #$00
0AFD 85 06              STA PNT
0AFF A9 00              LDA #$00
0B01 85 07              STA PNT+1
0B03 60                 RTS
0B04           ;
0B04           ; FILL SCREEN WITH BACKGROUND
0B04           ;
0B04 A9 20     CLS      LDA #$20           ; SET TO TOP OF
0B06 85 07              STA PNT+1          ; PAGE
0B08 A9 00              LDA #$00
0B0A 85 06              STA PNT
0B0C AA                 TAX
0B0D A8                 TAY
0B0E BD 2D 0B  NEXT     LDA TAB,X          ; DRAW CHECKER
0B11 91 06              STA (PNT),Y        ; PATTERN
0B13 C8                 INY
0B14 BD 2E 0B           LDA TAB+1,X
0B17 91 06              STA (PNT),Y
0B19 C8                 INY
0B1A D0 F2              BNE NEXT
0B1C E6 07              INC PNT+1
0B1E A5 07              LDA PNT+1
0B20 C9 40              CMP #$40
0B22 F0 08              BEQ SCRNCLR
0B24 29 04              AND #$04
0B26 18                 CLC
0B27 6A                 ROR
0B28 AA                 TAX
0B29 4C 0E 0B           JMP NEXT
0B2C 60        SCRNCLR  RTS
0B2D 2A 55 55  TAB      HEX 2A55552A
0B30 2A
```

```
0B31           ;
0B31           ; INITIALIZE WINDOW SYSTEM
0B31           ;
0B31 20 2F FB  INIT     JSR INITTEXT
0B34 20 58 FC           JSR HOME
0B37 20 4F 0B           JSR SWCHS
0B3A A9 20              LDA #$20
0B3C 85 E6              STA GPAGE
0B3E A9 00              LDA #$00
0B40 85 08              STA CAPS
0B42 20 04 0B           JSR CLS
0B45 2C 57 C0           BIT HIRES
0B48 2C 52 C0           BIT FULLSCRN
0B4B 2C 50 C0           BIT GRAPHICS
0B4E 60                 RTS
0B4F A9 4C     SWCHS    LDA #$4C           ; SET AMPERSAND
0B51 8D F5 03           STA $03F5          ; VECTOR
0B54 A9 8D              LDA #AMP
0B56 8D F6 03           STA $03F6
0B59 A9 0B              LDA /AMP
0B5B 8D F7 03           STA $03F7
0B5E AD D2 03           LDA $03D2          ; LOAD TEST BYTE
0B61 C9 BE              CMP #$BE           ; PRODOS?
0B63 D0 15              BNE DOS            ;NO SET UP DOS
0B65 A9 9D              LDA #COUT2
0B67 8D 30 BE           STA $BE30
0B6A A9 08              LDA /COUT2
0B6C 8D 31 BE           STA $BE31
0B6F A9 14              LDA #KEYIN2
0B71 8D 32 BE           STA $BE32
0B74 A9 08              LDA /KEYIN2
0B76 8D 33 BE           STA $BE33
0B79 60                 RTS
0B7A A9 9D     DOS      LDA #COUT2         ;DOS SET UP
0B7C 85 36              STA $36
0B7E A9 08              LDA /COUT2
0B80 85 37              STA $37
0B82 A9 14              LDA #KEYIN2
0B84 85 38              STA $38
0B86 A9 08              LDA /KEYIN2
0B88 85 39              STA $39
0B8A 4C EA 03           JMP $03EA
0B8D C9 97     AMP      CMP #$97           ; HOME?
0B8F D0 09              BNE NOTHOME
0B91 20 B1 00           JSR CHARGET
0B94 20 53 0A           JSR HOME1
0B97 4C B7 00           JMP CHARGOT
0B9A C9 AE     NOTHOME  CMP #$AE           ; RESTORE?
0B9C D0 09              BNE NORESTOR
0B9E 20 B1 00           JSR CHARGET
0BA1 20 4E 09           JSR CLOSE
0BA4 4C B7 00           JMP CHARGOT
0BA7 C9 94     NORESTOR CMP #$94           ; DRAW?
