GETTING THE BIG PICTURE

High-resolution graphics on the Apple II

This installment of Nibbling at Assembly Language provides an introduction to high-resolution graphics and arcade game design. Even if you aren't interested in arcade games, you can apply these programming techniques when you write software for computer-assisted instruction and business presentation graphics.

In this article you will learn about the "big picture": how to draw Hi-Res pictures and save them to disk.

THE HIGH-RESOLUTION SCREEN

The Apple high-resolution graphics screen consists of small dots or picture elements (pixels), each of which is referenced by a set of coordinates. The coordinates specify the horizontal (x) and vertical (y) locations of a particular pixel on the screen. The pixel in the upper-left corner of the screen has the coordinates x,y = (0, 0). In high-resolution graphics, there are 280 pixels on each row and 192 pixels on each vertical column as shown in **Figure 1.**

The Apple II supports a full-screen graphics mode, in which all 192 rows of pixels are displayed, and a mixed graphics/text mode, in which only 160 rows of pixels are displayed and four lines of text at the bottom of the screen are shown.

If you are interested in the IIGS's Super Hi-Res graphics, check the bibliography at the end of this article.

The seven possible pixel colors are shown in Table 1. Although any pixel can be black or white, only those pixels whose X-coordinate is even can assume the even-numbered colors (violet and blue), and only those pixels whose X-coordinate is odd can assume the odd-numbered colors (green and orange).

GRAPHICS APPLESOFT ROM ROUTINES

The simplest way of implementing Hi-Res graphics from assembly language is through the Applesoft ROM graphics routines. Below is an explanation of the nine major graphics routines.

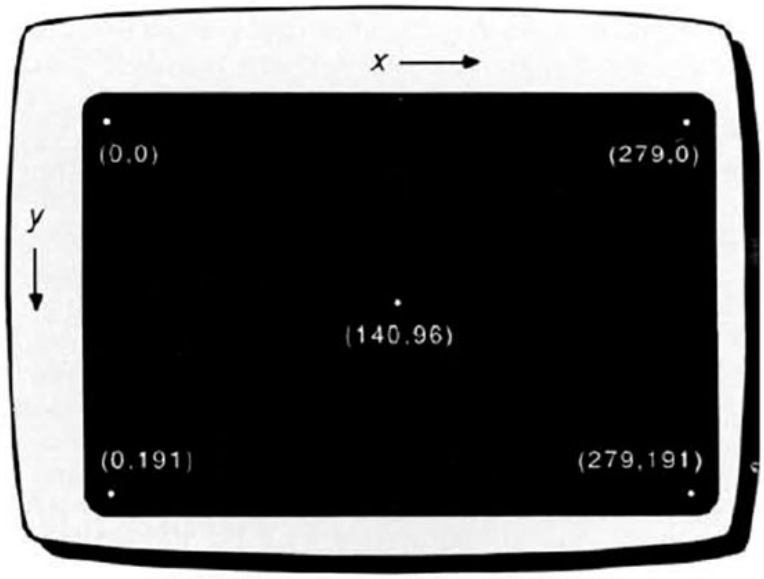

Figure 1: The Hi-Res Screen

HGR ($F3E2) and **HGR2** ($F3D8) initialize Hi-Res pages 1 and 2, respectively. Initialization involves six steps:

1. Setting the zero-page address HPAG ($E6) to the value $20 (for page 1 graphics) or $40 (for page 2 graphics)
2. Setting the system to either page 1 mode or page 2 mode
3. Setting the system to Hi-Res graphics (as opposed to Lo-Res graphics)
4. Setting the system to mixed graphics and text if HGR is called or to full graphics if HGR2 is called
5. Setting the system to display the graphics
6. Clearing the Hi-Res screen

This article will discuss each of these six steps individually.

HCLR ($F3F2) clears the current Hi-Res graphics screen. Make

sure you have initialized either page 1 or page 2 graphics before calling this routine.

BKGND ($F3F6) clears the Hi-Res screen to the color of the last plotted pixel. Use SETHCOL (see below) to set the desired color and HPLOT (see below) to plot a pixel, then call BKGND to fill the screen with that color.

Table 1: High-Resolution Graphics Colors

0 = black1	3 = white1	6 = blue
1 = green	4 = black2	7 = white2
2 = violet	5 = orange (red)	

SETHCOL ($F6EC) sets the Hi-Res color. To use this routine, load the desired color value (0 through 7) into the X-register, and execute a JSR SETHCOL. Your assembly code might look like this:

```
SETHCOL EQU $F6EC   ;Set hi-res color routine
        LDX #3      ;Set color to white1
        JSR SETHCOL
```

Be sure to call SETHCOL before trying to use any of the plotting functions described below in order to set the desired drawing color.

HPLOT ($F457) plots a pixel at the location specified in the 6502 registers. The high-order byte (HOB) of the X-coordinate is contained in the Y-register, the low-order byte (LOB) of the X-coordinate is contained in the X-register, and the Y-coordinate is contained in the Accumulator. For example,

```
HPLOT   EQU $F457   ;Plot-a-pixel routine
        LDY XPOS+1  ;Load x-coord HOB
        LDX XPOS    ;Load x-coord LOB
        LDA YPOS    ;Load y-coord
        JSR HPLOT   ;Plot the pixel
```

The color of the pixel is determined by the current Hi-Res color, as specified in the call to SETHCOL.

HPOSN ($F411) positions the internal (invisible) graphics cursor without plotting a point. This routine is used in drawing lines and vector shapes with the ROM routines HLIN, DRAW, and XDRAW, as explained below. To use HPOSN, place the X-coordinate HOB in the Y-register, the X-coordinate LOB in the X-register, the Y-coordinate in the Accumulator, and execute a JSR to HPOSN. For example,

```
HPOSN   EQU $F411   ;Position Hi-Res cursor routine
        LDY XPOS+1  ;Load HOB of x-coord
        LDX XPOS    ;Load LOB of x-coord
        LDA YPOS    ;Load y-coord
        JSR HPOSN   ;Put cursor there
```

HLIN ($F53A) draws a line, of the color specified with SETHCOL, from the location of the internal graphics cursor (positioned with HPOSN) to the location specified by the 6502 registers. The X-coordinate HOB is placed in the X-register, the X-coordinate LOB in the Accumulator, and the Y-coordinate in the Y-register. For example, to draw a line from x1, y1 = (15, 20) to x2, y2 = (200, 140), you would use the code shown below.

```
HLIN    EQU $F53A ;Draw a line
        LDY #0    ;Set x1 HOB to zero
        LDX #15   ;Get x1 LOB
        LDY #20   ;Get y1
        JSR HPOSN ;Position cursor
        LDX #200/ ;Get x2 HOB
        LDA #200  ;Get x2 LOB
        LDY #140  ;Get y2
        JSR HLIN  ;Draw the line
```

DRAW ($F601) and **XDRAW** ($F65D) draw Applesoft vector shapes on the Hi-Res screen. The DRAW routine simply draws the shape onto the screen and over any graphics currently on the screen. The XDRAW routine performs an exclusive OR between the shape and the current pixels on the screen. With a completely black (clear) background, DRAW and XDRAW produce exactly the same effect. With a completely white background, a white shape will not show up on the screen when drawn with DRAW but will produce a black image when drawn with XDRAW.

To use these routines, first define a vector shape as described in the *Applesoft BASIC Programming Reference Manual*. (I will not explain how to create vector shapes here because bit-mapped shapes — used in virtually all arcade games — are much faster than vector shapes.) Second, use HPOSN to position the internal graphics cursor at the location where you want to draw the vector shape. Third, load the rotation factor (see the ROT command in the Applesoft manual) into the Accumulator. Fourth, load the address of the individual shape (not of the entire shape table) into the Y-register (HOB) and the X-register (LOB). Fifth (and last), do a JSR to DRAW or XDRAW.

