VBXE DOCTOR

v1.5 User Manual

Tomasz Piórek / electron · 1 November 2026

Table of contents

Overview 2

Requirements 3

Memory used 4

Using the program 5

Test selection 7

Screen 9

Error log 10

Detail views 11

Verdicts 13

Examination time 14

Tests 15

Group A - 6502<->IO registers 15

Group B - 6502<->VRAM 16

Memory guard 17

Group C - Blitter<->VRAM 18

Test times 18

Overview

> **WARNING**
> Please note that this is an experimental tool; it may contain errors and exhibit unpredictable behavior. Its results can be affected by many factors related to the configuration and condition of the computer under test. It should not be used to draw conclusions about the condition of the hardware itself.

VBXE DOCTOR (`vdoctor.xex`) is a diagnostic program for VBXE with FX or GTIA core. It runs three groups of tests over and over until you leave it:

Code	Group	What is checked
A	**6502<->IO REGISTERS**	core registers (`$D640-$D67F` or `$D740-$D77F`)
B	**6502<->VRAM**	link between 6502 and VRAM without Blitter: all 512 KB through MEMAC-A and MEMAC-B windows; MEMAC-A windows of all sizes up to the end of the address space ($FFFF); ANTIC / CPU access of the windows
C	**BLITTER<->VRAM**	Blitter access to VRAM: BCBs in VRAM, blits started, results compared with values computed by CPU; also CPU access to VRAM while Blitter works

One run through all selected tests is a **pass**; the next pass starts at once. Random addresses and data change every pass, and passes 2, 3, 6, 7, ... verify memory from top down, so a longer run covers more combinations. All tests are selected at start; the key INVERSE opens the test selection.

Every error is counted. Within the examination time of its test the program also decides what most likely failed - write or read - and counts errors per group and per verdict.

Requirements

- VBXE with **FX core v1.26 or newer v1.2x** (VERSION $26-$2F, with or without RAMBO), or **GTIA core** of the same versions, at $D640 or $D740 (found automatically). With GTIA core only group A is available; groups B and C show N/A.
- Atari XL/XE with 64 KB or more: while tests run, OS ROM and BASIC are off and part of the program runs in RAM under OS ROM.
- CPU 6502 or 65816 (6502 emulation mode, documented 6502 instructions only). Waits and time limits are counted in video frames and scan lines, so a faster CPU does not change them. With a CPU accelerator, all accesses to $4000-$FFFF must reach the computer's bus (the MEMAC windows lie there).
- A cartridge is not needed; with one inserted, the parts of tests B04, B05 and B06 over $A000-$BFFF are skipped.

Without a suitable core or without RAM under OS ROM the program prints a message and returns to DOS.

> NOTE
>
> **For real hardware.** Under an emulator, tests A03 (Mirror) and A08 (Register load) can report errors although emulation is correct: they check details of the real core that are undocumented or deliberately left unspecified (the register mirror, unused and write-only addresses) that programs do not use. Such errors repeat in every round and can give group A a high `FAIL%`. Groups B and C under an emulator show only how it models VRAM and Blitter.

> CAUTION
>
> **SUBcart with its 1 MB PORTB memory extension enabled is not compatible with VBXE** (SUBcart firmware v012 and v013 beta 1). In this mode SUBcart puts its own memory in place of the computer's RAM at $A000-$BFFF in every CPU access (it pulls the REF line low, so the computer disables its memory there), so MEMAC windows over $A000-$BFFF do not work. The program then shows errors of B04, B05 and B06 at $A000-$BFFF only, all with `M` in column `S` of the log, while the memory guard finds no error (the memory of SUBcart is stable RAM). This is a limitation of SUBcart, not a VBXE malfunction: VBXE gives the bus to the device that took it, exactly as it does for ROM and cartridges. The smaller PORTB modes of SUBcart (128, 192, 320 and 576 KB), and SUBcart with the extension off, work with VBXE.

Memory used

Area	Use
$0400 - $05FF	C stack (while loading: short routines at $0480 that put code under OS ROM)
$0600 - $06FF	pattern table
$2000 - $7FBF	program and data
$4000 - $7FFF	also MEMAC-B window, on only for short moments
$7000 - $7FFF	also MEMAC-A window of B12, a few frames per pass
$7FC0 - $7FE7	display list of the program's screen ($7FF0: a byte used by A07)
$8000 - $8FFF	MEMAC-A window (4 KB) in groups B and C; free RAM checked by the memory guard while group A runs; while loading: code for the RAM under OS ROM before it is moved
$8000 - $FFFF	MEMAC-A windows of other sizes and bases (B04, B05, B06), only for short moments with the NMI disabled
$9000 - $9FFF	texts, screen, group statistics, copy of character set
$A000 - $BFFF	free RAM checked by the memory guard all the time
$C000 - $CFFF	screen code (RAM under OS ROM)
$D800 - $DEFF	test statistics (RAM under OS ROM)
$DF00 - $FEFF	more program code: test selection, detail views, test B05 (RAM under OS ROM)
$FFFA - $FFFF	6502 vectors (NMI, RESET, IRQ) while tests run; $FF00-$FFF9 (with the 65816 vectors) is not changed
VRAM	all 512 KB overwritten

IRQ is disabled while tests run; only the vertical blank interrupt (NMI) is used, also for the keyboard and the memory guard, and in B12 the display list interrupt (DLI).

