HI · RES HOUDINI

Perform your own Hi-Res magic with this machine language utility. Working with both Hi-Res screens, it will let you invert colors, shift the image in all four directions, merge the two screens and much more.

by S. Scott Zimmerman, Ph.D.
1129 East 470 North
Orem, UT 84057

Do you ever get frustrated with Apple high-resolution graphics and wish for a little magic? While creating graphics pictures, do you ever feel you want to cast a spell to get out of a tight spot? Do you ever fantasize about some graphics sleight-of-hand, or dream of a near-impossible graphics stunt?

Well, then, ladies and gentlemen, here it is, for your graphics enjoyment (ta-dum): **HI-RES HOUDINI**.

HI-RES HOUDINI can perform the following tricks:

- Scroll your Hi-Res picture left, right, up or down on the graphics screen. This allows you to center a picture, to make room on the edge of the screen for more graphics, or to change the colors of the shapes.
- Invert the colors of your Hi-Res picture. This converts the picture from white dots (or some other color) on a black background to dots of the opposite color.
- Change the color bit of all the Hi-Res screen bytes. You can quickly see what the picture looks like in other colors.
- Flip between the two Hi-Res screens.
- Merge two Hi-Res pictures into one. Just load one picture onto Hi-Res screen 1, and the other onto screen 2, and, with the stroke of a key — presto! — they become one.
- Transfer the picture on Hi-Res screen 1 to Hi-Res screen 2, and vice versa.
- Copy a picture from one part of the screen to another.

Of course, HI-RES HOUDINI is not really magic, and it certainly can't do *everything* you dream about. But it can do a few tricks that you might find helpful and fun.

Commands

HI-RES HOUDINI is actually quite easy to use. When you BRUN or CALL it, all of the commands are listed at the bottom of the screen. The following is an explanation of each.

→ (right arrow): Scroll the screen right one byte. This moves everything on the Hi-Res screen to the right one byte (seven dots) and places the original right-most column on the left border. In other words, the Hi-Res picture "wraps around" — whatever scrolls off the right column of the screen, reappears on the left column of the screen. Since dots in odd-numbered columns move to even-numbered columns, the colors will change when this command is executed.

← (left arrow): Scroll the screen left one byte. This does the same thing as the right arrow, except in the opposite direction.

> ("greater than" sign) or **.** (period): Scroll the screen right one bit, i.e., one pixel or one dot. This is similar to the right-arrow scroll command, except that it scrolls the screen only one bit rather than one byte. Since this changes odd-numbered columns to even-numbered columns, the colors change when this command is executed.

< ("less than" sign) or **,** (comma): Scroll the screen left one bit. This is the same as the '>' command, except in the opposite direction.

A: Scroll the screen up. This moves everything on the Hi-Res screen up one dot. It too has "wrap-around" — whatever scrolls off the top of the screen, reappears at the bottom. Since the columns in which the dots are located do not change, the colors of the picture remain unchanged.

Z: Scroll the screen down. This does the same thing as A, but in the opposite direction.

I: Invert the Hi-Res colors. This command clears all of the Hi-Res bits that were set, and sets all the Hi-Res bits that were clear. This has the effect of interchanging white and black, green and blue, and violet and red. It lets you see your picture "black on white" rather than the usual "white on black."

C: Change the color bit. This command clears all the Hi-Res color bits (bit 7 of each byte) that were set, and sets all the Hi-Res color bits that were clear. It has the effect of interchanging green and red, and violet and blue. It also converts WHITE1 to WHITE2 and BLACK1 to BLACK2 (and vice versa), but you usually cannot tell that this has happened.

F: FULL/MIXED screen toggle. This command, when executed while viewing Hi-Res page 1, switches from MIXED Hi-Res graphics/text mode to FULL graphics mode, or vice versa. When you first BRUN HI.RES.HOUDINI, you will be in MIXED graphics/text mode, viewing the HI-RES HOUDINI commands at the bottom of the screen. But when you first press F, the text at the bottom of the screen will disappear and you will see the full screen in graphics. When you press F again, the text at the bottom of the screen will reappear. (This command does not work if you are viewing Hi-Res page 2, since normal text cannot be viewed from page 2.)

M: Merge Hi-Res pictures. This command overlays one Hi-Res screen onto the other by performing an "exclusive-or" with all the bytes of the two screens. In other words, one of the Hi-Res screens is XDRAWn onto the other. This has the effect of overlaying one picture on top of the other, but in such a way that if a page 1 dot (pixel) is "on" and is overlayed on a page 2 dot that is also "on," the result will be an "off" pixel. This allows you to merge two pictures in one operation and then "unmerge" them in a second operation, leaving the original picture unchanged. If you are viewing Hi-Res page 1 when you press M, Hi-Res page 2 will merge onto page 1 without affecting page 2. If you are viewing page 2 when you press M, page 1 will merge onto page 2, but page 1 will be unaffected.

CTRL-@: Clear the Hi-Res screen. This erases (to black) the currently viewed Hi-Res screen. Note that on the Apple II Plus, this command is a <CTRL> <SHIFT>P, and on the Apple //e, it is a <CTRL> <SHIFT>2. (I like having to press three keys to clear the screen; it helps avoid unfortunate mistakes.)

P: Flip the Hi-Res page. This command switches from Hi-Res page 1 to Hi-Res page 2, or vice versa. (If you haven't loaded a graphics picture onto one or both of the pages when you BRUN HI.RES.HOUDINI, you might see "garbage" when viewing that particular Hi-Res page. Alternatively, you will see a blank screen if you have cleared the Hi-Res page with an HGR or HGR2 command.)

Q: Quit. Pressing Q exits HI-RES HOUDINI and returns you to a calling program or Applesoft, depending on the state prior to starting HI-RES HOUDINI. To re-enter the program from Applesoft, type **CALL 36608**.

If you press any key other than a legal command key, you will hear an error "beep" apprising you of your mistake.

Once you press a key and the command is executed, HI-RES HOUDINI waits for you to press another key and to execute the next command. In this way, you can quickly move your picture from one location to another by successively pressing one or more of the scrolling keys. It also allows you to retract a command. For example, if you scroll too far left, you can immediately press the → (right arrow) or > ("greater-than" sign) to move back to the right. If you merge two Hi-Res pictures and don't like what you see, you can immediately press M again and "unmerge" the pictures.

Magic Tricks

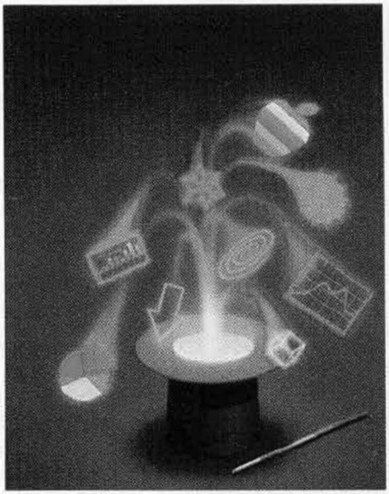
Now for a little magic. Here are three "tricks" you can do with the above set of commands:

Exchange the Hi-Res screens. Make sure you have a picture on both Hi-Res screens, then **BRUN HI.RES.HOUDINI** (or run the HOUDINI.DRIVER program). Now, while viewing page 1, press these keys in the following order:

```
M P M P M
```

You can verify that this sequence did indeed exchange the pictures by pressing P several times to switch back and forth between page 1 and page 2.

Make several copies of a shape on one screen. First, put the desired shape on Hi-Res page 1, with nothing else on the screen. Second, **BRUN HI.RES.HOUDINI** or, if it is in memory, **CALL 36608** (or run the driver program). Third, press **P** to view page 2, followed by **<CTRL>@** to clear it. Fourth, press **M** to merge page 1 onto page 2. Fifth, using the scroll commands, move the shape to a new location on page 2. Sixth, press **P** to get back to page 1. And finally, press **M** to merge page 2 onto page 1. The end result is that the original shape is now found *twice* on page 1, in two different locations. You can repeat this operation as often as you like to make multiple copies of the shape.



Create special effects. This is where your imagination can run wild. Try this: put a Hi-Res picture on page 1, press **P** to view page 2, **<CTRL>@** to clear that page, then **M** to merge (move) page 1 onto page 2. Now scroll the screen one dot right or left by pressing > or <. Finally, press **M** to merge the pictures. What you see depends on what was there to begin with. If you don't like what you see, just press M again, and you will "unmerge" the pictures. Try another: With a Hi-Res picture on one of the screens, type the sequence **I C >**. This has the effect of changing the background color (from black to white or white to black) *without* changing the colors of the shapes on the screen.

With these examples, you should be prepared to invent some of your own magic.

Running HI-RES HOUDINI

Before you use HI-RES HOUDINI, you will want to create a Hi-Res picture on one or both of the Hi-Res screens. You can draw the picture with a commercially available graphics utility or with a published program, such as "Apple Artist" by Tony Dahbura (*Nibble* Vol. 2/No. 6), or "The Apple Art Gallery" by Edgar Young (*Nibble* Vol. 3/No. 6), or you can simply draw your pictures with HPLOT, DRAW, and/or XDRAW commands.

Once the object code of HI.RES.HOUDINI is on disk, simply type **BRUN HI.RES.HOUDINI** to execute the program. If HI.RES.HOUDINI is already in memory, you can run it by typing **CALL 36608**.

To simplify this process, you may want to use the program, HOUDINI.DRIVER shown in **Listing 2**. To use it, you will need to save it to a disk that contains HI.RES.HOUDINI and one or more picture files. The driver program will first prompt you for the names of the picture files to be loaded on Hi-Res pages 1 and 2, and then load them and start HI-RES HOUDINI. When you quit HI-RES HOUDINI, the driver will then prompt you for confirmation, ask whether you want to start again with different picture files, and give you an opportunity to save the products of your labor on disk.

Entering HI-RES HOUDINI

Listing 1 gives the assembly language source code for **HI.RES.HOUDINI**. It was written using macros, which simplify entering often-repeated code. If you have the BIG MAC assembler from A.P.P.L.E. or the Merlin assembler from Southwestern Data Systems, type the code as it appears in the listing, omitting the macro code lines in the body of the program, i.e., type only the *first* line of any group of lines having the same line number. The macro code is also identified in the program by comments beginning with ";>>".

If your assembler does *not* have macro capabilities, omit the section labeled "Define MACROs." Then, within the body of the code listing, omit the lines with the ">>>" directive (which means "put macro"), but include the actual macro coding, i.e., the lines with the comments marked with ";>>". In other words, omit the first line in a group of lines having the same line number, but include all the other lines.

If you do not have an assembler, the machine code can be entered directly into the Monitor as explained in "A Welcome to New *Nibble* Readers" at the beginning of this issue. After entering the hexadecimal data, type:

```
BSAVE HI.RES.HOUDINI, A$8F00, L$669
```

HOUDINI.DRIVER (**Listing 2**) is an Applesoft program. After entering this program save it to disk with the command:

SAVE HOUDINI.DRIVER

De-mystifying the Magic

The magician, Harry Houdini (1874-1926), after whom this program is named, is famous not only for his sleight-of-hand and his ability to escape from tight situations, but also for his open explanations of how the tricks were performed. Likewise, I have tried to organize and document the program HI.RES .HOUDINI to remove some of the mystery from programming with high-resolution graphics.

