Project Documentation

Commodore VIC-20: RAM Expansion Test

Project number: 173

Revision: 1.0

Date: 05.05.2021

Commodore VIC-20 RAM Expansion Test v1.0

Module Description

Inhalt

Purpose of the Software

The primarily software was developed to test the VIC-20 Hyper Expander Cartridge (see http://github.com/svenpetersen1965). Nevertheless, it can be used to test other RAM expansions for the VIC-20 as well.

The VIC-20 expansion port provides a number of chip-select signals, that correspond with certain address ranges.

Signal	Address Range	Notes
$\overline{\text{RAM1}}$	$0400 - $07FF	3k RAM Expansion is tested as one block
$\overline{\text{RAM2}}$	$0800 - $0BFF	
$\overline{\text{RAM3}}$	$0C00 - $0FFF	
$\overline{\text{BLK1}}$	$2000 - $3FFF	1st 8k Block
$\overline{\text{BLK2}}$	$4000 - $5FFF	2nd 8k Block
$\overline{\text{BLK3}}$	$6000 - $7FFF	3rd 8k Block
$\overline{\text{BLK5}}$	$A000 - $BFFF	4th 8k Block/EPROM
$\overline{\text{I/O2}}$	$9800 - $9BFF	1st 1k Block
$\overline{\text{I/O3}}$	$9C00 - $9FFF	2nd 1k Block

The 3k RAM Expansion is treated as one block from $0400 - $0FFF.

Possible RAM blocks at $\overline{\text{I/O2}}$ and $\overline{\text{I/O3}}$ are now supported with this test software (v1.0).

There are two different versions of this software:

- A binary for programming an auto-start cartridge ROM at $A000
- A loadable version for the internal RAM as a .d64 disk image

Test Procedures

RAM Block Detection

First, the RAM blocks are detected. This is done by writing $00 to the first 256 bytes (first page) of the memory block to be detected. These 256 Bytes are then read back. The memory block is detected, in case all 256 Bytes are $00.

This test is conducted with every of the listed RAM blocks (except those connected to the I/O chip-selects).

Full RAM Block Test

All positively detected RAM blocks will be fully tested. The test consist of three parts:

1. Bit test by writing $55 into the complete RAM Block and reading it back. All bytes need to be $55
2. Bit test by writing $AA into the complete RAM Block and reading it back. All bytes need to be $AA
3. Pseudo Random Number (PRN) test by writing a sequence of PRNs into a complete RAM block and reading back. This procedure detects problems with the addressing of the RAM.

The first two tests will detect problems with single bits in the RAM. The 2nd value is the inverted value of the first test, so every bit will be tested for 0 and 1.

The pseudo random number test (3.) is generating a fix sequence of numbers. It is not really random. But the periodicity is odd and not a multiple of any power of two. This way, stuck address lines can be detected.

The polynome for generating the pseudo random numbers is

$$x^8 + x^5 + x^4 + 1$$

This is the same polynome used for generating the Dallas 1wire protocol CRC-8 code.

The polynome is applied as follows:

The byte is shift right, the LSB is shifted into the carry flag. In case carry is set, the polynome, represented by the bit pattern %10001100, which is XORed to the shift byte, in case the carry flag was set. A quite simple procedure. Initially the byte is $01.

```
PRNPoly     =%10001100        ; polynome for pseudo ramndom number generation (X^8 + X^5 +
X^4 + 1)
PRNSeed     =$01              ; seed value for pseudo random numbers
[...]

nextPRN     txa                         ; value in .x -> .a
            ror                         ; rotate right, LSB in carry flag
            bcc nextPRN1                ; branch if carry clear
            eor #PRNPoly                ; if carry set, apply polynome
                                        ; = invert the bits, that are contained in polynome
nextPRN1    tax                         ; save result in .x
            rts
```

```
; initialize the PRN generation ====================
initPRN     ldx #PRNSeed            ; seed value -> .x
            clc                     ; clear carry
            rts
```

