DOUBLE HI·RES GRAPHICS IV

Using routines developed in the first three parts of the series, this month's column presents a graphics utility that lets you create Double Hi-Res pictures and block shapes and save them to disk for later use or editing.

by Robert R. Devine
1415 West 19th St.
El Dorado, AR 71730

My original plans for this issue were to begin work on some horizontal shifting routines for use with Double Hi-Res; however, I think we'll put that discussion off until next time.

Instead we'll look at a program that allows you to create Double Hi-Res block shapes or complete pictures, in black and white or full color, directly on the Double Hi-Res screen.

I started out with the BLOCK SHAPE MAKER program from *Nibble* Vol. 4/No. 5. When I began the task, I figured "No sweat — the program is already written — I'll just change it over to Double Hi-Res." However, as it turned out, it wasn't quite that simple. All of the idiosyncrasies of Double Hi-Res, and all of the soft-switch flipping to make things work properly required so many modifications that little more than faint memories of the original program remain.

The Double Hi-Res Palette Program

To use the Double Hi-Res Palette program, you must first enter the program DHR .PALETTE (shown in **Listing 1**) and save it to a disk that contains the DHR.DRIVER program presented in Double Hi-Res Graphics II.

We've presented the DHR.DRIVER program in **Listing 2** for those of you who missed it last time. To save it to disk use the command:

```
BSAVE DHR.DRIVER,A$9283,L$37D
```

For more information on entering machine language programs, see "A Welcome to New *Nibble* Readers" in the beginning of this issue.

I would suggest leaving the ONERR GOTO 980 statement out of **line 880** until you've debugged your program. That prevents the ONERR from trapping any typos you might have made when entering the listing.

> **"...you can easily set individual color blocks to the desired color."**

When you first run the program you will be asked if you want to work with a (B)lank screen or an (E)xisting picture. If you want to create a Block Shape Table or a new Double Hi-Res picture you should select 'B'. If you want to borrow parts of an existing picture for use as block shapes or if you want to continue working on a picture that you've already started, you should select 'E'.

Next you'll be asked if the picture is already in memory (perhaps loaded or created by a different program). If you answer 'Y' the program will use whatever graphics are already in memory. If you select 'N' you will see a catalog of the disk and you will be asked the name of the picture to load.

All Double Hi-Res pictures will be stored in two disk files, each having "-PAGE1" or "-PAGE1X" added to the file name. Enter the **name only**; the DHR.PALETTE program will take care of adding "-PAGE1" or "-PAGE1X" to the name.

You will then be presented with the Double Hi-Res screen, either blank or displaying your picture, as well as a text window that contains a lot of useful information for help in shape or picture creation. You are probably aware that there are 16,384 bytes in Double Hi-Res memory and 107,520 individual screen dots. The information that appears at the bottom of the screen will apply to the individual dot (bit) of your current position on the screen. Here is a list of information that is provided:

COLUMN=(0-79) This is your current column position.

PAGE=(1 or 1X) Indicates whether your current position is in main or auxiliary memory.

HPLOT 'X'=(0-279) Shows the current Applesoft HPLOT value for this point.

X=(0-559) Indicates the current Double Hi-Res X-coordinate.

Y=(0-191) Indicates the current Y-coordinate value.

ADDRESS= Is the address of the byte that you are presently in.

BYTE VALUE= Shows the decimal value that is stored in the byte.

BITS= Shows the pattern of the bits within this byte (in reverse of normal order). The bit that you are presently on will be shown in INVERSE.

HL-HR=(0-39) Reveals the current address offset of the present byte and is used to determine HR and HL in block shape creation.

COLOR BLOCK=(0-139) Shows which of the 140 color blocks you are in, and can be used to determine when you should move from one color block to the next.

BIT#=(1-4) Indicates which of the four bits within the current color block you are presently sitting on. By setting each of the four bits according to the color pattern chart from the May issue's Part I, you can easily set individual color blocks to the desired color.

MODE=PLOT/NOPLOT/VISITING Indicates the present program drawing mode.

SPEED=FAST/SLOW Indicates whether you are moving four dots or one dot per move.

With the foregoing information at hand you should easily be able to control all aspects of graphics creation on the Double Hi-Res screen.

Choice of Commands

To move about the screen you should use the four arrow keys on your //e. You will note that in **line 320** the program interprets CHR$(32) as being the right arrow key. If you check the manual, you'll find that a right arrow key is CHR$(21), not CHR$(32). In Double Hi-Res, CHR$(21) (<CTRL> U) turns off the 80-column card and Double Hi-Res. To allow the use of the right arrow key, apparently it is interpreted as CHR$(32) while in Double Hi-Res.