SETTXT ($FB39) is not a graphics routine but is used to return from a graphics mode to text mode.

Most of the above routines are demonstrated in the program ARTIST at the end of this article.

SCREEN SOFT SWITCHES AND SPECIAL ADDRESSES

In addition to the ROM routines listed above, the Apple also contains a set of soft switches that give you flexibility and control over the Hi-Res modes. To turn on a switch, you simply have to access the given memory address. Traditionally, assembly language programmers use the BIT opcode for this purpose. **Table 2** describes all the graphics switches.

In addition to the above soft switches, the Apple also provides the three following zero-page addresses associated with graphics:

HPAG ($E6) contains the value $20 (decimal 32) for Hi-Res graphics page 1 or the value $40 (decimal 64) for page 2. These are the high-order bytes of the beginning addresses of the graphics screen memory maps. The HPAG value designates the graphics screen on which HPLOT, HLIN, DRAW, and XDRAW are active, not necessarily the graphics page being displayed.

ROT ($F9) contains the rotation factor (usually 0) for vector shapes drawn with DRAW and XDRAW; SCALE ($E7) contains the scaling factor (usually 1) for vector shapes drawn with DRAW and XDRAW.

By using the proper combination of these soft switches and special addresses, you can draw on one graphics screen while displaying the other (the HPAG zero-page address designates the screen on which drawing takes place), rapidly flip between the two graphics screens (using the FLIP1 and FLIP2 soft switches), initialize the graphics screen without clearing it (by setting HPAG to the desired value and accessing the correct combination of soft switches), and perform other graphics manipulations.

COMPACTING AND EXPANDING GRAPHICS PICTURES

Once you have drawn a picture on the graphics screen, you can save the picture to the disk by BSAVEing the memory range $2000 to $3FF7 (page 1) or $4000 to $5FF7 (page 2). This requires almost 8K (kilobytes) of disk space for each picture. But the normal graphics pictures usually consist of several shapes and drawings surrounded by a lot of blank space. For such pictures, the actual graphics information can fit into much less space than 8K.

Using HRCOMP to Compact Graphics

The program HRCOMP in **Listing 1** compresses Hi-Res graphics to 25 to 50 percent of the space required by normal pictures. To use

Table 2: Hi-Res Graphics Soft Switches

Switch	Function
SHOW ($C050)	Displays a graphics mode. Once you have used other switches to select the type of graphics (Hi- or Lo-Res) and the graphics page (1 or 2), the SHOW switch causes the system to flip from text mode to the selected graphics mode.
TEXT ($C051)	Displays text mode. Accessing the TEXT switch causes the text screen to be displayed. Once you access TEXT, you can still plot pixels and draw lines and shapes onto the (invisible) graphics screen, but only text will appear on the screen.
FULLSCRN ($C052)	Selects all text or all graphics (as opposed to mixed text and graphics).
MXEDSCRN ($C053)	Selects mixed graphics and text, with 160 rows of pixels and 4 lines of text visible on the screen. This works properly only in page 1 Hi-Res graphics mode.
FLIP1 ($C054)	Selects the primary page (page 1). Accessing this switch by itself does not cause the graphics page 1 to be displayed, but only selects page 1 if and when the SHOW switch is accessed.
FLIP2 ($C055)	Selects the secondary page (page 2). Accessing this switch while in text mode produces some strange results, since text page 2 (usually garbage) appears on the screen. None of the standard text commands (HOME, PRINT, etc.) has an effect on text page 2.
LRSCRN ($C056)	Selects Lo-Res graphics (as opposed to Hi-Res). Accessing this switch simply tells the system the type of graphics to display if and when the SHOW switch is accessed.
HRSCRN ($C057)	Selects Hi-Res graphics (as opposed to Lo-Res).

HRCOMP, BLOAD into memory the Hi-Res picture you wish to compact, or draw the picture directly onto one of the two Hi-Res screens. Then BLOAD HRCOMP into memory. Since HRCOMP is completely relocatable, you may BLOAD it into any free memory space. Tell HRCOMP the screen (page 1 or page 2) on which the picture is located by storing the value $00 (for page 1) or $20 (decimal 32, for page 2) in memory location $06. In HRCOMP (and in the other two program listings in this article), this memory location is labeled SCRNNUM (screen number). From BASIC, you would include the following code:

```
200 POKE 6,0: REM SET HR PAGE 1
```

or

```
200 POKE 6,32: REM SET HR PAGE 2
```

From assembly language, you would include the following code:

```
LDA #0       ;Set to page 1
STA SCRNNUM  ;Save at $06
```

or

```
LDA #$20     ;Set to page 2
STA SCRNNUM  ;Save at $06
```

Set memory locations $1E and $1F (labeled COMPTR in HRCOMP and the other programs here) to the address at which you want to store your compressed picture. In BASIC, you would include the following code:

```
210 POKE 30, LOB :REM SET LOB OF COMPACT PICTURE
    ADDRS
220 POKE 31, HOB :REM SET HOB OF COMPACT PICTURE
    ADDRS
```

In assembly language, you would include the following code:

```
LDA #COMPIC  ;Get LOB of address
STA COMPTR   ;COMPTR = $1E
LDA #COMPIC/ ;Get HOB of address
STA COMPTR+1
```

Call HRCOMP from your BASIC or assembly language program. For example, if you BLOADed HRCOMP into memory location $6000 (decimal 24576), your BASIC program would contain the following line of code:

```
230 CALL 24576 : REM EXECUTE HRCOMP
```

Your assembly language program would contain this code:

```
HRCOMP EQU $6000  ;Set BLOAD address
       JSR HRCOMP ;Call HRCOMP
```

Get the length of the compact picture from memory locations $08 (LOB) and $09 (HOB). For example in BASIC, you would include this code:

```
240 LCP = PEEK(8) + PEEK(9) * 256 : REM GET LENGTH
    OF COMPACT PICTURE
```

LCP is the variable to hold the length of the compact picture. In assembly language, you would type

```
LENGTH EQU $08  ;Compact picture length
```

and then simply use the label LENGTH to refer to the compact picture length.

BSAVE the compact picture to the disk. Use the same BSAVE address from **lines 210-220**, above, and the BSAVE length. In BASIC, the code would be as follows:

```
250 PRINT CHR$(4);"BSAVE CPICT,A";LOB+256*HOB;",L";
    LCP
```

In assembly language, you would use the starting and ending addresses above to execute a disk BSAVE. See Nibbling at Assembly Language Part XV, "Be an Assembly Language Disk Jockey," *Nibble*, March 1988, page 44, on using DOS and ProDOS from assembly language.

If you erase or modify the Hi-Res picture, use HREXP to expand the compressed picture back onto the graphics screen.

How HRCOMP Works

HRCOMP scans the memory range $2000 to $3FF7 or $4000 to $5FF7 searching for bytes in which pixels are turned on and ignoring most of the bytes in which the pixels are turned off. The program then stores the starting address of a series of nonzero bytes, followed by the series of nonzero bytes themselves. At the end of the group of nonzero bytes, HRCOMP places the value $80 in the compact picture file ($80 is used as an end marker, since it does not occur as an actual image byte). HRCOMP places three consecutive $80 values to mark the end of the compact picture file. The compact pictures created with HRCOMP, therefore, have the following data structure:

Byte 0: The LOB of the starting address of a series of nonzero bytes (pixels on).
Byte 1: The HOB of the starting address of the nonzero bytes.
Byte 2 through n: The series of nonzero bytes. This is the actual image of on pixels.
Byte n+1: The end marker value $80 for that series of on pixels
Byte n+2: The LOB of the starting address of the next series of non zero bytes.

And so forth, until all on pixels are saved, after which HRCOMP places three consecutive $80 bytes.

