BLOCK SHAPE ANIMATION IX

GRAPHICS WORKSHOP

DOS 3.3 O 0

Byte-sized block shapes move faster and take less code than larger block shapes. This month's Graphics Workshop returns from the exploration of Double Hi-Res Graphics to normal Hi-Res block shape animation with some routines that will move single-byte blocks at lightning speed.

ProDOS O 0

by Robert R. Devine, Computers and You, 1855 N. West Ave., El Dorado, AR 71730

In earlier installments of the Graphics Workshop series we developed a variety of machine code routines for use in Apple graphics animation. The routines are general enough to be used with block shapes of any size, from a single byte to shapes large enough to fill the entire Hi-Res screen. A recent program that I was working on required that almost all the shapes be no larger than seven dots (one byte) wide. I've also noticed that in many of the commercial programs now flooding the market, the authors used shapes only one byte wide to achieve maximum execution speed.

To meet my program needs I developed a special set of block shape routines designed specifically for shapes one byte wide. The routines that I modified for one-byte wide shapes were DRAW, SHIFTDN, SHIFTUP, SHIFTR and SHIFTL.

So, you say, if our existing routines already do the job, why bother with new ones? There are three major benefits. First, you'll need one less POKE to set the parameters for the SHIFTing routines. Second, since they don't waste time checking for HR and HL, the routines execute much faster (our ultimate goal!). Finally, we'll find that the new one-byte wide routines are much shorter. **Table 1** shows that the one-byte routines save 127 bytes compared to the old routines, and there is an obvious improvement in execution time when working with small shapes.

These new routines are designed to fit below our other driver routines, ending just below DRAWDN. The routines are, however, totally relocatable, so if you wanted to you could load them into memory at $92B5 along with YTABLE and YADDR, to create a completely new driver for shapes one byte wide. First, you will need to determine the new CALL addresses for each routine.

THE ONE-BYTE ROUTINES

DRAW $8F9A (CALL 36762) draws a one-byte wide block shape. There are a few differences between the use of this DRAW and our multi-byte DRAW. Since your shape tables will be smaller, you will probably want to pack several shapes on a given memory page. Therefore, this DRAW requires that you specify *both* the high byte (POKE 251, SHPHI) and the low byte (POKE 250, SHPLO) of the address where the shape table is stored in memory.

Next, you'll need to set VT (POKE 252, VT) and VB (POKE 253,VB) in the normal fashion. Since HR and HL will now be the same, we'll simply rename this value HB (POKE 254,HB). This DRAW does not have an EOR (erase function), so you can simply erase with 00 bytes.

The final difference is that this DRAW doesn't allow shape tables to run over onto the next memory page. Since the shapes are smaller (a maximum of 192 bytes) this should be easy to deal with, plus it avoids wasting time checking for page overflow. The following parameters are required:

POKE 250,SHPLO
POKE 251,SHPHI (formerly SHNUM)
POKE 252,VT
POKE 253,VB
POKE 254,HB (the HR/HL value 0-39)

SHFTDN $8FB8 (CALL 36792) shifts the shape down one Y-coordinate. This one-byte routine works exactly the same as our multi-byte SHFTDN, except that here you need only three routine parameters:

POKE 252,VT
POKE 253,VB
POKE 254,HB

SHIFTUP $8FE2 (CALL 36834) shifts the shape up one Y-coordinate. This routine works the same as SHIFTDN, and requires the same three parameters.

SHIFTR $9010 (CALL 36880) shifts the shape right one dot. Unlike our multi-byte horizontal shifting routines, which required that an extra byte be added ahead of the shape, this routine takes care of the problem automatically. To use SHIFTR, you just set the three parameters for VT, VB, and HB.

To use the routine, CALL or JSR SHIFTR seven times, then use the INCHB routine (CALL 36966), which will INCrement HB in readiness for the next seven shifts rightward. If you shift rightward more than seven times without INCrementing HB, your shape will begin to disappear from the screen.