> **CAUTION**
>
> On cores with RAMBO emulation ("r" cores) upper 256 KB of VRAM is extended memory of the computer, so a RAM disk kept there is lost too.

Using the program

Start `vdoctor.xex` from DOS. The program shows its own 29-line screen and starts all tests; with **INVERSE** held down while it starts, it opens the test selection first. Tests run pass after pass; the screen follows the running group (see Screen), and numbers change at once on the first error of a test in a pass, then about once a second.

The screens form levels: **main screen** → RETURN → **error log** → CAPS → **detail views** → CAPS → back to the log. In the log of group B the detail views are B04 → B05 → memory guard; in the log of group A or C only the memory guard. The test selection (INVERSE) lies over all of them.

Key	Main screen	Error log	Detail views
SPACE	pause / run again	pause / run again	pause / run again
TAB, cursor key right	next group	log of the next group	log of the next group
cursor key left	previous group	log of the previous group	log of the previous group
A, **B**, **C**	that group	log of that group	log of that group (B04 / B05: B does nothing)
CAPS	results in % or numbers of errors (`ERR`)	group B: detail view B04; A, C: memory guard	B04 → B05 → memory guard → back to the log
RETURN	(after the first error) error log on	log off	log off
INVERSE	test selection	test selection	test selection
ESC	exit (asks first)	back to the main screen	back to the log

- The cursor keys left and right are **+** and ***, with or without CTRL. SHIFT and CTRL do not change any key. On the Atari 400/800 INVERSE and CAPS are the Atari key (logo) and CAPS/LOWR.
- A group chosen with the group keys stays on the screen until it has run from its first to its last test; then the screen follows the running group again. While the log is on, the group shown stays.
- **SPACE** pauses the tests at once: `PAUSED` in the status line, the busy indicator stands still, the picture has its usual colours. All other keys work; SPACE again goes on. The pause does not count in the time of the running test.
- **ESC** on the main screen asks first (`ESC AGAIN: EXIT, OTHER KEY: CANCEL`). ESC again returns to DOS at once, also in the middle of a test, with the original screen and colours; any other key cancels.
- The keyboard is read once per video frame; a key takes effect when the running test next lets the program look at it, as a rule within a fraction of a second.

The key line shows the keys that work at the moment, with ESC always at its end:

```
SPC TAB:GRP CAPS:%/ERR INV RET:LOG ESC     main screen (RET:LOG after the first error)
SPC TAB:GRP CAPS:MEM INV RET:LOG   ESC     log of group A or C
SPC TAB:GRP CAPS:DET INV RET:LOG   ESC     log of group B
SPC TAB:GRP CAPS:B05 INV           ESC     detail view B04
SPC TAB:GRP CAPS:MEM INV           ESC     detail view B05
SPC TAB:GRP CAPS:LOG INV           ESC     detail view of the memory guard
```

At start the program loads VBXE palettes 1-3 with monochrome scales in the style of CRT monitors: 1 grey, 2 green, 3 amber (hue ignored, luminance in 16 steps of the PAL grey scale). Tests A06 and A08 write random values to GTIA_PALETTE, so while they run the picture changes between normal colours and these scales but stays readable. Palette 0 is not changed. On exit VBXE gets a soft reset, and MEMAC registers and FEATURE_CONTROL (v1.27 cores) get back their values from program start; palettes 1-3 keep the scales, and settings made only through VIDEO_CONTROL do not survive the soft reset.

Test selection

Key **INVERSE** opens a window with the tests of one group at a time (`[X]` selected, `[ ]` not selected, `[-]` not available on this core) and the number of selected tests of each group in the top line; it opens on the group shown on the screen. The tests wait while the window is shown (`PAUSED` in the status line), and this time does not count for the running test.

```
TESTS TO RUN
 A 08/08     B 11/12     C 09/09
──────────────────────────────────────
[-] GROUP B: 6502<->VRAM
  [X] B01 DATA LINES
  [X] B02 ADDRESS LINES
  [X] B03 FULL VRAM, ADDRESS PATTERN
  [ ] B04 MEMAC-A WINDOWS
  ...
  [X] B12 ACCESS DURING DISPLAY
──────────────────────────────────────
-,=:    MOVE      +,*,TAB: OTHER GROUP
SPACE:  SELECT    A,B,C:   SHOW GROUP
RETURN: ACCEPT    ESC,INV: CANCEL

S:      AUTO-STOP AFTER EACH PASS: OFF
M:      MEMORY GUARD:              ON
```

The line under the cursor is in inverse video; the first line of a group stands for the whole group (`[X]` all its tests selected, `[-]` some, `[ ]` none).

