# VICE, the Versatile Commodore Emulator

# 1 GNU GENERAL PUBLIC LICENSE

Version 2, June 1991



## Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

## TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

   These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:
   a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
   b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
   c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

   The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

   If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.
5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.
7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

   This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.
8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.
9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

   Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.
10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software

which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

## NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

## END OF TERMS AND CONDITIONS

## How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
one line to give the program's name and an idea of what it does.
Copyright (C) 19yy  name of author

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) 19yy name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details
type 'show w'.  This is free software, and you are welcome
to redistribute it under certain conditions; type 'show c'
for details.
```

The hypothetical commands '`show w`' and '`show c`' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than '`show w`' and '`show c`'; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright
interest in the program 'Gnomovision'
(which makes passes at compilers) written
by James Hacker.

signature of Ty Coon, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

# 2 About VICE

VICE is the one and only *Versatile Commodore Emulator*. It provides emulation of the Commodore C64, C64DTV, C128, VIC20, PET, PLUS4 and CBM-II computers within a single package. The emulators run as separate programs, but have the same user interface, share the same settings and support the same file formats.

**Important notice:** If you have no idea what a Commodore 8-bit computer is, or have questions about how these machines are used, how the file formats work or anything else that is not strictly related to VICE, you should read the appropriate FAQs *first*, as that kind of information is not available here. See Chapter 16 [Contacts], page 118. for information about how to retrieve the FAQs.

All the emulators provide an accurate 6502/6510 emulator, with emulation of all the opcodes (both documented and undocumented ones) and accurate timing. Unlike other emulators, VICE aims to be cycle accurate; it tries to emulate chip timings as precisely as possible and does so *efficiently*.

Please do *not* expect the C64DTV, C128, PET, PLUS4 and CBM-II emulators to be as good as the C64 or VIC20 one, as they are still under construction.

*Notice:* This documentation is written for the Unix release of VICE.

## 2.1 C64 emulator features

As of version 2.3, two C64 emulators are provided: '`x64`' (fast) and '`x64sc`' (accurate).

The fast C64 emulator, called '`x64`', features a fairly complete emulation of the VIC-II video chip: sprites, all registers and all video modes are fully emulated. The emulation has been fully cycle-accurate since version 0.13.0.

The accurate C64 emulator, called '`x64sc`', features a cycle-based and pixel-accurate VIC-II emulation. This requires a much faster machine than the old '`x64`'.

A rather complete emulation of the SID sound chip is also provided. All the basic features are implemented as well as most of the complex ones including synchronisation, ring modulation and filters. There are three emulators of the SID chip available: first is the "standard" VICE emulator, available since VICE 0.12; the second is Dag Lem's reSID engine and the third one is reSID-fp. The reSID engines are a lot more accurate than the standard engine, but they are also a lot slower, and only suitable for faster machines.

Naturally, also both CIAs (or VIAs, in some cases) are fully emulated and cycle accurate.

## 2.2 C64DTV emulator features

The C64DTV emulator, called '`x64dtv`', features emulation of C64DTV revisions 2 and 3. The emulator is under construction, but most of the DTV specific features are already supported (with varying accuracy).

Video cache is disabled by default as it currently doesn't work with some of C64DTV's new video modes. The new video modes have a simple "fake" video cache implementation that may give incorrect results and decreased performance.

## 2.3 C128 emulator features

The C128 emulator, called '`x128`', features a complete emulation of the internal MMU (*Memory Management Unit*), 80 column VDC screen, fast IEC bus emulation, 2 MHz mode, Z80 emulation plus all the features of the C64 emulation.

## 2.4 VIC20 emulator features

The VIC20 emulates all the internal hardware, including the VIA chips. The VIC-I video chip is fully emulated except NTSC interlace mode, so most graphical effects will work correctly.

Sound support is implemented, but is still at an experimental stage. If you think it could be improved and know how to do so, feel free to contact us (see Chapter 16 [Contacts], page 118).

The VIC20 emulator now allows the use of the VIC1112 IEEE488 interface. You have to enable the hardware (by menu, resource, or commandline option) and then load the IEEE488 ROM (see for example `http://www.funet.fi/pub/cbm/schematics/cartridges/vic20/ieee-488/325329-04.bin`, but you have to double the size to 4k for now). The IEEE-488 code is then started by `SYS45065`.

## 2.5 PET emulator features

The PET emulator emulates the 2001, 3032, 4032, 8032, 8096, 8296 and SuperPET (MicroMainFrame 9000) models, covering practically the whole series. The hardware is pretty much the same in each and that is why one single program is enough to emulate all of them. For more detailed information about PET hardware please refer to the '`PETdoc`' file.

Both the 40 column and 80 column CRTC video chips are emulated (from the 4032 onward), but a few of the features are not implemented yet (numbers of rasterlines per char and lines per screen). Fortunately, they are not very important for average applications.

Sound is available for the PET as well, but like the VIC20's it is still under construction.

The PET 8096 is basically a PET 8032 with a 64k extension board which allows remapping the upper 32k with RAM. You have to write to a special register at `$fff0` to remap the memory. The PET 8296 is a 8096 but with a completely redesigned motherboard with 128k RAM in total. Of the additional 32k RAM you can use only some in blocks of 4k, but you have to set jumpers on the motherboard for it. VICE uses the command line options '`-petram9`' and '`-petramA`' instead. Also, the video controller can handle a larger address range. The PET 8x96 model emulations run the Commodore LOS-96 operating system - basically an improved BASIC 4 version with up to 32k for BASIC text and 32k for variables. See '`PETdoc`' for more information.

The SuperPET also is a PET 8032 with an expansion board. It can map 4k at a time out of 64k into the `$9***` area. Also it has an ACIA 6551 for RS232 communication. The 6809 that is built into the SuperPET is not emulated, though.

The PET computers came with three major ROM revisions, so-called BASIC 1, 2 and 4, all of which are provided. The PET 2001 uses the version 1, the PET 3032 uses version 2, and the others use version 4. The 2001 ROM is horribly broken with respect to IEEE488

(they shipped it before they tested it with the floppy drive, so only tape worked. Therefore the emulator patches the ROM to fix the IEEE488 routines.

As well as other low-level fixes the 2001 patch obtains the load address for a program file from the first two bytes of the file. This allows the loading of both PET2001-saved files (that have $0400 as their load address) and other PET files (that have $0401). The PET2001 saves from $0400 and not from $0401 as other PETs do.

Moreover, the secondary addresses used are now `0` and `1` for load and save, respectively, and not arbitrary unused secondary addresses.

To select which model to run, specify it on the command line with the `-model MODEL` option, where `MODEL` can be one of a list of PET model numbers, all described in see Section 7.6.1 [PET model], page 69

## 2.6 CBM-II emulator features

The CBM-II emulator emulates several types of CBM-II models. Those models are known under different names in the USA and Europe. In the States they have been sold as `B128` and `B256`, in Europe as `CBM 610`, `CBM 620` (low-profile case) or `CBM 710` and `CBM 720` (high-profile case with monitor). In addition to that now an experimental C510 emulation is included. The C510 (also known as P500) is the little brother of the C600/700 machines. It runs at roughly 1 MHz and, surprise, it has a VIC-II instead of the CRTC. Otherwise the different line of computers are very similar.

These computers are prepared to take a coprocessor board with an 8088 or Z80 CPU. Indeed there are models `CBM 630` and `CBM 730` that supposedly had those processors. However these models are not emulated.

The basic difference is the amount of RAM these machines have been supplied with. The `B128` and the `CBM *10` models had 128k RAM, the others 256k. This implies some banking scheme, as the 6502 can only address 64k. And indeed those machines use a 6509, that can address 1 MByte of RAM. It has 2 registers at addresses 0 and 1. The indirect bank register at address 1 determines the bank (0-15) where the opcodes `LDA (zp),Y` and `STA (zp),Y` take the data from. The exec bank register at address 0 determines the bank where all other read and write addresses take place.

The business line machines (C6xx/7xx) have the RAM in banks 1-2, resp. 1-4. All available banks are used for BASIC, where program code is separated from all variables, resp. from normal variables, strings and arrays that are distributed over other banks. The C510 instead has RAM in banks 0 and 1, and uses bank 1 for program and all variables. Bank 0, though, can be accessed by the VIC-II to display graphics.

Many models have been expanded to more than the built-in memory. In fact some machines have been expanded to the full 1M. Bank 15 is used as system bank, with only little RAM, and lots of expansion cartridge ROM area, the I/O and the kernal/basic ROMs. Some models have been modified to map RAM into the expansion ROM area. Those modifications can be emulated as well.

The different settings are described in see Section 7.7.1 [CBM-II model], page 73.

## 2.7 The keyboard emulation

There are two ways of emulating the keyboard in VICE.

The default way (*symbolic mapping*) is to map every key combination to the corresponding key combination on the real machine: for example, if you press *, which is bound to *Shift-8* on a U.S. keyboard, in the C64 emulator, the emulated machine will have just the *unshifted* * key pressed (as * is unshifted on the C64 keyboard). Likewise, pressing ' on the same U.S. keyboard without any shift key will cause the combination *Shift-7* to be pressed in the emulated C64. This way, it becomes quite obvious what keys should be typed to obtain all the symbols.

There is, however, one problem with symbolic mapping: some keys really need to be mapped specially regardless. The most important examples being, in the VIC20, C64 and C128 emulators, that CTRL is mapped to TAB and that the COMMODORE key is mapped to the left CONTROL). The RUN/STOP key is mapped to the ESC key on the PC keyboard. The PET emulator, lacking the COMMODORE key but having an ESC key, uses the left CONTROL key as RUN/STOP and the ESC key as ESC of course.

The second way (*positional mapping*) is to map every key on the "real" keyboard to the key which has the same position on the keyboard of the emulated machine. This way, no SHIFT key is forced by the program (with the exception of the function keys F2, F4, F6 and F8, which require SHIFT on the Commodore keyboards), and the keyboard is more comfortable to use in those programs (such as some games) that require the keys to be in the correct positions.

**Warning:** unlike the real C64, VICE "presses" the SHIFT key *together* with the key to shift when the SHIFT must be forced. In most cases this should work fine, but some keyboard routines are quite picky and tend not to recognize the shift key because of this. For instance, *F6* (which on the real C64 is obtained with *Shift* + *F5*) could be recognized as *F5*. In that case, use the shift key manually (i.e., type *Shift* + *F5* in the example). Yes, we know this is a bug.

The *RESTORE* key is mapped to *Page Up* (or *Prev*) by default.

## 2.8 The joystick emulation

Joysticks can be emulated both via the keyboard and via a real joystick connected to the host machine (the latter only works on GNU/Linux systems).

There are two keyboard layouts for joystick use, known as *numpad* and *custom*.

The *numpad* layout uses the numeric keypad keys, i.e., the numbers 1. . . 9 which emulate all the directions including the diagonal ones; 0 emulates the fire button.

The *custom* layout uses the keys W, E, R, S, D, F, X, C, V for the directions and SPACE for the fire button instead.

## 2.9 The disk drive emulation

All the emulators support up to 4 external disk drives as devices 8, 9, 10 and 11. Each of these devices can emulate virtual Commodore 1541, 1541-II, 1571, 1581, 2031, 2040, 3040, 4040, 1001, 8050 and 8250 drives in one of four ways:

- using disk images, i.e., files that contain a dump of all the blocks contained in a real floppy disk (if you want more information about what a disk image is, consult the `comp.emulators.cbm` FAQ);

- accessing file system directories, thus giving you the use of files without having to copy them to disk images; this also allows you to read and write files in the `P00` format (again, consult the `comp.emulators.cbm` FAQ for more info).
- accessing a real device connected to the host machine. As of VICE 1.11 it is possible to connect real drives like Commodore 1541 to the printer port of the host using the XA1541 or XM1541 cable. Currently this only works on Linux or Windows using the OpenCBM library. You can get it from `http://www.lb.shuttle.de/puffin/cbm4linux` (cbm4linux, Linux version) or from `http://cbm4win.sf.net/` (cbm4win, Windows version).
- directly using the disk drive of the host. The 3.5" disk drive of the host can be used to read or write Commodore 1581 formatted disks. Currently this raw drive access feature is only available for Linux hosts.

When using disk images there are two available types of drive emulation. One of them the *virtual drive* emulation. It does *not* really emulate the serial line, but patches the kernal ROM (with the so-called *kernal traps*) so that serial line operations can be emulated via C language routines. This emulation is very fast, but only allows use of standard DOS functions (and not even all of them). For real device or raw drive access it is required to enable this type of emulation.

The IEEE488 drives (2031, 2040, 3040, 4040, 1001, 8050 and 8250) do not use kernal traps. Instead the IEEE488 interface lines are monitored and the data is passed to the drive emulation. To use them on the C64, you need to enable the IEEE488 interface emulation. Only if the IEEE488 emulation is enabled, those drives can be selected.

The other alternative is a *true drive* emulation. The Commodore disk drives are provided with their own CPU (a 6502 as the VIC20 and the PETs) and their own RAM and ROM. So, in order to more closely emulate its features, a complete emulation of this hardware must be provided and that is what the *hardware level* emulation does. When the *hardware level* emulation is used, the kernal routines are remain unpatched and the serial line is fully emulated. The problem with this emulation is that it needs a lot of processing power, mainly because the emulator has to emulate two CPUs instead of one.

The PETs do not use a serial IEC bus to communicate with the floppy drive but instead use the parallel IEEE488 bus. This does *byte by byte* transfers, as opposed to the *bit by bit* transfers of the C64 and VIC20, so making it feasible to emulate the parallel line completely while emulating the drive at DOS level only. The IEEE488 line interpreter maps the drives 8-11 (as described above) to the IEEE488 disk units, and no kernal traps are needed. The same emulation of the Commodore IEEE488 bus interface is available for the C64 and the VIC20. With IEEE488 drives you can have true 2031 emulation at unit #8, and still have filesystem access at units #10 or #11, because monitoring the IEEE488 lines does not interfere with the true drive emulation.

The IEEE488 disk drives 3040, 4040, 8050 and 8250 are Dual Drive Floppy Disks. This means that these drives handle two disks. To Accomplish the emulation, only one disk can be emulated, namely unit #8. The attached image, track display and LED display of unit #9 are used for the second drive of the dual disk drives. On unix the unit number display (8 or 9) in the emulation window changes to the drive number display (0 or 1).

The Commodore 3040, 4040, 1001, 8050 and 8250 disk drives are so-called "old-style" disk drives. Their architecture includes not one, but two processors of the 6502 type,

namely a 6502 for the file handling and communication with the PET (IP), and a 6504 (which is a 6502 with reduced address space) for the drive handling (FDC). Both processors communicate over a shared memory area. The IP writes commands to read/write blocks to this area and the FDC executes them. To make the emulation feasible, the FDC processor is not emulated cycle-exactly as a 6504, but simply by checking the commands and executing them on the host. This provides a fast FDC emulation, but disallows the sending the FDC processor commands to execute code. Applications where this is necessary are believed to be rather seldom. Only the format command uses this feature, but this is checked for.

The dual disk drive 2040 emulates one of the very first CBM disk drives. This drive has DOS version 1. DOS1 uses an own disk type, that is closely related to the 1541 disk image. Only on tracks 18-24 DOS1 disks have a sector more than 1541 disks. DOS1 disk images have the extension .d67.

The dual disk drives 3040 and 4040 use the same logical disk format as the VC1541 and the 2031. In fact, the 4040 was the first disk with DOS version 2. The 3040 emulated here originally was the same as 2040, only for the european 30xx PET series. As many of the original DOS1 disk drives were upgraded (a simple ROM upgrade!) to DOS2, I use the 3040 number for a DOS 2.0 disk drive, and 4040 for a revised DOS 2 disk drive. It is, however, not yet clear whether the disks here are write compatible to the 1541, as rumors exist that the write gap between sectors is different. But read compatible they are. As VICE emulates the FDC processor in C and not as 6504 emulation, this does not matter in VICE.

The drives 1001, 8050 and 8250 do actually have the very same DOS ROM. Only the code in the FDC is different, which is taken care of by VICE. So for all three of those disk drives, only `dos1001` is needed. The DOS version used is 2.7.

## 2.10 Supported file formats

VICE supports the most popular Commodore file formats:

- `X64` (preferred) or `D64` disk image files; Used by the 1541, 2031, 3040, 4040 drives.
- `G64` GCR-encoded 1541 disk image files;
- `D67` CBM2040 (DOS1) disk image format
- `D71` VC1571 disk image format
- `D81` VC1581 disk image format
- `D80` CBM8050 disk image format
- `D82` CBM8250/1001 disk image format
- `T64` tape image files (read-only);
- `P00` program files;

An utility (`c1541`, see Chapter 10 [c1541], page 99) is provided to allow transfers and conversions between these formats.

Notice that the use of the `X64` file format is depreciated now.

You can convert an `X64` file back into a `D64` file with the UNIX `dd` command:

```
dd bs=64 skip=1 if=IMAGE.X64 of=IMAGE.D64
```

See Chapter 13 [File formats], page 106. for a technical description of the supported file formats.

## 2.11 Common problems

This section tries to describe the most common known problems with VICE, and how to resolve them.

### 2.11.1 Sound problems

VICE should compile and run without major problems on many UNIX systems, but there are some known issues related to the sound driver. In fact, the sound code is the least portable part of the emulator and has not yet been thoroughly tested on all the supported platforms.

Linux, AIX and SGI systems should play sound without any problems; if you are running Linux please use a 2.x kernel, as VICE needs some features that were not implemented in older versions of the Linux sound driver.

On the other hand, HP-UX and Solaris machines are known to cause troubles. If you think you can help debugging the code for these systems, your help would be really appreciated. We are having troubles finding HP-UX and SUN consoles to work at...

Some problems have been reported with the proprietary version of the Open Sound System for Linux. With a Crystal sound card, sound output was significantly delayed and, apparently, the allocated buffer size was completely wrong. This is not a VICE bug, but rather an OSS bug.

### 2.11.2 Shared memory problems

If you cannot start VICE because you get errors about shared memory, try to run it with the '`+mitshm`' command-line option (see Section 6.4.2 [Video options], page 28). This will completely disable usage of the MITSHM extensions, that are normally used to speed up the emulation window updates. Of course, this will also result in a big loss in speed.

Reasons for this failure could be:

- IPC support has been disabled at the system level; some system administrators disable this for security reasons. If *you* are the system administrator, use a kernel that has IPC support compiled in and enabled.
- You are attempting to run the emulator across the network (i.e., the emulator runs on one machine, and the output is displayed on another machine that works as an X terminal) and for some reason VICE does not recognize this fact. In this case, you have found a bug, so please report it to us.

If you want to avoid running the emulator with '`+mitshm`' every time, run it once with '`+mitshm`' and then choose "Save settings" from the right-button menu.

### 2.11.3 Printer problems

VICE supports the emulation of a printer either on the userport or as IEC device 4. Unfortunately the Commodore IEC routines do not send all commands to the IEC bus. For example an `OPEN 1,4` is not seen on the IEC bus. Also a `CLOSE 1` after that is not seen. VICE can see from printing that there was an `OPEN`, but it cannot see when the close was. Also a "finish print job" cannot be seen on the userport device. To flush the printer buffer (write to `print.dump` or to the printer) now a menu entry can be used. Disabling and re-enabling the printer should work as well.

The printing services have not been extensively tested but apart from the problem mentioned above it should work fine now.

### 2.11.4 PET keyboard problems

If you find that the German keyboard mapping (plus German charset) does not print uppercase umlauts, then you are right. The umlauts replace the [,\ and ] characters in the charset. The keys that make these characters do not have a different entry in the PET editor ROM tables when shifted. Thus it is not possible to get the uppercase umlauts in the editor. Nevertheless other programs are reported to change the keyboard mapping table and thus allow the use of the shifted (uppercase) umlauts.

Anyway, the VICE keyboard mappings are far from being perfect and we are open to any suggestions.

# 3 Invoking the emulators

The names of the available emulators are:

- `x64`, the fast C64 emulator
- `x64sc`, the accurate C64 emulator
- `x64dtv`, the C64DTV emulator
- `x128`, the C128 emulator
- `xvic`, the VIC20 emulator
- `xpet`, the PET emulator
- `xplus4`, the PLUS4 emulator
- `xcbm2`, the CBM-II emulator

You can run each of them by simply typing the name from a shell. If you want to run them from another application (e.g., a window manager or some other sort of program launcher) you should always run them from a terminal window such as `xterm` or `rxvt` since VICE provides a lot of debugging information that is sent to the terminal and has built-in monitor that also appears there. For example, you could do

```
xterm -e x64
```

## 3.1 Command-line options used during initialization

There are several options you can specify on the command line. Some of them are used to specify emulation settings and will be described in detail later (see Chapter 6 [Settings and resources], page 25 for a complete list). The remaining options are used only to give usage information or to initialize the emulator in some way:

`-help`
`-?` List all the available command-line options and their meaning.

`-default` Set default resources (see Chapter 6 [Settings and resources], page 25). This will override all the settings specified before, but not the settings specified afterwards on the command line.

`-autostart IMAGE`
: Autostart '`IMAGE`' (see Section 3.2 [Command-line autostart], page 15).

`-autoload <name>`
: Attach and autoload tape/disk image <name>

`-basicload`
: On autostart, load to BASIC start (without ',1')

`+basicload`
: On autostart, load with ',1'

`-autostartwithcolon`
: On autostart, use the 'RUN' command with a colon, i.e., 'RUN:'

`+autostartwithcolon`
: On autostart, do not use the 'RUN' command with a colon; i.e., 'RUN'

`-autostart-handle-tde`
`+autostart-handle-tde`
: Handle/Do not handle True Drive Emulation on autostart

`+autostart-warp`
: Enable/Disable warp mode during autostart

`-autostartprgmode`
: Set autostart mode for PRG files

`-autostartprgdiskimage`
: Set disk image for autostart of PRG files

`-1 NAME` Attach '`NAME`' as a tape image file.

`-8 NAME`
`-9 NAME`
`-10 NAME`
`-11 NAME` Attach '`NAME`' as a disk image to device 8, 9, 10 or 11.

`-attach8ro`
`-attach9ro`
`-attach10ro`
`-attach11ro`
: Attach disk image for drive #88-11 read only

`-attach8rw`
`-attach9rw`
`-attach10rw`
`-attach11rw`
: Attach disk image for drive #8-11 read write (if possible)

## 3.2 Autostarting programs from the command-line

It is possible to let the emulator *autostart* a disk or tape image file, by simply specifying its name as the *last* argument on the command line, for example

```
x64 lovelygame.x64.gz
```

will start the C64 emulator, attaching '`lovelygame.x64.gz`' as a disk image and running the first program on it. You can also specify the name of the program on the fisk image by appending a colon ('`:`') the name itself to the argument; for example

```
x64 "lovelygame.x64.gz:run me"
```

will run the program named '`run me`' on '`lovelygame.x64.gz`' instead of the first one.

Using the command-line option `-autostart` is equivalent; so the same result can be obtained with

```
x64 -autostart "lovelygame.x64.gz:run me"
```

If you specify a raw CBM or P00 file, the emulator will setup the file system based drive emulation so that it is enabled and accesses the directory containing the file first. This is a very convenient way to start multi-file programs stored in file system directories and not requiring "true" drive emulation.

See Section 5.5 [Disk and tape images], page 22. for more information about images and autostart.

# 4 System files

In order to work properly, the emulators need to load a few system files:

- the *system ROMs*, raw binary files containing copies of the original ROMs of the machine you are emulating;
- the *keyboard maps*, text files describing the keyboard layout;
- the *palette files*, text files describing the colors of the machine you are emulating.
- the *romset files*, text files describing the different ROMs to load.

The place where they will be searched for depends on the value of the `Directory` resource, which is a colon (`:`)-separated search path list, like the UNIX `PATH` environment variable. The default value is

```
PREFIX/lib/vice/EMU:$HOME/.vice/EMU:BOOTPATH/EMU
```

Where `PREFIX` is the installation prefix (usually '`/usr/local`'), `EMU` is the name of the emulated machine (`C64`, `C128`, `PET`, `CBM-II` or `VIC20`) and `BOOTPATH` is the directory where the executable resides. The disk drive ROMs are looked for in a directory with `EMU` set to `DRIVES`. `$HOME` is the user's home directory.

For example, if you have the C64 emulator installed in

```
/usr/local/bin/x64
```

then the value will be

```
/usr/local/lib/vice/C64:$HOME/.vice/C64:/usr/local/bin/C64
```

And system files will be searched for under the following directories, in the specified order:

1. `/usr/local/lib/VICE/C64`
2. `$HOME/.vice/C64`
3. `/usr/local/bin/C64`

System files can still be installed in a different directory if you specify a complete path instead of just a file name. For example, if you specify '`./kernal`' as the kernal image name, the kernal image will be loaded from the current directory. This can be done by using command-line options or by modifying resource values (see Section 6.1 [Resource files], page 25).

## 4.1 ROM files

Every emulator requires its own ROM set. For the VIC20 and the C64, the ROM set consists of the following files:

- '`kernal`', the Kernal ROM (8 KBytes)
- '`basic`', the Basic ROM (8 KBytes)
- '`chargen`', the character generator ROM (4 Kbytes)

The C128 needs the following files:

- '`kernal`', the Kernal ROM (8 Kbytes)
- '`basic`', the Basic + Editor ROM (32 Kbytes)
- '`chargen`', the character generator ROM (4 Kbytes)

The C128, VIC20 and C64 emulators also need the following DOS ROMs for the hardware-level emulation of the 1541, 1571 and 1581 disk drives:

- '`dos1541`', the 1541 drive ROM (16 Kbytes)
- '`dos1541II`', the 1541-II drive ROM (16 Kbytes)
- '`dos1571`', the 1571 drive ROM (32 Kbytes)
- '`dos1581`', the 1581 drive ROM (32 Kbytes)

In addition to those all emulators can handle a parallel IEEE488 interface (the C64 and C128 via `$df**` extension, the VIC20 via VIC1112 emulation) so they also need the DOS ROM for the IEEE disk drives:

- '`dos2031`', the 2031 drive ROM (16 Kbytes) (DOS 2.6, Commodore ROM images 901484-03 and 901484-05)
- '`dos2040`', the 2040 drive ROM (8 Kbytes) (DOS 1, Commodore ROM images 901468-06, 901468-07)
- '`dos3040`', the 3040 drive ROM (12 Kbytes) (DOS 2, Commodore ROM images 901468-11, 901468-12 and 901468-13)
- '`dos4040`', the 4040 drive ROM (12 Kbytes) (DOS 2, Commodore ROM images 901468-14, 901468-15 and 901468-16)
- '`dos1001`', the 1001/8050/8250 drive ROM (16 Kbytes) (DOS 2.7, Commodore ROM images 901887-01 and 901888-01)

Note that there are other DOS images on the internet. The DOS 2.5 images might be used with the 8050, but it cannot handle the double sided drives of the 1001 and 8250 and it is not supported by VICE.

The PET emulator uses an expanded setup, because there are three major versions of the Basic and the Kernal, and many versions of the Editor ROM. In addition there are cartridge ROM sockets.

The Kernal files contain the memory from range $F000-$FFFF, the Basic ROMs either the range $C000-$DFFF or $B000-$DFFF. To handle the different screen sizes and keyboards, different so-called "editor-ROMs" for the memory range $E000-$E800 are provided. The PET ROMs have the following names:

- '`kernal1`', the PET2001 Kernal ROM (4 KBytes) (Commodore ROM images 901447-06 and 901447-07)
- '`kernal2`', the PET3032 Kernal ROM (4 KBytes) (Commodore ROM image 901465-03)
- '`kernal4`', the PET4032/8032 Kernal ROM (4 KBytes) (Commodore ROM image 901465-22)
- '`basic1`', the PET2001 Basic 1 ROM (8 KBytes) (Commodore ROM images 901447-09, 901447-02, 901447-03, 901447-04.bin. The -09 ROM is the revised -01 ROM)
- '`basic2`', the PET3032 Basic 2 ROM (8 KBytes) (Commodore ROM images 901465-01 and 901465-01)
- '`basic4`', the PET4032/8032 Basic 4 ROM (12 KBytes) (Commodore ROM images 901465-23, 901465-20 and 901465-21. The -23 ROM is a revised -19 ROM)
- '`edit1g`', the PET2001 editor for graphics keyboards (2 KBytes) (Commodore ROM image 901447-05)
- '`edit2b`', the PET3032 editor for business keyboards (2 KBytes) (Commodore ROM image 901474-01)
- '`edit2g`', the PET3032 editor for graphics keyboards (2 KBytes) (Commodore ROM image 901447-24)
- '`edit4g40`', the PET4032 editor for graphics keyboards (2 KBytes) (Commodore ROM image 901498-01)
- '`edit4b40`', the PET4032 editor for business keyboards (2 KBytes) (Commodore ROM image 901474-02)
- '`edit4b80`', the PET8032 editor for business keyboards (2 KBytes) (Commodore ROM image 901474-04-?)
- 
- '`chargen`', the character generator ROM (2k). It has two sets with 128 chars each. The second (inverted) half of each set is computed from the first half by inverting it. This is a PET hardware feature. (Commodore ROM image 901447-10)
- '`chargen.de`', the character generator ROM (2k). This version is a patched German charset, with the characters [, \ and ] replaced by umlauts. It has been provided by U. Guettich and he reports that it is supported by some programs.

The PETs also have sockets for extension ROMs for the addresses $9000-$9FFF, $A000-$AFFF and $B000-$BFFF (the last one for PET2001 and PET3032 only). You can specify ROM image files for those extensions command line options `-petrom9`, `-petromA` and `-petromB` resp.

An alternative would be to specify a long kernal ROM with the `-kernal` option that includes the extension ROM areas.

Also, you can specify replacements for the basic ROM at $B000-$DFFF with the `-petromBasic` option and for the editor ROM at $E000-$E7FF with the `-petromEditor` option.

The CBM-II emulator again uses another setup. For those models the kernal used is the same for all. However, for different amounts of memory exist different versions of the BASIC ROMs. The 128k RAM version (C610, C710, B128) uses one bank of 64k for the BASIC text and another one for all the variables. The 256k RAM version uses one bank for text, one for variables, one for arrays and one for strings.

Also the character generator ROMs have a format different from the above. The other character ROMs have 8 bytes of pixel data per character. Those ROMs have 16 bytes per character instead. The C6x0 only uses the first 8 of it, but the C7x0 uses 14 lines per character and needs those increased ROMs. Both ROMs hold, like the PET, two character sets with 128 characters each. Again the second half of the full (256 char) character set is computed by inverting.

- '`kernal`', the KERNAL (8k) for the business machines (6xx/7xx)
- '`kernal.500`', the KERNAL (8k) for the personal machine (510) (901234-02)
- '`basic.128`', the CBM-II 128k BASIC (16k)
- '`basic.256`', CBM-II 256k BASIC (16k)
- '`basic.500`', C510 BASIC (16k) (901236-02 + 901235-02)
- '`chargen.500`', character generator ROM for the C5x0 (4k) (901225-01)
- '`chargen.600`', character generator ROM for the C6x0 (4k)
- '`chargen.700`', character generator ROM for the C7x0 (4k)

## 4.2 Keymap files

*Keymap files* are used to define the keyboard layout, defining which key (or combination of keys) must be mapped to each keysym.

In other words, the keyboard emulation works like this: whenever the user presses or releases a key while the emulation window has the input focus, the emulator receives an X-Window event with a value that identifies that key. That value is called a *keysym* and is unique to that key. The emulator then looks up that keysym in an internal table that tells it which key(s) to press or release on the emulated keyboard.

This table is described by the keymap file, which is made up of lines like the following:

```
KEYSYM ROW COLUMN SHIFTFLAG
```

Where:

- `KEYSYM` is a string identifying the keysym: you can use the `xev` utility (shipped with the X Window system) to see what keysym is bound to any key;
- `ROW` and `COLUMN` identify the key on the emulated keyboard;
- `SHIFTFLAG` can have one of the following values:
  - `0`: the key is never shifted;
  - `1`: the key is shifted;
  - `2`: the key is the left shift;
  - `4`: the key is the right shift;
  - `8`: the key can be (optionally) shifted by the user.

The SHIFTFLAG is useful if you want certain keys to be "artificially" shifted by the emulator, and not by the user. For example, F2 is shifted on the C64 keyboard, but you might want it to be mapped to the unshifted F2 key on the PC keyboard. To do so, you just have to use a line like the following:

```
F2 0 4 1
```

where 0 and 4 identify the key (row 0, column 4 on the keyboard matrix), and 1 specifies that every time the user presses F2 the shift key on the C64 keyboard must be pressed.

There are also some special commands you can put into the keyboard file; they are recognized because they start with an exclamation mark:

- !CLEAR clears the currently loaded keyboard map; it is necessary to put this at the beginning of the file if you want the keymap file to override all of the current internal settings;
- !LSHIFT, !RSHIFT, followed by a row and a column value, specify where the left and right shift keys are located on the emulated keyboard; for example, C64 default keymaps will specify

  ```
  !LSHIFT 1 7
  !RSHIFT 6 4
  ```

Any line starting with the # sign, instead, is completely ignored. This is useful for adding comments within the keymap file.

VICE keymap files have the '.vkm' default extension, and every emulator comes with a default positional mapping and a default symbolic mapping.

## 4.3 Palette files

*Palette files* are used to specify the colors used in the emulators. They are made up of lines like the following:

```
RED GREEN BLUE DITHER
```

where RED, GREEN and BLUE are hexadecimal values ranging from 0 to FF and specifying the amount of red, green and blue you want for each color and DITHER is a 4-bit hexadecimal number specifying the pattern you want when rendering on a B/W display.

You have to include as many lines as the number of colors the emulated machine has, and the order of the lines must respect the one used in the machine (so the N'th line must contain the specifications for color N - 1 in the emulated machine).

Lines starting with the # sign are completely ignored. This is useful for adding comments (such as color names) within the palette file.

For example, the default PET palette file (which has only two colors, 0 for background and 1 for foreground), looks like the following:

```
#
# VICE Palette file
#
# Syntax:
# Red Green Blue Dither
#
```

```
# Background
00 00 00 0

# Foreground
00 FF 00 F
```

## 4.4 Romset files

The Romset files are not used by default on all emulators. You might have recognized that the names of the ROM images are saved in resources. Loading a Romset file now just means a 'shortcut' to changing all the resources with ROM image names and reloading the ROMs.

The PET and CBM-II emulators use this feature to change between the different ROM versions available for those machines. E.g. the Romset file for the PET 2001 is

```
KernalName="pet2001"
EditorName=
ChargenName="chargen"
RomModule9Name=
RomModuleAName=
RomModuleBName=
```

As you can see, the file even uses the same syntax as the resource file, it is just a bit stripped down.

# 5 Basic operation

This section describes the basic things you can do once the emulator has been fired up.

## 5.1 The emulation window

When the emulator is run, the screen of the emulated machine is displayed in a standard X Window which we will call the *emulation window*. This window will be updated in real time, displaying the same contents that a real monitor or TV set would.

Below the emulation window there is an area which is used to display information about the state of the emulator; we will call this area the *status bar*.

On the extreme left of the status bar, there is a *performance meter*. This displays the current relative speed of the emulator (as a percentage) and the update frequency (in frames per second). All the machines emulated are PAL, so the update frequency will be 50 frames per second if your system is fast enough to allow emulation at the speed of the real machine.

On the extreme right of the status bar, there is a *drive status indicator*. This is only visible if the hardware-level ("True") 1541 emulation is turned on. In that case, the drive status indicator will contain a rectangle emulating the drive LED and will display the current track position of the drive's read/write head.

## 5.2 Using the menus

It is possible to execute some commands and change emulation parameters while the emulator is running: when the pointer is over the emulation window, two menus are available

by pressing either the left or right mouse buttons. The left mouse button will open the *command menu* from which several emulation-related commands can be executed; the right mouse button will open the *settings menu* from which emulation parameters can be changed. The basic difference between the command and the settings menu is that, while commands have only effect on the current session, settings can be saved and later used with the "Save settings" and "Load settings" right-button menu items, respectively. "Restore default settings" restores the factory defaults. See Chapter 6 [Settings and resources], page 25. for more information about how settings work in VICE.

Sometimes commands can be reached via *shortcuts* or *hotkeys*, i.e., it is possible to execute them by pressing a sequence of keys instead of going through the menu with the mouse. Where shortcuts exist, they are displayed in parentheses at the right edge of the menu item. In VICE, all shortcuts must begin with the META or ALT key. So, for example, to attach a disk image to drive #8 (the corresponding menu item displays "M-8"), you have to press the META (or ALT) and then 8.

Note that no other key presses are passed on to the emulated machine while either META or ALT are held down.

## 5.3 Getting help

At any time, if you get stuck or do not remember how to perform a certain action, you can use the "Browse manuals" command (left button menu). This will popup a browser and open the HTML version of this documentation. Notice that this requires VICE to be properly (and fully) installed with a '`make install`'.

The browser can be specified via the `HTMLBrowserCommand` string resource (see Chapter 6 [Settings and resources], page 25 for information about resources). Every '`%s`' in the string will be replaced with a URL to the VICE HTML pages.

## 5.4 Using the file selector

In those situations where it is necessary to specify a file name, all of the VICE emulators will pop up a file selector window allowing you to select or specify a file interactively.

To the left of the file selector, there is a list of ancestor directories: by clicking on them, you can ascend the directory tree. To the right, there is a list of the files in the current directory; files can be selected by clicking on them. If you click on a directory, that directory becomes the current one; if you click on an ordinary file, it becomes the active selection.

At the top, there is a *directory box*, with the complete path of the current directory, and a *file name box*, with the name of the currently selected file. At the bottom there are two buttons: "OK" confirms the selected file and "Cancel" abandons the file selector without cancelling the operation.

It is also possible to specify what files you want to show in the file selector by writing an appropriate shell-like pattern in the directory box; e.g., '`~/*.[dx]64`' will only show files in the home directory whose name ends with either '`.d64`' or with '`.x64`'.

## 5.5 Using disk and tape images

The emulator is able to emulate disk drives and (read-only) tape recorders if provided with suitable *disk images* or *tape images*. An *image* is a raw dump of the contents of the media,

and must be *attached* before the emulator can use it. "Attaching" a disk or tape image is like "virtually" inserting a diskette or a cassette into the disk drive or the tape recorder: once an image is attached, the emulator is able to use it as a storage media.

There are five commands (in the left button menu) that deal with disk and tape images:

- Attach Disk Image
- Detach Disk Image
- Attach Tape Image
- Detach Tape Image
- Smart-attach a file

The first four commands are used to insert and remove the virtual disks and cassettes from the respective units. On the other hand, the last commands tries to guess the type of the image you are attaching from its name and size, and attaches it to the most reasonable device.

Supported formats are `D64` and `X64` for disk images (devices 8, 9 and 10) and `T64` for tape images. Notice that `T64` support is *read-only*, and that the cassette is automatically rewound when you reach its end.

Another important feature is that raw Commodore BASIC binary files and .P00 files can be attached as tapes. As you can autostart a tape image when it is attached (see Section 5.5.2 [Autostart], page 23), this allows you to autostart these particular files as well.

You can attach a disk for which you do not have write permissions: when this happens, the 1541 emulator will emulate a write-protected disk. This is also useful if you want to prevent certain disk images from being written to; in the latter case, just remove the write permission for that file, e.g., by doing a `chmod a-w`.

### 5.5.1 Previewing the image contents

It is possible to examine the directory of a disk or tape image before attaching it. Just press the "Contents" button in the file selector window and a new window will pop up with the contents of the selected image.

Notice that this function automatically translates the directory from PETSCII to ASCII; but, due to differences in the two encodings, it is not always possible to translate all the characters, so you might get funny results when "weird" characters such as the semi-graphical ones are being used.

### 5.5.2 "Autostarting" an image

If you want to reset the machine and run the first program on a certain image without typing any commands at the Commodore BASIC prompt, you can use the "Autostart" button in the file selector window after selecting a proper disk or tape image file.

Notice that, if true drive emulation is turned on, it will be turned off before running the program and then turned on again after it has been loaded. This way, you get the maximum possible speed while loading the file, but you do not lose compatibility once the program itself is running.

This method is not completely safe, because some autostarting methods might cause the true drive emulation not to be turned on again. In such cases, the best thing to do is to

disable kernal traps (which will cause true drive emulation to be always kept turned on), or to manually load the program with true drive emulation turned on.

### 5.5.3 Using compressed files

It is also possible to attach disk or tape images that have been compressed through various algorithms; compression formats are identified from the file extension. The following formats are supported (the expected file name extension is in parenthesis):

- GNU Zip (`.gz` or `.z`);
- BZip version 2 (`.bz2`);
- PkZip (`.zip`);
- GNU Zipped TAR archives (`.tar.gz`, `.tgz`);
- Zoo (`.zoo`).

PkZip, `tar.gz`, `lha` and `zoo` support is *read-only* and always uses the *first* `T64` or `D64` file in the archive. So archives containing multiple files will always be handled as if they contain only a single file.

Windows and MSDOS don't contain the needful programs to handle compressed archives. Get gzip and unzip for Windows at `ftp://ftp.freesoftware.com/pub/infozip/WIN32` and for MSDOS at `ftp://ftp.freesoftware.com/pub/infozip/MSDOS`. Don't use pkunzip for MSDOS, it doesn't work. The programs to use BZip2 archives may be found at `http://sourceware.cygnus.com/bzip2`. Just put the programs (unzip.exe, gzip.exe, bzip2.exe) into a directory of your search path (e.g. C:\DOS or C:\WINDOWS\COMMAND; have a look at the PATH variable).

### 5.5.4 Using Zipcode and Lynx images

Since version 0.15, the VICE emulators have been able to attach disks packed with Zipcode or Lynx directly, removing the need to manually convert them into `D64` or `X64` files with `c1541`. This is achieved by automatically invoking `c1541`, letting it decode the file into a temporary image and attaching the resulting temporary image read-only. For this to work, the directory containing `c1541` must be in your `PATH`.

This uses the `-unlynx` and `-zcreate` options of `c1541` (see Section 10.3 [c1541 commands and options], page 100); these commands are not very reliable yet, and could fail with certain kinds of Lynx and Zipcode images (for example, they cannot deal with `DEL` files properly). So please use them with caution.

Lynx files usually come as '`.lnx`' files which are unpacked into single disk images. On the other hand, Zipcode files do not have a particular extension (although '`.z64`' is sometimes used), and represent a disk by means of component files, named as follows:

- '`1!NAME`'
- '`2!NAME`'
- '`3!NAME`'
- '`4!NAME`'

If you attach as a disk image (or smart-attach) any one of these files, the emulator will simply pick up the other three (by examining the name) and then build a disk image using all four.

## 5.6 Resetting the machine

You can reset the emulated machine at any time by using the "Reset" command from the command menu. There are two types of reset:

- *soft reset*, which simply resets the CPU and all the other chips;
- *hard reset*, which also clears up the contents of RAM.

A *soft reset* is the same as a hardware reset achieved by pulling the RESET line down; a *hard reset* is more like a power on/power off sequence in that it makes sure the whole RAM is cleared.

It is possible that a soft reset may not be enough to take the machine to the OS initialization sequence: in such cases, you will have to do a hard reset instead.

This is especially the case for the CBM-II emulators. Those machines examine a memory location and if they find a certain `"magic"` value they only do what you know from the C64 as `Run/Stop-Restore`. Therefore, to really reset a CBM-II use hard reset.

# 6 Settings and resources

In the VICE emulators, all the settings are stored in entities known as called *resources*. Each resource has a name and a value which may be either an integer or a string. Integer values are often used as boolean values with the usual convention of using zero for "false" and any other value for "true".

Resource values can be changed via the right-button menu (the *settings* menu), via command-line options or via the *resource file*.

The *resource file* is a human-readable file containing resource values: it is called '`vicerc`' and is stored in the directory '`.vice/`' in the user's home directory. It is possible to dump the current values of the resources into that file or load the values stored into that file as the current values, at any time. This is achieved with the "Save settings" and "Load settings" right menu items. A third menu item, "Restore Default Settings", can be used to reset all the values to the factory defaults.

A special resource, `SaveResourcesOnExit`, if set to a non zero value, causes the emulator to ask you if you want to save the current (changed) settings before exiting, and can be toggled with the "Save settings on exit" command from the right-button menu.

Notice that not all the resources can be changed from the menus; some of them can only be changed by manually modifying the resource file or by using command-line options.

## 6.1 Format of resource files

A resource file is made up of several sections; sections have the purpose of separating the resources of a certain emulator from the ones of the other emulators. A section starts with the name of an emulator in brackets (e.g., '`[C64]`') and ends when another section starts or when the file ends.

Every line in a section has the following format:

```
RESOURCE=VALUE
```

where `RESOURCE` is the name of a resource and `VALUE` is its assigned value. Resource names are case-sensitive and resource values are either strings or integers. Strings must

start and end with a double quote character ("), while integers must be given in decimal notation.

Here is an example of a stripped-down '`.vice/vicerc`' file:

```
[VIC20]
HTMLBrowserCommand="netscape %s"
SaveResourcesOnExit=0
FileSystemDevice8=1
FSDevice8ConvertP00=1
FSDevice8Dir="/home/ettore/cbm/stuff/vic20p00"
FSDevice8SaveP00=1
FSDevice8HideCBMFiles=1
[C64]
HTMLBrowserCommand="netscape %s"
SaveResourcesOnExit=1
FileSystemDevice8=1
FSDevice8ConvertP00=1
FSDevice8Dir="/home/ettore/cbm/stuff/c64p00"
FSDevice8SaveP00=1
FSDevice8HideCBMFiles=1
```

Notice that, when resource values are saved with "Save settings", the emulator only modifies its own section, leaving the others unchanged.

## 6.2 Using command-line options to change resources

Resources can also be changed via command-line options.

Command-line options always override the defaults from `.vice/vicerc`, and their assignments last for the whole session. So, if you specify a certain command-line option that changes a certain resource from its default value and then use "Save Settings", the value specified with the command-line option will be saved back to the resource file.

Command-line options can begin with with a minus sign ('`-`') or with a plus sign ('`+`'). Options beginning with a minus sign may require an additional parameter, while the ones beginning with the plus sign never require one.

Moreover, options beginning with a plus sign always have a counterpart with the same name, but with a minus sign; in that case, the option beginning with a minus sign is used to *enable* a certain feature, while the one beginning with a plus sign is used to *disable* the same feature (this is an X11 convention). For example, `-mitshm` enables support of MITSHM, while `+mitshm` disables it.

## 6.3 Performance settings

It is possible to control the emulation speed by using the "Maximum speed" menu item in the right-button menu. The default setting is `100`, which causes the emulation to never run faster than the real machine. A higher value allows the emulator to run faster, a lower one may force it to run slower. The setting "No limit" means to run as fast as possible, without limiting speed.

It is also possible to control the emulator's rate of frame update using the "Refresh rate" setting; the value ranges from "1/1" (update 1/1 of the frames of the real machine, that

is 50 frames per second) to "1/10" (update 1 every 10 frames) and can be changed via the "Refresh Rate" submenu. The "Auto" setting means to dynamically adapt the refresh rate to the current speed of the host machine, making sure the maximum speed specified by the via "Maxium speed" is always reached if possible. In any case, the refresh rate will never be worse than 1/10 if this option is specified.

Note that you cannot simultaneously specify "Auto" as the refresh rate and "No limit" as the maximum speed..

Moreover, a special *warp speed* mode is provided and can be toggled with the "Enable Warp Mode" menu item. If this mode is enabled, it will cause the emulator to disable any speed limit, turn sound emulation off and use a 1/10 refresh rate, so that it will run at the maximum possible speed.

### 6.3.1 Performance resources

`Speed` Integer specifying the maximum relative speed, as a percentage. `0` stands for "no limit".

`RefreshRate`
Integer specifying the refresh rate; a value of `n` specifies a refresh rate of 1/`n`. A value of `0` enables automatic frame skipping.

`WarpMode` Booolean specifying whether "warp mode" is turned on or not.

### 6.3.2 Performance command-line options

`-speed VALUE`
Specifies the maximum speed as a percentage. `0` stands for "no limit". (Same as setting the `Speed` resource.)

`-refresh VALUE`
Specifies refresh rate; a value of `n` specifies a refresh rate of 1/`n`. A value of `0` enables automatic frame skipping. (Same as setting the `RefreshRate` resource.)

`-warp`
`+warp` Enables/disables warp mode (`WarpMode=1`, `WarpMode=0`).

## 6.4 Video settings

The following right-button menu items control the video output. On emulators that include two video chips (like `x128`) all options but XSync exist twice, once for each chip. XSync is shared between the video chips.

- "Video Cache" enables a video cache that can speed up the emulation when little graphics activity is going on; it is especially useful when you run the emulator on a networked X terminal as it can reduce the network bandwidth required. However, this setting can actually make the emulator slower when there is little graphics activity and the amount of work needed to maintain the cache is greater than the amount of work that would be wasted by not using it (if any).
- "Double Size" toggles *double-size mode*, which makes the emulation window twice as big. When emulating a 80-column PET, only the height is doubled, so that the aspect ratio is closer to that of the real thing.

- "Double Scan" toggles *double-scan mode*, which causes the emulator to draw only odd lines when running in double-size mode (this saves some CPU time and also makes the emulation window look more like an old monitor).
- 'Use XSync()" causes the emulator to call the X11 function `XSync()` before updating the emulation window: this might be necessary on low-end systems to prevent it from consuming so many system resources that it becomes impossible for the user to interact with it.

### 6.4.1 Video resources

The following resources affect the screen emulation. The prefix of some of the resources and commandline options denote the video chip the values apply to.

`UseXSync` Boolean specifying whether `XSync()` is called after updating the emulation window.

`MITSHM` Integer specifying whether VICE should try to use the shared memory extensions (MITSHM) when starting up. The shared memory extensions make things a lot faster but might not be available on your system. You will not be able to use these extensions if you are sitting at an X terminal while running the emulator on a remote machine across a network. Valid values are: 0 = do not use MITSHM, 1 = do use MITSHM, -1 = try to autodetect availability on startup (default). The last is a simple test if the emulator runs across a network and if so disables MITSHM (If you have problems with this test please report it).

`PrivateColormap`
Boolean specifying whether VICE should install a private colormap at startup. This makes sense for 8-bit displays that could run out of colors if other color-hungry applications are running at the same time.

`DisplayDepth`
Integer specifying the depth of the host display. The value '`0`' (the default) causes the emulator to autodetect it.

### 6.4.2 Video command-line options

`-xsync`
`+xsync` Enable/disable usage of `XSync()` when updating the emulation window (`UseXSync=1`, `UseXSync=0`).

`-mitshm`
`+mitshm` Enable/disable usage of the MITSHM extensions (`MITSHM=1`, `MITSHM=0`).

`-install`
`+install` Enable/disable installation of a private colormap (`PrivateColormap=1`, `PrivateColormap=0`).

`-displaydepth DEPTH`
Specify the display depth (`DisplayDepth`).

## 6.5 Keyboard settings

It is possible to specify whether the "positional" or "symbolic" keyboard mapping should be used with the "Keyboard mapping type" submenu (see Section 2.7 [Keyboard emulation], page 9 for an explanation of positional and symbolic mappings).

The keyboard settings submenu also allows you to:

- Load custom-made positional and symbolic keymap files ("Set symbolic keymap file" and "Set positional keymap file").
- Dump the current keymap to a user-defined keymap file ("Dump to keymap file").

### 6.5.1 Keyboard resources

`KeymapIndex`
: Integer identifying which keymap is being used; `0` indicates symbolic mapping, `1` positional mapping. For the PET the even values represent symbolic mapping, odd positional. Then add `0` for UK business keyboard or `2` for graphics keyboard.

`KeymapSymFile`
: String specifying the name of the keymap file for the symbolic mapping (see Section 2.7 [Keyboard emulation], page 9, all but PET and CBM-II).

`KeymapPosFile`
: String specifying the name of the keymap file for the positional mapping (see Section 2.7 [Keyboard emulation], page 9, all but PET and CBM-II).

`KeymapBusinessUKSymFile`
`KeymapBusinessUKPosFile`
: String specifying the name of the keymap file for the symbolic and positional mapping for the UK business keyboard (see Section 2.7 [Keyboard emulation], page 9, PET and CBM-II).

`KeymapGraphicsSymFile`
`KeymapGraphicsPosFile`
: String specifying the name of the keymap file for the symbolic and positional mapping for the graphics keyboard (see Section 2.7 [Keyboard emulation], page 9, PET only).

`KeymapBusinessDESymFile`
`KeymapBusinessDEPosFile`
: String specifying the name of the keymap file for the symbolic and positional mapping for the German business keyboard. (see Section 2.7 [Keyboard emulation], page 9, PET only).

### 6.5.2 Keyboard command-line options

`-keymap N` Specifies which keymap is being used; `0` indicates symbolic mapping, `1` positional mapping (as for the `KeymapIndex` resource).

`-symkeymap NAME`
: Specify '`NAME`' as the symbolic keymap file (`KeymapSymFile`).

`-poskeymap NAME`
: Specify '`NAME`' as the positional keymap file (`KeymapPosFile`).

`-buksymkeymap NAME`
`-bukposkeymap NAME`
: Specify '`NAME`' as the symbolic/positional keymap file for the UK business keyboard (`KeymapBusinessUKSymFile`, `KeymapBusinessUKPosFile`, PET and CBM-II).

`-grsymkeymap NAME`
`-grposkeymap NAME`
: Specify '`NAME`' as the symbolic/positional keymap file for the graphics keyboard (`KeymapGraphicsSymFile`, `KeymapGraphicsPosFile`, PET only).

`-bdesymkeymap NAME`
`-bdeposkeymap NAME`
: Specify '`NAME`' as the symbolic/positional keymap file for the German business keyboard (`KeymapBusinessDESymFile`, `KeymapBusinessDEPosFile`, PET only).

## 6.6 Sound settings

The following menu items control sound output:

- "Enable sound playback" turns sound emulation on and off.
- "Sound synchronization" specifies the method for syncronizing the sound playback. Possible settings are:
  - "Flexible", i.e., the audio renderer flexibly adds/removes samples to the output to smoothly adapt the playback to slight changes in the speed of the emulator.
  - "Adjusting" works like "flexible", but supports bigger differences in speed. For example, if the emulation speed drops down from from 100% to 50%, audio slows down by the same amount too.
  - "Exact", instead, makes the audio renderer output always the same sounds you would hear from the real thing, without trying to adapt the ratio; to compensate the tolerances in speed, some extra frames will be skipped or added.
- "Sample rate" specifies the sampling frequency, ranging from 8000 to 48000 Hz (not all the sound cards and/or sound drivers can support all the frequencies, so actually the nearest candidate will be chosen).
- "Buffer size" specifies the size of the audio buffer; the bigger the buffer, the longer the delay with which sounds are played. You should pick the smallest value your machine can handle without problems.
- "Sound suspend time", will cause the audio playback to pause for the specified number of seconds whenever some clicking happens. If "Keep going" is selected, no pausing is done.
- "Oversample" specifies an oversampling factor, from 1 to 8 times (warning: this eats CPU cycles!).

### 6.6.1 Sound resources

`Sound` Boolean specifying whether audio emulation is turned on.

`SoundSpeedAdjustment`
Integer specifying what speed adjustment method the audio renderer should use. Possible values are:

- `0`: "flexible"
- `1`: "adjusting"
- `2`: "exact"

`SoundSampleRate`
Integer specifying the sampling frequency, ranging from 8000 to 48000 Hz (not all the sound cards and/or sound drivers can support all the frequencies, so actually the nearest candidate will be chosen).

`SoundBufferSize`
Integer specifying the size of the audio buffer, in milliseconds.

`SoundSuspendTime`
Integer specifying the pause interval when audio underflows ("clicks") happen. `0` means no pause is done.

`SoundOversample`
Integer specifying the oversampling factor, ranging from `0` (no oversampling) to `3` (8 times oversampling).

`SoundDeviceName`
String specifying the audio driver.

Implemented drivers are:

- `aix`, for the IBM AIX sound driver.
- `uss`, for the Linux/FreeBSD Universal Sound System driver (`SoundDeviceArg` specifies the audio device, '`/dev/dsp`' by default);
- `sgi`, for the Silicon Graphics audio device (`SoundDeviceArg` specifies the audio device, '`/dev/audio`' by default);
- `sun`, for the Solaris audio device (unfinished; `SoundDeviceArg` specifies the audio device, '`/dev/audio`' by default).
- `hpux`, for the HP-UX audio device (unfinished; `SoundDeviceArg` specifies the audio device, '`/dev/audio`' by default).
- `sdl`, for the Simple DirectMedia Layer audio driver.
- `esd`, for EsounD, the Enlightened Sound Daemon; `SoundDeviceArg` specifies the ESD server ('`host:port`') to connect, empty by default.
- `dummy`, fully emulating the SID, but not actually playing samples.
- `dump`, writing all the write accesses to the registers to a file (specified by `SoundDeviceArg`, default value is `vicesnd.sid`);
- `speed`, like `dummy` but also calculating samples (mainly used to evaluate the speed of the sample generator);

- `fs`, writing samples to a file (specified by `SoundDeviceArg`; default is '`vicesnd.raw`');

These drivers will actually be present only if the VICE configuration script detected the corresponding devices at the time of compilation.

`SoundDeviceArg`
String specifying an additional parameter for the audio driver (see `SoundDeviceName`).

### 6.6.2 Sound command-line options

`-sound`
`+sound` Turns sound emulation on (`Sound=1`) and off (`Sound=0`).

`-soundsync N`
Specify `N` as the sound speed adjustment method (`SoundSpeedAdjustment`).

`-soundrate RATE`
Specifies the sound playback sample rate (`SoundSampleRate`).

`-soundbufsize SIZE`
Specifies the size of the audio buffer in milliseconds (`SoundBufferSize`).

`-sounddev NAME`
Specifies the name of the audio device (`SoundDeviceName`).

`-soundarg ARG`
Specifies an additional parameter for the audio device (`SoundDeviceArg`).

## 6.7 Drive settings

These settings are used to control the hardware-level emulation of the drive. When hardware-level emulation is turned on, only drives 8 and 9 are being emulated.

The following settings affect both drives:

- "Enable true drive emulation" enables the (slow) hardware-level emulation of the drives for maximum compatibility. This must be turned on for any of the following settings to have effect.
- "Drive sync factor" specifies the speed of the drive's CPU. This can be used to help loading certain programs that have trouble with the default PAL setting (for example, programs designed for NTSC machines). The ratio is calculated as follows:

```
sync_factor = 65536 * clk_drive / clk_machine
```

where `clk_drive` and `clk_machine` are clock speeds in MHz. The menu lets you choose between the PAL and NTSC values, and also lets you specify whatever value you want. Be careful when changing it, though, because a wrong value can break things and even corrupt disk images.

The following settings, instead, are specific of each drive:

- "Drive model" specifies the model of the drive being emulated. **Warning:** This will reset the drive.

- "Enable parallel cable" enables emulation of a SpeedDOS parallel cable; if you switch this option on and replace the original Commodore ROMs with SpeedDOS-compatible ones, you can speed up loading/saving times.
- "Idle method" specifies which method the drive emulation should use to save CPU cycles in the host CPU. There are three methods:
  - *Skip cycles*: Each time the serial line is accessed by the C64, the drive executes all the cycles since the last time it ran. If the number of elapsed cycles is larger than a certain value, the drive discards part of them.
  - *Trap idle*: The disk drive is still emulated upon serial line accesses as with the previous option, but it is also always emulated at the end of each screen frame. If the drive gets into the DOS idle loop, only pending interrupts are emulated to save time.
  - *No traps*: Like "Trap idle", but without any traps at all. So basically the drive works exactly as with the real thing, and nothing is done to reduce the power needs of the drive emulation.

  The first option ("Skip cycles") is usually best for performance, as the drive is emulated as little as possible; on the other hand, you may notice sudden slowdowns (when the drive executes several cycles at once) and the LED status is never updated (because it would not be possible to do correctly so). Moreover, if the drive tries to get in sync with the computer in some weird way and the computer does not access the serial line for a long time, it is possible that some cycles are discarded and the sync is lost. Notice that this hack will have no effect on performance if a program continuously reads from the IEC port, as the drive will have to be fully emulated in any case (some stupid programs do this, even when they don't actually need to use the drive).

  The second option ("Trap idle") is usually a bit slower, as at least interrupts are always emulated, but ensures the LED state is always updated correctly and always keeps the drive and the computer in sync. On the other hand, if a program installs a non-standard idle loop in the drive, the drive CPU has to be emulated even when not necessary and the global emulation speed is then *much* slower.
- "40-track image support" specifies how 40-track ("extended") disk images should be supported. There are three possible ways:
  - "Never extend" never extends disk images at all (so if a program tries to write tracks beyond the 35th, it is not allowed to do so);
  - "Ask on extend" prompts the user as soon as a program tries to write tracks beyond the 35th, and the user can then choose whether he wants the disk image to be extended or not;
  - "Extend on access" simply extends the disk image as soon the program needs it, without prompting the user.

### 6.7.1 Drive resources

`DriveTrueEmulation`

Boolean controlling whether the "true" drive emulation is turned on.

`Drive8Type`
`Drive9Type`
: Integers specifying the model number for drives 8 and 9. Possible values are `1541`, `1571`, `1581` and `2031`.

`Drive8ParallelCable`
`Drive9ParallelCable`
: Booleans controlling whether the SpeedDOS-compatible cable is emulated or not for drives 8 and 9.

`Drive8ExtendImagePolicy`
`Drive9ExtendImagePolicy`
: Integer specifying the policy for 40-track support for drives 8 and 9. Possible values are `0` (never extend), `1` (ask on extend), `2` (extend on access).

`Drive8IdleMethod`
`Drive9IdleMethod`
: Integers specifying the idling method for the drive CPU. Possible values are `0` (none), `1` (skip cycles), `2` (trap idle). See Section 6.7 [Drive settings], page 32.

`DriveSyncFactor`
: Integer specifying the drive's clock sync factor (see Section 6.7 [Drive settings], page 32). Special values `-1` and `-2` mean PAL and NTSC, respectively.

`DosName1541`
`DosName1571`
`DosName1581`
`DosName2031`
: Strings specifying the names of the ROM images for the drive emulation.

### 6.7.2 Drive command-line options

`-truedrive`
`+truedrive`
: Turns true drive emulation on (`DriveTrueEmulation=1`) and off (`DriveTrueEmulation=0`), respectively.

`-drive8type TYPE`
`-drive9type TYPE`
`-drive10type TYPE`
`-drive11type TYPE`
: Specifies the drive types for drives 8-11, respectively. Possible values for `TYPE` are `1541`, `1571`, `1581` and `2031`.

`-parallel8 <type>`
`-parallel9 <type>`
`-parallel10 <type>`
`-parallel11 <type>`
: Set parallel cable type (0: none, 1: standard, 2: Dolphin DOS)

`-drive8idle NUM`
`-drive9idle NUM`
`-drive10idle NUM`
`-drive11idle NUM`
Specifies `NUM` as the idling method for drives 8-11 (0: no traps, 1: skip cycles, 2: trap idle), respectively (`Drive8IdleMethod`, `Drive9IdleMethod`), `Drive10IdleMethod`), `Drive11IdleMethod`).

`-drive8extend NUM`
`-drive9extend NUM`
`-drive10extend NUM`
`-drive11extend NUM`
Specifies `NUM` as the track 40 extend policy in drives 8 and 9, respectively (`Drive8ExtendImagePolicy`, `Drive9ExtendImagePolicy`).

`-dos1541 <name>`
`-dos1541II <name>`
`-dos1551 <name>`
`-dos1570 <name>`
`-dos1571 <name>`
`-dos1571cr <name>`
`-dos1581 <name>`
`-dos2031 <name>`
`-dos2040 <name>`
`-dos3040 <name>`
`-dos4040 <name>`
`-dos1001 <name>`
Specify the ROM names for the 1541, 1541II, 1551, 1570, 1571, 1571cr, 1581, 2031, 2040, 3040, 4040 and 1001 emulation respectively.

`-drive8ram2000, +drive8ram2000`
`-drive9ram2000, +drive9ram2000`
`-drive10ram2000, +drive10ram2000`
`-drive11ram2000, +drive11ram2000`
Enable/Disable 8KB RAM expansion at $2000-$3FFF

`-drive8ram4000, +drive8ram4000`
`-drive9ram4000, +drive9ram4000`
`-drive10ram4000, +drive10ram4000`
`-drive11ram4000, +drive11ram4000`
Enable/Disable 8KB RAM expansion at $4000-$5FFF

`-drive8ram6000, +drive8ram6000`
`-drive9ram6000, +drive9ram6000`
`-drive10ram6000, +drive10ram6000`
`-drive11ram6000, +drive11ram6000`
Enable/Disable 8KB RAM expansion at $6000-$7FFF

`-drive8ram8000, +drive8ram8000`
`-drive9ram8000, +drive9ram8000`
`-drive10ram8000, +drive10ram8000`
`-drive11ram8000, +drive11ram8000`
Enable/Disable 8KB RAM expansion at $8000-$9FFF

`-drive8rama000, +drive8rama000`
`-drive9rama000, +drive9rama000`
`-drive10rama000, +drive10rama000`
`-drive11rama000, +drive11rama000`
Enable/Disable 8KB RAM expansion at $A000-$BFFF

`-drive8profdos, +drive8profdos`
`-drive9profdos, +drive9profdos`
`-drive10profdos, +drive10profdos`
`-drive11profdos, +drive11profdos`
Enable/Disable Professional DOS

`-profdos1571 <name>`
`Specify name of Professional DOS 1571 ROM image`

## 6.8 Peripheral settings

VICE is able to support some special peripherals:

- *file system devices*, pseudo-drives accessing the Unix file system;
- printers.

These features depend on some *kernal traps* that replace the existing routines in the original Commodore operating system with custom-made C routines.

### 6.8.1 Settings for file system devices

These settings deal with the drive-like peripherals connected to the bus of the emulated machine. The first setting relates to the parallel IEEE488 interface. With this interface a special engine is used to listen to the bus lines to translates them to the filesystem code. Thus the PET will always detect a drive for example, but it can also use drives 10 and 11 even together with true disk drive emulation.

- "Enable virtual devices", enables the peripheral access via the fast disk emulation (either kernal traps or IEEE488 interface). Both, filesystem and disk image access via fast drive emulation, are affected.

Four peripherals, numbered from 8 to 11, are accessible; each of them provides the following settings:

- "File system access", if enabled, allows the device to emulate a drive accessing a file system directory; note that when a disk image is attached to the same drive, the directory is no longer visible and the attached disk is used instead.
- "File system directory" specifies the directory to be accessed by the drive.
- "Convert P00 file names", if enabled, allows access to P00 files using their built-in name instead of the Unix one.

- "Create P00 files on save", if enabled, creates P00 files (instead of raw CBM files) whenever a program creates a file.

Note that, by default, all drives except 11 create P00 files on save, while drive 11 creates raw CBM files. Those files come without any header, but also with the filename restrictions given by the operating system VICE runs on.

#### 6.8.1.1 Resources for file system devices

`FSDevice8ConvertP00`
`FSDevice9ConvertP00`
`FSDevice10ConvertP00`
`FSDevice11ConvertP00`
: Booleans specifying whether on-read support for P00 files is enabled on drives 8, 9, 10 and 11 respectively (on by default).

`FSDevice8SaveP00`
`FSDevice9SaveP00`
`FSDevice10SaveP00`
`FSDevice11SaveP00`
: Booleans specifying whether the drives should create P00 files instead of plain CBM ones (on by default for drives 8-10, off for 11).

`FSDevice8HideCBMFiles`
`FSDevice9HideCBMFiles`
`FSDevice10HideCBMFiles`
`FSDevice11HideCBMFiles`
: Booleans specifying whether non-P00 files should be invisible to programs running in the emulator (do not hide by default).

`FSDevice8Dir`
`FSDevice9Dir`
`FSDevice10Dir`
`FSDevice11Dir`
: Strings specifying the directories to which drives 8, 9, 10 and 11 have access.

#### 6.8.1.2 Command-line options for file system devices

`-device8 <type>`
`-device9 <type>`
`-device10 <type>`
`-device11 <type>`
: Set device type for device 8-11 (0: NONE, 1: FILESYSTEM, 2: OPENCBM, 3: BLOCK DEVICE)

`-fs8 PATH`
`-fs9 PATH`
`-fs10 PATH`
`-fs11 PATH`
: Specify the paths for the file system access on drives 8, 9, 10 and 11, respectively (`FSDevice8Dir`, `FSDevice9Dir`, `FSDevice10Dir` and `FSDevice11Dir`).

### 6.8.2 Printer settings

The VICE emulators can emulate printers connected to either the IEC buffer or the user port. Emulation can be achieved by redirecting the printer output to a file or by piping it through an external process. This is defined by so-called *printer device file names*; a printer device file name can be either a simple path, or a command name preceeded by a pipe symbol '`|`'.

For example, printer device '`filename`' will cause the output to be appended to the file '`filename`', while printer device '`|lpr`' will cause the `lpr` command to be executed and be fed the printer output. The printer output will not be converted but saved as printed by the emulated machine.

Up to three printer devices may be specified through the following resources:

- device 1, whose default value is `print.dump`;
- device 2, whose default value is `|lpr`.
- device 3, whose default value is `|petlp -F PS|lpr`;

So, basically, by default printer device 1 will dump printer output to '`print.dump`'; printer device 2 will print it via `lpr` directly to the printer and device 3 will print it via `petlp` (a not-yet-complete utility that will produce Postscript output from the Commodore printer code) and then to the printer via `lpr`.

#### 6.8.2.1 Printer resources

`PrDevice1`
`PrDevice2`
`PrDevice3`
: Strings specifying the printer devices (see Section 6.8.2 [Printer settings], page 38).

`Printer4` Boolean specifying if the IEC printer (device 4) is being emulated.

`Printer4Dev`
: Integer (ranging from 0 to 2, for device 1-3) specifying what printer device (see Section 6.8.2 [Printer settings], page 38) the IEC printer is using.

`PrUser` Boolean specifying if the user-port printer is being emulated.

`PrUserDev`
: Integer (ranging from 0 to 2, for device 1-3) specifying what printer device the user-port printer is using.

#### 6.8.2.2 Printer command-line options

`-prtxtdev1 <name>`
`-prtxtdev2 <name>`
`-prtxtdev3 <name>`
: Specify name of printer text device or dump file

`-pr4txtdev <0-2>`
`-pr5txtdev <0-2>`
: Specify printer text output device for IEC printer #4-5

`-pr4output <name>`
`-pr5output <name>`
: Specify name of output device for device #4-5 Specify name of output device for device #5-5

`-pr4drv <name>`
`-pr5drv <name>`
: Specify name of printer driver for device #4-5 Specify name of printer driver for device #5-5

`-pruser`
`+pruser` Enable/disable emulation of the userport printer emulation (`PrUser=1`, `PrUser=0`).

`-prusertxtdev <0-2>`
: Specify printer text output device for userport printer

`-pruseroutput <name>`
: Specify name of output device for the userport printer

`-pruserdrv <name>`
: Specify name of printer driver for the userport printer

### 6.8.3 Disabling kernal traps

If you have compatibility problems, you can completely disable Kernal traps with the "Disable kernal traps" option. This will of course disable all the features that depend on it, such as the fast 1541 emulation (so you will have to turn true 1541 emulation on if you want to be able to read or write disk images) and tape support.

#### 6.8.3.1 Resources to control Kernal traps

`VirtualDevices`
: Boolean specifying whether all the mechanisms for virtual device emulation should be enabled. Serial IEC devices use kernal traps, parallel IEEE488 devices use an own IEEE488 engine. Both are switched on and off with this resource.

#### 6.8.3.2 Command-line options to control Kernal traps

`-virtualdev`
`+virtualdev`
: Enable (`VirtualDevices=1`) or disable (`VirtualDevices=0`) virtual devices.

## 6.9 RS232 settings

The VICE emulators can emulate the RS232 device most of the machines have. The C64, C128 and VIC20 emulators emulate the userport RS232 interface at 300 and 1200 baud. The C64 and C128 can also use the 9600 baud interface by Daniel Dallmann, using the shift registers of the two CIA 6526 chips. The PET can have a 6551 ACIA RS232 interface when running as a SuperPET, and the CBM-II has such an ACIA by default. The C64 and C128 emulators can emulate an ACIA 6551 (also known as Datapump for example) as extension at `$de**`.

Emulation can be achieved by either:

- connecting a real UNIX serial device;
- dumping to a file;
- piping through a process.

It is possible to define up to four UNIX serial devices, and then decide which interface should be connected to which device. This is done by so-called *rs232 device file names*; an rs232 device file name can be either a simple path, or a command name preceeded by a pipe symbol '`|`'. If the path specifies a special device (e.g. '`/dev/ttyS0`') it is recognized by VICE and the emulator can set the baudrate.

For example, rs232 device '`filename`' will cause the output to be written (not appended) to the file '`filename`', while printer device '`|lpr`' will cause the `lpr` command to be executed and be fed the rs232 output. The rs232 output will not be converted but saved as sent by the emulated machine. The same holds true for the rs232 input. If the command writes data to the standard output it will be caught by VICE and sent back to the emulator. Also the data sent by the pseudo device will be sent back to VICE.

For example you can setup a null-modem cable between two serial ports of your PC, setup one port for login and use the other in VICE. Then you can login from your emulator via the RS232 emulation and the null-modem cable to your machine again.

You can not simply run a shell from VICE, as the shell will notice that it does not run on its own pseudo terminal and will thus buffer its output. You need to write some program that opens an own pseudo terminal and runs the shell from there (not yet finished).

Up to four RS232 devices may be specified through the following resources:

- device 1, whose default value is `/dev/ttyS0`;
- device 2, whose default value is `/dev/ttyS1`;
- device 3, whose default value is `rs232.dump`;
- device 4, whose default value is `|lpr`.

For the first two devices you can change the baudrate the tty device is set to by specifying it on the commandline or in the menu. This baudrate is 9600 by default for the latter two, but can be changed only by resources (The baudrate is independent from the baudrate the emulator actually expects).

### 6.9.1 RS232 resources

`RsDevice1`
`RsDevice2`
`RsDevice3`
`RsDevice4`

Strings specifying the RS232 devices (see Section 6.9 [RS232 settings], page 39).

`RsDevice1Baud`
`RsDevice2Baud`
`RsDevice3Baud`
`RsDevice4Baud`

Integer specifying the RS232 baudrate devices if the device file points to a special device (like '`/dev/ttyS0`'; see Section 6.9 [RS232 settings], page 39).

`AciaDE` Boolean specifying whether C64 or C128 should emulate ACIA 6551 in I/O 1, at `$de**`.

`Acia1Dev` Integer (ranging from 0 to 3, for device 1-4) specifying what RS232 device (see Section 6.9 [RS232 settings], page 39) the ACIA is using (all except VIC20).

`Acia1Irq` Integer specifying which interrupt to use. 0 = none, 1 = IRQ, 2 = NMI (C64 and C128 only)

`RsUser` Integer specifying if the user-port RS232 interface is being emulated and at which baudrate it should have for the emulator. 0 = off; > 0 specifies the baudrate (C64, C128 and VIC20).

`RsUserDev`
Integer (ranging from 0 to 3, for device 1-4) specifying what RS232 device the user-port interface is using (C64, C128 and VIC20).

### 6.9.2 RS232 command-line options

`-rsdev1 NAME`
`-rsdev2 NAME`
`-rsdev3 NAME`
`-rsdev4 NAME`
Specify `NAME` as RS232 devices 1, 2, 3 and 4, respectively (`RsDevice1`, `RsDevice2` `RsDevice3` and `RsDevice4`).

`-rsdev1 BAUDRATE`
`-rsdev2 BAUDRATE`
`-rsdev3 BAUDRATE`
`-rsdev4 BAUDRATE`
Specify `BAUDRATE` as baudrate for the RS232 devices if the device name specifies a special device (like '`/dev/ttyS0`' for example, see Section 6.9 [RS232 settings], page 39; `RsDevice1Baud`, `RsDevice2Baud` `RsDevice3Baud` and `RsDevice4Baud`).

`-myaciadev <0-3>`
Specify RS232 device the ACIA should work on

`-rsuser BAUDRATE`
Enable (BAUDRATE not 0) or disable (BAUDRATE = 0) emulation of the userport RS232 emulation (`RsUser`; C64, C128 and VIC20)

`-rsuserdev DEV`
Specify device for the userport RS232 emulation (`RsUserDev`; C64, C128 and VIC20).

### 6.9.3 RS232 usage example

Here we give you a simple example how to set up an emulated C64 using the modem connected to your PC. The following list shows each step.

`Attach your modem to your PC at a serial port.`
Normally you should set it up to use the modem as "/dev/modem".

`start VICE`
`Setup VICE to use your modem as "serial device 1"`
: Go to the RS232 settings menu and change "Serial 1 device" to "/dev/modem" (or the device where you attached your modem to) Then go to the RS232 settings menu and change "Serial 1 baudrate" to the baudrate your modem should run at. Watch out, e.g. on Linux there is an additional multiplier to multiply with the baudrate (so e.g. 19200 gives 115200 or so baud) See the "setserial" manpage on Linux for example. However, most modems should be able to autodetect the speed to the computer as well.

`Select the RS232 emulation your programs use`
: If you want to use the Userport emulation, go to the RS232 settings and change "Userport RS232 Device" to "Serial 1". If you want ACIA emulation (swiftlink or what's it called?) then change "ACIA $DE** device" to "Serial 1".

`Enable the emulation`
: Go to the RS232 settings and select either "ACIA $DE** emulation" or Userport 300/1200 baud or CIA 9600 baud emulation.

`Load your program and start it.`
: If it is able to detect an RS232 cartridge like swiftlink or so, try to detect the ACIA emulation if enabled. Otherwise just set the baudrate to either 300, 1200 or 9600 according to what you enabled in the VICE menu for the userport.

## 6.10 Monitor settings

This section lists command-line options specific to the built-in monitor.

### 6.10.1 Monitor command-line options

`-moncommands FILENAME`
: Execute the commands from the file `FILENAME` in the monitor after starting up. This command line switch is mainly thought to load labels and to set breakpoints. Not all other commands are useful to be executed in this way, some may even lead to strange effects.

`-initbreak <address>`
: Set an initial breakpoint for the monitor. Addresses with prefix "0x" are hexadecimal.

`-remotemonitor`
`+remotemonitor`
: Enable/Disable remote monitor

`-remotemonitoraddress <name>`
: The local address the remote monitor should bind to

## 6.11 Miscellaneous settings

This section lists generic resources that do not fit in the other categories.

### 6.11.1 Miscellaneous resources

`Directory`

String specifying the search path for system files. It is defined as a sequence of directory names, separated by colons (‘`:`’), just like the `PATH` variable in the shell. The special string ‘`$$`’ stands for the default search path, which is initialized at startup to the following value:

```
LIBDIR/EMUID:$HOME/.vice/EMUID:BOOTPATH/EMUID:LIBDIR/DRIVES:$HOME/.vice/DRIV
```

where:

- `LIBDIR` is the VICE installation directory (usually ‘`/usr/local/lib/vice`’, ‘`/usr/lib/vice`’ or ‘`/opt/vice/lib`’);
- `EMUID` is the emulation identification string (`C64`, `C128`, `VIC20` or `PET`);
- `BOOTPATH` is the directory where the binary lies (usually ‘`/usr/local/bin`’, ‘`/usr/bin`’ or `/opt/vice/bin`).
- `DRIVES` is the directory called `"DRIVES"`, where the disk drive ROMs are. (The disk drive ROMs are used by all emulators, so there is an extra directory for them.)

Notice that the middle entry points to a default location in the user’s home directory. Here private ROM versions (e.g. speeddos or JiffyDos) can be stored for example.

See Chapter 4 [System files], page 16. for a description of the method used to load the emulator’s system files.

`HTMLBrowserCommand`

String specifying the command to run the help browser. The help browser can be any HTML browser, and every ‘`%s`’ in the string is replaced with the name of the toplevel file of the VICE documentation. For example, the default value ‘`netscape %s`’ runs Netscape Navigator.

`SaveResourcesOnExit`

Boolean specifying whether the emulator should save changed settings before exiting. If this is enabled, the user will be always prompted first, in case the settings have changed.

`DoCoreDump`

Boolean specifying whether the emulator should dump core when it gets a signal.

`JoyDevice1`
`JoyDevice2`

Integer specifying which joystick device the emulator should use for joystick emulation for ports 1 and 2, respectively (0=None, 1=Numpad, 2=Custom keys, 3=Analog joystick 1, 4=Analog joystick 2, 5=Digital joystick 1, 6=Digital joystick 2 on Unix) The available joysticks might differ depending on operating system and joystick support in the OS (Linux joystick module must be available for example).

### 6.11.2 Miscellaneous command-line options

`-directory SEARCHPATH`
: Specify the system file search path (`Directory`).

`-htmlbrowser COMMAND`
: Specify the command to run the HTML browser for the on-line help (`HTMLBrowserCommand`).

`-saveres`
`+saveres`
: Enable/disable automatic saving of settings on exit (`SaveResourcesOnExit=1`, `SaveResourcesOnExit=0`).

`-core`
`+core`
: Enable/disable generation of core dumps (`DoCoreDump=1`, `DoCoreDump=0`).

`-joydev1`
`-joydev2`
: Set the device for joystick emulation of port 1 and 2, respectively (`JoyDevice1`, `JoyDevice2`).

# 7 Machine-specific features

## 7.1 C64/128-specific commands and settings

This section lists the settings and commands that are C64/128 specific and thus are not present in the other emulators.

### 7.1.1 Using cartridges

The cartridge system is organized in "Slots" to allow more than one cartridge connected at a time, like it can be done using an expansion port expander on a real C64 (see below).

Generally a cartridge can be enabled by attaching its respective cartridge image, or using the respective menu option for cartridges that do not require an image.

x64, x64sc and x128 allow you to attach the following kinds of images:

- '`.crt`' images, as used by the CCS64 emulator by Per Hkan Sundell
- raw '`.bin`' images, with or without load address

*Cartridge images* are like disk images, but contain the contents of cartridge ROM and/or RAM images instead of disk images.

To attach cartridges, use the "Attach a cartridge image" submenu. When using '`.crt`' images, this will work for every cartridge which is supported. For raw '`.bin`' images you might have to use command line options.

When you have successfully attached a cartridge image, you should then reset the machine to make sure the cartridge initializes itself. (Or enable the "reset on cartridge change" option).

Of course, it is also possible to detach a currently attached cartridge image ("Detach cartridge image").

If you are using a freezer cart like an Action Replay cartridge, you can emulate the cartridge's freeze button with the "Cartridge freeze" command.

The imaginary expansion port expander is organized in 4 slots, the cartridges are associated with them like this:

**"Slot 0"**

All carts that have a passthrough connector go here. Once a "Slot 0" cartridge is enabled all further cartridges are connected to its respective passthrough port.

Only one cartridge of this type can be active at a time.

"Slot 0" carts have individual "enable" switches, enabling means enabling permanently.

The following cartridges are emulated in this slot:

- IEEE-488 Interface (`http://www.funet.fi/pub/cbm/schematics/cartridges/c64/ieee-488/eprom.bin`)
- Magic Voice
- MMC64

**"Slot 1"**

Mostly RAM based cartridges which for one reason or the other might make sense to be enabled together with one of the "Main Slot" cartridges go here.

Only one cartridge of this type can be active at a time.

"Slot 1" carts have individual "enable" switches, enabling means enabling permanently

The following cartridges are emulated in this slot:

- Double Quick Brown Box (DQBB)
- Expert Cartridge
- ISEPIC
- RamCart

**"Main Slot"**

All other cartridges which are not pure i/o extensions go here.

Only one cartridge of this type can be active at a time.

Cartridges in the "Main Slot" must be explicitly set as default to enable them permanently.

The following cartridges are emulated in this slot:

- generic 4K, 8K and 16K game- and ultimax cartridges
- Action Replay V5
- Action Replay MK2
- Action Replay MK3
- Action Replay MK4
- Atomic Power
- C64 Games System
- Capture
- Comal 80
- Dela EP64
- Dela EP7x8

- Dela EP256
- Diashow-Maker
- Dinamic
- EasyFlash
- Epyx FastLoad
- EXOS
- The Final Cartridge
- The Final Cartridge III
- Final Cartridge Plus
- Freeze Frame
- Freeze Machine
- Fun Play
- Game Killer
- IDE64 (`http://www.volny.cz/dundera/`)
- KCS Power Cartridge
- MACH 5
- Magic Desk
- Magic Formel
- Mikro Assembler
- MMC Replay
- Ocean
- Prophet64
- REX 256k EPROM Cart
- REX Utility
- Retro Replay
- ROSS
- Simons' BASIC
- Snapshot 64
- Stardos
- Structured BASIC
- Super Explode V5.0
- Super Games
- Super Snapshot V4
- Super Snapshot V5
- Warp Speed
- Westermann Learning
- Zaxxon

**"I/O Slot"**

All carts that are pure I/O extensions go here.

Any number of "I/O Slot" Carts may be active at a time.

"I/O Slot" carts have individual "enable" switches, enabling means enabling permanently.

The following cartridges are emulated in this slot:

- ACIA (Swiftlink, Turbo232)
- DigiMAX
- Ethernet (The Final Ethernet, RR-Net)
- GEO-RAM
- MIDI (Passport, Datel, Maplin, Namesoft, Sequential)
- RAM Expansion Module (REU)
- SFX Sound Expander
- SFX Sound Sampler

#### 7.1.1.1 The Final Cartridge 3

The Final Cartridge 3 detects wether a mouse is connected when it starts and disables mouse support if it doesnt detect one. So to make mouse emulation work you must either enable it on the command line, or reset the cartridge after enabling it from the user interface.

### 7.1.2 C64 cartridge settings

#### 7.1.2.1 C64 cartridge resources

`CartridgeReset`
`CartridgeType`
`CartridgeFile`
`DQBB` Boolean specifying whether the Double Quick Brown Box should be emulated or not.

`DQBBfilename`
`DQBBImageWrite`
`ExpertCartridgeEnabled`
Boolean specifying whether the Expert Cartridge should be emulated or not.

`Expertfilename`
`ExpertImageWrite`
`ExpertCartridgeMode`
`IDE64Image1`
`IDE64Image2`
`IDE64Image3`
`IDE64Image4`
`IDE64Config`
`IDE64Cylinders`
`IDE64Heads`
`IDE64Sectors`
`IDE64AutodetectSize`
`IDE64version4`
`IDE64RTCOffset`
`IEEE488` Boolean specifying whether the IEEE488 interface should be emulated or not.

`IEEE488Image`
`IsepicCartridgeEnabled`
Boolean specifying whether ISEPIC should be emulated or not.

`Isepicfilename`
`IsepicSwitch`
`IsepicImageWrite`
`MagicVoiceCartridgeEnabled`
Boolean specifying whether the Magic Voice should be emulated or not.

`MagicVoiceImage`
`MMC64` Boolean specifying whether the MMC64 should be emulated or not.

`MMC64BIOSfilename`
`MMC64_bios_write`
`MMC64_flashjumper`
`MMC64_revision`
`MMC64imagefilename`
`MMC64_RO`

`MMC64_sd_type`
`MMCRCardImage`
`MMCREEPROMImage`
`MMCRRescueMode`
`MMCRImageWrite`
`MMCRCardRW`
`MMCRSDType`
`MMCREEPROMRW`
`RAMCART` Boolean specifying whether the RAMCart should be emulated or not.

