

VBXE DOCTOR

v1.4 User Manual

Tomasz Piórek / electron · 28 October 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** **16**
 - Group A - 6502<->IO registers 16
 - Group B - 6502<->VRAM 17
 - Memory guard 13
 - Group C - Blitter<->VRAM 19
 - Test times 20

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; all tests are selected at start, and the key INVERSE opens a window in which groups and single tests can be switched off and on (see [Test selection](#)):

Code	Group	What is checked
A	6502<->IO REGISTERS	core registers (\$D640-\$D67F or \$D740-\$D77F)
B	6502<->VRAM	link between 6502 and VRAM: all 512 KB written and read through MEMAC-A and MEMAC-B windows, without Blitter; MEMAC-A windows over the cartridge area
C	BLITTER<->VRAM	everything that uses Blitter, aimed at 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**. Next pass starts at once, from the first selected test. Random addresses and data change every pass, and every second pair of passes (2, 3, 6, 7, ...) verifies memory from top down, so a longer run covers more combinations.

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. Screen shows one group at a time, one line per test.

Requirements

- VBXE with **FX core v1.26 or newer v1.2x** (VERSION \$26-\$2F, with or without RAMBO), or **GTIA core** of same versions. VBXE is found automatically at \$D640 or \$D740.
- With GTIA core only group A is available (no VRAM access, no Blitter); groups B and C show N/A.
- Atari XL/XE with 64 KB or more: while tests run, OS ROM is off and part of the program runs in RAM under it. Without that RAM the program prints a message and returns to DOS.
- While tests run, IRQ is disabled; only the vertical blank interrupt (NMI) is used, also for the keyboard and the **memory guard**.
- CPU 6502 or 65816 (the program runs in 6502 emulation mode and uses documented 6502 instructions only). Waits and time limits are counted in video frames and scan lines, not in CPU cycles, so a faster CPU does not change them. With a CPU accelerator, all accesses to \$4000-\$BFFF must reach the computer's bus (the MEMAC windows lie there).
- While tests run, BASIC is off. A cartridge is not needed; with one inserted, the window settings of B04, B05 and B06 over \$A000-\$BFFF are skipped.
- Without a suitable core 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: \$0480-\$04FF short routines that put code under OS ROM)
\$0600 - \$06FF	pattern table
\$2000 - \$7FFF	program and data
\$4000 - \$7FFF	also MEMAC-B window, on only for short moments
\$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 (tests B04, B05 and 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 (BASIC off)
\$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

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

1. Start `vdctor.exe` 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** window first, before any test has run.
2. Tests run on their own, pass after pass. Screen of the group under test is shown automatically (see **Screen**); numbers change at once on the first error of a test in a pass, and then about once a second.
3. Keys, at any time:

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** window (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.
- A group chosen with the group keys stays on the screen until it has run from its first to its last test; then the screens follow the running group again. While the log is on, the group shown stays even when the next group starts.
- **SPACE** pauses the tests at once: the running test waits, **PAUSED** stands in the status line in place of the test number, the busy indicator stands still, **SPC** in the key line is in inverse video, and the picture has its usual colours. All other keys work while the tests wait; SPACE again lets them go on. The time of the pause does not count for the running test.
- **ESC** on the main screen asks first: the key line shows **ESC AGAIN: EXIT, OTHER KEY: CANCEL**. ESC again returns to DOS at once (also in the middle of a test or during a pause), with the original screen and colours; any other key cancels and does nothing else.

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

In inverse video: **SPC** while the tests are paused, the mode in use (**%** or **ERR**), and **RET:LOG** while the log is on. On the Atari 400/800 the keys INVERSE and CAPS are the Atari key (logo) and CAPS/LOWR.

The keyboard is read once per video frame, so a key takes effect at the next moment the running test lets the program look at it: as a rule within a fraction of a second.

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 (colour bits 7-4) is ignored; luminance (bits 3-0) gives 16 steps of the PAL palette 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. The test selection window is always shown with the usual colours. 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 monochrome scales. Settings made only through VIDEO_CONTROL do not survive soft reset.

