Graphics Workshop: Part VIII

New Letter Scroll

ProDOS

DOS 3.3

Using the techniques presented in earlier columns, and two short new programs, you can easily scroll text across the Hi-Res screen. Each character makes a smooth entrance by using the "holes" in Hi-Res screen memory.

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

Since I began this series I've gotten many nice letters and calls from readers regarding the topic of Hi-Res graphics. Many of these were inquiries for help with special problems, or requests that I cover some particular graphics routines. Thank you all for your support; it has helped make this a worthwhile project, and along the way I've made some new friends.

This month's discussion was inspired by a problem from a reader in Brooklyn, New York which just happens to fit very nicely into a topic that I wanted to cover here. What he needed was a method for causing TEXT or some other message to scroll across the screen, using large Hi-Res letters.

A Peculiarity

Before we get into that subject directly, let's take a look at a peculiar thing about the Hi-Res screens that isn't clearly spelled out in any of the manuals that I've ever seen. This peculiarity will also work for us in our text scrolling project.

As we go through our discovery of this peculiarity I'd appreciate it if you'd turn on your Apple and try out the tests as we go along. **No fair reading ahead to get the answers!**

First let's go back and look at the STORM WARNING program that was presented in the last issue. What I'm specifically referring to here is the point where we **drove** the truck off the right side of the screen, just before the main game was loaded and run.

The question here is...**Do you know where the truck went as we drove it off the screen?**

As you'll recall, we first drew the truck at HR and HL equal to 15 and 10 respectively. When we SHIFTed the truck off the screen we moved it right 30 bytes, INCrementing HR and HL every seven shifts to prevent the truck from disappearing into that black hole we mentioned in our discussion of shift animation. We began with the truck's HR at 15 and then shifted it all the way over to HR=45, which as you know by now can't be done because the screen is only 40 bytes wide with a maximum HR of 39...right?

"There are 512 extra bytes allocated to each Hi-Res screen that are not actually displayed!"

Here is the crux of the peculiarity that we will look at this month. To add more spice to the mystery, let me give you these clues:

1. The truck is still on the Hi-Res screen.
2. If we wanted to, we could still REVDIR the truck and drive it back onto the screen with no need to reDRAW the shape.

Okay, let's try out a few experiments to see if we can find out what's happening.

To do so, **BLOAD** the **SHAPES** file from STORM WARNING; or if you don't have that entered (shame on you) then BLOAD whatever block shape driver routines you have on disk that include the DRAW and SHIFTL routines (or type in the program from **Listing 1**). Be sure to **CALL 37799** to set up YTABLE.

You'll also need some sort of shape to DRAW (we'll use shape #97), or you can simply select shape #148 and use the hex bytes of the driver itself as though they were shape bytes.

Now let's DRAW the house shape #97 (or alternate shape #148) at VT, VB, HR, and HL equal to 10, 49, 4, and 0, respectively. On the screen? Good.

Now let's see what happens if we try to use an HR and HL outside the legal limits of 0-39. **POKE 44 and 40** into HR and HL and try to **DRAW** the shape again. Surprised? The draw routine still worked — even with illegal values for HR and HL — but instead of the shape being DRAWn way off on the right side of the screen, it appeared **directly below** the first house we drew, even though VT and VB are still set the same as for the first house. What we have here is the normal rightside overflow move to the left side that you're accustomed to.

Let's try another test by reDRAWing the house at VT, VB, HR, and HL of 150, 189, 4, and 0. Now you should have three houses on the screen, one atop the other. To view the entire third house, enter **X=PEEK(49234)** which will cause a change to full-screen graphics.

Finally, let's see what happens if we again try to DRAW the house with VT and VB at the illegal values of 44 and 40 as we did before.

This time you won't see anything after DRAWing the shape, **so where is the shape?...Did we really draw it?**

If you're sneaky, you might think that what happened is that we drew the shape on HGR2 page 2, whose addresses begin immediately after page 1 because of the right side overflow.

Sorry folks, that's not where the house is!