SHIFTL $9038 (CALL 36920) shifts the shape left one dot. This routine also takes care of providing a shifting byte, and the only

TABLE 1
Comparison of Routine Lengths

Routine Name	Length of Old Routine	Length of One-byte Routine
DRAW	50 bytes	30 bytes
SHIFTR	80	40
SHIFTL	89	46
SHIFTD	54	42
SHIFTU	58	46
Total	331 bytes	204 bytes

parameters that need to be set are VT, VB and HB. To use SHIFTL, CALL or JSR SHIFTL seven times, then use the DECHB routine (CALL 36975) to DECrement HB for the next seven shifts. Again, more than seven shifts without changing HB will result in the loss of your shape.

INCHB $9066 (CALL 36966) INCrements HB for horizontal shifting. It has built-in protection preventing it from allowing HB to become greater than 39. DECHB $906F (CALL 36975) DECrements HB and prevents it from going to a value less then 0.

In the event that your horizontal moving shapes are likely to change direction, you should be careful not to make a direction change while the shape resides in more than one horizontal byte. This simply means that direction changes should only occur at the end of seven shifts right or left.

Another thing we've done to speed execution time is eliminate the effort to control the conditioning of bit 7 (the color bit). Since horizontal one-dot shifting doesn't work well with color shapes anyway, we've allowed the chips to fall where they may as far as bit 7 is concerned, which won't make a difference with black and white shapes.

Another benefit of our new one-byte wide shape routines is that these routines do not make use of as many zero page bytes for flags and holders. If you're calling the new routines from a machine language program, the zero page addresses $F9, $07, $08 and $09 (which were used by the multi-byte shape shift routines) will now be available for use. We now simply use the Carry to hold the horizontal bit shifting flags, and the X-Register to hold the screen byte during vertical shifting.

ENTERING THE ROUTINES

These new routines were created with the S-C Assembler, as were the other driver routines. You can either enter them with an assembler or simply type in the hex code from **Listing 1**. It is a good idea to save the new routines as a separate disk file rather than adding them to our old block shape driver. This way, you can load or leave them as needed, or simply BLOAD them to some other location to conform to your program's memory configuration. Save them to disk with the statement:

```
BSAVE BYTE.ROUTINES A$8F9A,
L$DC
```

To use the new routines you will need the block routines developed in previous installments of the Graphics Workshop. For those who may have missed an issue, the hex code of these routines is shown in **Listing 2**. Use the Monitor to enter this code and save it on disk with the command:

```
BSAVE BLOCK.ROUTINES,A$9076,
L$58A
```

For help in entering *Nibble* listings, see "A Welcome to New *Nibble* Readers" in the beginning of this issue.

TESTING THE NEW ROUTINES

To try out the one-byte routines, enter and RUN the Applesoft program shown in **Listing 3**. (Save it to disk with the command: SAVE BYTE.DEMO)

HOW THE PROGRAM WORKS

Lines 80-100 load the driver routines into memory, initialize YTABLE and set HIMEM to protect the driver. **Lines 110-120** POKE the shape of a small ball into memory at $8F00. **Line 130** sets up full screen graphics.

Line 140 establishes the starting parameters for DRAW. **Line 150** DRAWs the shape. **Lines 160-190** move the shape up and down on the screen using SHIFTDN and SHIFTUP.

Line 200 sets the starting parameters for DRAW and draws the shape on the left edge of the screen. You'll note that we didn't enter the POKE 251,SHPHI parameter since it hadn't changed since our last DRAW. However, it was necessary to POKE 250, SHPLO since the DRAW routine changes this value as it steps through the shape table.

Lines 210-240 move the shape back and forth across the screen. As we move the shape we first CALL SHIFTR or SHIFTL seven times, then CALL INCHB or DECHB to change the HB value for the next seven shifts.

INCREASING SPEED

As you can see, the shapes move quite smartly about the screen, even though we're still driving the shapes from Applesoft. If you want to see some real fireworks, let's see what happens when we let our routines run flat out in straight machine code.