The P key sets PLOT MODE in which the cursor changes every bit to the value 1 as it passes.

The N key sets NOPLOT MODE in which the cursor changes every bit it encounters to the value 0.

"The heart of the whole program is contained in one simple statement..."

The V key sets VISITING MODE (maybe that's stretching it a bit for a name) and allows you to move about the screen without affecting any of the bits that you pass over.

The G key sets the screen for FULL SCREEN GRAPHICS and eliminates the text window with all its useful information. You may find that G mode is very useful when drawing on the lower portions of the graphics screen.

The T key restores the TEXT WINDOW so that you can see the screen information.

The S key sets SLOW MODE so that your marker moves one dot per keypress.

The F key sets FAST MODE for moving four dots per move. This mode is useful in moving quickly to different parts of the screen. It will also aid in filling areas with the same color block bit patterns (when moving horizontally), as you will always stay on the same bit in moving from one color block to the next.

Typing <CTRL> S exits the drawing portion of the program and allows you to save your work to disk.

When you enter the saving portion of the program you will be asked if you want to save your work. If you answer 'Y' you will be asked whether you are creating a Block Shape Table or a picture.

If you are creating a Shape Table, you will need to enter the proper values for VT, VB, HR, HL, and SHNUM, which **you should write down** before entering this part of the program. Let's recap what these values are...

Every block shape is a rectangular block of Hi-Res bytes, bounded on the top and bottom by VT and VB respectively, and on the right and left sides by HR and HL respectively.

VT is the topmost Y-coordinate which contains shape bytes. VB is the lowermost Y-coordinate which contains shape bytes. The proper values for VT and VB can be found in the Y=(0-191) block at the bottom of the screen.

HR is the rightmost address offset that contains shape bytes. HL is the leftmost address offset that contains shape bytes. The proper values for HR and HL can be found in the HL-HR=(0-39) block in the text area.

The value that you enter for SHAPE# (SHNUM) indicates where in memory you want to store the Shape Table, and is the high byte of the hex memory address where the shape will begin. SHNUM should be in the range of 64 ($4000) to 146 ($9200). If the SHNUM that you select doesn't allow enough room under the driver, you will be directed to set a lower value.

The shape is then SCANned into memory, the screen erased, and your shape redrawn using the DHR.DRIVER DRAW routine. If you wish to save the shape to disk, simply give it a name, and the shape will be saved and the file locked.

If you're creating a Double Hi-Res picture you will be asked to give it a name, after which it will be saved in two separate disk files. The portion of the picture from main memory will be saved in a file that has "-PAGE1" appended to the name, and the portion from auxiliary memory will be in a file with "-PAGE1X" added to the name. Both files will be locked immediately after the save. The program will automatically check to make sure you don't enter too long a name.

How the Program Works

Once you've gone through this program and understand how it works, you should have a pretty good idea about working with Double Hi-Res. The program uses full and split screen graphics, coordinate translations, soft-switch flipping (lots of that), memory moves, and turns Double Hi-Res and the 80-column text display on and off.

The heart of the whole program is contained in one simple statement **HPLOT XC,Y** which does all the drawing for us. The trick, however, is determining just WHERE to do the HPLOT; then checking all of the appropriate flags to determine what effects this has had in memory.

Here is what the various parts of the program do.

Line 140 loads the DHR.DRIVER into memory (you'll need the driver routines from Parts I and II of the series), then sets HIMEM to protect the driver and calls SETUP to initialize the YTABLE pointers.

Lines 150-160 POKE a short machine code routine into memory. This code is used by the program to determine the bit pattern in any of the 16,384 Double Hi-Res screen bytes.

Lines 170-230 take care of getting the screen set up the way you want it, either blank, with whatever graphics happen to be in memory, or by loading a picture from disk. In loading a picture from disk, the PAGE1X picture is first loaded onto PAGE1, then HGR is used to move it to PAGE1X. After this the PAGE1 part of the picture is loaded.

Lines 240-250 are the same translation routines that we've used before. First they find which column we're in (CX) and flip the PAGE2 soft-switch to select main or auxiliary memory (depending on whether the column is odd or even). Then **line 22** translates our Double Hi-Res X-coordinate (0-559) to the proper HPLOT X-coordinate (0-279) that we need.

Line 260 turns on the extended 80-column card and INITializes Double Hi-Res.

Line 270 sets the starting modes for the program.

Line 280 uses the GET statement to get our input. While the *80-Column Text Card Manual* errata sheet indicates that the Applesoft GET is not supported by the card, I have not yet experienced any incompatibilities.

Lines 280-400 take your input, set the proper flags, and flip the proper switches to accomplish the various commands. Note that the normal Hi-Res POKEs are used by **lines 160-170** to switch between full screen and mixed text/graphics. **Line 195** turns the 80-column card and Double Hi-Res OFF, in preparation for the picture saving part of the program.

