MemTest64/128/65 (*v1.10) – 02/20/2020.*

MemTest64 is a memory test program by Robert Olessak (written in 2017-2020).

All of its contents are Public Domain: open-source and freeware.

The application is designed for the 8-bit Commodore computers or clones: mainly for the C64/128/65 family (including the MEGA65 or other FPGA ones), but also for the other machine types: PET/CBM/CBM-II, VIC-20, LCD and TED series.

The original package contains the executable together with the appropriate source code and the project files (and descriptions). You can download it from here:

http://istennyila.hu/stuff/memtest.zip

There is only one version of the same program, to be run on all computers. It detects the system and the machine type, the CPU and other controllers (REC or DMA), and also tries to detect the most common internal and external memory expansions: including REU, GeoRAM, SuperRAM, and so on… as well as the built-in VDC video memory of C128.

After detecting, it tests them, too: tries to write some random data into all (or at least the very most) of the available free spaces, and afterwards to read them back. If comparing the results succeeds, it simply reports that as “ok”; if it fails, it reports that as an “error” (which means that the given device or even the computer itself has indeed got some hardware failure in most cases, so thus it should no longer be recommended to be used this way). While testing, it also makes some speed measurements and prints them on the screen, so that the different kinds of memory transfers can be compared.

The program can be started in two ways:

```
LOAD “MEMTEST.PRG” ,8
RUN
```

(which on the C128 or C65 may also be typed as RUN “MEMTEST.PRG”), **or:**

```
LOAD “MEMTEST.PRG” ,8,1
SYS 5133
```

(which on the C128 or C65 may also be typed as BOOT “MEMTEST.PRG”).

There is usually not much difference between the two ways, considering both of them should work in most cases – but not always. (For example, the latter must be used on PET.)

On those machines having several operating modes (which are the C128, the C65 and the MEGA65) it can be both run in 64 and native modes. (The actual behaviour might be different, because some system components are hidden in the C64 mode by hardware, and consequently cannot be found.) Including the fast mode and the 80-column mode.

The display output:

A typical screen may look like this (when running on a C128 in slow mode):

```
c128 (8502) with 128k
pal on 312 scanlines
cpu speed at 0.93 mhz
vdc r1 with 16k
reu df00 128k
geo de00 64k

testing cpu memory… ok
testing cpu bank 1… ok
testing vdc dram 0… ok
testing reu memory… ok
testing geo memory… ok
testing colour ram… ok

normal      17345
zp+sp       6557
vdc/load    37367
vdc/copy    2916
vdc/fill    1772
reu         1388
geo         15393
```

The first line shows the machine type, the CPU and main memory detected. For example, if you run it on a C64, you will get a ***"c64 (6510) with 64k"*** there, but if you run it in the C64 mode of a C128, then you will get a ***"c64 (8502) with 64k"*** instead… and so on.

If you have an active accelerator card or board, or a cloned machine, then it shows up there instead of the CPU number. For example (there are also further possibilities):

65816	if you have a SuperCPU (or a Flash 8 or a Turbo Process)
65c02	if you have a Turbo Master board (or Schnedler card)
dtv/chm/ult/*etc.*	C64DTV / Turbo Chameleon / Ultimate *(etc.)*

The ***main memory*** must always be 64K for a C64, of course. It is typically 128K for a C128 or a C65, but sometimes may increase to 256K (if all possible four banks are filled in), or perhaps, rarely decrease to 64K (if the second bank is missing or broken). For the other computers, it may vary between 4 and 40 kilobytes (on VIC-20 and PET machines), or 16 and 64 kilobytes (on LCD and TED machines), depending on the actual configuration tested.

The ***PAL/NTSC type*** and the number of the scanlines are detected directly from the physical hardware (by counting the raster lines on screen), also on the VIC-20 and the TED machines. After having been scanned, the program sets it for the operating system, too. (Because many of the Kernals themselves do it simply false, or the value may have got corrupted later, so it needs to be fixed by the software.)

The ***CPU speed MHz*** is also measured directly, by using a pretty exact method, from the physical hardware (by counting the executed machine cycles in an exact 0.1 second interval based on the PAL/NTSC setting detected before and using the raster counter register as reference, then comparing the counted cycles to a "hypothetical" 1 MHz standard).

The ***VDC revision*** is normally 1 (or sometimes 0) for a flat C128 or a C128D (8563 chip), and 2 for the C128DCR model (8568 chip). The ***video memory*** is 16K in the first, and 64K in the second case by factory, but since any C128 machine can easily be expanded to 64K (by putting in a little expansion board by hand and without soldering), it can be found more and more often these days. It is likewise detected directly from the physical hardware (by writing to and reading back and comparing) as well as being set for the operating system, too. (Because the stock Kernal of the subsequently expanded machines does usually not recognize it, so the appropriate value of the VDC register #28 must be manually set.)