Using HREXP to Expand Pictures

The program **HREXP** (**Listing 3**) expands a compressed picture back onto the Hi-Res graphics screen. To use HREXP either in Applesoft BASIC or in assembly language, first BLOAD the compact picture into memory. Make sure you specify the BLOAD address rather than let the picture BLOAD at its default location, which may not be where you want it. Then BLOAD HREXP into memory. Since HREXP is only 64 bytes in length, it easily fits into the page 3 user space (at $300 or decimal 768), but since it is relocatable, you may BLOAD it wherever you wish.

You can apply these programming techniques when writing software for computer-assisted instruction and business presentation graphics.

Set memory location $08 (SCRNNUM) to $00 for Hi-Res page 1 or to $20 (decimal 32) for page 2. Save the address of the compact picture in memory location COMPTR at $1E (LOB of the address) and $1F (HOB). Call HREXP. The graphics image is drawn on the designated Hi-Res screen. Note that HREXP does not erase the current picture on the screen but draws over the top of what is already on the screen. You may need to call **HCLR** (at $F3F2) prior to calling HREXP.

You can also use HREXP to erase the image that it just expanded onto the screen by changing **line 53** (address $0325) in **Listing 3** from $EA (NOP) to $A9 (LDA) and **line 54** (address $0326) from $EA to $00 (#0), as indicated in the comments. This loads the Accumulator with zero before writing to the graphics memory locations, thus erasing the image on the screen. Using HREXP to erase a picture that it has previously expanded is much faster than using HRCLR, HGR, or HGR2.

The Pros and Cons of HRCOMP and HREXP

The programs described here for compacting and expanding graphics images have two disadvantages:

1. HRCOMP works only if the screen is mostly black ($00 and $80 byte values). If your picture has huge blocks of color or white, the compact picture may actually be greater than the normal 8K. If you know this fact in advance, you can design your screens with plenty of blank space, or modify HRCOMP to ignore all-white bytes ($7F and $FF) rather than all-black bytes ($00 and $80).
2. In the course of drawing a Hi-Res picture, occasionally a byte value of $80 is generated, which usually has no effect at all on the graphics image. Therefore, HRCOMP sets all $80 byte values in the picture to $00 and reserves the value $80 for an end marker. This sometimes (but fortunately rarely) causes slight distortions in some of the graphics pixels.

On the other hand, HRCOMP and HREXP have two major advantages:

1. Both programs are very fast. Compacting and expanding a typical picture occurs in a split second. In fact, the example program ARTIST (described below) uses HRCOMP and HREXP as an "undo" system, by saving the Hi-Res screen image after each major command and then restoring a previously saved image when the user invokes the Undo command.
2. The programs are short and therefore conserve disk and memory space. HRCOMP is only 328 bytes long and HREXP is only 64 bytes long.

A DEMONSTRATION

Listing 5 gives the simple graphics drawing program ARTIST, which demonstrates most of the Applesoft ROM routines, soft switches, and special memory locations involved in Hi-Res graphics, as well as demonstrating the use of HRCOMP and HREXP.

To use ARTIST, run the BASIC loader program ARTIST.LOAD (**Listing 7**). ARTIST will display a list of commands at the bottom of the screen:

The arrow keys move the primary cursor (the small, square, flashing box that starts in the middle of the screen).

The keys I (up), M (down), J (left), and K (right) move the secondary (or alternate) cursor (the single pixel, flashing dot that starts in the upper left corner of the screen).

The P key toggles the pen up and down. In pen down mode, the primary cursor leaves a trail of pixels wherever the cursor moves. In pen up mode, the cursor moves without drawing on the screen.

The L key draws a line between the primary cursor and the alternate cursor.

The F key toggles between mixed-screen graphics (where the keyboard commands are seen at the bottom of the screen) and full-screen graphics (where no text is seen at the bottom of the screen).

The C key clears the graphics screen.

The Escape key serves as the Undo function. Whenever you press P to toggle pen up and pen down, press L to draw a line, or press C to clear the screen, ARTIST uses HRCOMP to save the current picture as a compact file in memory. The Undo command restores the saved compact file onto the screen.

The number keys 0 through 7 set the current graphics color.

The Q key allows you to quit ARTIST. When you quit the program, ARTIST displays the address and length of the compact picture. If you want to BSAVE the picture, use these data to execute a BSAVE.

ARTIST is a simple (almost crude) drawing tool. A much longer program is required to provide sophisticated features you would want in a top-notch graphics utility. But ARTIST demonstrates most of the key features of an assembly language graphics program: use of the Applesoft ROM graphics routines and the graphics soft switches.

For example, **lines 93-102** show how to initialize high-resolution page 1 using the graphics soft switches and the zero-page address HPAG (SE6). **Lines 168-175** use a number key input to set the graphics color. **Lines 231-237** demonstrate how to use the soft switches to flip between full-screen graphics and mixed graphics and text. **Lines 242-249** draw a line on the graphics screen. **Lines 302-309** draw a vector shape (CURSOR) using the ROM routine XDRAW. **Lines 326-329** show how to plot a pixel on the screen. And finally, **lines 347-351** demonstrate the use of the compactor routine HRCOMP, and **lines 354-363** demonstrate the use of the expander routine HREXP.

You should take time to examine the entire source code of ARTIST to understand how to use graphics commands in your own assembly language programs.

Enhancing ARTIST

Adding commands to ARTIST is a relatively simple matter. All the keyboard commands are contained in **lines 119-264**. You only have to insert the code to check for any other key and include the operations of that key. You may, for example, wish to add commands to fill regions of the screen with color, draw boxes, and circles, and plot user-defined shape tables (in essentially the same way ARTIST now plots the cursors).

An addition of particular value would be a faster way to move the primary and secondary cursors. The current limitation is the repeat speed of the Apple keyboard. To give you an idea of how fast the cursors can move, BLOAD ARTIST, HRCOMP, and HREXP, then get into the Monitor (by typing CALL −151), and remove the code that clears the keyboard strobe by typing 8056:EA EA EA, and then run ARTIST by typing 8000G while still in the Monitor. Now press one of the arrow keys and watch the cursor fly across the screen. In fact, it now moves too fast to control. By using the appropriate delay in the main loop of the program and using an auxiliary key (such as the Open-Apple key) to start and stop the cursors, you could devise a workable system for moving the cursors quickly and with proper control.

In the next installment of this column you will learn about bit-mapped shapes, the key ingredients of arcade games and other software using fast, smooth animation.

ENTERING THE PROGRAMS

If you have an assembler, enter the source code from **Listing 1** and save the object code as HRCOMP. If you don't have an assembler, use the hex code in **Listing 2**, and save it to disk with the command

```
BSAVE HRCOMP,A$6000,L$148
```

Similarly, use the source code from **Listing 3** and save the object code as HREXP, or use **Listing 4** and save it to disk with

```
BSAVE HREXP,A$300,L$40
```

Use **Listing 5** and save the assembled object code as ARTIST, or **Listing 6**, saving the hex code with

```
BSAVE ARTIST,A$8000,L$304
```

Finally, enter the Applesoft program from **Listing 7** and save it to disk with

```
SAVE ARTIST.LOAD
```

For help with entering and saving the listings, see the Typing Tips section.

REFERENCES

1. *Applesoft II BASIC Programming Reference Manual*, Apple Computer, Inc., Cupertino, CA, 1978 (and later editions). Contains a complete explanation of Apple II graphics.
2. *Apple II Reference Manual*, Apple Computer, Inc., Cupertino, CA, 1982. Discusses graphics soft switches.
3. Val J. Golding (ed.), *All About Applesoft*, A.P.P.L.E., Renton, WA, 1981, pp. 55-56. This contains a description of the Applesoft ROM high-resolution graphics routines.
4. Lon Poole, *Apple II User's Guide*, Osborne/McGraw-Hill, Berkeley, CA, 1981, pp. 203-224. Primarily a book for Applesoft programmers, it provides an excellent discussion of Apple graphics.
5. Jeffrey Stanton, *Apple Graphics & Arcade Game Design*, The Book Co., Los Angeles, 1982. This book describes the Applesoft ROM graphics routines and vector shapes.
6. S. Scott Zimmerman, "Hi-Res Houdini," *Nibble*, October 1984, p. 14. Shows how to shift bits, scroll the graphics screen, merge high-resolution pictures, and perform other special effects in assembly language.