Key	Action
- and **=** (the cursor keys up and down, with or without CTRL)	move the cursor up / down; after the last line it goes to the first
+ and ***** (the cursor keys left and right), **TAB**, **A**, **B**, **C**	previous / next group, or that group, as on the main screen
SPACE	select / deselect the test under the cursor; on the first line of a group: select the whole group, or deselect it when all its tests are selected
S	auto-stop after each pass on / off (default off)
M	memory guard on / off (default on)
RETURN	accept the selection and start again from the beginning
ESC or **INVERSE**	close the window with the previous selection and settings; the tests go on where they waited

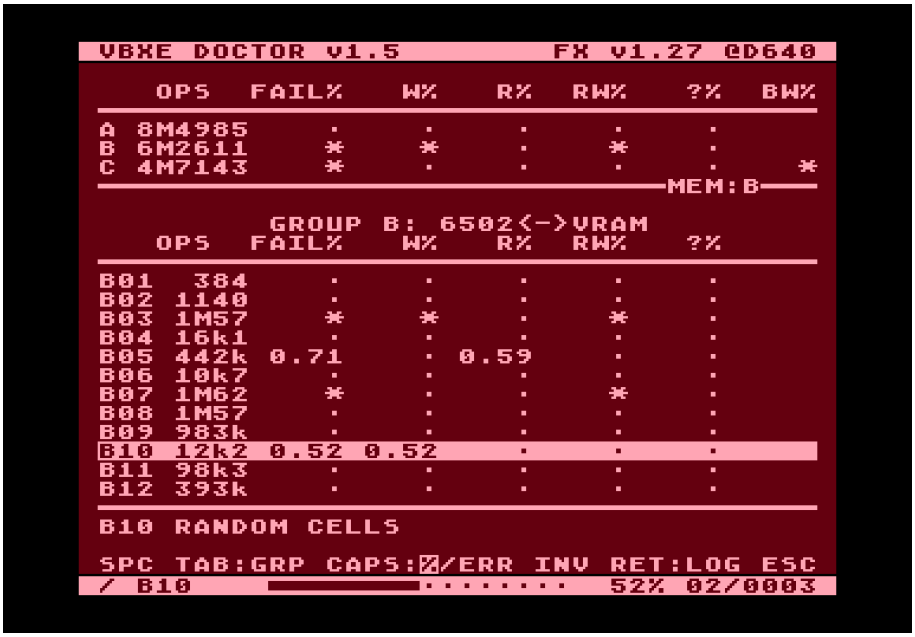
- **RETURN** starts the program again as after loading, also when nothing was changed: the running test is abandoned, and all results, the error log, the detail view statistics, the pass number and the screen colour are cleared. With no test selected, a message above the key help asks for at least one.
- With **auto-stop** on, the tests stop after each pass: the results stay on the screen, the status line shows `DONE`, all keys work as in a pause, and SPACE starts the next pass. Both options are kept when the tests start again.
- When the window was opened by INVERSE held down at the start, ESC or INVERSE close it and the tests start with the selection unchanged.

Tests that are not selected are left out of every pass and show `OFF` (a group without a selected test too); the progress bar counts the selected tests only. Tests that do not exist on the core (for example A04 and A05 on GTIA core, A06 on v1.26 cores) cannot be selected and show `N/A`.

A few tests use data written by another one:

- **B04-B07** use VRAM written by B03 (B05 and B06 leave it as it is for B07). When B03 is not selected, B04 and B05 write the blocks they read themselves.
- **C02-C09** use the BlitterList area and the source area written by C01. When C01 is not selected, these areas are written in the same way but not verified.
- The ANTIC part of **B12** checks its own windows first (see B12); errors of other tests do not stop it.

Screen



In inverse video: the title line, the status line, the line of the running test, and in the key line `SPC` while paused, the CAPS mode in use (`%` or `ERR`) and `RET:LOG` while the log is on. Screen colour: blue during the first pass; green when the whole first pass found no error; red from the first error on (also after green), until the tests start again or the program ends.

Line	Meaning
title	program version, core type and version (`r` = RAMBO core), VBXE register address
groups A, B, C	per group: `OPS` , `FAIL%` and one column per verdict, for all its tests; `N/A` when the group is not available on this core, `OFF` when none of its tests is selected
line below the groups	`MEM:` with `S` when the start test of the memory guard found an error, and the letters of the groups during which it found one (e.g. `MEM:SAB`); it stays until the tests start again
group shown	its name, then the same columns for each of its tests (`BW` in group C only)
running test	test that runs now, with its name
`512K FILL`	group C only, below the running test after an empty line: time Blitter needed to fill 512 KB in the last pass (information only, never an error), and `TIMEOUTS` - how many times Blitter did not finish (verdict T), shown only when not 0
keys	keys that can be used now (see Using the program)
status	busy indicator, running test (`PAUSED` while the tests wait, `DONE` after a pass with auto-stop), progress of the pass, second of the running test (00-99) and pass number (4 digits; after 9999 it goes on from 0000)

`OPS` counts checked operations since the tests started: each read compared with its expected value (in B06, each write checked in VRAM and RAM). Group lines show up to 6 characters: up to 999999 as is, then `M` (millions) or `G` (thousand millions) in place of the decimal point, e.g. `7M8123` = about 7 812 300. Test lines show 4 characters, again with a letter in place of the decimal point (`1140` , `49k1` , `393k` , `6M29` , `53M1` , `531M` , `4G29`).

