DOUBLE HI-RES GRAPHICS VI

DOS 3.3

In this final part of the Double Hi-Res series, vertical shifting routines are added to the DHR Driver. A short Applesoft program demonstrates their use, and a table summarizes the entry points for the entire DHR Driver, which can be used with any //c or //e with extended 80-column card.

by Robert R. Devine, Computers and You, Mellor Park Mall, 1855 North West Ave., El Dorado, AR 71730

In the Double Hi-Res Graphics series, we've created routines that SCAN shapes into shape tables, DRAW them on the screen (from top to bottom, or bottom to top), and SHIFT them bit-by-bit, left or right across the screen. We've already shown how you can move shapes up or down using the DRAW routine; this month we'll look at how to shift shapes vertically on the Double Hi-Res screen. We'll add two final routines to the DHR driver: **SHFTDN (CALL 37229)** moves a shape down one screen dot and **SHFTUP (CALL 37301)** moves a shape up one screen dot.

ENTERING THE NEW ROUTINES

To add these routines to the DHR driver, first load the driver into memory by typing:

```
BLOAD DHR.DRIVER $91FE
```

Now enter the Monitor using CALL-151, and type in the hex code shown in **Listing 1**. When it's all in memory, save the completed driver to disk by typing:

```
BSAVE DHR.DRIVER $916D,A$916D,
L$493
```

(For those who may have missed an issue, the entire driver is shown in **Listing 2**.) If you plan to use an assembler to enter the routines, you can use the source code part of the listing, which contains all the needed documentation describing how the routines work. (For help in entering machine language code, see "A Welcome to New *Nibble* Readers" at the beginning of this issue.)

HOW THE VERTICAL SHIFT ROUTINES WORK

As with our horizontal shift routines, the DRAW routine is only used to initially place the shape on the screen. From then on, you can simply shift the shape to wherever you want. Since neither the horizontal nor the vertical shift routines use shape tables to do their animation, you can shift *parts of shapes*, or even *background* graphics for which no shape tables exist.

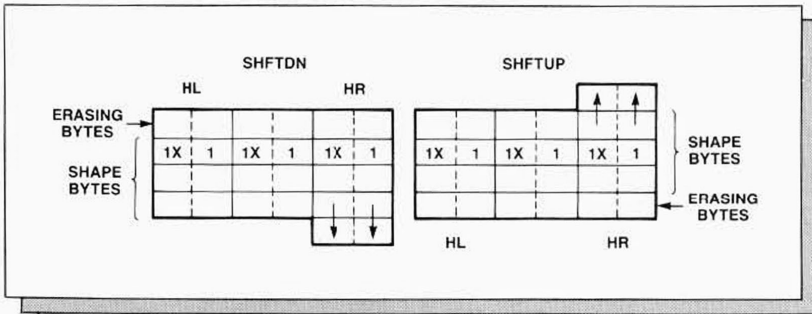
Both SHFTDN and SHFTUP are self-erasing routines. In our horizontal shift routines, an extra column of shifting bytes was placed *ahead* of the shape to handle movement and erasure. In the vertical shift routines it will be necessary to add an extra row of erasing bytes *behind* the shape. This extra row of bytes comes along behind the shape and cleans up any mess that it might try to leave behind.

Let's look at **Figure 1** to see how the vertical shift routines work. The first thing that you should notice is that, unlike SHIFTR and SHIFTL, which shift one bit per move, the SHFTDN and SHFTUP routines shift one byte per move. SHFTDN starts at VB/HR (like DRAW) and pulls the shape down, while SHFTUP starts at VT/HR (like DRAWDN) and pulls the shape up.

At each address there are two screen bytes to deal with, one on PAGE1 and the other on PAGE1X. First, we go to PAGE1X, get the shape byte from the screen and put it into a temporary holder at $08. Then we go to PAGE1, get that shape byte from the screen and store it in the X-Register. Next, we use the Apple's INCRY and DECRY routines to find the screen address directly above or below (depending on whether we're in SHFTUP or SHFTDN).