The first two sections of the program define the constants and variables. The program differentiates between variables (**lines 21-28**) and constants (**lines 34-42**) by using the different pseudo-opcodes 'EQU' and ' = '. A constant is a symbol (name) that represents a *number*, not an address; a variable is a symbol that represents an *address* in memory. Of course the assembler treats 'EQU' and ' = ' identically, but the programmer needs to keep the difference between a constant and a variable in mind at all times.

Lines 48-64 define the ROM addresses and routines that will be used. The use of Applesoft or Monitor routines saves time, effort, and memory. For example, the Applesoft zero-page location $E6 (dec 230), called HPAGE, contains a byte to indicate the current Hi-Res screen. If that location contains $20 (dec 32), then the current Hi-Res screen is page 1 (the starting address of page 1 is $2000; $20 is the high-order byte of $2000). If HPAGE contains $40 (dec 64), then the current Hi-Res screen is page 2 (starting address $4000). This means that Applesoft "looks" at that location before executing, for example, an HPLOT or DRAW command to determine the proper Hi-Res page. There is also a routine, HCLR, at $F3F2 (dec 62450) that clears the current Hi-Res screen. So, in HI.RES.HOUDINI, when you want to clear a screen (<CTRL>@), the program simply sets HPAGE to the appropriate value (see **lines 678-683**) and does a JSR (Jump to SubRoutine) to HCLR.

For an explanation of the other Applesoft or Monitor routines given in **lines 48-64**, consult one of the references in the bibliography at the end of this article.

Lines 66-110 define the macros, and have already been explained.

In the first section of the actual program (starting with **line 116**), the system is initialized. First, the various graphics hardware "switches" are accessed to select high-resolution graphics, page 1, and mixed graphics/text. (Many programmers use the BIT command to access the switches, but LDA or any other memory access command could be used.)

Second, the Hi-Res page variable HRPAGE is set to zero to indicate page 1 (it is set to $20 when you flip to page 2 with the P command). I could have used the Applesoft location HPAGE and set it to $20 to indicate page 1 and $40 to indicate page 2, but I prefer using my own variable.

Next the FMFLAG (Full/Mixed flag) is set for mixed graphics. This is used in the program by the F command to allow toggling between full and mixed graphics mode.

Finally in the initialization section, the commands are printed at the bottom of the screen. I feel this is important. When I need a program utility, I dislike having to search for the documentation just to recall a few command keys. I prefer having the commands listed *within* the program, so they are ready for use immediately.

> **"...whatever scrolls off the right column of the screen, reappears on the left column..."**

The next section of the program is the Keyboard Command Input. In a straightforward way, the program checks all the possible keyboard commands, and jumps to the appropriate location if a correct key is pressed. If a wrong command is given, the program sounds the error "beep" (**line 185**) and jumps back (**line 186**) to check for another key press. If Q is pressed, the program quits (**lines 188-191**) by flipping to page 1, running the Applesoft subroutine SETTXT (which is simply the Applesoft TEXT command), and then exiting with an RTS to the status of the Apple prior to running the program, whether that is Applesoft, the Monitor, or a calling program.

The rest of the program contains a section for each of the keyboard input commands. They make use of the Hi-Res screen row addresses given near the end of the listing. This data gives the starting address of every line of Hi-Res page 1. Addresses for Hi-Res page 2 are obtained by adding $20 to the high-order byte value. With this data table, the manipulations of the Hi-Res screen bytes can be carried out in a fast and simple manner. Most arcade games include a similar table. In fact, you may want to save the data table as a separate file for use in other graphics programs.

Headings for each section and comments on each line should help you understand most of the program code.

Color Capers

As you know, the Apple uses bit-mapped graphics. This means that when the computer is in graphics mode, the Apple scans a specific region of memory ($2000-$3FFF or $4000-$5FFF) for "on" or "off" bits within each byte. If the bit is "on" (that is, it has a value of one), a dot appears on the screen at a location corresponding to the memory location of the bit. If the bit is "off" (has a value of zero), no dot appears on the screen for that location. Page 21 of the *Apple II Reference Manual* (page 34 of the *Apple //e Reference Manual*) gives the map of the high-resolution graphics screen, showing which memory byte corresponds to which graphics screen location.

The system gets more complicated when color is involved. What I have said about graphics bits in the preceding paragraph applies only to bits 0 through 6 of each byte. Bit 7 is the color bit. It does not correspond to an "on" or "off" pixel on the graphics screen, but rather affects the color of each bit within its byte. When the color bit is clear (has a value of zero), the pixels (corresponding to the bits within that byte) that are found in *even* columns on the Hi-Res screen (X = 0, 2, 4...) are violet in color, and the pixels found in *odd* columns (X = 1, 3, 5...) are green. When the color bit is set (has a value of one), the pixels in *even* columns are blue and the pixels in *odd* columns are red. Whenever two adjacent dots are "on," the color is white (WHITE1 if the color bit is clear; WHITE2 if the color bit is set).

So now we get down to the problem in HI-RES HOUDINI. When the Hi-Res screen is scrolled by one dot, what happens to the color bit? Let's take two adjacent bytes. The one on the left we'll call byte A, and the one on the right we'll call byte B. If A has the color bit set and B has the color bit clear, what happens to the color bit when you press ' > ' in HI-RES HOUDINI to scroll the screen one pixel to the right? And when a pixel from byte A moves into byte B, what happens to its color?

HI-RES HOUDINI handles these problems in the following way: When the graphics bits are shifted from byte A to B, the color bit goes into B if, first, an "on" pixel actually moves from A to B, and second, if B had no pixels turned "on." If, on the other hand, no dot is shifted from A into B, obviously B keeps its own color bit; or if B still has some of its original pixels turned on after the shift, it again keeps its own color bit.

This algorithm (or any algorithm I can think of) is not without its problems. In adventure-game jargon, we can say that the "color alignment is chaotic good." It's not "evil" since it's not out to injure us (although we may sometimes think otherwise when programming Apple graphics). But it's not "lawful" either; it is "chaotic" in the sense that some single-dot scrolls will cause unexpected color changes in some of the shapes. This is especially apt to occur when scrolling colored shapes on a white background. (Incidentally, scrolling a full byte at a time with the arrow commands avoids *unexpected* color changes, although it does cause some *expected* ones.)

The moral to all this, I guess, is that you have to be careful how you apply your magic.

"...the commands are listed within the program, so they are ready for use immediately."

Customization

If you don't like the keys I have chosen for the commands, change them! For example, you might not like the purposefully difficult method for clearing the Hi-Res screen, but would rather just press E for erase. To do this, change **line 183** to CMP #"E". This revision is enough to change the command itself, although the message in the text window at the bottom of the screen will be wrong. To change the message, revise **line 732** from CTRL- to E. You will also have to omit **lines 140-142**, which were required for printing an inverse sentinel (@) on the Apple screen; this special code was required since the ASCII code for inverse @ is zero, which is also the value used by the MESSAGE macro to indicate end-of-message.

Another modification you might want to make is in the method used to merge the two graphics screens. The program was written to do an exclusive-or (analogous to the Applesoft XDRAW command) rather than a logical-or (analogous to the DRAW command). If you want the merge to DRAW one picture on top of the other, change **line 623** from EOR to ORA.

If you don't feel you will use one or more of the commands, and you don't want to type in the corresponding section of code, just leave it out. The various sections are clearly identified.

If you know assembly language, you can obviously add other sections and other commands. For example, a simple addition would be commands for scrolling *without* wraparound. They would be used for quickly erasing parts of the screen. The coding would be the same as for other scrolls, except that, rather than restoring the last row or column to the first, they would just erase the first row or column. Anything scrolled off the screen would be gone forever.

Summary

HI-RES HOUDINI is a utility that will help you solve graphics problems or just have fun. With proper use of its simple commands, you will be able to perform your own graphics magic.

LISTING 1: HI.RES.HOUDINI

```
1      ************************************************
2      *                                              *
3      *               HI.RES.HOUDINI                 *
4      *                                              *
5      *            by S. Scott Zimmerman             *
6      *                                              *
7      *             Copyright (c) 1984               *
8      *             by MicroSparc, Inc.              *
9      *             Concord, MA  Ø1742               *
10     *                                              *
11     *             Assembler:  BIG MAC              *
12     *                                              *
13     ************************************************
14
15              ORG  $8FØØ       ;Up by DOS (dec 366Ø8)
16
17     ************************************************
18     *  Variables:                                  *
19     ************************************************
20
21     FMFLAG    EQU  Ø           ;FULL/MIXED flag
22     COLORBIT  EQU  1           ;To save the color bit
23     NEWBYTE   EQU  2           ;Temp data storage
24     HRPAGE    EQU  $19         ;Which hi-res page
25     OLDPTR    EQU  $1A         ;Points to old HR byte
26     NEWPTR    EQU  $1C         ;Points to new HR byte
27     ROW       EQU  $1E         ;Current row number
28     COL       EQU  $1F         ;Current column number
29
30     ************************************************
31     *  Constants:                                  *
32     ************************************************
33
34     RIT_ARROW =    $95         ;ASCII for right arrow
35     LFT_ARROW =    $88         ;ASCII for left arrow
36     UP_A      =    "A"         ;ASCII for A
37     DOWN_Z    =    "Z"         ;ASCII for Z
38     CTRL_@    =    $8Ø         ;ASCII for CTRL-@
39
40     NUMCOL    =    39          ;Number of byte columns
41     TOP_ROW   =    Ø           ;Top row is Øth row
42     BOT_ROW   =    191         ;Bottom row is 191
43
44     ************************************************
45     *  ROM addresses and routines:                 *
46     ************************************************
47
48     CH        EQU  $24         ;Horizontal TAB value
49     HPAGE     EQU  $E6         ;Hi-res page indicator
50     APLSOFT   EQU  $3DØ        ;Enter Applesoft
51     KEYBD     EQU  $CØØØ       ;Keyboard input location
52     STROBE    EQU  $CØ1Ø       ;Clear keyboard input
53     SHOW      EQU  $CØ5Ø       ;Display GRAPHICS screen
54     FULLSCRN  EQU  $CØ52       ;Display FULL graphics
55     MXEDSCRN  EQU  $CØ53       ;Display MIXED graph/text
56     FLIP1     EQU  $CØ54       ;Display screen #1
57     FLIP2     EQU  $CØ55       ;Display screen #2
58     HRSCRN    EQU  $CØ57       ;Display Hi-Res graphics
59     HCLR      EQU  $F3F2       ;Clear hi-res page
60     SETTXT    EQU  $FB39       ;Set TEXT mode
61     TABV      EQU  $FB5B       ;Vertical TAB routine
62     HOME      EQU  $FC58       ;Clear the text screen
63     COUT      EQU  $FDED       ;Character output routine
64     BELL      EQU  $FF3A       ;Sounds the 'beep'
65
66     ************************************************
67     *  Define of MACROs:                           *
68     ************************************************
69     *                                              *
70     * This section contains macros, used to simplify *
71     *  entry of commonly used code lines.  If your  *
72     *  assembler does not support macros, leave out *
73     *  this section; then, in the main body of the  *
74     *  code, omit the lines with the >>> directive, *
75     *  and insert lines containing the ;>> comments. *
76     *  If your assembler supports macros, type this *
77     *  section.  Then in the main body, type the    *
78     *  lines with the >>> directive, but omit lines *
79     *  with the ;>> comments.                      *
80     *                                              *
       ************************************************
82
83              DO   Ø           ;Do not assemble macros
84
85     SETPTR   MAC              ;>> Sets a pointer
86              LDA  YLOW,X      ;>> Get the Hi-Res LOB
87              STA  ]1          ;>> Store in pointer
88              CLC              ;>> Prepare for addition
89              LDA  YHIGH,X     ;>> Get the Hi-Res HOB
90              ADC  HRPAGE      ;>> Add for hi-res page
91              STA  ]1+1        ;>> And store it too
92              <<<              ;>> End of macro
93
94     TABXY    MAC              ;>> Move cursor to X,Y
95              LDA  #]1         ;>> Get the X value
96              STA  CH          ;>> Store it
97              LDA  #]2         ;>> Get the Y value
98              JSR  TABV        ;>> Go set vertical tab
99              <<<              ;>> End of macro
100
101    MESSAGE  MAC              ;>> Send a message
102             LDY  #Ø          ;>> Set the index
103    MSGLOOP  LDA  ]1,Y        ;>> Get a character
104             BEQ  MEND        ;>> Is it done?
105             JSR  COUT        ;>> No, send character
106             INY              ;>> Go to next character
107             BNE  MSGLOOP     ;>> (always branch)
108    MEND     <<<              ;>> End of macro
109
110             FIN              ;End macros; start assem
111
112    ************************************************
113    *  Initialize:                                 *
114    ************************************************
115
```