By default (`%` in the key line) results are percentages of `OPS`: `FAIL%` - share of checked operations that found an error (T not counted); each verdict column - share of examined errors with that verdict. Errors found after the test's examination time are counted in `FAIL%` only, so the verdict columns add up to `FAIL%` unless the log line of the test has `+` in column `S`.

Shown	Meaning
`.` (dot in the middle of the field)	no error at all
`*`	more than 0, but less than 0.01 % (a single error is never shown as `.`)
`0.01` to `9.99`, `10.0` to `99.9`, `100`	share in %

With CAPS (`ERR`) the same columns show numbers of errors: up to 9999 as is, larger ones shortened the same way (`10k1`, `923k`, `5M31`); middle dot for 0. An empty field is a verdict that does not exist in that group (BW in groups A and B).

Groups A and B make about the same number of accesses (reads and writes) per pass, so their results can be compared directly. Their `OPS` differ even so, as `OPS` counts checked operations only: a whole pass without errors gives about A : B : C = 10 : 8 : 5.5. While a pass runs, the groups that have already run in it count one pass more than the others, so the ratio on the screen changes during a pass.

Error log

While the log is on (RETURN after the first error), each test line shows the first error of the test in the last pass in which it found one, e.g.:

```
     PASS  ERR V  ADDR    EXP GOT XOR S
  B03 0003 1k23 W  04C1A7   61  71  10 +
  B04 0003   12 R  900D/16  26  44  62
  B09 0003    3 W  010040:2 FF  EF  10
```

Column	Meaning
(test)	group letter and test number
`PASS`	pass number
`ERR`	number of errors of the test in that pass, shortened like `OPS` (`1k23`, `53k1`)
`V`	verdict of the first error
`ADDR`	address of the first error: VRAM address (6 digits, 19 bits) or register address (4 digits, `D650`), with extra details in some tests (below)
`EXP`	expected value
`GOT`	value read
`XOR`	the bits that differ (`EXP` XOR `GOT`): `04` = bit 2, `10` = bit 4
`S`	`M`: the RAM under the window answered or was written (B04, B05, B06, B12, see below); `+`: some errors of the test in that pass came after its examination time and have no verdict (`M` is shown in place of `+` when both apply)

Test	`ADDR`	Meaning
B04, B05, B06	`900D/16`	CPU address and size of the MEMAC-A window in KB (4, 8, 16, 32) instead of the VRAM address
A07	`8123/4`, `7FF0/16`	RAM address that a window gave after the soft reset: MEMAC-A (4 KB) or MEMAC-B (16 KB); `EXP` = the RAM byte, or the byte written there after the reset
B07	`020040:A`, `020040:B`	path that gave `GOT`: A = MEMAC-A, B = MEMAC-B (B with verdict R: MEMAC-A read the right value, MEMAC-B another one)
B09	`010040:1` … `:5`	March C- element: 1 read 0 write 1 upwards, 2 read 1 write 0 upwards, 3 read 0 write 1 downwards, 4 read 1 write 0 downwards, 5 final read 0
B12	`07DFC0:1` … `:4`	step of the ANTIC part: 1 MEMAC-A for ANTIC only, 2 MEMAC-A for CPU only, 3 MEMAC-B for ANTIC only, 4 MEMAC-B for CPU only. The address is in the VRAM copy of the display list (`07DFC0` MEMAC-A, `07BFC0` MEMAC-B); `EXP` / `GOT`: the number of DLIs (`06` or `00`, at the start of the copy), the VCOUNT of a DLI (`11`, `3D`, `6D`, at its line), or the byte the CPU read (a display list byte, at its place; with `M`: the RAM got the write)

`M` in column `S`:

- **B04, B05**: the value read through the window equals the byte of the RAM under it at that CPU address (the memory guard pattern at $8000-$8FFF and $A000-$BFFF, the RAM contents elsewhere). The window did not take the bus and the RAM answered. A wrong value that does not match the RAM (for example always $00) comes from somewhere else.
- **B06**: the write through the window reached the RAM under it. With a wrong value in VRAM the line shows VRAM (`EXP` = value written, `GOT` = VRAM); when VRAM got the right value and the RAM got it too, verdict W and the RAM byte (`EXP` = the old RAM byte, `GOT` = RAM).
- **B12**: a write of the CPU step reached the RAM display list instead of the window (`EXP` = the RAM byte, `GOT` = the same byte with the DLI bit).

A line changes at most once per pass, at the first new error, and stays as is in passes without errors; `ERR` counts the errors of that pass. A test that has run through at least once without an error shows `OK`; one that has not yet run through shows its number only.

Detail views

The detail views (CAPS in the error log, order above) are updated once a second while the tests keep running. RETURN: log off.

> **NOTE**
>
> **4K page** in this manual and on the screen means a 4 KB block of the CPU address space, $x000-$xFFF: `$8` is $8000-$8FFF, `$F` is $F000-$FFFF. MEMAC-A windows are placed in these units: a window starts at a 4K page and covers 1, 2, 4 or 8 of them (4, 8, 16 or 32 KB). A page without "4K" is the 256-byte page of the 6502 (as in zero page, $0000-$00FF).