Super high-resolution graphics is beyond the scope of this article. The following recent *Nibble* articles contain information about super high-resolution graphics:

1. Tom Dorris, "Hplot GS," *Nibble*, October 1987, p. 52. Contains assembly language routines for using super high-resolution graphics.
2. Tom Dorris, "Super Hi-Res Graphics Converter," *Nibble*, June 1988, p. 68. Presents a BASIC program for converting normal high-resolution graphics pictures to super high-resolution graphics.
3. Jeff Hurlburt, "Super Hi-Res Picture Packer," *Nibble*, January 1988, p. 78. Includes a picture compactor and expander for super high-resolution graphics.
4. Tim Meekins, "SuperGraphics GS," *Nibble*, February 1988, p. 58. Contains assembly language routines for using super high-resolution graphics on the Apple IIGS.
5. David L. Smith, "AmperPalette," *Nibble*, November 1987, p. 19. Provides assembly language routines for managing super high-resolution graphics.

LISTING 1: HRCOMP Source Code

```
0  :
1  ********************************************
2  *                                          *
3  *              HRCOMP Source Code          *
4  *                                          *
5  *            by S. Scott Zimmerman         *
6  *             Copyright (c) 1988           *
7  *             by MicroSPARC, Inc           *
8  *             Concord, MA  01742           *
9  *                                          *
10 *          The MicroSPARC Assembler        *
11 *                                          *
12 ********************************************
13
14          ORG $6000       ;Relocatable
15
16 ********************************************
17 *  EQUates:                                *
18 ********************************************
19
20 ENDCOMP  EQU $00         ;End addrs of compact
21 COMPSTRT EQU $02         ;Start addrs of compact
22 SCRNNUM  EQU $06         ;0=HR pg 1 ; 32=HR pg 2
23 LENGTH   EQU $08         ;Length compact file
24 SCREND   EQU $19         ;Pntr last scrn byte+1
25 HRPTR    EQU $1C         ;Pntr to HR screen
26 COMPTR   EQU $1E         ;Pntr to compact pict
27 ENDBYT   EQU $80         ;#$80 ends 'on' bytes
28 SCRN1    EQU $2000       ;HR pg 1 start
29 MAXLEN   EQU 5000        ;Max length allowed
30 ENDSCRN  EQU $3FF8       ;Last adrs of pg 1
31
32 ********************************************
33 *  Define macros:                          *
34 ********************************************
35
36 INCHR     MAC             ;Increment HR scrn byte
37          INC HRPTR       ;Go to next HR byte
38          BNE ]A
39          INC HRPTR+1
40 ]A       LDA HRPTR       ;Past HR screen?
41          CMP SCREND      ;Compare LOB
42          LDA HRPTR+1     ;Get HOB for compare
43          SBC SCREND+1    ;End of 16-bit compare
44          BCS :A          ;Branch to here
45          EMC
46
47 INCCO    MAC             ;Increment compact byte
48          INC COMPTR      ;Go to next compact byte
49          BNE ]A
50          INC COMPTR+1
```

LISTING 1: HRCOMP Source Code

```
]A      LDA COMPTR      ;Past max length?
        CMP ENDCOMP     ;Compare LOB
        LDA COMPTR+1    ;Get HOB for compare
        SBC ENDCOMP+1   ;End of 16-bit compare
        BCS :A          ;Branch to here
        EMC

*************************************************
*  Initialize program pointers:                 *
*************************************************

        LDA #SCRN1      ;Set scrn pntr address
        STA HRPTR       ;LOB
        CLC             ;Prepare to add
        LDA #SCRN1/     ;HOB
        ADC SCRNNUM     ;Add 32 if HR page 2
        STA HRPTR+1
        LDA #ENDSCRN    ;Set end pointer
        STA SCREND      ;LOB
        CLC             ;Prepare to add
        LDA #ENDSCRN/   ;HOB
        ADC SCRNNUM     ;Add 32 if HR page 2
        STA SCREND+1
        CLC             ;Prepare to add
        LDA #MAXLEN     ;Get maximum length
        ADC COMPTR      ;Add compact address
        STA ENDCOMP     ;Save as end of compact
        LDA #MAXLEN/    ;HOB
        ADC COMPTR+1
        STA ENDCOMP+1   ;Save HOB

        LDA COMPTR      ;Save compact address
        STA COMPSTRT
        LDA COMPTR+1
        STA COMPSTRT+1

*************************************************
*  Compact the Hi-Res picture:                  *
*************************************************

GETBYT  LDY #0          ;Zero the index
        LDA (HRPTR),Y   ;Get HR screen byte
        BEQ NODOT       ;Nothing there
        CMP #ENDBYT     ;Not 0, is it $80?
        BEQ NODOT       ;Yes, so treat as blank
        PHA             ;Save HR screen byte
        LDA HRPTR       ;Adrs of this byte
        STA (COMPTR),Y  ;Save in compact pict
        INCCO TOOLONG1  ;Increment compact byte
        LDA HRPTR+1     ;Get HOB curr HR addrs
        SEC             ;Prepare to subtract
        SBC SCRNNUM     ;Subtract 32 if HR pg 2
        STA (COMPTR),Y  ;Save HOB
        INCCO TOOLONG1  ;Increment compact byte
        PLA             ;Restore screen byte
        STA (COMPTR),Y  ;Store byte value
        INCCO TOOLONG1  ;Increment compact byte
        INCHR STLEN1    ;Increment HR byte
        CLV             ;To force branch
        BVC DOTON       ;Always branch
NODOT   INCHR STLEN1    ;Increment HR byte
        CLV             ;To force branch
        BVC GETBYT      ;Always branch

*************************************************
*  Relays (to avoid jumps for relocatability)   *
*************************************************

TOOLONG1 CLV            ;To force branch
        BVC TOOLONG2    ;Always branch
STLEN1  CLV             ;To force branch
        BVC STLEN       ;Always branch
GETBYT1 CLV             ;To force branch
        BVC GETBYT      ;Always branch

*************************************************
*  Check if next 4 bytes are zero:              *
*************************************************

DOTON   LDA (HRPTR),Y   ;Get HR byte
        INY             ;Go to next HR byte
        ORA (HRPTR),Y   ;OR it with previous
        INY             ;Go to next HR byte
        ORA (HRPTR),Y   ;Or it with previous
        INY             ;Got to next HR byte
        ORA (HRPTR),Y   ;Or this also
        BEQ CLEAR       ;All are clear
        CMP #ENDBYT     ;Also check for $80
        BEQ CLEAR       ;No pixel on either
        LDY #0          ;Go back to orig byte
        LDA (HRPTR),Y   ;Get it again
        CMP #ENDBYT     ;Is it $80?
        BNE ADDON       ;No, so add to table
        LDA #0          ;Yes, convert to 0
ADDON   STA (COMPTR),Y  ;Save byte in compact
        INCCO TOOLONG1  ;Increment compact
        INCHR STLEN     ;Increment HR byte
        CLV             ;To force branch
        BVC DOTON       ;Always branch
CLEAR   LDY #0          ;Set index back to 0
        LDA #ENDBYT     ;Get end byte $80
        STA (COMPTR),Y  ; Store in table
        INCCO TOOLONG   ;Increment compact
        INCHR STLEN     ;Increment HR byte
        CLV             ;To force branch
        BVC GETBYT1     ;Always branch

TOOLONG2 CLV            ;Relay (relocatability)
        BVC TOOLONG

*************************************************
*  Calculate length of compact picture:         *
*************************************************

STLEN   SEC             ;Prepare to subtract
        LDA COMPTR      ;Get current (end) adrs
        SBC COMPSTRT    ;Subtract starting adrs
        STA LENGTH      ;Save length LOB
        LDA COMPTR+1    ;Do HOB
        SBC COMPSTRT+1
        STA LENGTH+1
        LDA SCRNNUM     ;Restore scrn number
        STA HRPTR

        LDY #0          ;Put three $80 at end
        LDA #ENDBYT
        STA (COMPTR),Y
        INY
        STA (COMPTR),Y
        INY
        STA (COMPTR),Y

        LDA LENGTH      ;Check if zero
        ORA LENGTH+1
        BNE ADD3        ;No, so add 3
        RTS             ;Yes, so end here
ADD3    CLC             ;Add these 3 to length
        LDA LENGTH
        ADC #3
        STA LENGTH
        LDA LENGTH+1
        ADC #0
        STA LENGTH+1
QUIT    RTS             ;Done

TOOLONG LDA #0          ;Set length to 0...
        STA LENGTH      ; if compact picture
        STA LENGTH+1    ; is too long
        RTS             ;Abort
```

