PETTESTE2K – Version 03 beta. Copyright © 2018 David E. Roberts.

This test program was developed as a ‘cheap and cheerful’ test aid for the various versions of the Commodore PET computer (irrespective of whether it contains a CRT (Cathode Ray Tube) controller device or not).

On start-up, the 6502 CPU (Central Processing Unit) starts executing instructions from a RESET VECTOR found in high memory. In a standard Commodore PET machine this is where the Kernal ROM (Read Only Memory) resides at address $Fxxx.

The general problem with the PET is that most of the ROMs were soldered into the board – making it difficult to install a replacement ROM containing diagnostic code. However, in order to customise the PET for different countries, Commodore decided to place all of the customisation and internationalisation code into what they refer to as an EDIT ROM residing at $Exxx. This is a 2K ROM device and is generally installed in an IC (Integrated Circuit) socket. As a result, the EDIT ROM can be removed from the socket and a diagnostic replacement installed.

On start-up, the 6502 resets to a vector contained within the high memory of the Kernal ROM and starts to initialise the hardware. After only a few instructions, the Kernal ROM executes initialisation code stored in the EDIT ROM. This is our chance to ‘get in’ and slip our diagnostic code into the PET unnoticed!

The downside of this technique is that a small portion of the Kernal ROM must be working correctly in order for the 6502 instruction execution to reach our diagnostic ROM. This was considered a worthwhile trade-off – but should be considered if the diagnostic ROM fails to even start execution.

Of course, the diagnostic ROM outputs the results of the various tests to the screen; therefore a functioning video sub-system is required. This, however, is one of the first tests the diagnostic ROM performs!

On entry from the Kernel ROM, the diagnostics initialises any CRT controller (if present on the PET). The initialisation mode and parameters (e.g. 40/80 columns, 50/60 Hz etc.) can be configured by the user by editing the source code for the diagnostic ROM and inserting the appropriate table of register values at label CRTC\_INIT. If a CRT controller is not present on your particular PET, don’t worry, the initialisation will not take effect and no damage will occur. The electrons just spill out onto the floor ☺!

**VDU TEST**

This test writes an incrementing character code from $00 to $FF into all 8 pages (2K for an 80-column PET) of the VDU memory. If your machine is a 1K 40-column PET this should still be OK – the last 4 pages of data writes should either be discarded or overwrite the first 4 pages again. Either way, this should be fine.

The test continues by verifying that the first 4 pages (1K) of VDU memory contain the correct values. If not, the test program loops so that the test operator can see the results.

If the test fails – the diagnostic program loops, re-writes the test pattern and performs a further check. This process continues until such time as the correct data pattern is read from the VDU memory.

Obviously, a completely black picture could indicate that a number of problems exist:

* The monitor may be faulty.
* No video or synchronisation signals may be getting from the mainboard to the monitor.
* The timing chain is faulty.
* If a CRT Controller is fitted – it may be faulty.
* The CPU is not correctly executing instructions. This may be due to a number of reasons…

If the test pattern is not correct – the test operator can observe the actual characters that are being displayed as a result of the read/write operations from/to the VDU memory. This should give you some clue as to the potential fault (or at least where to start probing). Obvious errors could be a stuck data bit (either a ‘1’ where a ‘0’ should be or a ‘0’ where a ‘1’ should be). Less obvious errors are addressing faults where the correct character is stored at an incorrect address – thus either not being stored to the correct location, or overwriting another location. Of course, this may occur on the read or write cycle – or be a function of the memory device itself.

Data could also be changing. This would be indicated on the display as flickering characters (say alternating between ‘A’ and ‘B’ in the same character cell).

When the diagnostic reads the correct test pattern from the VDU memory – it performs a delay (to let the human see the results…) and moves on to the next test.



Figure 1 – A ‘PASS’ display for the VDU TEST (80-columns).



Figure 2 - A ‘PASS’ display for the VDU TEST (40-columns).

**PAGE 0/1 55AA TEST**

Assuming the VDU test completed without errors, the diagnostic ROM moves on to perform a basic ‘stuck bit’ test of page 0 and page 1 RAM (Random Access Memory). Page 0 is a special area of memory that is used by a particular addressing mode of the 6502 CPU. Page 1 (or part of page 1) is generally reserved for the stack. Each page consists of 256 bytes of RAM. A failure in either page 0 or page 1 of RAM could result in a catastrophic failure of the PET BASIC firmware stored in ROM.

This test first stores the pattern $55 (binary 01010101) into pages 0 and 1 of RAM and checks for the correct value being stored. The test next stores the pattern $AA (binary 10101010) into pages 0 and 1 of RAM and checks once again for the correct value being stored. The status of the testing (for each byte of page 0 and 1) is shown on the VDU. A ‘G’ or ‘g’ character is displayed for each byte that is GOOD/good whilst a ‘B’ or ‘b’ character is displayed for each byte that is BAD/bad respectively. Note that an upper or lower case character may be displayed – depending upon which variation of the character generator is actually fitted to your PET.

