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, SCPU64 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 20 [Contacts], page 375. 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, SCPU64 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, but is slowly being made universal.

2.1 C64 emulator features

As of version 2.3, two C64 emulators are provided: '`x64`' (fast) and '`x64sc`' (accurate). As of version 3.4 '`x64`' will no more get built by default and is not contained in the default binary packages.

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 two 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. The reSID engine is a lot more accurate than the standard engine, but it is 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.

The VIC20 emulator 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 4KiB 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 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 (`https://sourceforge.net/p/vice-emu/code/HEAD/tree/techdocs/PET/PETdoc.txt?format=raw`) 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.

The PET 8096 is basically a PET 8032 with a 64KiB extension board which allows remapping the upper 32KiB 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 128KiB RAM in total. Of the additional 32KiB RAM you can use only some in blocks of 4KiB, 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 32KiB for BASIC text and 32KiB for variables. See PETDoc (`https://sourceforge.net/p/vice-emu/code/HEAD/tree/techdocs/PET/PETdoc.txt?format=raw`) for more information.

The PET 8296D is an 8296 with built-in 8250 low-profile dual disk drive.

The PET 8296GD is an 8296D with additionally a "HiRes Emulator" (HRE). This is a cheaper version of a "HRG" hi-res board which was based on Thomson chips. This version instead uses no additional hardware support apart from some memory mapping tricks. It has supporting software in the hre-*.bin rom files.

The SuperPET also is a PET 8032 with an expansion board. It can map 4KiB at a time out of 64KiB into the `$9***` area. Also it has an ACIA 6551 for RS232 communication. The 6809 CPU that is built into the SuperPET is now emulated, since release 2.4, including the 6702 dongle chip.

The Super-OS-9 MMU expansion, developed by TPUG (Toronto PET Users Group) is also emulated.

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 7.7.1 [PET model], page 145,

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 128KiB RAM, the others 256KiB. This implies some banking scheme, as the 6502 can only address 64KiB. And indeed those machines use a 6509, that can address 1 MiB 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 1MiB. 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 7.8.1 [CBM-II model], page 156.

2.7 SCPU64 emulator features

The XSCPU64 emulator is a simulation of a C64 equipped with a SuperCPU64 V2B. Features:

- 20 MHz asynchronous single cycle 65816 CPU core with proper dummy and invalid cycle handling.
- 128 KiB static RAM, 0-16 MiB SIMM RAM, 64-512 KiB EPROM emulated and their respective timing details.
- All RAM optimization configurations supported with write buffer.
- I/O area access delays, write through to SRAM implemented.
- Memory mappings including cartridge and boot memory map and kernal shadow.
- Hardware registers and switches implemented.
- Replacement SCPU64 ROM compatible with the original to avoid distribution problems
- It's using the single cycle VICII core for accurate simulation

Still to do:

- Measure and verify VICII interrupt phase shift
- Measure and verify BA phase shift
- SIMM RAM extra 7.5 cycle refresh delay every 10us missing.
- CPU NMI support for "reset" button

The emulation is quite accurate but not perfect. If you code something timing intensive using this simulation please always check it on real hardware to avoid bad surprises.

The hardware itself is asynchronous in nature, therefore caution must be taken to not do long timing loops without synchronization in 20 MHz mode. Also don't squeeze out the last remaining cycles without leaving a safety buffer. Synchronization points can be created by doing I/O reads or writes and leaving a few hundred cycles left each frame will not hurt.

Otherwise it can happen that the code is running on this version of VICE or my SCPU64 V2+C128D perfectly but nowhere else due to manufacturing variations and frequency drifts.

2.8 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. *The key printed on the host keyboard will be pressed in the emulator.*

There is, however, one problem with symbolic mapping: some keys really need to be mapped specially regardless (those that do not exist on a PC keyboard). Some examples are the `Commodore` key, `RUN/STOP`, `Clear/Home`. The exact mapping depends on your host layout,

but should be easy to find out by try and error :) If in doubt, you can read the keyboard mapping files.

The second way (*positional mapping*) is to map every key on the host keyboard to the key which has the same position on the keyboard of the emulated machine. This way the keyboard is more comfortable to use in those programs (such as some games) that require the keys to be in the correct positions. On the other hand it can be quite confusing if you are not very familiar with the original emulated keyboards. Also not all keys can be mapped exactly this way either, which means some of them still need to be mapped to other keys (see above).

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.

We depend a lot on your support to improve the keyboard maps, as we can not test all emulators in all possible configurations and using all host keyboard mappings. Please report any problems to us so we can fix them!

If you experience problems with 'accent' keys such as acute, grave, tilde, circumflex, diaresis (and possibly more/other, depending on your host keyboard layout) try switching to a "no deadkeys" layout in your OS. In any case, please also report these problems so we can fix them!

To find out the keycodes to use, incase you want to edit the keymaps yourself, you can enable showing the keycodes in the status bar in the settings.

2.9 The joystick emulation

Joysticks can be emulated both via the keyboard and via a real joystick connected to the host machine.

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 is configurable to your liking.

2.10 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, 8250, and D9090/60 drives in one of the following ways:

- using disk images, i.e., files that contain a dump of all the blocks contained in a real floppy disk;
- 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.
- accessing a real device connected to the host machine. This works using the OpenCBM library. You can get it from `https://spiro.trikaliotis.net/opencbm`.

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 access it is required to enable this type of emulation.

The IEEE488 drives (2031, 2040, 3040, 4040, 1001, 8050, 8250, and D9090/60) 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 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 2040, 3040, 4040, 8050 and 8250 are Dual Drive Floppy Disks. This means that these drives handle two disks. To accomplish the emulation, only two disks can be emulated, namely units #8 and #10. The attached image, track display and LED display of unit #9 and #11 are used for the second drive of the dual disk drives. On unix the unit number display (8 or 9, 10 or 11) in the emulation window changes to the drive number display (0 or 1).

The Commodore 2040, 3040, 4040, 1001, 8050, 8250, and D9090/60 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.

The D9090/60 is the only Commodore branded hard drive produced for the PET series computers, and were often used by C64 and C128 users for their significant storage capacity (29162/19441 free blocks). Just like the other IEEE drives before it, it uses a separate CPU as the FDC which in turn communicates with the SASI-to-ST506 bridge (which is controlled by an AM2910). The hardware design is very similar to the 8050/8250 drive.

Creative Micro Designs (CMD) produced the last drives for the Commodore 8-bit systems. They first released the hard drive (HD) line, and later the floppy drive (FD) line. The CMD HD series can support up to 4 GiB HDs with 255 separate partitions, while the CMD FD series can support up to 3.3 MB extended density floppy disks with 31 separate partitions. The FD series are also backwards compatible with 1581 media. The DOS for the FD series is stored on a ROM (`dos2000` and `dos4000`, the latest versions being 1.40).

The CMD HD uses a small boot ROM (`dosCMDHD`, the latest version is 2.80) which loads the primary DOS (latest is 1.92) off the HD itself. This allows for easy upgrades and expandability. This is also the only drive to use the front panel buttons to control the mode of the drive on reset. There are three modes of operation: normal, configuration, and installation. VICE supports placing the drive in either of these modes through the "Reset" sub-menu on the status bar for the GTK3 interface, and the "Reset" menu in the SDL interface. When creating a new DHD image, simply create an EMPTY file and VICE will automatically place the drive in installation mode. Once the DOS is installed, the CMD "hd-tools" program can be used to configure various settings and partition the drive; this is done in configuration mode for safety. When in either configuration or installation mode, the device number is set to 30. Therefore, it is not suggested to place two or more CMD HDs in either of these modes on the same bus at the same time. When migrating from real CMD hardware, use any HDD imaging software ("dd" or GNU "ddrescue" on Linux) to copy the raw contents of a device to a file. The destination file should have a "DHD" extension. For those users with multiple disks, SCSI ID 0 should have the extension "DHD", but any other drives should have ".S<ID>0" where <ID> is the SCSI ID. Place all the files in the same folder when attaching to the "DHD" file. The other files will automatically be scanned for and connected as well. The CMD HD boot ROM is used for partition management, and ALL versions have a known bug which corrupts data when deleting paritions across multiple SCSI drives. To avoid this scenario, it is highly suggested that another full DHD image be created in VICE (on another unit) and all the files be copied over from multi-disk configurations, using CMD "fcopy" for example, to the new unit. This will allow the user to take advantage of all the CMD HD features without the potential for data loss.

2.11 Supported file formats

VICE supports the most popular Commodore file formats:

- `X64` or `D64` disk image files; Used by the 1541, 2031, 3040, 4040 drives.
- `G64` GCR-encoded 1541 disk image files
- `P64` lowlevel NRZI flux pulse 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
- `D90` CBM D9090/60 disk image format
- `D1M` FD2000/FD4000 DD disk image format
- `D2M` FD2000/FD4000 HD disk image format
- `D4M` FD4000 ED disk image format
- `DHD` CMD HD disk image format
- `T64` tape container files (read-only)
- `TAP` lowlevel tape image files
- `P00` program files
- `CRT` C64 cartridge image files
- `TCRT` tapecart image files

A utility (`c1541`, see Chapter 14 [c1541], page 224) is provided to allow transfers and conversions between these formats.

Notice that the use of the `X64` file format is deprecated 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 17 [File formats], page 236. for a technical description of the supported file formats.

2.12 Common problems

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

2.12.1 Sound problems

VICE should compile and run without major problems on many systems, but there are some known issues related to the sound driver.

If you are having sound problems, such as skipping, first monitor how much CPU time the respective emulator is taking on your system. To run smoothly, on a modern system, it should really never go over 50% or so, except for very short peaks that should also stay well beyond 90%. If you see it takes more, try disabling some of the most CPU intense features (disable CRT emulation, use fastsid instead of reSID, disable true drive emulation).

If the CPU usage is ok, try using a different sound driver. You may also try increasing the sound buffer size (although the default should be ok for modern systems).

All platforms that can run the SDL port (like Amiga, BeOS, etc) should be able to play sound via SDL.

2.12.2 Video problems

Due to the way VICE currently handles its main loop, a common problem with video sync occurs when the real (the monitors) frame rate is very close to, equal, or lower than the framerate of the emulated machine. We recommend to use a framerate which is slightly higher than that (eg 51Hz for the C64).

2.12.3 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.

2.12.4 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:

- `vsid`, the SID player
- `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 (CRTC models)
- `xcbm5x0`, the CBM-II emulator (VIC-II models)
- `xscpu64`, the SCPU64 emulator

You can run each of them by simply typing the name from a shell or by configuring your window manager for example to use them to open disk images.

If you want to look at the log output run them from a terminal window such as `xterm` or `rxvt`. For example, you could do

```
xterm -e x64sc
```

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 30, 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.

`-features`
: List all compile time features

`-default`
: Set default resources (see Chapter 6 [Settings and resources], page 30). This will override all the settings specified before, but not the settings specified afterwards on the command line.

`-config <filename>`
: Specify config file

`-addconfig <filename>`
: Specify extra config file for loading additional resources. This can be used to add "patch sets" for various configurations.

`-dumpconfig <filename>`
: Write the complete config into file. Normally only resources that have been changed from their default value would be written to the config file, however with this option you can see all resources and their current values.

`-logfile <name>`
: Specify log file name (`LogFileName`).

`-verbose` Enable verbose log output.

`-silent` Disable all log output (except errors).

`-keybuf <string>`
: Put the specified string into the keyboard buffer.

`-console` Console mode (for music playback, or for running the emulator test programs)

`-limitcycles <cycles>`
: Automatically exit the emulator after a given number of cycles.

`-chdir <directory>`
: Change the working directory.

`-autostart <name>`
: Autostart <name> (see Section 3.2 [Command-line autostart], page 18).

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

`-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.

`-8d1 <Name>`
`-9d1 <Name>`
`-10d1 <Name>`
`-11d1 <Name>`
: Attach <Name> as a disk image to the second drive of a dual-drive device 8, 9, 10 or 11.

`-attach8ro`
`-attach9ro`
`-attach10ro`
`-attach11ro`
: Attach disk image for drive #8-11 read only (`AttachDevice8Readonly=1`, `AttachDevice9Readonly=1`, `AttachDevice10Readonly=1`, `AttachDevice11Readonly=1`) (all emulators except vsid).