END OF LISTING 1

LISTING 2: HRCOMP

```
Start: 6000          Length:148

E6 6000:A9 00 85 1C 18 A9 20 65
19 6008:06 85 1D A9 F8 85 19 18
BB 6010:A9 3F 65 06 85 1A 18 A9
83 6018:88 65 1E 85 00 A9 13 65
5A 6020:1F 85 01 A5 1E 85 02 A5
28 6028:1F 85 03 A0 00 B1 1C F0
1B 6030:56 C9 80 F0 52 48 A5 1C
D8 6038:91 1E E6 1E D0 02 E6 1F
35 6040:A5 1E C5 00 A5 1F E5 01
EF 6048:B0 50 A5 1D 38 E5 06 91
DB 6050:1E E6 1E D0 02 E6 1F A5
C2 6058:1E C5 00 A5 1F E5 01 B0
21 6060:39 68 91 1E E6 1E D0 02
EF 6068:E6 1F A5 1E C5 00 A5 1F
B1 6070:E5 01 B0 26 E6 1C D0 02
96 6078:E6 1D A5 1C C5 19 A5 1D
8A 6080:E5 1A B0 19 B8 50 1C E6
75 6088:1C D0 02 E6 1D A5 1C C5
F1 6090:19 A5 1D E5 1A B0 06 B8
57 6098:50 91 B8 50 6F B8 50 6F
07 60A0:B8 50 88 B1 1C C8 11 1C
86 60A8:C8 11 1C C8 11 1C F0 33
02 60B0:C9 80 F0 2F A0 00 B1 1C
35 60B8:C9 80 D0 02 A9 00 91 1E
C0 60C0:E6 1E D0 02 E6 1F A5 1E
8E 60C8:C5 00 A5 1F E5 01 B0 CA
1F 60D0:E6 1C D0 02 E6 1D A5 1C
A1 60D8:C5 19 A5 1D E5 1A B0 2F
E8 60E0:B8 50 C0 A0 00 A9 80 91
C3 60E8:1E E6 1E D0 02 E6 1F A5
F0 60F0:1E C5 00 A5 1F E5 01 B0
7C 60F8:48 E6 1C D0 02 E6 1D A5
30 6100:1C C5 19 A5 1D E5 1A B0
70 6108:06 B8 50 94 B8 50 32 38
25 6110:A5 1E E5 02 85 08 A5 1F
26 6118:E5 03 85 09 A5 06 85 1C
75 6120:A0 00 A9 80 91 1E C8 91
E0 6128:1E C8 91 1E A5 08 05 09
5B 6130:D0 01 60 18 A5 08 69 03
8D 6138:85 08 A5 09 69 00 85 09
B8 6140:60 A9 00 85 08 85 09 60

TOTAL: 7C98
```

END OF LISTING 2

LISTING 3: HREXP Source Code

```
 0 :
 1 ********************************************
 2 *                                          *
 3 *              HREXP Source Code           *
 4 *                                          *
 5 *            by S. Scott Zimmerman         *
 6 *             Copyright (c) 1988           *
 7 *             by MicroSPARC, Inc           *
 8 *             Concord, MA  01742           *
 9 *                                          *
10 *              Merlin Assembler            *
11 *                                          *
12 ********************************************
13
14          ORG $300        ;Relocatable
15
16 ********************************************
17 *  EQUates:                                *
18 ********************************************
19
20 HRPTR    EQU $00         ;Hi-res screen pointer
21 SCRNNUM  EQU $06         ;0 = HR pg 1, 32 = pg 2
22 COMPTR   EQU $1E         ;Compact pict pointer
23 ENDBYT   EQU $80         ;#$80 ends 'on' bytes
24
25 ********************************************
26 *  Macro definition:                       *
27 ********************************************
28
29 INCR     MAC             ;16-Bit increment
30          INC :A
31          BNE ]A
32          INC :A+1
33 ]A       EMC
34
35 ********************************************
36 *  Program start:                          *
37 ********************************************
38
39 ADDRESS  LDY #0          ;Zero the index
40          LDA (COMPTR),Y  ;Get compact byte
41          INCR COMPTR     ;Point to next comp byt
42          STA HRPTR       ;Set hi-res address
43          CLC             ;Prepare to add
44          LDA (COMPTR),Y  ;Get HOB from compact
45          INCR COMPTR     ;Point to next comp byt
46          ADC SCRNNUM     ;Add HR page (0 or 32)
47          STA HRPTR+1     ;Set HOB
48
49 BYTVAL   LDA (COMPTR),Y  ;Get HR byte value
50          INCR COMPTR     ;Point to next comp byt
51          CMP #ENDBYT     ;End of string?
52          BEQ TABLEND     ;Yes, end of string
53          NOP             ;To erase, put $A9 here
54          NOP             ; and $00 here
55          STA (HRPTR),Y   ;Put pixels on screen
56          INCR HRPTR      ;Point to next HR byte
57          CLV             ;To force branch
58          BVC BYTVAL      ;Always branch
59 TABLEND  LDA (COMPTR),Y  ;Three $80's in a row?
60          CMP #ENDBYT
61          BNE ADDRESS     ;No, so proceed
62          INY             ;Go to next compact byt
63          LDA (COMPTR),Y
64          CMP #ENDBYT
65          BNE ADDRESS     ;No, so proceed
66          RTS             ;Yes, so done
```

END OF LISTING 3

LISTING 4: HREXP

```
Start: 300                    Length:40

DE 0300:A0 00 B1 1E E6 1E D0 02
16 0308:E6 1F 85 00 18 B1 1E E6
9E 0310:1E D0 02 E6 1F 65 06 85
C4 0318:01 B1 1E E6 1E D0 02 E6
E3 0320:1F C9 80 F0 0D EA EA 91
06 0328:00 E6 00 D0 02 E6 01 B8
06 0330:50 E7 B1 1E C9 80 D0 C8
F0 0338:C8 B1 1E C9 80 D0 C1 60

TOTAL: 5711
```