As a 1K (40-column) PET screen can display 25\*40 = 1,000 characters; I can almost use that to indicate the state of each byte of the test (although I do, unfortunately, loose the last 24 bytes). There is no problem on a 2K (80-column) PET screen though.

The first 256 bytes of the display indicate the state of storing and testing the memory in page 0 with a value of $55.

The second 256 bytes of the display indicate storing and testing the memory in page 1 with a value of $55.

The third 256 bytes of the display indicate storing and testing the memory in page 0 with a value of $AA.

The fourth 256 bytes of the display indicate storing and testing the memory in page 1 with a value of $AA.

As stated previously, the displayed character indicates if the test passed (character = ‘G’ or ‘g’) or the test failed (character = ‘B’ or ‘b’). The location of the character on the screen (i.e. the index into the 256 byte display block) indicates which byte of the memory page failed the test.

Unfortunately, using this means of testing and displaying the results of the test, it is not possible to display what the actual value was when the memory was read. I may change this test in the future to indicate the faulty value in some way. But, at the moment, a badly conceived test is better than no test at all!

If the test fails, the diagnostics keep looping on this test.

Again, it may be possible to detect ‘random’ memory faults by observing the displayed status character alternating from good to bad and vice-versa.

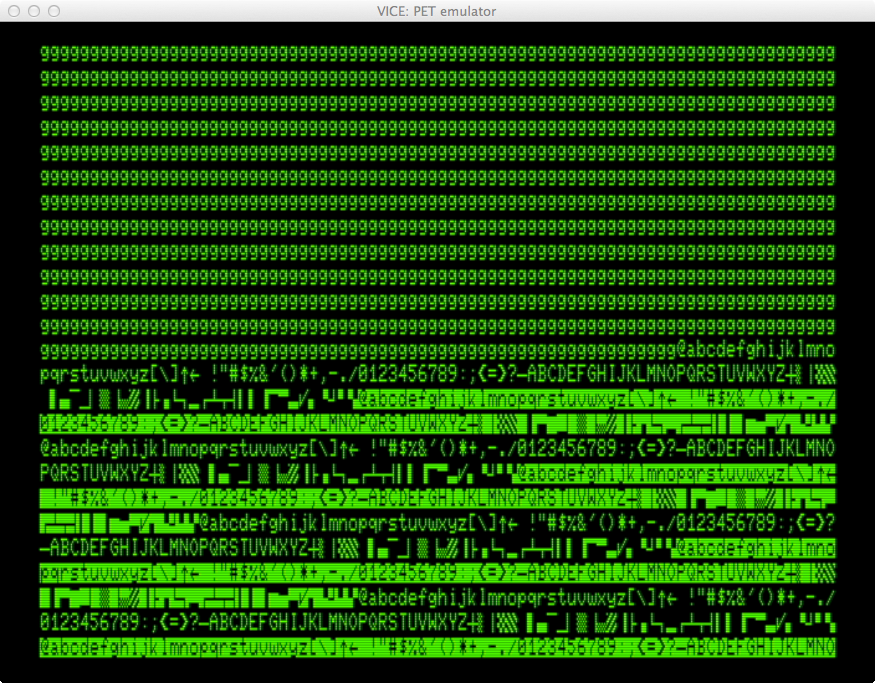


Figure 3 - A 'PASS' display for the PAGE 0/1 55AA TEST (80-columns).

Note that the upper half of the 80-column display memory remains from the previous VDU test.