Bibliography

1. Apple Computer, Inc., *Apple II Monitors Peeled*, 1981, especially Chapter Two. This book is a must for those using Monitor routines from assembly language.

2. Apple Computer, Inc., *Apple II Reference Manual*, 1979, pp. 19-21, 61-62, and 130-131. You would be amazed at what you can learn from this manual, which has been sitting there, mostly ignored and unopened, right on your desk all this time.

3. Crossley, John, "Applesoft Internal Entry Points," originally published in *Apple Orchard*; reprinted in *All About Applesoft*, A.P.P.L.E., 1982. A classic article on the use of ROM routines.

4. Luebbert, William F., *What's Where in the Apple* (with the new user's guide), 1982.

5. So, Edward C., "Hi-Res Full Scroll," *Call-A.P.P.L.E.*, Vol. 5/No. 2, February 1982, pp. 23-34. This was a source of inspiration, although HI-RES HOUDINI uses a different algorithm and of course has many other commands.

```
8FØØ: 2C 57 CØ  116          BIT  HRSCRN      ;Select hi-res screen
8FØ3: 2C 54 CØ  117          BIT  FLIP1       ;Make sure it's page 1
8FØ6: 2C 53 CØ  118          BIT  MXEDSCRN    ;Make it mixed GR/TEXT
8FØ9: 2C 5Ø CØ  119          BIT  SHOW        ;Now show the hi-res scrn
8FØC: 2C 1Ø CØ  120          BIT  STROBE      ;Clear keyboard input
8FØF: A9 ØØ     121          LDA  #Ø          ;Make default
8F11: 85 19     122          STA  HRPAGE      ;Hi-res page 1
8F13: 85 ØØ     123          STA  FMFLAG      ;Set flag to mixed GR
                124
                125 * Print keyboard commands at bottom of screen:
                126
8F15: 2Ø 58 FC  127          JSR  HOME        ;Clear the screen
                128          >>>  TABXY.Ø;2Ø
8F18: A9 ØØ     128          LDA  #Ø          ;>> Get the X value
8F1A: 85 24     128          STA  CH          ;>> Store it
8F1C: A9 14     128          LDA  #2Ø         ;>> Get the Y value
8F1E: 2Ø 5B FB  128          JSR  TABV        ;>> Go set vertical tab
                128          <<<              ;>> End of macro
                129          >>>  MESSAGE.SCLCMNDS
8F21: AØ ØØ     129          LDY  #Ø          ;>> Set the index
8F23: B9 D6 94  129 MSGLOOP  LDA  SCLCMNDS,Y  ;>> Get a character
8F26: FØ Ø6     129          BEQ  MEND        ;>> Is it done?
8F28: 2Ø ED FD  129          JSR  COUT        ;>> No, send character
8F2B: C8        129          INY              ;>> Go to next character
8F2C: DØ F5     129          BNE  MSGLOOP     ;>> (always branch)
                129 MEND     <<<              ;>> End of macro
                13Ø          >>>  TABXY.Ø;21
8F2E: A9 ØØ     13Ø          LDA  #Ø          ;>> Get the X value
8F3Ø: 85 24     13Ø          STA  CH          ;>> Store it
8F32: A9 15     13Ø          LDA  #21         ;>> Get the Y value
8F34: 2Ø 5B FB  13Ø          JSR  TABV        ;>> Go set vertical tab
                13Ø          <<<              ;>> End of macro
                131          >>>  MESSAGE.INV:I
8F37: AØ ØØ     131          LDY  #Ø          ;>> Set the index
8F39: B9 FA 94  131 MSGLOOP  LDA  INV:I,Y     ;>> Get a character
8F3C: FØ Ø6     131          BEQ  MEND        ;>> Is it done?
8F3E: 2Ø ED FD  131          JSR  COUT        ;>> No, send character
8F41: C8        131          INY              ;>> Go to next character
8F42: DØ F5     131          BNE  MSGLOOP     ;>> (always branch)
                131 MEND     <<<              ;>> End of macro
                132          >>>  TABXY.2Ø;21
8F44: A9 14     132          LDA  #2Ø         ;>> Get the X value
8F46: 85 24     132          STA  CH          ;>> Store it
8F48: A9 15     132          LDA  #21         ;>> Get the Y value
8F4A: 2Ø 5B FB  132          JSR  TABV        ;>> Go set vertical tab
                132          <<<              ;>> End of macro
                133          >>>  MESSAGE.CHGBIT:C
8F4D: AØ ØØ     133          LDY  #Ø          ;>> Set the index
8F4F: B9 ØB 95  133 MSGLOOP  LDA  CHGBIT:C,Y  ;>> Get a character
8F52: FØ Ø6     133          BEQ  MEND        ;>> Is it done?
8F54: 2Ø ED FD  133          JSR  COUT        ;>> No, send character
8F57: C8        133          INY              ;>> Go to next character
8F58: DØ F5     133          BNE  MSGLOOP     ;>> (always branch)
                133 MEND     <<<              ;>> End of macro
                134          >>>  TABXY.Ø;22
8F5A: A9 ØØ     134          LDA  #Ø          ;>> Get the X value
8F5C: 85 24     134          STA  CH          ;>> Store it
8F5E: A9 16     134          LDA  #22         ;>> Get the Y value
8F6Ø: 2Ø 5B FB  134          JSR  TABV        ;>> Go set vertical tab
                134          <<<              ;>> End of macro
```

```
                135          >>>  MESSAGE.FLMX:F
8F63: AØ ØØ     135          LDY  #Ø          ;>> Set the index
8F65: B9 1E 95  135 MSGLOOP  LDA  FLMX:F,Y    ;>> Get a character
8F68: FØ Ø6     135          BEQ  MEND        ;>> Is it done?
8F6A: 2Ø ED FD  135          JSR  COUT        ;>> No, send character
8F6D: C8        135          INY              ;>> Go to next character
8F6E: DØ F5     135          BNE  MSGLOOP     ;>> (always branch)
                135 MEND     <<<              ;>> End of macro
                136          >>>  TABXY.2Ø;22
8F7Ø: A9 14     136          LDA  #2Ø         ;>> Get the X value
8F72: 85 24     136          STA  CH          ;>> Store it
8F74: A9 16     136          LDA  #22         ;>> Get the Y value
8F76: 2Ø 5B FB  136          JSR  TABV        ;>> Go set vertical tab
                136          <<<              ;>> End of macro
                137          >>>  MESSAGE.MERGE:M
8F79: AØ ØØ     137          LDY  #Ø          ;>> Set the index
8F7B: B9 32 95  137 MSGLOOP  LDA  MERGE:M,Y   ;>> Get a character
8F7E: FØ Ø6     137          BEQ  MEND        ;>> Is it done?
8F8Ø: 2Ø ED FD  137          JSR  COUT        ;>> No, send character
8F83: C8        137          INY              ;>> Go to next character
8F84: DØ F5     137          BNE  MSGLOOP     ;>> (always branch)
                137 MEND     <<<              ;>> End of macro
                138          >>>  TABXY.Ø;23
8F86: A9 ØØ     138          LDA  #Ø          ;>> Get the X value
8F88: 85 24     138          STA  CH          ;>> Store it
8F8A: A9 17     138          LDA  #23         ;>> Get the Y value
8F8C: 2Ø 5B FB  138          JSR  TABV        ;>> Go set vertical tab
                138          <<<              ;>> End of macro
                139          >>>  MESSAGE.CLEAR:@
8F8F: AØ ØØ     139          LDY  #Ø          ;>> Set the index
8F91: B9 42 95  139 MSGLOOP  LDA  CLEAR:@,Y   ;>> Get a character
8F94: FØ Ø6     139          BEQ  MEND        ;>> Is it done?
8F96: 2Ø ED FD  139          JSR  COUT        ;>> No, send character
8F99: C8        139          INY              ;>> Go to next character
8F9A: DØ F5     139          BNE  MSGLOOP     ;>> (always branch)
                139 MEND     <<<              ;>> End of macro
                14Ø          >>>  TABXY.5;23
8F9C: A9 Ø5     14Ø          LDA  #5          ;>> Get the X value
8F9E: 85 24     14Ø          STA  CH          ;>> Store it
8FAØ: A9 17     14Ø          LDA  #23         ;>> Get the Y value
8FA2: 2Ø 5B FB  14Ø          JSR  TABV        ;>> Go set vertical tab
                14Ø          <<<              ;>> End of macro
8FA5: A9 ØØ     141          LDA  #$Ø         ;Get inverse @ (hex ØØ)
8FA7: 2Ø ED FD  142          JSR  COUT        ;And print it
                143          >>>  TABXY.2Ø;23
8FAA: A9 14     143          LDA  #2Ø         ;>> Get the X value
8FAC: 85 24     143          STA  CH          ;>> Store it
8FAE: A9 17     143          LDA  #23         ;>> Get the Y value
8FBØ: 2Ø 5B FB  143          JSR  TABV        ;>> Go set vertical tab
                143          <<<              ;>> End of macro
                144          >>>  MESSAGE.PAGE:P
8FB3: AØ ØØ     144          LDY  #Ø          ;>> Set the index
8FB5: B9 56 95  144 MSGLOOP  LDA  PAGE:P,Y    ;>> Get a character
8FB8: FØ Ø6     144          BEQ  MEND        ;>> Is it done?
8FBA: 2Ø ED FD  144          JSR  COUT        ;>> No, send character
8FBD: C8        144          INY              ;>> Go to next character
8FBE: DØ F5     144          BNE  MSGLOOP     ;>> (always branch)
                144 MEND     <<<              ;>> End of macro
                145          >>>  TABXY.33;23
8FCØ: A9 21     145          LDA  #33         ;>> Get the X value
8FC2: 85 24     145          STA  CH          ;>> Store it
8FC4: A9 17     145          LDA  #23         ;>> Get the Y value
8FC6: 2Ø 5B FB  145          JSR  TABV        ;>> Go set vertical tab
                145          <<<              ;>> End of macro
                146          >>>  MESSAGE.QUIT:Q
8FC9: AØ ØØ     146          LDY  #Ø          ;>> Set the index
8FCB: B9 62 95  146 MSGLOOP  LDA  QUIT:Q,Y    ;>> Get a character
8FCE: FØ Ø6     146          BEQ  MEND        ;>> Is it done?
8FDØ: 2Ø ED FD  146          JSR  COUT        ;>> No, send character
8FD3: C8        146          INY              ;>> Go to next character
8FD4: DØ F5     146          BNE  MSGLOOP     ;>> (always branch)
                146 MEND     <<<              ;>> End of macro
                147
                148 ***********************************************
                149 *  Keyboard command input:                    *
                15Ø ***********************************************
                151
8FD6: AD ØØ CØ  152 KEYIN    LDA  KEYBD       ;Has a key been pressed?
8FD9: 1Ø FB     153          BPL  KEYIN       ;No, go check again
8FDB: 2C 1Ø CØ  154          BIT  STROBE      ;Yes, clear strobe
8FDE: C9 D1     155          CMP  #"Q"        ;Quit?
8FEØ: FØ 3E     156          BEQ  QUIT        ;Yes, so quit now
8FE2: C9 C1     157          CMP  #UP_A       ;Scroll up?
8FE4: FØ 44     158          BEQ  SCRLUP      ;Yes, go scroll up
8FE6: C9 DA     159          CMP  #DOWN_Z     ;Scroll down?
8FE8: FØ 43     16Ø          BEQ  SCRLDWN     ;Yes, go scroll down
8FEA: C9 95     161          CMP  #RIT_ARROW  ;Scroll right?
8FEC: FØ 42     162          BEQ  SCRLRIT     ;Yes, go scroll right
8FEE: C9 88     163          CMP  #LFT_ARROW  ;Scroll left?
8FFØ: FØ 41     164          BEQ  SCRLLFT     ;Yes, go scroll left
8FF2: C9 BE     165          CMP  #">"        ;Move right one pixel?
8FF4: FØ 4Ø     166          BEQ  MOVERIGHT   ;Yes, go do it
8FF6: C9 AE     167          CMP  #"."        ;Move right one pixel?
```

```
8FF8: FØ 3C                BEQ  MOVERIGHT     ;Yes, go move right
8FFA: C9 BC                CMP  #"<"          ;Move left one pixel?
8FFC: FØ 3B                BEQ  MOVELEFT      ;Yes, go move left
8FFE: C9 AC                CMP  #","          ;Move left one pixel?
9ØØØ: FØ 37                BEQ  MOVELEFT      ;Yes, go move left
9ØØ2: C9 C9                CMP  #"I"          ;Set inverse?
9ØØ4: FØ 36                BEQ  INVERSE       ;Yes, go set inverse
9ØØ6: C9 C3                CMP  #"C"          ;Change the color bit?
9ØØ8: FØ 35                BEQ  CHHCLBIT      ;Yes, go change it
9ØØA: C9 DØ                CMP  #"P"          ;Change the hi-res page?
9ØØC: FØ 34                BEQ  PAGECHNG      ;Yes, go change page
9ØØE: C9 C6                CMP  #"F"          ;Toggle FULL/MIXED screen?
9Ø1Ø: FØ 33                BEQ  FULLMXD       ;Yes, go toggle it
9Ø12: C9 CD                CMP  #"M"          ;Merge page 1 into page 2?
9Ø14: FØ 32                BEQ  PICMERGE      ;Yes, go merge
9Ø16: C9 8Ø                CMP  #CTRL_@       ;Clear screen?