B04 and B05

```
B04 DETAIL: MEMAC-A WINDOWS

ERRORS BY WINDOW AND 4K PAGE
WINDOW  $8  $9  $A  $B  $C  $D  $E  $F
$9  4K   -   .   -   -   -   -   -   -
$B  4K   -   -   -   .   -   -   -   -
$F  4K   -   -   -   -   -   -   - 128
$8  8K   .   .   -   -   -   -   -   -
$B  8K   -   -   -   .   .   -   -   -
$E  8K   -   -   -   -   -   - 128 128
$F  8K   -   -   -   -   -   -   - 128
$8 16K   .   .   .   .   -   -   -   -
$9 16K   -   .   .   .   .   -   -   -
$C 16K   -   -   -   - 128 128 128 128
$E 16K   -   -   -   -   -   - 128 128
$8 32K 128 128 128 128 128 128 128 128
$A 32K   -   - 128 128 128 128 128 128
$C 32K   -   -   -   - 128 128 128 128

VALUES READ, MOST FREQUENT FIRST
        GOT  ERRORS       XOR  ERRORS
1ST      00    3584        6C      27
2ND       -                1B      24
3RD       -                D7      21
4TH       -                 -
MORE      -                 +
```

```
B05 DETAIL: WINDOW STRESS

ERRORS BY WINDOW AND 4K PAGE
WINDOW  $8  $9  $A  $B  $C  $D  $E  $F
$9  4K   -   .   -   -   -   -   -   -
$A  4K   -   -  55   -   -   -   -   -
$B  4K   -   -   -   .   -   -   -   -
$8  8K   .   .   -   -   -   -   -   -
$A  8K   -   -   .   .   -   -   -   -
$8 16K   .   .   .   .   -   -   -   -

ERRORS BY SECTION OF THE BURST (1/16)
SECTION      +0    +1    +2    +3
 0            3     7     4     5
 4            4     5     4     5
 8            4     5     4     5
12            .     .     .     .

VALUES READ, MOST FREQUENT FIRST
        GOT  ERRORS       XOR  ERRORS
1ST      00      41        04      22
2ND      FF       3        10      11
3RD       -                14       8
4TH       -                20       2
MORE      -                 +
```

- **Errors by window and 4K page**: a row for each window setting (`$8 32K`: base 4K page $8000, size 32 KB), a column for each 4K page of the address read. A dash marks a 4K page the window does not cover (beyond its size, or beyond $FFFF). In the B04 example every window that reaches $FFFF fails at all of its 4K pages and the others show no error; in the

B05 example the errors come only from reads at $A000-$AFFF through the 4 KB window at $A000.

- **Errors by section of the burst** (B05 only): the 512 bytes of a burst in 16 sections of 32 bytes; row `4`, column `+2` is section 6.
- **Values read**: the values most often read by the failing reads (`GOT`) and the bits most often wrong (`XOR`), each with its number of errors (shortened like `OPS`), up to four, the most frequent first. `-` marks an empty place. `+` in the line `MORE` means that more than four different values came: each new value then takes one from every number, so the numbers shown are lower than the true ones and a rare value can leave its place empty. The same `GOT` at every error (for example always `00` or `FF`) points to something else that answers in place of VRAM; the same `XOR` at different `GOT` values points to particular data bits.

The first two tables count every error, also those after the examination time; the values table of B05 counts only the examined errors (up to 6 of each burst). Counts stop at 65535; a middle dot means no error.

Memory guard view

```
MEMORY GUARD

$8000-$8FFF  RAM, CHECKED IN GROUP A
$A000-$BFFF  RAM, CHECKED ALL THE TIME

FIRST ERROR, LAST PASS WITH ERRORS
S: START TEST  A-C: GROUP RUNNING
    PASS  ERR V  ADDR    EXP GOT XOR S
──────────────────────────────────────
S   STRT    1    A123     27  37  10
A     OK
B   0004    2    A3F1/B05 08  00  08
C     OK
```

- The first two lines give the state of the areas: RAM and checked (in group A only, or all the time), `NOT RAM, NOT CHECKED` (for example a cartridge at $A000-$BFFF), or `RAM, GUARD OFF`.
- Then, in the columns of the error log, the first error the guard found in the last pass with errors: line `S` by the start test (`STRT` in place of the pass), lines `A`-`C` while that group ran. `ADDR` is the RAM address and the test that was running; no verdict. `OK`: no error; `OFF`: a group without a selected test; `N/A`: a group not available on this core.

Verdicts

Each error is examined right after it is found, within the examination time of the test. The location is read 4097 times in a row through the first path (VRAM: MEMAC-A window; registers: the register itself), which takes a few frames; a VRAM byte is then read 257 times through a second path (MEMAC-B window). A register has no second path (it is never read through its mirror at $Dx60-$Dx7F except in A03), so for registers `?` does not occur. For each path the program notes whether all reads were the same and how often the expected value was read; the read that found the error counts too. A few errors get verdict R without this examination: A03 (one register, two answers), the windows part of A07, B07 (MEMAC-A right, MEMAC-B wrong) and the ANTIC part of B12.

CPU access (all groups)

W, R, RW and ? describe how **CPU** writes and reads the location: a register in group A, a VRAM byte in groups B and C.