Now we retrieve the shape byte from the

FIGURE 1: Vertical Shifting Routines



X-Register and place it in the new address on PAGE1. Then we return to PAGE1X, retrieve that shape byte from $08 and place it in the new address on PAGE1X. Finally, we again use INCRY or DECRY to return to the original address.

Using this process, we step through all the addresses in the shape and move every byte up or down one position. Since the last line of addresses to be processed in each shape contains the same bit pattern as the background (normally, but not necessarily, all zeros), we automatically erase the last line of shape bytes.

TESTING THE VERTICAL SHIFT ROUTINES

Now that we have the new routines added to the driver, let's try them out and see how they work. First, enter the Applesoft program shown in **Listing 3** and save it on disk with the command:

SAVE VERTICAL.SHIFT.DEMO

On the same disk you will need a copy of DHR.DRIVER $916D and SHAPE#144, which we created in Part II. This is the spaceship shape that has an extra row of erasing bytes both above and below the actual shape bytes. If you do not have SHAPE#144 on disk, use the Monitor to enter the code shown in **Listing 4**, and save it on disk with the command:

BSAVE SHAPE#144,A$9000,L$54

When you run the program you'll see your shape move smoothly up and down the screen, as well as shifting rightward 14 dots each time it gets to the top or bottom of the screen.

Lines 80-130 should be old hat to you by now, as all they do is load the DHR driver and shape, then initialize Double Hi-Res and DRAW the shape at its starting point.

Line 140 adds one extra address to the right side of the shape for use as shifting bytes when we move rightward at the top and bottom of the screen. We could, if we wanted, simply install the extra shifting address just before each rightward move and then remove it again to eliminate the need for SHFTDN and SHFTUP to process extra bytes.

Line 150 moves the shape from the top to the bottom of the screen using CALL 37229. Note that we never needed to worry about changing the values of VT or VB because the SHFTDN routine automatically INCrements these values in readiness for the next downward move.

Lines 160 and 180 use the subroutine at **line 210** to shift the shape rightward 14 dots at the top or bottom of the screen. **Line 170** moves the shape from the bottom, back to the top of the screen, with SHFTUP again taking care of setting VT and VB for the next move.

TABLE 1
DHR Driver Map

Routine Name	Hex Address	CALL Address	Function
YTABLE	$9480		Hi-Res screen addresses
SETUP	$946F	37999	Initialize YTABLE pointers
YADDR	$9464	37988	Read byte 0 address
KILL	$944E	37966	Exit Double Hi-Res
INIT	$9441	37953	Enter Double Hi-Res
HGR	$9428	37928	Move a DHR graphics page
HOME	$941C	37916	Move a text page
SCAN	$93DA	37850	Create a block shape
DRAW	$9394	37780	Draw a shape from bottom to top
DRAWDN	$934C	37708	Draw a shape from top to bottom
REVDIR	$92F8	37624	Draw a reversed shape
YINCRD	$92E5	37605	Add YINCR to VT and VB
YINCRU	$92D4	37588	Subtract YINCR from VT and VB
GODOWN	$92C9	37577	INCrement VT and VB
GOUP	$92C0	37568	DECrement VT and VB
MOVELF	$92B7	37559	DECrement HR and HL
MOVERT	$92AC	37548	INCrement HR and HL
EOROFF	$928D	37517	Disable the EOR function
EORON	$9283	37507	Enable the EOR function
SHIFTR	$9244	37444	Move right one dot
SHIFTL	$91FE	37374	Move left one dot
SHFTUP	$91B5	37301	Move up one dot
SHFTDN	$916D	37229	Move down one dot

Special POKEs for Use With the Driver

POKE 251,SHNUM	Set high byte of the shape table address
POKE 37781,LO	Set low byte of the shape table address (DRAW)
POKE 37709,LO	Set low byte of the shape table address (DRAWDN)
POKE 37624,LO	Set low byte of the shape table address (REVDIR)
POKE 252,VT	Set top Y-coordinate of the shape
POKE 253,VB	Set bottom Y-coordinate of the shape
POKE 254,HR	Set rightmost address offset
POKE 255,HL	Set leftmost address offset
POKE 6,Y	Set Y-coordinate for YADDR
POKE 37948,24	Set HGR and HOME to move PAGE1X to PAGE1
POKE 37948,56	Set HGR and HOME to move PAGE1 to PAGE1X
POKE 227,YINCR	Set value for use by YINCRU and YINCRD

The values 0-179 and 179-0 used in the FOR-NEXT loops are simply a way of indicating the various values that will be associated with VT.