0BA9 F0 0B              BEQ DRAW
0BAB C9 A8              CMP #$A8           ; STORE?
0BAD F0 13              BEQ SAVWIND
0BAF C9 BF     FIRST    CMP #$BF           ; NEW?
0BB1 F0 1B              BEQ NEW
0BB3 4C C9 DE           JMP SYNERR
0BB6 20 B1 00  DRAW     JSR CHARGET
0BB9 20 0F 0C  DRAW1    JSR GETPAR
0BBC 20 D7 09           JSR CLEAR
0BBF 4C B7 00           JMP CHARGOT
0BC2 20 B1 00  SAVWIND  JSR CHARGET
0BC5 20 0F 0C           JSR GETPAR
0BC8 20 9E 09           JSR OPEN
0BCB 4C B7 00           JMP CHARGOT
0BCE 20 B1 00  NEW      JSR CHARGET
0BD1 20 31 0B           JSR INIT
0BD4 20 7B DD           JSR FRMEVL
0BD7 A0 02              LDY #$02
0BD9 B1 A0              LDA (STRPNT),Y
0BDB 85 07              STA PNT+1
0BDD 88                 DEY
0BDE B1 A0              LDA (STRPNT),Y
0BE0 85 06              STA PNT
0BE2 88                 DEY
0BE3 B1 A0              LDA (STRPNT),Y
0BE5 85 4E              STA RNDLO
0BE7 F0 0E              BEQ NULL           ; NULL STRING
0BE9 B1 06     NEXTCHAR LDA (PNT),Y
0BEB 20 9D 08           JSR COUT2
0BEE C8                 INY
0BEF C0 28              CPY #$28
0BF1 F0 0E              BEQ COMMA
0BF3 C4 4E              CPY RNDLO
0BF5 D0 F2              BNE NEXTCHAR
0BF7 A9 20     NULL     LDA #$20
0BF9 20 9D 08  NEXTSPC  JSR COUT2
0BFC C8                 INY
0BFD C0 28              CPY #$28
0BFF D0 F8              BNE NEXTSPC
0C01 20 BE DE  COMMA    JSR CHKCOM
```

```
0C04 A9 FF               LDA #$FF           ; SET STACK TO
0C06 85 06               STA PNT            ; BOTTOM
0C08 A9 1F               LDA /MAX-1
0C0A 85 07               STA PNT+1
0C0C 4C B9 0B            JMP DRAW1
0C0F 20 F8 E6  GETPAR    JSR GETBYTE1       ; GET PARAMS AND
0C12 8E 00 08            STX PAR
0C15 E0 25               CPX #$25           ; CHECK FOR ERR
0C17 B0 24               BCS ILQUANT
0C19 20 4C E7            JSR GETBYTE
0C1C 8E 01 08            STX PAR+1
0C1F 8A                  TXA
0C20 18                  CLC
0C21 6D 00 08            ADC PAR
0C24 C9 28               CMP #$28
0C26 B0 15               BCS ILQUANT
0C28 20 4C E7            JSR GETBYTE
0C2B 8E 02 08            STX PAR+2
0C2E E0 01               CPX #$01
0C30 90 0B               BCC ILQUANT
0C32 20 4C E7            JSR GETBYTE
0C35 8E 03 08            STX PAR+3
0C38 E0 18               CPX #$18
0C3A B0 01               BCS ILQUANT
0C3C 60                  RTS
0C3D 4C 99 E1  ILQUANT   JMP ILLEGAL
0C40 A5 28     KEYIN4    LDA BASL