`-attach8d1ro`
`-attach9d1ro`
`-attach10d1ro`
`-attach11d1ro`
: Attach disk image for second drive of a dual-drive #8-11 read only (`AttachDevice8d1Readonly=1`, `AttachDevice9d1Readonly=1`, `AttachDevice10d1Readonly=1`, `AttachDevice11d1Readonly=1`) (all emulators except vsid).

`-attach8rw`
`-attach9rw`
`-attach10rw`
`-attach11rw`
: Attach disk image for drive #8-11 read write (if possible) (`AttachDevice8Readonly=0`, `AttachDevice9Readonly=0`, `AttachDevice10Readonly=0`, `AttachDevice11Readonly=0`) (all emulators except vsid).

`-attach8d1rw`
`-attach9d1rw`
`-attach10d1rw`
`-attach11d1rw`
: Attach disk image for second drive of a dual-drive #8-11 read write (if possible) (`AttachDevice8d1Readonly=0`, `AttachDevice9d1Readonly=0`, `AttachDevice10d1Readonly=0`, `AttachDevice11d1Readonly=0`) (all emulators except vsid).

`-exitscreenshot <name>`
: Specify name of a screenshot file that will be written when the emulator exits. (`ExitScreenshotName`).

`-exitscreenshotvicii <name>`
: Specify name of a screenshot file that will be written when the emulator exits. (`ExitScreenshotName1`). (x128)

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

```
x64sc lovelygame.d64
```

will start the C64 emulator, attaching `lovelygame.d64` 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

```
x64sc "lovelygame.d64:run me"
```

will run the program named `run me` on `lovelygame.d64` instead of the first one.

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

```
x64sc -autostart "lovelygame.d64: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 27. 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

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

Where `PREFIX` is the installation prefix (usually `/usr/local`), `EMU` is the name of the emulated machine (`C64`, `C64DTV`, `C128`, `PET`, `PLUS4`, `CBM-II`, `SCPU64` 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

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

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

1. `$HOME/.local/share/vice/C64`
2. `/usr/local/lib/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 30).

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 KiB)
- `basic`, the Basic ROM (8 KiB)
- `chargen`, the character generator ROM (4 KiB)

The C128 needs the following files:

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

The C128, VIC20, SCPU64 and C64 emulators also need the following DOS ROMs for the hardware-level emulation of the 1540, 1541, 1571, 1581, 2000, and 4000 disk drives, as well as the CMD hard drive:

- `dos1540`, the 1540 drive ROM (16 KiB)
- `dos1541`, the 1541 drive ROM (16 KiB)
- `dos1541II`, the 1541-II drive ROM (16 KiB)
- `dos1571`, the 1571 drive ROM (32 KiB)
- `dos1581`, the 1581 drive ROM (32 KiB)
- `dos2000`, the 2000 drive ROM (32 KiB)
- `dos4000`, the 4000 drive ROM (32 KiB)
- `dosCMDHD`, the CMD HD boot ROM (16 KiB)

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 KiB) (DOS 2.6, Commodore ROM images 901484-03 and 901484-05)
- `dos2040`, the 2040 drive ROM (8 KiB) (DOS 1, Commodore ROM images 901468-06, 901468-07)
- `dos3040`, the 3040 drive ROM (12 KiB) (DOS 2, Commodore ROM images 901468-11, 901468-12 and 901468-13)
- `dos4040`, the 4040 drive ROM (12 KiB) (DOS 2, Commodore ROM images 901468-14, 901468-15 and 901468-16)
- `dos1001`, the 1001/8050/8250 drive ROM (16 KiB) (DOS 2.7, Commodore ROM images 901887-01 and 901888-01)
- `dos9000`, the D9090/60 drive ROM (16 KiB) (DOS 3.0, Commodore ROM images 300516-RevC and 300517-RevC)

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:

- `kernal-1.901439-04-07.bin`, the PET2001 Kernal ROM (4 KiB) (Commodore ROM images 901447-06 and 901447-07, same as 901439-04 and 901439-07)
- `kernal-2.901465-03.bin`, the PET3032 Kernal ROM (4 KiB) (Commodore ROM image 901465-03)
- `kernal-4.901465-22.bin`, the PET4032/8032 Kernal ROM (4 KiB) (Commodore ROM image 901465-22)

- `basic-1.901439-09-05-02-06.bin`, the PET2001 Basic 1 ROM (8 KiB) (Commodore ROM images 901447-09, 901447-02, 901447-03, 901447-04.bin. The -09 ROM is the revised -01 ROM. Same as images 901439-09, 901439-05, 901439-02, 901439-06. The -09 ROM is the revised -01 ROM)
- `basic-2.901465-01-02.bin`, the PET3032 Basic 2 ROM (8 KiB) (Commodore ROM images 901465-01 and 901465-01)
- `basic-4.901465-23-20-21.bin`, the PET4032/8032 Basic 4 ROM (12 KiB) (Commodore ROM images 901465-23, 901465-20 and 901465-21. The -23 ROM is a revised -19 ROM)
- `edit-1-n.901439-03.bin`, the PET2001 editor for graphics keyboards (2 KiB) (Commodore ROM image 901447-05, same as 901439-03)
- `edit-2-b.901474-01.bin`, the PET3032 editor for business keyboards (2 KiB) (Commodore ROM image 901474-01)
- `edit-2-n.901447-24.bin`, the PET3032 editor for graphics keyboards (2 KiB) (Commodore ROM image 901447-24)
- `edit-4-40-n-50Hz.901498-01.bi`, the PET4032 editor for graphics keyboards (2 KiB) (Commodore ROM image 901498-01)
- `edit-4-40-b-50Hz.ts.bin`, the PET4032 editor for business keyboards (2 KiB) (Said to be "901498-01 modified to use a business keyboard on a 50Hz 4032")
- `edit-4-80-b-50Hz.901474-04_.bin`, the PET8032 editor for business keyboards (2 KiB) (Commodore ROM image 901474-04-?)
- `characters-2.901447-10.bin`, the character generator ROM (2KiB). 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 (2KiB). 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.
- `characters.901640-01.bin`, the SuperPET character generator ROM (4KiB). The first half is the same as `characters-2.901447-10.bin`, the second half contains, instead of an upper and lower case set, an ASCII character set and an APL character set. For these sets, the screen code is equal to the ASCII/APL code.
- `waterloo-[abcdf]000.901898-0[1-5].bin`, `waterloo-e000.901897-01.bin`. The Waterloo system ROMs for the 6809 CPU in the SuperPET.
- `hre-9000.324992-02.bin` HiRes Emulator (at $9000) and `hre-a000.324993-02.bin` HiRes BASIC (at $A000). These are the two roms for supporting the HRE on the 8296. The ROMs are initialized by the command `SYS 36864`.

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 128KiB RAM version (C610, C710, B128) uses one bank of 64KiB for the BASIC text and another one for all the variables. The 256KiB 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 larger 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 (8KiB) for the business machines (6xx/7xx)
- `kernal.500`, the KERNAL (8KiB) for the personal machine (510) (901234-02)
- `basic.128`, the CBM-II 128KiB BASIC (16KiB)
- `basic.256`, CBM-II 256KiB BASIC (16KiB)
- `basic.500`, C510 BASIC (16KiB) (901236-02 + 901235-02)
- `chargen.500`, character generator ROM for the C5x0 (4KiB) (901225-01)
- `chargen.600`, character generator ROM for the C6x0 (4KiB)
- `chargen.700`, character generator ROM for the C7x0 (4KiB)

The SCPU64 needs the following files:

- `scpu64`, the SCPU64 ROM (128 KiB)
- `chargen`, the character generator ROM (4 KiB)

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 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 (GTK) or number (SDL) identifying the keysym: you can use the "show keycodes in statusbar" feature 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.
- `16`: deshift key for this keysym/scancode
- `32`: another definition for this keysym/scancode follows
- `64`: key is shift-lock on emulated machine
- `128`: shift modifier required on host
- `256`: key is used for an alternative keyboard mapping
- `512`: alt-r (alt-gr) modifier required on host
- `1024`: ctrl modifier required on host
- `2048`: key is combined with cbm for this keysym/scancode
- `4096`: key is combined with ctrl for this keysym/scancode
- `8192`: key is (left) cbm on emulated machine
- `16384`: key is (left) ctrl on emulated machine

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;
- `!INCLUDE` followed by `"filename"` reads (inserts) file as mapping file. This is useful when adding local mappings to an otherwise generic file (so you dont have to copy the while file, but just add/modify a few keys).
- `!UNDEF` keysym' remove keysym from table
- `!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
  ```

- `!LCTRL`, followed by a row and a column value, specifiy where the left control key is located on the emulated keyboard.
- `!LCBM`, followed by a row and a column value, specifiy where the left CBM key is located on the emulated keyboard.

- `!VSHIFT`, followed by a shiftkey (RSHIFT or LSHIFT), specify what key will be used as a virtual shift key when the shift flag is set.
- `!SHIFTL`, followed by a shiftkey (RSHIFT or LSHIFT), specify what key will be used as a virtual shift-lock key.
- `!VCTRL`, followed by a ctrlkey (LCTRL), specify what key will be used as a virtual control key.
- `!VCBM`, followed by a cbmkey (LCBM), specify what key will be used as a virtual CBM key.

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.

While a Romset file is processed, the directory where the Romset file was found is temporarily prepended to the search path (`Directory` resource). This also means that if you have a setting for `Directory` in it, its effect is limited to the Romset file itself.

4.4.1 Romset command line options

`-romsetfile <File>`
: load the given romset file

`-romsetarchive <File>`
: load the given romset archive

`-romsetarchiveselect <Item number>`
: select the given item from the current romset archive

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 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: most emulation settings can be changed in the *options menu*. Additionally clicking on the various widgets in the status bar gives access to related settings. Settings can be saved and later used with the "Save settings" and "Load settings" menu items, respectively. Also by default, settings will get saved when exiting the emulator. "Restore default settings" restores the factory defaults. See See Chapter 6 [Settings and resources], page 30, for more information about how settings work in VICE.

A lot of settings and actions 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 (from the help menu). This will open either the PDF or popup a browser and open the HTML version of the documentation.

Notice that on Linux this requires VICE to be properly (and fully) installed, eg with a '`make install`'.

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 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 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`, `G64`, `P64` for disk images (devices 8, 9 and 10) and `T64` and `TAP` for tape images.

The ancient `X64` format is deprecated and subject for removal.

Notice that `T64` support is *read-only*, and that the cassette is automatically rewound when you reach its end. For actually emulating tape, the `TAP` format is highly recommended.

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 5.5.1 [Autostart], page 28), 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 "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.

If true drive emulation is turned on, and the *Handle TDE at Autostart* setting is enabled, true drive emulation 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, which is why it is disabled by default.

5.5.2 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 DOS don't contain the needful programs to handle compressed archives. Get gzip and unzip for Windows and for DOS at `http://infozip.sourceforge.net`. Don't use pkunzip for DOS, it doesn't work. The programs to use BZip2 archives may be found at `https://sourceware.org/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).

5.5.3 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` 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 `-unzip` options of `c1541` (see Section 14.3 [c1541 commands and options], page 225); 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 file 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 the *options* menu, via command-line options, using monitor commands, or via the *resource file*.

The *resource file* is a human-readable file containing resource values: On windows systems it is called `vice.ini` and is usually stored in the users roaming profile directory (something like `C:\Users\<username>\AppData\Roaming\vice`). Additionally the emulators will look in the program directory at startup, and use `vice.ini` if it exists there. On unix systems the resource file is called `vicerc` and is stored in the users config directory according to the XDG specification (usually `/home/<username>/.config/vice/`). Additionally the emulators will look if `.vicerc` exists in the users home directory, and use that if so.

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 save the current (changed) settings before exiting, and can be toggled with the "Save settings on exit" item from the options menu.

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]
SaveResourcesOnExit=0
FileSystemDevice8=1
FSDevice8ConvertP00=1
FSDevice8Dir="/home/ettore/cbm/stuff/vic20p00"
FSDevice8SaveP00=1
FSDevice8HideCBMFiles=1
[C64]
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, `-warp` enables warp mode, while `+warp` disables it.