END OF LISTING 4

LISTING 5: ARTIST Source Code

```
 0 :
 1 ********************************************
 2 *                                          *
 3 *              ARTIST Source Code          *
 4 *                                          *
 5 *            by S. Scott Zimmerman         *
 6 *             Copyright (c) 1988           *
 7 *             by MicroSPARC, Inc           *
 8 *             Concord, MA  01742           *
 9 *                                          *
10 *           The MicroSPARC Assembler       *
11 *                                          *
12 ********************************************
13
14          ORG $8000       ;Decimal 32768
15
16 ********************************************
17 *  EQUates:                                *
18 ********************************************
19
20 XALT     EQU $04         ;Altern X position
21 SCRNNUM  EQU $06         ;Scrn num for HRCOMP
22 LENGTH   EQU $08         ;Length of compact pict
23 CH       EQU $24         ;Cursor horizontal
24 COMPTR   EQU $1E         ;Compact start
25 XPOS     EQU $CE         ;Cursor X position
26 YALT     EQU $E3         ;Altern Y position
27 YPOS     EQU $FA         ;Cursor Y position
28 FLSHDEL  EQU $FB         ;Flash delay
29 MSGPTR   EQU $FE         ;Message pointer
30 HREXP    EQU $300        ;Expander routine
31 APLSOFT  EQU $3D0        ;Aplsoft warm start
32 COMPIC   EQU $4000       ;Compressed picture
33 HRCOMP   EQU $6000       ;Compactor routine
34 KEYBD    EQU $C000       ;Keyboard input
35 STROBE   EQU $C010       ;Clear keyboard strobe
36 XDRAW    EQU $F65D       ;XOR shape onto scrn
37 PRNTAX   EQU $F941       ;Print hex number
38 SETTXT   EQU $FB39       ;Set text mode
39 TABV     EQU $FB5B       ;Vertical tab routine
40 HOME     EQU $FC58       ;Clear text screen
41 CROUT    EQU $FD8E       ;Output carriage return
42 COUT     EQU $FDED       ;Output a character
43 ----------------------------------------------
44 ESC      EQU $9B         ;Escape code
45 LARR     EQU $8B         ;Left arrow code
46 DARR     EQU $8A         ;Down arrow code
47 UARR     EQU $8B         ;Up arrow code
48 RARR     EQU $95         ;Right arrow code
49 DELVAL   EQU $2000       ;Delay value
50
51 ********************************************
52 *  Hi-Res Graphics ROM routines and switches:
53 ********************************************
54
55 ROT      EQU $F9         ;Shape rotation
56 HPAG     EQU $E6         ;Hi-Res page ($20/$32)
57 SCALE    EQU $E7         ;Shape scale
58 SHOW     EQU $C050       ;Display graphics scrn
59 FULLSCRN EQU $C052       ;Display full scrn
60 MXEDSCRN EQU $C053       ;Display mixed gr/txt
61 FLIP1    EQU $C054       ;Display screen #1
62 HRSCRN   EQU $C057       ;Display Hi-Res gr
63 HCLR     EQU $F3F2       ;Clear Hi-Res screen
64 HPOSN    EQU $F411       ;Position HR cursor
65 HPLOT    EQU $F457       ;Plot a pixel
66 HLIN     EQU $F53A       ;Draw a line
67 SETHCOL  EQU $F6EC       ;Set HR color
68
69 ********************************************
70 *  Initialize:                             *
71 ********************************************
72
73          LDA #0          ;Set things to zero
74          STA XALT        ;Alt cursor
75          STA XALT+1
76          STA YALT
77          STA CURSFLG     ;Init cursor flag
78          STA ROT         ;Set rotation to 0
79          STA SCRNNUM     ;Set to page 1
80          STA PENFLG      ;Set to pen up
81          STA FMFLG       ;Set to mixed
82          STA XPOS+1      ;Clear HOB of X pos
83          STA LENGTH      ;Init to no length
84          STA LENGTH+1
85          LDA #140        ;Set main cursor pos
86          STA XPOS
87          LDA #96         ;Set Y position
88          STA YPOS
89          LDX #3          ;Set color to white
90          JSR SETHCOL     ;Call to set color
91          LDA #1          ;Set the scale to 1
92          STA SCALE
93          LDA #$20        ;Set HR page 1
94          STA HPAG
95          JSR HCLR        ;Clear HR screen
96          JSR HOME        ;Clear text scrn
97          JSR PRNTMENU    ;Print menu
98          BIT HRSCRN      ;Select hi-res screen
99          BIT FLIP1       ;Make sure it's pg 1
100         BIT MXEDSCRN    ;Set to mixed scrn
101         BIT SHOW        ;Show HR screen
102         BIT STROBE      ;Clear kb strobe
103         JSR PLOTCURS    ;Plot the cursor
104
105 ********************************************
106 *  Start ARTIST main loop:                 *
107 ********************************************
108
109 MAINLOOP BIT KEYBD       ;Keypressed?
```

```
110             BMI GETKEY      ;Yes, get key
111             JMP NOHIT       ;No, proceed
112 GETKEY      LDA KEYBD       ;Get key
113             BIT STROBE      ;Clear keybd strobe
114             PHA             ;Save keyboard input
115             LDA CURSFLG     ;Is flag on?
116             BEQ CHECKIN     ;No, just check input
117             JSR PLOTCURS    ;Yes, erase cursors
118 CHECKIN     PLA             ;Restore input
119             CMP #RARR       ;Right arrow?
120             BNE NEXT1       ;No, check next
121             INC XPOS        ;Move right a pixel
122             BNE ]A
123             INC XPOS+1
124 ]A          LDA XPOS        ;Is it too high?
125             CMP #280        ;Past 279?
126             LDA XPOS+1
127             SBC #280/
128             BCC SETNEW      ;No, just set new
129             LDA #0          ;Yes, wrap back to 0
130             STA XPOS
131             STA XPOS+1
132 SETNEW      JSR PLOT        ;If pen down, plot point
133             JSR PLOTCURS    ;Go turn cursor on
134             JMP MAINLOOP
135 NEXT1       CMP #LARR       ;Left arrow?
136             BNE NEXT2       ;No, check next
137             LDA XPOS        ;Is it at zero?
138             ORA XPOS+1
139             BNE ]A          ;No, so decrement
140             LDA #279        ;Yes, so wrap
141             STA XPOS
142             LDA #279/
143             STA XPOS+1
144             BNE SETNEW      ;Always
145 ]A          LDA XPOS        ;Decrement X
146             BNE ]B
147             DEC XPOS+1
148 ]B          DEC XPOS
149             JMP SETNEW      ;Go set new pos
150 NEXT2       CMP #UARR       ;Up arrow?
151             BNE NEXT3       ;No, check next
152             LDA YPOS        ;Is it at zero?
153             BNE ]A          ;No, so decrement
154             LDA #191        ;Yes, so wrap
155             STA YPOS
156             JMP SETNEW      ;Go set new pos
157 ]A          DEC YPOS        ;Decrement Y
158             JMP SETNEW      ;Go set new pos
159 NEXT3       CMP #DARR       ;Down arrow?
160             BNE NEXT4       ;No, check next
161             INC YPOS        ;Go down one pixel
162             LDA YPOS        ;Get current value
163             CMP #192        ;Is it past bottom?
164             BNE OKAY        ;No, it's okay
165             LDA #0          ;Yes, set to top
166             STA YPOS
167 OKAY        JMP SETNEW      ;Go set new pos
168 NEXT4       CMP #"0         ;Input a number?
169             BCC NEXT5       ;Too low
170             CMP #"8         ;Is it too high?
171             BCS NORMKEY     ;Yes, go check next
172             SEC             ;Prepare to subtract
173             SBC #"0         ;Subtract ASCII for 0
174             TAX             ;Put color in X
175             JSR SETHCOL     ;Set new color
176             JMP MAINLOOP
177 NEXT5       CMP #ESC        ;Undo last screen
178             BNE NORMKEY     ;No, proceed
179             JSR RESTORE     ;Restore from save
180             JMP MAINLOOP
181 NORMKEY     AND #%11011111  ;Conv lower->upper
182             CMP #"I         ;Aux cursor up?
183             BNE CHKJ        ;No, go check J
184             LDA YALT        ;Is it at zero?
185             BNE ]A          ;No, so decrement
186             LDA #191        ;Yes, so wrap
187             STA YALT
188             JMP SETNEW      ;Go set new pos
