Graphics and bigger screen in Basic on the expanded vic-20.

By Anders “Boray” Persson

First, (as everyone probably know), when having expansion memory (ram in block 1), then both the basic memory and the screen is moved to other positions in memory.

- The screen now starts at: 4096 (7680 on unexpanded).
- The colors for the screen starts at 37888 (38400 on unex.)
- The basic memory starts at 18*256+1=4609 (4097 on unex.)

If you are going to use expansion ram and want to use your own graphics (or a bigger screen), then you need to move the basic memory. The reason for this is that the vic chip only can access the internal memory (not the expansion memory). And by default, it only have enough memory for the default screen before the basic memory starts. It's made this way so that all of the memory can be used for basic programs by default.

So you need to alter the start address of the basic memory and move it forward in memory to make room for things that needs access of the vic chip. So that these things can be put in the internal memory.

(It is however possible to put graphics data in the internal tape buffer if you think it's enough with only 38 characters and a normal sized screen. Then you don't need to move the basic memory at all. More about this later.)

Moving the basic memory

If you for example do this:

poke44,32:poke32*256,0:new

Then your basic memory will start at the expansion ram (8192) and leave the whole graphics mem free for your own graphics, a bigger screen, etc... (32*256 = 8192). You might not need to move it that far though. For example on my Tribbles game, I have it at 28*256 I think (poke 44,28:poke28*256,0:new)

REMEMBER - Always load basic programs with just ,8 !!! If you load it ,8,1 then it will be loaded to the same memory position from where it was saved, and that is not very good when you have moved the basic memory.

A bigger screen

Now when you have moved your memory, you can use some of the internal memory for a bigger screen. (Mostly usable on PAL where quite much of the display is unused.) You alter the screen size and positions with the following vic registers:

- 36864 bits 0-6: screen horizontal center
- 36865: screen vertical center
- 36866 bits 0-6: number of columns
- 36867 bits 1-6: number of rows

In basic, to set:

- screen x position: poke 36864,x
- screen y position: poke 36865,y
- screen width: poke 36866,(peek(36866)and128)+w
- screen height: poke 36867,(peek(36867)and129)+h*2

For an example in basic (and for trying different settings out), download my "overscan" program.

When you have opened a bigger screen with a different column size than the default, then the lines will behave strangely, but don't worry, this is normal! The system screen still is 22x23 regardless of how big you screen is, and this is why the screen seems to behave strangely. If you don't like this, then keep the original width and only make the screen taller. It's here much of the unused space is anyway (on PAL). You can't use print commands to put stuff on the extra space. So the only way (in pure basic) is to poke. The screen just continues beyond where the default screen ends at 4602, so pokeing values greater than that changes the contents of the extra space on the screen. It works exactly as pokeing on the normal screen, but you just have a bigger space now. Another way is to use my "Extra Screen" program, then you can use normal print commands (almost). (You are allowed to include the machine language part in your programs)

> An example using "Extra screen":
>
> ```
> 10 print "{clr}This text will be put"
> 20 print "on the extra space below"
> 30 print "the 'normal' screen."
> 40 sys 5352
> 50 print "{clr}And this will be on the"
> 60 print "very top of the whole screen."
> ```
>
> SYS 5352 just copies the "normal" screen to below the "normal" screen in a jiffy. In other words, you don't need to start the whole "EXTRA SCREEN" program, just use this little routine to copy the screen. I did this in my Mega Omega game.

Graphics

This is pretty straight forward. You reserve memory the way described above and then find a suitable place in memory (after the space used for the bigger screen). The register to use is:

36869 bits 0-3

In basic: poke 36869,(peek(36869)and240)+**a**

The addresses **a** you can use are

- 13 for 5120
- 14 for 6144
- 15 for 7168
- 8 for 824 (chars 103-127) and 664 (chars 83-95) (Thanks Mobsie!)
- 0-3 for the character rom.

So for example poke 36869,(peek(36869)and240)+14 will use the graphics data at memory address 6144 and forward.

The easiest way to do graphics is to use some program to draw directly into the memory and then just save the memory out as a file (for example with a machine language monitor). I usually use "The final Cartridge"'s ML monitor on the C64 to input and save the graphics. Then when you are to use it, you just load ,8,1 and it will load in the right place.

The tape buffer - Selecting 8 as "a" above makes it possible to put graphics in the internal tape buffer. For this you don't need to move the basic memory or anything. And it should work on both expanded and unexpanded vics!!! Also seems to be the only possible position if you intend to compile your program with the Austro Compiler. Here is a demo of how to use this position: Winter simulator.

Working / and Saving

When working on a project, it's easiest to first load any graphics and ML files,8,1, then move the basic memory as described earlier and last load the basic ,8. Then just save the basic program ,8 as you progress with better and better versions.... But when you are ready and want to turn it into something that others easily can load on their vic... Then there are two approaches.

1. The mutliple file approach
2. Single file approach

The multiple file thing is simply that you make a loader that moves the memory and loads all the files. It can be a little tricky as the NEW command is used to move the basic memory... A tip if you like to use this approach is to use the keyboard buffer that starts at position 631. Position 198 holds the number of letters in the buffer. So by doing this: poke631,131:poke198,1 You put a "load/run" keypress in the keyboard buffer. And if you before that print something like LOAD "PROGRAM",8{up}{up}{up} on the screen, then you can make it load a program even after the new command. Take a look at the first file of my "VIC EXTRA SCREEN" as an example.

The Single file thing is a lot nicer, and it also makes your game work on both disk and tape. It involves having two basic programs as well as any graphics and ML parts in memory at the same time and then saving the whole thing. The main basic program at your new moved basic position plus a little starting program at the original basic position that moves the memory and runs the main program.

To run a basic program that is somewhere else in memory and not in the current basic memory, you have to set these pointers first:

```
43-44   Start of Basic
45-46   Start of Variables  First byte after program
47-48   Start of Arrays     First byte after program
49-50   End of Arrays       First byte after program
51-52   String storage      End of memory+1
```

Let's say we have the whole memory set up like this:

-Small Basic start
-Room for bigger screen
-Machine Language
-Graphics Data
-Main Basic program

When a program is loaded, the poiners 45,46 etc. are automatically set to behind the loaded program, and because I have the basic program last in the resulting file, those pointers are set by themselves when the file is loaded. Because of this, I only have to change the start pointer.

So, if you have the basic starting at 8192, you only have to do the following...

(reset)

10 poke 44,32:run
(make the small basic start)

load "gfx and machine language part",8,1
(loads into the graphics mem)

poke 44,32: poke 32*256,0: new
(move basic to 8192)

load "basic part",8
(load main basic part)

poke 44,18
(move back the beginning of basic to default)

save "whole program",8
(And everything saves as one big file)

Very nice as you don't even have to bother to look what the ending address is...

Good luck!

Anders Persson
http://www.boray.se

Back