The first thing you'll need to do is enter the hex code shown in **Listing 4**. This is an exact machine code translation of **lines 140-240**. Save it on disk with the command:

```
BSAVE BYTE.ML,A$6000,L$7B
```

Next, delete **lines 140-260** from BYTE .DEMO and add this new **line 140**:

```
140 PRINT CHR$(4)"BRUN BYTE
.ML"
```

It makes quite a difference, doesn't it? Things are moving along at such a clip now that you'll think you're seeing the ball in several locations at the same time. That's the power of one-byte block shape graphics!

Editor's Note: *The Graphics Workshop series on block shape animation first appeared in Vol. 4/No. 3. "Block Shape Animation, Part VIII" (Vol. 5/No. 4) added the final routines to the block shape animation program, BLOCK.ROUTINES, shown in* ***Listing 2****.*

LISTING 1: BYTE.ROUTINES

```
                0900 * BYTE.ROUTINES
                0910 * BY ROBERT DEVINE
                0920 * COPYRIGHT 1985
                0930 * BY MICROSPARC, INC.
                0940 * CONCORD, MA 01742
                0950 *
                0960 *
                0970 * S-C ASSEMBLER
                0980 *
                0990 *
                1000  .OR $8F9A
                1010  .TA $800
00FA-           1090 SHPLO .EQ $FA      ** DECIMAL 250
00FB-           1100 SHPHI .EQ $FB      ** DECIMAL 251
00FC-           1110 VT .EQ $FC         ** DECIMAL 252
00FD-           1120 VB .EQ $FD         ** DECIMAL 253
00FE-           1130 HB .EQ $FE         ** DECIMAL 254
9391-           1150 YADDR .EQ $9391
0026-           1160 HBASL .EQ $26
F504-           1170 INCRY .EQ $F504
F4D5-           1180 DECRY .EQ $F4D5
0006-           1190 YO .EQ $06
8F9A- A2 00     1500 DRAW LDX #0        ** CALL 36762
8F9C- A5 FD     1510  LDA VB            ** GET LOWEST Y-COORDINATE
8F9E- 85 06     1520  STA YO            ** STORE IN $6 FOR USE BY YADDR
8FA0- 20 91 93  1530 L1 JSR YADDR       ** PUT SCREEN ADDRESS IN HBASL/HBASH
8FA3- A4 FE     1540  LDY HB            ** SET ADDRESS OFFSET
8FA5- A1 FA     1550  LDA (SHPLO,X)     ** GET SHAPE BYTE FROM TABLE
8FA7- 91 26     1560  STA (HBASL),Y     ** PUT SHAPE BYTE ON SCREEN
8FA9- E6 FA     1570  INC SHPLO         ** POINT TO NEXT TABLE ELEMENT
8FAB- C6 06     1580  DEC YO            ** POINT TO NEXT HIGHER SCREEN BYTE
8FAD- A5 06     1590  LDA YO            ** FIND OUT WHERE WE ARE
8FAF- C9 FF     1600  CMP #$FF          ** OFF TOP OF SCREEN?
8FB1- F0 04     1610  BEQ RTN           ** YES-GET OUT OF HERE
8FB3- C5 FC     1620  CMP VT            ** HAVE WE PASSED VT?
8FB5- B0 E9     1630  BCS L1            ** NO-PROCESS NEXT TABLE ELEMENT
8FB7- 60        1640 RTN RTS            ** WE'RE ALL DONE
8FB8- A5 FD     1700 SHFTDN LDA VB      ** CALL 36792
8FBA- C9 BD     1710  CMP #189         ** ARE WE TOO FAR DOWN?
8FBC- B0 23     1720  BCS RTN1          ** YES-GET OUT OF HERE