9Ø18: FØ 31                BEQ  CLEARSCRN     ;Yes, go clear it
9Ø1A: 2Ø 3A FF             JSR  BELL          ;Wrong key; sound bell
9Ø1D: 4C D6 8F             JMP  KEYIN         ;Go check for right key

9Ø2Ø: 2C 54 CØ   QUIT      BIT  FLIP1         ;Set page 1
9Ø23: 2Ø 39 FB             JSR  SETTXT        ;Go back to text mode
9Ø26: 2Ø 58 FC             JSR  HOME          ;Clear the screen
9Ø29: 6Ø                   RTS                ;End of HIRES HELPER

                 * JUMP table:

9Ø2A: 4C 4E 9Ø   SCRLUP    JMP  SCROLL_UP      ;Go scroll up
9Ø2D: 4C CE 9Ø   SCRLDWN   JMP  SCROLL_DOWN    ;Go scroll down
9Ø3Ø: 4C 58 91   SCRLRIT   JMP  SCROLL_RIGHT   ;Go scroll right
9Ø33: 4C EC 91   SCRLLFT   JMP  SCROLL_LEFT    ;Go scroll left
9Ø36: 4C 87 91   MOVERIGHT JMP  MOVE_RIGHT     ;Go move pixel right
9Ø39: 4C 1D 92   MOVELEFT  JMP  MOVE_LEFT      ;Go move pixel left
9Ø3C: 4C 8A 92   INVERSE   JMP  SET_INVERSE    ;Go inverse colors
9Ø3F: 4C AC 92   CHHCLBIT  JMP  CHG_COLOR_BIT  ;Go change color bit
9Ø42: 4C CE 92   PAGECHNG  JMP  CHANGE_PAGE    ;Go change page
9Ø45: 4C FØ 92   FULLMXD   JMP  TOGGL_FULLMXD  ;Go toggle full/mixd
9Ø48: 4C ØE 93   PICMERGE  JMP  PICTURE_MERGE  ;Go merge pictures
9Ø4B: 4C 49 93   CLEARSCRN JMP  CLEAR_SCREEN   ;Go clear screen

                 ****************************************************
                 *  Scroll up:                                      *
                 ****************************************************

                 SCROLL_UP

                 * Store top in buffer for later restore at bottom:

9Ø4E: A9 69                LDA  #<BUFFER      ;Get the buffer LOB
9Ø5Ø: 85 1C                STA  NEWPTR        ;Set new location LOB
9Ø52: A9 95                LDA  #>BUFFER      ;Get the buffer HOB
9Ø54: 85 1D                STA  NEWPTR+1      ;And save it in pointer

9Ø56: A2 ØØ                LDX  #TOP_ROW       ;Set old location to
                           >>>  SETPTR.OLDPTR  ; top row
9Ø58: BD 56 93             LDA  YLOW,X        ;>> Get the Hi-Res LOB
9Ø5B: 85 1A                STA  OLDPTR        ;>> Store in pointer
9Ø5D: 18                   CLC                ;>> Prepare for addition
9Ø5E: BD 16 94             LDA  YHIGH,X       ;>> Get the Hi-Res HOB
9Ø61: 65 19                ADC  HRPAGE        ;>> Add for hi-res page
9Ø63: 85 1B                STA  OLDPTR+1      ;>> And store it too
                           <<<                ;>> End of macro

9Ø65: 2Ø 4E 91             JSR  MOVEROW       ;Go move the row

                 * Move the next rows up:

9Ø68: A2 ØØ                LDX  #TOP_ROW      ;Start at top of row
9Ø6A: 86 1E                STX  ROW           ;Save in row counter

                           >>>  SETPTR.NEWPTR ;Set pntr to new row
9Ø6C: BD 56 93             LDA  YLOW,X        ;>> Get the Hi-Res LOB
9Ø6F: 85 1C                STA  NEWPTR        ;>> Store in pointer
9Ø71: 18                   CLC                ;>> Prepare for addition
9Ø72: BD 16 94             LDA  YHIGH,X       ;>> Get the Hi-Res HOB
9Ø75: 65 19                ADC  HRPAGE        ;>> Add for hi-res page
9Ø77: 85 1D                STA  NEWPTR+1      ;>> And store it too
                           <<<                ;>> End of macro
9Ø79: E6 1E                INC  ROW           ;Go to next row
9Ø7B: A6 1E                LDX  ROW           ;Put counter in register
                           >>>  SETPTR.OLDPTR ;Set pntr to old row
9Ø7D: BD 56 93             LDA  YLOW,X        ;>> Get the Hi-Res LOB
9Ø8Ø: 85 1A                STA  OLDPTR        ;>> Store in pointer
9Ø82: 18                   CLC                ;>> Prepare for addition
9Ø83: BD 16 94             LDA  YHIGH,X       ;>> Get the Hi-Res HOB
9Ø86: 65 19                ADC  HRPAGE        ;>> Add for hi-res page
9Ø88: 85 1B                STA  OLDPTR+1      ;>> And store it too
                           <<<                ;>> End of macro

9Ø8A: 2Ø 4E 91   UP:LOOP   JSR  MOVEROW       ;Go move the row

9Ø8D: A6 1E                LDX  ROW           ;Get counter again
                           >>>  SETPTR.NEWPTR ;Set new pointer
9Ø8F: BD 56 93             LDA  YLOW,X        ;>> Get the Hi-Res LOB
9Ø92: 85 1C                STA  NEWPTR        ;>> Store in pointer
9Ø94: 18                   CLC                ;>> Prepare for addition
9Ø95: BD 16 94             LDA  YHIGH,X       ;>> Get the Hi-Res HOB
9Ø98: 65 19                ADC  HRPAGE        ;>> Add for hi-res page
9Ø9A: 85 1D                STA  NEWPTR+1      ;>> And store it too
                           <<<                ;>> End of macro
9Ø9C: E6 1E                INC  ROW           ;Go to next row
9Ø9E: A6 1E                LDX  ROW           ;Put counter in register
                           >>>  SETPTR.OLDPTR ;Set pntr to old row
9ØAØ: BD 56 93             LDA  YLOW,X        ;>> Get the Hi-Res LOB
9ØA3: 85 1A                STA  OLDPTR        ;>> Store in pointer
9ØA5: 18                   CLC                ;>> Prepare for addition
9ØA6: BD 16 94             LDA  YHIGH,X       ;>> Get the Hi-Res HOB
9ØA9: 65 19                ADC  HRPAGE        ;>> Add for hi-res page
9ØAB: 85 1B                STA  OLDPTR+1      ;>> And store it too
                           <<<                ;>> End of macro

9ØAD: EØ CØ                CPX  #BOT_ROW+1    ;Is it past bottom row?
9ØAF: 9Ø D9                BCC  UP:LOOP       ;No, so go do another row

                 * Restore old top row to new bottom row:

9ØB1: A9 69                LDA  #<BUFFER      ;Set pointer to buffer
9ØB3: 85 1A                STA  OLDPTR
9ØB5: A9 95                LDA  #>BUFFER
9ØB7: 85 1B                STA  OLDPTR+1
9ØB9: A2 BF                LDX  #BOT_ROW      ;Set new pntr to bottom
                           >>>  SETPTR.NEWPTR
9ØBB: BD 56 93             LDA  YLOW,X        ;>> Get the Hi-Res LOB
9ØBE: 85 1C                STA  NEWPTR        ;>> Store in pointer
9ØCØ: 18                   CLC                ;>> Prepare for addition
9ØC1: BD 16 94             LDA  YHIGH,X       ;>> Get the Hi-Res HOB
9ØC4: 65 19                ADC  HRPAGE        ;>> Add for hi-res page
9ØC6: 85 1D                STA  NEWPTR+1      ;>> And store it too
                           <<<                ;>> End of macro

9ØC8: 2Ø 4E 91             JSR  MOVEROW       ;Go move the row
9ØCB: 4C D6 8F             JMP  KEYIN         ;Go look for another cmnd

                 ****************************************************
                 *  Scroll down:                                    *
                 ****************************************************

                 SCROLL_DOWN

                 * Store bottom row in buffer for later restore:

9ØCE: A9 69                LDA  #<BUFFER      ;Set the pointer to
9ØDØ: 85 1C                STA  NEWPTR        ; the buffer
9ØD2: A9 95                LDA  #>BUFFER
9ØD4: 85 1D                STA  NEWPTR+1

9ØD6: A2 BF                LDX  #BOT_ROW        ;Set old location to
                           >>>  SETPTR.OLDPTR   ; top row
9ØD8: BD 56 93             LDA  YLOW,X        ;>> Get the Hi-Res LOB
9ØDB: 85 1A                STA  OLDPTR        ;>> Store in pointer
9ØDD: 18                   CLC                ;>> Prepare for addition
9ØDE: BD 16 94             LDA  YHIGH,X       ;>> Get the Hi-Res HOB
9ØE1: 65 19                ADC  HRPAGE        ;>> Add for hi-res page
9ØE3: 85 1B                STA  OLDPTR+1      ;>> And store it too
                           <<<                ;>> End of macro

9ØE5: 2Ø 4E 91             JSR  MOVEROW       ;Go move the row

                 * Move the next rows down:

9ØE8: A2 BF                LDX  #BOT_ROW      ;Start at bottom row
9ØEA: 86 1E                STX  ROW           ;Save in counter

                           >>>  SETPTR.NEWPTR
9ØEC: BD 56 93             LDA  YLOW,X        ;>> Get the Hi-Res LOB
9ØEF: 85 1C                STA  NEWPTR        ;>> Store in pointer
9ØF1: 18                   CLC                ;>> Prepare for addition
9ØF2: BD 16 94             LDA  YHIGH,X       ;>> Get the Hi-Res HOB
9ØF5: 65 19                ADC  HRPAGE        ;>> Add for hi-res page
9ØF7: 85 1D                STA  NEWPTR+1      ;>> And store it too
                           <<<                ;>> End of macro
9ØF9: C6 1E                DEC  ROW           ;Go to next row
9ØFB: A6 1E                LDX  ROW           ;Put counter in register
                           >>>  SETPTR.OLDPTR
9ØFD: BD 56 93             LDA  YLOW,X        ;>> Get the Hi-Res LOB
91ØØ: 85 1A                STA  OLDPTR        ;>> Store in pointer
91Ø2: 18                   CLC                ;>> Prepare for addition
91Ø3: BD 16 94             LDA  YHIGH,X       ;>> Get the Hi-Res HOB
91Ø6: 65 19                ADC  HRPAGE        ;>> Add for hi-res page
91Ø8: 85 1B                STA  OLDPTR+1      ;>> And store it too
                           <<<                ;>> End of macro

91ØA: 2Ø 4E 91   DWN:LOOP  JSR  MOVEROW       ;Go move the row

91ØD: A6 1E                LDX  ROW           ;Restore the row counter
                           >>>  SETPTR.NEWPTR ;Set pntr for new row
91ØF: BD 56 93             LDA  YLOW,X        ;>> Get the Hi-Res LOB
9112: 85 1C                STA  NEWPTR        ;>> Store in pointer
9114: 18                   CLC                ;>> Prepare for addition
9115: BD 16 94             LDA  YHIGH,X       ;>> Get the Hi-Res HOB
9118: 65 19                ADC  HRPAGE        ;>> Add for hi-res page
911A: 85 1D                STA  NEWPTR+1      ;>> And store it too
                           <<<                ;>> End of macro
911C: C6 1E                DEC  ROW           ;Go to next row
911E: A6 1E                LDX  ROW           ;Put counter in register
                           >>>  SETPTR.OLDPTR
912Ø: BD 56 93             LDA  YLOW,X        ;>> Get the Hi-Res LOB
9123: 85 1A                STA  OLDPTR        ;>> Store in pointer
9125: 18                   CLC                ;>> Prepare for addition
9126: BD 16 94             LDA  YHIGH,X       ;>> Get the Hi-Res HOB
9129: 65 19                ADC  HRPAGE        ;>> Add for hi-res page
912B: 85 1B                STA  OLDPTR+1      ;>> And store it too
                           <<<                ;>> End of macro

912D: EØ FF                CPX  #TOP_ROW-1    ;Is it past top row?
912F: 9Ø D9                BCC  DWN:LOOP      ;No, so go do another row

                 * Restore old bottom row to new top row:

9131: A9 69                LDA  #<BUFFER      ;Set pntr for buffer
9133: 85 1A                STA  OLDPTR
9135: A9 95                LDA  #>BUFFER
9137: 85 1B                STA  OLDPTR+1
9139: A2 ØØ                LDX  #TOP_ROW      ;Set new pntr for top row
                           >>>  SETPTR.NEWPTR
913B: BD 56 93             LDA  YLOW,X        ;>> Get the Hi-Res LOB
913E: 85 1C                STA  NEWPTR        ;>> Store in pointer
914Ø: 18                   CLC                ;>> Prepare for addition
9141: BD 16 94             LDA  YHIGH,X       ;>> Get the Hi-Res HOB
9144: 65 19                ADC  HRPAGE        ;>> Add for hi-res page
9146: 85 1D                STA  NEWPTR+1      ;>> And store it too
                           <<<                ;>> End of macro

9148: 2Ø 4E 91             JSR  MOVEROW       ;Go move the row
914B: 4C D6 8F             JMP  KEYIN         ;Go check next key input

                 ****************************************************
                 *  SUBROUTINE MOVEROW                              *
                 ****************************************************

