Block Shape Animation — III

by Robert R. Devine
P.O. Box 10
Adona, Arkansas 72001

INTRODUCTION

In this series about Block shapes I've been showing how you can take graphics images from almost any program you might have and convert them into Block Shape Tables. While this is certainly true, it's a little easier said than done. The main problem is that since you don't know the coordinates of the existing shapes, it's somewhat difficult, short of hit-and-miss, to know the proper values for **VT, VB, HR, and HL (see NIBBLE Vol. 4 Nos. 3 and 4).**

The following program is designed to overcome all of these problems. The program that I wound up with is quite a bit more than I had in mind when I decided to write it. With Block Shape Maker, you can actually create your shapes on a blank screen (however, that wasn't the original idea). If you're trying to copy some existing shapes into a Block table, the program will easily provide you with the current value of the Y-coordinate, as it relates to VT and VB.

It will also provide the current value of the X-coordinate, as well as the value of the current horizontal byte that you're in, as it relates to HR or HL. If you need further details, you'll also be provided with the Hi-Res screen address (in decimal form) of the byte that you're in, as well as the value (in decimal form) that the byte contains.

Finally, you'll see the pattern of the bits within the byte (in reverse order), as well as the individual bit that you're presently sitting on. If that doesn't give you all the information that you want . . . nothing will!

While the program is specifically designed to work with the Block Shape Driver, being presented in Nibble, once created, the shapes should run with any other Block type routines that you may run across. Once created, the shape table doesn't care what program created, or wishes to run it.

WHAT YOU'LL NEED

To RUN this program you'll need to enter the Applesoft listing shown here, and have the Block routines on the same disk. The routines through Part 1 will do; however, I've written it to load the routines through Part 2, (which is reprinted as a memory listing in this article — **listing 2**).

A WORD OF WARNING

I've run the program into the ground testing it out, and it works as advertised. There is, however, one thing that seems to act strangely. If you RUN the program without ever having entered HGR, either to load a picture, or to simply clear the screen, (with the normal vertical start-up bars displayed on the screen), the program will eventually begin to modify itself, finally stopping with some sort of strange error message.

So always be sure to work with a clear screen, or a picture on-board. If anyone out there has an explanation, I'd love to hear from you (and be sure to send a copy of your letter to NIBBLE).

HOW TO USE THE SHAPE MAKER

When you first RUN the program, it will ask if you have a picture in memory. If you don't, it will ask you to load one, and then stop. If you do have a picture loaded, it will next ask if it's on the graphics page 1. If it's on page 2 (HGR2), you will be given brief instructions on how to move it to page 1. We need to work on page 1 so that the Text window is available for our information.

If you want to use the Shape Maker to create a graphics shape, then make sure that you have typed **HGR** from the keyboard before running the program. Then answer **Y** when asked whether a picture is loaded and you will be presented with a blank screen upon which to draw your picture.

SUMMARY OF COMMANDS

U Move dot UP

D Move dot DOWN

→ (Right arrow) Move dot RIGHT

← (Left arrow) Move dot LEFT

F Change to FAST-5 dots per move

S Change to SLOW-1 dot per move

(Control/S) Exit to shape saving routines

(Space Bar) Change to full-screen graphics

ESCape Change back to TEXT window

P Change to PLOT mode

N Change to NO-PLOT mode

R Reset color bit — Cancel HPLOT mode

C Cancel HPLOT mode for use with color bytes

W Reset HPLOT mode for use with white bytes.

The entire picture that you loaded will then be displayed on the screen. Using the U and D keys, you can move UP and DOWN, or RIGHT and LEFT with the arrow keys. You can change from one dot per move, to five dots per move with the F (Fast) key. To return to one dot per move, use the S (Slow) key.

As you move around the screen, the present X and Y coordinates of the dot will appear at the bottom of the screen, as well as the address of the byte you're presently in. BYTE will be the current value of HR or HL. When you're on top of the topmost dot of the shape, Y will equal VT; when on top of the lowermost dot, Y will equal VB. When on top of the rightmost or leftmost dot in the shape, BYTE will equal HR or HL, as appropriate.