0C42 85 1C               STA GBASL
0C44 A5 29               LDA BASH
0C46 09 3C               ORA #$3C
0C48 85 1D               STA GBASL+1
0C4A A4 24               LDY CH
0C4C B1 1C               LDA (GBASL),Y
0C4E 49 7F               EOR #$7F
0C50 91 1C               STA (GBASL),Y
0C52 2C 00 C0  NOKEY     BIT KEYBOARD
0C55 10 FB               BPL NOKEY
0C57 49 7F               EOR #$7F
0C59 91 1C               STA (GBASL),Y
0C5B AD 00 C0            LDA KEYBOARD
0C5E 2C 10 C0            BIT KBDSTRB
0C61 60                  RTS
0C62 E6 24     ADVANCE   INC CH
0C64 A5 24               LDA CH
0C66 C5 21               CMP WNDWDTH
0C68 90 18               BCC SAMELIN
0C6A 4C E7 08            JMP CR
0C6D C6 24     BS        DEC CH
0C6F 10 11               BPL SAMELIN
0C71 A5 21               LDA WNDWDTH
0C73 85 24               STA CH
0C75 C6 24               DEC CH
0C77 A5 22     UP        LDA WNDTOP
0C79 C5 25               CMP CV
0C7B B0 05               BCS SAMELIN
0C7D C6 25               DEC CV
0C7F 4C 22 FC            JMP VTAB
0C82 60        SAMELIN   RTS
0C83 CB CA CD  TABLE     HEX CBCACDC995888A8B
0C86 C9 95 88
0C89 8A 8B
0C8B 98        ESCON     TYA
0C8C 29 03               AND #$03
0C8E 18                  CLC
0C8F 69 C1               ADC #$C1
0C91 20 A3 0C            JSR ESCDO
0C94 20 40 0C  ESCAPE    JSR KEYIN4
0C97 85 1B               STA CHAR
0C99 A0 07               LDY #$07
0C9B D9 83 0C  NEXTCODE  CMP TABLE,Y
0C9E F0 EB               BEQ ESCON
0CA0 88                  DEY
0CA1 10 F8               BPL NEXTCODE
0CA3 38        ESCDO     SEC
0CA4 49 C0               EOR #$C0
0CA6 D0 03               BNE NOTAT
0CA8 4C 53 0A            JMP HOME1
0CAB 69 FD     NOTAT     ADC #$FD
0CAD 90 B3               BCC ADVANCE
0CAF F0 BC               BEQ BS
0CB1 69 FD               ADC #$FD
0CB3 F0 C2               BEQ UP
0CB5 B0 05               BCS NOCODE
0CB7 A9 8A               LDA #$8A
0CB9 4C 9D 08            JMP COUT2
0CBC 60        NOCODE    RTS
0CBD                     END

***** END OF ASSEMBLY
```

END OF LISTING 1

```
              KEY PERFECT 5.0
                  RUN ON
                   SHOW
=========================================
CODE-5.0    ADDR# - ADDR#     CODE-4.0
--------    -------------     --------
0D66683D    0804 - 0853         248F
37DB475E    0854 - 08A3         226E
0F2E3C00    08A4 - 08F3         28DB
18A13199    08F4 - 0943         2640
A0CDCF6D    0944 - 0993         29A9
98FE06BB    0994 - 09E3         2432