```
RAMCARTfilename
RAMCARTImageWrite
RAMCART_RO
RAMCARTsize
RRFlashJumper
RRBankJumper
RRBiosWrite
```

#### 7.1.2.2 C64 cartridge command-line options

`+cart` Disable all cartridges (which would eventually be enabled in the config file).

`-cartreset`
`+cartreset`
Reset/Do not reset machine if a cartridge is attached or detached

`-cart8 <name>`
Attach generic 8KB cartridge image

`-cart16 <name>`
Attach generic 16KB cartridge image

`-cartultimax <name>`
Attach generic 16kB Ultimax cartridge image

`-cartcrt <name>`
Attach CRT cartridge image

`-cartap <name>`
Attach raw 32KB Atomic Power cartridge image

`-cartar2 <name>`
Attach raw 16kB Action Replay MK2 cartridge image

`-cartar3 <name>`
Attach raw 16KB Action Replay MK3 cartridge image

`-cartar4 <name>`
Attach raw 32KB Action Replay MK4 cartridge image

`-cartar5 <name>`
Attach raw 32KB Action Replay cartridge image

`-cartcap <name>`
Attach raw 8kB Capture cartridge image

`-cartcomal <name>`
Attach raw 64kB Comal 80 cartridge image

`-cartdep256 <name>`
Attach raw Dela EP256 cartridge image

`-cartdep64 <name>`
Attach raw Dela EP64 cartridge image

`-cartdep7x8 <name>`
Attach raw Dela EP7x8 cartridge image

`-cartdin <name>`
Attach raw 128kB Dinamic cartridge image

`-cartdsm <name>`
Attach raw 8kB Diashow-Maker cartridge image

`-cartdqbb <name>`
Attach raw 16kB Double Quick Brown Box cartridge image

`-dqbb`
`+dqbb` Enable/Disable Double Quick Brown Box

`-dqbbimage <name>`
Specify Double Quick Brown Box filename

`-dqbbimagerw`
`+dqbbimagerw`
Allow/Disallow writing to DQBB image

`-carteasy <name>`
Attach raw EasyFlash cartridge image

`-easyflashjumper`
`+easyflashjumper`
Enable/Disable EasyFlash jumper

`-easyflashcrtwrite`
`+easyflashcrtwrite`
Allow/Disallow writing to EasyFlash .crt image

`-cartepyx <name>`
Attach raw 8KB Epyx FastLoad cartridge image

`-cartexos <name>`
Attach raw 8kB EXOS cartridge image

`-cartexpert <name>`
Attach raw 8kB Expert Cartridge image

`-expert`
`+expert` Enable/Disable the Expert Cartridge

`-expertimagename <name>`
Set Expert Cartridge image name

`-expertimagerw`
`+expertimagerw`
Allow/Disallow writing to Expert Cartridge image

`-cartfc1 <name>`
Attach raw 16kB Final Cartridge image

`-cartfc3 <name>`
Attach raw 64kB Final Cartridge III image

`-cartfcplus <name>`
Attach raw 32kB Final Cartridge Plus image

`-cartff <name>`
Attach raw 8kB Freeze Frame image

`-cartfm <name>`
Attach raw 32kB Freeze Machine image

`-cartfp <name>`
Attach raw 128kB Fun Play/Power Play cartridge image

`-cartgk <name>`
Attach raw 8KB Game Killer cartridge image

`-cartgs <name>`
Attach raw 512kB Game System cartridge image

`-cartide64 <name>`
Attach raw 64KB IDE64 cartridge image

`-IDE64image1 <name>`
Specify name of IDE64 image file

`-IDE64image2 <name>`
Specify name of IDE64 image file

`-IDE64image3 <name>`
Specify name of IDE64 image file

`-IDE64image4 <name>`
Specify name of IDE64 image file

`-IDE64cyl <value>`
Set number of cylinders for the IDE64 emulation

`-IDE64hds <value>`
Set number of heads for the IDE64 emulation

`-IDE64sec <value>`
Set number of sectors for the IDE64 emulation

`-IDE64autosize`
`+IDE64autosize`
Autodetect/do not autodetect geometry of formatted images

`-IDE64version4`
`+IDE64version4`
Emulate version 4 hardware/Emulate pre version 4 hardware

`-cartieee <name>`
Attach CBM IEEE-488 cartridge image

`-ieee488`
`+ieee488` Enable (`IEEE488=1`) or disable (`IEEE488=0`) emulation of the IEEE488 interface.

`-ieee488image <name>`
Set IEEE488 interface image name

`-isepic`
`+isepic` Enable/Disable the ISEPIC cart

`-cartisepic <name>`
Attach raw 2kB ISEPIC cartridge image

`-isepicimagename <name>`
Set ISEPIC image name

`-isepicimagerw`
`+isepicimagerw`
Allow/Disallow writing to ISEPIC image

`-cartkcs <name>`
Attach raw 16kB KCS Power cartridge image

`-cartmach5 <name>`
Attach raw 8kB MACH 5 cartridge image

`-cartmd <name>`
Attach raw 32/64/128kB Magic Desk cartridge image

`-cartmf <name>`
Attach raw Magic Formel cartridge image

`-cartmikro <name>`
Attach raw 8kB Mikro Assembler cartridge image

`-mmc64`
`+mmc64` Enable/Disable the MMC64 expansion

`-cartmmc64 <name>`
Attach raw 8kB MMC64 cartridge image

`-mmc64bios <name>`
Specify name of MMC64 BIOS image

`-mmc64image <name>`
Specify name of MMC64 image

`-mmc64readonly`
Set the MMC64 card to read-only

`-mmc64readwrite`
Set the MMC64 card to read/write

`-mmc64bioswrite`
Save the MMC64 bios when changed

`-cartmmcr <name>`
Attach raw 512kB MMC Replay cartridge image

`-mmcrrescue`
`+mmcrrescue`
Enable/Disable MMC Replay rescue mode

`-mmcrimagerw`
`+mmcrimagerw`
Allow/Disallow writing to MMC Replay image

`-mmcrcardimage <filename>`
Specify MMC Replay card image filename

`-mmcrcardrw`
`+mmcrcardrw`
Allow/Disallow writes to MMC Replay card image

`-mmcreepromimage`
Specify MMC Replay EEPROM image filename

`-mmcreepromrw`
`+mmcreepromrw`
Allow/Disallow writes to MMC Replay EEPROM image

`-cartmv <name>`
Attach raw 16kB Magic Voice cartridge image

`-cartocean <name>`
Attach raw Ocean cartridge image

`-cartp64 <name>`
Attach raw 256KB Prophet 64 cartridge image

`-cartramcart <name>`
Attach raw RamCart cartridge image

`-ramcart`
`+ramcart` Enable/Disable the RAMCART expansion

`-ramcartsize <size in KB>`
Size of the RAMCART expansion

`-ramcartimage <name>`
Specify name of RAMCART image

`-ramcartimagerw`
`+ramcartimagerw`
Allow/Disallow writing to RAMCart image

`-cartrep256 <name>`
Attach raw REX EP256 cartridge image

`-cartross <name>`
Attach raw 16/32kB ROSS cartridge image

`-cartrr <name>`
Attach raw 64KB Retro Replay cartridge image

`-rrbioswrite`
`+rrbioswrite`
Enable/Disable saving of the RR ROM at exit

`-rrbankjumper`
`+rrbankjumper`
Set/Unset RR Bank Jumper

`-rrflashjumper`
`+rrflashjumper`
Set/Unset RR Flash Jumper

`-cartru <name>`
Attach raw 8kB REX Utility cartridge image

`-carts64 <name>`
Attach raw 4kB Snapshot 64 cartridge image

`-cartsb <name>`
Attach raw Structured Basic cartridge image

`-cartse5 <name>`
Attach raw 16kB Super Explode V5 cartridge image

`-cartsg <name>`
Attach raw 64kB Super Games cartridge image

`-cartsimon <name>`
Attach raw 16kB Simons Basic cartridge image

`-cartss4 <name>`
Attach raw 32KB Super Snapshot V4 cartridge image

`-cartss5 <name>`
Attach raw 64KB Super Snapshot V5 cartridge image

`-cartstar <name>`
Attach raw 16KB Stardos cartridge image

`-cartwl <name>`
Attach raw 16KB Westermann Learning cartridge image

`-cartws <name>`
Attach raw 8kB Warp Speed cartridge image

`-cartzaxxon <name>`
Attach raw 16kB Zaxxon cartridge image

### 7.1.3 VIC-II settings

These settings control the emulation of the VIC-II (MOS6569) video chip used in both the C64 and the C128.

- "Sprite-sprite collisions" and "Sprite-background collisions", if enabled, cause the hardware detection of sprite-to-sprite and sprite-to-background collisions of the VIC-II to be emulated. This feature is used by many games, and disabling either of the two detection systems can sometimes make you invincible (although there is also a chance that also enemies become invincible then).
- "Color set" can be used to dynamically change the palette file being used by choosing one of the available predefined color sets:

- ‘`default.vpl`’ (“default”), the default VICE palette;
- ‘`c64s.vpl`’ (“C64S”), palette taken from the shareware C64S emulator by Miha Peternel.
- ‘`ccs64.vpl`’ (“CCS64”), palette taken from the shareware CCS64 emulator by Per Hkan Sundell.
- ‘`frodo.vpl`’ (“Frodo”), palette taken from the free Frodo emulator by Christian Bauer (`http://www.uni-mainz.de/~bauec002/FRMain.html`).
- ‘`pc64.vpl`’ (“PC64”), palette taken from the free PC64 emulator by Wolfgang Lorenz.
- ‘`godot.vpl`’ (“GoDot”), palette as suggested by the authors of the C64 graphics package GoDot (`http://users.aol.com/howtogodot/welcome.htm`).

#### 7.1.3.1 VIC-II resources

`VICIICheckSsColl`
: Boolean specifying whether the sprite-sprite hardware collision detection must be emulated.

`VICIICheckSbColl`
: Boolean specifying whether the sprite-background hardware collision detection must be emulated.

`VICIIVideoCache`
: Boolean specifying whether the video cache is turned on.

`VICIIDoubleSize`
: Boolean specifying whether double-size mode is turned on.

`VICIIDoubleScan`
: Boolean specifying whether double-scan mode is turned on.

`VICIIPaletteFile`
: String specifying the name of the palette file being used. The ‘`.vpl`’ extension is optional.

#### 7.1.3.2 VIC-II command-line options

`-VICIIcheckss`
`+VICIIcheckss`
: Enable (`VICIICheckSsColl=1`) and disable (`VICIICheckSsColl=0`) emulation of hardware sprite-sprite collision detection, respectively.

`-VICIIchecksb`
`+VICIIchecksb`
: Enable (`VICIICheckSbColl=1`) and disable (`VICIICheckSbColl=0`) emulation of hardware sprite-background collision detection, respectively.

`-VICIIvcache`
`+VICIIvcache`
: Enable/disable the video cache (`VICIIVideoCache=1`, `VICIIVideoCache=0`).

`-VICIIdsize`
`+VICIIdsize`
Enable/disable the double size mode (`VICIIDoubleSize=1`, `VICIIDoubleSize=0`).

`-VICIIdscan`
`+VICIIdscan`
Enable/disable the double scan mode (`VICIIDoubleScan=1`, `VICIIDoubleScan=0`).

`-VICIIhwscale`
`+VICIIhwscale`
Enable/Disable hardware scaling

`-VICIIscale2x`
`+VICIIscale2x`
Enable/Disable Scale2x

`-VICIIintpal`
Use an internal calculated palette

`-VICIIextpal`
Use an external palette (file)

`-VICIIpalette NAME`
Specify `NAME` as the palette file (`VICIIPaletteFile`).

`-VICIIfulldevice <device>`
Select fullscreen device

`-VICIIXRANDRfullmode <mode>`
Select fullscreen mode

`-VICIIVidmodefullmode <mode>`
Select fullscreen mode

`-VICIIborders <mode>`
Set VIC-II border display mode (0: normal, 1: full, 2: debug)

`-VICIImodel <model>`
Set VIC-II model (6569/6569r1/8565/6567/8562/6567r56a). This setting is only available in `x64sc`.

### 7.1.4 SID settings

These settings control the emulation of the SID (MOS6581 or MOS8580) audio chip.

- "Second SID" maps a second SID chip into the address space for stereo sound. This emulates e.g. the "SID Symphony Stereo Cartridge" from Dr. Evil Laboratories. The second SID can be used with software such as "Stereo SID Player" by Mark Dickenson or "The Enhanced Sidplayer" by Craig Chamberlain.
- "Second SID base address" sets the start address for the second SID chip. Software normally uses $DE00 or $DF00, since $DE00-$DEFF and $DF00-$DFFF can be mapped through the cartridge port of the C64. The default start address is $DE00.

- "Emulate filters" causes the built-in programmable filters of the SID chip to be emulated. A lot of C64 music requires them to be emulated properly, but their emulation requires some additional processor power.
- "ChipModel" specifies the model of the SID chip being emulated: there are two slightly different generations of SID chips: MOS6581 ones and MOS8580 ones.
- "Use reSID emulation" specifies whether the more accurate (and resource hungry) reSID emulation is turned on or off.
- "reSID sampling method" selects the method for conversion of the SID output signal to a sampling rate appropriate for playback by standard digital sound equipment. Possible settings are:
  - "Fast" simply clocks the SID chip at the output sampling frequency, picking the nearest sample. This yields acceptable sound quality, but sampling noise is noticeable in some cases, especially with SID combined waveforms. The sound emulation is still cycle exact.
  - "Interpolating" clocks the SID chip each cycle, and calculates each sample with linear interpolation. The sampling noise is now strongly attenuated by the SID external filter (as long as "Emulate filters" is selected), and the linear interpolation further improves the sound quality.
  - "Resampling" clocks the SID chip each cycle, and uses the theoretically correct method for sample generation. This delivers CD quality sound, but is extremely CPU intensive, and is thus most useful for non-interactive sound generation. Unless you have a very fast machine, that is.
- "reSID resampling passband" specifies the percentage of the total bandwidth allocated to the resampling filter passband. The work rate of the resampling filter is inversely proportional to the remaining transition band percentage. This implies that e.g. with the transition band starting at ~ 20kHz, it is faster to generate 48kHz than 44.1kHz samples. For CD quality sound generation at 44.1kHz the passband percentage should be set to 90 (i.e. the transition band starting at almost 20kHz).

#### 7.1.4.1 SID resources

`SidStereo`
: Boolean selecting emulation of a second SID.

`SidStereoAddressStart`
: Integer specifying the start address for the second SID.

`SidFilters`
: Boolean specifying whether the built-in SID filters must be emulated.

`SidModel`
: Integer specifying what model of the SID must be emulated (`0`: MOS6581, `1`: MOS8580).

`SidUseResid`
: Boolean specifying whether the accurate reSID emulation is being used.

`SidResidSampling`
: Integer specifying the sampling method (`0`: Fast, `1`: Interpolation, `2`: Resampling)

`SidResidPassband`
: Integer specifying the resampling filter passband in percentage of the total bandwidth (`0 - 90`).

#### 7.1.4.2 SID command-line options

`-sidstereo`
: Emulates a second SID chip for stereo sound (`SidStereo`).

`-sidstereoaddress ADDRESS`
: Specifies the start address for the second SID chip (`SidStereoAddressStart`).

`-sidenginemodel <engine and model>`
: Specify `SID engine` and `MODEL` for the emulated SID chip ( 0: FastSID 6581, 1: FastSID 8580, 256: ReSID 6581, 257: ReSID 8580, 258: ReSID 8580 + digiboost, 1024: ParSID in par port 1, 1280: ParSID in par port 2, 1536: ParSID in par port 3, 1800: ReSID-FP 6581R3 4885, 1801: ReSID-FP 6581R3 0486S, 1802: ReSID-FP 6581R3 3984, 1803: ReSID-FP 6581R4 AR 3789, 1804: ReSID-FP 6581R3 4485, 1805: ReSID-FP 6581R4 1986S, 1808: ReSID-FP 8580R5 3691, 1809: ReSID-FP 8580R5 3691 + digiboost, 1810: ReSID-FP 8580R5 1489, 1811: ReSID-FP 8580R5 1489D).

`-sidfilters`
`+sidfilters`
: Enable (`SidFilters=1`) or disable (`SidFilters=0`) emulation of the built-in SID filters.

`-residsamp METHOD`
: Specifies the sampling method; fast (`SidResidSampling=0`), interpolating (`SidResidSampling=1`), resampling (`SidResidSampling=2`), fast resampling (`SidResidSampling=3`).

`-residpass PERCENTAGE`
: Specifies the resampling filter passband in percentage of the total bandwidth (`SidResidPassband=0-90`).

`-residgain PERCENTAGE`
: Specifies reSID gain in percent (90 - 100).

### 7.1.5 C64 I/O extension settings

I/O extensions are (usually) cartridges which do not map into ROM space, but use only the I/O space at address range $DE00 . . . $DEFF and/or $DF00 . . . $DFFF.

Please use these extensions only when needed, as they might cause compatibility problems.

The following I/O extensions are available:

- ACIA (Swiftlink, Turbo232)
- DigiMAX
- Ethernet (The Final Ethernet, RR-Net)
- GEO-RAM

- MIDI (Passport, Datel, Maplin, Namesoft, Sequential)
- REU - The "RAM Expansion Module" extension emulates a standard Commodore RAM Expansion Unit; this can be used with GEOS and other programs that are designed to take advantage of it. This currently works only in the C64 emulator.
- SFX Sound Expander
- SFX Sound Sampler

#### 7.1.5.1 C64 I/O extension resources

TODO

`Acia1Enable`
: Boolean specifying whether the ACIA (Swiftlink, Turbo232) cartridge should be emulated or not.

`DIGIMAX`
: Boolean specifying whether the DigiMAX cartridge should be emulated or not.

`DIGIMAXbase`
`ETHERNET_INTERFACE`
`ETHERNET_DISABLED`
`ETHERNET_ACTIVE`
`ETHERNET_AS_RR`
`GEORAM`
: Boolean specifying whether the GEO-RAM cartridge should be emulated or not.

`GEORAMfilename`
`GEORAMImageWrite`
`GEORAMsize`
`MIDIEnable`
: Boolean specifying whether the MIDI cartridge should be emulated or not.

`MIDIMode`

`REU`
: Boolean specifying whether the RAM Expansion Module should be emulated or not.

`REUfilename`
`REUImageWrite`
`REUsize`

`SFXSoundExpander`
: Boolean specifying whether the SFX Sound Expander should be emulated or not.

`SFXSoundExpanderChip`
`SFXSoundSampler`
: Boolean specifying whether the SFX Sound Sampler should be emulated or not.

#### 7.1.5.2 C64 I/O extension command-line options

`-acia1`
`+acia1`
: Enable/Disable the $DE** ACIA RS232 interface emulation

`-digimax`
`+digimax` Enable/Disable the DigiMAX cartridge

`-digimaxbase <base address>`
Base address of the DigiMAX cartridge

`-miditype <0-4>`
MIDI interface type (0: Sequential, 1: Passport, 2: DATEL, 3: Namesoft, 4: Maplin)

`-midi`
`+midi` Enable/Disable MIDI emulation

`-midiin <name>`
Specify MIDI-In device

`-midiout <name>`
Specify MIDI-Out device

`-mididrv <driver>`
Specify MIDI driver (0 = OSS, 1 = ALSA)

`-georam`
`+georam` Enable/Disable the GEORAM expansion unit

`-cartgeoram <name>`
Attach raw GEO-RAM cartridge image

`-georamimage <name>`
Specify name of GEORAM image

`-georamimagerw`
`+georamimagerw`
Allow/Disallow writing to GEORAM image

`-georamsize <size in KB>`
Size of the GEORAM expansion unit

`-reu`
`+reu` Enable (`REU=1`) or disable (`REU=0`) emulation of the RAM Expansion Module.

`-cartreu <name>`
Attach raw REU cartridge image

`-reuimage <name>`
Specify name of REU image

`-reuimagerw`
`+reuimagerw`
Allow/Disallow writing to REU image

`-reusize <size in KB>`
Size of the RAM expansion unit

`-sfxse`
`+sfxse` Enable/Disable the SFX soundexpander cartridge

`-sfxsetype <type>`
Set YM chip type (3526 / 3812)

`-sfxss`
`+sfxss` Enable/Disable the SFX Sound Sampler cartridge

`-tfe`
`+tfe` Enable/Disable the TFE ("The Final Ethernet") unit

`-tfeif <name>`
Set the system ethernet interface for TFE emulation

`-tferrnet`
`+tferrnet`
Enable/Disable RRNet mode of TFE emulation

### 7.1.6 C64/128 system ROM settings

These settings can be used to control what system ROMs are loaded in the C64/128 emulators at startup. They cannot be changed from the menus.

#### 7.1.6.1 C64/128 system ROM resources

`KernalName`
String specifying the name of the Kernal ROM (default '`kernal`').

`BasicName`
String specifying the name of the Basic ROM (default '`basic`'). In the C128 emulator, the ROM image must actually include the editor ROM too.

`ChargenName`
String specifying the name of the character generator ROM (default '`chargen`').

`KernalRev`
String specifying the Kernal revision. This resource can be used to control what revision of the C64 kernal is being used; it cannot be changed at runtime. VICE is able to automatically convert one ROM revision into another, by manually patching the loaded image. This way, it is possible to use any of the ROM revisions without changing the ROM set. Valid values are:

`0` Kernal revision 0;

`3` Kernal revision 3;

`sx`
`67` Commodore SX-64 ROM;

`100`

`4064` Commodore 4064 (also known as "PET64" or "Educator 64") ROM.

#### 7.1.6.2 C64/128 system ROM command-line options

`-kernal NAME`
Specify '`NAME`' as the Kernal ROM file (`KernalName`).

`-basic NAME`
Specify '`NAME`' as the Basic ROM file (`BasicName`).

`-chargen NAME`
Specify '`NAME`' as the character generator ROM file (`ChargenName`).

`-kernalrev REVISION`
Specify Kernal revision (`KernalRev`).

## 7.2 C128-specific commands and settings

### 7.2.1 VDC settings

#### 7.2.1.1 VDC command-line options

`-VDCvcache`
`+VDCvcache`
Enable/Disable the video cache

`-VDCdsize`
`+VDCdsize`
Enable/Disable double size

`-VDCdscan`
`+VDCdscan`
Enable/Disable double scan

`-VDChwscale`
`+VDChwscale`
Enable/Disable hardware scaling

`-VDCintpal`
Use an internal calculated palette

`-VDCextpal`
Use an external palette (file)

`-VDCpalette <name>`
Specify name of file of external palette

`-VDCfulldevice <device>`
Select fullscreen device

`-VDCXRANDRfullmode <mode>`
Select fullscreen mode

`-VDCVidmodefullmode <mode>`
Select fullscreen mode

`-VDC16KB` Set the VDC memory size to 16KB

`-VDC64KB` Set the VDC memory size to 64KB

`-VDCRevision <number>`
Set VDC revision (0..2)

## 7.3 C64DTV-specific commands and settings

This section lists the settings and commands that are C64DTV specific and thus are not present in the other emulators.

### 7.3.1 C64DTV ROM image

The DTV has a 2MB Flash chip which contains the kernal, basic and character set ROMs along with other data, such as games in the case of the original C64DTV ROM.

The image file is a dump of the flash chip. It is exactly 2MB (2097152 bytes).

If you do not have a suitable image file, an image using the C64 kernal, basic and charset is automatically created.

If writing to the C64DTV ROM is enabled, the image file is rewritten with the current data when exiting x64dtv.

Note that x64dtv tries to load the image file from the C64DTV directory first, and if it isn't found there, x64dtv tries to load it from the current directory. If you do not have '`dtvrom.bin`' in your C64DTV directory and writing to DTV ROM is enabled, the '`dtvrom.bin`' file is created to the current directory.

NOTE: The original C64DTV ROM has somewhat distorted colors, normally you should use a patched rom.

`-c64dtvromimage NAME`
: Specify '`NAME`' as the C64DTV ROM image

`-c64dtvromrw`
`+c64dtvromrw`
: Enable or disable writing to C64DTV ROM image

The trueflashfs option is analogous to True drive emulation. If disabled, any file access to the flash filesystem (device 1) will go to the local file system instead.

`-trueflashfs`
`+trueflashfs`
: Enable or disable true hardware flash file system

`-fsflash NAME`
: Specify '`NAME`' as directory for flash file system device

### 7.3.2 DTV revision

The DTV revision 2 has a bug in the Blitter. Using revision 3 is recommended. Emulation of DTV revision 2 including Blitter bug is intended for testing DTV software.

`-dtvrev REVISION`
: Specify DTV '`REVISION`' (2 or 3)

### 7.3.3 LumaFix

The PAL C64DTVs have wrong resistors in the video output circuit, which causes incorrect luminances. Several hardware solutions ("LumaFixes") have been developed to fix this flaw.

The fixed video output is emulated by selecting "New Luminances". The unmodified C64DTV video output can be emulated with "Old Luminances".

The default setting is "New Luminances".

### 7.3.4 Userport

The C64DTV userport emulation currently supports three devices: Hummer ADC, userport joystick and PS/2 mouse.

The joystick that controls either the Hummer ADC or userport joystick can be selected using the same parameter or menu option.

While using the Hummer ADC, joystick UP and DOWN are mapped to the Hummer buttons A and B respectively. LEFT and RIGHT set the ADCs output to 0 and 255. Centering the joystick results in the ADC value of 128.

Currently the Hummer ADC and userport joystick are mutually exclusive. This means that enabling one disables the other. PS/2 mouse emulation can be used simultaneously with either Hummer ADC or userport joystick.

`-hummeradc`
`+hummeradc`
: Enable/Disable Hummer ADC Aktivieren

`-ps2mouse`
`+ps2mouse`
: Enable or disable PS/2 mouse on userport

### 7.3.5 Debug

Debugging information on Blitter, DMA and Flash can be enabled with command line parameters. This can be useful for DTV software development.

`-dtvblitterlog`
`+dtvblitterlog`
: Enable or disable DTV Blitter log

`-dtvdmalog`
`+dtvdmalog`
: Enable or disable DTV DMA log

`-dtvflashlog`
`+dtvflashlog`
: Enable or disable DTV Flash log

### 7.3.6 Monitor DTV features

Currently the registers A, Y and X are registers R0, R1 and R2 regardless of the mapping, which can be seen and modified via the registers ACM and XYM.

The monitor can access all 2MB of RAM and 2MB of Flash, but only 64 kB at a time. The 64kB bank can be selected with "bank ram00".."ram1f" for RAM and "bank rom00".."rom1f" for Flash.

The "load" command can load large files (>64kB) correctly if the bank is set to "ramXX", where XX is the starting bank (usually "bank00").

## 7.4 VIC20-specific commands and settings

This section lists the settings and commands that are VIC20-specific and thus are not present in the other emulators.

### 7.4.1 Using cartridge images

As with the C64 (see Section 7.1.1 [C64 cartridges], page 44), it is possible to attach several types of cartridge images:

- 4 or 8 Kbyte cartridges located at $2000;
- 4 or 8 Kbyte cartridges located at $4000;
- 4 or 8 Kbyte cartridges located at $6000;
- 4 or 8 Kbyte cartridges located at $A000;
- 4 Kbyte cartridges located at $B000.

This can all be done via the "Attach cartridge image..." command in the left-button menu. It is also possible to let xvic "guess" the type of cartridge using "Smart-attach cartridge image...".

Notice that several cartridges are actually made up of two pieces (and two files), that need to be loaded separately at different addresses. In that case, you have to know the addresses (which are usually specified in the file name) and use the "attach" command twice.

A special kind of cartridge file is where the two files mentioned above are concatenated (with removing the two byte load address of the second image) into one 16k image. There are only few of those images, though. Normally the second part is located at $A000. Vice can now attach such concatenated files at the start address $2000, $4000, and $6000. The second half of such an image is moved to $A000. If you encounter 16k images that have the second half not at $A000 you can split the image into two halfs (i.e. one 8194 byte and one 8192 byte, because the first has the load address) and attach both files separately.

One cartridge that is currently only partially supported here is the VIC1112 IEEE488 interface. You have to load the ROM as a cartridge, but you also have to enable the IEEE488 hardware by menu.

### 7.4.2 VIC settings

#### 7.4.2.1 VIC resources

`VICVideoCache`
: Boolean specifying whether the video cache is turned on.

`VICDoubleSize`
: Boolean specifying whether double-size mode is turned on.

`VICDoubleScan`
: Boolean specifying whether double-scan mode is turned on.

`VICPaletteFile`
: String specifying the name of the palette file being used. The '`.vpl`' extension is optional.

#### 7.4.2.2 VIC command-line options

`-VICvcache`
`+VICvcache`
: Enable/disable the video cache (`VICVideoCache=1`, `VICVideoCache=0`).

`-VICdsize`
`+VICdsize`
Enable/disable the double size mode (`VICDoubleSize=1`, `VICDoubleSize=0`).

`-VICdscan`
`+VICdscan`
Enable/disable the double scan mode (`VICDoubleScan=1`, `VICDoubleScan=0`).

`-VIChwscale`
`+VIChwscale`
Enable/Disable hardware scaling

`-VICscale2x`
`+VICscale2x`
Enable/Disable Scale2x

`-VICpalette NAME`
Specify `NAME` as the palette file (`VICPaletteFile`).

`-VICintpal`
Use an internal calculated palette

`-VICextpal`
Use an external palette (file)

`-VICfulldevice <device>`
Select fullscreen device

`-VICXRANDRfullmode <mode>`
Select fullscreen mode

`-VICVidmodefullmode <mode>`
Select fullscreen mode

### 7.4.3 Changing memory configuration

It is possible to change the VIC20 memory configuration in two ways: by enabling and/or disabling certain individual memory blocks, or by choosing one among a few typical memory configurations. The former can be done by modifying resource values directly or from the right-button menu; the latter can only be done from the menu.

There are 5 RAM expansion blocks in the VIC20, numbered 0, 1, 2, 3 and 5:

- block 0 (3 Kbytes at $0400-$0FFF);
- block 1 (8 Kbytes at $2000-$3FFF);
- block 2 (8 Kbytes at $4000-$5FFF);
- block 3 (8 Kbytes at $6000-$7FFF);
- block 5 (8 Kbytes at $A000-$BFFF).

These blocks are called *expansion blocks* because they are not present a stock ("unexpanded") machine. Each of them is associated to a boolean `RamBlockX` resource (where `X` is the block number) that specifies whether the block is enabled or not.

There are also some common memory configurations you can pick from the right-button menu:

- no RAM expansion blocks at all;
- all RAM expansion blocks enabled;
- 3K expansion (only block 0 is enabled);
- 8K expansion (only block 1 is enabled);
- 16K expansion (only blocks 1 and 2 are enabled);
- 24K expansion (only blocks 1, 2 and 3 are enabled).

#### 7.4.3.1 VIC20 memory configuration resources

`RAMBlock0`
`RAMBlock1`
`RAMBlock2`
`RAMBlock3`
`RAMBlock5`

Booleans specifying whether RAM blocks 0, 1, 2, 3 and 5 must be enabled.

#### 7.4.3.2 VIC20 memory configuration command-line options

`-memory CONFIG`

Specify memory configuration. It must be a comma-separated list of options, each of which can be one the following:

- `none` (no extension);
- `all` (all blocks);
- `3k` (3k space in block 0);
- `8k` (first 8k extension block);
- `16k` (first and second 8k extension blocks);
- `24k` (first, second and 3rd extension blocks);
- `0`, `1`, `2`, `3`, `5` (memory in respective blocks);
- `04`, `20`, `40`, `60`, `A0` (memory at respective address.

For example,

