A New Character Set for the VIC-20

by Mike Bassman

A technique to design your own VIC characters is described. This involves changing the character-ROM printer and copying character definitions into RAM from their ROM locations.

Custom Characters
requires:
VIC-20

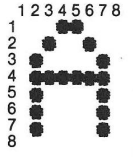
The new VIC-20 from Commodore packs a lot of wallop for $300. It includes many features that were previously found only on more expensive computers. The VIC uses a character set nearly identical to the PET's and very similar to those of Ohio Scientific, Sorcerer, and Atari computers. Special graphic characters are provided along with the regular alphanumerics. With these graphic characters, you can draw lines, build pictures, etc. The problem is that there are never enough characters available.

Although the VIC allows you to draw almost anything made of straight or curved lines, many games are more difficult to program because of the lack of tank, boat, or plane characters such as those found in OSI computers. Though the manual does not document it, it is possible to change some of the VIC's characters to those of your own choosing.

To create new characters, you must understand how the VIC stores its old ones. As with most computers, the VIC's characters are defined in an 8 by 8 dot matrix. That is, any character can be made of up to eight small dots (called pixels) across by eight down. Each pixel is represented in memory by one bit. If the bit is a 1, then the corresponding pixel will be lit when the character is displayed on the screen. An entire character consists of 64 bits, or eight bytes. The format in which the bytes are stored is as follows: the first byte represents the first row of pixels going across, the second byte is the second row, and so forth. For a more detailed example, see figure 1, where the character 'A' is shown in both pixel and binary formats.

The VIC keeps all of its characters in 4K of ROM. Since each character occupies eight bytes of memory, the ROM must contain 512 characters. At first glance, it does not seem to have that many, but this is how the 4K is used. The first 128 characters (1K) are the familiar upper case/graphics set. The next 128 are the same ones, only in reverse. This is followed by the lower/upper case set and its reverse set.

Figure 1: **Binary Representation of Character In Memory**



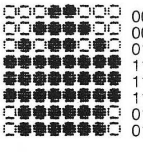
0001 1000
0010 0100
0100 0010
0111 1110
0100 0010
0100 0010
0100 0010
0000 0000

On most computers, including the PET, the character ROM is not addressed in the microprocessor's normal memory space, but is addressed so that it is only available to the CRT controller chip. In the VIC the ROM is actually addressed from $8000 to $8FFF in the 6502's address space!

Since the character information is stored in ROM, it can't be changed. However, since the pointer to the character ROM is stored in RAM, it can be changed to point anywhere else, including RAM. The pointer is at 36869, 36870. Its normal value is 240, enabling the upper case/graphic set, while a value of 242 will enable the alternate lower/upper case set. POKEing a 255 into 36869 will move the pointer from its normal 32768 address to 7168. This just happens to be right at the top of BASIC RAM, which normally runs from 4096 to 7679. By lowering the upper limit from 7679 to 7168, we have stolen 512 bytes from BASIC, or enough for 64 characters. Since it is impossible to put 4K worth of characters into 512 bytes, you have to select the 64 used most. Normally these would be the upper case letters, the numbers, and various punctuation marks, or the characters with screen codes ranging from 0 to 63. Of these, certain ones, such as the @ and the British pound sterling symbol, are seldom used, and can be replaced with your custom characters.

There are a few tricks you should know before attempting to make new characters. Most importantly, you have to reserve locations 7168 to 7679 so they are not disturbed by BASIC. There are two pointers that must be adjusted to accomplish this: the top-of-BASIC pointer (51-52), and the top-of-string storage pointer (55-56). As with every

Figure 2: **Alien Character**



Binary	Hex	Decimal
00011000	18	24
00111100	3C	60
01011010	5A	90
11111111	FF	255
11111111	FF	255
11111111	FF	255
01111110	7E	106
01111110	7E	106

6502 pointer, the low byte is stored first, followed by the high byte. Both pointers point to 7680, and must be changed to point to 7168. Instead of recreating all of these characters, it is faster to copy them from their original ROM locations.

We are now ready to define a custom character — an alien, which you might find useful in game programs. The easiest way to go about this is to take a piece of graph paper, mark off an 8 by 8 segment, and fill in the appropriate squares. Now take each row and make a binary representation for it. Convert the binary for each row into decimal, because this information will be stored in BASIC DATA statements. Our alien is defined and digitized in figure 2.