1CF494AB    09E4 - 0A33         29ED
DC44E1B5    0A34 - 0A83         277F
03DE10F1    0A84 - 0AD3         26A5
8F2E3E6C    0AD4 - 0B23         273B
F67441A1    0B24 - 0B73         222B
23F5A389    0B74 - 0BC3         266D
9F95F3F8    0BC4 - 0C13         2B99
19429375    0C14 - 0C63         26AB
476F654D    0C64 - 0CB3         2569
FCF62E6F    0CB4 - 0CBC         048F
092577A3 = PROGRAM TOTAL =      04B9
```

## LISTING 2: CHAR.SET

```
0D00- 00 00 00 00 00 00 00 00
0D08- 00 08 08 08 08 00 08 00
0D10- 00 14 14 00 00 00 00 00
0D18- 00 14 3E 14 14 3E 14 00
0D20- 00 1C 0A 1C 28 1C 08 00
0D28- 00 24 1A 0C 18 2C 12 00
0D30- 00 04 0A 04 2A 12 2C 00
0D38- 00 08 08 00 00 00 00 00
0D40- 00 08 04 04 04 04 08 00
0D48- 00 08 10 10 10 10 08 00
0D50- 00 08 2A 1C 1C 2A 08 00
0D58- 00 00 08 08 3E 08 08 00
0D60- 00 00 00 00 00 08 08 04
0D68- 00 00 00 00 3E 00 00 00
0D70- 00 00 00 00 00 00 08 00
0D78- 00 00 20 10 08 04 02 00
0D80- 00 1C 32 2A 2A 26 1C 00
0D88- 00 08 0C 08 08 08 1C 00
0D90- 00 1C 22 20 1C 02 3E 00
0D98- 00 3E 10 08 10 22 1C 00
0DA0- 00 22 22 22 3E 20 20 00
0DA8- 00 3E 02 1E 20 20 1E 00
0DB0- 00 1C 02 1E 22 22 1C 00
0DB8- 00 3E 20 10 10 08 08 00
0DC0- 00 1C 22 1C 22 22 1C 00
0DC8- 00 1C 22 22 3C 20 1C 00
0DD0- 00 00 00 08 00 08 00 00
0DD8- 00 00 00 08 00 08 08 04
0DE0- 00 00 10 08 04 08 10 00
0DE8- 00 00 00 3E 00 3E 00 00
0DF0- 00 00 04 08 10 08 04 00
0DF8- 00 1C 22 10 08 00 08 00
0E00- 00 1C 22 3A 1A 02 3C 00
0E08- 00 1C 22 22 3E 22 22 00
0E10- 00 1E 22 1E 22 22 1E 00
0E18- 00 1C 22 02 02 22 1C 00
0E20- 00 1E 22 22 22 22 1E 00
0E28- 00 3E 02 1E 02 02 3E 00
0E30- 00 3E 02 1E 02 02 02 00
0E38- 00 1C 22 02 32 22 3C 00
0E40- 00 22 22 3E 22 22 22 00
0E48- 00 1C 08 08 08 08 1C 00
0E50- 00 20 20 20 20 22 1C 00
0E58- 00 22 12 0A 0E 12 22 00
0E60- 00 02 02 02 02 02 3E 00
0E68- 00 22 36 2A 2A 22 22 00
0E70- 00 22 26 2A 32 22 22 00
0E78- 00 1C 22 22 22 22 1C 00
0E80- 00 1E 22 22 1E 02 02 00
0E88- 00 1C 22 22 2A 12 2C 00
0E90- 00 1E 22 22 1E 12 22 00
0E98- 00 3C 02 1C 20 22 1C 00
0EA0- 00 3E 08 08 08 08 08 00
```

**LISTING 2: CHAR.SET** *(continued)*

```
0EA8- 00 22 22 22 22 22 1C 00
0EB0- 00 22 22 22 14 14 08 00
0EB8- 00 22 22 2A 2A 36 22 00
0EC0- 00 22 14 08 08 14 22 00
0EC8- 00 22 22 14 08 08 08 00
0ED0- 00 3E 20 18 0C 02 3E 00
0ED8- 00 3E 06 06 06 06 3E 00
0EE0- 00 00 02 04 08 10 20 00
0EE8- 00 3E 30 30 30 30 3E 00
0EF0- 00 00 08 14 22 00 00 00
0EF8- 00 00 00 00 00 00 00 7F
0F00- 10 08 77 1F 1F 7F 36 00
0F08- 00 00 00 3C 22 22 3C 00