If your shape extends down into the Text window part of the screen, press the Space Bar to change to full-screen graphics so that you can see what you're doing. Once the dot is properly located, press the ESCape key to get back to the Text window to read the needed values.

On the bottom line you will find the present value stored in the byte, as well as the bit pattern in the byte. The bit pattern is shown in reverse order due to the normal reversing effects of bytes on the Hi-Res screen. The color bit will be shown in INVERSE, and the bit that your dot is presently on will be shown in FLASHing mode.

You'll probably want to move around the screen in P (PLOT) mode so that you can see where you are. When your dot enters a byte it will automatically change the bit it's sitting on to a **1**. If you want to turn that bit back off, press N (NO PLOT) and the bit will change to **0**. Using the P and N keys, you can move about your shape turning bits on and off at will. The effects that you're having on your shape will be shown in the bit display as well as on the screen. If you plan to save the shape without any modification, be sure that you use the P and N keys to reset each bit to its prior status as you move about the shape.

SAVING THE SHAPE

Once you've established, and WRITTEN DOWN, the values of VT, VB, HR, and HL, enter CTRL-S to enter the SAVE portion of the program.

At this point you'll be asked to enter the values of VT, VB, HR, and HL. You will also need to choose a Shape number (SHAPE#) and enter it now. **Be careful to allow enough space for your shape to fit UNDER the driver.**

Once the proper values are entered, your shape is SCANned into a Block Shape Table. Finally, the picture that you loaded is erased, and the shape that you created is DRAWn on the screen, using the data now stored in the Shape Table. Here you will be asked if you want to save your shape to disk. If it's not what you wanted, answer "N" and the program will end. To try again you'll need to reload the original picture, and reRUN the program. If you answer "Y", your shape will be SAVEd to disk with the name of your choice.

HOW TO LOAD YOUR ORIGINAL PICTURE

It's difficult for a program like this to automatically load your picture, as it has no idea where the picture is, what it's called, or what method is being used to create it. If your picture is generated by another program, first RUN that program, and stop it when the needed graphics are on the screen. Then RUN the Shape Maker program which will load the driver, and use the graphics that are in memory.

If you plan to play around with the same picture several times, and don't want to go through the trouble of running the first program, as well as the Shape Maker every time, you might try saving the desired graphics from the first program as a disk file. To do this, RUN the first program until the desired graphics are on the screen. Then stop the program with CTRL-C, or RESET, as appropriate. **To save the entire graphics screen, enter BSAVE PICTURE, A$2000, L$2000 to save page 1, or BSAVE PICTURE A$4000, L$2000 to save page 2.**

Now whenever you want to reload the picture, you can simply enter BLOAD PICTURE.

HOW THE PROGRAM WORKS

Line 20 loads the Block Shape Driver and sets HIMEM to protect it.

Lines 30-40 POKE a brief machine language routine into memory which will be used to find the bit patterns in the Hi-Res screen bytes and set the color bit.

Lines 50-90 check to see if a picture is loaded, and is on page 1.

Line 100 displays the original Hi-Res picture, and sets the Move flag to slow.

Line 110 sets the YTABLE pointers and sets the PLOT mode flag.

Line 120 GETs your input.

Lines 130-270 check the key input and take appropriate action.

Lines 280-350 check to see if a point should be plotted, and keep it on the screen.

Line 360 prints the present values of X, Y, and BYTE.

Line 370 jumps to YADDR to get 0 of the line you're on, then calculates and prints the decimal value of the current byte your dot is in.

Line 380 prints the current value stored in the byte that you're in.

Lines 390-440 use the machine language routine to determine the bit pattern in the current byte and convert that information into B$.

Lines 450-480 print the bit pattern and check to see which bit you're presently sitting on.

Lines 490-530 get the input values for VT, VB, HR, HL, and SHAPE#.

Line 540 resets HIMEM to protect your new shape.

Line 550 creates the new shape table.

Line 560 erases the Hi-Res screen using HGR, and DRAWs the shape from the table.