8FBE- 85 06     1730  STA YO            ** STORE IN $6 FOR USE BY YADDR
8FC0- 20 91 93  1740 L2 JSR YADDR       ** PUT SCREEN ADDRESS IN HBASL/HBASH
8FC3- A4 FE     1750  LDY HB            ** SET ADDRESS OFFSET
8FC5- B1 26     1760  LDA (HBASL),Y     ** GET BYTE FROM SCREEN
8FC7- AA        1770  TAX               ** STORE IT IN X-REGISTER
8FC8- 20 04 F5  1780  JSR INCRY         ** POINT TO NEXT LOWER ADDRESS
8FCB- 8A        1790  TXA               ** RETRIEVE SCREEN BYTE
8FCC- 91 26     1800  STA (HBASL),Y     ** RETURN BYTE TO SCREEN
8FCE- 20 D5 F4  1810  JSR DECRY         ** RETURN TO START ADDRESS
8FD1- C6 06     1820  DEC YO            ** POINT TO NEXT HIGHER BYTE
8FD3- A5 06     1830  LDA YO            ** FIND OUT WHERE WE ARE
8FD5- C9 FF     1840  CMP #$FF          ** OFF TOP OF SCREEN?
8FD7- F0 04     1850  BEQ J1            ** YES-GET OUT OF HERE
8FD9- C5 FC     1860  CMP VT            ** HAVE WE PASSED VT?
8FDB- B0 E3     1870  BCS L2            ** NO-PROCESS NEXT SCREEN BYTE
8FDD- E6 FD     1880 J1 INC VB          ** BUMP VB AND VT FOR
8FDF- E6 FC     1890  INC VT            ** NEXT DOWNWARD MOVE
8FE1- 60        1900 RTN1 RTS           ** WE'RE ALL DONE
8FE2- A5 FC     2000 SHIFTUP LDA VT     ** CALL 36834
8FE4- C9 01     2010  CMP #1            ** WILL WE GO OFF TOP OF SCREEN?
8FE6- 90 27     2020  BCC RTN2          ** YES-GET OUT OF HERE
8FE8- 85 06     2030  STA YO            ** STORE IN $6 FOR USE BY YADDR
8FEA- E6 FD     2040  INC VB            ** GIVE US AN ERASE CUSHION
8FEC- 20 91 93  2050 L3 JSR YADDR       ** PUT SCREEN ADDRESS IN HBASL/HBASH
8FEF- A4 FE     2060  LDY HB            ** SET ADDRESS OFFSET
8FF1- B1 26     2070  LDA (HBASL),Y     ** GET BYTE FROM SCREEN
8FF3- AA        2080  TAX               ** STORE IT IN X-REGISTER
8FF4- 20 D5 F4  2090  JSR DECRY         ** POINT TO NEXT HIGHER ADDRESS
8FF7- 8A        2100  TXA               ** RETRIEVE SCREEN BYTE
8FF8- 91 26     2110  STA (HBASL),Y     ** RETURN BYTE TO SCREEN
8FFA- 20 04 F5  2120  JSR INCRY         ** RETURN TO START ADDRESS
8FFD- E6 06     2130  INC YO            ** POINT TO NEXT LOWER ADDRESS
8FFF- A5 06     2140  LDA YO            ** FIND OUT WHERE WE ARE
9001- C9 BE     2150  CMP #190          ** ARE WE DOWN TOO FAR?
9003- F0 04     2160  BEQ J2            ** YES-GET OUT OF HERE
9005- C5 FD     2170  CMP VB            ** HAVE WE PASSED VB?
9007- 90 E3     2180  BCC L3            ** NO-PROCESS THE NEXT BYTE
9009- C6 FD     2190 J2 DEC VB          ** BUMP VT AND VB FOR
900B- C6 FC     2200  DEC VT            ** REMOVE ERASE CUSHION
900D- C6 FC     2210  DEC VT            ** NEXT UPWARD MOVE
900F- 60        2220 RTN2 RTS           ** WE'RE ALL DONE
9010- A5 FD     2300 SHIFTR LDA VB      ** CALL 36880
9012- 85 06     2310  STA YO            ** STORE IN $6 FOR USE BY YADDR
```