0F10- 00 02 02 1E 22 22 1E 00
0F18- 00 00 00 3C 02 02 3C 00
0F20- 00 20 20 3C 22 22 3C 00
0F28- 00 00 00 1C 12 0A 3C 00
0F30- 00 18 24 04 1E 04 04 00
0F38- 00 00 00 1C 22 3C 20 1C
0F40- 00 02 02 1E 22 22 22 00
0F48- 00 08 00 0C 08 08 1C 00
0F50- 00 10 00 10 10 10 12 0C
0F58- 00 02 02 32 0A 16 22 00
0F60- 00 0C 08 08 08 08 1C 00
0F68- 00 00 00 36 2A 2A 22 00
0F70- 00 00 00 1C 22 22 22 00
0F78- 00 00 00 1C 22 22 1C 00
0F80- 00 00 00 1E 22 22 1E 02
0F88- 00 00 00 3C 22 22 3C 20
0F90- 00 00 00 3A 06 02 02 00
0F98- 00 00 00 1C 04 08 0E 00
0FA0- 00 04 04 1E 04 24 18 00
0FA8- 00 00 00 22 22 22 3C 00
0FB0- 00 00 00 22 22 14 08 00
0FB8- 00 00 00 22 2A 2A 36 00
0FC0- 00 00 00 12 0C 0C 12 00
0FC8- 00 00 00 22 22 3C 20 1C
0FD0- 00 00 00 3E 30 0C 3E 00
0FD8- 00 18 08 0C 0C 08 18 00
0FE0- 00 08 08 08 08 08 08 00
0FE8- 00 0C 08 18 18 08 0C 00
0FF0- 00 2C 1A 00 00 00 00 00
0FF8- 00 2A 14 2A 14 2A 14 00
```

END OF LISTING 2

```
          KEY PERFECT 5.0
              RUN ON
             CHAR.SET
====================================
CODE-5.0    ADDR# - ADDR#   CODE-4.0
--------    -------------   --------
80DE32A9     0D00 - 0D4F      2554
AA12F088     0D50 - 0D9F      272D
12F929F7     0DA0 - 0DEF      2D02
DEC240F1     0DF0 - 0E3F      26ED
D299E49F     0E40 - 0E8F      266C
504C0669     0E90 - 0EDF      23D8
1B5D0C92     0EE0 - 0F2F      2623
05F50F0A     0F30 - 0F7F      27EA
8D0ED7AE     0F80 - 0FCF      23AC
EE65751D     0FD0 - 0FFF      1720
3417AF88 = PROGRAM TOTAL =    0300
```

**LISTING 3: SHOW.DEMO**

```
10  REM ***********************
20  REM * SHOW.DEMO           *
30  REM * BY STEPHEN LEW      *
40  REM * COPYRIGHT (C) 1986  *
50  REM * BY MICROSPARC, INC  *
60  REM * CONCORD, MA  01742  *
70  REM ***********************
80  IF  PEEK (104) <  > 64 THEN  POKE 103,1: POKE
     104,64: POKE 16384,0: PRINT  CHR$ (4)"RU
     NSHOW.DEMO"
90  PRINT  CHR$ (4)"BLOADCHAR.SET"
100  PRINT  CHR$ (4)"BRUNSHOW"
110  &  NEW " STANDARD HIRES OUTPUT WINDOWS (
     SHOW)",1,38,2,23
120  POKE 32,2: POKE 33,36: POKE 34,3: POKE 3
     5,22: &  HOME
130  VTAB 22: PRINT "S.H.O.W." TAB( 14)"WRITT
     EN BY STEPHEN LEW"
140  PRINT "COPYRIGHT 1986 BY MICROSPARC, INC
     "
150  VTAB 4: HTAB 1: PRINT "WELCOME TO SHOW."
160  PRINT : PRINT "A HIRES GRAPHICS-WINDOW P
     ROGRAM."
170  VTAB 10: PRINT "THIS PROGRAM IS DESIGNED
      TO ALLOW"
180  PRINT : PRINT "YOU TO INCORPORATE PROFES
     SIONAL-"
190  PRINT : PRINT "LOOKING WINDOWS INTO YOUR
      PROGRAMS!"