SPECIAL NOTE

I'd like to make a special note at this time. You'll note that there is a POKE 251, SHAPE in line 550, and another one in line 560. Not having that second POKE in line 560 caused me real headaches for awhile. The routine worked perfectly until I tried creating shapes larger that one memory page (256 bytes), at which time I began getting a real mess of a shape.

Read this carefully so that you won't have the same type of problem with programs that have multiple-page shapes.

Let's assume that you were starting your shape at $9000 (Shape #144) and that it is 2¼ pages long. First you POKE 251, 144 in line 550 and use a driver routine (in this case it was SCAN). When you exit the routine you would have (with no effort on your part) INCremented the Shape# to 146. Which is exactly what should have happened.

Now you try to use another driver routine without resetting your Shape#. Why bother you say, since we already set our Shape#? When you run through the second routine, your Shape# starts out at 146 where you left it, and finishes the second routine at 148. The explanation is that your Shape Table was properly created in the first routine, but when you tried to draw it, you only got the last ¼ page of your shape. The rest of what you see on the screen is garbage, and finally you're DRAWING THE HEX BYTES IN THE BLOCK SHAPE DRIVER!

The point is simply this: as long as your shapes are less than one page in length, everything is cool; you won't need to rePOKE the Shape# unless you change shapes. However, if you get into larger shapes (or as in this program need to allow for the **possibility** of someone else using larger shapes), be sure that you remember this lesson, and **reset your Shape# each time you use a Driver Shape routine.**

Lines 570-620 complete the program, saving your completed shape to disk.

SPECIAL CARE FOR THE COLOR BIT

The next two commands that you'll need to become familiar with are the **C (Color) and W (White only) commands.**

If you're dealing exclusively with non-colored (black or white) bytes, you can remain under W mode, which gives you the option of using the P (Plot) or N (No-Plot) keys to change bits and plot points as you move around the board. The problem, however, is that while in W mode, any time you HPLOT a single point within a byte — regardless of whether you're turning a bit on or off —you will clear the Color bit (bit 7), if it was set.

If you enter a byte where the Color bit is set (while in W mode), you will always change the Color bit to zero. **If you are approaching a byte that has (or may have) any color in it, change to C (Color) mode first to avoid clearing the Color bit.**

You'll lose the dot on the screen that shows where you are while in C mode. Once you leave the Color byte (you're now in a byte where the Color bit is zero), you may return to W mode. **While in C mode, the P and N keys will not function so you will not be able to change the status of any bits. However, the BIT display will continue to indicate where you are within the byte.**

Note: There are two color options for white and black. HCOLOR=3 or 7 for white, and HCOLOR=0 or 4 for black. In HCOLOR=7 or HCOLOR=4, the high bit is also set.

The final command that you should become aware of is the R (Reset Color bit) command.

If you accidently clear a Color bit, or if you want to rearrange the bit patterns in a byte (perhaps to add color, or correct some color conflicts), first use the W mode, and the P and N keys to set bits 0-6 the way you want them. Then use the R key to reset the Color bit.

Using the R key will also return you to C mode, so that you can safely leave the byte without again clearing the Color bit.

SUMMARY

Using all the options and commands in Block Shape Maker gives you the flexibility to totally control what's happening on the Hi-Res screen. By cancelling the screen erase in line 560, you can also use the routine to create and save entire Hi-Res pictures.