LISTING 1: BYTE.ROUTINES *(continued)*

```
9014- 20 91 93  2320 L4 JSR YADDR       ** PUT SCREEN ADDRESS IN HBASL/HBASH
9017- A4 FE     2330  LDY HB            ** GET SCREEN OFFSET
9019- C0 27     2340  CPY #39           ** CAN WE STILL MOVE RIGHT?
901B- B0 1A     2350  BCS RTN3          ** NO-GET OUT OF HERE
901D- B1 26     2360  LDA (HBASL),Y     ** GET SCREEN BYTE
901F- 18        2370  CLC               ** SET TO ERASE LEFTMOST DOT
9020- 2A        2380  ROL               ** SHIFT THE BITS RIGHT
9021- 91 26     2390  STA (HBASL),Y     ** RETURN BYTE TO SCREEN
9023- 2A        2400  ROL               ** GET THE PRE-SHIFTED BIT 6
9024- C8        2410  INY               ** MOVE TO NEXT BYTE -->
9025- B1 26     2420  LDA (HBASL),Y     ** GET BYTE FROM SCREEN
9027- 2A        2430  ROL               ** SHIFT BITS RIGHT/MOVE UP OLD BIT 6
9028- 91 26     2440  STA (HBASL),Y     ** RETURN BYTE TO SCREEN
902A- 88        2450  DEY               ** MOVE BACK TO HB
902B- C6 06     2460  DEC YO            ** MOVE TO NEXT HIGHER ADDRESS
902D- A5 06     2470  LDA YO            ** FIND OUT WHERE WE ARE
902F- C9 FF     2480  CMP #$FF          ** OFF TOP OF SCREEN?
9031- F0 04     2490  BEQ RTN3          ** YES-GET OUT OF HERE
9033- C5 FC     2500  CMP VT            ** HAVE WE PASSED VT?
9035- B0 DD     2510  BCS L4            ** NO-PROCESS THE NEXT BYTE
9037- 60        2520 RTN3 RTS           ** WE'RE ALL DONE
9038- A5 FD     2600 SHIFTL LDA VB      ** CALL 36920
903A- 85 06     2610  STA YO            ** STORE IN $6 FOR USE BY YADDR
903C- 20 91 93  2620 L5 JSR YADDR       ** PUT SCREEN ADDRESS IN HBASL/HBASH
903F- A4 FE     2630  LDY HB            ** SET ADDRESS OFFSET
9041- F0 22     2640  BEQ RTN4          ** IF WE CAN'T MOVE LEFT-GET OUT OF HERE
9043- B1 26     2650  LDA (HBASL),Y     ** GET SCREEN BYTE
9045- 29 7F     2660  AND #$7F          ** SET TO ERASE TRAILING DOT
9047- 6A        2670  ROR               ** SHIFT THE BITS LEFT
9048- 91 26     2680  STA (HBASL),Y     ** RETURN BYTE TO SCREEN
904A- 88        2690  DEY               ** MOVE TO NEXT BYTE <--
904B- B1 26     2700  LDA (HBASL),Y     ** GET SCREEN BYTE
904D- 90 04     2710  BCC J3            ** PRE-SHIFTED BIT 0 WAS 0
904F- 09 80     2720  ORA #$80          ** PRE-SHIFTED BIT 0 WAS 1/SET BIT 7
9051- B0 02     2730  BCS J4            ** (RELOCATABLE JMP)
9053- 29 7F     2740 J3 AND #$7F        ** CLEAR BIT 7
9055- 6A        2750 J4 ROR             ** SHIFT THE BITS LEFT
9056- 91 26     2760  STA (HBASL),Y     ** RETURN BYTE TO SCREEN
9058- C8        2770  INY               ** RETURN TO HB
9059- C6 06     2780  DEC YO            ** POINT TO NEXT HIGHER ADDRESS
905B- A5 06     2790  LDA YO            ** FIND OUT WHERE WE ARE
905D- C9 FF     2800  CMP #$FF          ** OFF TOP OF SCREEN?
905F- F0 04     2810  BEQ RTN4          ** YES-GET OUT OF HERE
9061- C5 FC     2820  CMP VT            ** HAVE WE PASSED VT?
9063- B0 D7     2830  BCS L5            ** NO-PROCESS THE NEXT BYTE
9065- 60        2840 RTN4 RTS           ** WE'RE ALL DONE
9066- A5 FE     2850 INCHB LDA HB       ** CALL 36966
9068- C9 27     2860  CMP #39           ** IS IT ALREADY 39?
906A- B0 02     2870  BCS RTN5          ** YES-GET OUT OF HERE
906C- E6 FE     2880  INC HB            ** INCREMENT HB
906E- 60        2890 RTN5 RTS           ** WE'RE ALL DONE
906F- A5 FE     2900 DECHB LDA HB       ** CALL 36975
9071- F0 02     2910  BEQ RTN6          ** IF ALREADY 0-GET OUT OF HERE
9073- C6 FE     2920  DEC HB            ** DECREMENT HB
9075- 60        2930 RTN6 RTS           ** WE'RE ALL DONE
```