Lines 410-440 test to prevent the cursor from moving off the screen, which would result in a program error and subsequent CRASH.

Lines 450-460 handle the actual drawing. If you're VISITING, the HPLOT is bypassed so the screen is unaffected. If you're PLOTting, the current bit is set to 1. A few extra steps are taken to NOPLOT, as we first need to go back to the last point and erase it before moving forward to indicate the current marker location.

Line 470 uses the YADDR routine to find the address of our byte and stores the address in B, with the value of the byte going into V.

Lines 480-560 begin printing the various informational values at the bottom of the screen. You should note the use of POKE 1403 statements (as required by the 80-column card) in place of HTAB or POKE 36.

Lines 570-590 determine the bit pattern of your present byte. First the byte value V is put in memory location 251. Then the bit retriever routine is CALLed eight times. Each time, one bit is taken from location 251 and placed in location 252, where it is tested and built, bit by bit, into the B$ string.

Lines 600-620 print the proper bit pattern "0 1 2 3 4 5 6 7" (the reverse of normal order) and show the bit you're presently on, in INVERSE.

Lines 630-660 print the balance of the screen information and then jump back to **line 90** where the next keyboard command is obtained.

Line 670 is the beginning of the shape or picture saving routines. If you don't want to save your work, execution jumps to **line 790** where the program ends.

Lines 670-770 prompt you to enter the proper values for VT, VB, HR, HL, and SHNUM. The legal ranges for each value are shown. Before proceeding, a check is made to be sure that, based on the size of the shape and the SHNUM that you've selected, there is room for the Shape Table under the driver. If you don't leave enough room, you will be prompted to enter a lower value for SHNUM.

Line 790 handles the Shape Table creation. Note that we have turned the card and Double Hi-Res back ON, turned 80STORE ON, and used SCAN to create the table.

Line 800 erases the shape from the screen using the HGR routine, then reDRAWs the shape using DRAW and the information contained in the Shape Table that is now in memory. Notice that we turned 80STORE OFF before using the DHR.DRIVER's HGR routine. This is because the HGR routine (which is really a memory move) **will not work with 80STORE ON**. After making the move (which erases PAGE1X) we turn 80STORE back ON so that the DRAW routine will work.

Special Note: At this point, let's look at the last statement in **line 800**, POKE 49236,0. Once you have the whole program in memory, running properly AND SAFELY SAVED TO DISK, try this experiment. Remove the POKE 49236,0 from **line 800**, then RUN the program. First draw a small shape, and then enter <CTRL>S to use the saving routines. Enter the shape parameters. When asked if you want to save the shape, answer 'NO'. You should now see the screen go slightly bonkers and everything will hang. Now press <CTRL><RESET> to recover control and LIST the program. As you'll quickly detect, **your program has been destroyed!!** The question of course is WHY??

You'll notice that when you answered no, execution jumped to **line 800** where the 80-column card and Double Hi-Res were turned OFF; then you went to **line 970** which simply CATALOGed the disk. There doesn't seem to be anything harmful in any of these instructions. The method used to turn the card OFF is the same as was used in **line 390**, which didn't cause any problems. As it happens it was the CATALOG that destroyed the program.

Now go back to **line 800** and you'll note that the instruction just before the POKE 49236,0 (which we removed) was a CALL to our DRAW routine, which executed properly and drew the shape. Next, going back to the DRAW routine (see Double Hi-Res Part II, *Nibble* Vol. 5/No. 8 for the DRAW routine listing), you'll find that when we leave the DRAW routine the PAGE2 soft-switch is set to PAGE1X. When using the 80-column card, this soft-switch is used to select between main and auxiliary memory. However, when the card is OFF, this same soft-switch is used to select between HGR and HGR2, or TEXT page 1 and TEXT page 2.

When we turned the card OFF and CATALOGed the disk, the catalog display that we saw (what there was of it) was presented to us on TEXT page 2 because of the way our soft-switch was set. If you refer to page 26 of your *Apple //e Reference Manual*, you'll see that TEXT page 2 resides in memory area $800-$BFF, **the same area of memory that your program occupies**. The catalog of the disk was written on top of your program!!

The whole point of this discussion is to demonstrate the importance of leaving the PAGE2 soft-switch set properly when you exit Double Hi-Res.

Lines 820-860 take care of saving your shape to disk and locking the file.

Line 870 turns OFF the 80-column card and Double Hi-Res.

Lines 880-920 save your Hi-Res picture from main memory (PAGE1 — odd columns) and lock the file.