Verdict	Reads	Meaning
W	both paths stable, same wrong value	Write fails: wrong value stored.
R	expected value in at least 10% of reads of one path	Read fails: stored value is right, but reading it back gives wrong values.
RW	unstable, expected value in less than 10% of reads of each path	Most likely write and read fail: stored value is wrong, and reading is unstable too. Not certain: a right value read wrongly in over 90% of reads, or a cell that loses its contents between reads, gives the same result.
?	both paths stable, two different wrong values	Error could not be classified.

The 10% limit keeps random read errors from looking like a right stored value: random data hits the expected value in only about 1 of 256 reads.

> NOTE
> If a part shared by both read paths is damaged (e.g. a data line between FPGA and VRAM chip), a permanent read error looks exactly like a write error and is shown as W.

Blitter (group C)

Before a blit **CPU writes** the source area, the destination area and the BCBs and **reads them back**; an error there gets a CPU verdict and the blit is not run. **Blitter** then writes the destination area, and **CPU reads** each byte of it (through MEMAC-A and MEMAC-B) and compares it with the value it computed. A wrong byte is examined like any other location; when both paths show the same wrong value, CPU writes the expected value there and reads it back once through each path: if this works, VRAM works there and Blitter is to blame (BW); if not, it is a CPU write error (W). Other cases get CPU verdicts too.

Verdict	Meaning
BW	Blitter wrote a wrong value: CPU reads the same wrong value through both paths, and can write and read back the right value at that address.
T	Blitter did not finish within 150 video frames (3 s PAL, 2.5 s NTSC; BLITTER_BUSY did not return to 0); Blitter is stopped. Counted as `TIMEOUTS` below the table.

Examination time

Errors do not stop a test: every error is counted in `FAIL`, so `OPS` and `FAIL%` cover the whole pass also when much is wrong. Examining one error takes about 1/30 s (register) to 1/20 s (VRAM), so each test has an examination time per pass (see Test times) - enough for about 1000 verdicts, which gives the shares of W, R and RW within about 3 points. Further errors of the test in that pass are counted in `FAIL` only, with no verdict, so no verdict is ever guessed; the first error of a test in a pass is always examined. `+` in column `S` of the log marks such a pass; in `ERR` mode their number is `FAIL` minus the verdict columns. Without errors examination costs nothing.

Tests

Group A - 6502<->IO registers

Test	Description
A01	**Core identification.** CORE_TYPE and CORE_REVISION read 8192 times; every read must return the values found at start.
A02	**Read stability.** All registers with a defined read value are read 2048 times each and must always return the same value: COLDETECT, BLT_COLLISION_CODE, BLITTER_BUSY (must be 0), IRQ_STATUS (must be 0), MEMAC_CONTROL and MEMAC_BANK_SEL (FX core), FEATURE_CONTROL (v1.27).
A03	**Mirror.** Registers are decoded without address bit 5: each register read at $Dx60-$Dx7F must return the same value as at $Dx40-$Dx5F (512 rounds). A difference is a read error (R).
A04	**MEMAC_CONTROL.** 4096 random values written and read back; the window stays disabled (MGE = 0).
A05	**MEMAC_BANK_SEL.** 4096 random values (including MGE) written and read back, with MCE = MAE = 0 so the window stays disabled.
A06	**FEATURE_CONTROL** (v1.27 cores). 2048 random values written and read back, compared over the writable bits (PXGF, CLIP, XCOLOR, GTIA_PALETTE, and BLEND on GTIA core); the original value is restored afterwards.
A07	**Soft reset.** Known values set, soft reset made (write to $D080), registers with a defined reset state checked: MCE and MAE cleared with base and size kept, MEMAC_BANK_SEL kept (written with MGE = 0), BLITTER_BUSY = 0, IRQ_STATUS = 0; on v1.27 cores FEATURE_CONTROL cleared with PXGF = 0, kept with PXGF = 1. Then (FX core) the soft reset must also close the windows for every device on the bus: a MEMAC-A window of 4 KB at $8000 (when it is RAM) and a MEMAC-B bank, each over one RAM byte ($8123, $7FF0), with the interrupt off. When the window works (a byte written through it reads back, the RAM under it stays as it was), it is opened again and the soft reset made; the address must then give the RAM byte, and a byte written there must be kept. A device that keeps its own copy of the window registers and does not clear it on the soft reset (for example a 6502 replacement that decides from it which cycles go to the bus) fails here: errors with verdict R and the CPU address and window size in the log (`8123/4` , `7FF0/16`). The registers are written with 0 afterwards and the RAM byte gets its value back. Only a few reads per pass (FX v1.27: 14), so its `OPS` grows slowly.
A08	**Register load.** About 4.2 million register accesses - as many as group B makes to VRAM in one pass, in a similar mix (about 2/3 reads, 1/3 writes). Every address $Dx40-$Dx5F is read except COLDETECT (changes with the picture) and the R/W registers below: CORE_TYPE and CORE_REVISION must return their values, BLT_COLLISION_CODE its value from test start, BLITTER_BUSY and IRQ_STATUS 0 (FX core), every unused and write-only address $FF. After each read a random value is written to one of the R/W registers and read back at once: MEMAC_CONTROL (MCE = MAE = 0) and MEMAC_BANK_SEL (MGE = 0) on FX core, FEATURE_CONTROL on v1.27 cores (restored afterwards). With GTIA core v1.26 the test only reads. All accesses are always made and every error is counted.