```

```
                  MOVEROW
914E: AØ 27                 LDY  #NUMCOL      ;Get no. columns in row
9150: B1 1A       MOVEIT    LDA  (OLDPTR),Y   ;Get the old byte
9152: 91 1C                 STA  (NEWPTR),Y   ;Store it in the new loc
9154: 88                    DEY               ;End of row?
9155: 1Ø F9                 BPL  MOVEIT       ;No, go move next byte
9157: 6Ø                    RTS               ;End of subroutine

                  ***********************************************
                  *  Scroll bytes RIGHT:                        *
                  ***********************************************

                  SCROLL_RIGHT

9158: A2 ØØ                 LDX  #TOP_ROW     ;Start with top row
                  RIT:MOVE >>> SETPTR.OLDPTR  ;Set byte pointer
915A: BD 56 93              LDA  YLOW,X       ;>> Get the Hi-Res LOB
915D: 85 1A                 STA  OLDPTR       ;>> Store in pointer
915F: 18                    CLC               ;>> Prepare for addition
916Ø: BD 16 94              LDA  YHIGH,X      ;>> Get the Hi-Res HOB
9163: 65 19                 ADC  HRPAGE       ;>> Add for hi-res page
9165: 85 1B                 STA  OLDPTR+1     ;>> And store it too
                            <<<               ;>> End of macro

9167: AØ 27                 LDY  #NUMCOL      ;Point to right column
9169: B1 1A                 LDA  (OLDPTR),Y   ;Get that byte
916B: 8D 69 95              STA  BUFFER       ;Store for wrap-around
916E: 88                    DEY               ;Start loop at penult col

916F: B1 1A       RIT:LOOP  LDA  (OLDPTR),Y   ;Get the old byte value
9171: C8                    INY               ;Point to new byte
9172: 91 1A                 STA  (OLDPTR),Y   ;And store at new loc
9174: 88                    DEY               ;Move back to moved byte
9175: 88                    DEY               ;End of row?
9176: 1Ø F7                 BPL  RIT:LOOP     ;No, go get next byte

9178: AØ ØØ                 LDY  #Ø           ;Point to left column
917A: AD 69 95              LDA  BUFFER       ;Restore right byte
917D: 91 1A                 STA  (OLDPTR),Y   ; to left column

917F: E8                    INX               ;Go to next row down
918Ø: EØ CØ                 CPX  #BOT_ROW+1   ;Past bottom row?
9182: 9Ø D6                 BCC  RIT:MOVE     ;No, go do next row
9184: 4C D6 8F              JMP  KEYIN        ;Yes, go check key input

                  ***********************************************
                  *  Move dots right:                           *
                  ***********************************************

                  MOVE_RIGHT

9187: A2 ØØ                 LDX  #TOP_ROW     ;Start at top row
                  MBR:LOOP >>> SETPTR.NEWPTR  ;Set the pointer
9189: BD 56 93              LDA  YLOW,X       ;>> Get the Hi-Res LOB
918C: 85 1C                 STA  NEWPTR       ;>> Store in pointer
918E: 18                    CLC               ;>> Prepare for addition
918F: BD 16 94              LDA  YHIGH,X      ;>> Get the Hi-Res HOB
9192: 65 19                 ADC  HRPAGE       ;>> Add for hi-res page
9194: 85 1D                 STA  NEWPTR+1     ;>> And store it too
                            <<<               ;>> End of macro

                  * Zero the buffer bytes:

9196: AØ 28                 LDY  #NUMCOL+1    ;Get the no. columns
9198: A9 ØØ                 LDA  #Ø           ;Set things to zero
919A: 85 Ø2                 STA  NEWBYTE      ;Clear the new byte too
919C: 99 69 95    RCLOOP    STA  BUFFER,Y     ;Clear the buffer byte
919F: 88                    DEY               ;Done?
91AØ: 1Ø FA                 BPL  RCLOOP       ;No, go loop

                  * Shift the bits:

91A2: AØ ØØ                 LDY  #Ø           ;Set to first column
91A4: B1 1C       MBR:SHFT  LDA  (NEWPTR),Y   ;Get the current byte
91A6: 48                    PHA               ;Save the byte
91A7: 29 8Ø                 AND  #%1ØØØØØØØ   ;Mask out pixel bits
91A9: 85 Ø1                 STA  COLORBIT     ;Save the result
91AB: 68                    PLA               ;Get back the byte
91AC: ØA                    ASL               ;Shift pixels right!
91AD: ØA                    ASL               ;Now bit 6 is in carry
91AE: 26 Ø2                 ROL  NEWBYTE      ;Roll it into new bit 1
91BØ: 4A                    LSR               ;Shift back one
91B1: FØ ØA                 BEQ  MBR:2        ;If no dot, buffer unchng
91B3: 19 69 95              ORA  BUFFER,Y     ;Get what's there
91B6: 29 7F                 AND  #%Ø1111111   ;Clear buffer color bit
91B8: Ø5 Ø1                 ORA  COLORBIT     ;Add current color bit
91BA: 99 69 95              STA  BUFFER,Y     ;Save the results

91BD: C8          MBR:2     INY               ;Go to next buffer byte
91BE: A5 Ø2                 LDA  NEWBYTE      ;Get the new byte
91CØ: FØ Ø5                 BEQ  MBR:3        ;Don't save if zero
91C2: Ø5 Ø1                 ORA  COLORBIT     ;And put in new color bit
91C4: 99 69 95              STA  BUFFER,Y     ;Store the results

91C7: A9 ØØ       MBR:3     LDA  #Ø           ;Zero the new byte
91C9: 85 Ø2                 STA  NEWBYTE

91CB: CØ 28                 CPY  #NUMCOL+1    ;Past last column?
91CD: 9Ø D5                 BCC  MBR:SHFT     ;No, go to next byte

                  * Move the buffer row back to the hi-res screen:

91CF: AØ 27                 LDY  #NUMCOL      ;Get the no. of columns
91D1: B9 69 95    MBLOOP    LDA  BUFFER,Y     ;Get the shifted byte
91D4: 91 1C                 STA  (NEWPTR),Y   ;Store it on screen
91D6: 88                    DEY               ;End of row?
91D7: 1Ø F8                 BPL  MBLOOP       ;No, go move next byte

                  * Restore last byte to the first:

91D9: AØ 28                 LDY  #NUMCOL+1    ;Set to end of buffer
91DB: B9 69 95              LDA  BUFFER,Y     ;Get the buffer value
91DE: AØ ØØ                 LDY  #Ø           ;Point to first column
91EØ: 11 1C                 ORA  (NEWPTR),Y   ;Get what's there
91E2: 91 1C                 STA  (NEWPTR),Y   ;Save the results

91E4: E8                    INX               ;Go to next row down
91E5: EØ CØ                 CPX  #BOT_ROW+1   ;Past bottom row?
91E7: 9Ø AØ                 BCC  MBR:LOOP     ;No, go do next row
91E9: 4C D6 8F              JMP  KEYIN        ;Yes, go check key input

                  ***********************************************
                  *  Scroll bytes LEFT:                         *
                  ***********************************************

                  SCROLL_LEFT

91EC: A2 ØØ                 LDX  #TOP_ROW     ;Start with top row
                  LFT:MOVE >>> SETPTR.OLDPTR  ;Set byte pointer
91EE: BD 56 93              LDA  YLOW,X       ;>> Get the Hi-Res LOB
91F1: 85 1A                 STA  OLDPTR       ;>> Store in pointer
91F3: 18                    CLC               ;>> Prepare for addition
91F4: BD 16 94              LDA  YHIGH,X      ;>> Get the Hi-Res HOB
91F7: 65 19                 ADC  HRPAGE       ;>> Add for hi-res page
91F9: 85 1B                 STA  OLDPTR+1     ;>> And store it too
                            <<<               ;>> End of macro

91FB: AØ ØØ                 LDY  #Ø           ;Point to left column
91FD: B1 1A                 LDA  (OLDPTR),Y   ;Get that byte
91FF: 8D 69 95              STA  BUFFER       ;Store for wrap-around
92Ø2: C8                    INY               ;Start loop second col

92Ø3: B1 1A       LFT:LOOP  LDA  (OLDPTR),Y   ;Get the old byte value
92Ø5: 88                    DEY               ;Point to new byte
92Ø6: 91 1A                 STA  (OLDPTR),Y   ;And store at new loc
92Ø8: C8                    INY               ;Move back to moved byte
92Ø9: C8                    INY               ;Point to new byte
92ØA: CØ 28                 CPY  #NUMCOL+1    ;Past column end?
92ØC: 9Ø F5                 BCC  LFT:LOOP     ;No, go get next byte

92ØE: AØ 27                 LDY  #NUMCOL      ;Point to right column
921Ø: AD 69 95              LDA  BUFFER       ;Restore left byte
9213: 91 1A                 STA  (OLDPTR),Y   ; to right column

9215: E8                    INX               ;Go to next row down
9216: EØ CØ                 CPX  #BOT_ROW+1   ;Past bottom row?
9218: 9Ø D4                 BCC  LFT:MOVE     ;No, go do next row
921A: 4C D6 8F              JMP  KEYIN        ;Yes, go check key input

                  ***********************************************
                  *  Move dots left:                            *
                  ***********************************************

                  MOVE_LEFT

921D: A2 ØØ                 LDX  #TOP_ROW     ;Start at top row
                  MBL:LOOP >>> SETPTR.NEWPTR  ;Set the pointer
921F: BD 56 93              LDA  YLOW,X       ;>> Get the Hi-Res LOB
9222: 85 1C                 STA  NEWPTR       ;>> Store in pointer
9224: 18                    CLC               ;>> Prepare for addition
9225: BD 16 94              LDA  YHIGH,X      ;>> Get the Hi-Res HOB
9228: 65 19                 ADC  HRPAGE       ;>> Add for hi-res page
922A: 85 1D                 STA  NEWPTR+1     ;>> And store it too
                            <<<               ;>> End of macro

                  * Zero the buffer bytes:

922C: AØ 28                 LDY  #NUMCOL+1    ;Get the no. columns
922E: A9 ØØ                 LDA  #Ø           ;Set things to zero
923Ø: 85 Ø2                 STA  NEWBYTE      ;Clear the new byte too
9232: 99 69 95    LCLOOP    STA  BUFFER,Y     ;Clear the buffer byte
9235: 88                    DEY               ;Done?
9236: 1Ø FA                 BPL  LCLOOP       ;No, go loop

                  * Shift the bits:

```

```
9238: AØ 27                LDY  #NUMCOL     ;Set to last column
923A: B1 1C      MBL:SHFT  LDA  (NEWPTR),Y  ;Get the current byte
923C: 48                   PHA              ;Save the byte
923D: 29 8Ø                AND  #%1ØØØØØØØ  ;Mask out pixel bits
923F: 85 Ø1                STA  COLORBIT    ;Save the result
9241: 68                   PLA              ;Get back the byte
9242: 29 7F                AND  #%Ø1111111  ;Delete color bit
9244: 4A                   LSR              ;Shift pixels left!
9245: Ø8                   PHP              ;Save P register
9246: 66 Ø2                ROR  NEWBYTE     ;Roll it into new bit 7
9248: 46 Ø2                LSR  NEWBYTE     ;Move it into bit 6
924A: 28                   PLP              ;Restore P register
924B: FØ ØA                BEQ  MBL:2       ;If no dot, buffer unchng
924D: 19 69 95             ORA  BUFFER,Y    ;Get what's there
925Ø: 29 7F                AND  #%Ø1111111  ;Clear buffer color bit
9252: Ø5 Ø1                ORA  COLORBIT    ;Add current color bit
9254: 99 69 95             STA  BUFFER,Y    ;Save the results

9257: 88         MBL:2     DEY              ;Go to next buffer byte
9258: 1Ø Ø2                BPL  MBL:4       ;Past end?
925A: AØ 28                LDY  #NUMCOL+1   ;Then set new buff index
925C: A5 Ø2      MBL:4     LDA  NEWBYTE     ;Get the new byte
925E: FØ Ø5                BEQ  MBL:3       ;Don't save if zero
926Ø: Ø5 Ø1                ORA  COLORBIT    ;And put in new color bit
9262: 99 69 95             STA  BUFFER,Y    ;Store the results

9265: A9 ØØ      MBL:3     LDA  #Ø          ;Zero the new byte
9267: 85 Ø2                STA  NEWBYTE

9269: CØ 28                CPY  #NUMCOL+1   ;Past last column?
926B: DØ CD                BNE  MBL:SHFT    ;No, go to next byte

                 * Move the buffer row back to the hi-res screen:

926D: AØ 27                LDY  #NUMCOL     ;Get the no. of columns
926F: B9 69 95   LBLOOP    LDA  BUFFER,Y    ;Get the shifted byte
9272: 91 1C                STA  (NEWPTR),Y  ;Store it on screen
9274: 88                   DEY              ;End of row?
9275: 1Ø F8                BPL  LBLOOP      ;No, go move next byte

                 *  Restore last byte to the first:

9277: AØ 28                LDY  #NUMCOL+1   ;Set to end of buffer
9279: B9 69 95             LDA  BUFFER,Y    ;Get the buffer value
927C: AØ 27                LDY  #NUMCOL     ;Point to last column
927E: 11 1C                ORA  (NEWPTR),Y  ;Get what's there
928Ø: 91 1C                STA  (NEWPTR),Y  ;Save the results

9282: E8                   INX              ;Go to next row down
9283: EØ CØ                CPX  #BOT_ROW+1  ;Past bottom row?
9285: 9Ø 98                BCC  MBL:LOOP    ;No, go do next row

9287: 4C D6 8F             JMP  KEYIN       ;Yes, go check key input

                 **************************************************
                 *  Inverse the colors:                           *
                 **************************************************

                 SET_INVERSE

928A: A2 ØØ                LDX  #TOP_ROW    ;Start at top row
                 INVLOOP   >>>  SETPTR.NEWPTR
928C: BD 56 93             LDA  YLOW,X      ;>> Get the Hi-Res LOB
928F: 85 1C                STA  NEWPTR      ;>> Store in pointer
9291: 18                   CLC              ;>> Prepare for addition
9292: BD 16 94             LDA  YHIGH,X     ;>> Get the Hi-Res HOB
9295: 65 19                ADC  HRPAGE      ;>> Add for hi-res page
9297: 85 1D                STA  NEWPTR+1    ;>> And store it too
                           <<<              ;>> End of macro
9299: AØ 27                LDY  #NUMCOL
929B: B1 1C      ROWLOOP   LDA  (NEWPTR),Y  ;Get hi-res screen byte
929D: 49 FF                EOR  #%11111111  ;XOR it with all ones
929F: 91 1C                STA  (NEWPTR),Y  ;And store it back
92A1: 88                   DEY              ;Past first column?
92A2: 1Ø F7                BPL  ROWLOOP     ;No, go to next byte

92A4: E8                   INX              ;Go to next lower row
92A5: EØ CØ                CPX  #BOT_ROW+1  ;Gone below last row?
92A7: 9Ø E3                BCC  INVLOOP     ;No, go to next row
92A9: 4C D6 8F             JMP  KEYIN       ;Go check for next key

                 **************************************************
                 *  Change the color bit:                         *
                 **************************************************

                 CHG_COLOR_BIT

92AC: A2 ØØ                LDX  #TOP_ROW    ;Start at top row
                 CBLOOP    >>>  SETPTR.NEWPTR
92AE: BD 56 93             LDA  YLOW,X      ;>> Get the Hi-Res LOB
92B1: 85 1C                STA  NEWPTR      ;>> Store in pointer
92B3: 18                   CLC              ;>> Prepare for addition
92B4: BD 16 94             LDA  YHIGH,X     ;>> Get the Hi-Res HOB
92B7: 65 19                ADC  HRPAGE      ;>> Add for hi-res page
92B9: 85 1D                STA  NEWPTR+1    ;>> And store it too
                           <<<              ;>> End of macro
92BB: AØ 27                LDY  #NUMCOL
92BD: B1 1C      RLOOP     LDA  (NEWPTR),Y  ;Get hi-res screen byte
92BF: 49 8Ø                EOR  #%1ØØØØØØØ  ;XOR the color bit
92C1: 91 1C                STA  (NEWPTR),Y  ;And store it back
92C3: 88                   DEY              ;Past first column?
92C4: 1Ø F7                BPL  RLOOP       ;No, go to next byte

92C6: E8                   INX              ;Go to next lower row
92C7: EØ CØ                CPX  #BOT_ROW+1  ;Gone below last row?
92C9: 9Ø E3                BCC  CBLOOP      ;No, go to next row
92CB: 4C D6 8F             JMP  KEYIN       ;Go check for next key