Line 930 moves the picture portion from auxiliary memory into main memory. First Double Hi-Res is INITialized, the PAGE2 soft-switch is set to main memory, and 80STORE is turned OFF. Then the HGR routine is modified to add the CLC (Clear Carry) instruction, indicating that we want to move from PAGE1X to PAGE1, and the actual move is made. Finally Double Hi-Res is killed again. The portion of the picture from auxiliary memory (PAGE1X — even columns) now resides in main memory.

Lines 940-970 save the second half of your picture to disk, lock the file, CATALOG the disk, and end the program.

Line 980 protects the program from crashing in the event of a disk error during a SAVE. Disk errors could be caused by many things, the most common being trying to reSAVE a picture that's in a locked disk file. If this occurs simply use a different file name.

In Conclusion

Since drawing on the Double Hi-Res screens can be difficult, you should find this program very helpful in block shape and picture creation. The program does quite a job of manipulating Double Hi-Res and the DHR .DRIVER routines, so if you understand how the various program parts work, you should have no problem creating programs on your own.

Next month we'll begin work on developing Double Hi-Res shift animation routines that will provide the smoothest horizontal movement possible for any of your shapes.

LISTING 1: DHR.PALETTE

```
10  REM   ***********************
20  REM   *     DHR.PALETTE     *
30  REM   *  BY ROBERT DEVINE   *
40  REM   * COPYRIGHT (C) 1984 *
50  REM   * BY MICROSPARC, INC *
60  REM   * LINCOLN, MA. Ø1773 *
70  REM   ***********************
140  PRINT  CHR$ (4)"BLOAD DHR.DRIVER": HIMEM:
     37507: CALL 37999: REM   LOAD DRIVER/PRO
     TECT/SET-UP YTABLE
150  FOR X = 768 TO 776: READ Y: POKE X,Y: NEXT
     : REM  POKE BIT RETRIEVER INTO MEMORY
160  DATA  162,Ø,134,252,7Ø,251,38,252,96
170  TEXT : HOME : VTAB 22: PRINT "** COPYRIG
     HT 1984 BY MICROSPARC, INC. **": VTAB 5:
      PRINT "WILL YOU WORK WITH (B)LANK SCREE
     N": PRINT "OR (E)XISTING PICTURE ?";: GET
     A$
180  PRINT : PRINT : IF A$ = "B" THEN  HGR : CALL
     37928: GOTO 26Ø: REM  CLEAR DHR SCREEN
190  PRINT "IS PICTURE IN MEMORY Y/N ?";: GET
     A$: PRINT : IF A$ = "Y" THEN 26Ø
200  HOME : PRINT  CHR$ (4)"CATALOG": PRINT :
      INPUT "WHAT IS PICTURE NAME ?";A$: ONERR
      GOTO 87Ø: REM IF 'FILE NOT FOUND ERROR'
      OCCURS TURN DHR OFF - EXIT PROGRAM
210  PRINT  CHR$ (4)"PR#3": CALL 37953: POKE
     49235,Ø: HOME : CALL 37916: REM   INIT D
     HR/MIXED TEXT/CLEAR TEXT WINDOW
220  PRINT  CHR$ (4)"BLOAD"A$"-PAGE1X": POKE
     49236,Ø: POKE 49152,Ø: POKE 37948,56: CALL
     37928: REM  LOAD PICTURE TO AUXILIARY ME
     MORY
230  PRINT  CHR$ (4)"BLOAD"A$"-PAGE1": GOTO 2
     7Ø: REM  LOAD PICTURE TO MAIN MEMORY
240  POKE 49236,Ø:CX =  INT (X / 7): IF CX /
     2 =  INT (CX / 2) THEN  POKE 49237,Ø
250 XC =  INT (CX / 2) + X / 7 - CX:XC =  INT
     (XC * 7 + .5): RETURN : REM  TRANSLATE X
      (Ø-559) TO XC (Ø-279)
260  PRINT  CHR$ (4)"PR#3": CALL 37953: REM