The Solution to the Mystery

In order to find the fourth house (and convince yourself that we really **did** DRAW it), let's try something, even though you may be a bit skeptical at this point. After all, it's hard to hide a 200-byte block shape on the screen...**Or is it?**

From the keyboard, enter the following commands:

```
POKE 255,10
FOR X=1 TO 280: CALL (SHIFTL): NEXT
```

The proper CALL for SHIFTL will depend on whether you're using the block routines from STORM WARNING or from our regular driver. Use **CALL 37437** (STORM WARNING) or **CALL 37301** (Workshop Series or BLOCK.ROUTINES).

If you had DRAWn the shape at HR and HL equal to 44 and 40 as we suggested, you should now see your phantom shape slowly shifting leftward across the screen, and finally disappearing into the black hole when it reaches HL=10 and has nowhere to go. The reason that we didn't worry about DECrementing HR and HL every seven shifts is because we set a scrolling window 34 bytes wide; the reason it's rather slow is that the window includes 1360 bytes, which is considerably bigger than the entire Lo-Res screen.

These tests tell you a few things:

Scroll

1. We really **did** draw the shape, even though we couldn't see it.

2. Depending on whether you're at the top or the bottom of the screen, you can sometimes DRAW shapes at HR and HL values which are greater than 39.

The Mystery Explained

In all of the manuals that we keep reading, and all the screen address maps that we see, the Hi-Res screens are always shown to be 280 dots wide and 192 dots high, and are made up of a block of memory 40 bytes wide and 192 bytes high. The fact that we keep getting ILLEGAL QUANTITY ERRORs any time we tread outside these limits certainly helps reinforce the fact that they are real...**but are they always real?**

We keep hearing that the Hi-Res screens are each 8192 bytes, but have you ever checked it out? Forty bytes wide by 192 bytes high **does not** equal 8192; instead, it equals 7680. There are 512 extra bytes allocated to each Hi-Res screen that are not actually displayed!

What we did in our test was to DRAW our phantom shape in this extra block of 512 bytes.

What the Hi-Res Screens Really Look Like

The representation shown in **Figure 1** of HGR page 1 (page 2 is laid out in the same way) is a more accurate picture of Hi-Res memory than the traditional rectangle that seems to be so popular.

The screen is first divided into three equal parts, and each of the three parts is further broken down into eight segments of eight lines each; i.e., 8 * 8 * 3 = 192 lines total.

Figure 1 shows how the first 129 bytes, addresses 8192 through 8320, are arranged on the screen. You should note that bytes 8312 through 8319 (8 bytes) do not appear as display bytes and are in an area which we will call the "nonvisible screen." All of the Y-coordinates from 128 through 191 are 48 bytes wide.

In order to maintain some sort of order in mapping the starting address for each horizontal line, and so that six lines per memory page could be displayed, the designers of the Apple decided to leave 8 undisplayed bytes after every 120 displayed bytes; i.e., 120 (displayed bytes) * 2 + 8 (undisplayed bytes) * 2 = 256 bytes per memory page.