With the addition of the SHFTDN and SHFTUP routines, the DHR driver is complete. (See **Table 1** for an easy-reference map of the driver.) You may think of some other features that you'd like to add, which can be easily placed below the present driver routines. The entire driver uses only 1,171 bytes of memory, so for a rather small investment in space you now have a complete set of graphics routines for animation on the Double Hi-Res screen. Perhaps some of you will create your own programs using the driver, and send them to Mike Harvey for publication in *Nibble*!

LISTING 1: SHIFTU/D

```
                1000  .OR $916D       ** SHIFTU/D
                1010  .TA $800         ** BY ROBERT DEVINE
                1015 ** COPYRIGHT 1985 BY MICROSPARC, INC.
0008-           1020 HLDR .EQ $08      ** S-C ASSEMBLER
00FC-           1040 VT .EQ $FC
00FD-           1050 VB .EQ $FD
00FE-           1060 HR .EQ $FE
00FF            1070 HL .EQ $FF
0026-           1080 HBASL .EQ $26
0027-           1090 HBASH .EQ $27
0006-           1100 YO .EQ $06
9464-           1130 YADDR .EQ $9464
F504-           1140 INCRY .EQ $F504
F4D5-           1150 DECRY .EQ $F4D5
C054-           1160 PAGE1 .EQ $C054
C055-           1170 PAGE1X .EQ $C055
916D- A5 FD     1180 SHFTDN LDA VB     ** CALL 37229 TO ENTER
916F- C9 BD     1190  CMP #189         ** IS VB>=189 ? (158 FOR HGR)
9171- B0 41     1200  BCS RTN1         ** YES-WE'LL GO OFF SCREEN-EXIT
9173- 85 06     1210  STA YO           ** STORE IN $6 FOR USE BY YADDR
9175- 20 64 94  1220 L1 JSR YADDR      ** GET BYTE ZERO ADDRESS
9178- A4 FE     1230  LDY HR           ** SET Y-REG TO RIGHTMOST ADDRESS
917A- 8D 55 C0  1235 L2 STA PAGE1X     ** READ AUXILIARY MEMORY
917D- B1 26     1240  LDA (HBASL),Y    ** GET SHAPE BYTE FROM SCREEN
917F- 85 08     1250  STA HLDR         ** STORE IN HOLDER
9181- 8D 54 C0  1255  STA PAGE1        ** READ/DRAW MAIN MEMORY
9184- B1 26     1260  LDA (HBASL),Y    ** GET SHAPE BYTE FROM SCREEN
9186- AA        1265  TAX              ** STORE IN X-REGISTER
9187- 20 04 F5  1270  JSR INCRY        ** POINT TO NEXT LOWER ADDRESS
918A- 8A        1275  TXA              ** RETRIEVE SCREEN BYTE
918B- 91 26     1280  STA (HBASL),Y    ** LOAD BYTE TO SCREEN
918D- 8D 55 C0  1285  STA PAGE1X       ** DRAW AUXILIARY MEMORY
9190- A5 08     1290  LDA HLDR         ** RETRIEVE SCREEN BYTE
9192- 91 26     1295  STA (HBASL),Y    ** LOAD BYTE ON SCREEN
9194- 8D 54 C0  1297  STA PAGE1        ** EXECUTE ROM INSTRUCTIONS
9197- 20 D5 F4  1300  JSR DECRY        ** RETURN TO ORIGINAL ADDRESS
919A- 88        1305  DEY              ** POINT TO NEXT ADDRESS <---
919B- 18        1310  CLC
919C- C0 FF     1320  CPY #$FF        ** HAS Y-REGISTER PASSED 0?
919E- F0 04     1330  BEQ NXTLN1       ** YES-GOTO NEXT LINE
91A0- C4 FF     1340  CPY HL           ** HAVE WE REACHED HL?
91A2- B0 D6     1350  BCS L2           ** NO-GOTO NEXT ADDRESS
91A4- C6 06     1360 NXTLN1 DEC YO     ** MOVE UP TO NEXT LINE
91A6- A5 06     1370  LDA YO           ** GET NEXT Y-COORDINATE
91A8- C9 FF     1380  CMP #$FF         ** HAVE WE DONE 0?
91AA- F0 04     1390  BEQ J1           ** YES-WE'RE DONE
91AC- C5 FC     1400  CMP VT           ** HAVE WE REACHED VT?
91AE- B0 C5     1410  BCS L1           ** NO-CONTINUE
91B0- E6 FC     1420 J1 INC VT         ** MOVE VT DOWN 1
91B2- E6 FD     1430  INC VB           ** MOVE VB DOWN 1
91B4- 60        1440 RTN1 RTS          ** DONE-EXIT ROUTINE
91B5- A5 FC     1500 SHFTUP LDA VT     ** CALL 37301 TO ENTER
91B7- C9 01     1510  CMP #1           ** IS VT<1?
91B9- 90 42     1520  BCC RTN2         ** YES-WE'LL GO OFF SCREEN-EXIT
91BB- E6 FD     1530  INC VB           ** MOVE VB DOWN 1 LINE
91BD- 85 06     1540  STA YO           ** STORE VT IN $6 FOR USE BY YADDR
91BF- 8D 55 C0  1550  STA PAGE1X       ** READ AUXILIARY MEMORY
91C2- 20 64 94  1560 LP1 JSR YADDR     ** GET ADDRESS OF BYTE 0
91C5- A4 FE     1570  LDY HR           ** SET Y-REG TO RIGHTMOST ADDRESS
91C7- B1 26     1580 LP2 LDA (HBASL),Y ** GET SHAPE BYTE FROM SCREEN
91C9- 85 08     1590  STA HLDR         ** STORE IN HOLDER
91CB- 8D 54 C0  1600  STA PAGE1        ** READ/DRAW MAIN MEMORY
91CE- B1 26     1610  LDA (HBASL),Y    ** GET SHAPE BYTE FROM SCREEN
91D0- AA        1620  TAX              ** STORE IN X-REGISTER
91D1- 20 D5 F4  1630  JSR DECRY        ** POINT TO NEXT HIGHER ADDRESS
91D4- 8A        1640  TXA              ** RETRIEVE SCREEN BYTE
91D5- 91 26     1650  STA (HBASL),Y    ** LOAD BYTE ON SCREEN
91D7- 8D 55 C0  1660  STA PAGE1X       ** DRAW AUXILIARY MEMORY
91DA- A5 08     1670  LDA HLDR         ** RETRIEVE SCREEN BYTE
91DC- 91 26     1680  STA (HBASL),Y    ** LOAD BYTE ON SCREEN
91DE- 20 04 F5  1690  JSR INCRY        ** RETURN TO ORIGINAL ADDRESS
91E1- 88        1700  DEY              ** POINT TO NEXT ADDRESS <---
91E2- 18        1710  CLC
91E3- C0 FF     1720  CPY #$FF        ** HAS Y-REG PASSED 0?
91E5- F0 04     1730  BEQ NXTLN2       ** YES-GOTO NEXT LINE
91E7- C4 FF     1740  CPY HL           ** HAVE WE REACHED HL?
91E9- B0 DC     1750  BCS LP2          ** NO-GOTO NEXT ADDRESS
91EB- E6 06     1760 NXTLN2 INC YO     ** MOVE DOWN TO NEXT LINE
91ED- A5 06     1770  LDA YO           ** GET NEW Y-COORDINATE
91EF- C9 BE     1780  CMP #190         ** HAVE WE DONE 190? (159 FOR HGR)
91F1- F0 04     1790  BEQ J2           ** YES-WE'RE DONE
91F3- C5 FD     1800  CMP VB          ** HAVE WE REACHED VB?
91F5- 90 CB     1810  BCC LP1          ** NO-CONTINUE
91F7- C6 FD     1820 J2 DEC VB         ** RESTORE ORIGINAL VB
91F9- C6 FD     1830  DEC VB          ** MOVE VB UP 1 LINE
91FB- C6 FC     1840  DEC VT          ** MOVE VT UP 1 LINE
91FD- 60        1850 RTN2 RTS          ** DONE-EXIT ROUTINE
```