6.3 Autostart settings

6.3.1 Autostart resources

All these resources are available for all emulators except vsid.

`AutostartPrgDiskImage`
: String specifying the filename of the disk image used when autostarting a prg file and "copy to D64" is enabled (all emulators except vsid).

`AutostartBasicLoad`
: Boolean, if true load to basic start using ,8 when autostarting from disk, else use ,8,1 to load absolute. (all emulators except vsid).

`AutostartTapeBasicLoad`
: Boolean, if true load to basic start using ,1 when autostarting from tape, else use ,1,1 to load absolute. (all emulators except vsid).

`AutostartRunWithColon`
: Boolean, if true put a colon after the load command when autostarting (all emulators except vsid).

`AutostartHandleTrueDriveEmulation`
: Boolean, if true handle (enable/disable) True Drive Emulation on autostart (all emulators except vsid). This is disabled by default to ensure maximum compatibility.

`AutostartWarp`
: Boolean, if true temporarily enable warp mode when autostarting (all emulators except vsid).

`AutostartPrgMode`
: Integer specifying the autostart mode for prg files (all emulators except vsid). (0: virtual filesystem, 1: inject to RAM, 2: copy to D64)

`AutostartDelayRandom`
: Boolean, enables a short (0-10 frames) random delay on autostart. This is added to the `AutostartDelay` (all emulators except vsid).

`AutostartDelay`
: Integer specifying the delay in seconds required to wait for the kernal reset routine before autostart. (0: use builtin value for standard kernal) When tweaking this value start with 'large' values and then lower it, a value that is too small results in autostart not happening. (all emulators except vsid). (0..1000)

6.3.2 Autostart command-line options

All these command-line options are available for all emulators except vsid.

`-autostartprgdiskimage <Name>`
: Set disk image for autostart of PRG files (`AutostartPrgDiskImage`) (all emulators except vsid).

`-basicload`
: On autostart from disk, load to BASIC start (without ',1') (`AutostartBasicLoad=1`) (all emulators except vsid).

`+basicload`
: On autostart from disk, load with ',1' (`AutostartBasicLoad=0`) (all emulators except vsid).

`-tapebasicload`
: On autostart from tape, load to BASIC start (without ',1') (`AutostartTapeBasicLoad=1`) (all emulators except vsid).

`+tapebasicload`
: On autostart from tape, load with ',1' (`AutostartTapeBasicLoad=0`) (all emulators except vsid).

`-autostartwithcolon`
: On autostart, use the 'RUN' command with a colon, i.e., 'RUN:' (`AutostartRunWithColon=1`). (all emulators except vsid)

`+autostartwithcolon`
: On autostart, do not use the 'RUN' command with a colon; i.e., 'RUN' (`AutostartRunWithColon=0`) (all emulators except vsid).

`-autostart-handle-tde`
`+autostart-handle-tde`
: Handle/Do not handle True Drive Emulation on autostart (`AutostartHandleTrueDriveEmulation=1`, `AutostartHandleTrueDriveEmulation=0`) (all emulators except vsid).

`-autostart-warp`
`+autostart-warp`
: Enable/disable warp mode during autostart (`AutostartWarp=1`, `AutostartWarp=0`) (all emulators except vsid).

`-autostartprgmode <Mode>`
: Set autostart mode for PRG files (`AutostartPrgMode`) (all emulators except vsid). (0: virtual filesystem, 1: inject to RAM, 2: copy to D64)

`-autostart-delay-random`
`+autostart-delay-random`
: Enable/disable random delay on autostart (`AutostartDelayRandom`) (all emulators except vsid).

`-autostart-delay <seconds>`
: Set initial autostart delay in seconds for the kernal reset routine before autostart. (0: use builtin value for standard kernal). When tweaking this value start with 'large' values and then lower it, a value that is too small results in autostart not happening. (`AutostartDelay`) (all emulators except vsid). (0..1000)

6.4 Performance settings

It is possible to control the emulation speed by using the "Maximum speed" item in the speed setting. 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. A specific FPS may also be targeted.

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.

6.4.1 Performance resources

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

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

6.4.2 Performance command-line options

`-speed <value>`
: A positive <value> specifies the maximum speed as a percentage. A negative <value> results in a target FPS of 0 - <value> (`Speed`).

`-warp`
`+warp` Enable/Disable warp mode (`WarpMode=1`, `WarpMode=0`).

6.5 Video settings

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

- "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. It also selects a more accurate CRT emulation renderer with scanlines.
- "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).

6.5.1 Video resources

The following resources affect the screen emulation.

`HwScalePossible`
: Boolean that indicates whether hardware scaling is possible or not.

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

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

`FullscreenEnable`
: Boolean specifying whether "fullscreen" is turned on at startup or not.

6.5.2 Video command line options

`-hwscalepossible`
`+hwscalepossible`
: Enable/Disable the possibility of hardware scaling `HwScalePossible=1` or `HwScalePossible=1`).

6.6 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 2.8 [Keyboard emulation], page 10, for an explanation of positional and symbolic mappings).

The keyboard settings 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").

6.6.1 Keyboard resources

`KeymapIndex`
: Integer identifying which keymap is being used. (0: symbolic, 1: positional, 2: symbolic (user), 3: positional (user)).

`KeymapSymFile`
: String specifying the name of the keymap file for the symbolic mapping (see Section 2.8 [Keyboard emulation], page 10, (this will be set indirectly by changing `KeymapIndex`, `KeyboardMapping` or `KeyboardType`).

`KeymapPosFile`
: String specifying the name of the keymap file for the positional mapping (see Section 2.8 [Keyboard emulation], page 10, (this will be set indirectly by changing `KeymapIndex`, `KeyboardMapping` or `KeyboardType`).

`KeymapUserSymFile`
: String specifying the name of the user keymap file for symbolic mapping (see Section 2.8 [Keyboard emulation], page 10,

`KeymapUserPosFile`
: String specifying the name of the user keymap file for positional mapping (see Section 2.8 [Keyboard emulation], page 10,

`KeyboardMapping`
: Integer specifying the keyboard layout of the host. (0: American 1: British 2: German 3: Danish 4: Norwegian 5: Finnish 6: Italian).

`KeyboardType`
: Integer specifying the emulated type of keyboard. (machine specific, currently always 0 except for xpet: 0: Business (us) 1: Business (uk) 2: Business (de) 3: Business (jp) 4: Graphics (us)).

6.6.2 Keyboard command-line options

`-keymap <number>`
: Specifies which keymap is being used (`KeymapIndex` 0: symbolic, 1: positional, 2: symbolic (user), 3: positional (user)).

`-symkeymap <Name>`
: Specify filename of the symbolic user keymap file (`KeymapUserSymFile`).

`-poskeymap <Name>`
: Specify filename of the positional user keymap file (`KeymapUserPosFile`).

`-keyboardmapping <number>`
: Specifies the keyboard layout of the host (`KeyboardMapping` 0: American 1: British 2: German 3: Danish 4: Norwegian 5: Finnish 6: Italian).

`-keyboardtype <number>`
: Specifies the emulated type of keyboard (`KeyboardType` machine specific, currently always 0 except for xpet: 0: Business (us) 1: Business (uk) 2: Business (de) 3: Business (jp) 4: Graphics (us)).

The control port settings submenu allows you to select which control port device is connected to a control port.

6.6.3 Control port resources

`JoyPort1Device`
: Set the device attached to: control port 1 (x64, x64sc, x64dtv, xscpu64, x128, xcbm5x0, xplus4) control port (xvic). (x64, x64sc, xscpu, x128, xcbm5x0, xvic: 0: None, 1: Joystick, 2: Paddles, 3: Mouse (1351), 4: Mouse (NEOS), 5: Mouse (Amiga), 6: Mouse (CX-22), 7: Mouse (Atari ST), 8: Mouse (SmartMouse), 9: Mouse (Micromys), 10: KoalaPad, 11: Light Pen (up trigger), 12: Light Pen (left trigger), 13: Datel Light Pen, 14: Magnum Light Phaser, 15: Stack Light Rifle, 16: Inkwell Light Pen, 17: Sampler (2bit)) (x64dtv, xplus4: 0: None, 1: Joystick, 6: Mouse (CX-22), 17: Sampler (2bit))

`JoyPort2Device`
: Set the device attached to: control port 2 (x64, x64sc, x64dtv, xscpu64, x128, xcbm5x0, xplus4) (x64, x64sc, xscpu, x128, xcbm5x0: 0: None, 1: Joystick, 2: Paddles, 3: Mouse (1351), 4: Mouse (NEOS), 5: Mouse (Amiga), 6: Mouse (CX-22), 7: Mouse (Atari ST), 8: Mouse (SmartMouse), 9: Mouse (Micromys), 10: KoalaPad, 17: Sampler (2bit)) (x64dtv, xplus4: 0: None, 1: Joystick, 6: Mouse (CX-22), 17: Sampler (2bit))

`JoyPort3Device`
: Set the device attached to: userport joystick adapter port 1 (x64, x64sc, xscpu64, x128, xcbm2, xpet, xvic) userport joystick adapter port (x64dtv) sidcart control port (xplus4). (x64, x64sc, x64dtv, xscpu, x128, xcbm2, xpet, xvic: 0: None, 1: Joystick, 6: Mouse (CX-22), 17: Sampler (2bit)) (xplus4: 0: None, 1: Joystick, 2: Paddles, 3: Mouse (1351), 4: Mouse (NEOS), 5: Mouse (Amiga), 6: Mouse (CX-22), 7: Mouse (Atari ST), 8: Mouse (SmartMouse), 9: Mouse (Micromys), 10: KoalaPad, 17: Sampler (2bit))

`JoyPort4Device`
: Set the device attached to: userport joystick adapter port 2 (x64, x64sc, xscpu64, x128, xcbm2, xpet, xvic) (x64, x64sc, xscpu, x128, xcbm2, xpet, xvic: 0: None, 1: Joystick, 6: Mouse (CX-22), 17: Sampler (2bit))

`JoyPort5Device`
: Set the device attached to: SidCart on the Plus4 (xplus4)

`BBRTCSave`
: Enable saving of the battery-backed real time clock data.

6.6.4 Control port command-line options

`-controlport1device <device>`
: Set the device attached to: control port 1 (x64, x64sc, x64dtv, xscpu, x128, xcbm5x0, xplus4) control port (xvic) (`JoyPort1Device`). (x64, x64sc, xscpu, x128, xcbm5x0, xvic: 0: None, 1: Joystick, 2: Paddles, 3: Mouse (1351), 4: Mouse (NEOS), 5: Mouse (Amiga), 6: Mouse (CX-22), 7: Mouse (Atari ST), 8: Mouse (SmartMouse), 9: Mouse (Micromys), 10: KoalaPad, 11: Light Pen (up trigger), 12: Light Pen (left trigger), 13: Datel Light Pen, 14: Magnum Light Phaser, 15: Stack Light Rifle, 16: Inkwell Light Pen, 17: Sampler (2bit)) (x64dtv, xplus4: 0: None, 1: Joystick, 6: Mouse (CX-22), 17: Sampler (2bit))

`-controlport2device <device>`
: Set the device attached to: control port 2 (x64, x64sc, x64dtv, xscpu, x128, xcbm5x0, xplus4) (`JoyPort2Device`). (x64, x64sc, xscpu, x128, xcbm5x0: 0: None, 1: Joystick, 2: Paddles, 3: Mouse (1351), 4: Mouse (NEOS), 5: Mouse (Amiga), 6: Mouse (CX-22), 7: Mouse (Atari ST), 8: Mouse (SmartMouse), 9: Mouse (Micromys), 10: KoalaPad, 17: Sampler (2bit)) (x64dtv, xplus4: 0: None, 1: Joystick, 6: Mouse (CX-22), 17: Sampler (2bit))

`-controlport3device <device>`
: Set the device attached to: userport joystick adapter port 1 (x64, x64sc, xscpu, x128, xcbm2, xpet, xvic) userport joystick adapter port (x64dtv) sidcart control port (xplus4) (`JoyPort3Device`). (x64, x64sc, x64dtv, xscpu, x128, xcbm2, xpet, xvic: 0: None, 1: Joystick, 6: Mouse (CX-22), 17: Sampler (2bit)) (xplus4: 0: None, 1: Joystick, 2: Paddles, 3: Mouse (1351), 4: Mouse (NEOS), 5: Mouse (Amiga), 6: Mouse (CX-22), 7: Mouse (Atari ST), 8: Mouse (SmartMouse), 9: Mouse (Micromys), 10: KoalaPad, 17: Sampler (2bit))

`-controlport4device <device>`
: Set the device attached to: userport joystick adapter port 2 (x64, x64sc, xscpu, x128, xcbm2, xpet, xvic) (`JoyPort4Device`). (x64, x64sc, xscpu, x128, xcbm2, xpet, xvic: 0: None, 1: Joystick, 6: Mouse (CX-22), 17: Sampler (2bit))

`-controlport5device <device>`
: Set the device attached to: SID cart joystick port (xplus4) (`JoyPort5Device`). (xplus4: 0: None, 1: Joystick, 2: Paddles, 3: Mouse (1351), 4: Mouse (NEOS), 5: Mouse (Amiga), 6: Mouse (CX-22), 7: Mouse (Atari ST), 8: Mouse (SmartMouse), 9: Mouse (Micromys), 10: KoalaPad, 17: Sampler (2bit), 18: Sampler (4bit), 19: BBRTC, 20: Paperclip64 dongle, 21: Coplin Keypad, 22: Cardco Cardkey 1 keypad, 23: Atari CX85 keypad, 24: RushWare Keypad, 25: Atari CX21 keypad)

`-bbrtcsave`
`+bbrtcsave`
: Enable/Disable saving of the battery-backed real time clock data (`BBRTCSave`).