Group B does not use the register mirror, unused or write-only addresses, so errors of A03 and A08 at those addresses say nothing about group B results.

Group B - 6502<->VRAM

Test	Description
B01	**Data lines.** At 8 random addresses (one in each 64 KB part of VRAM) values with one bit set and one bit cleared (walking 1, walking 0) are written and read back, so a stuck or shorted data line D0-D7 shows up.
B02	**Address lines.** Addresses 0 and 2^k (k = 0..18) written with two different values, one address line at a time; the log gives the address whose line is stuck or shorted ($000001 = A0 ... $040000 = A18).
B03	**Full VRAM, address pattern.** All 512 KB written with values that depend on the address (low address byte combined with the number of the 256-byte page), then verified, so mixed-up addresses (aliasing) show up. The pattern changes every pass.
B04	**MEMAC-A windows.** VRAM written by B03 is read through MEMAC-A windows of 14 settings over $8000-$FFFF: 4 KB at $9000, $B000, $F000; 8 KB at $8000, $B000, $E000, $F000; 16 KB at $8000, $9000, $C000, $E000; 32 KB at $8000, $A000, $C000. They include windows that end exactly at $FFFF, windows cut off at $FFFF (8 KB at $F000, 16 KB at $E000, 32 KB at $A000 and $C000), one across $C000 (8 KB at $B000) and one that does not start at a multiple of its size (16 KB at $9000; the VRAM block starts at the window base). Each 4K page a window covers (42 in all) is read at 16 samples, for 8 window positions (VRAM blocks) per setting: 5376 reads per pass. In $D000-$DFFF only $D800-$DFFF is read ($D000-$D7FF is I/O, where no window acts).
B05	**MEMAC-A window stress.** Six window settings over the cartridge area and the RAM below it: 4 KB at $9000, $A000 and $B000, 8 KB at $8000 and $A000, 16 KB at $8000. For each, 48 bursts, one per video frame: the CPU reads two 256-byte pages of VRAM through the window as fast as it can (NMI off, started right after the vertical blank interrupt so that it ends before the next one) and compares every byte, so every read goes through the window decoding and is exposed to the signals of the cartridge area. Up to 6 errors of a burst are examined, the others are counted without a verdict. A burst counts 512 checked reads.
B06	**MEMAC-A window writes.** 11 window settings over RAM: 4 KB at $A000, $B000, $F000; 8 KB at $A000, $E000, $F000; 16 KB at $8000, $C000, $E000; 32 KB at $8000, $C000 (7 of them reach $FFFF). They are written in $8000-$8FFF and $A000-$FFFF (never in $9000-$9FFF, where the program lies, nor in the I/O area $D000-$D7FF). For each 4K page and 8 window positions, 16 random bytes are written through the window (interrupt off); then, with the window off, the same addresses are read in RAM, and in VRAM through the usual window. A write that did not reach VRAM is an error examined as usual; a write that reached the RAM under the window gets `M` in the log. The RAM gets its old bytes back at once (the memory guard pattern, or the program's own code and data above $C000), and VRAM the data of B03. 3584 checked writes per pass.
B07	**MEMAC-B read and write.** All 512 KB read through MEMAC-B window ($4000-$7FFF, 16 KB banks) and compared with the same VRAM read through MEMAC-A. Where MEMAC-A gives the expected value and MEMAC-B another one, the verdict is R at once; other differences are examined as usual. Then one 16 KB bank (a different one every pass) is written through MEMAC-B and verified through MEMAC-A.
B08	**Full VRAM, alternating pattern.** All 512 KB written and verified with a pattern where neighbouring bytes differ in all bits, so all data lines change at every access. One pattern per pass, in turn: $00/$FF, $FF/$00, $55/$AA, $AA/$55, random data.
B09	**March C-.** Classic RAM test (write 0; read 0 write 1 upwards; read 1 write 0 upwards; read 0 write 1 downwards; read 1 write 0 downwards; read 0) on one 64 KB part of VRAM; every 8 passes cover all of VRAM. Finds faults where writing one cell changes another.
B10	**Random cells.** 4096 different random addresses written with values derived from the address, then verified.
B11	**Random blocks.** 64 blocks at random addresses, 1 to 1024 bytes long, filled with a constant, increasing or random sequence, each verified right after writing.

Test	Description
B12	**VRAM access during display.** Overlay is on (one transparent VRAM line repeated over the whole screen, invisible), so VBXE reads VRAM for display all the time, while CPU writes and verifies 128 KB. Then **ANTIC / CPU access of the MEMAC windows**: a MEMAC-A window of 4 KB at $7000, then a MEMAC-B bank, each in two steps of 2 frames. (1) Window for CPU only: through it the CPU writes to the display list address ($7FC0) a copy of the program's display list with the DLI bit in text lines 3, 14 and 26, and reads it back twice; the copy must come back right and ANTIC, still reading the RAM, must give no DLI. Only when the window works for the CPU this way comes (2) window for ANTIC only: ANTIC must read the copy (3 DLIs per frame, each at its line, VCOUNT within a few scan lines), the CPU must read the RAM. A write of step 1 that reached the RAM display list is an error with `M` (the RAM is repaired at once). The picture does not change, and windows and Overlay are switched in the vertical blank only. Errors: verdict R, see the error log; 312 checked operations per pass.