I hope you'll find this a valuable addition to your library of Hi-Res utilities.

```
]LIST

10  REM     *********************
11  REM     * BLOCK SHAPE MAKER  *
12  REM     * BY ROBERT DEVINE   *
13  REM     * COPYRIGHT (C) 1983 *
14  REM     * BY MICROSPARC, INC *
15  REM     * LINCOLN, MA. 01773 *
16  REM     *********************
20  PRINT  CHR$ (4)"BLOAD BLOCK ROUTINES $925E": HIMEM:
     37470: REM   THIS LOADS ROUTINES THROUGH PART 2 A
     ND ONLY SETS HIMEM TO PROTECT THOSE
30  FOR X = 768 TO 791: READ Y: POKE X,Y: NEXT :X = 0:
     Y = 0: REM     POKE BIT RETRIEVER IN MEMORY
40  DATA   164,250,177,38,133,251,162,0,134,252,70,251
     ,38,252,96,164,250,177,38,9,128,145,38,96
50  TEXT : HOME : PRINT "IF PICTURE IS NOT IN MEMORY,
     LOAD IT": PRINT "BEFORE RUNNING THIS PROGRAM.": PRINT
     : PRINT
60  PRINT "IS PICTURE IN MEMORY Y/N ?";: GET A$: PRINT
     : IF A$ = "N" THEN  PRINT : PRINT "PLEASE LOAD PI
     CTURE NOW.": END
70  PRINT : PRINT "WHAT PAGE IS YOUR PICTURE ON (1/2)
     ?";: GET A: PRINT : PRINT : PRINT : IF A < 1 OR A
      > 2 THEN 70
80  ON (A <  > 2) GOTO 100: PRINT "PLEASE ENTER THE FO
     LLOWING COMMANDS SO": PRINT "THAT WE CAN WORK ON
     PAGE 1": PRINT "CALL-151:2000<4000.5FFFM:3D0G:GOT
     040": PRINT : PRINT
90  PRINT "ENTER <RETURN> AFTER EACH COMMAND": END
100  POKE 49232,0: POKE 49239,0: POKE 49235,0:F = 1: REM
      DISPLAY PICTURE/SET SPEED=SLOW
110  CALL 37799:P = 3: REM  SETUP YTABLE POINTERS/SET
     PLOT MODE
115  POKE 230,32:C = 0: GOTO 340
120  VTAB 15: GET A$: PRINT
130  IF A$ = "U" THEN Y = Y - F: GOTO 280: REM  MOVE U
     P
140  IF A$ = "D" THEN Y = Y + F: GOTO 280: REM  MOVE D
     OWN
150  IF A$ =  CHR$ (8) THEN X = X - F: GOTO 280: REM
     MOVE LEFT
160  IF A$ =  CHR$ (21) THEN X = X + F: GOTO 280: REM
      MOVE RIGHT
170  IF A$ = "F" THEN F = 5: GOTO 120: REM   CHANGE TO
      FAST
180  IF A$ = "S" THEN F = 1: GOTO 120: REM   CHANGE TO
      SLOW
190  IF A$ =  CHR$ (32) THEN  POKE 49234,0: GOTO 120: REM
       SPACE BAR/FULL SCREEN GRAPHICS
200  IF A$ =  CHR$ (27) THEN  POKE 49235,0: GOTO 120: REM
      ESCAPE/GET TEXT WINDOW BACK
210  IF A$ = "R" THEN C = 1: CALL 783: GOTO 370: REM
     CANCEL HPLOT/SET COLOR BIT
220  IF A$ =  CHR$ (19) THEN  TEXT : GOTO 490: REM  RE
     ADY TO SAVE SHAPE