END OF LISTING 1

LISTING 3: VERTICAL.SHIFT.DEMO

```
10  REM ************************
20  REM * VERTICAL.SHIFT.DEMO *
30  REM *  BY  ROBERT DEVINE   *
40  REM * COPYRIGHT (C)  1985 *
50  REM * BY MICROSPARC, INC. *
60  REM * CONCORD, MA  01742  *
70  REM ************************
80  PRINT  CHR$ (4)"BLOAD DHR.DRIVER $916D": CALL
    37999: HIMEM: 37229
90  PRINT  CHR$ (4)"BLOAD SHAPE#144"
100  CALL 37953: REM  INIT
110  HGR : CALL 37928: REM  CLEAR DHR SCREEN
120  POKE 49153,0: POKE 49234,0: REM  80STORE
     /FULL SCREEN
130  POKE 251,144: POKE 252,0: POKE 253,13: POKE
     254,2: POKE 255,0: CALL 37780: REM  DRAW
      SHAPE ON THE SCREEN
140  POKE 254,3: REM  ADD 1 ADDRESS TO THE RI
     GHT
150  FOR VT = 0 TO 179: CALL 37229: NEXT VT: REM
       MOVE SHAPE DOWN
160  GOSUB 210
170  FOR VT = 179 TO 0 STEP  - 1: CALL 37301:
     NEXT VT: REM  MOVE SHAPE UP
180  GOSUB 210
190  IF  PEEK (254) = 39 THEN  POKE 49152,0: GOTO
     110
200  GOTO 150
210  FOR SHFT = 1 TO 14: CALL 37444: NEXT SHF
     T: CALL 37548: RETURN : REM  MOVE RIGHT
     14 DOTS
```