Test selection

At start all tests available on the core are selected. Key **INVERSE** opens a window with the tests of one group at a time (☒ 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 running test waits while the window is shown, and the busy indicator stands still.

```
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 (☒ all its tests selected, ☐ some, ☐ none). While the window is shown the status line shows **PAUSED**.

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, with or without CTRL), TAB , A , B , C	previous / next group (TAB: next), 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; ON in inverse video)
M	memory guard on / off (default on)
RETURN	accept the selection and start the tests again from the beginning (at least one test must be selected)
ESC or INVERSE	close the window, keep the previous selection and go on with the tests

A message in the empty line above the options asks for at least one selected test when RETURN finds none.

With auto-stop on, the tests stop after each pass, when all selected tests have run once: the results of the pass stay on the screen, the status line shows **DONE** in place of the test number, and all keys work as in a pause; SPACE starts the next pass. Both settings are kept when the tests start again; ESC or INVERSE in the window bring back the previous settings.

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.

RETURN always starts the program again from the beginning, as after loading, also when the selection was not changed: the running test is abandoned, all results, the error log, the detail view statistics, the pass number and the screen colour are cleared, and the first pass begins with the selection shown. So INVERSE and RETURN together also clear all results. After ESC or INVERSE the window closes to the screen it was opened from (a pause made with SPACE stays) and the tests go on from the place where they waited; the time of the window does not count for the running test.

Tests that are not selected are left out of every pass: their lines show OFF , and a group without a selected test shows OFF in its summary line. The progress bar counts the selected tests only. Tests that do not exist on the core (for example A04 and A05 on GTIA core, groups B and C) 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, and the other Blitter tests run in every pass.

Screen

VBXE DOCTOR v1.4				FX v1.27 0D640			
	OPS	FAIL%	W%	R%	RW%	?%	BW%
A	8M4985	.	.	.	.	.	.
B	6M2611	*	*	.	*	.	.
C	4M7143	*	.	.	.	.	*

MEM: B

	OPS	GROUP	B: 6502<->URAM	FAIL%	W%	R%	RW%	?%
B01	384	.	.	.	.	.	.	.
B02	1140	.	.	.	.	.	.	.
B03	1M57	*	*	.	*	.	.	.
B04	4992	.	.	.	.	.	.	.
B05	101k	0.71	0.59	.	.	.	.	.
B06	3840	.	.	.	.	.	.	.
B07	1M62	*	.	.	*	.	.	.
B08	1M57	.	.	.	.	.	.	.
B09	983k	.	.	.	.	.	.	.
B10	12k2	0.52	0.52	.	.	.	.	.
B11	98k3	.	.	.	.	.	.	.
B12	393k	.	.	.	.	.	.	.

B10 RANDOM CELLS

SPC	TAB:GRP	CAPS: 52/ERR	INV	RET:LOG	ESC
/ B10			52%	02/0003	

Title line and status line are in inverse video, separated from the rest by one empty scan line. Also in inverse video: 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 (RETURN in the selection window) or the program ends.

Line	Meaning
title	program version, core type and version (<code>r</code> = RAMBO core), VBXE register address
groups A, B, C	per group: <code>OPS</code> - checked operations of all its tests, then <code>FAIL%</code> and one column per verdict (see below); <code>N/A</code> when the group is not available on this core, <code>OFF</code> when none of its tests is selected
line below the groups	<code>MEM:</code> with <code>S</code> when the start test of the memory guard found an error, and the letters of the groups during which it found one (e.g. <code>MEM:SAB</code>); nothing while it has found none. It stays until the tests start again
group shown	name in the middle of the line, then column names of its tests (same as above; <code>BW</code> in group C only)
test lines	one line per test of the group shown: <code>OPS</code> - checked operations of the test, then <code>FAIL%</code> and one column per verdict; <code>OFF</code> for a test that is not selected, <code>N/A</code> for one that does not exist on this core
running test	test that runs now, with its name
<code>512K FILL</code>	group C only: time Blitter needed to fill 512 KB in the last pass (information only, never an error), and <code>TIMEOUTS</code> - 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); ESC always at the end
status	busy indicator, running test (<code>PAUSED</code> while the tests wait, <code>DONE</code> after a pass with auto-stop), progress of the pass, second of the running test (from 00, counted from its start, at most 99) and pass number (4 digits; after 9999 it goes on from 0000)

Both tables have the same columns. OPS counts checked operations: each read compared with its expected value, counted when made, since the tests started (program start, or RETURN in the selection window). In group lines OPS is the sum over the group's tests, 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. In test lines it has 4 characters, letter again in place of the decimal point (1140 , 49k1 , 393k , 6M29 , 53M1 , 531M , 4G29).

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