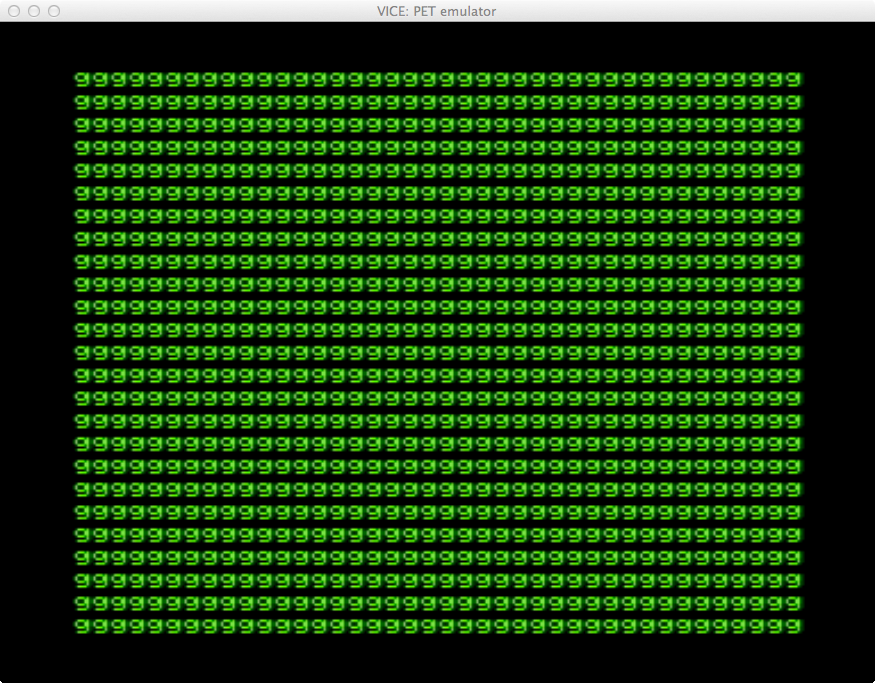


Figure 4 – A 'PASS' display for the PAGE 0/1 55AA TEST (40-columns).

**PAGE 0/1 002FF TEST**

@@@

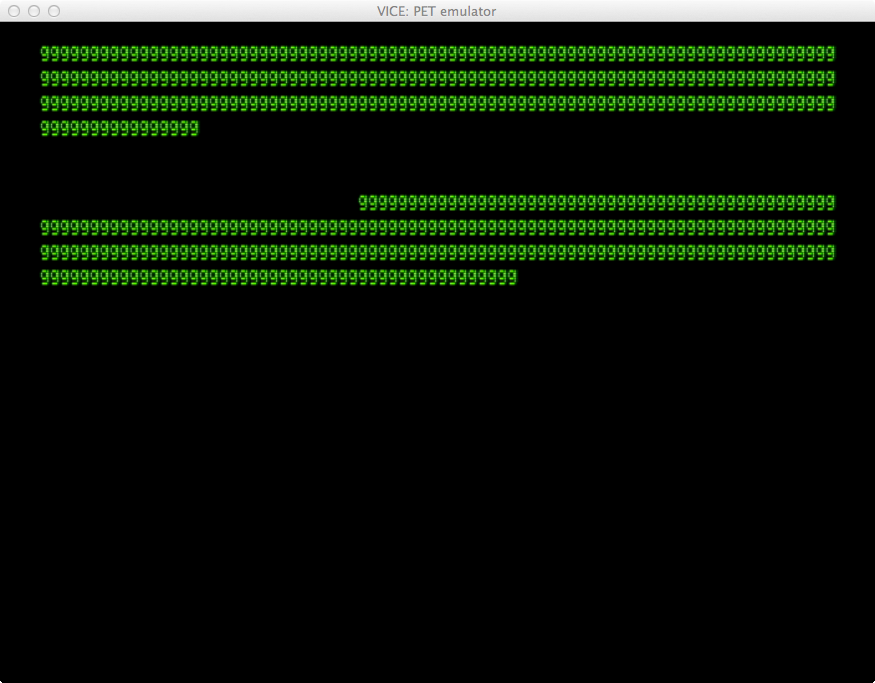


Figure 5 – A 'PASS' display for the PAGE 0/1 002FF TEST (80-columns).

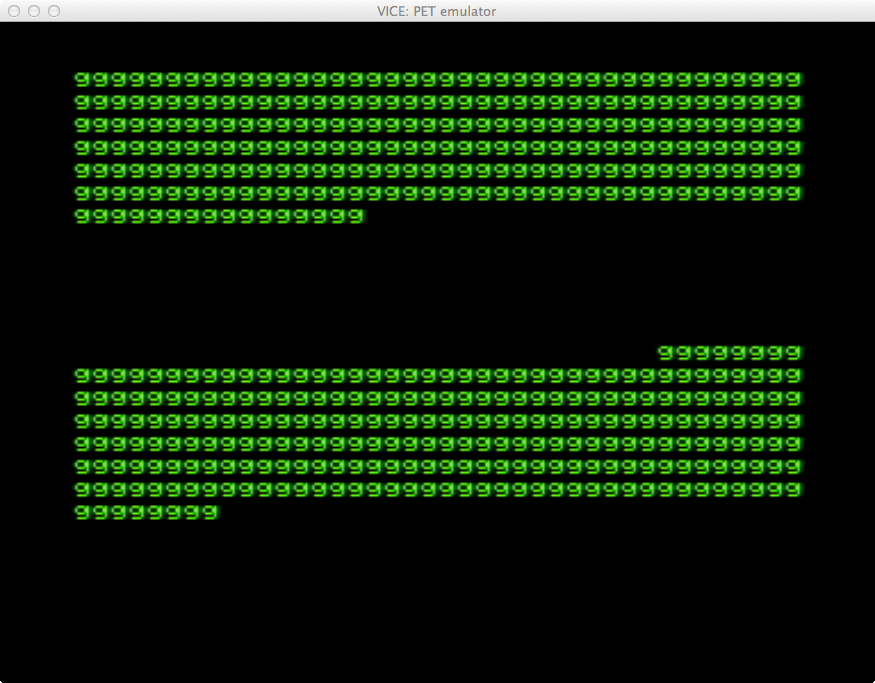


Figure 6 - A 'PASS' display for the PAGE 0/1 002FF TEST (40-columns).