      CARD ON/INIT DHR
270 P = 3:CB = 1:F = 1:X = Ø:Y = Ø: POKE 23Ø,
     32: GOSUB 24Ø: HCOLOR= P:XO = X:YO = Y: GOTO
     47Ø: REM  INITIALIZE PLOT MODE/COLOR BIT
     /SPEED=SLOW
280  VTAB 15: GET A$: PRINT : IF A$ = "V" THEN
     L = 1: GOTO 51Ø: REM    ENTER 'VISITING'
      MODE-MOVEMENT WILL NOT AFFECT SCREEN
290  IF A$ =  CHR$ (11) THEN Y = Y - F: GOTO
     41Ø: REM    MOVE UP
300  IF A$ =  CHR$ (1Ø) THEN Y = Y + F: GOTO
     41Ø: REM    MOVE DOWN
310  IF A$ =  CHR$ (8) THEN X = X - F: ON (F =
     4) GOTO 41Ø:CB = CB - 1: GOTO 41Ø: REM
     MOVE LEFT
320  IF A$ =  CHR$ (32) THEN X = X + F: ON (F
      = 4) GOTO 41Ø:CB = CB + 1: GOTO 41Ø: REM
      MOVE RIGHT
```

```
330  IF A$ = "F" THEN F = 4: GOTO 51Ø: REM
     CHANGE TO FAST
340  IF A$ = "S" THEN F = 1: GOTO 51Ø: REM  C
     HANGE TO SLOW
350  IF A$ = "G" THEN  POKE 49234,Ø: GOTO 28Ø
     : REM  FULL SCREEN GRAPHICS
360  IF A$ = "T" THEN  POKE 49235,Ø: GOTO 28Ø
     : REM   RETURN TO TEXT WINDOW
370  IF A$ = "N" THEN P = Ø:L = Ø: GOTO 45Ø: REM
      ENTER NOPLOT MODE
380  IF A$ = "P" THEN P = 3:L = Ø: GOTO 45Ø: REM
      ENTER PLOT MODE
390  IF A$ =  CHR$ (19) THEN  HOME : PRINT  CHR$
     (12); CHR$ (21): CALL 37966: GOTO 67Ø: REM
      READY TO SAVE WORK
400  GOTO 28Ø: REM  NO LEGAL COMMAND FOUND
410  IF X < Ø THEN X = Ø: GOTO 28Ø
420  IF X > 559 THEN X = 559: GOTO 28Ø
430  IF Y < Ø THEN Y = Ø: GOTO 28Ø
440  IF Y > 191 THEN Y = 191: GOTO 28Ø
450  GOSUB 24Ø: ON L GOTO 47Ø: HCOLOR= P: IF
     P = 3 THEN  HPLOT XC,Y:XO = X:YO = Y: GOTO
     47Ø
460 X1 = X:Y1 = Y:X = XO:Y = YO: GOSUB 24Ø: HPLOT
     XC,Y:X = X1:Y = Y1: GOSUB 24Ø: HCOLOR= 3
     : HPLOT XC,Y:XO = X:YO = Y: REM  NOPLOT
     MODE
470  POKE 6,Y: CALL 37988:B =  PEEK (38) +  PEEK
     (39) * 256 +  INT (XC / 7):V =  PEEK (B)
480  VTAB 21: PRINT "COLUMN="CX" ";: POKE 140
     3,12: IF CX / 2 =  INT (CX / 2) THEN  PRINT
     "PAGE=1X";: GOTO 5ØØ
490  PRINT "PAGE=1 ";
500  POKE 1403,24: PRINT "HPLOT 'X'="XC"  ";:
      POKE 1403,40: PRINT "X="X"  ";: POKE 14
     Ø3,48: PRINT "Y="Y"  ";: POKE 1403,6Ø: PRINT
     "ADDRESS="B" "
510  VTAB 22: IF L = 1 THEN  PRINT "VISITING-
     MODE";: GOTO 54Ø
520  IF P = 3 THEN  PRINT "PLOT-MODE    ";: GOTO
     54Ø
530  PRINT "NOPLOT-MODE  ";
540  POKE 1403,4Ø: IF F = 1 THEN  PRINT "SPEE
     D=SLOW": GOTO 56Ø
550  PRINT "SPEED=FAST"
560  VTAB 23: PRINT "BYTE VALUE="V"  ";: POKE
     1403,16: PRINT "BITS=";: REM  PRINT VALU
     E OF BYTE
570  POKE 251,V:B$ = ""
580  FOR M = 1 TO 8: CALL 768: IF  PEEK (252)
      = 1 THEN B$ = B$ + "1": GOTO 6ØØ
590 B$ = B$ + "Ø"
600  NEXT : FOR M = 1 TO 8
610  IF M = XC - (7 * ( INT (XC / 7))) + 1 THEN
      INVERSE : REM  INVERSE FOR THE BIT WE'R
     E ON
620  PRINT  MID$ (B$,M,1);: NORMAL : PRINT "
     ";: NEXT : PRINT " ";
630  POKE 1403,4Ø: PRINT "HL-HR=" INT (XC / 7
     )" ";: POKE 1403,52: PRINT "COLOR BLOCK=
     " INT (X / 4)"  ";
640  POKE 1403,7Ø: IF CB = 5 THEN CB = 1
650  IF CB = Ø THEN CB = 4
660  PRINT "BIT #";CB: GOTO 28Ø
670  HOME : VTAB 6: PRINT "WANT TO SAVE YOUR
     WORK (Y/N) ?";: GET A$: PRINT : IF A$ =