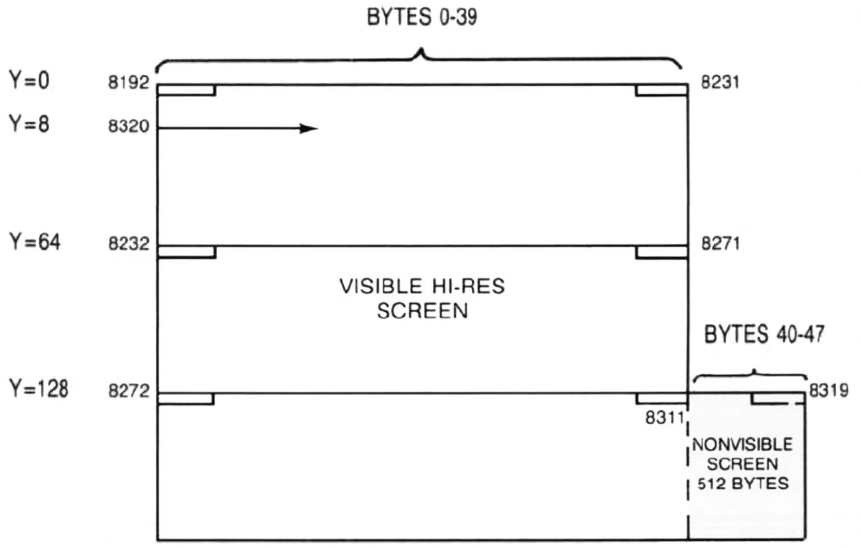

FIGURE 1 — HI-RES PAGE 1

What this leaves us with is the fact that one-third of the Hi-Res screen is actually 48 bytes (336 dots) wide.

Using These Extra 512 Bytes

As seems to be tradition in working with computers, it's almost obligatory for us to think of ways to make use of these little quirks in system design. If you were working with a program that used **both** Hi-Res pages, and you were short of memory, it might be very tempting to see if you could find ways to make use of this extra 1K of memory that now seems available. There are quite a few things that these bytes could be used for:

1. Pointers or some machine language variables could be stored here.

2. These bytes could be strung together for machine language routines. However, this isn't very practical as no more than eight bytes have consecutive addresses.

3. Block Shape Tables could be stored here. While it wouldn't be very easy to store Shape Tables here (unless they were eight bytes or less), completely DRAWn shapes could be. For example, there would have been enough room to store four of our STORM WARNING truck shapes, as well as all the scoreboard numbers. We could have allocated one regular Shape Table area to all these shapes. Then when we needed a shape, we could have SCANned it out of this area into the regular Shape Table and then DRAWn the SCANned shape onto the regular Hi-Res screen. We could also do some shape REVDIRing here without the user's knowledge.

4. As with our house test, we now have an area where we can actually DRAW a shape **off** the **visible** screen and then SHIFTL it, dot by dot, onto the screen for a smoother effect.

CAUTION: Use of this area is, however, rather dangerous, as it can be easily erased. Any HGR or HGR2 command will erase whatever is stored here, as will a CALL 62454 which sets the screen to the last HPLOTted color. If you're using SHIFT animation, it is also possible for some dummy to drive his truck over whatever you've got there; so use it with great caution. About the only way to maintain the integrity of these 512 bytes is to use the soft-switches to set graphics mode and clear the screen with something like **HCOLOR=0: FOR Y=0 TO 191: HPLOT 0,Y TO 279,Y: NEXT.**

continued on next page

Save it to disk with **BSAVE SET.SHAPE, A$8500,L$88**. This routine is completely relocatable and may be placed anywhere in memory that you desire.

The final program, LETTER.SCROLL (**Listing 4**), demonstrates the scrolling feature. This is an example of how you would write your own Applesoft program. LETTER .SCROLL is heavily REMed, so you should be able to find you way through it easily.

Briefly, here's how it works:

After loading all the needed routines (along with the block shape driver), the program jumps to **line 320** to set up the shape pointers which will be used by SET.SHAPE for the alphabet and number characters. The locations for these tables can also be changed; however, you'll need to modify SET.SHAPE so that it knows where to find them.

In **line 130** VT and VB are set to 170 and 182 which is where all the text will be DRAWn and scrolled. You can change these to whatever locations you'd like as long as they're in the lower one-third of the screen within the range of 128 through 191. Since HL will never be changed, we set it here to 41, which is two bytes off the visible screen.

After getting the character that we want to "PRINT," either from the keyboard or by pulling it out of our predefined string, we POKE the ASCII code for the character into memory location 25 ($19) and immediately **CALL SET.SHAPE**. Upon returning from SET.SHAPE, memory location 251 is tested for 1 to determine if the character was a legal one.

Next we move HL over to 40 (all our shapes are two bytes wide so HL and HR are set at 40 and 41 to DRAW) and the character is DRAWn on the nonvisible screen.

Now HL is moved back to 0, the leftmost edge of the screen, and the entire area from HL=0 to HR=41 is scrolled left two bytes. As we do this the new character is brought onto the screen, and whatever has already made the full trip falls off the left edge into oblivion.