189 ]A          DEC YALT        ;Decrement Y
190             JMP SETNEW      ;Go set new pos
191 CHKJ        CMP #"J         ;Aux cursor left?
192             BNE CHKK        ;No, go check K
193             LDA XALT        ;Is it at zero?
194             ORA XALT+1
195             BNE ]A          ;No, so decrement
196             LDA #279        ;Yes, so wrap
197             STA XALT
198             LDA #279/
199             STA XALT+1
200             BNE OKAY        ;Always
201 ]A          LDA XALT        ;Decrement X
202             BNE ]B
203             DEC XALT+1
204 ]B          DEC XALT
205             JMP SETNEW      ;Go set new pos
206 CHKK        CMP #"K         ;Aux cursor right?
207             BNE CHKM        ;No, go check M
208             INC XALT        ;Move right a pixel
209             BNE ]A
210             INC XALT+1
211 ]A          LDA XALT        ;Is it too high?
212             CMP #280        ;Past 279?
213             LDA XALT+1
214             SBC #280/
215             BCC OK          ;No, just set new
216             LDA #0          ;Yes, wrap back to 0
217             STA XALT
218             STA XALT+1
219             BEQ OK          ;Always branch
220 CHKM        CMP #"M         ;Aux cursor down?
221             BNE CHKF        ;No, go check F
222             INC YALT        ;Go down one pixel
223             LDA YALT        ;Get current value
224             CMP #192        ;Is it past bottom?
225             BNE OK          ;No, it's okay
226             LDA #0          ;Yes, set to top
227             STA YALT
228 OK          JMP SETNEW      ;Go set new pos
229 CHKF        CMP #"F         ;Full/mixed toggle?
230             BNE CHKL        ;No, go check L
231             LDA FMFLG       ;Get full/mixed flag
232             EOR #1          ;Toggle it
233             STA FMFLG       ;Save result
234             BNE SETFULL     ;Go set to full
235             BIT MXEDSCRN    ;Set to mixed
236             JMP MAINLOOP
237 SETFULL     BIT FULLSCRN    ;Set to full
238             JMP MAINLOOP
239 CHKL        CMP #"L         ;Draw line?
240             BNE CHKC        ;No, go check C
241             JSR SAVE        ;Save current screen
242             LDY XPOS+1      ;Get cursor position
243             LDX XPOS
244             LDA YPOS
245             JSR HPOSN       ;Set that position
246             LDX XALT+1      ;Get alt cursor pos
247             LDA XALT
248             LDY YALT
249             JSR HLIN        ;Draw a line
250             JMP MAINLOOP
251 CHKC        CMP #"C         ;Clear screen?
252             BNE CHKP        ;No, go check P
253             JSR SAVE        ;Save current screen
254             JSR HCLR        ;Clear graphics scrn
255             JMP MAINLOOP
256 CHKP        CMP #"P         ;Toggle pen up/down?
257             BNE CHKQ        ;No, go check Q
258             JSR SAVE        ;Save current screen
259             LDA PENFLG      ;Get current setting
260             EOR #1          ;Toggle 0 <--> 1
261             STA PENFLG
262             JMP MAINLOOP
263 CHKQ        CMP #"Q         ;Quit?
264             BEQ QUIT        ;Yes
265 NOHIT       INC FLSHDEL     ;Increment delay
266             BNE ]A
267             INC FLSHDEL+1
268 ]A          LDA FLSHDEL     ;End of flash delay?
269             CMP #DELVAL
270             LDA FLSHDEL+1
271             SBC #DELVAL/
272             BCS BLINK       ;Yes, blink cursor
273             JMP MAINLOOP
274 BLINK       JSR PLOTCURS
275             JMP MAINLOOP
276 ;-----------------------------------------------
277 QUIT        JSR SAVE        ;Save the HR screen
278             JSR SETTXT      ;Set back to text mode
279             JSR HOME        ;Clear the text screen
280             LDX #ADRMSG     ;Print address message
281             LDY #ADRMSG/
282             JSR MESSAGE
283             LDA #COMPIC/    ;Get compact address
284             LDX #COMPIC
285             JSR PRNTAX
286             LDX #LENMSG
287             LDY #LENMSG/
288             JSR MESSAGE
289             LDA LENGTH+1    ;Get compact length
290             LDX LENGTH
291             JSR PRNTAX      ;Print hex value
292             JSR CROUT       ;Carriage return
293             JMP APLSOFT
294
295 ;-----------------------------------------------
296 ;  Auxiliary routines:                          *
297 ;-----------------------------------------------
298
299 PLOTCURS    LDA CURSFLG     ;Get current flag
300             EOR #1          ;Toggle 1 <--> 0
301             STA CURSFLG
302             LDY XPOS+1      ;Set X location
303             LDX XPOS
304             LDA YPOS        ;Set Y location
305             JSR HPOSN       ;Set its position
306             LDA #0          ;Zero rotation
307             LDY #CURSOR/    ;Set shape location
308             LDX #CURSOR
309             JSR XDRAW       ;Draw the shape
310             LDY XALT+1      ;Set X location
311             LDX XALT
312             LDA YALT        ;Set Y location
313             JSR HPOSN       ;Set its position
314             LDA #0          ;Zero rotation
315             LDY #ALTCURS/   ;Set shape location
316             LDX #ALTCURS
317             JSR XDRAW       ;Draw the shape
318             LDA #0          ;Zero the flash delay
319             STA FLSHDEL
320             STA FLSHDEL+1
321             RTS
322 ;-----------------------------------------------
323 PLOT        LDA PENFLG      ;Is pen on?
324             BNE ON          ;Yes, so plot
325             RTS             ;No, so just return
326 ON          LDY XPOS+1      ;Set location
327             LDX XPOS
328             LDA YPOS
329             JMP HPLOT       ;Plot the point, return
330 ;-----------------------------------------------
331 PRNTMENU    LDX #0
332             STX CH          ;Set horiz location
```

```
333           LDA #20          ;Set verical location
334           JSR TABV         ;Move cursor there
335           LDX #MENUMSG     ;Get message address
336           LDY #MENUMSG/
337 MESSAGE   STX MSGPTR       ;Set message pointer
338           STY MSGPTR+1
339           LDY #0           ;Init index
340 MSGLOOP   LDA (MSGPTR),Y   ;Get character
341           BEQ MSGEND       ;Quit if zero
342           JSR COUT         ;Print character
343           INY              ;Go to next
344           BNE MSGLOOP      ;Always branch
345 MSGEND    RTS              ;End of message
346 ----------------------------------------
347 SAVE      LDA #COMPIC      ;Set address of compact
348           STA COMPTR
349           LDA #COMPIC/
350           STA COMPTR+1
351           JSR HRCOMP
352           RTS
353 ----------------------------------------
354 RESTORE   LDA LENGTH       ;Is a compact pict there?
355           ORA LENGTH+1
356           BNE RES          ;Yes, restore it
357           RTS              ;No, just return
358 RES       JSR HCLR         ;Clear current screen
359           LDA #COMPIC      ;Set compact pict adrs
360           STA COMPTR
361           LDA #COMPIC/
362           STA COMPTR+1
363           JSR HREXP
364           RTS
365
366 ----------------------------------------------
367 -  Data:                                      -
368 ----------------------------------------------
369
370 MENUMSG   ASC "ESC=UNDO  0-7=COLOR  F=FULL/MXED"
371           ASC "  L=LINE"
372           ASC "C=CLEAR SCREEN    Q=QUIT     "
373           ASC "P=PEN UP/DOWN"
374           ASC "IJKM=MOVE ALT CURSOR  "
375           ASC "ARROWS=MOVE CURSOR"
376           BRK
377 ADRMSG    ASC "COMPACT PICT: A$"
378           BRK
379 LENMSG    ASC ",L$"
380           BRK
381 ----------------------------------------------
382 CURSFLG   DFS 1            ;0=Cursor off, 1=on
383 PENFLG    DFS 1            ;0=Pen up, 1=down
384 FMFLG     DFS 1            ;0=Mixed, 1=full
385 ----------------------------------------------
386 CURSOR    DFC $3A,$24,$2D,$36,$07,$00
387 ALTCURS   DFC $04,$00
```