6.7 Joystick settings

6.7.1 Joystick resources

`JoyDevice1`
`JoyDevice2`
`JoyDevice3`
`JoyDevice4`
`JoyDevice5`
: Integer specifying which joystick device the emulator should use for joystick emulation for ports 1 - 5, respectively. (0=None, 1=Joystick 1, 2=Joystick 2, 4=Numpad, 8=Keyset 1, 16=Keyset 2 on OS/2) (0=None, 1=Numpad, 2=Keyset 1, 3=Keyset 2, 4=HID joystick 0, 5=HID joystick 1 on Mac OS X) The available joysticks might differ depending on operating system and joystick support in the OS (Linux joystick module must be available for example).

`JoyOpposite`
: Boolean, if true allow (usually impossible) bitcombinations for opposite directions. (all emulators except vsid)

`UserportJoy`
: Boolean to enable/disable extra joysticks (all emulators except xcbm5x0 and vsid).

`UserportJoyType`
: Integer specifying the type of adapter used for the extra joysticks (all emulators except xcbm5x0 and vsid). (0: Classical Games/Protovision, 1: PET, 2: Hummer, 3: OEM, 4: Digital Excess/Hitmen, 5: Kingsoft, 6: Starbyte) 4, 5 and 6 are x64, x64sc, xscpu64 and x128 only.

`Mouse` Boolean, enables mouse emulation

`SmartMouseRTCSave`
: Boolean, specified whether to save real time clock data for the SmartMouse

`KeySet1NorthWest`
`KeySet1North`
`KeySet1NorthEast`
`KeySet1East`
`KeySet1SouthEast`
`KeySet1South`
`KeySet1SouthWest`
`KeySet1West`
`KeySet1Fire`
`KeySet1Fire2`
`KeySet1Fire3`
: Integers specifying the keycodes for keyset 1 (all emulators except vsid).

`KeySet2NorthWest`
`KeySet2North`
`KeySet2NorthEast`
`KeySet2East`
`KeySet2SouthEast`
`KeySet2South`
`KeySet2SouthWest`
`KeySet2West`
`KeySet2Fire`
`KeySet2Fire2`
`KeySet2Fire3`
: Integers specifying the keycodes for keyset 2 (all emulators except vsid).

`KeySetEnable`
: Boolean that specifies whether user defined keysets are enabled (all emulators except vsid).

`KbdbufDelay`
: Integer specifying the additional keyboard delay. (0: use default)

6.7.2 Joystick command-line options

`-joydev1 <range>`
`-joydev2 <range>`
: Set the device for joystick emulation of port 1 and 2, respectively. (`JoyDevice1`, `JoyDevice2`). The range for OS/2 is, valid numbers in the range are 0, 1, 2, 4, 8 and 16.

`-extrajoydev1 <0-8>`
`-extrajoydev2 <0-8>`
`-extrajoydev3 <0-8>`
: Set device for extra joystick port 1, 2 and 3.

`-joyopposite`
`+joyopposite`
: Enable/disable opposite joystick directions (`JoyOpposite=1`, `JoyOpposite=0`). (all emulators except vsid)

`-userportjoy`
`+userportjoy`
: Enable/disable extra joystick(s) (`UserportJoy=1`, `UserportJoy=0`). (all emulators except xcbm5x0 and vsid).

`-userportjoytype <Type>`
: Set extra joystick type (`UserportJoyType`) (all emulators except xcbm5x0 and vsid). (0: Classical Games/Protovision, 1: PET, 2: Hummer, 3: OEM, 4: Digital Excess/Hitmen, 5: Kingsoft, 6: Starbyte) 4, 5 and 6 are x64, x64sc, xscpu64 and x128 only.

`-mouse`
`+mouse` Enable/Disable mouse grab

`-smartmousertcsave`
`+smartmousertcsave`
: Enable/Disable saving of the real time clock data for the SmartMouse

`-keyset`
`+keyset` Enable/disable user defined keyset (`KeySetEnable=1`, `KeySetEnable=0`) (all emulators except vsid).

`-keybuf-delay <value>`
: Set additional keyboard buffer delay (`KbdbufDelay`). (0: use default)

6.8 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.

  The following menu items control sound input:
- "Sampler Device" specifies the host device/method used for sampling/sound input, currently portaudio (if linked in) and file methods are supported.
- "Sampler Gain" specifies the amount of gain (increase the input volume if above 100 or decrease the input volume if below 100) for the input of the sampler device.
- "Sampler File" specifies the name of the file to be uses as an input source for the 'file' method/device.

6.8.1 Sound resources

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

`SoundSpeedAdjustment`
: Integer specifying what speed adjustment method the audio renderer should use. (0: flexible, 1: adjusting, 2: exact)

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

`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.

`SoundDeviceName`
: String specifying the audio driver.

  Implemented drivers are:

  - `ahi`, for the Amiga/Morphos/Aros sound driver.
  - `aix`, for the IBM AIX sound driver.

- `allegro`, for the DOS Allegro sound driver.
- `alsa`, for the linux ALSA sound driver.
- `arts`, for the *nix ARTS sound driver.
- `beos`, for the BeOS/Zeta/Haiku sound driver.
- `bsp`, for the BeOS/Zeta/Haiku BeOS Media Kit sound driver.
- `coreaudio`, for the Mac OS X sound driver (`SoundDeviceArg` specifies the audio device, default system output by default).
- `dart`, for the OS/2 sound driver.
- `dummy`, fully emulating the sound output chip(s), but not actually playing samples.
- `dx`, for the Windows Direct-X sound driver.
- `hpux`, for the HP-UX audio device (unfinished; `SoundDeviceArg` specifies the audio device, `/dev/audio` by default).
- `midas`, for the DOS Midas sound driver.
- `pulse`, for the Pulseaudio sound driver.
- `sdl`, for the Simple DirectMedia Layer audio driver.
- `sgi`, for the Silicon Graphics audio device (`SoundDeviceArg` specifies the audio device, `/dev/audio` by default);
- `speed`, like `dummy` but also calculating samples (mainly used to evaluate the speed of the sample generator);
- `sun`, for the Solaris and NetBDS audio device (unfinished; `SoundDeviceArg` specifies the audio device, `/dev/audio` by default).
- `uss`, for the Linux/FreeBSD Universal Sound System driver (`SoundDeviceArg` specifies the audio device, `/dev/dsp` by default);
- `wmm`, for the Windows Multimedia Waveout sound device.

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

`SoundDeviceArg`

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

`SoundRecordDeviceName`

String specifying the driver used for sound recording.

Implemented drivers are:

- `aiff`, for the Apple Interchange File Format 16bit sound recorder driver.
- `dump`, writing all the write accesses to the registers to a file (specified by `SoundDeviceArg`, default value is `vicesnd.sid`);
- `fs`, writing samples to a file (specified by `SoundDeviceArg`; default is `vicesnd.raw`); `iff`, for the Amiga Interchange File Format (8SVX) 8bit sound recorder driver.
- `mp3`, for the MP3 sound recorder driver.
- `flac`, for the FLAC sound recorder driver.

- `ogg`, for the ogg/vorbis sound recorder driver.
- `voc`, for the Creative Voice (VOC) sound recorder driver.
- `wav`, for the RIFF/WAV sound recorder driver.

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

`SoundRecordDeviceArg`
String specifying additional arguments for sound recording.

`SoundFragmentSize`
Integer specifying the fragment size. (0: very small, 1: small, 2: medium, 3: large, 4: very large)

`SoundVolume`
Integer specifying the master volume in percent. (0..100)

`SoundOutput`
Integer specifying the type of sound output. Output is selectable between 'system' (system decides to use mono or stereo output based on the presence of a stereo sid), 'always mono' (output is always mono, stereo streams are mixed into a mono stream) or 'always stereo' (output is always stereo, mono streams are multiplexed to a stereo stream). (0: system, 1: mono, 2: stereo)

`SamplerDevice`
Integer specifying the device/method to be used for sound input. (0: sample file device, 1: PortAudio device)

`SamplerGain`
Integer specifying the gain to be used for sound input. (>100 increase input volume, <100: decrease input volume)

`SampleName`
String specifying the name of the file/sample to be used as the input source for the 'file' sampler device.

6.8.2 Sound command-line options

`-sound`
`+sound` Enable/disable sound emulation (`Sound=1`, `Sound=0`).

`-soundrate <value>`
Specify the sound playback sample rate (`SoundSampleRate`). (8000..48000)

`-soundoutput <output mode>`
Sound output mode (`SoundOutput`). (0: system decides mono/stereo, 1: always mono, 2: always stereo)

`-soundbufsize <value>`
Specify the size of the audio buffer in milliseconds (`SoundBufferSize`).

`-soundfragsize <value>`
Set sound fragment size (`SoundFragmentSize`). (0: very small, 1: small, 2: medium, 3: large, 4: very large)

`-sounddev <Name>`
: Specifies the name of the audio device (`SoundDeviceName`). (ahi, aix, allegro, alsa, arts, beos, bsp, coreaudio, dart, dummy, dx, hpux, midas, pulse, sdl, sgi, sun, uss, wmm)

`-soundarg <args>`
: Specifies an additional parameter for the audio device (`SoundDeviceArg`).

`-soundrecdev <name>`
: Specify recording sound driver (`SoundRecordDeviceName`). (aiff, dump, fs, iff, mp3, flac, ogg, speed, voc, wav)

`-soundrecarg <args>`
: Specify initialization parameters for recording sound driver (`SoundRecordDeviceArg`).

`-soundsuspend <seconds>`
: Specify the pause interval when audio underflows (clicks) happen. 0 means no pause is done (`SoundSuspendTime`).

`-soundvolume <volume>`
: Specify the sound volume (`SoundVolume`). (0..100)

`-samplerdev <device number>`
: Specify the device to use for audio input (`SamplerDevice`). (0: file device, 1: portaudio device)

`-samplergain <percent>`
: Specify the amount of gain (volume increase/decrease) for the audio input device (`SamplerGain`). (0..200)

`-samplename <name>`
: Specify the name of the file to use for the 'file' audio input device (`SampleFile`).

6.9 Tape settings

These settings are used to control the hardware-level emulation of the Tape drive.

6.9.1 Tape resources

`Datasette`
: Boolean specifying whether to emulate the datasette.

`DatasetteResetWithCPU`
: Boolean specifying whether to reset (rewind) the tape when resetting the CPU.

`DatasetteZeroGapDelay`
: Integer specifying the delay in cycles for a zero in the tap.

`DatasetteSpeedTuning`
: Integer specifying the number of cycles added to each gap in the tap.