If you move the Block Shape Table of characters, you'll need to either change the pointers in **lines 380-410** and modify SET.SHAPE for the special character pointers, or you can simply modify location 251 (SHNUM) before DRAWing shapes. For example, let's say you moved the Shape Table to begin at $7000. All the low table bytes would be $10 bytes ($80-$70=$10) lower. In other words, the SHNUM value would be 16 ($10) bytes lower for every shape in the table. Therefore, after returning from SET.SHAPE, and just before CALLing DRAW, you could modify the high byte in SHNUM with:

POKE 251,PEEK(251)-16

Conclusion

I hope you now have a clearer picture of how the Hi-Res screens are really laid out, and that you find this Hi-Res graphics technique as useful as I have.

The LETTER.SCROLL program can be used "as is" if you want; however, there is certainly plenty of room for enhancement, which I leave to your creative imagination.

This article marks the end of our conventional block shape discussion. Next month we'll begin a new series on Double Hi-Res Graphics.

LISTING 1 — 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 00 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 3F
```

KEY PERFECT 4.0
RUN ON
BLOCK.ROUTINES

CODE	ADDR# - ADDR#
2683	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
1C6A	95C6 - 95FF

PROGRAM CHECK IS : 058A

CHECK CODE 3.0

ON: BLOCK.ROUTINES
TYPE: B

LENGTH: 058A
CHECKSUM: 11

LISTING 2 — BIG.LETTERS