```
xvic -memory none
```

gives an unexpanded VIC20. While

```
xvic -memory 60,a0
```

or

```
xvic -memory 3,5
```

enables memory in blocks 3 and 5, which is the usual configuration for 16k ROM modules.

### 7.4.4 VIC20 system ROM settings

These settings can be used to control what system ROMs are loaded in the VIC20 emulator at startup. They cannot be changed from the menus.

#### 7.4.4.1 VIC20 system ROM resources

`KernalName`
: String specifying the name of the Kernal ROM (default '`kernal`').

`BasicName`
: String specifying the name of the Basic ROM (default '`basic`').

`ChargenName`
: String specifying the name of the character generator ROM (default '`chargen`').

`GenericCartridgeFile2000`
`GenericCartridgeFile4000`
`GenericCartridgeFile6000`
`GenericCartridgeFileA000`
`GenericCartridgeFileB000`
: String specifying the name of the respective cartridge ROM images.

#### 7.4.4.2 VIC20 system ROM command-line options

`-kernal NAME`
: Specify '`NAME`' as the Kernal ROM file (`KernalName`).

`-basic NAME`
: Specify '`NAME`' as the Basic ROM file (`BasicName`).

`-chargen NAME`
: Specify '`NAME`' as the character generator ROM file (`ChargenName`).

`-cart2 NAME`
`-cart4 NAME`
`-cart6 NAME`
`-cartA NAME`
`-cartB NAME`
: Specify '`NAME`' as the cartridge image to attach. (`CartridgeFile2000`,...,`CartridgeFileB000`).

## 7.5 PLUS 4-specific commands and settings

### 7.5.1 TED settings

#### 7.5.1.1 TED command-line options

`-TEDvcache`
`+TEDvcache`
: Enable/Disable the video cache

`-TEDdsize`
`+TEDdsize`
: Enable/Disable double size

`-TEDdscan`
`+TEDdscan`
: Enable/Disable double scan

`-TEDscale2x`
`+TEDscale2x`
: Enable/Disable Scale2x filter

`-TEDhwscale`
`+TEDhwscale`
: Enable/Disable hardware scaling

`-TEDintpal`
: Use an internal calculated palette

`-TEDextpal`
: Use an external palette (file)

`-TEDpalette <name>`
: Specify name of file of external palette

`-TEDfulldevice <device>`
: Select fullscreen device

`-TEDXRANDRfullmode <mode>`
: Select fullscreen mode

`-TEDVidmodefullmode <mode>`
: Select fullscreen mode

## 7.6 PET-specific commands and settings

This section lists the settings and commands that are PET-specific and thus are not present in the other emulators.

### 7.6.1 Changing PET model settings

With `xpet`, it is possible to change at runtime the characteristics of the emulated PET so that it matches (or not) the ones of a certain PET model, and it is also possible to select from a common set of PET models so that all the features are selected accordingly.

The former is done by changing the following resources (via resource file, command line options or right-menu items):

`RamSize` Size of memory in kByte. 96k denotes a 8096, 128k a 8296.

`IOSize` Size of I/O area in Byte. Either 2048 or 256 for 8296.

`VideoSize`
: The number of columns on the screen (40 or 80). A 0 auto-detects this from the ROM.

`Ram9` The 8296 can map RAM into the address range $9***

`RamA` The 8296 can map RAM into the address range $A***

`SuperPET` This resource enables the SuperPET (MicroMainFrame 9000) I/O and disables the 8x96 mappings.

`Basic1` If (by checksum) a version 1 kernal is detected, then the kernal ROM is patched to make the IEEE488 interface work.

`Basic1Chars`
: Exchanges some character in the character ROM that have changed between the first PET 2001 and all newer versions.

`EoiBlank`
: This resource enables the "blank screen on EOI" feature of the oldest PET 2001.

`DiagPin`
: Set the diagnositc pin on the PET userport (see below).

`ChargenName`
: Specify '`NAME`' as the character generator ROM file

`KernalName`
: Specify '`NAME`' as the kernal ROM file. This file contains the complete BASIC, EDITOR and KERNAL ROMs and is either 16k (BASIC 1 and 2) or 20k (BASIC 4) in size.

`EditorName`
: Specify '`NAME`' as the editor ROM file. This file contains an overlay for the editor ROM at $E000-$E7FF if necessary.

`RomModule9Name`
: Specify '`NAME`' as the $9*** Expansion ROM file. This file contains an expansion ROM image of 4k.

`RomModuleAName`
: Specify '`NAME`' as the $A*** Expansion ROM file. This file contains an expansion ROM image of 4k.

`RomModuleBName`
: Specify '`NAME`' as the $B*** Expansion ROM file. This file contains an expansion ROM image of 4k. This file overlays the lowest 4k of a BASIC 4 ROM.

Choosing a common PET model is done from the right-button menu instead, by choosing an item from the "Model defaults" submenu. Available models are:

- PET 2001-8N
- PET 3008
- PET 3016
- PET 3032
- PET 3032B
- PET 4016
- PET 4032
- PET 4032B
- PET 8032
- PET 8096
- PET 8296
- SuperPET

Notice that this will **reset the emulated machine**.

It is also possible to select the PET model at startup, with the `-model` command-line option: for example, '`xpet -model 3032`' will emulate a PET 3032 while '`xpet -model 8296`' will emulate a PET 8296.

### 7.6.2 CRTC Settings

#### 7.6.2.1 CRTC resources

`Crtc` Enables CRTC 6545 emulation (all models from 40xx and above)

`CrtcVideoCache`
: Boolean specifying whether the video cache is turned on.

`CrtcDoubleSize`
: Boolean specifying whether double-size mode is turned on.

`CrtcDoubleScan`
: Boolean specifying whether double-scan mode is turned on.

`CrtcPaletteFile`
: String specifying the name of the palette file being used. The '`.vpl`' extension is optional.

#### 7.6.2.2 CRTC command-line options

`-Crtcvcache`
`+Crtcvcache`
: Enable/Disable the video cache

`-Crtcdsize`
`+Crtcdsize`
: Enable/Disable double size

`-Crtcdscan`
`+Crtcdscan`
: Enable/Disable double scan

`-Crtcscale2x`
`+Crtcscale2x`
: Enable/Disable Scale2x filter

`-Crtchwscale`
`+Crtchwscale`
: Enable/Disable hardware scaling

`-Crtcintpal`
: Use an internal calculated palette

`-Crtcextpal`
: Use an external palette (file)

`-Crtcpalette NAME`
: Specify `NAME` as the palette file (`CrtcPaletteFile`).

`-Crtcfulldevice <device>`
: Select fullscreen device

`-CrtcXRANDRfullmode <mode>`
: Select fullscreen mode

`-CrtcVidmodefullmode <mode>`
: Select fullscreen mode

### 7.6.3 The PET diagnostic pin

It is possible to enable or disable emulation of the PET diagnostic pin via the `DiagPin` resource, or the "PET userport diagnostic pin" item in the right-button menu.

When the diagnostic pin is set, the Kernal does not try to initialize the BASIC, but directly jumps into the builtin machine monitor.

### 7.6.4 PET command line options

These are the commandline options specific for the CBM-II models.

`-model MODEL`
: Specify the PET model you want to emulate.

`-kernal NAME`
: Specify '`NAME`' as the Kernal/BASIC ROM file (`KernalName`).

`-editor NAME`
: Specify '`NAME`' as the editor ROM file (`EditorName`).

`-chargen NAME`
: Specify '`NAME`' as the character generator ROM file (`ChargenName`).

`-rom9 NAME, -romA NAME, -romB NAME`
: Specify '`NAME`' as the ROM image file for the respective cartridge areas (`RomModule9Name`, `RomModuleAName`, `RomModuleBName`).

`-petram9, +petram9`
: Switch on RAM mapping on addresses $9000-$9fff (`Ram9`).

`-petramA, +petramA`
: Switch on RAM mapping on addresses $a000-$afff (`RamA`).

`-superpet, +superpet`
: Enable/Disable SuperPET I/O emulation (`SuperPET`).

`-basic1, +basic1`
: Enable/Disable patching the IEEE488 section of the PET2001 ROM when detected (`Basic1`).

`-basic1char, +basic1char`
: Enable/Disable PET 2001 character generator (`Basic1Chars`).

`-eoiblank, +eoiblank`
: Enable/Disable EOI blanking the screen (`EoiBlank`).

`-diagpin`
`+diagpin`
: Enable (`DiagPin=1`) or disable (`DiagPin=0`) the diagnostic pin at the PET userport.

`-petreu, +petreu`
: Enable or disable the PET Memory Expansion Unit.

`-userportdac, +userportdac`
: Enable or disable the userport DAC.

### 7.6.5 Changing screen colors

It is also possible to choose what color set is used for the emulation window. This is done by specifying a palette file name (see Section 4.3 [Palette files], page 20) in the `PaletteName` resource. The menu provides the following values:

- `green.vpl` (default, "green)"), the good old green-on-black feeling;
- `amber.vpl` ("amber"), an amber phosphor lookalike;
- `white.vpl` ("white"), simple white-on-black palette.

## 7.7 CBM-II-specific commands and settings

This section lists the settings and commands that are CBM-II-specific and thus are not present in the other emulators.

### 7.7.1 Changing CBM-II model

With `xcbm2`, it is possible to change at runtime the characteristics of the emulated CBM so that it matches (or not) the ones of a certain CBM model, and it is also possible to select from a common set of CBM models so that all the features are selected accordingly.

The former is done by changing the following resources (via resource file, command line options or right-menu items):

`UseVicII` Whether to use VIC-II for video output (value 1) or the CRTC for the other machines (value 0)

`RamSize` Size of memory in kByte. Possible values are 128, 256, 512 and 1024

`Ram08, Ram1, Ram2, Ram4, Ram6, RamC`
Expanded CBM-II models could map RAM to the expansion ROM areas at $0800-$0fff, $1000-$1fff, $2000-$3FFF, $4000-$5FFF, $6000-$7FFF and $c000-$cfff respectively.

`Cart2Name, Cart4Name, Cart6Name`
Specify '`NAME`' as the $2000-$3FFF, $4000-$5FFF or $6000-$6FFF Expansion ROM file. This file contains an 8k ROM dump.

`ModelLine`
The CBM-II business models have two hardcoded lines at one of the I/O ports. From those lines the kernal determines how it should init the CRTC video chip for either 50Hz (Europe) or 60Hz (North America), and either for 8 (C6x0) or 14 (C7x0) scanlines per character. 0 = CBM 7x0 (50Hz), 1 = 60Hz C6x0, 2 = 50Hz C6x0).

Choosing a common CBM-II model is done from the right-button menu instead, by choosing an item from the "Model defaults" submenu. Available models are:

- C510 (128k RAM)
- C610 (128k RAM)
- C620 (256k RAM)
- C620+ (1024k RAM, expanded)
- C710 (128k RAM)

- C720 (256k RAM)
- C720+ (1024k RAM, expanded)

Notice that this will **reset the emulated machine**.

**Warning:** At this time switching between 510 and other machines during runtime is not supported and will not work.

It is also possible to select the CBM model at startup, with the `-model` command-line option: for example, '`xcbm2 -model 610`' will emulate a CBM 610 while '`xcbm2 -model 620`' will emulate a CBM 620. Notably this is the only way to start a C510 emulation, with `-model 510`.

### 7.7.2 CBM-II command line options

These are the commandline options specific for the CBM-II models.

`-usevicii`
`+usevicii`
: Specify whether to use (-usevicii) or not to use (+usevicii) the VIC-II emulation.

`-kernal NAME`
: Specify '`NAME`' as the Kernal ROM file (`KernalName`).

`-basic NAME`
: Specify '`NAME`' as the Basic ROM file (`BasicName`).

`-chargen NAME`
: Specify '`NAME`' as the character generator ROM file (`ChargenName`).

`-cart2 NAME, -cart4 NAME, -cart6 NAME`
: Specify '`NAME`' as the ROM image file for the respective cartridge areas (`Cart2Name`, `Cart4Name`, `Cart6Name`).

`-ram08, -ram1, -ram2, -ram4, -ram6, -ramC`
: Switch on RAM mapping in bank 15 on addresses $0800-$0fff, $1000-$1fff, $2000-$3fff, $4000-$5fff, $6000-$7fff resp (`Ram08`, `Ram1`, `Ram2`, `Ram4`, `Ram6`, `RamC`).

`-modelline`
: Define the hardcoded model switch in the CBM-II models.

### 7.7.3 Changing screen colors

It is also possible to choose what color set is used for the emulation window. This is done by specifying a palette file name (see Section 4.3 [Palette files], page 20) in the `PaletteName` resource. The menu provides the following values:

- `green.vpl` (default, "green"), the good old green-on-black feeling;
- `amber.vpl` ("amber"), an amber phosphor lookalike;
- `white.vpl` ("white"), simple white-on-black palette.

# 8 Snapshots

Every VICE emulator has a built-in snapshot feature, that saves the complete emulator state into one file for later use. You can therefore save the emulator state - including the state of the game you are playing for example - in a single file.

## 8.1 Snapshot usage

A snapshot is one file containining the complete emulator state. A snapshot file can be generated by selecting the "Save snapshot" command at any time. This will pop up a requester from which you can specify whether the snapshot should also contain the disk and ROM status.

A snapshot file can be used to restore the emulator state by selecting the `load snapshot` menu entry at any time. Unfortunately attached ROM images/cartridges are only supported in the VIC20, the PET and the CBM-II emulators at this time.

The memory configuration of the emulator is saved in the snapshot file as well. This configuration is restored when the snapshot is loaded.

A quick snapshot can now be made by pressing the `M-F11` key and reloaded by pressing the `M-F10` key.

## 8.2 Snapshot format

A snapshot file consists of several modules of mostly different types. Each module has a name and saves the state of an entity like a CIA, the CPU, or the memory.

### 8.2.1 Emulator modules

This section lists the modules that are contained in each of the emulators snapshot files.

#### 8.2.1.1 x64 modules

The modules in the x64 emulator are:

| Name | Type | Description |
|---|---|---|
| MAINCPU | 6502 | The Main CPU - although it is a 6510, only the 6502 core is saved here |
| C64MEM | Memory | Holds the RAM contents of the C64. Also the CPU I/O register contents are saved here. |
| C64ROM | ROM images | Dump of the system ROMs |
| VIC-II | 656* | The VIC-II of the C64/128 |
| CIA1 | 6526 | The CIA for the interrupts and the keyboard |
| CIA2 | 6526 | The CIA for the userport, IEC-bus and RS232. |
| SID | 6581 | The SID sound chip of the C64/C128 |
| REU* | | The RAM Extension Unit state (optional) |
| ACIA1 | 6551 | An ACIA (RS232 interface) at $DE00 (optional) |

| | | |
|---|---|---|
| TPI | 6525 | A TPI at $DF00 for a parallel IEEE488 interface (optional) |
| * | Drive modules | The emulated drive(s) have their own modules see Section 8.2.1.6 [Drive modules], page 78 |

Some of the modules are optional and are only saved if the specific feature is enabled at save-time. If the module is found when restoring the state the optional features are enabled, and disabled otherwise.

### 8.2.1.2 x128 modules

The modules in the x128 emulator are:

| Name | Type | Description |
|---|---|---|
| MAINCPU | 6502 | The Main CPU - although it is a 6510, only the 6502 core is saved here |
| C128MEM | Memory | Holds the RAM contents of the C64. Also the CPU I/O register contents are saved here. |
| C128ROM | ROM images | Dump of the system ROMs |
| VIC-II | 656* | The VIC-II of the C64/128 |
| CIA1 | 6526 | The CIA for the interrupts and the keyboard |
| CIA2 | 6526 | The CIA for the userport, IEC-bus and RS232. |
| SID | 6581 | The SID sound chip of the C64/C128 |
| ACIA1 | 6551 | An ACIA at $DE00 (optional) |
| TPI | 6525 | A TPI at $DF00 for a parallel IEEE488 interface (optional) |
| * | Drive modules | The emulated drive(s) have their own modules see Section 8.2.1.6 [Drive modules], page 78 |

Some of the modules are optional and are only saved if the specific feature is enabled at save-time. If the module is found when restoring the state the optional features are enabled, and disabled otherwise.

Not yet supported are the 80 column video chip, cartridges and RAM expansion unit.

### 8.2.1.3 xvic modules

The modules in the xvic emulator are:

| Name | Type | Description |
|---|---|---|
| MAINCPU | 6502 | The Main CPU |
| VIC20MEM | Memory | Holds the RAM contents of the VIC20. |
| VIC20ROM | ROM images | Holds the ROM images of the VIC20, including possibly attached cartridges |
| VIC-I | 656* | The VIC-I of the VIC20 |
| VIA1 | 6522 | The VIA for the interrupts and the keyboard |

| | | |
|---|---|---|
| VIA2 | 6522 | The VIA for the userport, IEC-bus and RS232. |
| * | Drive modules | The emulated drive(s) have their own modules see Section 8.2.1.6 [Drive modules], page 78 |

### 8.2.1.4 xpet modules

The modules in the xpet emulator are:

| Name | Type | Description |
|---|---|---|
| MAINCPU | 6502 | The Main CPU |
| PETMEM | Memory | Holds the RAM contents of the PET. |
| PETROM | ROM images | Holds the ROM images of the PET, including possibly attached cartridges |
| CRTC | 6545 | The CRTC of the PET. This is also included if it is a dump of a PET without CRTC, because the video state is saved here anyway. |
| PIA1 | 6520 | The PIA for the interrupts, tape and the keyboard |
| PIA2 | 6520 | The PIA for the IEEE488-bus |
| VIA | 6522 | The VIA for IEEE488, userport, sound |
| ACIA1 | 6551 | The ACIA for the SuperPET. This module is optional. |
| * | Drive modules | The emulated drive(s) have their own modules see Section 8.2.1.6 [Drive modules], page 78 |

### 8.2.1.5 xcbm2 modules

The modules in the xcbm2 emulator are:

| Name | Type | Description |
|---|---|---|
| MAINCPU | 6502 | The Main CPU - although it is a 6509, only the 6502 core is saved here |
| CBM2MEM | Memory | Holds the RAM contents of the CBM-II models. Also holds the exec-bank and indirection bank registers |
| C500DATA | | Holds additional state information necessary for the C500 (e.g. cycles till the next IRQ) |
| CBM2ROM | Memory | optional. Holds the ROM images. |
| CRTC | 6545 | The video chip for the C6*0 and C7*0 models (only those models). |
| VIC-II | 656? | The video chip for the C5*0 models (only the C5*0 models). |
| CIA1 | 6526 | The CIA for IEEE 488 and userport. |
| TPI1 | 6525 | TPI 1 for IEEE488 |

| | | |
|---|---|---|
| TPI2 | 6525 | TPI 2 for interrupts and keyboard. |
| ACIA1 | 6551 | The RS232 interface |
| SID | 6581 | The CBM2s SID sound chip |
| * | Drive modules | The emulated drive(s) have their own modules see Section 8.2.1.6 [Drive modules], page 78 |

The snapshot either contains CRTC or VIC-II snapshot modules, but not both. Currently switching between the two video emulations is not possible at runtime, so only snapshots that fit the current UseVicII resource are accepted.

#### 8.2.1.6 Drive modules

The modules for the real disk drive emulation are included in the emulator when the emulation is enabled during the writing of the snapshot.

| Name | Type | Description |
|---|---|---|
| *CPU | 6502 | The Drive 0 CPU |
| * | * | * |

### 8.2.2 Module formats

This section shows the basic module framework and the contents of the different types of modules.

The single chip modules contain the **chip** state, not the state of the emulator. We tried to make the format as implementation-independent as possible, to allow reuse of snapshots in later versions of this emulator, or even in other emulators.

#### 8.2.2.1 Terminology

In this section we use certain abbreviations to define the types of the data saved in the snapshot.

`BYTE` 8 bit integer.

`WORD` 16 bit integer. Saved with low-byte first, high-byte last.

`DWORD` 32 bit integer. Saved with low-word first, then high-word. Each word saved with its low-byte first.

`ARRAY` Array of BYTE values. Length depends on the description.

The tables for the single modules state the type, name and description of the data saved in the modules. The data is saved in the order it is in the tables, so no offset is given.

#### 8.2.2.2 Module framework

The VICE snapshot file starts with the magic string and includes the fileformat version number.

| Type | Name | Description |
|---|---|---|
| 19 BYTE | MAGIC | "VICE Snapshot File\032", padded with 0 |
| BYTE | VMAJOR | fileformat major version number |
| BYTE | VMINOR | fileformat minor version number |

| | | |
|---|---|---|
| 16 BYTE | MACHINENAME | Name of emulated machine, like "PET", "CBM-II", "VIC20", "C64" or "C128". zerobyte-padded. |

The file header is followed by a number of different snapshot modules.

Each module has a header with the information given in the table below. The header includes two version numbers, VMAJOR and VMINOR. Modules with the same VMAJOR should be able to be exchanged. I.e. higher VMINOR numbers only append to the data for lower VMINOR. This additional data is ignored by older restore routines. The other way around newer restore routines must accept the fewer info from lower VMINOR dumps. Changes in VMAJOR might introduce any incompatibility you like, but that's what VMAJOR is for after all :-)

| Type | Name | Description |
|---|---|---|
| 16 BYTE | MODULENAME | The name of the module in ASCII, padded with 0 to 16 byte. |
| BYTE | VMAJOR | major version number |
| BYTE | VMINOR | minor version number |
| DWORD | SIZE | size of the module, including this header |

### 8.2.2.3 CPU module

This module saves the core 6502 state. You will find a clock value there. All other modules save their own clock values relative to this value. However, the drive modules save their clocks relative to their appropriate CPUs of course.

**Warning:** This module is still under construction and saves some information that is not sure to be VICE-independent. If in doubt, read the source.

| Type | Name | Description |
|---|---|---|
| DWORD | CLK | the current CPU clock value. All other clock values are relative to this. |
| BYTE | AC | Accumulator |
| BYTE | XR | X index register |
| BYTE | YR | Y index register |
| BYTE | SP | Stack Pointer |
| WORD | PC | Programm Counter |
| BYTE | ST | Status Registers |
| DWORD | LASTOPCODE | ? |
| DWORD | IRQCLK | absolute CLK when the IRQ line came active |
| DOWRD | NMICLK | absolute CLK when the NMI line came active |
| DWORD | ? | ? |
| DWORD | ? | ? |

### 8.2.2.4 CIA module

The CIA 6526 is an I/O port chip with 2 8-bit I/O ports, a shift register, two timers, a Time of Day clock and interrupts.

Version numbers: Major 1, Minor 1.

| Type | Name | Description |
|---|---|---|
| BYTE | ORA | Output register A |
| BYTE | ORB | Output register B |
| BYTE | DDRA | Data direction register A |
| BYTE | DDRB | Data direction register B |
| WORD | TAC | Timer A counter value |
| WORD | TBC | Timer B counter value |
| BYTE | TOD_TEN | Time of Day - current tenth of second |
| BYTE | TOD_SEC | Time of Day - current seconds |
| BYTE | TOD_MIN | Time of Day - current minutes |
| BYTE | TOD_HR | Time of Day - current hours |
| BYTE | SDR | contents of shift register |
| BYTE | IER | mask of enabled interrupt masks |
| BYTE | CRA | Control register A |
| BYTE | CRB | Control register B |
| WORD | TAL | Timer A latch value |
| WORD | TBL | Timer B latch value |
| BYTE | IFR | mask of currently active interrupts |
| BYTE | PBSTATE | Bit 6/7 reflect the PB6/7 toggle bit state. Bit 2/3 reflect the corresponding port bit state. |
| BYTE | SRHBITS | number of half-bits to still shift in/out SDR |
| BYTE | ALARM_TEN | Time of Day - alarm tenth of second |
| BYTE | ALARM_SEC | Time of Day - alarm seconds |
| BYTE | ALARM_MIN | Time of Day - alarm minutes |
| BYTE | ALARM_HR | Time of Day - alarm hours |
| BYTE | READICR | current clock minus the clock when ICR was read last plus 128. |
| BYTE | TODLATCHED | Bit 0: 1= latched for reading, Bit 1: 2=stopped for writing |
| BYTE | TODL_TEN | Time of Day - latched tenth of second |
| BYTE | TODL_SEC | Time of Day - latched seconds |
| BYTE | TODL_MIN | Time of Day - latched minutes |
| BYTE | TODL_HR | Time of Day - latched hours |
| DWORD | TOD_TICKS | clk ticks till next tenth of second |
| – | – | The next items have been added in V1.1 |
| WORD | TASTATE | The state bits of the CIA timer A, according to ciatimer.h |
| WORD | TBSTATE | The state bits of the CIA timer B, according to ciatimer.h |

The last two items have been added in CIA snapshot version 1.1 due to the improved CIA emulation in the newer VICE versions. Some state bits correspond to the CIA state as described in the "A Software Model of the CIA 6526" document by Wolfgang Lorenz, some are delayed versions. For more read the source file `ciatimer.h`.

#### 8.2.2.5 VIA module

The VIA 6522 is the predecessor of the CIA and also an I/O port chip with 2 8-bit I/O ports, a shift register, two timers and interrupts.

Version numbers: Major 1, Minor 0.

| Type | Name | Description |
|---|---|---|
| BYTE | ORA | Output register A |
| BYTE | DDRA | Data direction register A |
| BYTE | ORB | Output register B |
| BYTE | DDRB | Data direction register B |
| WORD | T1L | Timer 1 Latch value |
| WORD | T1C | Timer 1 counter value |
| BYTE | T2L | Timer 2 latch (8 bit as only lower byte is used) |
| WORD | T2C | Timer 2 counter value |
| BYTE | RUNFL | bit 7: timer 1 will generate IRQ on underflow; bit 6: timer 2 will generate IRQ on underflow |
| BYTE | SR | Shift register value |
| BYTE | ACR | Auxiliary control register |
| BYTE | PCR | Peripheral control register |
| BYTE | IFR | active interrupts |
| BYTE | IER | interrupt mask |
| BYTE | PB7 | bit 7 = pb7 state |
| BYTE | SRHBITS | number of half-bits to shift out on SR |
| BYTE | CABSTATE | bit 7: state of CA2 pin, bit 6: state of CB2 pin |
| BYTE | ILA | Port A Input Latch (see ACR bit 0) |
| BYTE | ILB | Port B Input Latch (see ACR bit 1) |

#### 8.2.2.6 PIA module

The PIA 6520 is a chip with two I/O ports (Parallel Interface Adapter) and four additional handshake lines. The chip is pretty the same for Port A and B, only that Port A implements handshake on read operation and port B on write operation.

Version numbers: Major 1, Minor 0.

| Type | Name | Description |
|---|---|---|
| UBYTE | ORA | Output register A |
| UBYTE | DDRA | Data Direction Register A |
| UBYTE | CTRLA | Control Register A |
| UBYTE | ORB | Output register B |
| UBYTE | DDRB | Data Direction Register B |
| UBYTE | CTRLB | Control Register B |
| UBYTE | CABSTATE | Bit 7 = state of CA2, Bit 6 = state of CB2 |

#### 8.2.2.7 TPI module

The TPI 6525 is a chip with three I/O ports (Tri-Port-Interface). One of the ports can double as an interrupt prioritizer. Therefore we also have to save the states of the interrupt stack etc.

Version numbers: Major 1, Minor 0.

| Type | Name | Description |
|---|---|---|
| BYTE | PRA | Port A output register |
| BYTE | PRB | Port B output register |
| BYTE | PRC | Port C output register (doubles as IRQ latch register) |
| BYTE | DDRA | Port A data direction register |
| BYTE | DDRB | Port B data direction register |
| BYTE | DDRC | Port C data direction register (doubles as IRQ mask register) |
| BYTE | CR | Control Register |
| BYTE | AIR | Active interrupt register |
| BYTE | STACK | Interrupt stack - the interrupt bits that are not (yet) served. |
| BYTE | CABSTATE | State of CA/CB pins. Bit 7 = state of CA, Bit 6 = state of CB |

#### 8.2.2.8 RIOT module

The RIOT 6532 is a chip with two I/O ports, some RAM and a Timer. The chip contains 128 byte RAM, but the RAM is not saved in the RIOT snapshot, but in the memory section.

**Warning:** This module is still under construction

Version numbers: Major 0, Minor 0.

| Type | Name | Description |
|---|---|---|
| BYTE | ORA | Port A output register |
| BYTE | DDRA | Port A data direction register |
| BYTE | ORB | Port B output register |
| BYTE | DDRB | Port B data direction register |
| BYTE | EDGECTRL | Bit 0/1: A0/A1 address bits written to edgecontrol registers |
| BYTE | IRQFL | Bit 6/7: A6/A7 IRQ flag register. Bit 0: state of the IRQ line (0=inactive, 1=active) |
| BYTE | N | timer value |
| WORD | DIVIDER | Pre-scale divider value (1, 8, 64, or 1024) |
| WORD | REST | cycles since the last counter change |
| BYTE | IRQEN | Bit 0: 0= timer IRQ disabled, 1= timer IRQ enabled |

#### 8.2.2.9 SID module

**Warning:** This module is still under construction.

#### 8.2.2.10 ACIA module

The ACIA 6551 is an RS232 interface chip. VICE emulates RS232 connections via `/dev/ttyS*` (Unix) or `COM:` (DOS/WIN - not yet?). When saving a snapshot, those connections are of course lost. The state of the ACIA however is restored if possible. I.e. if a connection is already open when restoring the snapshot, this connection is used instead. If no connection is open, a carrier/DTR drop is emulated.

Version numbers: Major 1, Minor 0.

| Type | Name | Description |
|---|---|---|
| BYTE | TDR | Transmit Data Register |
| BYTE | RDR | Receiver Data Register |
| BYTE | SR | Status Register |
| BYTE | CMD | Command Register |
| BYTE | CTRL | Ctrl Register |
| BYTE | INTX | 0 = no data to tx; 1 = Data is being transmitted; 2 = Data is being transmitted while data in TDR waiting to be put to internal transmit register |
| DWORD | TICKS | Clock ticks till the next TDR empty interrupt |

#### 8.2.2.11 VIC-I module

**Warning:** This module is still under construction.

#### 8.2.2.12 VIC-II module

**Warning:** This module is still under construction.

#### 8.2.2.13 CRTC module

**Warning:** After VICE version 1.0 the CRTC emulation has improved considerably. Especially it is now cycle exact. Therefore a lot more variables must be saved. The snapshot module version jumped from 0.0 to 1.0. Newer versions of VICE can read the old snapshots, but older versions (1.0 and below) cannot read the new snapshots.

**Warning:** This module is still under construction. Especially the RASTERY and RASTERLINE values might be bogus.

Version numbers: Major 1, Minor 1.

| Type | Name | Description |
|---|---|---|
| | | Hardware options |
| WORD | VADDR_MASK | Mask of the address bits valid when accessing the video memory |
| WORD | VADDR_CHARSWITCH | If one bit in the video address is used to switch the character generator, it is masked here. |
| WORD | VADDR_CHAROFFSET | The offset in characters in the character generator that CHARSWITCH switches. |
| WORD | VADDR_REVSWITCH | If one bit in the video address inverts the screen, it is masked here. |

| | | |
|---|---|---|
| WORD | CHARGEN_MASK | size of character generator in byte - 1 |