Memory dump of the first 256 bytes after writing a sequence of pseudo random numbers:

```
>C:0400  8c c6 63 bd  52 a9 d8 ec  76 3b 91 44  a2 51 a4 d2  69 b8 dc 6e  37 97 47 2f
>C:0418  1b 01 0c 86  43 ad 5a ad  da ed fa fd  f2 f9 f0 f8  7c 3e 1f 83  4d 2a 95 c6
>C:0430  e3 fd 72 b9  d0 e8 74 3a  1d 82 c1 ec  f6 7b b1 54  aa 55 a6 d3  e5 7e bf d3
>C:0448  65 3e 9f c3  6d 3a 9d c2  e1 fc fe 7f  b3 55 26 93  c5 6e b7 d7  67 3f 13 05
>C:0460  0e 87 cf 6b  39 10 88 44  22 11 84 c2  61 bc de 6f  bb 51 24 92  49 a8 d4 6a
>C:0478  35 96 cb e9  78 bc 5e 2f  9b 41 2c 96  4b a9 58 ac  56 2b 99 40  a0 50 28 14
>C:0490  0a 05 8e c7  ef 7b 31 14  8a 45 ae d7  e7 7f 33 15  06 83 cd 6a  b5 d6 eb f9
>C:04a8  70 b8 5c 2e  17 87 4f 2b  19 00 80 40  20 10 08 04  02 01 8c c6  63 bd 52 a9
>C:04c0  d8 ec 76 3b  91 44 a2 51  a4 d2 69 b8  dc 6e 37 97  47 2f 1b 01  0c 86 43 ad
>C:04d8  5a ad da ed  fa fd f2 f9  f0 f8 7c 3e  1f 83 4d 2a  95 c6 e3 fd  72 b9 d0 e8
>C:04f0  74 3a 1d 82  c1 ec f6 7b  b1 54 aa 55  a6 d3 e5 7e
```

Looking at the memory dump above, it is visible, that the sequence repeats after $BA (=186) bytes. The begin of the 2nd sequence is marked.

Cross Link Test

It is reported, that some RAM Expansions can be cross linked. That means, writing to one RAM block might corrupt another RAM block.

This is also tested by this software. Before testing one block like described in the previous chapter, the all RAM blocks are filled with $00. Then, the three tests are conducted with one RAM block and finally, all other RAM are read back. In case a byte is found, that is not $00 anymore, a cross link failure of this block is detected.

Test Display

The results of the test are displayed in a table:

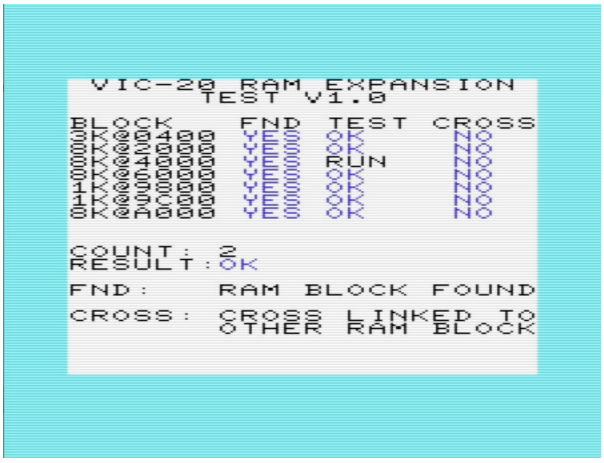

Figure 1: Display of the running test

The tested RAM blocks depend on the version of the test program. Since an auto-start cartridge resides at the addresses $A000 and following, this RAM-Block is not detected by the ROM software and thus, it is not tested. The software version shown in Figure 1 is the loadable version from the .d64 disk image.

- "**BLOCK**" enumerates the tested RAM blocks.
- "**FND**" states, if this ram block was detected or not.
- "**TEST**" shows if a test is performed ("RUN") and if the last result is "OK" or "BAD".
- "**CROSS**" shows, if a RAM block was corrupted dur to cross linking.

A single result may change from "BAD" to "OK", in case the last test was performing ok. The old "BAD" results are sticky, though. An OK, that was BAD once before is displayed in red (other than blue, which shows, that s test has never resulted in "BAD" before).

The line "**RESULT**" shows the final verdict. A once tested BAD or crosslink block will turn the result from OK to BAD. This will not change, even if all block have been tested good ever after.

The "**COUNT**" is a 16bit value. After 65535 has been reached it will revolve to 0 again.

Versions of The Software

Software Sources

The software was written with **C64 Studio** (https://www.georg-rottensteiner.de/de/index.html) It consists of three source files.

- overlay.asm
- ramtest_a000.asm
- vic20ramtest.bas

overlay.asm contains the complete test. If this is assembled with the compiler flag ROM not defined, the loadable machine code program "overlay.prg" is the result of the build.

ramtest_a000.asm includes the files overlay.asm and sets the compiler flag ROM so, that the output is ramtest_a000.prg, which contains two bytes of start address and the binary for the auto-start cartridge. To get a *.bin file for an EPROM programmer, the two bytes start address have to be removed with a HEX editor (e.g. HxD).

vic20ramtest.bas is a short basic program, which is loading the overlay.prg and executing it (address $1300 → SYS 4864). Loading such an overlay machine code program is not trivial (see the source code).