The ***"dram x"*** part refers to the ***DRAM refreshing rate***, which is always set to 5 by default (by the operating system), but it can be manually altered (at VDC register #36) for improving the speedness of the VDC. This value may be set to 0 through 15, and the program always tries to set it as low as possible. (Once having set, it tests the memory again, and on encountering any data corruption, it sets the value a bit higher, until getting stable or reaching the maximum.) For most computers, it should normally be able to stay zero (if you see it being set at any other value, it means your video memory actually requires that change).

The ***REU memory*** (or *RAM Expansion Unit*) was the official external memory expansion made by Commodore (from 128K to 512K, but later re-manufactured by others up to 2 MB). Its theoretical maximum is 16 MB (and also has a DMA controller for fast memory block copying). The original device is now rare and expensive, but very fortunately it is well emulated by the *Turbo Chameleon*, the *U64* or the *1541 Ultimate I-II-II+* cartridges (and the VICE emulator, too, of course). It is generally mapped into the $DF00 I/O area by default, but sometimes can be found at the $DE00 area (if you have a cartridge multiplier for swapping the lines). Both the address and the size are automatically recognized: e.g. ***"reu df00 xxx"***.

The ***GeoRAM*** was another (much cheaper and simpler) solution (mostly used for GEOS). Nowadays it is also emulated by the above-mentioned devices, up to 4 MB. It may comfortably co-exist with the REU in the same computer even at the same time. Although it uses both I/O areas a bit, they still do not interfere, because only two bytes are taken from the very end of $DF00 normally (and the whole $DE00 for the data interchange). They can be similarly swapped with the other space sometimes (see above for cases). So the normal case is displayed as just ***"geo de00 xxx"*** by the program, yet in the opposite case it should be also recognized. *(Warning: if you use some other cartridges or emulations simultaneously, like e.g. a freezer card or RR-Net, or even a second SID at $DF00, they might cause the REU or the GeoRAM or either to not appear, by taking off the corresponding I/O areas from them.)*

The *GeoRAM* might also be used on VIC-20 (either by the MasC=erade cartridge or some kind of emulation). The I/O areas are $9800 and $9C00 there. It should also be recognized at either place (which has only been tested in the VICE emulator yet).

The ***SuperRAM*** is an add-on for the SuperCPU: up to 16 MB (the contemporary SIMM RAM modules used in 486 machines) and be handled natively by the 24-bit addressing modes of the 16-bit CPU (so it is the best of all). It is displayed as ***"srm scpu xxx"*** if found.

The built-in ***DMA*** controllers of the C65 (at $D700) and the C64DTV (at $D300) work more or less similarly to the REU (and also to each other) while handling up to 8 MB (C65) or 2 MB (DTV). Either of them is simply reported as ***"dma dx00 xxx"*** if found.

The ***colour RAM*** test examines the whole amount of CRAM: 1K nybbles (4-bit) on the C64 (and the same in 8-bit on the DTV or the C65 in C64 mode); 2 x 1K nybbles on the C128 (in native mode); and 2K on the C65/MEGA65 (in native mode).

The speed measurements:

These are only available on the C64/128/65 platforms (by using the timer of the CIA).

Each value represents the aggregate number counted on **one kilobyte** transfers. (Thus, the lower means the faster, and the higher means the slower.) The actual numbers depend on the actual operating of the CIA (that normally means counting the machine cycles at the standard 1 MHz frequency, but in some rare and special cases might be something else). They are always being summed together, up to 128 times of 256-byte transfer rounds (so thus up to 32K in total), and then divided back to get the 1K average after all.

The program does not change the actual fast mode or turbo settings of the computer (apart from some temporary disabling of fast mode that the REU requires, but it does not affect the speed at all). However, it *disables the interrupts* (so that they do not disturb). These are still *not* the fastest values ever possible, because ***the screen is on***, so the video chip takes its own cycles away during all the time. (Which is the case on normal operating, too, of course.) You should manually turn off the screen before running the memtest program, if you want to see the maximum. Similarly, if you want to see the changes made by any fast or turbo modes, then you should manually set these modes up at your will before.

These following transfers can be tested by the program:

Normal: this is just the normal way of copying an area (by making a repeating cycle for the main CPU: reading and writing byte by byte). The result is the average speed of normal memory access being executed by the CPU and consisting of **LDA ($xx),Y** and **STA ($xx),Y** (indirect-indexed addressing mode) instruction pairs organized in cycles.

ZP+SP: at least four platforms have the ability of relocating ZP + SP: i.e. both the zero page (or as called then, base page) and the stack. Which can be used for faster memory access (for example copying). So can do the following: C128, C65, DTV and SuperCPU. The copying is made via some pairs of **LDA $xx** and **PHA** instructions organized in speed code in this case. *(See for more details on this method in my corresponding article below!)*

VDC (load): writing by the CPU into the video memory of VDC (or reading from). This is generally rather sluggish, since the CPU needs checking and waiting for the ready state of VDC chip after every single byte (and this is why the VDC is said to be so "slow").

