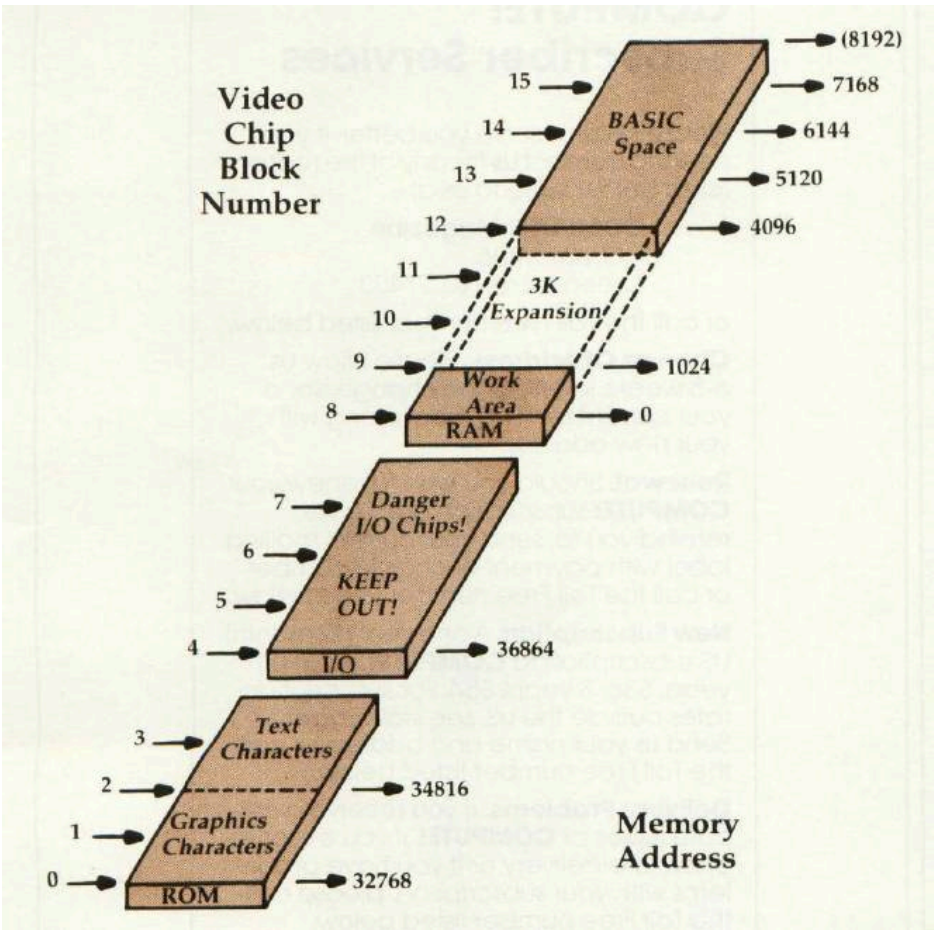
Part I

Visiting The VIC-20 Video

Jim Butterfield, Associate Editor

In which the traveler discovers a new way of viewing the computer's memory: through a video chip. This is the first of a multi-part series about the structure and uses of the VIC's video chip.

If we want to put the VIC-20 video chip to work, we must learn to see things from its standpoint. It sees the computer memory in a way that differs significantly from the way the processor chip sees it. Let's look at what the video chip sees:



How the video chip sees memory.

The video chip sees only the memory shown above. Even if you have expanded your computer to include lots of extra RAM above address 8191, the chip can't see it. The chip sees only the character ROM, in blocks 0, 1, 2, and 3; and the lowest 8K of RAM (in blocks 8 to 15). Blocks 4, 5, 6, and 7 would look at the Input/Output area, but take my advice: don't do it – no good will come from these addresses.

What The Chip Wants

The video chip wants to dig out two things from memory and deliver them to the screen. It wants to look at "screen memory" – usually the characters you have typed. On a minimum 5K VIC, that's block 15.5, which corresponds to decimal address 7680 or hexadecimal 1E00. Did I mention that for screen memory, we can look at "half blocks"? It makes sense, since only five hundred odd characters are needed to fill the screen.