This detour is required, because the start of the VIC-20 BASIC RAM varies depending on the installed RAM expansions.

EPROM Version

The EPROM version contains a header for an A000 auto-start cartridge, which is 2 bytes software start address, two bytes NMI routine address (that is $FEAD, the default address) and the signature "A0CBM" (`$41, $30, $c3, $c2, $cd`).

The resulting file „ramtest_a000.prg" needs to be modified, like mentioned before, to get a binary file, that can be programmed.

Loadable Disk Version

The disk image (VIC20RAMTest.d64) contains two files:

- The boot laoder “vic20ramtest.prg”
- The actual test software “overlay.prg”

The boot loaded should be imported to the disk image first, so it will be loaded with LOAD”*”,8

The overlay.prg is imported 2nd.

The software is executed with

LOAD”*”,8 (not ,8,1 !!!)

RUN

Loading the software with ,8,1 might not execute the software properly, since the basic start addresses can vary in a VIC-20.

Testing the Testsoftware

The test software was tested with VICE (xvic.exe). The Monitor (ALT+H) was used to set break points to display RAM content and to modify it.

This way, the RAM was corrupted manually before the test routine was checking the content.

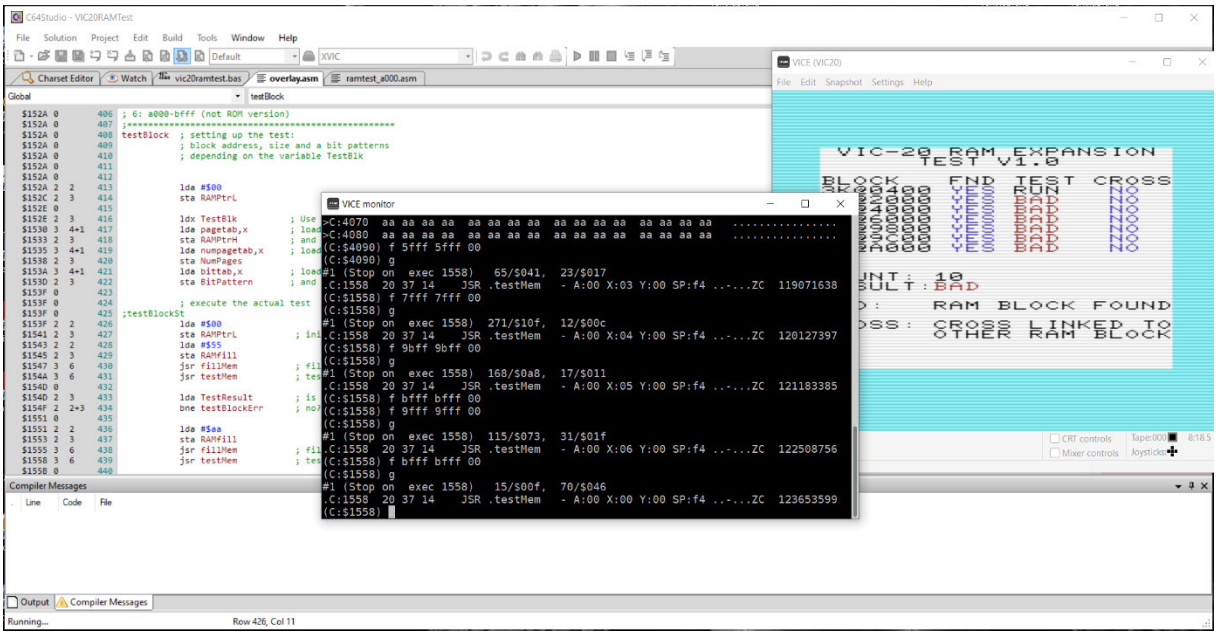

Figure 2: Manually corrupted RAM content was detected

Every single test routine was able to test the RAM content in every possible block. Different RAM configurations were set in the Settings →VIC-20 settings… menu and tested.

The corruptions in tests and cross-linked blocks were detected and the result was displayed properly.