| WORD | CHARGEN_OFFSET | offset given by external circuitry |
| BYTE | HW_CURSOR | external hardware cursor circuitry enabled |
| BYTE | HW_COLS | number of displayed columns during one character clock cycle |
| BYTE | HW_BLANK | set if the hardware blank feature is available |
| | | CRTC register |
| 20 BYTE | REGISTERS | register DUMP of the CRTC registers 0-19. |
| | | CRTC internal registers |
| BYTE | REGNO | The current index in the CRTC register file |
| BYTE | CHAR | The current cycle within the current rasterline |
| BYTE | CHARLINE | The current character line |
| BYTE | YCOUNTER | The current rasterline in the character |
| BYTE | CRSRCNT | Framecounter for the blinking cursor |
| BYTE | CRSRSTATE | if set the hardware cursor is visible |
| BYTE | CRSRLINES | set if ycounter is within the active cursor rasterlines for a char |
| WORD | CHARGEN_REL | relative base of currently used character generator in ROM (in byte) |
| WORD | SCREEN_REL | screen address to load the counter at the beginning of the next rasterline |
| WORD | VSYNC | number of rasterlines left within vsync; 0 = not in vsync |
| BYTE | VENABLE | vertical enable flipflop; 1= display, 0= blank. |
| | | (VICE-dependent?) variables |
| WORD | SCREEN_WIDTH | width of the current display window |
| WORD | SCREEN_HEIGHT | height of the current display window |
| WORD | SCREEN_XOFFSET | x position where the first character in a line starts in the window... |
| WORD | HJITTER | ...but only after adding this jitter |
| WORD | SCREEN_YOFFSET | x position where the first character in a line starts in the window... |
| WORD | FRAMELINES | expected number of rasterlines for the current frame |
| WORD | CURRENT_LINE | current rasterline as seen from the CRTC<br>This value has been added in module version V1.1 |
| BYTE | FLAG | Bit 0: If 1 then bit in VADDR_REVSWITCH must be set for reverse; if 0 then bit must be cleared for reverse. |

Here is the reference for the previous CRTC snapshot module. It is outdated and will not be read by this and later versions of VICE.

Version numbers: Major 0, Minor 0.

| Type | Name | Description |
|---|---|---|
| BYTE | RASTERY | The number of clock cycles from rasterlines start |
| WORD | RASTERLINE | The current rasterline |
| WORD | ADDRMASK | The address mask valid for the CRTC. All memory accesses are masked with this value |
| BYTE | HWFLAG | Bit 0: 1= hardware cursor available. Bit 1: 1= number of columns is doubled by external hardware |
| 20 BYTE | REGISTERS | register DUMP of the CRTC registers 0-19. |
| BYTE | CRSRSTATE | Hardware cursor: Bits 0-3: frame counter till next crsr line toggle. Bit 7: 1= cursor line active |

### 8.2.2.14 C64 memory module

The C64 memory module actually consists of two modules. The "C64MEM" module is mandatory and contains the RAM dump. The "C64ROM" module is optional and contains a dump of the ROM images.

The size of the C64 memory modules differs with each different memory configuration. The RAM configuration is saved in the snapshot, and restored when the snapshot is loaded. The attached cartridges are **not yet(!)** saved and not yet restored upon load.

Version numbers: Major 0, Minor 0

**The C64MEM module**

| Type | Name | Description |
|---|---|---|
| BYTE | CPUDATA | CPU port data byte |
| BYTE | CPUDIR | CPU port direction byte |
| BYTE | EXROM | state of the EXROM line (?) |
| BYTE | GAME | state of the GAME line (?) |
| ARRAY | RAM | 64k RAM dump |

**The C64ROM module**

| Type | Name | Description |
|---|---|---|
| ARRAY | KERNAL | 8k dump of the kernal ROM |
| ARRAY | BASIC | 8k dump of the basic ROM |
| ARRAY | CHARGEN | 4k dump of the chargen ROM |

### 8.2.2.15 C128 memory module

The C128 memory module actually consists of two modules. The "C128MEM" module is mandatory and contains the RAM dump. The "C128ROM" module is optional and contains a dump of the ROM images.

The size of the C128 memory modules differs with each different memory configuration. The RAM configuration is saved in the snapshot, and restored when the snapshot is loaded. The attached cartridges are also restored upon load if they have been saved in the snapshot.

Version numbers: Major 0, Minor 0

**The C128MEM module**

| Type | Name | Description |
|---|---|---|
| 12 BYTE | MMU | dump of the 12 MMU registers |
| ARRAY | RAM | 128k RAM dump banks 0 and 1 |

**The C128ROM module**

| Type | Name | Description |
|---|---|---|
| ARRAY | KERNAL | 8k dump of the kernal ROM |
| ARRAY | BASIC | 32k dump of the basic ROM |
| ARRAY | EDITOR | 4k dump of the editor ROM |
| ARRAY | 4k CHARGEN | dump of the chargen ROM |

### 8.2.2.16 VIC20 memory module

The VIC20 memory module actually consists of two modules. The "VIC20MEM" module is mandatory and contains the RAM dump. The "VIC20ROM" module is optional and contains a dump of the ROM images.

The size of the VIC20 memory modules differs with each different memory configuration. The RAM configuration is saved in the snapshot, and restored when the snapshot is loaded. The attached cartridges are also restored upon load if they have been saved in the snapshot.

**The VIC20MEM module**

Version numbers: Major 1, Minor 0

| Type | Name | Description |
|---|---|---|
| BYTE | CONFIG | Configuration register. Bits 0,1,2,3,5 reflect if the corresponding memory block is RAM (bit=1) or not (bit=0). |
| ARRAY | RAM0 | 1k RAM dump $0000-$03ff |
| ARRAY | RAM1 | 4k RAM dump $1000-$1fff |
| ARRAY | COLORRAM | 2k Color RAM, $9400-$9bff |
| ARRAY | BLK0 | if CONFIG & 1 then: 3k RAM dump $0400-$0fff |
| ARRAY | BLK1 | if CONFIG & 2 then: 8k RAM dump $2000-$3fff |
| ARRAY | BLK2 | if CONFIG & 4 then: 8k RAM dump $4000-$5fff |
| ARRAY | BLK3 | if CONFIG & 8 then: 8k RAM dump $6000-$7fff |
| ARRAY | BLK5 | if CONFIG & 32 then: 8k RAM dump $a000-$bfff |

**The VIC20ROM module**

Version numbers: Major 1, Minor 1

| Type | Name | Description |
|---|---|---|
| BYTE | CONFIG | Bit 0: 1= ROM block $2*** enabled. Bit 1: 1= ROM block $3*** enabled. Bit 2: 1= ROM block $4*** enabled. Bit 3: 1= ROM block $5*** enabled. Bit 4: 1= ROM block $6*** enabled. Bit 5: 1= ROM block $7*** enabled. Bit 6: 1= ROM block $A*** enabled. Bit 7: 1= ROM block $B*** enabled. |
| ARRAY | KERNAL | 8k KERNAL ROM image $e000-$ffff |
| ARRAY | BASIC | 16k BASIC ROM image $c000-$dfff |
| ARRAY | CHARGEN | 4k CHARGEN ROM image |
| ARRAY | BLK1A | 4k ROM image $2*** (if CONFIG & 1) |
| ARRAY | BLK1B | 4k ROM image $3*** (if CONFIG & 2) |
| ARRAY | BLK3A | 4k ROM image $6*** (if CONFIG & 16) |
| ARRAY | BLK3B | 4k ROM image $7*** (if CONFIG & 32) |
| ARRAY | BLK5A | 4k ROM image $A*** (if CONFIG & 64) |
| ARRAY | BLK5B | 4k ROM image $B*** (if CONFIG & 128) |
| ARRAY | BLK2A | 4k ROM image $4*** (if CONFIG & 4; added in V1.1) |
| ARRAY | BLK2B | 4k ROM image $5*** (if CONFIG & 8; added in V1.1) |

#### 8.2.2.17 PET memory module

The PET memory module actually consists of two modules. The "PETMEM" module is mandatory and contains the RAM dump. The "PETROM" module is optional and contains a dump of the ROM images.

The size of the PET memory modules differs with each different memory configuration. The RAM configuration is saved in the snapshot, and restored when the snapshot is loaded.

**The PETMEM module**

Version numbers: Major 1, Minor 2

| Type | Name | Description |
|---|---|---|
| BYTE | CONFIG | Configuration value. Bits 0-3: 0= 40 col PET without CRTC; 1= 40 col PET with CRTC; 2 = 80 col PET (with CRTC); 3= SuperPET; 4= 8096; 5= 8296. Bit 6: 1= RAM at $9***. Bit 7: 1= RAM at $A***. |
| BYTE | KEYBOARD | Keyboard type. 0= UK business; 1= Graphics; 2= German business |
| BYTE | MEMSIZE | memory size of low 32k in k (possible values 4, 8, 16, 32) |
| BYTE | CONF8X96 | Value of the 8x96 configuration register |

| | | |
|---|---|---|
| BYTE | SUPERPET | SuperPET config. Bit 0: 1= $9*** RAM enabled. Bit 1: 1= RAM write protected. Bit 2: 1= CTRL register write protected. Bit 3: 0= DIAG pin active. Bits 4-7: RAM block in use. |
| ARRAY | RAM | 4-32k RAM (not 8296, size depends on MEMSIZE) |
| ARRAY | VRAM | 2/4k RAM (not 8296, size depends on CONFIG) |
| ARRAY | EXTRAM | 64k expansion RAM (SuperPET and 8096 only) |
| ARRAY | RAM | 128k RAM (8296 only) |
| – | – | The following item has been added in V1.1 |
| BYTE | POSITIONAL | bit 0=0 = symbolic keyboard mapping, bit 0=1 = positional mapping. |
| – | – | The following item has been added in V1.1 |
| BYTE | EOIBLANK | bit 0=0 = EOI does not blank screen, bit 0=1 = EOI blanks screen. |

The last item has been added in PETMEM snapshot version 1.1. It is ignored by earlier restore routines (V1.0) and the V1.1 restore routines do not change the current setting when reading a V1.0 snapshot.

In V1.2 the new EOIBLANK variable has been added. This implements the "blank screen on EOI" feature that was previously linked to a wrong resource.

**The PETROM module**

Version numbers: Major 1, Minor 0

| Type | Name | Description |
|---|---|---|
| BYTE | CONFIG | Bit 0: 1= $9*** ROM included. Bit 1: 1= $A*** ROM included. Bit 2: 1= $B*** ROM included. Bit 3: 1= $e900-$efff ROM included |
| ARRAY | KERNAL | 4k KERNAL ROM image $f000-$ffff |
| ARRAY | EDITOR | 2k EDITOR ROM image $e000-$e7ff |
| ARRAY | CHARGEN | 2k CHARGEN ROM image |
| ARRAY | ROM9 | 4k $9*** ROM image (if CONFIG & 1) |
| ARRAY | ROMA | 4k $A*** ROM image (if CONFIG & 2) |
| ARRAY | ROMB | 4k $B*** ROM image (if CONFIG & 4) |
| ARRAY | ROMC | 4k $C*** ROM image |
| ARRAY | ROMD | 4k $D*** ROM image |
| ARRAY | ROME9 | 7 blocks $e900-$efff ROM image (if CONFIG & 8) |

### 8.2.2.18 CBM-II memory module

The CBM-II memory module actually consists of two modules. The "CBM2MEM" module is mandatory and contains the RAM dump. The "CBM2ROM" module is optional and contains a dump of the ROM images.

The size of the CBM-II memory modules differs with each different memory configuration. The RAM configuration is saved in the snapshot, and restored when the snapshot is loaded.

Version numbers: Major 1, Minor 0

**The CBM2MEM module**

| Type | Name | Description |
|---|---|---|
| UBYTE | MEMSIZE | Memory size in 128k blocks (1=128k, 2=256k, 4=512k, 8=1024k) |
| UBYTE | CONFIG | Bit 0 = $f0800-$f0fff RAM, Bit 1 = $f1000-$f1fff RAM, Bit 2 = $f2000-$f3fff RAM, Bit 3 = $f4000-$f5fff RAM, Bit 4 = $f6000-$f7fff RAM, Bit 5 = $fc000-$fcfff RAM, Bit 6 = is a C500 |
| UBYTE | HWCONFIG | Bit 0/1: model line configuration |
| UBYTE | EXECBANK | CPUs execution bank register |
| UBYTE | INDBANK | CPUs indirection bank register |
| ARRAY | SYSRAM | 2k system RAM $f0000-$f07ff |
| ARRAY | VIDEO | 2k video RAM $fd000-$fd7ff |
| ARRAY | RAM | RAM dump, size according to MEMSIZE |
| ARRAY | RAM08 | if memsize < 1M and CONFIG & 1 : 2k RAM $f0800-$f0fff |
| ARRAY | RAM1 | if memsize < 1M and CONFIG & 2 : 4k RAM $f1000-$f1fff |
| ARRAY | RAM2 | if memsize < 1M and CONFIG & 4 : 8k RAM $f2000-$f3fff |
| ARRAY | RAM4 | if memsize < 1M and CONFIG & 8 : 8k RAM $f4000-$f5fff |
| ARRAY | RAM6 | if memsize < 1M and CONFIG & 16 : 8k RAM $f6000-$f7fff |
| ARRAY | RAMC | if memsize < 1M and CONFIG & 32 : 4k RAM $fc000-$fcfff |

The RAM* arrays are only saved if the RAM itself is less than 1M. If the memory size is 1M then those areas are taken from the bank 15 area of the normal RAM.

The memory array starts at $10000 if the memory size is less than 512k, or at $00000 if 512k or more. In case of a C510, then the memory array also always starts at $00000.

**The CBM2ROM module**

| Type | Name | Description |
|---|---|---|
| UBYTE | CONFIG | Bit 1: 1= $1*** ROM image included. Bit 2: 1= $2000-$3fff ROM image included. Bit 3: 1= $4000-$5fff ROM image included. Bit 4: 1= $6000-$7fff ROM image included. Bit 5: 1= chargen ROM is VIC-II chargen, 0= CRTC chargen. |
| ARRAY | KERNAL | 8 KERNAL ROM image ($e000-$efff) |
| ARRAY | BASIC | BASIC ROM image ($8000-$bfff) |

| | | |
|---|---|---|
| ARRAY | CHARGEN | 4k CHARGEN ROM image |
| ARRAY | ROM1 | 4k cartridge ROM image for $1*** (if CONFIG & 2) |
| ARRAY | ROM2 | 8k cartridge ROM image for $2000-$3fff (if CONFIG & 4) |
| ARRAY | ROM4 | 8k cartridge ROM image for $4000-$5fff (if CONFIG & 8) |
| ARRAY | ROM6 | 8k cartridge ROM image for $6000-$7fff (if CONFIG & 16) |

#### 8.2.2.19 C500 data module

The C500 data module contains simple state information not already saved in the other modules.

Version numbers: Major 0, Minor 0

**The C500DATA module**

| Type | Name | Description |
|---|---|---|
| DWORD | IRQCLK | CPU clock ticks till next 50Hz IRQ |

# 9 Monitor

Every VICE emulator has a complete built-in monitor, which can be used to examine, disassemble and assemble machine language programs, as well as debug them through breakpoints. It can be activated by using the "Activate monitor" command (left button menu). Notice that you have to run the emulator from a terminal emulation program (such as `rxvt` or `xterm`) in order to use the monitor.

**Warning**: this version of the monitor is still under construction, and some of the features are not fully working yet.

## 9.1 Terminology

'`address_space`'
: This refers to the range of memory locations and a set of registers. This can be the addresses available to the computer's processor, the disk drive's processor or a specific memory configuration of one of the mentioned processors.

'`bankname`'
: The CPU can only see 64k of memory at any one time, due to its 16 bit address bus. The C64 and other computers have more than this amount, and this is handled by banking: a memory address can have different contents, depending on the active memory bank. A bankname names a specific bank in the current address_space.

'`register`'
: One of the following: program counter (PC), stack pointer (SP), accumulator (A), X register (X), or Y register (Y).

'`address`' A specific memory location in the range $0000 to $FFFF.

'address_range'
: Two addresses. If the second address is less than the first, the range is assumed to wraparound from $FFFF to $0000. Both addresses must be in the same address space.

'address_opt_range'
: An address or an address range.

'prompt'
: The prompt has the format [x:y]. If x is -, memory reads from the monitor do not have side effects. Otherwise, x is S. The second part of the prompt, y, shows the default address space.

'checkpoint'
: The monitor has the ability to setup triggers that perform an action when a specified situation occurs. There are three types of checkpoints; breakpoints, tracepoints and watchpoints.

'breakpoint'
: A breakpoint is triggered based on the program counter. When it is triggered, the monitor is entered.

'tracepoint'
: Like breakpoints, a tracepoint is triggered based on the program counter. Instead of entering the monitor, the program counter is printed and execution continues.

'watchpoint'
: Watchpoints are triggered by a read and/or write to an address. When a watchpoint is triggered, the monitor is entered.

'memmap'
: The memmap keeps track of RAM/ROM/IO read/write/execute accesses. The feature must be enabled with "–enable-memmap" configure option, as it might decrease performance notably on slower hardware. The option also enables CPU history.

'<...>'
: A data type.

'*'
: Zero or more occurrences.

'[...]'
: An optional argument.

## 9.2 Machine state commands

backtrace
bt
: Print JSR call chain (most recent call first). Stack offset relative to SP+1 is printed in parentheses. This is a best guess only.

cpuhistory [<count>]
chis [<count>]
: Show <count> last executed commands. (disabled by default; configure with –enable-memmap to enable)

`dump "<filename>"`
: Write a snapshot of the machine into the file specified. This snapshot is compatible with a snapshot written out by the UI. Note: No ROM images are included into the dump.

`goto <address>`
`g <address>`
: Change the PC to address and continue execution.

`io [<address>]`
: Display i/o registers. Invoking without an address shows a dump of the entire io range, if an address is given then details for the chip at the respective (base-)address are displayed (if available).

`next [<count>]`
`n [<count>]`
: Advance to the next instruction. Subroutines are treated as a single instruction.

`registers [<reg_name> = <number> [, <reg_name> = <number>]*]`
`r [<reg_name> = <number> [, <reg_name> = <number>]*]`
: Assign respective registers. With no parameters, display register values.

`reset [<type>]`
: Reset the machine or drive. `type`: 0 = soft, 1 = hard, 8-11 = drive.

`return`
`ret`
: Continues execution and returns to the monitor just after the next RTS or RTI is executed.

`step [<count>]`
`z [<count>]`
: Single step through instructions. An optional count allows stepping more than a single instruction at a time.

`undump "<filename>"`
: Read a snapshot of the machine from the file specified.

## 9.3 Memory commands

`bank [<bankname>]`
: Without a bankname, display all available banks for the current address_space. With a bankname given, switch to the specified bank. If a bank is not completely filled (ROM banks for example) normally the `ram` bank is used where the bank has holes. The `cpu` bank uses the bank currently used by the CPU.

`compare <address_range> <address>`
`c <address_range> <address>`
: Compare memory from the source specified by the address range to the destination specified by the address. The regions may overlap. Any values that miscompare are displayed using the default displaytype.

`device [c:|8:|9:]`
: Set the default address space to either the computer ‘c:’ or the specified drive ‘8:’ or ‘9:’

`fill <address_range> <data_list>`
`f <address_range> <data_list>`
: Fill memory in the specified address range with the data in <data_list>. If the size of the address range is greater than the size of the data_list, the data_list is repeated.

`hunt <address_range> <data_list>`
`h <address_range> <data_list>`
: Hunt memory in the specified address range for the data in <data_list>. If the data is found, the starting address of the match is displayed. The entire range is searched for all possible matches. The data list may have 'xx' as a wildcard.

`i <address_opt_range>`
: Display memory contents as PETSCII text.

`ii <address_opt_range>`
: Display memory contents as screen code text

`mem [<data_type>] [<address_opt_range>]`
`m [<data_type>] [<address_opt_range>]`
: Display the contents of memory. If no datatype is given, the default is used. If only one address is specified, the length of data displayed is based on the datatype. If no addresses are given, the 'dot' address is used.

`memmapshow [<mask>] [<address_opt_range>]`
`mmsh [<mask>] [<address_opt_range>]`
: Show the memmap. The mask can be specified to show only those locations with accesses of certain type(s). The mask is a number with the bits "ioRWXrwx", where RWX are for ROM and rwx for RAM. Optionally, an address range can be specified. (disabled by default; configure with –enable-memmap to enable)

`memmapzap`
`mmzap`
: Clear the memmap. (disabled by default; configure with –enable-memmap to enable)

`memmapsave "<filename>" <format>`
`mmsave "<filename>" <format>`
: Save the memmap as a picture. `format`: 0 = BMP, 1 = PCX, 2 = PNG, 3 = GIF, 4 = IFF. (disabled by default; configure with –enable-memmap to enable)

`memchar [<data_type>] [<address_opt_range>]`
`mc [<data_type>] [<address_opt_range>]`
: Display the contents of memory as character data. If only one address is specified, only one character is displayed. If no addresses are given, the "dot" address is used.

`memsprite [<data_type>] [<address_opt_range>]`
`ms [<data_type>] [<address_opt_range>]`
: Display the contents of memory as sprite data. If only one address is specified, only one sprite is displayed. If no addresses are given, the "dot" address is used.

`move <address_range> <address>`
`t <address_range> <address>`
: Move memory from the source specified by the address range to the destination specified by the address. The regions may overlap.

`screen`
`sc`
: Displays the contents of the screen.

`sidefx [on|off|toggle]`
`sfx [on|off|toggle]`
: Control how monitor generated reads affect memory locations that have read side-effects, like CIA interrupt registers for example. If the argument is 'on' then reads may cause side-effects. If the argument is 'off' then reads don't cause side-effects. If the argument is 'toggle' then the current mode is switched. No argument displays the current state.

`> [<address>] <data_list>`
: Write the specified data at `address`.

## 9.4 Assembly commands

`a <address> [ <instruction> [: <instruction>]* ]`
: Assemble instructions to the specified address. If only one instruction is specified, enter assembly mode (enter an empty line to exit assembly mode).

`disass [<address> [<address>]]`
`d [<address> [<address>]]`
: Disassemble instructions. If two addresses are specified, they are used as a start and end address. If only one is specified, it is treated as the start address and a default number of instructions are disassembled. If no addresses are specified, a default number of instructions are disassembled from the dot address.

## 9.5 Checkpoint commands

`break [<address> [if <cond_expr>] ]`
: This command allows setting a breakpoint or listing the current breakpoints. If no address is given, the currently valid checkpoints are printed. If an address is given, a breakpoint is set for that address and the breakpoint number is printed. A conditional expression can also be specified for the breakpoint. For more information on conditions, see the CONDITION command.

`enable <checknum>`
`disable <checknum>`
: Each checkpoint can be enabled or disabled. This command allows changing between these states.

`command <checknum> "<command>"`
: When checkpoint `checknum` is hit, the specified command is executed by the monitor. Note that the `x` command is not yet supported as a command argument.

`condition <checknum> if <cond_expr>`
`cond <checknum> if <cond_expr>`

Each time the specified checkpoint is examined, the condition is evaluated. If it evalutes to true, the checkpoint is activated. Otherwise, it is ignores. If registers are specified in the expression, the values used are those at the time the checkpoint is examined, not when the condition is set.

Currently, the cond_expr is very limited. You can use registers (.A, .X, .Y, .PC, and .SP) and compare against other registers or absolute values. For example, the following are all valid conditions: .A == 0, .X == .Y, 8:.X == .X, .A != 5, .A < .X.

However, you cannot specify memory contents and compare that.

`delete <checknum>`
`del <checknum>`

Delete the specified checkpoint.

`ignore <checknum> [<count>]`

Ignore a checkpoint after a given number of crossings. If no count is given, the default value is 1.

`trace [address [address]]`
`tr [address [address]]`

This command is similar to the `break` command except that it operates on tracepoints. A tracepoint differs from a breakpoint by not stopping execution but simply printing the PC, giving the user an execution trace. The second optional address can be used to specify the end of an range of addresses to be traced.

`until [<address>]`
`un [<address>]`

If no address is given, the currently valid breakpoints are printed. If an address is given, a temporary breakpoint is set for that address and the breakpoint number is printed. Control is returned to the emulator by this command. The breakpoint is deleted once it is hit.

`watch [loadstore] [address [address]]`
`w [loadstore] [address [address]]`

This command is similar to the previous two commands except that it operates on watchpoints. A watchpoint differs from the others by stopping on a read and/or write to an address or range of addresses. If no addresses are given, a list of all the watchpoints is printed. The loadstore parameter can be either "load" or "store" to determine on which operation the monitor breaks. If not specified, the monitor breaks on both operations.

## 9.6 General commands

`cd <directory>`

Change the working directory.

`device [c:|d:]`
`dev [c:|d:]`
Set the default memory device to either the computer (`c:`) or the disk (`d:`).

`dir [<directory>]`
`ls [<directory>]`
Display the directory contents.

`pwd` Show current working directory.

`radix [H|D|O|B]`
`rad [H|D|O|B]`
Set the default radix to hex, decimal, octal, or binary. With no argument, the current radix is printed.

## 9.7 Disk commands

`attach <filename> <device>`
Attach file to device. (device 32 = cart)

`block_read <track> <sector> [<address>]`
`br <track> <sector> [<address>]`
Read the block at the specified track and sector. If an address is specified, the data is loaded into memory. If no address is given, the data is displayed using the default datatype.

`block_write <track> <sector> <address>`
`bw <track> <sector> <address>`
Write a block of data at `address` to the specified track and sector of disk in drive 8.

`detach <device>`
Detach file from device. (device 32 = cart)

`@<disk command>`
Perform a disk command on the currently attached disk image on drive 8. The specified disk command is sent to the drive's channel #15.

`load "<filename>" <device> [<address>]`
`l "<filename>" <device> [<address>]`
Load the specified file into memory. If no address is given, the file is loaded to the address specified by the first two bytes read from the file. If address is given, the file is loaded to the specified address and the first two bytes read from the file are skipped. If device is 0, the file is read from the file system.

`list [<directory>]`
List disk contents.

`bload "<filename>" <device> <address>`
`bl "<filename>" <device> <address>`
Load the specified file into memory at the specified address. If device is 0, the file is read from the file system.

`save "<filename>" <device> <address1> <address2>`
`s "<filename>" <device> <address1> <address2>`
Save the memory from address1 to address2 to the specified file. Write two-byte load address. If device is 0, the file is written to the file system.

`bsave "<filename>" <device> <address1> <address2>`
`bs "<filename>" <device> <address1> <address2>`
Save the memory from address1 to address2 to the specified file. If device is 0, the file is written to the file system.

## 9.8 Command file commands

`playback "<filename>"`
`pb "<filename>"`
Monitor commands from the specified file are read and executed. This command stops at the end of file or when a STOP command is read.

`record "<filename>"`
`rec "<filename>"`
After this command, all commands entered are written to the specified file until the STOP command is entered.

`stop` Stop recording commands. See `record`.

## 9.9 Label commands

`add_label <address> <label>`
`al <address> <label>`
Map a given address to a label. This label can be used when entering assembly code and is shown during disassembly. Additionally, it can be used whenever an address must be specified.

<label> is the name of the label; it must start with a dot (".") in order for the monitor to recognize it as a label.

`delete_label [<memspace>] <label>`
`dl [<memspace>] <label>`
Remove the specified label from the label tables. If no memory space is checked, all tables are checked.

`load_labels [<memspace>] "<filename>"`
`ll [<memspace>] "<filename>"`
Load a file containing a mapping of labels to addresses. If no memory space is specified, the default readspace is used.

The file must contain commands the monitor understands, e.g. add_label. The compiler cc65 can create such label files.

Vice can also load label files created by the Acme assembler. Their syntax is e.g. "labelname = $1234 ; Maybe a comment". A dot will be added automatically to label names assigned in this way to fit to the Vice label syntax. Normally the semicolon seperates commands but after an assignment of this kind it may

be used to start a comment to end of line, so unchanged Acme label files can be fed into Vice.

`save_labels [<memspace>] "<filename>"`
`sl [<memspace>] "<filename>"`
Save labels to a file. If no memory space is specified, all of the labels are saved.

`show_labels [<memspace>]`
`shl [<memspace>]`
Display current label mappings. If no memory space is specified, show all labels.

## 9.10 Miscellaneous commands

`cartfreeze`
Use cartridge freeze.

`cpu <type>`
Specify the type of CPU currently used (6502/z80).

`exit`
`x` Leave the monitor and return to execution.

`export`
`exp` Print out list of attached expansion port devices.

`help [<command>]`
If no argument is given, prints out a list of all available commands. If an argument is given, prints out specific help for that command.

`keybuf "<string>"`
Put the specified string into the keyboard buffer.

`print <expression>`
`p <expression>`
Evaluate the specified expression and output the result.

`resourceget "<resource>"`
`resget "<resource>"`
Displays the value of the `resource`.

`resourceset "<resource>" "<value>"`
`resset "<resource>" "<value>"`
Sets the value of the `resource`.

`screenshot "<filename>" [<format>]`
`scrsh "<filename>" [<format>]`
Take a screenshot. `format`: default = BMP, 1 = PCX, 2 = PNG, 3 = GIF, 4 = IFF.

`tapectrl <command>`
Control the datasette. `command`: 0 = stop, 1 = start, 2 = forward, 3 = rewind, 4 = record, 5 = reset, 6 = reset counter.

`quit` Exit the emulator immediately.

`~ <number>`
Display the specified number in decimal, hex, octal and binary.

# 10 c1541

VICE is provided with a complete stand-alone disk image maintenance utility, called `c1541`.

You can either invoke it from the command line or from within one of the VICE emulators, using the "Run c1541" command which will open a new `xterm` window with a running `c1541` in it.

The syntax is:

```
c1541 [IMAGE1 [IMAGE2]] [COMMAND1 COMMAND2 ... COMMANDN]
```

`IMAGE1` and `IMAGE2` are disk image names that can be attached before `c1541` starts. `c1541` can handle up to two disk images at the same time by using two virtual built-in drives, numbered `8` and `9`; `IMAGE1` (if present) is always attached to drive `8`, while `IMAGE2` is attached to drive `9`.

`COMMAND`s specified on the command-line all begin with the minus sign (`-`); if present, `c1541` executes them in the same order as they are on the command line and returns a zero error code if they were successful. If any of the `COMMAND`s fails, `c1541` stops and returns a nonzero error code.

If no `COMMAND`s are specified at all, `c1541` enters interactive mode, where you can type commands manually. Commands in interactive mode are the same as commands in batch mode, but do not require a leading `-`. As with the monitor, file name completion and command line editing with history are provided via GNU `readline`. Use the command '`quit`' or press *`C-d`* to exit.

## 10.1 Specifying files in c1541

When accessing CBM DOS files (i.e. files that reside on disk images), c1541 uses a special syntax that lets you access files on both drive 8 and 9. If you prepend the file name with `@8:` or `@9:`, you will specified that file is to be found or created on drive 8 and 9, respectively.

For instance,

```
@8:somefile
```

will name file named `somefile` on unit 8, while

```
@9:somefile
```

will name file named `somefile` on unit 9.

## 10.2 Using quotes and backslashes

You can use quotes (`"`) in a command to embed spaces into file names. For instance,

```
read some file
```

will read file `some` from the disk image and write it into the file system as `file`, while

```
read "some file"
```

will copy `some file` into the file system, with the name `some file`.

The backslash character (\) has a special meaning too: it lets you literally insert the following character no matter what it is. For example,

```
read some\ file
```

will copy file `some file` into the file system, while

```
read some\ file this\"file
```

will copy `some file` into the file system with name `this"file` (with an embedded quote).

## 10.3 c1541 commands and options

This is a list of the `c1541` commands. They are shown in their interactive form, without the leading `-`. Square brackets [] indicate an optional part, and `"<COMMAND>"` translates to a disk command according to CBM DOS, like `"i0"` for example.

`[<command>]`
: Execute specified CBM DOS command and print the current status of the drive. If no `command` is specified, just print the status.

`? [<command>]`
: Explain specified command. If no command is specified, list available ones.

`attach <diskimage> [<unit>]`
: Attach `diskimage` to `unit` (default unit is 8).

`block <track> <sector> <disp> [<drive>]`
: Show specified disk block in hex form.

`copy <source1> [<source2> ... <sourceN>] <destination>`
: Copy `source1` ... `sourceN` into destination. If N > 1, `destination` must be a simple drive specifier (`@n:`).

`delete <file1> [<file2> ... <fileN>]`
: Delete the specified files.

`exit`
: Exit (same as `quit`).

`extract`
: Extract all the files to the file system.

`format <diskname,id> [<type> <imagename>] [<unit>]`
: If `unit` is specified, format the disk in unit `unit`. If `type` and `imagename` are specified, create a new image named `imagename`, attach it to unit 8 and format it. `type` is a disk image type, and must be either `x64`, `d64` (both VC1541/2031), `g64` (VC1541/2031 but in GCR coding), `d71` (VC1571), `d81` (VC1581), `d80` (CBM8050) or `d82` (CBM8250/1001). Otherwise, format the disk in the current unit, if any.

`gcrformat <diskname,id> <imagename>`
: Create and format a G64 disk image named `imagename`.

`help [<command>]`
: Explain specified command. If no command is specified, list available ones.

`info [<unit>]`
: Display information about unit `unit` (if unspecified, use the current one).

`list [<pattern>]`
: List files matching `pattern` (default is all files).

`quit`
: Exit (same as `exit`).

`read <source> [<destination>]`
: Read `source` from the disk image and copy it into `destination` in the file system. If `destination` is not specified, copy it into a file with the same name as `source.`",

`rename <oldname> <newname>`
: Rename `oldname` into `newname`. The files must be on the same drive.

`tape <t64name> [<file1> ... <fileN>]`
: Extract files from a T64 image.

`unit <number>`
: Make unit `number` the current unit.

`unlynx <lynxname> [<unit>]`
: Extract the specified Lynx image file into the specified unit (default is the current unit).

`validate [<unit>]`
: Validate the disk in unit `unit`. If `unit` is not specified, validate the disk in the current unit.

`write <source> [<destination>]`
: Write `source` from the file system into `destination` on a disk image.

`zcreate <x64name> <zipname> [<label,id>]`
: Create an X64 disk image out of a set of four Zipcoded files named `1!zipname`, `2!zipname`, `3!zipname` and `4!zipname`.

## 10.4 Executing shell commands

If you want to execute a shell command from withing `c1541`, just prepend it with an exclamation mark (`!`). For example,

```
!ls -la
```

will execute the command `ls -la`, which will show you all the files in the current directory.

# 11 cartconv

The cartconv program is a cartridge conversion utility, it can convert between binary and .crt images and it can 'insert' binary and/or .crt images into the EPROM type of cartridges.

## 11.1 cartconv command line options

The cartconv program has the following parameters:

`-i "input name"`
: This parameter is mandatory, it should contain the name of the binary/.crt file you want to convert. For the EPROM type of cartridges this parameter can be used multiple times to insert images into the resulting file.

`-o "output name"`
: This parameter is mandatory, it should contain the name of the binary/.crt file you want to convert the input file to.

`-t carttype`
: This parameter is optional. It is only needed when converting to a .crt file. See below for the supported cartridge types.

`-n "cart name"`
: This parameter is optional and is used as the cartridge name when creating a .crt file.

`-l loadaddress`
: This parameter is optional and is used as the load-address when converting a .crt file to a .prg file, or when converting to a generic type .crt file.

`-f "input name"`
: This parameter is optional, and is meant to output information about the named file. It can't be used in conjuction with any of the other parameters.

`-r` This parameter is optional, it enables repair mode (accept broken input files)

The following cartridge types are supported:

| | |
|---|---|
| `bin` | Binary .bin file (Default crt->bin) |
| `normal` | Generic 8kB/12kB/16kB .crt file (Default bin->crt) |
| `prg` | Binary C64 .prg file with load-address |
| `ulti` | Ultimax mode 4kB/8kB/16kB .crt file |
| `ap` | Atomic Power .crt file |
| `ar2` | Action Replay MK2 .crt file |
| `ar3` | Action Replay MK3 .crt file |
| `ar4` | Action Replay MK4 .crt file |
| `ar5` | Action Replay V5 .crt file |
| `cap` | Capture .crt file |
| `comal` | Comal 80 .crt file |
| `dep256` | Dela EP256 .crt file, extra files can be inserted (1)(2) |
| `dep64` | Dela EP64 .crt file, extra files can be inserted (1) |
| `dep7x8` | Dela EP7x8 .crt file, extra files can be inserted (1)(2)(3) |
| `din` | Dinamic .crt file |
| `dsm` | Diashow-Maker .crt file |
| `easy` | EasyFlash .crt file |
| `epyx` | Epyx FastLoad .crt file |
| `exos` | EXOS .crt file |

| | |
|---|---|
| `expert` | Expert Cartridge .crt file |
| `fc1` | The Final Cartridge .crt file |
| `fc3` | The Final Cartridge III .crt file |
| `fcp` | Final Cartridge Plus .crt file |
| `ff` | Freeze Frame .crt file |
| `fm` | Freeze Machine .crt file |
| `fp` | Fun Play .crt file |
| `gk` | Game Killer .crt file |
| `gs` | C64 Games System .crt file |
| `ide64` | IDE64 .crt file |
| `ieee` | IEEE-488 Interface .crt file |
| `kcs` | KCS Power Cartridge .crt file |
| `mach5` | MACH 5 .crt file |
| `md` | Magic Desk .crt file |
| `mf` | Magic Formel .crt file |
| `mikro` | Mikro Assembler .crt file |
| `mmc64` | MMC64 .crt file |
| `mmcr` | MMC Replay .crt file |
| `mv` | Magic Voice .crt file |
| `ocean` | Ocean .crt file |
| `p64` | Prophet64 .crt file |
| `rep256` | REX 256k EPROM Cart .crt file, extra files can be inserted (1)(2)(3) |
| `ross` | ROSS .crt file |
| `rr` | Retro Replay .crt file |
| `ru` | REX Utility .crt file |
| `s64` | Snapshot 64 .crt file |
| `sb` | Structured BASIC .crt file |
| `se5` | Super Explode V5.0 .crt file |
| `sg` | Super Games .crt file |
| `simon` | Simons' BASIC .crt file |
| `ss4` | Super Snapshot V4 .crt file |
| `ss5` | Super Snapshot V5 .crt file |
| `star` | Stardos .crt file |

`wl` Westermann Learning .crt file

`ws` Warp Speed .crt file

`zaxxon` Zaxxon .crt file

- (1) insertion of 32kB EPROM files supported.
- (2) insertion of 8kB .crt/binary files supported.
- (3) insertion of 16kB .crt/binary files supported.

## 11.2 cartconv examples

`cartconv -i foo.crt -o foo.bin`
Convert a .crt file to a binary file with no load-address.

`cartconv -t prg -i foo.crt -o foo.prg`
Convert a .crt file to a .prg file with default load-address.

`cartconv -t prg -l 49152 -i foo.crt -o foo.prg`
Convert a .crt file to a .prg file with 49152 as the load-address.

`cartconv -t ocean -i foo.bin -o foo.crt`
Convert a binary file to an ocean type cartridge.

`cartconv -t dep64 -i dep64.bin -i eprom.prg -o foo.crt`
Inserting a 32kB EPROM file into an dep64 type cartridge.

- step 1 : use the dep64 binary file in VICE as a generic 8kB cartridge.
- step 2 : generate an EPROM file.
- step 3 : get the EPROM file to the host computer.
- step 4 : insert the EPROM file into the final dep64 .crt file:

`cartconv -t dep256 -i dep256.bin -i somegame.crt -o foo.crt`
Insert an 8kB .crt file into a dep256 type cartridge.

`cartconv -t rep256 -i rep256.bin -i foo1.crt -i foo2.crt -i foo3.crt -o foo.crt`
Insert multiple 8kB .crt files into a rep256 type cartridge.

`cartconv -f foo.crt`
Get information about a .crt file.

# 12 petcat

The petcat program is a text conversion utility, it can convert between ASCII, PETSCII and tokenized BASIC.

## 12.1 petcat command line options

`-help` Output help text

`-v` Same as above

`-c` controls (interpret also control codes) (default if textmode)

`-nc` no controls (suppress control codes in printout) (default if non-textmode)

`-ic` interpret control codes case-insensitive

`-h` write header (default if output is stdout)

`-nh` no header (default if output is a file)

`-skip <n>` Skip <n> bytes in the beginning of input file. Ignored on P00.

`-text` Force text mode

`-<version>`
use keywords for <version> instead of the v7.0 ones

`-w<version>`
tokenize using keywords on specified Basic version.

`-k<version>`
list all keywords for the specified Basic version

`-k` list all Basic versions available.

`-l` Specify load address for program (in hex, no loading chars!).

`-o <name>` Specify the output file name

`-f` Force overwritten the output file. The default depends on the BASIC version.

BASIC Versions:

`1` PET Basic V1.0

`2` Basic v2.0

`superexp` Basic v2.0 with Super Expander (VIC20)

`turtle` Basic v2.0 with Turtle Basic by Craig Bruce (VIC20)

`mighty` Basic v2.0 with Mighty Basic by Craig Bruce (VIC20)

`a` Basic v2.0 with AtBasic (C64)

`simon` Basic v2.0 with Simon's Basic extension (C64)

`speech` Basic v2.0 with Speech Basic v2.7 (C64)

`F` Basic v2.0 with Final Cartridge III (C64)

`ultra` Basic v2.0 with Ultrabasic-64 (C64)

`graph` Basic v2.0 with Graphics basic (C64)

`WSB` Basic v2.0 with WS basic (C64)

`WSBF` Basic v2.0 with WS basic final (C64)

`Pegasus` Basic v2.0 with Pegasus basic 4.0 (C64)

`Xbasic` Basic v2.0 with Xbasic (C64)

`Drago` Basic v2.0 with Drago basic 2.2 (C64)

`REU` Basic v2.0 with REU-basic (C64)

`Lightning`
: Basic v2.0 with Basic Lightning (C64)

`magic` Basic v2.0 with Magic Basic (C64)

`easy` Basic v2.0 with Easy Basic (VIC20)

`blarg` Basic v2.0 with Blarg (C64)

`Game` Basic v2.0 with Game Basic (C64)

`BSX` Basic v2.0 with Basex (C64)

`superbas` Basic v2.0 with Super Basic (C64)

`exp20` Basic 2.0 with Expanded Basic (VIC20)

`exp64` Basic 2.0 with Expanded Basic (C64)

`sxc` Basic 2.0 with Super Expander Chip (C64)

`warsaw` Basic 2.0 with Warsaw Basic (C64)

`4v` Basic 2.0 with Basic 4.0 extensions (VIC20)

`4 -w4e` PET Basic v4.0 program (PET/C64)

`5` Basic 2.0 with Basic 5.0 extensions (VIC20)

`3` Basic v3.5 program (C16)

`70` Basic v7.0 program (C128)

`71` Basic v7.1 program (C128)

`10` Basic v10.0 program (C64DX)

## 12.2 petcat examples

`petcat -2 -o outputfile.txt -- inputfile.prg`
: Convert inputfile.prg to a text file in outputfile.txt, using BASIC V2 only

`petcat -wsimon -o outputfile.prg -- inputfile.txt`
: Convert inputfile.txt to a PRG file in outputfile.prg, using Simon's BASIC

# 13 The emulator file formats

This chapter gives a technical description of the various files supported by the emulators.

## 13.1 The T64 tape image format

(This section was taken from the C64S distribution.)

The `T64` File Structure was developed by Miha Peternel for use in the C64S emulator. It is easy to use and allows future extensions.

### 13.1.1 T64 File structure

| Offset | Size | Description |
|---|---|---|
| 0 | 64 | tape record |
| 64 | 32*n | file records for n directory entries |
| 64+32*n | varies | binary contents of the files |

### 13.1.2 Tape Record

| Offset | Size | Description |
|---|---|---|
| 0 | 32 | DOS tape description + EOF (for type) |
| 32 | 2 | tape version ($0200) |
| 34 | 2 | number of directory entries |
| 36 | 2 | number of used entries (can be 0 in my loader) |
| 38 | 2 | free |
| 40 | 24 | user description as displayed in tape menu |

### 13.1.3 File record

| Offset | Size | Description |
|---|---|---|
| 0 | 1 | entry type (see below) |
| 1 | 1 | C64 file type |
| 2 | 2 | start address |
| 4 | 2 | end address |
| 6 | 2 | free |
| 8 | 4 | offset of file contents start within T64 file |
| 12 | 4 | free |
| 16 | 16 | C64 file name |

Valid entry types are:

| Code | Explanation |
|---|---|
| 0 | free entry |
| 1 | normal tape file |
| 2 | tape file with header: header is saved just before file data |
| 3 | memory snapshot v0.9, uncompressed |
| 4 | tape block |
| 5 | digitized stream |
| 6 ... 255 | reserved |

Notes:

- VICE only supports file type 1.
- Types 3, 4 and 5 are subject to change (and are rarely used).

## 13.2 The G64 GCR-encoded disk image format

(This section was contributed by Peter Schepers and slightly edited by Ettore Perazzoli.)

This format was defined in 1998 as a cooperative effort between several emulator people, mainly Per Hkan Sundell, author of the CCS64 C64 emulator, Andreas Boose of the VICE CBM emulator team and Joe Forster/STA, the author of Star Commander. It was the first real public attempt to create a format for the emulator community which removed almost all of the drawbacks of the other existing image formats, namely `D64`.

The intention behind `G64` is not to replace the widely used `D64` format, as `D64` works fine with the vast majority of disks in existence. It is intended for those small percentage of programs which demand to work with the 1541 drive in a non-standard way, such as reading or writing data in a custom format. The best example is with speeder software such as Action Cartridge in Warp Save mode or Vorpal which write track/sector data in another format other than standard GCR. The other obvious example is copy-protected software which looks for some specific data on a track, like the disk ID, which is not stored in a standard `D64` image.

`G64` has a deceptively simply layout for what it is capable of doing. We have a signature, version byte, some predefined size values, and a series of offsets to the track data and speed zones. It is what's contained in the track data areas and speed zones which is really at the heart of this format.

Each track entry in simply the raw stream of GCR data, just what a read head would see when a diskette is rotating past it. How the data gets interpreted is up to the program trying to access the disk. Because the data is stored in such a low-level manner, just about anything can be done. Most of the time I would suspect the data in the track would be standard sectors, with SYNC, GAP, header, data and checksums. The arrangement of the data when it is in a standard GCR sector layout is beyond the scope of this document.

Since it is a flexible format in both track count and track byte size, there is no "standard" file size. However, given a few constants like 42 tracks and halftracks, a track size of 7928 bytes and no speed offset entries, the typical file size will a minimum of 333744 bytes.

Below is a dump of the header, broken down into its various parts. After that will be an explanation of the track offset and speed zone offset areas, as they demand much more explanation.