`DatasetteTapeWobble`
: Integer specifying the maximum random number of cycles added to each gap in the tap.

`DatasetteSound`
: Boolean specifying whether to produce audible sound when playing a tape on the datasette

`DatasetteSoundVolume`
: Integer specifying the volume of the tape sound. Meaningful values are in the range 1-32767

6.9.2 Tape command-line options

`-datasette`
`+datasette`
: Enable/disable datasette emulation (`Datasette=1`, `Datasette=0`).

`-dsresetwithcpu`
`+dsresetwithcpu`
: Enable/disable automatic Datasette-Reset (`DatasetteResetWithCPU=1`, `DatasetteResetWithCPU=0`).

`-dszerogapdelay <value>`
: Set delay in cycles for a zero in the tap (`DatasetteZeroGapDelay`).

`-dsspeedtuning <value>`
: Set number of cycles added to each gap in the tap (`DatasetteSpeedTuning`).

`-dstapewobble <value>`
: Set maximum random number of cycles added to each gap in the tap (`DatasetteTapeWobble`).

`-datasettesound`
`+datasettesound`
: Enable/disable Datasette sound emulation (`DatasetteSound=1`, `DatasetteSound=0`).

`-dssoundvolume <value>`
: Set the volume of the Datasette sound (`DatasetteSoundVolume`).

6.10 Drive settings

These settings are used to control the hardware-level emulation of the Disk drives. 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.

6.10.1 Drive resources

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

`DriveSoundEmulation`
: Boolean controlling whether the drive noise emulation is turned on (all emulators except vsid).

`DriveSoundEmulationVolume`
: Integer specifying the volume of the drive noise emulation (all emulators except vsid). (0..4000)

`Drive8Type`
`Drive9Type`
`Drive10Type`
`Drive11Type`
: Integers specifying the model number for drives 8 to 11. Possible values are `1541` [all emulators except xcbm2, xcbm5x0, xpet and vsid], `1542` (1541-II) [all emulators except xcbm2, xcbm5x0, xpet and vsid], `1570` [all emulators except xcbm2, xcbm5x0, xpet and vsid], `1571` [all emulators except xcbm2, xcbm5x0, xpet and vsid], `1573` (1571CR) [x128 only], `1551` [xplus4 only], `1581` [all emulators except xcbm2, xcbm5x0, xpet and vsid], `1001` [all emulators except x64dtv, xplus4 and vsid], `2000` [all emulators except xcbm2, xcbm5x0, xpet and vsid], `2031` [all emulators except x64dtv, xplus4 and vsid], `2040` [all emulators except x64dtv, xplus4 and vsid], `3040` [all emulators except x64dtv, xplus4 and vsid], `4000` [all emulators except xcbm2, xcbm5x0, xpet and vsid], `4040` [all emulators except x64dtv, xplus4 and vsid], `4844` (CMD HD) [all emulators except xcbm2, xcbm5x0, xpet and vsid], `8050` [all emulators except x64dtv, xplus4 and vsid], `8250` [all emulators except x64dtv, xplus4 and vsid]. `9000` [all emulators except x64dtv, xplus4 and vsid].

`Drive8RTCSave`
: Integer specifying whether the RTC data of drive 8 should be saved when changed or not (drive type 2000/4000 only).

`Drive9RTCSave`
: Integer specifying whether the RTC data of drive 9 should be saved when changed or not (drive type 2000/4000 only).

`Drive10RTCSave`
: Integer specifying whether the RTC data of drive 10 should be saved when changed or not (drive type 2000/4000 only).

`Drive11RTCSave`
: Integer specifying whether the RTC data of drive 11 should be saved when changed or not (drive type 2000/4000 only).

`Drive8ParallelCable`
`Drive9ParallelCable`
`Drive10ParallelCable`
`Drive11ParallelCable`

integers controlling what type of parallel cable is emulated for drives 8 to 11 (x64, x64sc, xscpu64, x128 and xplus4 only). x64, x64sc, xscpu64, x128: (0: None, 1: Standard, 2: Dolphin DOS 3, 3: Formel64) xplus4: (0: None, 1: Standard)

`Drive8ProfDOS`
`Drive9ProfDOS`
`Drive10ProfDOS`
`Drive11ProfDOS`

Booleans controlling whether Professional DOS is emulated or not for drives 8 to 11 (x64, x64sc, xscpu64 and x128 only).

`Drive8SuperCard`
`Drive9SuperCard`
`Drive10SuperCard`
`Drive11SuperCard`

Booleans controlling whether Supercard is emulated or not for drives 8 to 11 (x64, x64sc, xscpu64 and x128 only).

`Drive8StarDos`
`Drive9StarDos`
`Drive10StarDos`
`Drive11StarDos`

Booleans controlling whether StarDOS is emulated or not for drives 8 to 11 (x64, x64sc, xscpu64 and x128 only).

`Drive8RAM2000`
`Drive8RAM4000`
`Drive8RAM6000`
`Drive8RAM8000`
`Drive8RAMA000`
`Drive9RAM2000`
`Drive9RAM4000`
`Drive9RAM6000`
`Drive9RAM8000`
`Drive9RAMA000`
`Drive10RAM2000`
`Drive10RAM4000`
`Drive10RAM6000`
`Drive10RAM8000`
`Drive10RAMA000`
`Drive11RAM2000`
`Drive11RAM4000`
`Drive11RAM6000`
`Drive11RAM8000`
`Drive11RAMA000`

Booleans controlling whether a RAM block is emulated at the respective block or not for drives 8 to 11 respectively.

`Drive8ExtendImagePolicy`
`Drive9ExtendImagePolicy`
`Drive10ExtendImagePolicy`
`Drive11ExtendImagePolicy`

Integer specifying the policy for 40-track support for drives 8 to 11. (0: never extend, 1: ask on extend, 2: extend on access)

`Drive8IdleMethod`
`Drive9IdleMethod`
`Drive10IdleMethod`
`Drive11IdleMethod`

Integers specifying the idling method for the drive CPU. See Section 6.10 [Drive settings], page 44. (0: none, 1: skip cycles, 2: trap idle)

`Drive8RPM`
`Drive9RPM`
`Drive10RPM`
`Drive11RPM`

Integers specifying the rotation speed of the drive, multiplied by 100, so 300rpm equals 30000.

`Drive8WobbleFrequency`
`Drive9WobbleFrequency`
`Drive10WobbleFrequency`
`Drive11WobbleFrequency`

Integers specifying the drive wobble frequency

`Drive8WobbleAmplitude`
`Drive9WobbleAmplitude`
`Drive10WobbleAmplitude`
`Drive11WobbleAmplitude`
: Integers specifying the drive wobble amplitude

`DosName1540`
`DosName1541`
`DosName1541ii`
`DosName1570`
`DosName1571`
`DosName1581`
`DosName2000`
`DosName4000`
`DosNameCMDHD`
: Strings specifying the names of the ROM images for the drive emulation. (all emulators except xcbm2, xcbm5x0, xpet and vsid)

`DosName1551`
: String specifying the name of the ROM image for the drive emulation. (xplus4 only)

`DosName1571cr`
: String specifying the name of the ROM image for the drive emulation. (x128 only)

`DosName2031`
`DosName2040`
`DosName3040`
`DosName4040`
`DosName1001`
`DosName9000`
: Strings specifying the names of the ROM images for the drive emulation. (all emulators except x64dtv, xplus4 and vsid)

`DriveProfDOS1571Name`
: String specifying the filename of the 1571 professional DOS ROM image (x64, x64sc, xscpu64 and x128 only).

`DriveSuperCardName`
: String specifying the filename of the Supercard ROM image (x64, x64sc, xscpu64 and x128 only).

`DriveStarDosName`
: String specifying the filename of the image of the lower half of the StarDOS ROM (x64, x64sc, xscpu64 and x128 only).

6.10.2 Drive command-line options

`-truedrive`
`+truedrive`
: Enable/disable true drive emulation (`DriveTrueEmulation=1`, `DriveTrueEmulation=0`).

`-drivesound`
`+drivesound`
: Enable/disable drive sound emulation (`DriveSoundEmulation=1`, `DriveSoundEmulation=0`) (all emulators except vsid).

`-drivesoundvolume <Volume>`
: Set the volume of the drive sound emulation (`DriveSoundEmulationVolume=1`, `DriveSoundEmulationVolume=0`) (all emulators except vsid).

`-drive8type <Type>`
`-drive9type <Type>`
`-drive10type <Type>`
`-drive11type <Type>`
: Specifies the drive types for drives 8-11, respectively. Possible values for `TYPE` are `1541` [all emulators except xcbm2, xcbm5x0, xpet and vsid], `1542` (meaning 1541-II) [all emulators except xcbm2, xcbm5x0, xpet and vsid], `1551` [xplus4 only], `1570` [all emulators except xcbm2, xcbm5x0, xpet and vsid], `1571` [all emulators except xcbm2, xcbm5x0, xpet and vsid], `1573` (meaning 1571cr) [x128 only], `1581` [all emulators except xcbm2, xcbm5x0, xpet and vsid], `2000` [all emulators except xcbm2, xcbm5x0, xpet and vsid], `4000` [all emulators except xcbm2, xcbm5x0, xpet and vsid], `4844` (meaning CMD HD) [all emulators except xcbm2, xcbm5x0, xpet and vsid], `2031` [all emulators except x64dtv, xplus4 and vsid], `2040` [all emulators except x64dtv, xplus4 and vsid], `3040` [all emulators except x64dtv, xplus4 and vsid], `4040` [all emulators except x64dtv, xplus4 and vsid], `1001` [all emulators except x64dtv, xplus4 and vsid], `8050` [all emulators except x64dtv, xplus4 and vsid] `8250` [all emulators except x64dtv, xplus4 and vsid]. and `9000` [all emulators except x64dtv, xplus4 and vsid].

`-drive8rtcsave`
`-drive8rtcsave`
: Enable/disable the saving of the RTC data for drive 8 when changed (drive type 2000/4000/4844 only) (`Drive8RTCSave=1`, `Drive8RTCSave=0`).

`-drive9rtcsave`
`-drive9rtcsave`
: Enable/disable the saving of the RTC data for drive 9 when changed (drive type 2000/4000/4844 only) (`Drive9RTCSave=1`, `Drive9RTCSave=0`).

`-drive10rtcsave`
`-drive10rtcsave`
: Enable/disable the saving of the RTC data for drive 10 when changed (drive type 2000/4000/4844 only) (`Drive10RTCSave=1`, `Drive10RTCSave=0`).

`-drive11rtcsave`
`-drive11rtcsave`
: Enable/disable the saving of the RTC data for drive 11 when changed (drive type 2000/4000/4844 only) (`Drive11RTCSave=1`, `Drive11RTCSave=0`).

`-parallel8 <type>`
`-parallel9 <type>`
`-parallel10 <type>`
`-parallel11 <type>`
: Set parallel cable type for drives 8-11 respectively (`Drive8ParallelCable`, `Drive9ParallelCable`, `Drive10ParallelCable`, `Drive11ParallelCable`) (x64, x64sc, xscpu64, x128 and xplus4 only). x64, x64sc, xscpu64, x128: (0: None, 1: Standard, 2: Professional DOS, 3: Formel64) xplus4: (0: None, 1: Standard)

`-drive8idle <method>`
`-drive9idle <method>`
`-drive10idle <method>`
`-drive11idle <method>`
: Specifies <method> as the idling method for drives 8-11 respectively (`Drive8IdleMethod`, `Drive9IdleMethod`, `Drive10IdleMethod`), `Drive11IdleMethod`). (0: none, 1: skip cycles, 2: trap idle)

`-drive8extend <method>`
`-drive9extend <method>`
`-drive10extend <method>`
`-drive11extend <method>`
: Specifies <method> as the track 40 extend policy in drives 8-11 respectively (`Drive8ExtendImagePolicy`, `Drive9ExtendImagePolicy`, `Drive10ExtendImagePolicy`, `Drive11ExtendImagePolicy`). (0: never extend, 1: ask on extend, 2: extend on access)

`-drive8rpm <rpm>`
`-drive9rpm <rpm>`
`-drive10rpm <rpm>`
`-drive11rpm <rpm>`
: Specifies the rotation speed of the drive, multiplied by 100 so 300rpm equal 30000.