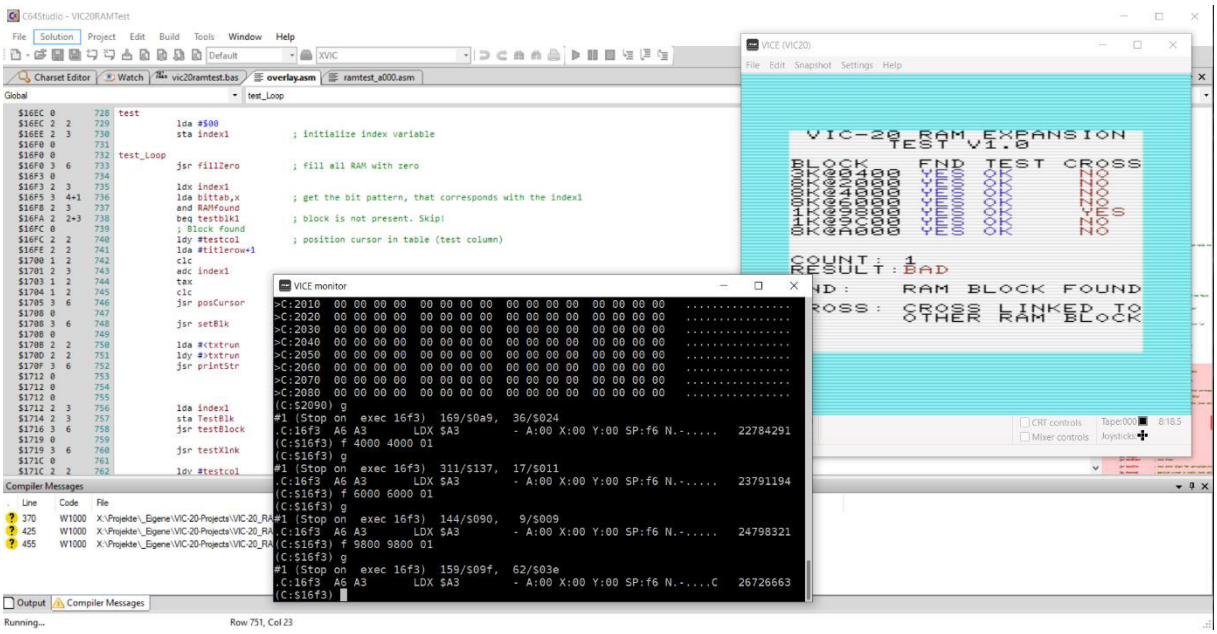

Figure 3: Simulating cross-linked blocks

Finally, an EPROM was burned and tested in a VIC-20 Hyper Expander cartridge (this is a new VIC-20 project, which can be found in my github repositories mentioned here: Purpose of the Software.