```
8000- 06 18 06 18 06 18 06 18
8008- 07 78 07 78 06 18 06 18
8010- 06 18 07 38 03 70 01 60
8018- 00 40 FF 03 78 07 78 0E
8020- 18 06 18 0C 18 0E 18 07
8028- 78 0E 18 0C 18 0C 18 0E
8030- 38 07 78 03 78 FF 03 60
8038- 07 70 0E 38 0C 18 00 18
8040- 00 18 00 18 00 18 00 18
8048- 0C 18 0E 38 07 70 03 60
8050- FF 03 78 07 78 0E 18 0C
8058- 18 0C 18 0C 18 0C 18 0C
8060- 18 0C 18 0C 18 0E 38 07
8068- 78 03 78 FF 07 78 07 78
8070- 00 18 00 18 00 18 00 18
8078- 03 78 03 78 00 18 00 18
8080- 00 18 07 78 07 78 FF 00
8088- 18 00 18 00 18 00 18 00
8090- 18 00 18 03 78 03 78 00
8098- 18 00 18 00 18 07 78 07
80A0- 78 FF 03 60 07 70 0E 38
80A8- 0C 18 0C 18 0F 18 0F 18
80B0- 00 18 00 18 0C 18 0E 38
80B8- 07 70 03 60 FF 06 18 06
80C0- 18 06 18 06 18 06 18 06
80C8- 18 07 78 07 78 06 18 06
80D0- 18 06 18 06 18 06 18 FF
80D8- 07 70 07 70 01 40 01 40
80E0- 01 40 01 40 01 40 01 40
80E8- 01 40 01 40 01 40 07 70
80F0- 07 70 FF 01 60 03 70 07
80F8- 38 06 18 06 00 06 00 06
8100- 00 06 00 06 00 06 00 06
8108- 00 0F 00 0F 00 FF 0C 18
8110- 0C 18 06 18 07 18 03 58
8118- 01 58 01 78 01 78 03 58
8120- 07 18 0E 18 0C 18 0C 18
8128- FF 07 78 07 78 00 18 00
8130- 18 00 18 00 18 00 18 00
8138- 18 00 18 00 18 00 18 00
8140- 18 00 18 FF 0C 0C 0C 0C
8148- 0C 0C 0C 0C 0C 0C 0C 4C
8150- 0C 4C 0D 6C 0D 6C 0F 3C
8158- 0F 3C 0C 0C 0C 0C FF 0C
8160- 18 0C 18 0C 18 0E 18 0F
8168- 18 0F 58 0D 58 0C 78 0C
8170- 78 0C 38 0C 18 0C 18 0C
8178- 18 FF 03 60 07 70 0E 38
8180- 0C 18 0C 18 0C 18 0C 18
8188- 0C 18 0C 18 0C 18 0E 38
8190- 07 70 03 60 FF 00 18 00
8198- 18 00 18 00 18 00 18 03
81A0- 78 07 78 0E 18 0C 18 0C
81A8- 18 0E 18 07 78 03 78 FF
81B0- 0D 60 0F 70 07 38 0F 18
81B8- 0D 58 0D 58 0C 18 0C 18
81C0- 0C 18 0C 18 0E 38 07 70
81C8- 03 60 FF 0C 18 0E 18 06
81D0- 18 03 18 03 18 03 78 07
81D8- 78 0E 18 0C 18 0C 18 0E
81E0- 18 07 78 03 78 FF 03 60
81E8- 07 70 0E 38 0C 18 0C 00
81F0- 0E 00 07 70 01 78 00 18
81F8- 0C 18 0E 38 07 70 03 60
8200- FF 01 40 01 40 01 40 01
8208- 40 01 40 01 40 01 40 01
8210- 40 01 40 01 40 01 40 0F
8218- 78 0F 78 FF 03 60 07 70
8220- 0E 38 0C 18 0C 18 0C 18
8228- 0C 18 0C 18 0C 18 0C 18
8230- 0C 18 0C 18 0C 18 FF 01
8238- 40 01 40 03 60 07 70 0E
8240- 38 0C 18 0C 18 0C 18 0C
8248- 18 0C 18 0C 18 0C 18 0C
8250- 18 FF 02 10 07 38 07 38
8258- 0F 7C 0D 6C 0C 4C 0C 4C
8260- 0C 0C 0C 0C 0C 0C 0C 0C
8268- 0C 0C 0C 0C FF 0C 18 0C
8270- 18 06 30 06 30 03 60 03
8278- 60 01 40 03 60 03 60 06
8280- 30 06 30 0C 18 0C 18 FF
8288- 01 40 01 40 01 40 01 40
8290- 01 40 01 40 03 60 07 70
8298- 0E 38 0E 18 0C 18 0C 18
82A0- 0C 18 FF 0F 78 0F 78 00
82A8- 18 00 30 00 30 00 60 01
82B0- 40 03 00 06 00 06 00 0C
82B8- 00 0F 78 0F 78 FF 07 70
82C0- 07 70 01 40 01 40 01 40
82C8- 01 40 01 40 01 40 01 40
82D0- 01 60 01 60 01 60 01 40
82D8- FF 0F 78 0F 78 00 38 00
82E0- 70 01 60 03 40 07 00 0E
82E8- 00 0C 00 0C 18 0E 38 07
82F0- 70 03 60 FF 03 60 07 70
82F8- 0E 38 0C 18 0C 00 0E 00
8300- 07 00 0E 00 0C 00 0C 18
8308- 0E 38 07 70 03 60 FF 06
8310- 00 06 00 06 00 06 00 0F
8318- 7C 0F 7C 06 0C 06 1C 06
8320- 38 06 70 07 60 07 40 07
8328- 00 FF 03 60 07 70 0E 38
8330- 0C 18 0C 00 0E 00 07 78
8338- 03 78 00 18 00 18 00 18
8340- 0F 78 0F 78 FF 03 60 07
8348- 70 0E 38 0C 18 0C 18 0E
8350- 18 07 78 03 78 00 70 01
8358- 60 03 40 07 00 0E 00 FF
8360- 00 60 00 60 00 60 00 60
8368- 00 60 01 60 03 40 07 00
8370- 0E 00 0C 00 0C 00 0F 78
8378- 0F 78 FF 03 60 07 70 0E
8380- 38 0C 18 0C 18 0E 18 07
8388- 70 0F 78 0C 18 0C 18 0E
8390- 38 07 70 03 60 FF 00 38
8398- 00 70 01 60 03 40 07 00
83A0- 0F 60 0F 70 0C 38 0C 18
83A8- 0C 18 0E 38 07 70 03 60
83B0- FF 01 40 01 40 00 00 01
83B8- 40 01 40 03 40 07 00 0E
83C0- 00 0C 00 0C 18 0E 38 07
83C8- 70 03 60 FF 01 40 01 40
83D0- 00 00 01 40 01 40 01 40
83D8- 01 40 01 40 01 40 01 40
83E0- 01 40 01 40 01 40 FF 01
83E8- 60 01 60 01 60 00 00 00
83F0- 00 00 00 00 00 00 00 00
83F8- 00 00 00 00 00 00 00 00
8400- 00 FF 00 40 01 00 01 60
8408- 01 60 01 60 00 00 00 00
8410- 00 00 00 00 00 00 00 00
8418- 00 00 00 00 FF 00 00 00
8420- 00 00 00 00 00 00 00 00
8428- 00 00 00 00 00 00 00 00
8430- 00 00 00 00 00 00 00 FF
```

KEY PERFECT 4.0
RUN ON
BIG.LETTERS

CODE	ADDR# - ADDR#
2514	8000 - 804F
2AD3	8050 - 809F
2188	80A0 - 80EF
28FF	80F0 - 813F
2AFB	8140 - 818F
2A4F	8190 - 81DF
2C1B	81E0 - 822F
27CF	8230 - 827F
24E2	8280 - 82CF
2675	82D0 - 831F
28C6	8320 - 836F
24B1	8370 - 83BF
293C	83C0 - 840F
0FD2	8410 - 8437

PROGRAM CHECK IS : 0438

CHECK CODE 3.0

ON: BIG.LETTERS
TYPE: B

LENGTH: 0438
CHECKSUM: 48

LISTING 3 — SET.SHAPE