230  IF A$ = "C" THEN C = 1: GOTO 120: REM  CANCEL HPL
     OT
240  IF A$ = "W" THEN C = 0: GOTO 120: REM  RESET HPLO
     T
250  IF A$ = "P" THEN P = 3: GOTO 340: REM  ENTER PLOT
      MODE
260  IF A$ = "N" THEN P = 0: GOTO 340: REM  ENTER NOPL
     OT MODE
270  GOTO 120: REM  NO LEGAL COMMAND FOUND
280  ON (C = 1) GOTO 360: HCOLOR= 0: ON (P = 0) GOTO 2
     90: HCOLOR= 3
290  HPLOT XC,YC
300  IF X < 0 THEN X = 0
310  IF X > 279 THEN X = 279
320  IF Y < 0 THEN Y = 0
330  IF Y > 191 THEN Y = 191
340  ON (C = 1) GOTO 360:XC = X:YC = Y: HCOLOR= P: ON
     (P = 0) GOTO 350: HCOLOR= 3
350  HPLOT XC,YC
360  VTAB 21: PRINT "X="X"  ";: HTAB 8: PRINT "Y="Y"
     ";: HTAB 16: PRINT "BYTE=" INT (X / 7)"  ";: HTAB
     25: PRINT "ADDRESS=";
370  POKE 6,Y: CALL 37777:B =  PEEK (38) +  PEEK (39) *
     256 +  INT (X / 7): PRINT B"   ": REM  PRINT BYTE
      DECIMAL ADDRESS
380  VTAB 23: PRINT "BYTE VALUE=" PEEK (B)"  ";: HTAB
     16: PRINT "BITS=";: REM  PRINT VALUE OF BYTE
390  POKE 250, INT (X / 7): REM  SET BYTE 0 OFFSET
400 B$ = "": CALL 768: IF  PEEK (252) = 1 THEN B$ = "1
     ": GOTO 420
410 B$ = "0"
420  FOR M = 1 TO 7: CALL 774: IF  PEEK (252) = 1 THEN
     B$ = B$ + "1": GOTO 440
430 B$ = B$ + "0"
440  NEXT : FOR M = 1 TO 8
450  IF M = X - (7 * ( INT (X / 7))) + 1 THEN  FLASH :
      REM  FLASH FOR BIT WE'RE ON
460  IF M = 8 THEN  INVERSE : REM  INVERSE FOR COLOR B
     IT
470  PRINT  MID$ (B$,M,1);: NORMAL : PRINT " ";
480  NEXT : PRINT "  ": GOTO 120
490  HOME : INPUT "ENTER VT ";VT: PRINT
500  INPUT "ENTER VB ";VB: PRINT
510  INPUT "ENTER HR ";HR: PRINT
520  INPUT "ENTER HL ";HL: PRINT : PRINT : PRINT
530  INPUT "WHAT IS THE SHAPE# ";SHAPE: PRINT
540  HIMEM: SHAPE * 256: REM  RESET HIMEM TO PROTECT S
     HAPE
550  POKE 251,SHAPE: POKE 252,VT: POKE 253,VB: POKE 25
     4,HR: POKE 255,HL: CALL 37729: REM   CREATE SHAPE
      TABLE
560  HGR : POKE 251,SHAPE: CALL 37679: REM  DRAW IT ON
      THE SCREEN
570  VTAB 22: PRINT "HERE IS THE SHAPE IN YOUR SHAPE T
     ABLE.": PRINT "DO YOU WANT TO SAVE IT (Y/N) ?";: GET
     A$
580  IF A$ = "N" THEN  END
590  HOME : VTAB 22: INPUT "WHAT IS THE NAME ";A$
600  HOME : VTAB 22: PRINT "SAVING SHAPE TO DISK"
610  PRINT  CHR$ (4)"BSAVE "A$",A"SHAPE * 256",L"(VB -
     VT + 1) * (HR - HL + 1)
620  END
```