END OF LISTING 3

LISTING 2: DHR.DRIVER $916D

```
916D- A5 FD C9
9170- BD B0 41 85 06 20 64 94
9178- A4 FE 8D 55 C0 B1 26 85
9180- 08 8D 54 C0 B1 26 AA 20
9188- 04 F5 8A 91 26 8D 55 C0
9190- A5 08 91 26 8D 54 C0 20
9198- D5 F4 88 18 C0 FF F0 04
91A0- C4 FF B0 D6 C6 06 A5 06
91A8- C9 FF F0 04 C5 FC B0 C5
91B0- E6 FC E6 FD 60 A5 FC C9
91B8- 01 90 42 E6 FD 85 06 8D
91C0- 55 C0 20 64 94 A4 FE B1
91C8- 26 85 08 8D 54 C0 B1 26
91D0- AA 20 D5 F4 8A 91 26 8D
91D8- 55 C0 A5 08 91 26 20 04
91E0- F5 88 18 C0 FF F0 04 C4
91E8- FF B0 DC E6 06 A5 06 C9
91F0- BE F0 04 C5 FD 90 CB C6
91F8- FD C6 FD C6 FC 60 A5 FD
9200- 85 06 20 64 94 A4 FE A2
9208- 00 86 08 8D 54 C0 B1 26
9210- A6 08 F0 02 09 80 18 6A
9218- 91 26 8D 55 C0 B1 26 90
9220- 02 09 80 18 6A 91 26 A2
9228- 00 90 01 E8 86 08 88 C0
9230- FF F0 04 C4 FF B0 D4 C6
9238- 06 A5 06 C9 FF F0 04 C5
9240- FC B0 BF 60 A5 FD 85 06
9248- 20 64 94 A4 FF A2 00 86
9250- 08 8D 55 C0 18 A5 08 F0
9258- 01 38 B1 26 2A 91 26 8D
9260- 54 C0 2A B1 26 2A 91 26
9268- 86 08 2A 90 02 E6 08 C8
9270- C4 FE 90 DD F0 DB C6 06
9278- A5 06 C9 FF F0 04 C5 FC
9280- B0 C6 60 A9 51 20 92 92
9288- A9 26 4C 9F 92 A9 EA 20
9290- 9F 92 8D 63 93 8D 72 93
9298- 8D AB 93 8D BA 93 60 8D
92A0- 64 93 8D 73 93 8D AC 93
92A8- 8D BB 93 60 A5 FE C9 27
92B0- B0 04 E6 FE E6 FF 60 A5
92B8- FF F0 04 C6 FE C6 FF 60
92C0- A5 FC F0 04 C6 FC C6 FD
92C8- 60 A5 FD C9 BF B0 04 E6
92D0- FC E6 FD 60 A5 FC 38 E5
92D8- E3 30 09 85 FC A5 FD 38
92E0- E5 E3 85 FD 60 A5 FD 18
92E8- 65 E3 C9 C0 B0 09 85 FD
92F0- A5 FC 18 65 E3 85 FC 60
92F8- A9 00 8D 01 C0 85 FA A5
9300- FD 85 06 20 64 94 A4 FF
9308- 8D 55 C0 20 2B 93 8D 54
9310- C0 20 2B 93 C8 C4 FE 90
9318- EF F0 ED C6 06 A5 06 C9
9320- FF F0 04 C5 FC B0 DC 20
9328- DA 93 60 A2 00 A1 FA C9
9330- 7F F0 10 C9 01 90 0C 86
9338- F9 4A 26 F9 E8 E0 07 90
9340- F8 A5 F9 91 26 E6 FA D0
9348- 02 E6 FB 60 A9 00 8D 01
9350- C0 85 FA A5 FC 85 06 20
9358- 64 94 A4 FE A2 00 A1 FA
9360- 8D 54 C0 51 26 91 26 E6
9368- FA D0 02 E6 FB A1 FA 8D
9370- 55 C0 51 26 91 26 E6 FA
9378- D0 02 E6 FB 88 C0 FF F0
9380- 04 C4 FF B0 D9 E6 06 A5
9388- 06 C9 FF F0 06 C5 FD 90
9390- C6 F0 C4 60 A9 00 8D 01
9398- C0 85 FA A5 FD 85 06 20
93A0- 64 94 A4 FE A2 00 A1 FA
93A8- 8D 54 C0 51 26 91 26 E6