This program will put the character shown in figure 2 over the character with screen code 0 (the '@'). Now when you hit the '@' key, you will get an alien. The alien can now be used in your own program for gaming or anything else. The unfortunate part is that all those other characters are no longer available; only the 64 you originally moved down are available. If you try to use any others, you will get whatever random gibberish happens to lie beyond 7679.

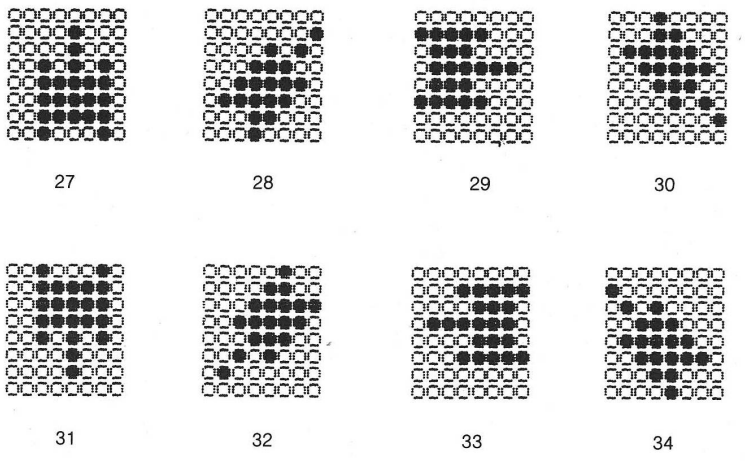
Listing 2 is a program that defines a whole series of tank characters — one for each of eight 45-degree angles. They are shown in figure 3, in their pixel formats. Below them are the screen codes they will be assuming. Now many games that were before restricted to the computers with the appropriate game characters can be programmed on the VIC.

The screen codes for the tank characters are from 27 to 34, as shown in figure 3. They are contiguous for programming ease. The tanks replace the brackets, the up and left arrows, the British pound sign, the space, the exclamation point, and the quote (in that order). If you list a program with this character set implemented, your punctuation is going to look awfully funny!

In addition to defining the tank characters, the program lets you drive the tank around the screen. The controls are 'Z' to turn left, '/' to turn right, and space to move in the direction you are pointed. To make a complete program, the lines of the program in listing 2 must be overlayed onto those of the program in listing 1.

Contact the author at 39-65 52nd St., Woodside, NY 11377.

Figure 3: **Eight Tank Characters**



Listing 1

```
10 POKE 52,28
20 POKE 56,28
30 S=7168
40 F=32768
50 FOR K=0 TO 63*8
60 X=PEEK(F+K)
70 POKE S+K,X
80 NEXT K
90 READ X
100 IF X=-1 THEN POKE 36869,255: GOTO 1000
110 FOR K=0 TO 7
120 READ N
130 POKE S+8*K,N
140 NEXT K
150 GOTO 90
160 DATA 00,24,60,90,255,255,255,126,126
500 DATA -1
1000 PRINT "CHARACTER SET COMPLETED": END
```

Listing 2

```
5 PRINT"■": FOR K=7680 TO 8185: POKE K,35: NEXT
200 DATA 27,0,8,8,42,62,62,62,34
210 DATA 28,0,1,10,28,62,124,24,16
220 DATA 29,0,248,112,126,112,248,0,0
230 DATA 30,16,24,124,62,28,10,1,0
240 DATA 31,34,62,62,62,42,8,8,0
250 DATA 32,4,12,31,62,28,40,64,0
260 DATA 33,0,31,14,126,14,31,0,0
270 DATA 34,0,128,80,56,124,62,24,8
280 DATA 35,0,0,0,0,0,0,0,0
1000 POKE 36879,14: UL=7680: LL=22: S=35
1005 FOR K=0 TO 21: POKE UL+K,45: POKE 8164+K,45
1010 POKE UL+K*LL,45: POKE UL+K*LL+21,45: NEXT
1020 L=UL+LL*5+5:C=27:D(1)=-22:D(2)=-21:D(3)=1:D(4)=23:D(5)=22
1025 D(6)=21: D(7)=-1: D(8)=-23
1030 POKE L,C
1050 GET A$: IF A$="" THEN 1050
1060 IF A$<>"Z" THEN 1090
1070 C=C-1: IF C=26 THEN C=34
1080 GOTO 1030
1090 IF A$<>"/" THEN 1120
1100 C=C+1: IF C=35 THEN C=27
1110 GOTO 1130
1120 IF A$<>" " THEN 1050
1130 IF PEEK(L+D(C-26))<>35 THEN 1050
1140 POKE L,S: L=L+D(C-26): GOTO 1030
```

MICRO™