KEY PERFECT 4.0
RUN ON
BLOCK SHAPE MAKER

CODE	LINE# - LINE#
CB30	10 - 40
FC00	50 - 130
BFD2	140 - 230
6449	240 - 330
B6BB	340 - 430
615F	440 - 530
A404	540 - 620

TOTAL PROGRAM CHECK IS : 09FB

APPLE CHECKER

ON: BLOCK SHAPE MAKER
TYPE: A

LENGTH: 091B
CHECKSUM: F2

Block Shape Animation — III (Cont.)

BLOCK SHAPE ROUTINES $925E

```
*925E.95FF

925E- 38 A5
9260- FC E5 E3 85 FC 38 A5 FD
9268- E5 E3 85 FD 60 18 A5 FC
9270- 65 E3 85 FC 18 A5 FD 65
9278- E3 85 FD 60 A9 00 8D 54
9280- C0 A9 40 85 E6 A5 FC C5
9288- E3 90 0F 20 6D 92 20 2F
9290- 93 20 5E 92 20 5E 92 20
9298- 2F 93 60 A9 00 8D 55 C0
92A0- A9 20 85 E6 20 6D 92 20
92A8- 2F 93 20 5E 92 20 5E 92
92B0- 20 2F 93 60 A9 00 8D 54
92B8- C0 A9 40 85 E6 20 5E 92
92C0- 20 2F 93 20 6D 92 20 6D
92C8- 92 20 2F 93 60 A9 00 8D
92D0- 55 C0 A9 20 85 E6 20 5E
92D8- 92 20 2F 93 20 6D 92 20
92E0- 6D 92 20 2F 93 60 A9 00
92E8- 85 FA A5 FD 85 06 20 91
92F0- 93 A4 FF A2 00 A1 FA C9
92F8- 7F F0 15 C9 01 90 11 86
9300- F9 4A 26 F9 E8 E0 07 90
9308- F8 4A A5 F9 90 02 09 80
9310- 91 26 C8 E6 FA D0 02 E6
9318- FB C4 FE 90 D6 F0 D4 C6
9320- 06 A5 06 C9 FF F0 04 C5
9328- FC B0 C3 20 61 93 60 A9
9330- 00 85 FA A5 FD 85 06 20
9338- 91 93 A4 FE A2 00 A1 FA
9340- 51 26 91 26 88 18 E6 FA
9348- D0 02 E6 FB C0 FF F0 04
9350- C4 FF B0 EA C6 06 A5 06
9358- C9 FF F0 04 C5 FC B0 D7
9360- 60 A9 00 85 FA A5 FD 85
9368- 06 20 91 93 A4 FE A2 00
9370- B1 26 81 FA 88 18 E6 FA
9378- D0 02 E6 FB C0 FF F0 04
9380- C4 FF B0 EC C6 06 A5 06
9388- C9 FF F0 04 C5 FC B0 D9
9390- 60 A4 06 B1 CE 85 26 A5
9398- E6 C9 40 D0 05 B1 DE 85
93A0- 27 60 B1 EE 85 27 60 A9
93A8- 80 85 CE A9 94 85 CF A9
93B0- 40 85 EE A9 95 85 EF A9
93B8- C0 85 DE A9 93 85 DF 60
93C0- 40 44 48 4C 50 54 58 5C
93C8- 40 44 48 4C 50 54 58 5C
93D0- 41 45 49 4D 51 55 59 5D
93D8- 41 45 49 4D 51 55 59 5D
93E0- 42 46 4A 4E 52 56 5A 5E
93E8- 42 46 4A 4E 52 56 5A 5E
93F0- 43 47 4B 4F 53 57 5B 5F
93F8- 43 47 4B 4F 53 57 5B 5F
9400- 40 44 48 4C 50 54 58 5C
9408- 40 44 48 4C 50 54 58 5C
9410- 41 45 49 4D 51 55 59 5D
9418- 41 45 49 4D 51 55 59 5D
9420- 42 46 4A 4E 52 56 5A 5E
9428- 42 46 4A 4E 52 56 5A 5E
9430- 43 47 4B 4F 53 57 5B 5F
9438- 43 47 4B 4F 53 57 5B 5F
9440- 40 44 48 4C 50 54 58 5C
9448- 40 44 48 4C 50 54 58 5C
9450- 41 45 49 4D 51 55 59 5D
9458- 41 45 49 4D 51 55 59 5D
9460- 42 46 4A 4E 52 56 5A 5E
9468- 42 46 4A 4E 52 56 5A 5E
9470- 43 47 4B 4F 53 57 5B 5F
9478- 43 47 4B 4F 53 57 5B 5F
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
```

KEY PERFECT 4.0
RUN ON
BLOCK ROUTINES $925E

CODE	ADDR# - ADDR#
28B4	925E - 92AD
2E74	92AE - 92FD
271E	92FE - 934D
2B08	934E - 939D
2E93	939E - 93ED
2D75	93EE - 943D
29A1	943E - 948D
234F	948E - 94DD
2837	94DE - 952D
2E55	952E - 957D
2728	957E - 95CD
1AD8	95CE - 95FF

TOTAL PROGRAM CHECK IS : 03A2

APPLE CHECKER

ON: BLOCK ROUTINES $925E
TYPE: B

LENGTH: 03A2
CHECKSUM: EE