```

```
                 **************************************************
                 *  Change the Hi-Res page:                       *
                 **************************************************

                 CHANGE_PAGE

92CE: A5 19                LDA  HRPAGE      ;Is it page 1?
92DØ: DØ ØD                BNE  MAKE_P1     ;No, so make it page 1
92D2: 2C 52 CØ             BIT  FULLSCRN    ;Must be FULL graphics
92D5: 2C 55 CØ             BIT  FLIP2       ;Flip to page 2
92D8: A9 2Ø                LDA  #$2Ø        ;Store the HOB page byte
92DA: 85 19                STA  HRPAGE
92DC: 4C D6 8F             JMP  KEYIN

92DF: A5 ØØ      MAKE_P1   LDA  FMFLAG      ;Full or mixed?
92E1: DØ Ø3                BNE  HRP:1       ;It was mixed, do nothing
92E3: 2C 53 CØ             BIT  MXEDSCRN    ;Make it mixed screen
92E6: A9 ØØ      HRP:1     LDA  #Ø          ;Store the HOB page byte
92E8: 85 19                STA  HRPAGE
92EA: 2C 54 CØ             BIT  FLIP1       ;Flip to page 1
92ED: 4C D6 8F             JMP  KEYIN

                 **************************************************
                 *  Toggle between FULL/MIXED hi-res screen:      *
                 **************************************************

                 TOGGL_FULLMXD

92FØ: A5 19                LDA  HRPAGE      ;Is it page 2?
92F2: FØ Ø6                BEQ  TOGGLE      ;No, toggle FULL/MIXED
92F4: 2Ø 3A FF             JSR  BELL        ;Can't toggle from page 2
92F7: 4C D6 8F             JMP  KEYIN

92FA: A5 ØØ      TOGGLE    LDA  FMFLAG      ;Get the flag
92FC: 49 Ø1                EOR  #1          ;Switch its value
92FE: 85 ØØ                STA  FMFLAG      ;Save new value
93ØØ: DØ Ø6                BNE  MAKEFULL    ;If 1, make FULL

93Ø2: 2C 53 CØ             BIT  MXEDSCRN    ;Make it MIXED screen
93Ø5: 4C D6 8F             JMP  KEYIN

93Ø8: 2C 52 CØ   MAKEFULL  BIT  FULLSCRN    ;Make it FULL graphics
93ØB: 4C D6 8F             JMP  KEYIN

                 **************************************************
                 *  Merge the Hi-Res screens:                     *
                 **************************************************

                 PICTURE_MERGE

93ØE: A2 ØØ                LDX  #TOP_ROW    ;Start at top row
                 MLOOP     >>>  SETPTR.NEWPTR
931Ø: BD 56 93             LDA  YLOW,X      ;>> Get the Hi-Res LOB
9313: 85 1C                STA  NEWPTR      ;>> Store in pointer
9315: 18                   CLC              ;>> Prepare for addition
9316: BD 16 94             LDA  YHIGH,X     ;>> Get the Hi-Res HOB
9319: 65 19                ADC  HRPAGE      ;>> Add for hi-res page
931B: 85 1D                STA  NEWPTR+1    ;>> And store it too
                           <<<              ;>> End of macro
931D: A5 19                LDA  HRPAGE      ;Change the page
931F: 49 2Ø                EOR  #$2Ø        ;Make $2Ø->Ø, Ø->$2Ø
9321: 85 19                STA  HRPAGE      ;Store other page number
                           >>>  SETPTR.OLDPTR ;Set pntr to that page
9323: BD 56 93             LDA  YLOW,X      ;>> Get the Hi-Res LOB
9326: 85 1A                STA  OLDPTR      ;>> Store in pointer
9328: 18                   CLC              ;>> Prepare for addition
9329: BD 16 94             LDA  YHIGH,X     ;>> Get the Hi-Res HOB
932C: 65 19                ADC  HRPAGE      ;>> Add for hi-res page
932E: 85 1B                STA  OLDPTR+1    ;>> And store it too
                           <<<              ;>> End of macro
933Ø: AØ 27                LDY  #NUMCOL     ;Get number of columns
9332: B1 1A      RMLOOP    LDA  (OLDPTR),Y  ;Get hi-res byte old page
9334: 51 1C                EOR  (NEWPTR),Y  ;XOR it with what's there
9336: 91 1C                STA  (NEWPTR),Y  ;And store it on new page
9338: 88                   DEY              ;Past first column?
9339: 1Ø F7                BPL  RMLOOP      ;No, go to next byte

933B: A5 19                LDA  HRPAGE      ;Change page back
933D: 49 2Ø                EOR  #$2Ø        ;Ø->$2Ø, $2Ø->Ø
933F: 85 19                STA  HRPAGE      ;Restore HR screen page
9341: E8                   INX              ;Go to next lower row
9342: EØ CØ                CPX  #BOT_ROW+1  ;Gone below last row?
9344: 9Ø CA                BCC  MLOOP       ;No, go to next row
9346: 4C D6 8F             JMP  KEYIN       ;Go check for next key

                 **************************************************
                 *  Clear the screen:                             *
                 **************************************************

                 CLEAR_SCREEN

9349: A5 19                LDA  HRPAGE      ;Which page is it?
934B: 18                   CLC              ;Prepare to add
934C: 69 2Ø                ADC  #$2Ø        ;Add for mon HPAGE
934E: 85 E6                STA  HPAGE       ;Store in aplsoft loc
935Ø: 2Ø F2 F3             JSR  HCLR        ;Clear that screen
9353: 4C D6 8F             JMP  KEYIN       ;Go get next key

                 **************************************************
                 *  Hi-Res Screen addresses:                      *
                 **************************************************