By the way, the official name for screen memory is the "video matrix." Whatever you call it, if you POKE 7680,1 on an unexpanded VIC, you'll see the letter A appear at the start of the screen. Unless, of course, you're printing white on white, in which case you need very good vision to see it.

The second thing that the chip wants from memory is the "character set" – instructions on how to draw each character on the screen. On a typical VIC, this will be either block 0 for the graphics character set or block 2 for the text mode (upper- and lowercase). You can change it, but you'll usually want to stay with even numbers: a full character set including the reversed characters takes up 2048 bytes of memory.

The official name for the character set is "Character Cells," although the term "Character Base" is coming into use. Whatever you call it, you can't POKE 32768,55 and expect anything to happen – the standard characters are in ROM and cannot be changed. They're carved in stone, or silicon, to be more exact. If you want to switch to custom characters, you'll need to stage them in RAM and tell the chip which block to take them from.

There's a third thing that the chip uses, but it doesn't come from regular memory in the usual way. That's the screen colors (the "Color Matrix"). This color information for each character comes through the back door, so to speak, and we won't worry about the details too much here. When we need to, we'll set the color and assume everything will work.

Architecture

Looking at the diagram, we can begin to see why the VIC does its odd screen switch when you add memory. In the 5K VIC, the screen sits at the top of memory – and that's the highest address that the video chip can see (block 15.5). If we add 3K RAM expansion, the screen can stay where it is above the BASIC RAM area. But if we add 8K or more, the video chip can't see that high, and the screen memory must flip down to the bottom where it won't get in the way of your BASIC program. Which bottom, you may ask? It turns out to be block 12, which is memory address 4096 or hexadecimal 1000, even if the 3K expansion is in place.

You can move this around yourself, of course, and we'll be doing that in just a few moments.

The trick is mostly location 36869, which contains instructions on which blocks to use for screen and characters. We do it this way: select which blocks you want for each. Now, multiply the screen block (not including the .5 if you're using it) by 16 and add the character block. POKE the result into 36869, and the job's done. We'll need to do a couple of other things for sanity's sake, but that's the main job.

The "half page" for the screen memory goes into location 36866; you invoke it by adding 128 to the "column count" if you want to go the extra distance. That means that under normal circumstances (22 columns), you want to POKE 36866,22 for an exact block number, and POKE 36866,150 to nudge to the extra half page.

An Adventure

Let's do something useless, but fun. We'll move the screen memory down to address zero (that's block 8). We can't play with this area – too many important things are happening there – but we can watch interesting things in progress, like the timer and the cursor doing their peculiar things.

First, the calculation. We want the character set to stay the way it is (block 0 for graphics), and we want to move the screen memory to block 8. Eight times 16 plus zero gives 128. No half block, so 36866 should be 22.

A preliminary step: let's make sure that we don't print white-on-white by clearing the screen and typing:

```
FOR J = 37888 TO 38911:POKE J,0:NEXT J
```

Ready? Here goes: enter POKE 36869,128:POKE 36866,22. Press RETURN. No, we haven't crashed, but we'll have to type blind from now on.

First, examine the fascinating busy things that are under way. The timer is working away in three bytes. At first glance, only one byte seems to be changing. The cursor flash is being logged and timed somewhat below. And if you start typing, you'll see a whole new series of working values coming into play. Indeed, if you can type blind, you might try PRINT 1234 + 5678 and watch the flurry of activity.

If you type a lot, the screen will start to scroll, and the display will start to vanish as the colors are rolled off the top.

Restore everything to normal by holding down RUN/STOP and tapping the RESTORE key.

This has been a first exploration, but you may feel that you understand better what the video chip is up to. Indeed, you may feel that you have gained some measure of control.

There's much more to be learned. This is a start.

- Back to previous page
- See this article as it appeared in the magazine
- View this issue's table of contents