```
8500- A5 19 C9 41 90 14 C9 5B
8508- B0 7C 38 E9 41 AA BD 60
8510- 84 85 FB BD 40 84 8D 30
8518- 93 60 C9 20 D0 0A A9 84
8520- 85 FB A9 1D 8D 30 93 60
8528- C9 2C D0 0A A9 84 85 FB
8530- A9 02 8D 30 93 60 C9 2E
8538- D0 0A A9 83 85 FB A9 E7
8540- 8D 30 93 60 C9 21 D0 0A
8548- A9 83 85 FB A9 CC 8D 30
8550- 93 60 C9 3F D0 0A A9 83
8558- 85 FB A9 B1 8D 30 93 60
8560- C9 30 D0 0A A9 81 85 FB
8568- A9 7A 8D 30 93 60 C9 31
8570- 90 14 C9 3A B0 10 38 E9
8578- 31 AA BD 90 84 85 FB BD
8580- 80 84 8D 30 93 60 A9 01
8588- 85 FB 60
```

LISTING 4 — LETTER.SCROLL

```
10  REM   **********************
20  REM   *    LETTER.SCROLL    *
30  REM   *  BY ROBERT DEVINE   *
40  REM   * COPYRIGHT (C) 1984 *
50  REM   * BY MICROSPARC, INC *
60  REM   * LINCOLN, MA. 01773 *
70  REM   **********************
80  HIMEM: 32768: REM  SET HIMEM TO PROTECT BLOCK SH
    APE TABLE
90  PRINT  CHR$ (4)"BLOAD BLOCK.ROUTINES,A$9876": CALL
    37799: REM  LOAD BLOCK ROUTINES/SET UP YTABLE P
    OINTERS
100 PRINT  CHR$ (4)"BLOAD BIG.LETTERS"
110 PRINT  CHR$ (4)"BLOAD SET.SHAPE": REM  ASC TO S
    HAPE ADDRESS ROUTINE
120 GOSUB 320: REM  SET UP TABLE ADDRESS ARRAY
130 POKE 252,170: POKE 253,182: POKE 254,41: REM
    SET VT,VB, AND HR THAT WE WILL SCROLL
140 HOME : VTAB 22: PRINT "** COPYRIGHT 1984 BY MIC
    ROSPARC, INC. **": VTAB 12: INPUT "WHAT IS YOUR
     NAME? ";NAME$
150 PRINT : PRINT "(K)EYBOARD OR (T)EXT STRING "NAM
    E$" ? ";: GET A$
160 HGR :X =  PEEK (49234): REM  SET PAGE 1 WITH FU
    LL SCREEN GRAPHICS
170 IF A$ = "T" THEN 240
180 GET A$: POKE 25, ASC (A$): CALL 34848: ON ( PEEK
    (251)) GOTO 180: REM  GET CHARACTER/TRANSLATE T
    O SHAPE TABLE/TEST FOR ILLEGAL
190 POKE 255,40: REM   SETUP TO DRAW 2 BYTES OFF TH
    E VISIBLE SCREEN
200 CALL 37679: REM  DRAW THE LETTER
210 POKE 255,0: REM  MOVE HL TO LEFT EDGE OF SCREEN