```

```
9356: ØØ ØØ ØØ  653  YLOW     HEX  ØØØØØØØØØØØØØØØØ
9359: ØØ ØØ ØØ ØØ ØØ
935E: 8Ø 8Ø 8Ø  654           HEX  8Ø8Ø8Ø8Ø8Ø8Ø8Ø8Ø
9361: 8Ø 8Ø 8Ø 8Ø 8Ø
9366: ØØ ØØ ØØ  655           HEX  ØØØØØØØØØØØØØØØØ
9369: ØØ ØØ ØØ ØØ ØØ
936E: 8Ø 8Ø 8Ø  656           HEX  8Ø8Ø8Ø8Ø8Ø8Ø8Ø8Ø
9371: 8Ø 8Ø 8Ø 8Ø 8Ø
9376: ØØ ØØ ØØ  657           HEX  ØØØØØØØØØØØØØØØØ
9379: ØØ ØØ ØØ ØØ ØØ
937E: 8Ø 8Ø 8Ø  658           HEX  8Ø8Ø8Ø8Ø8Ø8Ø8Ø8Ø
9381: 8Ø 8Ø 8Ø 8Ø 8Ø
9386: ØØ ØØ ØØ  659           HEX  ØØØØØØØØØØØØØØØØ
9389: ØØ ØØ ØØ ØØ ØØ
938E: 8Ø 8Ø 8Ø  66Ø           HEX  8Ø8Ø8Ø8Ø8Ø8Ø8Ø8Ø
9391: 8Ø 8Ø 8Ø 8Ø 8Ø
9396: 28 28 28  661           HEX  2828282828282828
9399: 28 28 28 28 28
939E: A8 A8 A8  662           HEX  A8A8A8A8A8A8A8A8
93A1: A8 A8 A8 A8 A8
93A6: 28 28 28  663           HEX  2828282828282828
93A9: 28 28 28 28 28
93AE: A8 A8 A8  664           HEX  A8A8A8A8A8A8A8A8
93B1: A8 A8 A8 A8 A8
93B6: 28 28 28  665           HEX  2828282828282828
93B9: 28 28 28 28 28
93BE: A8 A8 A8  666           HEX  A8A8A8A8A8A8A8A8
93C1: A8 A8 A8 A8 A8
93C6: 28 28 28  667           HEX  2828282828282828
93C9: 28 28 28 28 28
93CE: A8 A8 A8  668           HEX  A8A8A8A8A8A8A8A8
93D1: A8 A8 A8 A8 A8
93D6: 5Ø 5Ø 5Ø  669           HEX  5Ø5Ø5Ø5Ø5Ø5Ø5Ø5Ø
93D9: 5Ø 5Ø 5Ø 5Ø 5Ø
93DE: DØ DØ DØ  67Ø           HEX  DØDØDØDØDØDØDØDØ
93E1: DØ DØ DØ DØ DØ
93E6: 5Ø 5Ø 5Ø  671           HEX  5Ø5Ø5Ø5Ø5Ø5Ø5Ø5Ø
93E9: 5Ø 5Ø 5Ø 5Ø 5Ø
93EE: DØ DØ DØ  672           HEX  DØDØDØDØDØDØDØDØ
93F1: DØ DØ DØ DØ DØ
93F6: 5Ø 5Ø 5Ø  673           HEX  5Ø5Ø5Ø5Ø5Ø5Ø5Ø5Ø
93F9: 5Ø 5Ø 5Ø 5Ø 5Ø
93FE: DØ DØ DØ  674           HEX  DØDØDØDØDØDØDØDØ
94Ø1: DØ DØ DØ DØ DØ
94Ø6: 5Ø 5Ø 5Ø  675           HEX  5Ø5Ø5Ø5Ø5Ø5Ø5Ø5Ø
94Ø9: 5Ø 5Ø 5Ø 5Ø 5Ø
94ØE: DØ DØ DØ  676           HEX  DØDØDØDØDØDØDØDØ
9411: DØ DØ DØ DØ DØ
                677  *
9416: 2Ø 24 28  678  YHIGH    HEX  2Ø24282C3Ø34383C
9419: 2C 3Ø 34 38 3C
941E: 2Ø 24 28  679           HEX  2Ø24282C3Ø34383C
9421: 2C 3Ø 34 38 3C
9426: 21 25 29  68Ø           HEX  2125292D3135393D
9429: 2D 31 35 39 3D
942E: 21 25 29  681           HEX  2125292D3135393D
9431: 2D 31 35 39 3D
9436: 22 26 2A  682           HEX  22262A2E32363A3E
9439: 2E 32 36 3A 3E
943E: 22 26 2A  683           HEX  22262A2E32363A3E
9441: 2E 32 36 3A 3E
9446: 23 27 2B  684           HEX  23272B2F33373B3F
9449: 2F 33 37 3B 3F
944E: 23 27 2B  685           HEX  23272B2F33373B3F
9451: 2F 33 37 3B 3F
9456: 2Ø 24 28  686           HEX  2Ø24282C3Ø34383C
9459: 2C 3Ø 34 38 3C
945E: 2Ø 24 28  687           HEX  2Ø24282C3Ø34383C
9461: 2C 3Ø 34 38 3C
9466: 21 25 29  688           HEX  2125292D3135393D
9469: 2D 31 35 39 3D
946E: 21 25 29  689           HEX  2125292D3135393D
9471: 2D 31 35 39 3D
9476: 22 26 2A  69Ø           HEX  22262A2E32363A3E
9479: 2E 32 36 3A 3E
947E: 22 26 2A  691           HEX  22262A2E32363A3E
9481: 2E 32 36 3A 3E
9486: 23 27 2B  692           HEX  23272B2F33373B3F
9489: 2F 33 37 3B 3F
948E: 23 27 2B  693           HEX  23272B2F33373B3F
9491: 2F 33 37 3B 3F
9496: 2Ø 24 28  694           HEX  2Ø24282C3Ø34383C
9499: 2C 3Ø 34 38 3C
949E: 2Ø 24 28  695           HEX  2Ø24282C3Ø34383C
94A1: 2C 3Ø 34 38 3C
94A6: 21 25 29  696           HEX  2125292D3135393D
94A9: 2D 31 35 39 3D
94AE: 21 25 29  697           HEX  2125292D3135393D
94B1: 2D 31 35 39 3D
94B6: 22 26 2A  698           HEX  22262A2E32363A3E
94B9: 2E 32 36 3A 3E
94BE: 22 26 2A  699           HEX  22262A2E32363A3E
94C1: 2E 32 36 3A 3E
94C6: 23 27 2B  7ØØ           HEX  23272B2F33373B3F
94C9: 2F 33 37 3B 3F
94CE: 23 27 2B  7Ø1           HEX  23272B2F33373B3F
94D1: 2F 33 37 3B 3F
```

```
                7Ø2
                7Ø3  ***************************************************
                7Ø4  *  Messages (commands at bottom of screen)        *
                7Ø5  ***************************************************
                7Ø6
94D6: D3 C3 D2  7Ø7  SCLCMNDS ASC  "SCROLL COMMANDS: "
94D9: CF CC CC AØ C3 CF CD CD
94E1: C1 CE C4 D3 BA AØ
94E7: 3C 2D     7Ø8           INV  "<-"
94E9: AC AØ     7Ø9           ASC  ", "
94EB: 2D 3E     71Ø           INV  "->"
94ED: AC AØ     711           ASC  ", "
94EF: 3C        712           INV  "<"
94FØ: AC AØ     713           ASC  ", "
94F2: 3E        714           INV  ">"
94F3: AC AØ     715           ASC  ", "
94F5: Ø1        716           INV  "A"
94F6: AC AØ     717           ASC  ", "
94F8: 1A        718           INV  "Z"
94F9: ØØ        719           BRK
94FA: Ø9        72Ø  INV:I    INV  "I"
94FB: BA C9 CE  721           ASC  ":INVERSE COLORS"
94FE: D6 C5 D2 D3 C5 AØ C3 CF
95Ø6: CC CF D2 D3
95ØA: ØØ        722           BRK
95ØB: Ø3        723  CHGBIT:C INV  "C"
95ØC: BA C3 C8  724           ASC  ":CHANGE COLOR BIT"
95ØF: C1 CE C7 C5 AØ C3 CF CC
9517: CF D2 AØ C2 C9 D4
951D: ØØ        725           BRK
951E: Ø6        726  FLMX:F   INV  "F"
951F: BA C6 D5  727           ASC  ":FULL/MIXED TOGGLE"
9522: CC CC AF CD C9 D8 C5 C4
952A: AØ D4 CF C7 C7 CC C5
9531: ØØ        728           BRK
9532: ØD        729  MERGE:M  INV  "M"
9533: BA CD C5  73Ø           ASC  ":MERGE SCREENS"
9536: D2 C7 C5 AØ D3 C3 D2 C5
953E: C5 CE D3
9541: ØØ        731           BRK
9542: Ø3 14 12  732  CLEAR:@  INV  "CTRL- "
9545: ØC 2D 2Ø
9548: BA C3 CC  733           ASC  ":CLEAR SCREEN"
954B: C5 C1 D2 AØ D3 C3 D2 C5
9553: C5 CE
9555: ØØ        734           BRK
9556: 1Ø        735  PAGE:P   INV  "P"
9557: BA C6 CC  736           ASC  ":FLIP PAGE"
955A: C9 DØ AØ DØ C1 C7 C5
9561: ØØ        737           BRK
9562: 11        738  QUIT:Q   INV  "Q"
9563: BA D1 D5  739           ASC  ":QUIT"
9566: C9 D4
9568: ØØ        74Ø           BRK
                741
                742  BUFFER   DS   1          ;Buffer to store line
                743
                744  *  NOTE:  Allow 41 bytes for this data buffer.
                745  *         Thus for the normal 48K Apple with DOS,
                746  *         the buffer must be before $95D8 (3836Ø)


--End assembly--

1642 bytes

Errors: Ø
```

```
         KEY PERFECT 4.Ø
             RUN ON
         HI.RES.HOUDINI
  ==========================
    CODE       ADDR# - ADDR#
   ------     -------------
    2CA9       8FØØ - 8F4F
    2CD4       8F5Ø - 8F9F
    2DØC       8FAØ - 8FEF
    2887       8FFØ - 9Ø3F
    2668       9Ø4Ø - 9Ø8F
    28FE       9Ø9Ø - 9ØDF
    286E       9ØEØ - 912F
    28D2       913Ø - 917F
    2689       918Ø - 91CF
    2AB5       91DØ - 921F
    2535       922Ø - 926F
    24E4       927Ø - 92BF
    2948       92CØ - 93ØF
    25EE       931Ø - 935F
    2956       936Ø - 93AF
    2ECE       93BØ - 93FF
    27C9       94ØØ - 944F
    26E1       945Ø - 949F
    2736       94AØ - 94EF
    2827       94FØ - 953F
    1884       954Ø - 9568
  PROGRAM CHECK IS : Ø669
```

```
CHECK CODE 3.Ø

ON: HI.RES.HOUDINI
TYPE: B

LENGTH: Ø669
CHECKSUM: 9Ø
```

Hi-Res Houdini

LISTING 2: HOUDINI.DRIVER

```
10  REM ***********************
20  REM *    HOUDINI.DRIVER    *
30  REM *  BY SCOTT ZIMMERMAN  *
40  REM *  COPYRIGHT (C) 1984  *
50  REM *  BY MICROSPARC, INC  *
60  REM *  CONCORD, MA, Ø1742  *
70  REM ***********************
8Ø D$ =  CHR$ (4)
9Ø  ONERR  GOTO 36Ø
1ØØ  HOME : VTAB 2: HTAB 1Ø: INVERSE : PRINT
     "HI-RES HOUDINI DRIVER": NORMAL
11Ø  VTAB 12: CALL  - 958: VTAB 22: PRINT "'?
     ' FOR DISK CATALOG": PRINT "<RETURN> TO
     SKIP": VTAB 12: PRINT "FILE NAME FOR PAG
     E 1 PICTURE:": INPUT "";P1$
12Ø  IF P1$ = "?" THEN  HOME : PRINT D$"CATAL
     OG": PRINT "PRESS ANY KEY TO CONTINUE": GET
     K$: PRINT : HOME : GOTO 11Ø
13Ø  IF P1$ = "" THEN 15Ø
14Ø  PRINT D$;"BLOAD";P1$;",A$2ØØØ"
15Ø  VTAB 12: CALL  - 958: VTAB 22: PRINT "'?
     ' FOR DISK CATALOG": PRINT "<RETURN> TO
     SKIP": VTAB 12: PRINT "FILE NAME FOR PAG
     E 2 PICTURE:": INPUT "";P2$
16Ø  IF P2$ = "?" THEN  HOME : PRINT D$"CATAL
     OG": PRINT "PRESS ANY KEY TO CONTINUE": GET
     K$: PRINT : HOME : GOTO 15Ø
17Ø  IF P2$ = "" THEN 19Ø
18Ø  PRINT D$;"BLOAD";P2$;",A$4ØØØ"
19Ø  PRINT D$;"BLOAD HI.RES.HOUDINI"
2ØØ  CALL 366Ø8: TEXT : HOME
21Ø  HOME : VTAB 12: PRINT "DO YOU REALLY WAN
     T TO QUIT? (Y/N)";: GET K$: PRINT : IF K
     $ = "N" THEN  CALL 366Ø8: GOTO 21Ø
22Ø  IF K$ <  > "Y" THEN 21Ø
23Ø  HOME : VTAB 12: PRINT "LOAD NEW PICTURES
      AND RE-START? (Y/N)";: GET K$: PRINT : IF
     K$ = "Y" THEN 1ØØ
24Ø  IF K$ <  > "N" THEN 23Ø
25Ø  ONERR  GOTO 38Ø
26Ø  HOME : VTAB 12: PRINT "SAVE PICTURE ON P
     AGE 1? (Y/N)";: GET K$: PRINT : IF K$ =
     "N" THEN 3ØØ
27Ø  IF K$ <  > "Y" THEN 26Ø
28Ø  VTAB 12: CALL  - 958: PRINT "ENTER FILE
     NAME:": INPUT "";F$: IF  LEN (F$) > 15 OR
      VAL (F$) > Ø THEN  PRINT "ILLEGAL FILE
     NAME. TRY AGAIN.": FOR I = 1 TO 1ØØØ: NEXT
     : GOTO 28Ø
29Ø  PRINT D$"BSAVE";F$;",A$2ØØØ,L$2ØØØ"
3ØØ  HOME : VTAB 12: PRINT "SAVE PICTURE ON P
     AGE 2? (Y/N)";: GET K$: PRINT : IF K$ =
     "N" THEN 34Ø
31Ø  IF K$ <  > "Y" THEN 3ØØ
32Ø  VTAB 12: CALL  - 958: PRINT "ENTER FILE
     NAME:": INPUT "";F$: IF  LEN (F$) > 15 OR
      VAL (F$) > Ø THEN  PRINT "ILLEGAL FILE
     NAME. TRY AGAIN.": FOR I = 1 TO 1ØØØ: NEXT
     : GOTO 32Ø
33Ø  PRINT D$"BSAVE";F$;",A$4ØØØ,L$2ØØØ"
34Ø  HOME : END
35Ø  REM  ERROR TRAP #1
36Ø  VTAB 22: PRINT "ERROR NUMBER "; PEEK (22
     2): PRINT "PRESS ANY KEY TO START AGAIN"
     ;: GET K$: GOTO 1ØØ
37Ø  REM  ERROR TRAP #2
38Ø  VTAB 22: PRINT "ERROR NUMBER "; PEEK (22
     2): PRINT "PRESS ANY KEY TO TRY AGAIN";:
      GET K$: GOTO 26Ø
```

```
        KEY PERFECT 4.Ø
            RUN ON
        HOUDINI.DRIVER
 ==============================
   CODE      LINE# - LINE#
   ------    -------------
    6645       1Ø  -  1ØØ
    D58B      11Ø  -  2ØØ
    CE94      21Ø  -  3ØØ
    9698      31Ø  -  38Ø
 PROGRAM CHECK IS : Ø5A2
```

```
   CHECK CODE 3.Ø

ON: HOUDINI.DRIVER
TYPE: A

LENGTH: Ø528
CHECKSUM: BØ
```