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, e.g. 10k1 (10 100 to 10 199), 923k , 5M31 , 53M1 ; middle dot for 0. Empty field: 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: OPS counts checked reads only, and a whole pass without errors gives about A : B : C = 10 : 8 : 5.5. The group lines count since the start: while a pass runs, the groups that have already run in it count one pass more than the others, so the ratio seen on the screen changes during a pass (at the end of group A in the third pass, for example, it is about 10 : 5 : 4).

Error log

While the log is on (RETURN pressed once after the first error), test lines show the first error of each 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 (see test list)
PASS	pass number
ERR	number of errors of the test in that pass, shortened like 0PS (1k23 , 53k1)
V	verdict of the first error (see below)
ADDR	address of the first error (8 characters): VRAM address (6 digits, 19 bits), or register address (4 digits, D650). Extra details in some tests, below
EXP	expected value
G0T	value read
X0R	the bits that differ (EXP XOR G0T): 04 = bit 2, 10 = bit 4
S	status: M = B04, B05: G0T is the byte of the RAM under the window at that CPU address (the window did not answer, the RAM did); B06: the write through the window reached the RAM under it; + = in that pass some errors of the test came after its examination time; they are in FAIL and ERR but have no verdict (M is shown in place of + when both apply)

Extra details in the address field:

Test	Shown	Meaning
B04, B05	900D/16	CPU address read and size of the MEMAC-A window in KB (4, 8, 16, 32) instead of the VRAM address; window base is that 4 KB area for 4 KB windows, \$8000 for larger ones (B05: \$A000 for the 8 KB window over the cartridge area)
B07	020040:A , 020040:B	path that gave G0T : 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

Each line keeps the first error from the last pass in which the test found one: it changes at most once per pass, at the first new error, and stays as is in passes without errors; ERR keeps counting the errors of that pass. A test that has run through at least once without an error shows OK; a test that has not yet run through shows its number only. Log is updated while shown; tests keep running.

Detail views

In the error log, key **CAPS** replaces the log with a detail view; the tests keep running and the numbers are updated once a second. In the log of group B: first press B04, second press B05, third press the **memory guard**, fourth press (or ESC) back to the log. In the log of group A or C: first press the memory guard, second press (or ESC) back to the log. RETURN: log off.

B04 and B05

B05 DETAIL: WINDOW STRESS

ERRORS BY WINDOW AND CPU PAGE

WINDOW		\$8000	\$9000	\$A000	\$B000
\$9000	4K	-	.	-	-
\$A000	4K	-	-	55	.
\$B000	4K	-	-	-	.
\$8000	8K	.	.	-	-
\$A000	8K	-	-	.	.
\$8000	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

	G0T	ERRORS	X0R	ERRORS
1ST	00	41	04	22
2ND	FF	3	10	11
3RD	-		14	8
4TH	-		20	2
MORE	-		+	

- The first table gives the errors of each window setting (rows) by the 4 KB page of the CPU address that was read (columns). A dash marks a page that the window does not cover. In the example the errors come from reads at \$A000-\$AFFF through the 4 KB window at \$A000, and from nowhere else.
- The second table gives the errors by the position in the sequence of reads. B04: number of the read (0-15) among the 16 reads of one visit. B05: section of the burst (the 512 bytes of a burst in 16 sections of 32 bytes; the position at the start of the sequence is read 0 of section 0). Both tables count every error, also those found after the examination time and those of a burst after its first 6. The number of the first column is added to the column heading: row 4, column +2 is sample 6.
- The last table gives the values most often read by the failing reads (G0T) and the bits most often wrong (X0R , EXP XOR G0T), each value with the number of errors that gave it (shortened like OPS : 1664 , 24k4), up to four values, the most frequent first (lines 1ST to 4TH). - 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 values shown are the most frequent ones, their numbers are lower than the true ones, and a rare value can leave its place empty (-). B04 counts every error here; B05 counts the errors it examines (up to 6 of each burst). The same G0T value at every error (for example always 00 or FF) points to something else that answers in place of the VRAM; the same X0R at different G0T values points to particular data bits.

Counts stop at 65535. A middle dot means no error. The detail view shows that the errors depend on the CPU address range (for example only the cartridge area \$A000-\$BFFF), on the window setting, or on the moment within the sequence of reads.

Memory guard

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: in line S by the start test (STRT in place of the pass), in lines A - C while that group ran. Errors of the guard in that pass, no verdict, RAM address / test that was running, expected and read value, bits that differ. 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 (see below). 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 read only at its own address, never through its mirror at \$Dx60-\$Dx7F (used by test A03 only), so for registers the table below has one path, and ? 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: when all later reads give the right value, reads are still unstable.