220 FOR X = 1 TO 14: CALL 37381: NEXT : REM  SHIFT
    ENTIRE LINE OF TEST 2 BYTES LEFT
230 GOTO 180: REM  GET ANOTHER LETTER
240 A$ = "AS YOU CAN SEE " + NAME$ + ", YOU CAN SCRO
    LL ANY KIND OF TEXT ACROSS THE SCREEN.....ABCDE
    FGHIJKLMNOPQRSTUVWXYZ12345678982!.,"
250 FOR Y = 1 TO  LEN (A$): POKE 25, ASC ( MID$ (A$
    ,Y,1))
260 CALL 34848: ON ( PEEK (251)) GOTO 290: REM  TRA
    NSLATE/TEST FOR ILLEGAL
270 POKE 255,40: CALL 37679: POKE 255,0: REM  DRAW
    OFF SCREEN/SETUP FOR SCROLL
280 FOR X = 1 TO 14: CALL 37381: NEXT X: REM  SHIFT
     IT ALL LEFT 2 BYTES
290 NEXT Y: REM  NEXT CHARACTER
300 FOR X = 1 TO 270: CALL 37381: NEXT : REM  SHIFT
     IT ALL THE WAY ACROSS
310 TEXT : HOME : GOTO 150
320 REM  SETUP ADDRESS BYTE TABLES IN MEMORY
330 FOR X = 33856 TO 33881: READ Y: POKE X,Y: NEXT
    : REM   SET ALPHA LOW BYTE TABLE   STARTS AT $8
    440
340 FOR X = 33888 TO 33913: READ Y: POKE X,Y: NEXT
    : REM   SET ALPHA HIGH BYTE TABLE  STARTS AT $8
    460
350 FOR X = 33920 TO 33928: READ Y: POKE X,Y: NEXT
    : REM   SET NUMBER LOW BYTE TABLE  STARTS AT $8
    480
360 FOR X = 33936 TO 33944: READ Y: POKE X,Y: NEXT
    : REM   SET NUMBER HIGH BYTE TABLE STARTS AT $8
    490
370 RETURN
380 DATA    0,27,54,81,108,135,162,189,216,243,14,4
    1,68,95,122,149,176,203,230,1,28,55,82,109,136,
    163: REM ALPHA TABLE LOW BYTES
390 DATA   128,128,128,128,128,128,128,128,128,128,
    129,129,129,129,129,129,129,129,129,130,130,130
    ,130,130,130,130: REM  ALPHA TABLE HIGH BYTES
400 DATA  190,217,244,15,42,69,96,123,150: REM  NUM
    BER TABLE LOW BYTES
410 DATA  130,130,130,131,131,131,131,131,131: REM
     NUMBER TABLE HIGH BYTES
```

```
KEY PERFECT 4.0
RUN ON
LETTER.SCROLL

CODE      LINE# - LINE#
------    -------------
 8EC2        10 -  100
 F9C7       110 -  200
 D530       210 -  300
812DC6      310 -  400
  190B      410 -  410
PROGRAM CHECK IS : 07A5
```

```
CHECK CODE 3.0

ON: LETTER.SCROLL
TYPE: A

LENGTH: 0722
CHECKSUM: 8E
```