In B04, B05 and B06 the log shows the CPU address and the window size of the first error, and the detail views count the errors by window setting and 4K page. Window settings or 4K pages over $A000-$BFFF (B05: whole settings; B04, B06: 4K pages; B06 also $8000-$8FFF) are used only when the start test of the memory guard found RAM there.

Memory guard

The program keeps a known pattern (low byte XOR high byte of the address XOR $5A) in the free RAM at $A000-$BFFF and $8000-$8FFF and checks that nothing changes it. A change shows that something other than the RAM answers or writes there: a cartridge or another device on the bus, or a MEMAC-A window that is on where it should not be. It also tells B04, B05 and B06 what lies under their windows.

- **Start test** (program start and RETURN in the selection window): each byte of both areas is written with the inverted pattern and with the pattern, and read back after each write. An area with half of its bytes or more wrong is not RAM (for example a cartridge at $A000-$BFFF): `NOT RAM`, not checked further, its errors not counted. Errors of an area that is RAM are counted: `S` in `MEM:`.
- **While the tests run**: the vertical blank interrupt checks 8 bytes of $A000-$BFFF per video frame, so the whole area is checked about every 20 seconds (PAL), and a change can be found in the group after the one during which it happened. $8000-$8FFF is checked the same way only while group A runs (in groups B and C the MEMAC-A window lies there), and gets its pattern again each time group A starts; with group A off, only by the start test. Group A switches a window on only in A07, for a moment with the interrupt off, over one byte that gets its value back at once, so a change during group A means that a window register took a value that was not written.
- A wrong byte is counted for the group running at that moment, the first one of each pass is kept, and the byte gets the pattern again. The windows of B04, B05 and B06 over these areas are used with the interrupt off, so they are never checked by mistake, and B06 gives the RAM its bytes back before the guard sees them. During the window steps of B12 (a few frames) the guard waits.
- Results: `MEM:` below the group lines and the detail view. The guard is not part of `OPS`, `FAIL%` or the verdict columns, but an error of the guard turns the screen red.
- With the guard off (key M in the test selection) the start test is still made, only to find out whether the areas are RAM; nothing is checked or reported while the tests run (the detail view shows `RAM, GUARD OFF`).

Group C - Blitter<->VRAM

After a small blit (C02-C06) every byte of the destination area and a one-byte frame around it is compared with the value computed by CPU, so writes outside the area show up too.

Test	Description
C01	**BCB area and source area.** BlitterList area of VRAM ($7E000-$7EFFF) written and verified with two patterns, then the source area of the Blitter tests written and verified. If this fails, the other group C tests are not run in this pass.
C02	**Fill**: Blitter only writes (64 x 16 bytes).
C03	**Copy**: Blitter reads source and writes destination (100 x 20 bytes).
C04	**XOR**: Blitter reads source and destination and writes destination (MODE 5, 80 x 16 bytes).
C05	**Negative steps**: copy with falling addresses (destination mirrored horizontally and vertically).
C06	**BCB chain**: two BCBs linked with the NEXT bit (fill and copy) in one BlitterList.
C07	**Full VRAM copy**: CPU writes lower 256 KB with random data, Blitter copies it to upper 256 KB, then back to lower 256 KB (different XOR value each time); after each copy CPU verifies the result while Blitter keeps copying the verified area onto itself, so VRAM is read and written by CPU and Blitter at the same time.
C08	**512 KB fill timing**: 8 BCBs fill all of VRAM; time shown as `512K FILL` on the group C screen, then all of VRAM is verified.
C09	**Blitter and CPU together**: while Blitter fills 64 KB, CPU writes and verifies 128 KB elsewhere in VRAM; then the 64 KB filled by Blitter are verified.

Test times

Without errors A08, B03, B08 and C07 take longest; the other tests take a few seconds (B05 about 6, one burst per video frame). A test that suddenly takes much longer than before shows on the counter in the status line. With errors, each test may add its examination time per pass:

Test	Examination		Test	Examination		Test	Examination
A01	35 s		B03	45 s		C01	45 s
A02	35 s		B04	45 s		C02	50 s
A03	no limit		B05	45 s		C03	50 s
A04	35 s		B06	45 s		C04	50 s
A05	35 s		B07	45 s		C05	50 s
A06	35 s		B08	45 s		C06	50 s
A07	no limit		B09	45 s		C07	50 s
A08	35 s		B10	45 s		C08	50 s
B01	no limit		B11	45 s		C09	50 s
B02	no limit		B12	45 s			

"No limit": the test can find only a few errors (B01 at most 128, B02 at most 380, A07 about 14), or its verdict needs no examination (A03: one register, two answers - always R). When everything fails, the examination times add up to about 20 minutes per pass.

VBXE DOCTOR v1.5 - Tomasz Piórek / electron