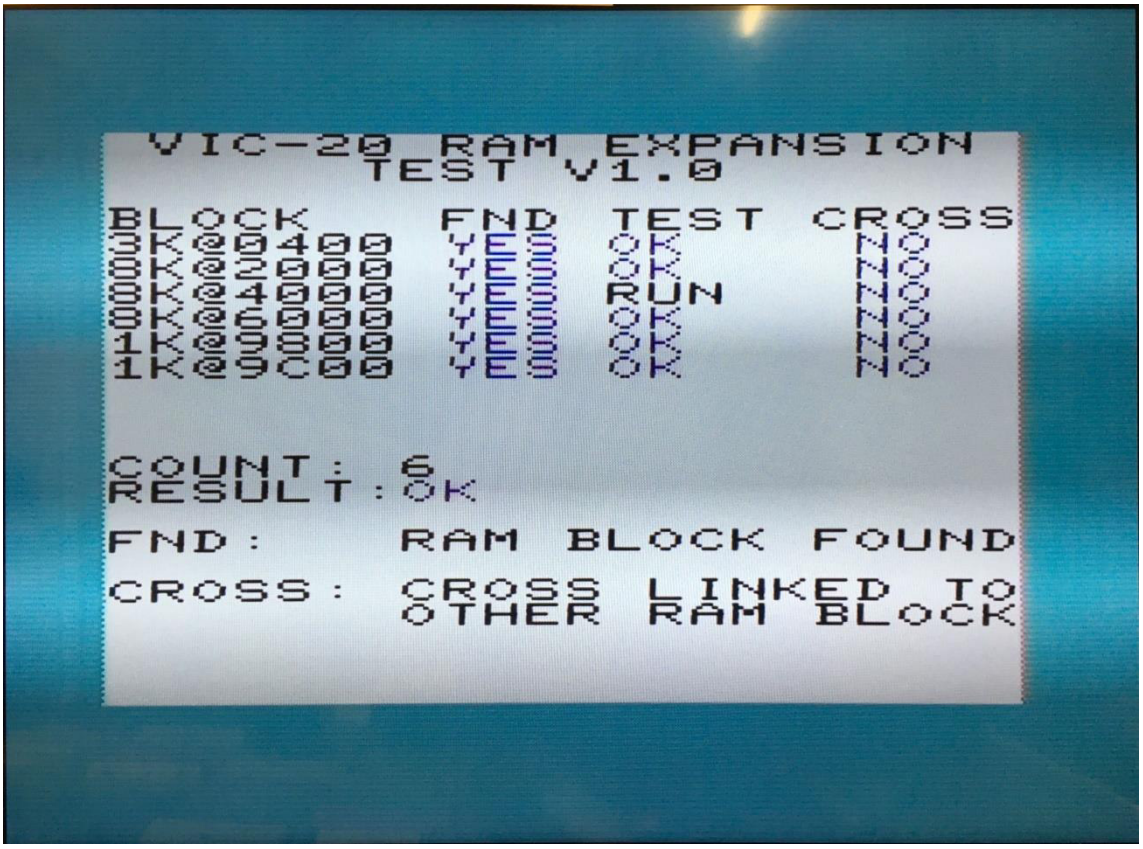

Figure 4: Running on real hardware from EPROM

All RAM blocks were detected and tested successfully.

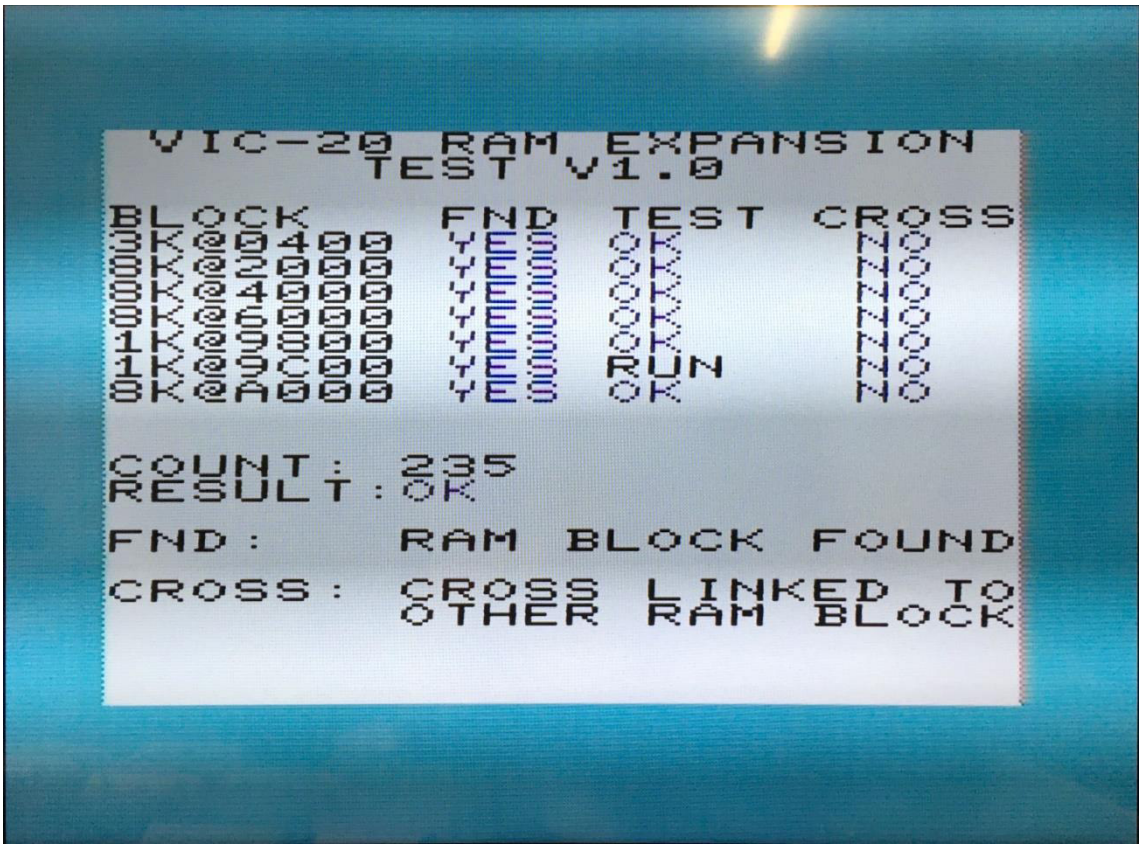

Figure 5: Running on real hardware from floppy disk

The loadable version was tested on real hardware with a Hyper Expander cartridge without a ROM (so RAM at $A000 could be tested as well).

Conclusion:

The software version 1.0 is fully functional.