     "N" THEN 97Ø
680  VTAB 1Ø: PRINT "(S)HAPE TABLE or (P)ICTU
     RE ?";: GET A$: PRINT : IF A$ <  > "S" AND
     A$ <  > "P" THEN 68Ø
690  IF A$ = "P" THEN 88Ø
700  HOME : INPUT "ENTER VT (Ø -39) ";VT: PRINT

710  INPUT "ENTER VB (VT-39) ";VB: PRINT
720  INPUT "ENTER HR (HL-39) ";HR: PRINT
730  INPUT "ENTER HL (Ø -39) ";HL: PRINT : PRINT
740  INPUT "WHAT IS THE SHAPE# (64-146) ?";SH
     NUM: PRINT
750  IF ((VB - VT + 1) * (HR - HL + 1)) * 2 +
     SHNUM * 256 > 375Ø6 THEN  PRINT : FLASH
     : PRINT "SHAPE TABLE TOO BIG TO FIT UNDE
     R DRIVER": NORMAL : PRINT "SELECT LOWER
     SHAPE #": PRINT : GOTO 74Ø
760  PRINT "ARE ALL SHAPE PARAMETERS CORRECT
     ? (Y/N)";: GET A$: PRINT : IF A$ <  > "Y
     " THEN 7ØØ
770  HIMEM: SHNUM * 256: REM  RESET HIMEM TO
     PROTECT SHAPE
780  POKE 251,SHNUM: POKE 252,VT: POKE 253,VB
     : POKE 254,HR: POKE 255,HL
790  PRINT  CHR$ (4)"PR#3": CALL 37953: POKE
     49153,Ø: CALL 3785Ø: REM  CREATE SHAPE T
     ABLE
800  HGR : POKE 49152,Ø: CALL 37928: POKE 251
     ,SHNUM: POKE 49153,Ø: CALL 3778Ø: POKE 4
     9236,Ø: REM  ERASE SCREEN AND DRAW SHAPE
      FROM TABLE
810  HOME : VTAB 21: PRINT "HERE IS THE SHAPE
      IN YOUR SHAPE TABLE.": PRINT "DO YOU WA
     NT TO SAVE IT (Y/N) ?";: GET A$: PRINT
820  IF A$ = "N" THEN 87Ø
830  HOME : VTAB 22: INPUT "WHAT IS THE NAME
     ?";A$
840  HOME : VTAB 22: PRINT "SAVING SHAPE TO D
     ISK"
850  PRINT  CHR$ (4)"BSAVE "A$",A"SHNUM * 256
     ",L"((VB - VT + 1) * (HR - HL + 1)) * 2
860  PRINT  CHR$ (4)"LOCK"A$
870  PRINT  CHR$ (12); CHR$ (21): CALL 37966:
      GOTO 97Ø
880  PRINT : PRINT : INPUT "ENTER PICTURE NAM
     E: ";A$: ONERR  GOTO 98Ø
890  IF  LEN (A$) > 23 THEN  PRINT : FLASH : PRINT
     "NAME IS TOO LONG": NORMAL : GOTO 88Ø
900  PRINT : PRINT "SAVING PICTURE FROM MAIN
     MEMORY"
910 F$ = A$ + "-PAGE1": PRINT  CHR$ (4)"BSAVE
     "F$",A$2ØØØ,L$2ØØØ"
920  PRINT  CHR$ (4)"LOCK"F$
930  CALL 37953: POKE 49236,Ø: POKE 49152,Ø: POKE
     37948,24: CALL 37928: CALL 37966: REM  M
     OVE PAGE1X TO PAGE1
940  PRINT : PRINT "SAVING PICTURE FROM AUXIL
     IARY MEMORY"
950 F$ = A$ + "-PAGE1X": PRINT  CHR$ (4)"BSAV
     E"F$",A$2ØØØ,L$2ØØØ"
960  PRINT  CHR$ (4)"LOCK"F$
970  PRINT : PRINT  CHR$ (4)"CATALOG": END
980  PRINT : PRINT "DISK ERROR OCCURED-TRY AG
     AIN": GOTO 88Ø: REM  IF TRYING TO RESAVE
      INTO LOCKED DISK FILE - USE A DIFFERENT
      PICTURE NAME
```

```
            KEY PERFECT 4.Ø
                 RUN ON
              DHR.PALETTE
      ==========================
       CODE       LINE# - LINE#
      ------      -------------
        9453         1Ø  -  16Ø
      Ø15197        17Ø  -  26Ø
        F323        27Ø  -  36Ø
        B15D        37Ø  -  46Ø
        BEØ7        47Ø  -  56Ø
        86EE        57Ø  -  66Ø
        D5BB        67Ø  -  76Ø
        E4DA        77Ø  -  86Ø
        B9F9        87Ø  -  96Ø
        374F        97Ø  -  98Ø
     PROGRAM CHECK IS : ØF9B
```

```
          CHECK CODE 3.Ø

          ON: DHR.PALETTE
          TYPE: A

          LENGTH: ØE78
          CHECKSUM: CD
```