`-drive8wobblefrequency <frequency>`
`-drive9wobblefrequency <frequency>`
`-drive10wobblefrequency <frequency>`
`-drive11wobblefrequency <frequency>`
: Specifies frequency of the drive wobble

`-drive8wobbleamplitude <amplitude>`
`-drive9wobbleamplitude <amplitude>`
`-drive10wobbleamplitude <amplitude>`
`-drive11wobbleamplitude <amplitude>`
: Specifies amplitude of the drive wobble

`-dos1540 <name>`
: Specify the ROM name for the 1540 emulation (`DosName1540`). (all emulators except xcbm2, xcbm5x0, xpet and vsid)

`-dos1541 <name>`
: Specify the ROM name for the 1541 emulation (`DosName1541`). (all emulators except xcbm2, xcbm5x0, xpet and vsid)

`-dos1541II <name>`
: Specify the ROM name for the 1541-II emulation (`DosName1541ii`). (all emulators except xcbm2, xcbm5x0, xpet and vsid)

`-dos1551 <name>`
: Specify the ROM name for the 1551 emulation (`DosName1551`). (xplus4 only)

`-dos1570 <name>`
: Specify the ROM name for the 1570 emulation (`DosName1570`). (all emulators except xcbm2, xcbm5x0, xpet and vsid)

`-dos1571 <name>`
: Specify the ROM name for the 1571 emulation (`DosName1571`). (all emulators except xcbm2, xcbm5x0, xpet and vsid)

`-dos1571cr <name>`
: Specify the ROM name for the 1571 emulation (`DosName1571CR`). (x128 only)

`-dos1581 <name>`
: Specify the ROM name for the 1581 emulation (`DosName1581`). (all emulators except xcbm2, xcbm5x0, xpet and vsid)

`-dos2000 <name>`
: Specify the ROM name for the FD2000 emulation (`DosName2000`). (all emulators except xcbm2, xcbm5x0, xpet and vsid)

`-dos4000 <name>`
: Specify the ROM name for the FD4000 emulation (`DosName4000`). (all emulators except xcbm2, xcbm5x0, xpet and vsid)

`-dosCMDHD <name>`
: Specify the boot ROM name for the CMD HD emulation (`DosNameCMDHD`). (all emulators except xcbm2, xcbm5x0, xpet and vsid)

`-dos2031 <name>`
: Specify the ROM name for the 2031 emulation (`DosName2031`). (all emulators except x64dtv, xplus4 and vsid)

`-dos2040 <name>`
: Specify the ROM name for the 2040 emulation (`DosName2040`). (all emulators except x64dtv, xplus4 and vsid)

`-dos3040 <name>`
: Specify the ROM name for the 3040 emulation (`DosName3040`). (all emulators except x64dtv, xplus4 and vsid)

`-dos4040 <name>`
Specify the ROM name for the 4040 emulation (`DosName4040`). (all emulators except x64dtv, xplus4 and vsid)

`-dos1001 <name>`
Specify the ROM name for the 1001, 8050 and 8250 emulations (`DosName1001`). (all emulators except x64dtv, xplus4 and vsid)

`-dos9000 <name>`
Specify the ROM name for the D9090/60 emulations (`DosName9000`). (all emulators except x64dtv, xplus4 and vsid)

`-drive8ram2000, +drive8ram2000`
Enable/disable 8KiB RAM expansion at $2000-$3FFF for drive 8 (`Drive8RAM2000=1`, `Drive8RAM2000=0`).

`-drive9ram2000, +drive9ram2000`
Enable/disable 8KiB RAM expansion at $2000-$3FFF for drive 9 (`Drive9RAM2000=1`, `Drive9RAM2000=0`).

`-drive10ram2000, +drive10ram2000`
Enable/disable 8KiB RAM expansion at $2000-$3FFF for drive 10 (`Drive10RAM2000=1`, `Drive10RAM2000=0`).

`-drive11ram2000, +drive11ram2000`
Enable/disable 8KiB RAM expansion at $2000-$3FFF for drive 11 (`Drive11RAM2000=1`, `Drive11RAM2000=0`).

`-drive8ram4000, +drive8ram4000`
Enable/disable 8KiB RAM expansion at $4000-$5FFF for drive 8 (`Drive8RAM4000=1`, `Drive8RAM4000=0`).

`-drive9ram4000, +drive9ram4000`
Enable/disable 8KiB RAM expansion at $4000-$5FFF for drive 9 (`Drive9RAM4000=1`, `Drive9RAM4000=0`).

`-drive10ram4000, +drive10ram4000`
Enable/disable 8KiB RAM expansion at $4000-$5FFF for drive 10 (`Drive10RAM4000=1`, `Drive10RAM4000=0`).

`-drive11ram4000, +drive11ram4000`
Enable/disable 8KiB RAM expansion at $4000-$5FFF for drive 11 (`Drive11RAM4000=1`, `Drive11RAM4000=0`).

`-drive8ram6000, +drive8ram6000`
Enable/disable 8KiB RAM expansion at $6000-$7FFF for drive 8 (`Drive8RAM6000=1`, `Drive8RAM6000=0`).

`-drive9ram6000, +drive9ram6000`
Enable/disable 8KiB RAM expansion at $6000-$7FFF for drive 9 (`Drive9RAM6000=1`, `Drive9RAM6000=0`).

`-drive10ram6000, +drive10ram6000`
Enable/disable 8KiB RAM expansion at $6000-$7FFF for drive 10 (`Drive10RAM6000=1`, `Drive10RAM6000=0`).

`-drive11ram6000, +drive11ram6000`
Enable/disable 8KiB RAM expansion at $6000-$7FFF for drive 11 (`Drive11RAM6000=1`, `Drive11RAM6000=0`).

`-drive8ram8000, +drive8ram8000`
Enable/disable 8KiB RAM expansion at $8000-$9FFF for drive 8 (`Drive8RAM8000=1`, `Drive8RAM8000=0`).

`-drive9ram8000, +drive9ram8000`
Enable/disable 8KiB RAM expansion at $8000-$9FFF for drive 9 (`Drive9RAM8000=1`, `Drive9RAM8000=0`).

`-drive10ram8000, +drive10ram8000`
Enable/disable 8KiB RAM expansion at $8000-$9FFF for drive 10 (`Drive10RAM8000=1`, `Drive10RAM8000=0`).

`-drive11ram8000, +drive11ram8000`
Enable/disable 8KiB RAM expansion at $8000-$9FFF for drive 11 (`Drive11RAM8000=1`, `Drive11RAM8000=0`).

`-drive8rama000, +drive8rama000`
Enable/disable 8KiB RAM expansion at $A000-$BFFF for drive 8 (`Drive8RAMA000=1`, `Drive8RAMA000=0`).

`-drive9rama000, +drive9rama000`
Enable/disable 8KiB RAM expansion at $A000-$BFFF for drive 9 (`Drive9RAMA000=1`, `Drive9RAMA000=0`).

`-drive10rama000, +drive10rama000`
Enable/disable 8KiB RAM expansion at $A000-$BFFF for drive 10 (`Drive10RAMA000=1`, `Drive10RAMA000=0`).

`-drive11rama000, +drive11rama000`
Enable/disable 8KiB RAM expansion at $A000-$BFFF for drive 11 (`Drive11RAMA000=1`, `Drive11RAMA000=0`).

`-drive8profdos`
`+drive8profdos`
Enable/disable Professional DOS for drive 8 (`Drive8ProfDOS=1`, `Drive8ProfDOS=0`) (x64, x64sc, xscpu64 and x128 only).

`-drive9profdos`
`+drive9profdos`
Enable/disable Professional DOS for drive 9 (`Drive9ProfDOS=1`, `Drive9ProfDOS=0`) (x64, x64sc, xscpu64 and x128 only).

`-drive10profdos`
`+drive10profdos`
Enable/disable Professional DOS for drive 10 (`Drive10ProfDOS=1`, `Drive10ProfDOS=0`) (x64, x64sc, xscpu64 and x128 only).

`-drive11profdos`
`+drive11profdos`
Enable/disable Professional DOS for drive 11 (`Drive11ProfDOS=1`, `Drive11ProfDOS=0`) (x64, x64sc, xscpu64 and x128 only).

`-profdos1571 <name>`
: Specify name of Professional DOS 1571 ROM image (`DriveProfDOS1571Name`) (x64, x64sc, xscpu64 and x128).

`-drive8supercard`
`+drive8supercard`
: Enable/disable Supercard for drive 8 (`Drive8SuperCard=1`, `Drive8SuperCard=0`) (x64, x64sc, xscpu64 and x128 only).

`-drive9supercard`
`+drive9supercard`
: Enable/disable Supercard for drive 9 (`Drive9SuperCard=1`, `Drive9SuperCard=0`) (x64, x64sc, xscpu64 and x128 only).

`-drive10supercard`
`+drive10supercard`
: Enable/disable Supercard for drive 10 (`Drive10SuperCard=1`, `Drive10SuperCard=0`) (x64, x64sc, xscpu64 and x128 only).

`-drive11supercard`
`+drive11supercard`
: Enable/disable Supercard for drive 11 (`Drive11SuperCard=1`, `Drive11SuperCard=0`) (x64, x64sc, xscpu64 and x128 only).

`-supercard <name>`
: Specify name of Supercard ROM image (`DriveSuperCardName`) (x64, x64sc, xscpu64 and x128 only).

`-drive8stardos`
`+drive8stardos`
: Enable/disable StarDOS for drive 8 (`Drive8StarDos=1`, `Drive8StarDos=0`) (x64, x64sc, xscpu64 and x128 only).

`-drive9stardos`
`+drive9stardos`
: Enable/disable StarDOS for drive 9 (`Drive9StarDos=1`, `Drive9StarDos=0`) (x64, x64sc, xscpu64 and x128 only).

`-drive10stardos`
`+drive10stardos`
: Enable/disable StarDOS for drive 10 (`Drive10StarDos=1`, `Drive10StarDos=0`) (x64, x64sc, xscpu64 and x128 only).

`-drive11stardos`
`+drive11stardos`
: Enable/disable StarDOS for drive 11 (`Drive11StarDos=1`, `Drive11StarDos=0`) (x64, x64sc, xscpu64 and x128 only).

`-stardos <name>`
: Specify name of the image of the lower half of the StarDOS ROM. (Attach the upper half using the -dos1541 option.) (`DriveStarDosName`) (x64, x64sc, xscpu64 and x128 only).

6.11 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.

6.11.1 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 create P00 files on save.

6.11.1.1 Resources for file system devices

`IECDevice8`
`IECDevice9`
`IECDevice10`
`IECDevice11`

Booleans that specify whether IEC device emulation for device #8 to #11 is enabled.

`FileSystemDevice8`
`FileSystemDevice9`
`FileSystemDevice10`
`FileSystemDevice11`

Integers specifying the device type for device 8-11 respectively (all emulators except vsid). (0: None, 1: Filesystem, 2: OpenCBM (Real))

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

`FSDevice8SaveP00`
`FSDevice9SaveP00`
`FSDevice10SaveP00`
`FSDevice11SaveP00`
: Booleans specifying whether the drives should create P00 files instead of plain CBM ones for drives 8, 9, 10 and 11 respectively (all emulators except vsid).

`FSDevice8HideCBMFiles`
`FSDevice9HideCBMFiles`
`FSDevice10HideCBMFiles`
`FSDevice11HideCBMFiles`
: Booleans specifying whether non-P00 files should be invisible for drives 8, 9, 10 and 11 respectively (all emulators except vsid).

`FSDevice8Dir`
`FSDevice9Dir`
`FSDevice10Dir`
`FSDevice11Dir`
: Strings specifying the directories to which drives 8, 9, 10 and 11 have access (all emulators except vsid).

`FSDeviceLongNames`
: Boolean specifying whether filenames will get converted to 16 character long short names or not. Disabled by default.

`FSDeviceOverwrite`
: Boolean specifying whether files in the file system can be overwritten without using `"@:name"`, or not. Disabled by default.

6.11.1.2 Command-line options for file system devices

`-iecdevice8`
`+iecdevice8`
: Enable/disable IEC device emulation for device #8 (`IECDevice8=1`, `IECDevice8=0`).