VDC (copy): using a built-in feature of the VDC chip, called as ***VDC block copy***, which copies a given area within the video memory. (It is sometimes also referred to as ***blitter***, although being not so powerful as the real blitter features of some more advanced machines, like the Amiga.) Compared to the above one, it's very fast!

VDC (fill): another built-in feature of the VDC chip, called as ***VDC block fill***, which writes the same value repeatedly into the memory at given times (e.g. to clear the screen).

An important note on both of the above features, that they are executed by the video controller chip on its own, i.e. completely independently of the main CPU (so when you really use them, you need not waiting for the result, but can do anything else in parallel). That is a great advantage, though this program cannot make much use of that (so it is just waiting).

Either on 16K or 64K VDC, each and every byte and bit of the whole amount gets tested (then the not used areas are erased, while the system areas are restored).

If you want another really thorough, definitely VDC-specific test, then please run the ***vdcdump.prg***, too (only in the 40-column C128 or C64 mode, to leave the VDC intact), which is also included in a separate folder together with its own descriptions (its own "readme.txt" and "patterns.txt" files, please read them before running it!). That will run for about *half an hour* on an average machine, while writing all the VRAM full with specific values and reading them back, then comparing the given results to the known patterns of my test machines (and the VICE emulator's layout as well). Also every bit is tested in this way.

*(The **vdcdump.prg** is also suitable for checking the accuracy of an emulator.)*

SRM: using SuperRAM (on SuperCPU). That is being accessed directly via 24-bit indexed addressing mode instruction pairs like **LDA $xxxxxx,X** or **STA $xxxxxx,X** organized in cycles.

DMA: on C65 or DTV, by using their own integrated DMA features for copying.

REU: making an REU transfer. (That is also DMA.)

An important difference between the two cases is that whereas the former two can access the main memory, the REU not (directly for copying within). Thus, if you use them for main memory copying, then it will just need *one* transfer for the C65/DTV, while *two* transfers for the REU (once from the main memory to the REU, and once again from REU back to main memory). Here is always measured only one transfer.

GEO: using GeoRAM. This is basically not so different from using normal memory in most cases, nevertheless it seems a little bit faster here, just because of being implemented through some **LDA $xxxx,X** and **STA $xxxx,X** instruction pairs (i.e. normal indexed addressing modes instead of the indirect-indexed), which is slightly faster in advance.

Please also keep in mind that if you have some very large memories (of several megabytes), then it may last rather long for testing them (for several minutes or more!).

On the C65 and the MEGA65 also a summary of addressable memory for the mapper and the DMA is displayed (as well as the 32-bit address space if available).

If you want to see some more details or gain some better understanding of the methods (and the overall working of the program, especially its platform independent launching), then please either look into the source code (also in the zip file included), and/or read my longer article (for which the link is at the bottom of the next page).

For the C65 users:

As for the C65 compatibility, it could only be tested in the MESS emulator, and thus it is not so guaranteed to work the same well in any other circumstances (e.g. on a real prototype or MEGA65). (But let us hope it works the same…)

If you would like to also try it, then the ***MESS v0.111*** version (from 2006) will be needed (and possibly its GUI extension, called MessGui, too). This is necessary unfortunately because all newer versions of MESS have got their entire C65 emulations broken (thus totally useless). That is not my fault, of course… Once having the right version, it is recommended to be started with the *v0.9.**910111*** (i.e. '91. January) Kernal ROM version (that originates from the earliest and probably the most common *Rev2B* motherboard); as it is the most solid and reliable Kernal version, too. (The MESS version has also got a lot of other flaws and quirks, and because of them the internal drive with device number 8 is not usable. So must you then attach a D64 image as an external drive with 10 or 11, and start it all from there.)

Actually, it is being tested on the MEGA65 (but since the platform itself is still under construction, there is no guarantee for the future). If you are interested, just see it there:

http://forum64.de/index.php?thread/90572-memtest64/

About the author:

This entire *MemTest64 project* initially started as an independent part of my *Rosetta Interactive Fiction* project. (As I needed some kind of testing utility, and since I had not found any, I have had to make it by myself.) However, you can also freely apply it, of course.

See more details in my article called as ***Writing Platform Independent Code on CBM Machines*** (of 51 pages in the PDF):

http://istennyila.hu/dox/cbmcode.pdf

MemTest64 *project homepage:*

http://istennyila.hu/memtest64

SDOS *project homepage:*

http://istennyila.hu/sdos

Rosetta Interactive Fiction *project homepage:*

http://istennyila.hu/rosetta

(On opening them please click onto the greeting images for entering the main page!)

Robert Olessak (2012-2020)