93B0- FA D0 02 E6 FB A1 FA 8D
93B8- 55 C0 51 26 91 26 E6 FA
93C0- D0 02 E6 FB 88 C0 FF F0
93C8- 04 C4 FF B0 D9 C6 06 A5
93D0- 06 C9 FF F0 04 C5 FC B0
93D8- C6 60 A9 00 8D 01 C0 85
93E0- FA A5 FD 85 06 20 64 94
93E8- A4 FE A2 00 8D 54 C0 B1
93F0- 26 81 FA E6 FA D0 02 E6
93F8- FB 8D 55 C0 B1 26 81 FA
9400- E6 FA D0 02 E6 FB 88 C0
9408- FF F0 04 C4 FF B0 DD C6
9410- 06 A5 06 C9 FF F0 04 C5
9418- FC B0 CA 60 A9 04 85 3D
9420- 85 43 A9 07 85 3F D0 0A
9428- A9 20 85 3D 85 43 A9 3F
9430- 85 3F A9 00 85 3C 85 42
9438- A9 FF 85 3E 38 20 11 C3
9440- 60 8D 5E C0 8D 0D C0 8D
9448- 50 C0 8D 57 C0 60 8D 5F
9450- C0 8D 0C C0 8D 51 C0 8D
9458- 56 C0 8D 00 C0 8D 54 C0
9460- 20 58 FC 60 A4 06 B1 CE
9468- 85 26 B1 EE 85 27 60 A9
9470- 80 85 CE A9 94 85 CF A9
9478- 40 85 EE A9 95 85 EF 60
9480- 00 00 00 00 00 00 00 00
9488- 80 80 80 80 80 80 80 80
9490- 00 00 00 00 00 00 00 00
9498- 80 80 80 80 80 80 80 80
94A0- 00 00 00 00 00 00 00 00
94A8- 80 80 80 80 80 80 80 80
94B0- 00 00 00 00 00 00 00 00
94B8- 80 80 80 80 80 80 80 80
94C0- 28 28 28 28 28 28 28 28
94C8- A8 A8 A8 A8 A8 A8 A8 A8
94D0- 28 28 28 28 28 28 28 28
94D8- A8 A8 A8 A8 A8 A8 A8 A8
94E0- 28 28 28 28 28 28 28 28
94E8- A8 A8 A8 A8 A8 A8 A8 A8
94F0- 28 28 28 28 28 28 28 28
94F8- A8 A8 A8 A8 A8 A8 A8 A8
9500- 50 50 50 50 50 50 50 50
9508- D0 D0 D0 D0 D0 D0 D0 D0
9510- 50 50 50 50 50 50 50 50
9518- D0 D0 D0 D0 D0 D0 D0 D0
9520- 50 50 50 50 50 50 50 50
9528- D0 D0 D0 D0 D0 D0 D0 D0
9530- 50 50 50 50 50 50 50 50
9538- D0 D0 D0 D0 D0 D0 D0 D0
9540- 20 24 28 2C 30 34 38 3C
9548- 20 24 28 2C 30 34 38 3C
9550- 21 25 29 2D 31 35 39 3D
9558- 21 25 29 2D 31 35 39 3D
9560- 22 26 2A 2E 32 36 3A 3E
9568- 22 26 2A 2E 32 36 3A 3E
9570- 23 27 2B 2F 33 37 3B 3F
9578- 23 27 2B 2F 33 37 3B 3F
9580- 20 24 28 2C 30 34 38 3C
9588- 20 24 28 2C 30 34 38 3C
9590- 21 25 29 2D 31 35 39 3D
9598- 21 25 29 2D 31 35 39 3D
95A0- 22 26 2A 2E 32 36 3A 3E
95A8- 22 26 2A 2E 32 36 3A 3E
95B0- 23 27 2B 2F 33 37 3B 3F
95B8- 23 27 2B 2F 33 37 3B 3F
95C0- 20 24 28 2C 30 34 38 3C
95C8- 20 24 28 2C 30 34 38 3C
95D0- 21 25 29 2D 31 35 39 3D
95D8- 21 25 29 2D 31 35 39 3D
95E0- 22 26 2A 2E 32 36 3A 3E
95E8- 22 26 2A 2E 32 36 3A 3E
95F0- 23 27 2B 2F 33 37 3B 3F
95F8- 23 27 2B 2F 33 37 3B 3F
```