**ROM/KBD/IO TEST**

@@@

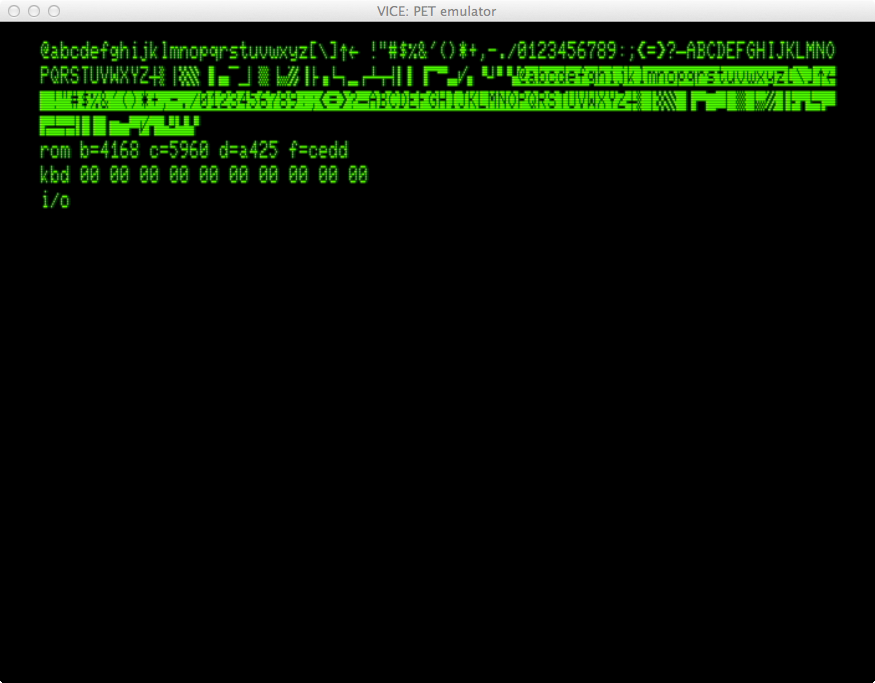


Figure 7 - A 'PASS' display for the ROM/KBD/IO TEST (80-columns).

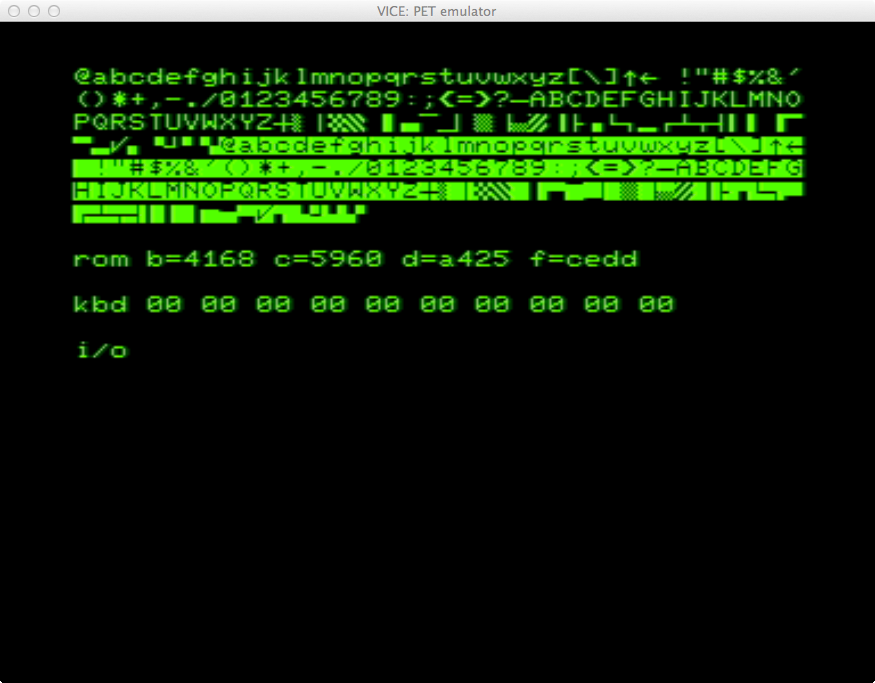


Figure 8 - A 'PASS' display for the ROM/KBD/IO TEST (40-columns).

To be continued…