`-iecdevice9`
`+iecdevice9`
: Enable/disable IEC device emulation for device #9 (`IECDevice9=1`, `IECDevice9=0`).

`-iecdevice10`
`+iecdevice10`
: Enable/disable IEC device emulation for device #10 (`IECDevice10=1`, `IECDevice10=0`).

`-iecdevice11`
`+iecdevice11`
Enable/disable IEC device emulation for device #11 (`IECDevice11=1`, `IECDevice11=0`).

`-device8 <type>`
`-device9 <type>`
`-device10 <type>`
`-device11 <type>`
Set device type for device 8-11 respectively (`FileSystemDevice8`, `FileSystemDevice9`, `FileSystemDevice10`, `FileSystemDevice11`) (all emulators except vsid). (0: None, 1: Filesystem, 2: OpenCBM (Real))

`-fs8 <Name>`
`-fs9 <Name>`
`-fs10 <Name>`
`-fs11 <Name>`
Specify the paths for the file system access on drives 8, 9, 10 and 11, respectively (`FSDevice8Dir`, `FSDevice9Dir`, `FSDevice10Dir` and `FSDevice11Dir`) (all emulators except vsid).

`-fs8convertp00`
`+fs8convertp00`
Enable/disable on-read support for P00 files on drive 8 (`FSDevice8ConvertP00=1`, `FSDevice8ConvertP00=0`) (all emulators except vsid).

`-fs9convertp00`
`+fs9convertp00`
Enable/disable on-read support for P00 files on drive 9 (`FSDevice9ConvertP00=1`, `FSDevice9ConvertP00=0`) (all emulators except vsid).

`-fs10convertp00`
`+fs10convertp00`
Enable/disable on-read support for P00 files on drive 10 (`FSDevice10ConvertP00=1`, `FSDevice10ConvertP00=0`) (all emulators except vsid).

`-fs11convertp00`
`+fs11convertp00`
Enable/disable on-read support for P00 files on drive 11 (`FSDevice11ConvertP00=1`, `FSDevice11ConvertP00=0`) (all emulators except vsid).

`-fs8savep00`
`+fs8savep00`
Enable/disable saving P00 files on drive 8 (`FSDevice8SaveP00=1`, `FSDevice8SaveP00=0`) (all emulators except vsid).

`-fs9savep00`
`+fs9savep00`
Enable/disable saving P00 files on drive 9 (`FSDevice9SaveP00=1`, `FSDevice9SaveP00=0`) (all emulators except vsid).

`-fs10savep00`
`+fs10savep00`
: Enable/disable saving P00 files on drive 10 (`FSDevice10SaveP00=1`, `FSDevice10SaveP00=0`) (all emulators except vsid).

`-fs11savep00`
`+fs11savep00`
: Enable/disable saving P00 files on drive 11 (`FSDevice11SaveP00=1`, `FSDevice11SaveP00=0`) (all emulators except vsid).

`-fs8hidecbm`
`+fs8hidecbm`
: Enable/disable hiding of CBM files for drive 8 (`FSDevice8HideCBMFiles=1`, `FSDevice8HideCBMFiles=0`) (all emulators except vsid).

`-fs9hidecbm`
`+fs9hidecbm`
: Enable/disable hiding of CBM files for drive 9 (`FSDevice9HideCBMFiles=1`, `FSDevice9HideCBMFiles=0`) (all emulators except vsid).

`-fs10hidecbm`
`+fs10hidecbm`
: Enable/disable hiding of CBM files for drive 10 (`FSDevice10HideCBMFiles=1`, `FSDevice10HideCBMFiles=0`) (all emulators except vsid).

`-fs11hidecbm`
`+fs11hidecbm`
: Enable/disable hiding of CBM files for drive 11 (`FSDevice11HideCBMFiles=1`, `FSDevice11HideCBMFiles=0`) (all emulators except vsid).

`-fslongnames`
`+fslongnames`
: Enable/disable whether filenames will get converted to 16 character long short names (`FSDeviceLongNames=1`, `FSDeviceLongNames=0`) (all emulators except vsid).

`-fsoverwrite`
`+fsoverwrite`
: Enable/disable whether existing files in the file system can be overwritten by default. Using `OPEN` or `SAVE"@:name"` this is always possible. (`FSDeviceOverwrite=1`, `FSDeviceOverwrite=0`) (all emulators except vsid).

`-flipname <name>`
: Specify name of the flip list file image (`FliplistName`) (all emulators except vsid).

6.11.2 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`.

6.11.2.1 Printer resources

`IECDevice4`
`IECDevice5`
`IECDevice6`
`IECDevice7`
: Booleans that specify whether IEC device emulation for device #4, #5, #6 and #7 is enabled.

`PrinterTextDevice1`
`PrinterTextDevice2`
`PrinterTextDevice3`
: Strings specifying the printer devices (see Section 6.11.2 [Printer settings], page 59).

`Printer4TextDevice`
`Printer5TextDevice`
`Printer6TextDevice`
: Integer (ranging from 0 to 2, for device 1-3) specifying what printer device (see Section 6.11.2 [Printer settings], page 59) the IEC printer is using.

`Printer4`
`Printer5`
`Printer6`
: Integer specifying how the printer (device 4-6) is being emulated. (0: None, 1: Filesystem, 2: Real)

`Printer7`
: Integer specifying how printer 7 is being emulated. (0: None, 2: Real)

`Printer4Driver`
: String specifying the printer output driver. (raw, ascii, mps803, nl10)

`Printer5Driver`
: String specifying the printer output driver. (raw, ascii, mps803, nl10)

`Printer6Driver`
: String specifying the printer output driver. (raw, 1520)

`Printer4Output`
`Printer5Output`
`Printer6Output`
: Strings specifying the IEC printer output device. (text, graphics)

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

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

`PrinterUserportDriver`
: String specifying the user-port printer output driver. (ascii/nl10/raw)

`PrinterUserportOutput`
: String specifying the user-port printer output device. (text, graphics)

6.11.2.2 Printer command-line options

`-iecdevice4`
`+iecdevice4`
: Enable/disable IEC device emulation for device #4 (`IECDevice4=1`, `IECDevice4=0`).

`-iecdevice5`
`+iecdevice5`
: Enable/disable IEC device emulation for device #5 (`IECDevice5=1`, `IECDevice5=0`).

`-iecdevice6`
`+iecdevice6`
: Enable/disable IEC device emulation for device #6 (`IECDevice6=1`, `IECDevice6=0`).

`-iecdevice7`
`+iecdevice7`
: Enable/disable IEC device emulation for device #7 (`IECDevice7=1`, `IECDevice7=0`).

`-device4 <type>`
: Set device type for device 4 (`Printer4`). (0: None, 1: Filesystem, 2: Real)

`-device5 <type>`
: Set device type for device 5 (`Printer5`). (0: None, 1: Filesystem, 2: Real)

`-device6 <type>`
: Set device type for device 6 (`Printer6`). (0: None, 1: Filesystem, 2: Real)

`-device7 <type>`
: Set device type for device 7 (`Printer7`). (0: None, 2: Real)

`-prtxtdev1 <name>`
`-prtxtdev2 <name>`
`-prtxtdev3 <name>`
Specify name of printer text device or dump file (`PrinterTextDevice1`, `PrinterTextDevice2`, `PrinterTextDevice3`).

`-pr4txtdev <0-2>`
`-pr5txtdev <0-2>`
`-pr6txtdev <0-2>`
Specify printer text output device for IEC printer #4-6 (`Printer4TextDevice`, `Printer5TextDevice`, `Printer6TextDevice`).

`-pr4output <name>`
Specify name of output device for device #4 (`Printer4Output`). (text, graphics)

`-pr5output <name>`
Specify name of output device for device #5 (`Printer5Output`). (text, graphics)

`-pr6output <name>`
Specify name of output device for device #6 (`Printer6Output`). (text, graphics)

`-pr4drv <name>`
Specify name of printer driver for device #4 (`Printer4Driver`). (raw, ascii, mps803, nl10)

`-pr5drv <name>`
Specify name of printer driver for device #5 (`Printer5Driver`). (raw, ascii, mps803, nl10)

`-pr6drv <name>`
Specify name of printer driver for device #6 (`Printer6Driver`). (raw, 1520)

`-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 (`PrinterUserportTextDevice`).

`-pruseroutput <name>`
Specify name of output device for the userport printer (`PrinterUserportOutput`). (text, graphics)

`-pruserdrv <name>`
Specify name of printer driver for the userport printer (`PrinterUserportDriver`).

6.11.3 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 (t64) support.

Since the 3.0 release kernal traps have been disabled by default to ensure maximum compatibility.

6.11.3.1 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.

6.11.3.2 Command-line options to control Kernal traps

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

6.12 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, 1200 and 2400 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 Swithlink, Datapump or Turbo232 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.
- connect through a network socket.

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, a network address, 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.

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 `127.0.0.1:25232`;
- device 4, whose default value is `|nc 127.0.0.1:25232`.

You can change the baudrate the tty device is set to by specifying it on the commandline or in the menu.

6.12.1 RS232 resources

`RsDevice1`
`RsDevice2`
`RsDevice3`
`RsDevice4`
: Strings specifying the RS232 devices (see Section 6.12 [RS232 settings], page 63).

`RsDevice1ip232`
`RsDevice2ip232`
`RsDevice3ip232`
`RsDevice4ip232`
: Boolean specifying whether the respective RS232 devices (see Section 6.12 [RS232 settings], page 63) use the IP232 protocol supported by tcpser. This only works with Socket connections.

`Acia1Enable`
: Boolean specifying whether the ACIA (Swiftlink, Turbo232) cartridge should be emulated or not (x64, x64sc, xscpu64, x128 and xvic only, and only if RS232 support is enabled and supported at compile time).

`Acia1Dev` Integer specifying what RS232 device (see Section 6.12 [RS232 settings], page 63) the ACIA is using (all emulators except x64dtv and vsid, and only if RS232 support is enabled and supported at compile time).

`Acia1Base`
: Integer specifying the base address for the emulated ACIA chip (x64, x64sc, xscpu64, xvic and x128 only, and only if RS232 support is enabled and supported at compile time). (xvic: $9800/$9C00, x128: $D700/$DE00/$DF00, x64, x64sc, xscpu64: $DE00/$DF00)

`Acia1Mode`
: Integer specifying the type of emulated RS232 interface (x64, x64sc, xscpu64, xvic and x128 only, and only if RS232 support is enabled and supported at compile time). (0: normal, 1: Swiftlink, 2: Turbo232)

`Acia1Irq` Integer specifying which interrupt to use (x64, x64sc, xscpu64, xvic and x128 only, and only if RS232 support is enabled and supported at compile time). (0 = none, 1 = NMI, 2 = IRQ)

`RsUserEnable`
: Boolean specifying if the user-port RS232 interface is being emulated (C64, C128 and VIC20).

`RsUserBaud`
: Integer specifying the baudrate of the user-port RS232 interface (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).

The following resources are only available if RS232 device support or RS232 network support is available at compile time.

`RsDevice1Baud`
`RsDevice2Baud`
`RsDevice3Baud`
`RsDevice4Baud`
: Integers specifying the RS232 baudrate devices if the device file points to a special device (like `/dev/ttyS0`; see Section 6.12 [RS232 settings], page 63) (all emulators except vsid).

6.12.2 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`).

`-rsdev1ip232`
`+rsdev1ip232`
`-rsdev2ip232`
`+rsdev2ip232`
`-rsdev3ip232`
`+rsdev3ip232`
`-rsdev4ip232`
`+rsdev4ip232`
: Enable/Disable whether the respective RS232 devices (see Section 6.12 [RS232 settings], page 63) use the IP232 protocol supported by tcpser. This only works with Socket connections.

`-acia1`
`+acia1` Enable/Disable the $DE** ACIA RS232 interface emulation (`Acia1Enable=1`, `Acia1Enable=0`) (x64, x64sc, xscpu64, x128 and xvic only, and only if RS232 support is enabled and supported at compile time).