200  PRINT
210  PRINT : GOSUB 840
220  &  STORE 2,15,3,10
230  PRINT "FIRST, YOU NEED";
240  PRINT "TO INITIALIZE"
250  PRINT "THE SYSTEM WITH"
260  INVERSE : PRINT "&NEW A$,L,W,T,B": NORMAL

270  GOSUB 840
280  &  RESTORE : &  STORE 19,19,3,10
290  PRINT "ONCE EVERYTHING IS"
300  PRINT "INITIALIZED, THE": PRINT "COMMAND
     S:";
310  INVERSE : PRINT "&DRAW";: NORMAL : PRINT
     " AND"
320  INVERSE : PRINT "&STORE";: NORMAL : PRINT
     " CAN BE USED"
330  PRINT "TO OPEN UP A WINDOW";
340  PRINT "ON THE SCREEN."
350  GOSUB 840
360  &  RESTORE : &  STORE 2,10,10,15
370  PRINT "THESE TWO COMMANDS"
380  PRINT "ARE ALMOST";
390  PRINT "THE SAME "
400  PRINT "EXCEPT...";
410  GOSUB 850
420  &  RESTORE : &  STORE 13,18,10,16
430  INVERSE : PRINT "&STORE";: NORMAL : PRINT
     " ALLOWS YOU"
440  PRINT "TO RESTORE THE OLD";
450  PRINT "WINDOW WITH THE"
460  PRINT "COMMAND: ";: INVERSE : PRINT "&RE
     STORE": NORMAL
470  PRINT : GOSUB 840
480  &  RESTORE : &  STORE 2,17,15,22
490  PRINT "ONE FINAL COMMAND"
500  INVERSE : PRINT "&HOME";: NORMAL : PRINT
     " IS USED TO"
510  PRINT "CLEAR THE HIRES"
520  PRINT "WINDOW."
530  PRINT : GOSUB 840
540  &  RESTORE : &  DRAW 9,21,5,10
550  PRINT "NOW THAT YOU KNOW THE";
560  PRINT "COMMANDS, HERE'S WHAT";
570  PRINT "YOU CAN DO..."
580  PRINT : GOSUB 840
590  FOR I = 1 TO 10
600  &  STORE I + 4,5,I + 3,I + 8: PRINT I
610  FOR J = 1 TO 500: NEXT
620  NEXT
630  FOR I = 1 TO 10: &  RESTORE
640  FOR J = 1 TO 500: NEXT
650  NEXT : &  HOME
660  PRINT "LET'S DO IT AGAIN,"
670  PRINT "ONLY FASTER...."
680  PRINT : GOSUB 840
690  FOR I = 1 TO 10
700  &  STORE I + 3,5,I + 3,I + 8: PRINT I
710  NEXT : FOR I = 1 TO 10
720  &  RESTORE : NEXT : &  NEW " COLORS BRUS
     HES SHAPES PRINTER FILE",1,38,2,22
730  VTAB 5: PRINT "YOU CAN EVEN SIMULATE PUL
     L DOWN MENUS"
740  PRINT : GOSUB 840
750  &  STORE 1,6,1,7: INVERSE : PRINT "RED
     ";
760  NORMAL : VTAB 3: PRINT "BLUE": PRINT "VI
     OLET": VTAB 5: PRINT "GREEN": PRINT "BLA
     CK": PRINT "WHITE";
770  FOR J = 1 TO 4000: NEXT
780  &  RESTORE
790  &  HOME : PRINT "THE POSSIBILITIES ARE E
     NDLESS!"
800  PRINT
810  PRINT : GOSUB 840
820  &  DRAW 1,38,2,23
830  END
840  PRINT "PRESS ";: INVERSE : PRINT "RETURN
     ";: NORMAL
850  WAIT 49152,128: POKE 49168,0: RETURN
```

END OF LISTING 3

```
            KEY PERFECT 5.0
                RUN ON
               SHOW.DEMO
======================================
  CODE-5.0    LINE# - LINE#   CODE-4.0
  --------    -------------   --------
  AB510E49      10  -  100      7DAF
  D489CB4A     110  -  200      AB5D
  1A18280A     210  -  300      551A
  737246FF     310  -  400      509B
  4F505A21     410  -  500      5D14
  FD6B76FD     510  -  600      511C
  C6943B60     610  -  700      4411
  21134CB7     710  -  800      7B1B
  DD629DA8     810  -  850      2567
  5807A441 = PROGRAM TOTAL =    0753
```