END OF LISTING 2

```
          KEY PERFECT 4.0
              RUN ON
        DHR.DRIVER $916D
=================================
   CODE        ADDR# - ADDR#
  ------     --------------
   273E        916D - 91BC
   2A60        91BD - 920C
   2AD1        920D - 925C
   2BBF        925D - 92AC
   25E1        92AD - 92FC
   2B2A        92FD - 934C
   2842        934D - 939C
   25B6        939D - 93EC
   2A00        93ED - 943C
   2945        943D - 948C
   206A        948D - 94DC
   299A        94DD - 952C
   2D0B        952D - 957C
   287D        957D - 95CC
   1C18        95CD - 95FF
 PROGRAM CHECK IS : 0493
```

LISTING 4: SHAPE#144

```
9000- 00 00 00 00 00 00 00 00
9008- 01 70 00 00 00 00 7F 7F
9010- 60 00 00 0F 7F 7F 7E 00
9018- 00 3F 7F 7F 7F 40 01 7F
9020- 7F 7F 7F 70 07 7F 7F 7F
9028- 7F 7C 1F 43 61 70 78 3F
9030- 1F 7F 7F 7F 7F 7F 01 7F
9038- 7F 7F 7F 70 00 0F 7F 7F
9040- 7E 00 00 00 7F 7F 60 00
9048- 00 00 07 7C 00 00 00 00
9050- 00 00 00 00
```

END OF LISTING 4