END OF LISTING 1

```
KEY PERFECT 4.0
RUN ON
BYTE.ROUTINES
```

CODE	ADDR# - ADDR#
2585	8F9A - 8FE9
28A9	8FEA - 9039
1E13	903A - 9075

PROGRAM CHECK IS : DC

LISTING 2: BLOCK.ROUTINES

```
9076- A9 00
9078- 85 FA A5 FC 85 06 20 91
9080- 93 A4 FE A2 00 A1 FA 51
9088- 26 91 26 88 18 E6 FA D0
9090- 02 E6 FB C0 FF F0 04 C4
9098- FF B0 EA E6 06 A5 06 C9
90A0- FF F0 06 C5 FD 90 D7 F0
90A8- D5 60 A5 FD C9 BD B0 2F
90B0- 85 06 20 91 93 A4 FE B1
90B8- 26 85 F9 20 04 F5 A5 F9
90C0- 91 26 20 D5 F4 88 18 C0
90C8- FF F0 04 C4 FF B0 E8 C6
90D0- 06 A5 06 C9 FF F0 04 C5
90D8- FC B0 D7 E6 FC E6 FD 60
90E0- A5 FC C9 01 90 33 E6 FD
90E8- 85 06 20 91 93 A4 FE B1
90F0- 26 85 F9 20 D5 F4 A5 F9
90F8- 91 26 20 04 F5 88 18 C0
9100- FF F0 04 C4 FF B0 E8 E6
9108- 06 A5 06 C9 BE F0 04 C5
9110- FD 90 D7 C6 FD C6 FD C6
9118- FC 60 A9 0E A6 FB 9D 6F
9120- 8F 60 A9 00 8D 54 C0 A9
9128- 40 85 E6 60 A9 00 8D 55
9130- C0 A9 20 85 E6 60 20 22
9138- 91 18 90 03 20 2C 91 20
```

```
9140- 0E 92 20 0E 92 20 7A 91
9148- 20 0E 92 20 0E 92 20 7A
9150- 91 A5 E6 C9 40 F0 E5 60
9158- 20 22 91 18 90 03 20 2C
9160- 91 20 B5 91 20 B5 91 20
9168- 8A 91 20 B5 91 20 B5 91
9170- 20 8A 91 A5 E6 C9 40 F0
9178- E5 60 A6 FB DE 6F 8F D0
9180- 08 20 AC 91 A9 0E 9D 6F
9188- 8F 60 A6 FB DE 6F 8F D0
9190- F8 20 97 91 18 90 ED A5
9198- FF C9 02 90 0E C6 FF C6
91A0- FE A5 FF C9 01 90 04 C6
91A8- FF C6 FE 60 E6 FF E6 FE
91B0- E6 FF E6 FE 60 A5 FD 85
91B8- 06 20 91 93 18 A4 FE A9
91C0- 00 85 08 85 09 85 07 90
91C8- 02 E6 08 B1 26 C9 80 90
91D0- 02 E6 09 A5 08 F0 07 B1
91D8- 26 09 80 4C E2 91 B1 26
91E0- 29 7F 6A 91 26 90 02 E6
91E8- 07 A5 09 C9 01 90 06 B1
91F0- 26 09 80 91 26 C4 FF F0
91F8- 08 88 A5 07 C9 01 4C BF
9200- 91 C6 06 A5 06 C9 FF F0
9208- 04 C5 FC B0 AC 60 A5 FD
9210- 85 06 20 91 93 18 A4 FF
9218- A9 00 85 08 85 09 B1 26
9220- 2A 91 26 B0 02 90 02 E6
9228- 08 C9 80 B0 02 90 02 E6
9230- 09 A5 08 D0 09 B1 26 29
9238- 7F 91 26 4C 44 92 B1 26
9240- 09 80 91 26 C4 FE F0 09
9248- C8 18 A5 09 C9 01 4C 18
9250- 92 C6 06 A5 06 C9 FF F0
9258- 04 C5 FC B0 B5 60 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 20
```