```
Addr  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
----  -----------------------------------------------
0000: 47 43 52 2D 31 35 34 31 00 54 F8 1E .. .. .. ..
```

| **Offset** | **Description** |
|---|---|
| $0000-0007 | File signature (`GCR-1541`) |
| $0008 | `G64` version (presently only $00 defined) |
| $0009 | Number of tracks in image (usually $54, decimal 84) |
| $000A-000B | Size of each stored track in bytes (usually 7928, or $1EF8) in LO/HI format. |

An obvious question here is "why are there 84 tracks defined when a normal `D64` disk only has 35 tracks?" Well, by definition, this image includes all half-tracks, so there are actually 42 tracks and 42 half tracks. The 1541 stepper motor can access up to 42 tracks and the in-between half-tracks. Even though using more than 35 tracks is not typical, it was important to define this format from the start with what the 1541 is capable of doing, and not just what it typically does.

At first, the defined track size value of 7928 bytes may seem to be arbitrary, but it is not. It is determined by the fastest write speed possible (speed zone 0), coupled with the average rotation speed of the disk (300 rpm). After some math, the answer that actually comes up is 7692 bytes. Why the discrepency between the actual size of 7692 and the defined size of 7928? Simply put, not all drives rotate at 300 rpm. Some can be faster or slower, so a upper safety margin of +3% was built added, in case some disks rotate slower and can write more data. After applying this safety factor, and some rounding-up, 7928 bytes per track was arrived at.

Also note that this upper limit of 7928 bytes per track really only applies to 1541 and compatible disks. If this format were applied to another disk type like the SFD1001, this value would be higher.

Below is a dump of the first section of a `G64` file, showing the offsets to the data portion for each track and half-track entry. Following that is a dump of the speed zone offsets.

```
Addr  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
----  -----------------------------------------------
0000: .. .. .. .. .. .. .. .. .. .. .. .. AC 02 00 00
0010: 00 00 00 00 A6 21 00 00 00 00 00 00 A0 40 00 00
0020: 00 00 00 00 9A 5F 00 00 00 00 00 00 94 7E 00 00
0030: 00 00 00 00 8E 9D 00 00 00 00 00 00 88 BC 00 00
0040: 00 00 00 00 82 DB 00 00 00 00 00 00 7C FA 00 00
0050: 00 00 00 00 76 19 01 00 00 00 00 00 70 38 01 00
0060: 00 00 00 00 6A 57 01 00 00 00 00 00 64 76 01 00
0070: 00 00 00 00 5E 95 01 00 00 00 00 00 58 B4 01 00
0080: 00 00 00 00 52 D3 01 00 00 00 00 00 4C F2 01 00
0090: 00 00 00 00 46 11 02 00 00 00 00 00 40 30 02 00
00A0: 00 00 00 00 3A 4F 02 00 00 00 00 00 34 6E 02 00
00B0: 00 00 00 00 2E 8D 02 00 00 00 00 00 28 AC 02 00
00C0: 00 00 00 00 22 CB 02 00 00 00 00 00 1C EA 02 00
00D0: 00 00 00 00 16 09 03 00 00 00 00 00 10 28 03 00
00E0: 00 00 00 00 0A 47 03 00 00 00 00 00 04 66 03 00
00F0: 00 00 00 00 FE 84 03 00 00 00 00 00 F8 A3 03 00
0100: 00 00 00 00 F2 C2 03 00 00 00 00 00 EC E1 03 00
0110: 00 00 00 00 E6 00 04 00 00 00 00 00 E0 1F 04 00
0120: 00 00 00 00 DA 3E 04 00 00 00 00 00 D4 5D 04 00
0130: 00 00 00 00 CE 7C 04 00 00 00 00 00 C8 9B 04 00
0140: 00 00 00 00 C2 BA 04 00 00 00 00 00 BC D9 04 00
0150: 00 00 00 00 B6 F8 04 00 00 00 00 00 .. .. .. ..
```

| **Offset** | **Description** |
|---|---|
| $000C-000F | Offset to stored track 1.0 ($000002AC, in LO/HI format, see below for more) |
| $0010-0013 | Offset to stored track 1.5 ($00000000) |
| $0014-0017 | Offset to stored track 2.0 ($000021A6) |
| ... | |
| $0154-0157 | Offset to stored track 42.0 ($0004F8B6) |
| $0158-015B | Offset to stored track 42.5 ($00000000) |

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
```

```
      -----------------------------------------------
0150: .. .. .. .. .. .. .. .. .. .. .. .. 03 00 00 00
0160: 00 00 00 00 03 00 00 00 00 00 00 00 03 00 00 00
0170: 00 00 00 00 03 00 00 00 00 00 00 00 03 00 00 00
0180: 00 00 00 00 03 00 00 00 00 00 00 00 03 00 00 00
0190: 00 00 00 00 03 00 00 00 00 00 00 00 03 00 00 00
01A0: 00 00 00 00 03 00 00 00 00 00 00 00 03 00 00 00
01B0: 00 00 00 00 03 00 00 00 00 00 00 00 03 00 00 00
01C0: 00 00 00 00 03 00 00 00 00 00 00 00 03 00 00 00
01D0: 00 00 00 00 03 00 00 00 00 00 00 00 03 00 00 00
01E0: 00 00 00 00 02 00 00 00 00 00 00 00 02 00 00 00
01F0: 00 00 00 00 02 00 00 00 00 00 00 00 02 00 00 00
0200: 00 00 00 00 02 00 00 00 00 00 00 00 02 00 00 00
0210: 00 00 00 00 02 00 00 00 00 00 00 00 01 00 00 00
0220: 00 00 00 00 01 00 00 00 00 00 00 00 01 00 00 00
0230: 00 00 00 00 01 00 00 00 00 00 00 00 01 00 00 00
0240: 00 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00
0250: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0260: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0270: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0280: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0290: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
02A0: 00 00 00 00 00 00 00 00 00 00 00 00 .. .. .. ..
```

| Offset | Description |
|---|---|
| $015C-015F | Speed zone entry for track 1 ($03, in LO/HI format, see below for more) |
| $0160-0163 | Speed zone entry for track 1.5 ($03) |
| ... | |
| $02A4-02A7 | Speed zone entry for track 42 ($00) |
| $02A8-02AB | Speed zone entry for track 42.5 ($00) |

Starting here at $02AC is the first track entry (from above, it is the first entry for track 1.0)

The track offsets (from above) require some explanation. When one is set to all 0's, no track data exists for this entry. If there is a value, it is an absolute reference into the file (starting from the beginning of the file). From the track 1.0 entry we see it is set for $000002AC. Going to that file offset, here is what we see...

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
      -----------------------------------------------
02A0: .. .. .. .. .. .. .. .. .. .. .. .. 0C 1E FF FF
02B0: FF FF FF 52 54 B5 29 4B 7A 5E 95 55 55 55 55 55
02C0: 55 55 55 55 55 55 FF FF FF FF FF 55 D4 A5 29 4A
02D0: 52 94 A5 29 4A 52 94 A5 29 4A 52 94 A5 29 4A 52
```

| Offset | Description |
|---|---|
| $02AC-02AD | Actual size of stored track (7692 or $1E0C, in LO/HI format) |
| $02AE-02AE+$1E0C | Track data |

Following the track data is filler bytes. In this case, there are 368 bytes of unused space. This space can contain anything, but for the sake of those wishing to compress these images for storage, they should all be set to the same value. In the sample I used, these are all set to $FF.

Below is a dump of the end of the track 1.0 data area. Note the actual track data ends at address $20B9, with the rest of the block being unused, and set to $FF.

```
       00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
       -----------------------------------------------
1FE0: 52 94 A5 29 4A 52 94 A5 29 4A 52 94 A5 29 4A 52
1FF0: 94 A5 29 4A 52 94 A5 29 4A 52 94 A5 29 4A 52 94
2000: A5 29 4A 52 94 A5 29 4A 52 94 A5 29 4A 52 94 A5
2010: 29 4A 52 94 A5 29 4A 52 94 A5 29 4A 52 94 A5 29
2020: 4A 52 94 A5 29 4A 52 94 A5 29 4A 52 94 A5 29 4A
2030: 55 55 55 55 55 55 FF FF FF FF FF FF FF FF FF FF
2040: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
2050: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
2060: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
2070: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
2080: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
2090: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
20A0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
20B0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
20C0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
20D0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
20E0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
20F0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
2100: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
2110: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
2120: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
2130: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
2140: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
2150: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
2160: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
2170: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
2180: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
2190: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
21A0: FF FF FF FF FF FF .. .. .. .. .. .. .. .. .. ..
```

The speed offset entries can be a little more complex. The 1541 has four speed zones defined, which means the drive can write data at four distinct speeds. On a normal 1541 disk, these zones are as follows:

| **Track Range** | **Speed Zone** |
|---|---|
| 1-17 | 3 (highest writing speed) |
| 18-24 | 2 |
| 25-30 | 1 |
| 31 and up | 0 (lowest writing speed) |

Note that you can, through custom programming of the 1541, change the speed zone of any track to something different (change the 3 to a 0) and write data differently. From the dump of the speed offset entries above, we see that all the entries are in the range of 0-3. If any entry is less than 4, this is not considered a speed offset but defines the whole track to be recorded at that one speed.

In the example I had, there were no offsets defined, so no speed zone dump can be shown. However, I can define what should be there. You will have a block of data, 1982 bytes long. Each byte is encoded to represent the speed of 4 bytes in the track offset area, and is broken down as follows:

```
Speed entry $FF:  in binary %11111111
                             |'|'|'|'
                             | | | |
                             | | | +- 4'th byte speed (binary 11, 3 dec)
                             | | +--- 3'rd byte speed (binary 11, 3 dec)
                             | +----- 2'nd byte speed (binary 11, 3 dec)
                             +------- 1'st byte speed (binary 11, 3 dec)
```

It was very smart thinking to allow for two speed zone settings, one in the offset block and another defining the speed on a per-byte basis. If you are working with a normal disk, where each track is one constant speed, then you don't need the extra blocks of information hanging around the image, wasting space.

What may not be obvious is the flexibility of this format to add tracks and speed offset zones at will. If a program decides to write a track out with varying speeds, and no speed offset exist, a new block will be created by appending it to the end of the image, and the offset pointer for that track set to point to the new block. If a track has no offset yet, meaning it doesn't exist (like a half-track), and one needs to be added, the same procedure applies. The location of the actual track or speed zone data is not important, meaning they do not have to be in any particular order since they are all referenced by the offsets at the beginning of the image.

# 14 Acknowledgments

VICE derives from X64, the first Commodore 64 emulator for the X Window System. Here is an informal list of the people who were mostly involved in the development of X64 and VICE:

The VICE core team:

- **Andreas Boose** gave lots of information and bug reports about the VIC-II, the 6510 and the CIAs; moreover, he wrote several test-routines that were used to improve the emulation. He also added cartridge support and has been the main head behind the drive and datasette emulation since version 0.15. Also added several UI elements to the MSDOS, MS-Windows and Unix ports. He rewrote the C128 emulation adding Z80 mode, C64 mode and function ROM support, wrote the screenshot and the event system and started the plus4 emulator. Restructured the serial bus emulation and added realdrive and rawdrive support.
- **Dag Lem** implemented the reSID SID emulation engine and video hardware scaling.

- **Tibor Biczo** improved the MS-Windows port and plus4 emulation.
- **Andreas Dehmel** wrote the Acorn RISC OS port.
- **Andreas Matthies** improved the datasette support, the VIC20 video emulation and some ui stuff in the Win32 and MSDOS port. He also wrote the BeOS port and implemented video/audio capture support. Improved history recording/playback and implented support for video recording and netlink feature. Made the win32 user changable keyboard shortcut system. Various bug(fixe)s. ;-)
- **Martin Pottendorfer** Implemented Gnome Port based on Oliver Schaertels GTK+ port Added support code for internationalization based on gettext Improved the *nix fullscreen support Translated the Unix Port to German Implemented the fliplists + ui (unix).
- **Spiro Trikaliotis** wrote the Win32 console implementation for the built-in monitor, corrected some REU related bugs, improved the CIA emulation, added com-port CIA support to the win32 port, added text copy and paste support to the win32 port, added support for the TFE and RR-Net (cs8900a), and wrote some further patches.
- **Markus Brenner** added VDC emulation to x128 and added support for some more cartridges.
- **David Hansel** wrote the Star NL10 printer driver, implemented IEC devices and improved the tape emulation.
- **Marco van den Heuvel** Translated the UI to Dutch. Made the Internationalization support for the Win32 and Amiga ports. Wrote the GEORAM and RamCart cartridge code. Wrote the c64 +60K, +256K and 256K memory expansions code. Wrote the pet REU code. Wrote the plus4 memory expansions code. Made the ethernet support for the Msdos port. Maintains the RiscOS, QNX 4.x, QNX 6.x, SkyOS, Solaris, Openserver, Unixware, HPUX, GP2X, Minix 3.x, Amiga, ppc-BeOS and newly resurrected OS/2 binary ports. Maintains the win64 and watcom project files. Co-maintains the SDL port(s). Added new .crt support. Added new screenshot formats. Added new sound recording support. Added SIDcart support for pet, plus4 and vic20. Improved the MMC64 emulation. Added 2 MHz mode and banks 2/3 support for x128. Added the various userport joystick emulations. Added text copy and paste support to the amiga and beos ports. Added DQBB and Isepic cartridge support. Added SFX soundsampler and soundexpander support. Added digiblaster support. And lots of other fixes and improvements.
- **Christian Vogelgsang** maintains the Mac OS X port. Added Intel Mac support and universal binary creation. Wrote the build scripts for all external Mac libraries and the bindist bundle tool. Improved the TFE chip emulation. Added some Gtk+ fixes.
- **Fabrizio Gennari** added some improvements to the MSDOS- and GTK port.
- **Hannu Nuotio** implemented DTV flash emulation, DTV support in the monitor, large parts of the DTV VIC emulation, burst mode and skip cycle emulation as well as many other things. Added NEOS and Amiga mouse, paddle and light pen support. Added new monitor commands and features, including memmap. Made MIDI support and OSS MIDI driver. Implemented most of the SDL UI. Rewrote xvic CPU/VIC-I core for cycle based emulation.
- **Daniel Kahlin** Worked on DTV VIC emulation, palette, DTV SID support in resid, better DMA/Blitter support and did lots of refactoring. Added new monitor commands

and features. Improved the VIC emulation for xvic. Made MIDI driver code for win32. Rewrote the xvic cartridge system. Added Mega-Cart and Final Expansion V3.2 support to xvic.

- **Antti S. Lankila** Made the ReSID-fp engine, rewrote the PAL emulation code and fixed the sound core for lower latency. Rewrote DTV SID support (ReSID-dtv).

Former team members:

- **M. Kiesel** started implementing x64dtv. The C64DTV memory model and early versions of the DMA and Blitter engine have been implemented by him. Added new monitor commands and features.
- **Thomas Bretz** Copyright © 1999-2004 Started the OS/2 port.
- **Daniel Sladic** Copyright © 1997-2001 Started the work on hardware-level 1541 emulation and wrote the new monitor introduced with VICE 0.15.
- **Andr Fachat** Copyright © 1996-2001 Wrote the PET and CBM-II emulators, the CIA and VIA emulation, the IEEE488 interface, implemented the IEC serial bus in 'xvic' and made tons of bug fixes.
- **Ettore Perazzoli** Copyright © 1996-1999 Made the 6510, VIC-II, VIC-I and CRTC emulations, part of the hardware-level 1541 emulation, speed optimizations, bug fixes, the event-driven cycle-exact engine, the Xt/Xaw/Xfwf-based GUI for X11, a general code reorganization, the new resource handling, most of the documentation. He also wrote the MS-DOS port and the initial MS-Windows port (well, somebody had to do it).
- **Teemu Rantanen** Copyright © 1993-1994, 1997-1999 Implemented the SID emulation and the trap-based disk drive and serial bus implementation; added support for multiple display depths under X11. Also wrote 'c1541'
- **Jouko Valta** Copyright © 1993-1996 Wrote 'petcat' and 'c1541', 'T64' handling, user service and maintenance (most of the work in x64 0.3.x was made by him); retired from the project in July 96, after VICE 0.10.0.
- **Jarkko Sonninen** Copyright © 1993-1994 He was the founder of the project, wrote the old version of the 6502 emulation and the XDebugger, and retired from the project after x64 0.2.1.

Internationalization Team:

- **Peter Krefting** provided the Swedish user interface translations.

  **Andrea Musuruane** provided the Italian user interface translations.

  **Czirkos Zoltan** and **Karai Csaba** provided the Hungarian user interface translations.

  **Emir Akaydin** provided the Turkish user interface translations (in world record time).

  **Mikkel Holm Olsen** provided the Danish user interface translations and fixed a few monitor bugs.

  **Paul Dub** from Rivire-du-Loup, Qubec, provided the French user interface translations.

  **Flooder** provided parts of the Polish user interface translations.

External contributors:

- **Michael Schwendt** helped with the SID (audio) chip emulation, bringing important suggestions and bug reports, as well as the wave tables and filter emulation from his SIDplay emulator.

- **Christian Bauer** wrote the very interesting "VIC article" from which we got invaluable information about the VIC-II chip: without this, the VIC-II implementation would have not been possible.
- **Wolfgang Lorenz** wrote an excellent 6510 test suite that helped us to debug the CPU emulation.
- **Giuliano Procida** is the maintainer of the VICE `deb` package for the Debian distribution, and also helped proofreading the documentation.
- **Marko Mkel** wrote lots of CPU documentation.
- **Chris Sharp** wrote the AIX sound driver.
- **Krister Walfridsson** implemented joystick and sound support for NetBSD.
- **Peter Andrew Felvegi aka Petschy** fixed a couple of bugs in the fast serial emulation.
- **Olaf Seibert** contributed some PET, and disk drive patches.
- **Daniel Fandrich** contributed some disk drive patches.
- **Heiko Selber** contributed some VIC20 I/O patches.
- **Steven Tieu** added initial support for 16/24 bpp X11 displays.
- **Alexander Lehmann** added complete support for all the VIC20 memory configurations for the old VICE 0.12.
- **Lionel Ulmer** implemented joystick support for Linux and a first try of a SID emulation for SGI machines.
- **Bernhard Kuhn** made some joystick improvements for Linux.
- **Gerhard Wesp** contributed the `extract` command in `c1541`.
- **Ricardo Ferreira** contributed the `unlynx` and `system` commands in `c1541` and added aRts sound support.
- **Tomi Ollila** donated `findpath.c`.
- **Richard Hable** contributed the initial version of the REU emulation.
- **Vesa-Matti Puro** wrote the very first 6502 CPU emulator in x64 0.1.0. That was the beginning of the story. . .
- **Dan Miner** contributed some patches to the fast disk drive emulation.
- **Frank Prindle** contributed some patches.
- **Peter Weighill** gave many ideas and contributed the ROM patcher.
- **Dominique Strigl**, **Craig Jackson** and **Lasse Jyrkinen** contributed miscellaneous patches in the old X64 times.
- **Per Olofsson** digitalized the C64 colors used in the (old) default palette.
- **Paul David Doherty** wrote `zip2disk`, on which the Zipcode support in `c1541` is based.
- **Robert H. Forsman Jr.**, **Brian Totty** and **Robert W. McMullen** provided the widget set for implementing the `Xaw` GUI.
- **Shawn Hargreaves** wrote Allegro, the graphics and audio library used in the MS-DOS version.
- **Peter Schepers** contributed a document describing the G64 image format.
- **Oliver Schaertel** wrote the X11 full screen, parts of custom ROM set support and 1351 mouse emulation for unix.

- **Luca Montecchiani** contributed a new Unix joystick driver.
- **Dirk Farin** rewrote the MITSHM code.
- **Manfred Spraul** wrote the MS-Windows text lister.
- **Peter Karlsson** provided the swedish UI translations.
- **Paul Dub** provided the french translation for the Unix ports.
- **Nathan Huizinga** added support for Expert and Super Snapshot carts.
- **Andrea Musuruane** provided the italian UI translations.
- **Flooder** provided the polish translation for the Unix ports
- **Michael Klein** contributed the ESD sound driver, basic support for the OPENCBM library and some other patches.
- **David Holz** provided a label file which gives the built-in monitor the labels for the C64.
- **Lasse rni** contributed the Windows Multimedia sound driver
- **Frank Knig** contributed the Win32 joystick autofire feature.
- **Philip Timmermann** did a lot of research about the VIC-II colors.
- **John Selck** improved the video rendering and added the fast PAL emulation. Implemented new color generation based on P. Timmermanns knowledge.
- **webulator** provided win32 drag & drop support
- **Robert Willie** added some additional commands to the fsdevice emulation.
- **Kajtar Zsolt** wrote the IDE64 interface emulation and did some small fixes.
- **ck!** provided a win32 cbm character font.
- **Dirk Jagdmann** wrote the Catweasel sound driver.
- **Maciej Witkowiak** did some IDE64 and C1541 fixes.
- **Roberto Muscedere** improved support for REL files.
- **Rami Rasanen** rewrote the VIC20 sound code.
- **iAN CooG** Added win32 vsid GUI and contributed various small patches.
- **Mike Dawson** provided the GP2X port and co-maintains the GP2X ports.
- **Mathias Roslund** provided the AmigaOS4 port and co-maintains the AmigaOS ports.
- **Eliseo Bianchi** provided the italian Amiga translations.
- **Johan Samuelsson** provided the swedish Amiga translations.
- **Harry "Piru" Sintonen** provided lots of fixes and improvements for the Amiga ports.
- **Ilkka "itix" Lehtoranta** provided the routines for the cybergraphics support for the Amiga ports.
- **Wolfgang Moser** provided small optimization fixes to the GCR code, provided an excellent REU test suite and added REU fixes, and is always the good guy reviewing and commenting changes in the background.
- **Peter Gordon** provided support for native AmigaOS4 compiling.
- **Bernd Kortz** provided some fixes for ZETA and the ZETA binary package.
- **Andr351 'JoBBo' Siegel** provided the native morphos icons.
- **Markus Stehr** provided the MMC64 emulation.

- **Gunnar Ruthenberg** provided some VIC-II enhancements and improved the win32 port.
- **Groepaz/Hitmen** Added new more precise PAL emulation. Added support for the Action Replay 4 and StarDos cartridges. Provided several fixes.
- **Ingo Korb** Corrected block allocation and interleave for c1541/vdrive, added rudimentary xplus4 tape recording support, corrected a case of missing Pi symbols in petcat, changed the trap opcode byte, stopped the high-level serial drive code from responding to addresses 16-30 and was forced to update this entry himself.
- **Georg Feil** Added support for toggling CB2 sound output line in the PET emulator.
- **Greg King** Added a working RTC to the emulation of the IDE64 cartridge.
- **Thomas Giesel** Added new monitor commands and features.
- **Marcus Sutton** Made some console, dialog and joystick fixes for the BeOS port.
- **Mustafa 'GnoStiC' Tufan** Made improvements to the gp2x port.
- **Ville-Matias Heikkila** Rewrote the vic20 sound code.
- **Errol Smith** improved VDC emulation.
- **Peter Edwards** implemented the SDL UI slider control and fixed some GP2X/Dingoo SDL UI issues.
- **Lutz Sammer**
- **Ralph Mason**
- **George Caswell**
- **Jasper Phillips**
- **Luca Forcucci**
- **Asger Alstrup**
- **Bernhard Schwall**
- **Salvatore Valente**
- **Arthur Hagen**
- **Douglas Carmichael**
- **Ferenc Veres**
- **Frank Reichel**
- **Ullrich von Bassewitz**
- **Holger Busse**

Last but not least, a very special thank to Andreas Arens, Lutz Sammer, Edgar Tornig, Christian Bauer, Wolfgang Lorenz, Miha Peternel and Per Hkan Sundell for writing cool emulators to compete with. `:-)`

# 15 Copyright

# 16 Contact information

## 16.1 VICE home page

You can find the latest news about VICE at the official VICE home page:

`http://www.viceteam.org/`

VICE has moved its source repository to public services provided by SourceForge. You can find it at

`http://sf.net/projects/vice-emu`.

We would like to thank the SourceForge staff for that help.

If you are going to report a bug, please check those pages *first*; it is possible that the problem you encountered has already been fixed with a newer version.

Please, also have a look at the VICE knowledge base at

`http://vicekb.trikaliotis.net/`

## 16.2 How to send feedback

Before contacting us, have a look at the VICE knowledge base at `http://vicekb.trikaliotis.net/` if your question is answered there. Keep in mind that we work on VICE in our spare-time, so the more time we don't need to answer the same questions over and over again, the more time we have to improve the emulation itself. On the other hand, that does not mean that you should not contact us, especially if you find bugs or have suggestions which might improve the emulation.

Bug reports, suggestions, support requests should be directed to the SourceForge trackers at

- `http://sourceforge.net/tracker/?group_id=223021`.

This way, you, the users, and we, the developers, can track what has been reported and what has been already fixed. Ideally, also sent the report to the mailing address of the Vice team at

- VICE Mailing List (`vice-devel@firenze.linux.it`) for all general questions, bug reports, suggestions.

You can also contact (some of) us on IRC, at #vice-dev on freenode.

It's always nice to receive feedback and/or bugreports about VICE, but please read these few notes before sending mail to anybody in the team.

- Please put the word '`VICE`' *in all capitals* in your subject line (e.g., '`VICE fails to run game XXX`'). This helps mail splitting and reduces chances that your message is unintentionally deleted, forgotten or lost.
- Please don't send any HTML mail (we really hate that!). If you use M$ Outlook or Netscape Communicator, make sure you turn off the "rich text" (HTML) feature.
- Please don't send *any* binaries without asking first.
- Please read the following documents carefully before reporting a bug or a problem you cannot solve:
  - the VICE documentation (you are reading it!);
  - the VICE FAQ (it is available on the Internet, and reachable from the VICE home page: `http://www.viceteam.org/`);
  - The VICE knowledge base (it is available on the Internet at `http://vicekb.trikaliotis.net/`);
  - the `comp.emulators.cbm` and `comp.sys.cbm` FAQs (see Section 16.5 [FAQs you should read], page 121).
- When you report a bug, please try to be as accurate as possible and describe how it can be reproduced to the very detail. You should also tell us what machine you are running on, what operating system you are using as well as the version of it.

- Please don't ask us how to transfer original C64 disk or tapes to your PC; this has been asked a gazillion times through email. To transfer disks, you can use the Star Commander (http://sta.c64.org/sc.html) on DOS, and OpenCBM (http://www.trikaliotis.net/opencbm) on Windows and Linux. And no, you cannot read C64 disks with your old 5"1/4 PC drive.
- Please don't ask us where to find games for the emulator on the Internet.
- Please don't ask us when the next version will be out, because we really don't know.
- Please write in English.

In any case, we would be *really* glad to receive your comments about VICE. We cannot always answer all the email, but we surely read all of it.

Thanks!

## 16.3 How to contribute

If you want to make a major contribution, please *ask* first. It has already happened a couple of times that somebody started working at something that had already been done but not released to the public yet, and we really do *not* want anybody to waste time.

If you are going to make a patch, please make sure the patch is relative to the very latest version, and provide us with the following:

- Make sure you are giving us a diff against the latest Subversion trunk version of VICE. For instructions on accessing the Subversion repository, first read http://sourceforge.net/svn/?group_id=223021 and get it with the command:

  ‘`svn co https://vice-emu.svn.sourceforge.net/svnroot/vice-emu/trunk vice-src`’
- send a unified diff file against the trunk version of VICE (see above bullet point) by using the command: ‘`svn diff`’ inside of the SVN workspace you checked out before.
- If you cannot use SVN for one or the other reason, send a unified diff file containing all the changes you have made ‘`diff -u`’; please don't use plain ‘`diff`’), as it adds much work for us to get it working;
- GNU-style ‘`ChangeLog`’ entries with a description of the changes you have made (look at the ‘`ChangeLog`’s provided with the original VICE sources for an example).

This is very important, and makes adding patches much smoother and safer.

People willing to port VICE to other platforms are always welcome. But notice from experience it will take at least a full year of continuous work to write a well working and stable port.

## 16.4 Interesting newsgroups

There are some Usenet newsgroups you might be interested in:

- `comp.emulators.cbm`, discussing about emulators of Commodore 8-bit machines (definitely not Amiga emulators).
- `comp.sys.cbm`, discussing various topics regarding real Commodore 8-bit machines. This newsgroup is mainly for people who actually use original Commodore equipment (so please don't talk about emulation here).
- `comp.emulators.misc`, discussing emulators in general.

## 16.5 FAQs you should read

We recommend reading the `comp.emulators.cbm` and `comp.sys.cbm` FAQs, which are posted regularly on the corresponding newsgroups and are also available via FTP from `ftp://rtfm.mit.edu`.

# Concept Index

## A



## C



## D



## E



## G



## H



## L



## M



## O

# Index of Resources

## F



## G



## H



## I



## J



## K



## M



## P



## R

## S



## U



## V



## W

# Table of Contents