LISTING 2: DHR.DRIVER

```
9283- A9 51 2Ø 92 92
9288- A9 26 4C 9F 92 A9 EA 2Ø
929Ø- 9F 92 8D 63 93 8D 72 93
9298- 8D AB 93 8D BA 93 6Ø 8D
92AØ- 64 93 8D 73 93 8D AC 93
92A8- 8D BB 93 6Ø A5 FE C9 27
92BØ- BØ Ø4 E6 FE E6 FF 6Ø A5
92B8- FF FØ Ø4 C6 FE C6 FF 6Ø
92CØ- A5 FC FØ Ø4 C6 FC C6 FD
92C8- 6Ø A5 FD C9 BF BØ Ø4 E6
92DØ- FC E6 FD 6Ø A5 FC 38 E5
92D8- E3 3Ø Ø9 85 FC A5 FD 38
92EØ- E5 E3 85 FD 6Ø A5 FD 18
92E8- 65 E3 C9 CØ BØ Ø9 85 FD
92FØ- A5 FC 18 65 E3 85 FC 6Ø
92F8- A9 ØØ 8D Ø1 CØ 85 FA A5
93ØØ- FD 85 Ø6 2Ø 64 94 A4 FF
93Ø8- 8D 55 CØ 2Ø 2B 93 8D 54
931Ø- CØ 2Ø 2B 93 C8 C4 FE 9Ø
9318- EF FØ ED C6 Ø6 A5 Ø6 C9
932Ø- FF FØ Ø4 C5 FC BØ DC 2Ø
9328- DA 93 6Ø A2 ØØ A1 FA C9
933Ø- 7F FØ 1Ø C9 Ø1 9Ø ØC 86
9338- F9 4A 26 F9 E8 EØ Ø7 9Ø
934Ø- F8 A5 F9 91 26 E6 FA DØ
9348- Ø2 E6 FB 6Ø A9 ØØ 8D Ø1
935Ø- CØ 85 FA A5 FC 85 Ø6 2Ø
9358- 64 94 A4 FE A2 ØØ A1 FA
936Ø- 8D 54 CØ 51 26 91 26 E6
9368- FA DØ Ø2 E6 FB A1 FA 8D
937Ø- 55 CØ 51 26 91 26 E6 FA
9378- DØ Ø2 E6 FB 88 CØ FF FØ
938Ø- Ø4 C4 FF BØ D9 E6 Ø6 A5
9388- Ø6 C9 FF FØ Ø6 C5 FD 9Ø
939Ø- C6 FØ C4 6Ø A9 ØØ 8D Ø1
9398- CØ 85 FA A5 FD 85 Ø6 2Ø
93AØ- 64 94 A4 FE A2 ØØ A1 FA
93A8- 8D 54 CØ 51 26 91 26 E6
93BØ- FA DØ Ø2 E6 FB A1 FA 8D
93B8- 55 CØ 51 26 91 26 E6 FA
```

```
93CØ- DØ Ø2 E6 FB 88 CØ FF FØ
93C8- Ø4 C4 FF BØ D9 C6 Ø6 A5
93DØ- Ø6 C9 FF FØ Ø4 C5 FC BØ
93D8- C6 6Ø A9 ØØ 8D Ø1 CØ 85
93EØ- FA A5 FD 85 Ø6 2Ø 64 94
93E8- A4 FE A2 ØØ 8D 54 CØ B1
93FØ- 26 81 FA E6 FA DØ Ø2 E6
93F8- FB 8D 55 CØ B1 26 81 FA
94ØØ- E6 FA DØ Ø2 E6 FB 88 CØ
94Ø8- FF FØ Ø4 C4 FF BØ DD C6
941Ø- Ø6 A5 Ø6 C9 FF FØ Ø4 C5
9418- FC BØ CA 6Ø A9 Ø4 85 3D
942Ø- 85 43 A9 Ø7 85 3F DØ ØA
9428- A9 2Ø 85 3D 85 43 A9 3F
943Ø- 85 3F A9 ØØ 85 3C 85 42
9438- A9 FF 85 3E 38 2Ø 11 C3
944Ø- 6Ø 8D 5E CØ 8D ØD CØ 8D
9448- 5Ø CØ 8D 57 CØ 6Ø 8D 5F
945Ø- CØ 8D ØC CØ 8D 51 CØ 8D
9458- 56 CØ 8D ØØ CØ 8D 54 CØ
946Ø- 2Ø 58 FC 6Ø A4 Ø6 B1 CE
9468- 85 26 B1 EE 85 27 6Ø A9
947Ø- 8Ø 85 CE A9 94 85 CF A9
9478- 4Ø 85 EE A9 95 85 EF 6Ø
948Ø- ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
9488- 8Ø 8Ø 8Ø 8Ø 8Ø 8Ø 8Ø 8Ø
949Ø- ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