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

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. One such case is enough for W.
R	unstable, 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.

Blitter (group C)

Who writes and who reads in group C:

- before the blit **CPU writes** source area, destination area and BCBs, and **reads them back**; an error here gets a CPU verdict (W, R, RW or ?) and the test is not run;
- **Blitter writes** the destination area (it reads BCBs and source area itself);
- after the blit **CPU reads** each byte of the destination area (through MEMAC-A and MEMAC-B, as above) and compares it with the value it computed.

In group C CPU verdicts are additional reports of Blitter tests: they come from preparing a test (including C01) and from reading its result. A wrong byte of the result is examined like any other location. When both paths show the same wrong value, **CPU writes** the expected value to that byte 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 (R, RW, ?) are 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.

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.

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 - enough for about 1000 verdicts, which gives the shares of W, R and RW within about 3 points. The times are in **Test times**.

When the time is used up, further errors of the test in that pass are counted in **FAIL** only: they get no verdict and are not put into any verdict column, so no verdict is ever guessed. The first error of a test in a pass is always examined, so the log always shows a verdict; **+** in column **S** of the log shows that some errors of that pass were not examined. 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). Error log shows the first register that gave an unstable read. Unused and write-only addresses are checked by A08.
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; each must come back exactly. Window stays disabled (MGE = 0), so no memory is covered.
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 writable bits (PXGF, CLIP, XCOLOR, GTIA_PALETTE, and BLEND on GTIA core). Original value restored afterwards. Picture colours change during this test (see Using the program).
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 also FEATURE_CONTROL: cleared after reset with PXGF = 0, kept with PXGF = 1. Only a few reads per pass (FX v1.27: 2 resets x 5 registers = 10), 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) - so error rates of groups A and B can be compared. Every address \$Dx40-\$Dx5F is read except COLDETECT (changes with the picture), FEATURE_CONTROL and, on FX core, MEMAC_CONTROL and MEMAC_BANK_SEL (written and read back instead, 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; picture colours change while the test runs). With GTIA core v1.26 the test only reads. Errors do not stop the load: all accesses are always made and every error is counted.

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; error 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. Mixed-up addresses (aliasing) show up as errors. Pattern changes every pass.
B04	MEMAC-A windows. VRAM written by B03 is read through MEMAC-A windows of 4 KB at \$9000, \$A000 and \$B000, and of 8, 16 and 32 KB at \$8000 (16 samples in each 4 KB part below \$C000, 8 window positions per setting). The log shows the CPU address and the window size of the first error; the detail view (CAPS in the log of group B) shows the errors by window setting, CPU page and number of the read. Settings that need \$A000-\$BFFF are used only when the start test of the memory guard found RAM there (no cartridge). When B03 is not selected, B04 writes the blocks it reads itself.
B05	MEMAC-A window stress. Six window settings: 4 KB at \$9000, \$A000 and \$B000, 8 KB at \$8000 and \$A000, 16 KB at \$8000. For each of them, 48 bursts: the CPU reads two 256-byte pages of VRAM through the window as fast as it can (a plain compare loop with the NMI off, about 120 scan lines, up to about 220 when every byte is wrong; started right after the vertical blank interrupt, so that it ends before the next one) and compares every byte; one burst per video frame. The data is the one B03 wrote (when B03 is not selected, B05 writes the blocks it reads itself). The windows lie over the area where a cartridge sits (\$A000-\$BFFF) and over the RAM below it, so every read goes through the window decoding and is exposed to the signals the cartridge area adds, as densely as the CPU can read. Up to 6 errors of one burst are examined; the other errors of the burst are counted without a verdict. The log shows the CPU address and the window size of the first error; the detail view (CAPS in the log of group B, second press) counts every error of every burst, by window setting, CPU page and section of the burst. Settings that need \$A000-\$BFFF are used only when the start test of the memory guard found RAM there. A burst counts 512 checked reads.