END OF LISTING 2

```
KEY PERFECT 4.0
RUN ON
BLOCK.ROUTINES
==========================
  CODE       ADDR# - ADDR#
 ------     -------------
  264E       9076 - 90C5
  2931       90C6 - 9115
  286D       9116 - 9165
  2A6E       9166 - 91B5
  28F9       91B6 - 9205
  222F       9206 - 9255
  238D       9256 - 92A5
  2592       92A6 - 92F5
  24D1       92F6 - 9345
  2EC8       9346 - 9395
  2B27       9396 - 93E5
  3226       93E6 - 9435
  2748       9436 - 9485
  2418       9486 - 94D5
  2DFF       94D6 - 9525
  2DE8       9526 - 9575
  2937       9576 - 95C5
  1C7C       95C6 - 95FF
PROGRAM CHECK IS : 058A
```

LISTING 3: BYTE.DEMO

```
10  REM  ************************
20  REM  *      BYTE.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 BYTE.ROUTINES"
90  PRINT  CHR$ (4)"BLOAD BLOCK.ROUTINES"
100  CALL 37799: HIMEM: 35584
110  FOR X = 36608 TO 36616: READ A: POKE X,A
     : NEXT : REM  ENTER SHAPE
120  DATA  0,28,62,127,127,127,62,28,0
130  HGR :X =  PEEK (49234)
140  POKE 251,143: POKE 250,0: POKE 252,0: POKE
     253,8: POKE 254,20
150  CALL 36762
160  FOR Y = 1 TO 2
170  FOR X = 1 TO 190: CALL 36792: NEXT
180  FOR X = 1 TO 190: CALL 36834: NEXT
190  NEXT Y
200  POKE 250,0: POKE 252,100: POKE 253,108: POKE
     254,0: CALL 36762
210  FOR Y = 1 TO 2
220  FOR X = 0 TO 39: FOR SHFT = 1 TO 7: CALL
     36880: NEXT SHFT: CALL 36966: NEXT X
230  FOR X = 0 TO 39: FOR SHFT = 1 TO 7: CALL
     36920: NEXT SHFT: CALL 36975: NEXT X
240  NEXT Y
250  IF  PEEK ( - 16384) > 128 THEN  TEXT : HOME
     : END
260  GOTO 140
```

END OF LISTING 3

LISTING 4: BYTE.ML

```
6000- A9 8F 85 FB A9 00 85 FA
6008- 85 FC A9 08 85 FD A9 14
6010- 85 FE 20 9A 8F A9 02 85
6018- 09 A9 BE 85 08 20 B8 8F
6020- C6 08 D0 F9 A9 BE 85 08
6028- 20 E2 8F C6 08 D0 F9 C6
6030- 09 D0 E6 A9 00 85 FA 85
6038- FE A9 64 85 FC A9 6C 85
6040- FD 20 9A 8F A9 02 85 09
6048- A9 27 85 08 A9 07 85 07
6050- 20 10 90 C6 07 D0 F9 20
6058- 66 90 C6 08 D0 EE A9 27
6060- 85 08 A9 07 85 07 20 38
6068- 90 C6 07 D0 F9 20 6F 90
6070- C6 08 D0 EE C6 09 D0 D0
6078- 4C 00 60
```

END OF LISTING 4