9498- 8Ø 8Ø 8Ø 8Ø 8Ø 8Ø 8Ø 8Ø
94AØ- ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
94A8- 8Ø 8Ø 8Ø 8Ø 8Ø 8Ø 8Ø 8Ø
94BØ- ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ
94B8- 8Ø 8Ø 8Ø 8Ø 8Ø 8Ø 8Ø 8Ø
94CØ- 28 28 28 28 28 28 28 28
94C8- A8 A8 A8 A8 A8 A8 A8 A8
94DØ- 28 28 28 28 28 28 28 28
94D8- A8 A8 A8 A8 A8 A8 A8 A8
94EØ- 28 28 28 28 28 28 28 28
94E8- A8 A8 A8 A8 A8 A8 A8 A8
94FØ- 28 28 28 28 28 28 28 28
94F8- A8 A8 A8 A8 A8 A8 A8 A8
95ØØ- 5Ø 5Ø 5Ø 5Ø 5Ø 5Ø 5Ø 5Ø
95Ø8- DØ DØ DØ DØ DØ DØ DØ DØ
951Ø- 5Ø 5Ø 5Ø 5Ø 5Ø 5Ø 5Ø 5Ø
```

```
LENGTH: Ø37D     ON: DHR.DRIVER
CHECKSUM: 11     TYPE: B
```

```
9518- DØ DØ DØ DØ DØ DØ DØ DØ
952Ø- 5Ø 5Ø 5Ø 5Ø 5Ø 5Ø 5Ø 5Ø
9528- DØ DØ DØ DØ DØ DØ DØ DØ
953Ø- 5Ø 5Ø 5Ø 5Ø 5Ø 5Ø 5Ø 5Ø
9538- DØ DØ DØ DØ DØ DØ DØ DØ
954Ø- 2Ø 24 28 2C 3Ø 34 38 3C
9548- 2Ø 24 28 2C 3Ø 34 38 3C
955Ø- 21 25 29 2D 31 35 39 3D
9558- 21 25 29 2D 31 35 39 3D
956Ø- 22 26 2A 2E 32 36 3A 3E
9568- 22 26 2A 2E 32 36 3A 3E
957Ø- 23 27 2B 2F 33 37 3B 3F
9578- 23 27 2B 2F 33 37 3B 3F
958Ø- 2Ø 24 28 2C 3Ø 34 38 3C
9588- 2Ø 24 28 2C 3Ø 34 38 3C
959Ø- 21 25 29 2D 31 35 39 3D
9598- 21 25 29 2D 31 35 39 3D
95AØ- 22 26 2A 2E 32 36 3A 3E
95A8- 22 26 2A 2E 32 36 3A 3E
95BØ- 23 27 2B 2F 33 37 3B 3F
95B8- 23 27 2B 2F 33 37 3B 3F
95CØ- 2Ø 24 28 2C 3Ø 34 38 3C
95C8- 2Ø 24 28 2C 3Ø 34 38 3C
95DØ- 21 25 29 2D 31 35 39 3D
95D8- 21 25 29 2D 31 35 39 3D
95EØ- 22 26 2A 2E 32 36 3A 3E
95E8- 22 26 2A 2E 32 36 3A 3E
95FØ- 23 27 2B 2F 33 37 3B 3F
95F8- 23 27 2B 2F 33 37 3B 3F
```

```
        KEY PERFECT 4.Ø
            RUN ON
          DHR.DRIVER
==================================
    CODE        ADDR# - ADDR#
    -------     --------------
     272Ø        9283 - 92D2
     2BBF        92D3 - 9322
     25A3        9323 - 9372
     2B1F        9373 - 93C2
     25FD        93C3 - 9412
     2CF3        9413 - 9462
     1F8D        9463 - 94B2
     271A        94B3 - 95Ø2
     2CE8        95Ø3 - 9552
     247C        9553 - 95A2
     29Ø7        95A3 - 95F2
     Ø62E        95F3 - 95FF
   PROGRAM CHECK IS : Ø37D
```

```
CHECK CODE 3.Ø
```