B06	MEMAC-A window writes. Writes through MEMAC-A windows over the RAM the memory guard keeps: 4 KB at \$A000 and \$B000, 8 KB at \$A000, 16 and 32 KB at \$8000 (only the 4 KB parts \$8000, \$A000 and \$B000; never \$9000, where the program lies). For each 4 KB part 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 the RAM, and then in the VRAM through the usual window. A write that did not reach the VRAM is an error examined as usual; a write that reached the RAM under the window gets M in column S of the log (and is an error of its own, verdict W , when the VRAM got the right value). The RAM gets the pattern back at once and the VRAM the data of B03. Parts of \$A000-\$BFFF are used only when the start test found RAM there. Each written byte counts one checked operation (1280 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 algorithm (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.

Test	Description
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.
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.

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.

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-\$FFFF and checks that nothing changes it. A change shows that something other than the RAM answers there 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): it is shown as `NOT RAM`, not checked further and its errors are not counted; B04, B05 and B06 then skip the window settings over \$A000-\$BFFF. Errors of an area that is RAM are counted: `S` in `MEM:`, and line `S` of the [detail view](#).
- **While the tests run:** the vertical blank interrupt checks 8 bytes of \$A000-\$BFFF per video frame, all the time, so the whole area is checked about every 20 seconds (PAL). \$8000-\$FFFF is checked the same way only while group A runs, because in groups B and C the MEMAC-A window lies there; it gets its pattern again each time group A starts. With no test of group A selected, \$8000-\$FFFF is checked by the start test only. Group A never switches a window on, so a change of \$8000-\$FFFF or \$A000-\$BFFF during group A means that a window register took a value that was not written.
- A wrong byte is counted for the group that runs at that moment, the first one of each pass is kept (address, expected and read value, running test) and the byte is written again with the pattern. The windows of B04, B05 and B06 over these areas are used with the interrupt off, so they are never checked by mistake.
- Results: `MEM:` with `S` (start test) and the letters of these groups at the right end of the line below the group lines; the [detail view of the memory guard](#) (CAPS in the error log). The guard is not part of `OPS`, `FAIL%` or the verdict columns. 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 for B04, B05 and B06; nothing is checked while the tests run and nothing is reported (no `MEM:`, the detail view shows `RAM, GUARD OFF`).
- A change of \$A000-\$BFFF is found within about 20 seconds, so it can be found in the group after the one during which it happened. Changes of \$8000-\$FFFF always belong to group A.
- `M` in column `S` of the log, B04 and B05: the value read through the window equals the byte of the RAM under it at that CPU address (the pattern at \$8000-\$FFFF and \$A000-\$BFFF, the program data at \$9000-\$9FFF). The window did not take the bus for that read and the RAM answered. A wrong value that does not match the RAM (for example always \$00) comes from somewhere else.

- **M** in column **S** of the log, B06: the write through the window reached the RAM under it (B06 finds it and writes the pattern back before the guard checks that byte, so the guard does not count it too). With a wrong value in the VRAM the line shows the VRAM (**EXP** = value written, **G0T** = VRAM); when the VRAM got the right value and the RAM got it too, verdict W and the RAM byte (**EXP** = guard pattern, **G0T** = RAM).

Group C - Blitter<->VRAM

Everything that uses Blitter, aimed at Blitter access to VRAM. Errors found before a blit get a CPU verdict and the test is not run (see **Blitter verdicts**). 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 source area of the Blitter tests written and verified. If this fails, other group C tests are not run in this pass (their OPS do not grow).
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. Blitter reads and writes every byte of VRAM.
C08	512 KB fill timing: 8 BCBs fill all of VRAM; time shown as 512K FILL on 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

Status line shows the second of the running test. How long a test takes without errors depends on the computer (CPU clock, memory, core), so no fixed times are given; A08, B03, B08 and C07 are the longest, the other tests take a few seconds (B05 about 6 seconds, because it makes one burst per video frame). A test that suddenly takes much longer than before is easy to see on the counter. The table gives the examination time each test may add per pass when it finds errors. Counting errors after the examination time adds a little more. The counter stops at 99.

Test	Examination	Test	Examination	Test	Examination
A01	35 s	B03	45 s	C01	45 s
A02	35 s	B04	no limit	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, B04 at most 1664, A07 about 10), 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.