END OF LISTING 5

LISTING 6: ARTIST

```
Start: 8000          Length:304

7A|8000:A9 00 85 04 85 05 85 E3
58|8008:8D F9 82 85 F9 85 06 8D
B1|8010:FA 82 8D FB 82 85 CF 85
E2|8018:08 85 09 A9 8C 85 CE A9
E7|8020:60 85 FA A2 03 20 EC F6
B4|8028:A9 01 85 E7 A9 20 85 E6
3F|8030:20 F2 F3 20 58 FC 20 2B
A5|8038:82 2C 57 C0 2C 54 C0 2C
C9|8040:53 C0 2C 50 C0 2C 10 C0
26|8048:20 E9 81 2C 00 C0 30 03
8F|8050:4C A5 81 AD 00 C0 2C 10
FE|8058:C0 48 AD F9 82 F0 03 20
F7|8060:E9 81 68 C9 95 D0 1F E6
09|8068:CE D0 02 E6 CF A5 CE C9
DE|8070:18 A5 CF E9 01 90 06 A9
0D|8078:00 85 CE 85 CF 20 1C 82
9E|8080:20 E9 81 4C 4B 80 C9 88
13|8088:D0 1B A5 CE 05 CF D0 0A
E4|8090:A9 17 85 CE A9 01 85 CF
BD|8098:D0 E3 A5 CE D0 02 C6 CF
07|80A0:C6 CE 4C 7D 80 C9 8B D0
03|80A8:10 A5 FA D0 07 A9 BF 85
AB|80B0:FA 4C 7D 80 C6 FA 4C 7D
DD|80B8:80 C9 8A D0 0F E6 FA A5
FE|80C0:FA C9 C0 D0 04 A9 00 85
8C|80C8:FA 4C 7D 80 C9 B0 90 0E
72|80D0:C9 B8 B0 14 38 E9 B0 AA
06|80D8:20 EC F6 4C 4B 80 C9 9B
A4|80E0:D0 06 20 55 82 4C 4B 80
D8|80E8:29 DF C9 C9 D0 10 A5 E3
13|80F0:D0 07 A9 BF 85 E3 4C 7D
DE|80F8:80 C6 E3 4C 7D 80 C9 CA
D2|8100:D0 1B A5 04 05 05 D0 0A
A5|8108:A9 17 85 04 A9 01 85 05
E8|8110:D0 B7 A5 04 D0 02 C6 05
B7|8118:C6 04 4C 7D 80 C9 CB D0
DD|8120:18 E6 04 D0 02 E6 05 A5
08|8128:04 C9 18 A5 05 E9 01 90
AD|8130:18 A9 00 85 04 85 05 F0
E1|8138:10 C9 CD D0 0F E6 E3 A5
52|8140:E3 C9 C0 D0 04 A9 00 85
6C|8148:E3 4C 7D 80 C9 C6 D0 16
BA|8150:AD FB 82 49 01 8D FB 82
43|8158:D0 06 2C 53 C0 4C 4B 80
BA|8160:2C 52 C0 4C 4B 80 C9 CC
9A|8168:D0 18 20 49 82 A4 CF A6
4C|8170:CE A5 FA 20 11 F4 A6 05
F6|8178:A5 04 A4 E3 20 3A F5 4C
2C|8180:4B 80 C9 C3 D0 09 20 49
72|8188:82 20 F2 F3 4C 4B 80 C9
52|8190:D0 D0 0E 20 49 82 AD FA
B1|8198:82 49 01 8D FA 82 4C 4B
CC|81A0:80 C9 D1 F0 19 E6 FB D0
C5|81A8:02 E6 FC A5 FB C9 00 A5
58|81B0:FC E9 20 B0 03 4C 4B 80
34|81B8:20 E9 81 4C 4B 80 20 49
FF|81C0:82 20 39 FB 20 58 FC A2
8B|81C8:E4 A0 82 20 38 82 A9 40
E9|81D0:A2 00 20 41 F9 A2 F5 A0
4A|81D8:82 20 38 82 A5 09 A6 08
81|81E0:20 41 F9 20 8E FD 4C D0
2A|81E8:03 AD F9 82 49 01 8D F9
68|81F0:82 A4 CF A6 CE A5 FA 20
FD|81F8:11 F4 A9 00 A0 82 A2 FC
29|8200:20 5D F6 A4 05 A6 04 A5
5B|8208:E3 20 11 F4 A9 00 A0 83
E5|8210:A2 02 20 5D F6 A9 00 85
59|8218:FB 85 FC 60 AD FA 82 D0
95|8220:01 60 A4 CF A6 CE A5 FA
E1|8228:4C 57 F4 A2 00 86 24 A9
61|8230:14 20 5B FB A2 6B A0 82
28|8238:86 FE 84 FF A0 00 B1 FE
9A|8240:F0 06 20 ED FD C8 D0 F6
97|8248:60 A9 00 85 1E A9 40 85
FB|8250:1F 20 00 60 60 A5 08 05
C5|8258:09 D0 01 60 20 F2 F3 A9
AD|8260:00 85 1E A9 40 85 1F 20
AF|8268:00 03 60 C5 D3 C3 BD D5
2D|8270:CE C4 CF A0 A0 B0 AD B7
E8|8278:BD C3 CF CC CF D2 A0 A0
04|8280:C6 BD C6 D5 CC CC AF CD
95|8288:D8 C5 C4 A0 A0 CC BD CC
0D|8290:C9 CE C5 C3 BD C3 CC C5
9B|8298:C1 D2 A0 D3 C3 D2 C5 C5
D4|82A0:CE A0 A0 A0 D1 BD D1 D5
B4|82A8:C9 D4 A0 A0 A0 A0 D0 BD
F2|82B0:D0 C5 CE A0 D5 D0 AF C4
30|82B8:CF D7 CE C9 CA CB CD BD
0A|82C0:CD CF D6 C5 A0 C1 CC D4
39|82C8:A0 C3 D5 D2 D3 CF D2 A0
39|82D0:A0 C1 D2 D2 CF D7 D3 BD
1C|82D8:CD CF D6 C5 A0 C3 D5 D2
63|82E0:D3 CF D2 00 C3 CF CD D0
AB|82E8:C1 C3 D4 A0 D0 C9 C3 D4
BF|82F0:BA A0 C1 A4 00 AC CC A4
4B|82F8:00 00 00 00 3A 24 2D 36
3A|8300:07 00 04 00
```

TOTAL: CFC1

END OF LISTING 6

LISTING 7: ARTIST.LOAD

```
37 |  10 REM  ************************
C0 |  20 REM  * ARTIST.LOAD           *
B9 |  30 REM  * BY S. SCOTT ZIMMERMAN *
AE |  40 REM  * COPYRIGHT(C) 1988     *
CB |  50 REM  * MICROSPARC, INC.      *
24 |  60 REM  * CONCORD, MA 01742     *
45 |  70 REM  ************************
FA |  80 D$ = CHR$ (4)
EB |  90 HOME : PRINT  CHR$ (21): REM  SWITCH TO 40
       COLUMNS
95 | 100  PRINT D$"BLOAD HRCOMP,A$6000"
D3 | 110  PRINT D$"BLOAD HREXP,A$300"
EC | 120  PRINT D$"BLOAD ARTIST,A$8000"
B1 | 130  CALL 32768
7E | 140  END
```

TOTAL: 97C3

END OF LISTING 7