`-myaciadev <0-3>`
: Specify RS232 device the ACIA should work on (all emulators except x64dtv and vsid, and only if RS232 support is enabled and supported at compile time)

`-acia1base <Base address>`
: Set the base address of the ACIA cartridge (`Acia1Base`) (x64, x64sc, xscpu, x128 and xvic only, and only if RS232 support is enabled and supported at compile time). (xvic: $9800/$9C00, x128: $D700/$DE00/$DF00, x64, x64sc, xscpu: $DE00/$DF00)

`-acia1mode <mode>`
: Set the ACIA mode (`Acia1Mode`) (x64, x64sc, xscpu64, xvic and x128 only, and only if RS232 support is enabled and supported at compile time). (0: Normal, 1: Swiftlink, 2: Turbo232)

`-acia1irq <interrupt>`
: Set the ACIA interrupt (`Acia1Irq`) (x64, x64sc, xscpu64, xvic and x128 only, and only if RS232 support is enabled and supported at compile time). (0: None, 1: NMI, 2: IRQ)

`-rsuser`
`+rsuser` Enable or disable emulation of the userport RS232 emulation (`RsUser`; C64, C128 and VIC20)

`-rsuserbaud <baud>`
: Set the baud rate of the RS232 userport emulation.

`-rsuserdev <0-3>`
: Specify device for the userport RS232 emulation (`RsUserDev`; C64, C128 and VIC20).

The following command-line options are only available if RS232 device support or RS232 network support is available at compile time.

`-rsdev1baud <baudrate>`
`-rsdev2baud <baudrate>`
`-rsdev3baud <baudrate>`
`-rsdev4baud <baudrate>`
: Specify <baudrate> as baudrate for the RS232 devices if the device name specifies a special device (like `/dev/ttyS0` for example, see Section 6.12 [RS232 settings], page 63; `RsDevice1Baud`, `RsDevice2Baud` `RsDevice3Baud` and `RsDevice4Baud`) (all emulators except vsid).

6.12.3 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, 2400 or 9600 according to what you enabled in the VICE menu for the userport.

6.13 Tape port devices

6.13.1 Tape port resources

`CPClockF83`
: Boolean specifying whether the CP Clock F83 (PCF8583 RTC) is enabled.

`CPClockF83Save`
: Boolean specifying whether the CP Clock F83 (PCF8583 RTC) data is saved when changed.

`TapeSenseDongle`
: Boolean specifying whether the tape sense dongle is enabled.

`DTLBasicDongle`
: Boolean specifying whether the DTL Basic dongle is enabled.

`TapecartEnabled`
: Boolean that specifies if a tapecart is attached.

`TapecartUpdateTCRT`
: Boolean, if true write back the tapecart memory contents back to the .tcrt file when detaching the image or quitting the emulator.

`TapecartOptimizeTCRT`
: Boolean, when set to true the .tcrt image will be optimized by leaving out blank space at the end when saving.

`TapecartLogLevel`
: Integer that specifies the tapecart emulation log level. At the default level of 0, only errors are logged. Level 1 additionally logs mode changes and command bytes and level 2 adds details of command parameters.

`TapecartTCRTFilename`
: String specifying the file name of the current tapecart image.

6.13.2 Tape port command line options

`-cpclockf83`
`+cpclockf83`
: Enable/Disable CP Clock F83 (PCF8583 RTC) (`CPClockF83=1`, `CPClockF83=0`).

`-cpclockf83save`
`+cpclockf83save`
: Enable/Disable saving of the CP Clock F83 (PCF8583 RTC) data when changed (`CPClockF83Save=1`, `CPClockF83Save=0`).

`-tapesensedongle`
`+tapesensedongle`
: Enable/Disable tape sense dongle (`TapeSenseDongle=1`, `TapeSenseDongle=0`).

`-dtlbasicdongle`
`+dtlbasicdongle`
: Enable/Disable DTL Basic dongle (`DTLBasicDongle=1`, `DTLBasicDongle=0`).

`-tapecart`
`+tapecart`
: Enable/disable tapecart emulation (`TapecartEnabled=1`, `TapecartEnabled=0`)

`-tcrt <name>`
: Specify tapecart .tcrt image filename (`TapecartTCRTFilename`)

`-tapecartupdatetcrt`
`+tapecartupdatetcrt`
: Specify if the attached .tcrt image should be updated when the emulated tapecart is written to. (`TapecartUpdateTCRT=1`, `TapecartUpdateTCRT=0`)

`-tapecartoptimizetcrt`
`+tapecartoptimizetcrt`
: Specify if writing to a .tcrt file should optimize its size by leaving out blank space at the end. (`TapecartOptimizeTCRT=1`, `TapecartOptimizeTCRT=0`)

`-tapecartloglevel <number>`
: Specify the tapecart log level. The default level of 0 only logs errors. Level 1 additionally logs mode changes and command bytes and level 2 adds details of command parameters.

6.14 Userport devices

6.14.1 Userport resources

`UserportCollisionHandling`
: Boolean specifying the way the Userport collisions should be handled, (0: error message and detach all involved devices, 1: error message and detach last attached involved device, 2: warning in log and 'AND' the valid return values)

`UserportDIGIMAX`
: Boolean specifying whether the userport DigiMAX device is enabled.

`Userport4bitSampler`
: Boolean specifying whether the Userport 4bit sampler is enabled.

`Userport8BSS`
: Boolean specifying whether the Userport 8bit stereo sampler is enabled.

`UserportRTC58321a`
: Boolean specifying whether the userport RTC is emulated or not (xpet, cbm2, x64, x64sc, xscpu64 and x128 only).

`UserportRTC58321aSave`
: Boolean specifying whether the userport RTC data is saved when changed or not (xpet, cbm2, x64, x64sc, xscpu64 and x128 only).

`UserportRTCDS1307`
: Boolean specifying whether the userport RTC is emulated or not (xpet, cbm2, x64, x64sc, xscpu64 and x128 only).

`UserportRTCDS1307Save`
: Boolean specifying whether the userport RTC data is saved when changed or not (xpet, cbm2, x64, x64sc, xscpu64 and x128 only).

6.14.2 Userport command line options

`-userportcollision`
: Select the way the Userport collisions should be handled, (0: error message and detach all involved devices, 1: error message and detach last attached involved device, 2: warning in log and 'AND' the valid return values) (`UserportCollisionHandling`)

`-userportdigimax`
`+userportdigimax`
: Enable/Disable the userport DigiMAX device (`UserportDIGIMAX=1`, `UserportDIGIMAX=0`)

`-userport4bitsampler`
`+userport4bitsampler`
: Enable/Disable Userport 4bit sampler (`Userport4bitSampler=1`, `Userport4bitSampler=0`)

`-userport8bss`
`+userport8bss`
: Enable/Disable Userport 8bit stereo sampler (`Userport8BSS=1`, `Userport8BSS=0`)

`-userportrtc58321a`
`+userportrtc58321a`
: Enable/disable the userport RTC emulation (`UserportRTC58321a=1`, `UserportRTC58321a=0`) (xpet, cbm2, x64, x64sc, xscpu64 and x128 only).

`-userportrtc58321asave`
`+userportrtc58321asave`
: Disable/enable saving of the userport RTC data when changed (`UserportRTC58321aSave=1`, `UserportRTC58321aSave=0`) (xpet, cbm2, x64, x64sc, xscpu64 and x128 only).

`-userportrtcds1307`
`+userportrtcds1307`
: Enable/disable the userport RTC emulation (`UserportRTCDS1307=1`, `UserportRTCDS1307=0`) (xpet, cbm2, x64, x64sc, xscpu64 and x128 only).

`-userportrtcds1307save`
`+userportrtcds1307save`
: Disable/enable saving of the userport RTC data when changed (`UserportRTCDS1307Save=1`, `UserportRTCDS1307Save=0`) (xpet, cbm2, x64, x64sc, xscpu64 and x128 only).

6.15 Monitor settings

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

6.15.1 Monitor resources

`KeepMonitorOpen`
: Boolean, if true the monitor window may stay open when the emulation is running, eg to look at trace-point output. (Not all ports/UIs support this, in that case this setting has no effect.)

`RefreshOnBreak`
: Boolean, if true, the emulated display(s) will update as the monitor is entered and also after each command executed via the monitor. Not entirely accurate yet, as updates become visible one scanline at a time. In addtion, the image can be not quite right around the raster line being updated.

`MonitorServer`
: Boolean specifying whether the remote monitor server is enabled.

`MonitorServerAddress`
: String specifying the address the remote monitor server listens to (ip4://127.0.0.1:6510)

`BinaryMonitorServer`
: Boolean specifying whether the binary monitor server is enabled.

`BinaryMonitorServerAddress`
: String specifying the address the binary monitor server listens to (ip4://127.0.0.1:6502)

`NativeMonitor`
: Boolean specifying whether the native monitor is enabled. When enabled, the monitor will not run in the interface of the emulator, but instead work in the spawning terminal.

`MonitorLogEnabled`
: Boolean specifying whether the output of the monitor will be logged to a file.

`MonitorLogFileName`
: String specifying the logfile name for the monitor.

`MonitorChisLines`
: Integer specifying the number of lines to keep in the cpu history. (only when enabled in configure)

`MonitorScrollbackLines`
: Integer specifying the number of lines to keep in the monitor scrollback buffer (-1 for no limit).

`MonitorFont`
: String specifying the font to use in the Gtk3 UI's VTE monitor window. Should be in the form accepted by Pango, for example "Liberation Mono 11".

6.15.2 Monitor command-line options

`-moncommands <Name>`
: Execute the commands from the file <Name> in the monitor after starting up. All monitor commands are supported, including playback of other command files via `pb`. By default, the monitor opens and starts executing the commands just before the kernal reset routine starts. The kernal starup sequence may interfere with what you are trying to achieve, for example by clearing any basic listing you may have loaded. In this case, use the `-initbreak` option to control when the monitor will open and execute your script. You can choose an address, or use `-initbreak ready` to attempt to detect when the kernel is ready and execute the script then.

`-initbreak <address>`
`-initbreak reset`
`-initbreak ready`
: Set an initial breakpoint for the monitor. Addresses with prefix "0x" are hexadecimal. `-initbreak reset` will break just before the reset routine starts. `-initbreak ready` will attempt to break when the kernal is 'ready'.

`-keepmonopen`
`+keepmonopen`
: Enable/disable keeping the monitor window open (`KeepMonitorOpen=1`, `KeepMonitorOpen=0`).

`-refreshonbreak`
`+refreshonbreak`
: Enable/Disable refresh display when entering monitor and after each monitor command. (`RefreshOnBreak=1`, `RefreshOnBreak=0`).

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

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

`-binarymonitor`
`+binarymonitor`
Enable/Disable binary monitor

`-binarymonitoraddress <name>`
The local address the binary monitor should bind to

`-nativemonitor`
`+nativemonitor`
Enable/Disable native monitor. (`NativeMonitor=1`, `NativeMonitor=0`).

`-monlog`
`+monlog` Enable/Disable logging monitor output to a file. (`MonitorLogEnabled=1`, `MonitorLogEnabled=0`).

`-monlogname <name>`
Specify logfile name for the monitor. (`MonitorLogFileName`).

`-monchislines <value>`
Set number of lines to keep in the cpu history. (only when enabled in configure) (`MonitorChisLines`).

`-monscrollbacklines <value>`
Set number of lines to keep in the monitor scrollback buffer (-1 for no limit). (`MonitorScrollbackLines`).

`-monitorfont <font-description>`
Set the monitor font for the Gtk3 UI's VTE-monitor. (`MonitorFont`).

6.16 RAM init pattern settings

The initial content of uninitialized RAM depends on many factors, the inner workings of the RAM chip, the motherboard, etc. Generally, many RAM chips are build in a way that - however - at powerup about half the bits turn out 1, and about the other half turns out 0. As a consequence, and depending on what kind of RAM chips have been used in what kind of way, the typical patterns can be "seen" by the CPU right after powerup.

VICE tries to model (some of) the real patterns accurately. Please be aware of the fact that there is - technically buggy - software out there that will only work with a certain RAM init pattern.

How generating the pattern works is easily described in three simple steps:

1. A base pattern is created using blocks of `RAMInitStartValue`, which is inverted every `RAMInitValueInvert` bytes. This pattern can be offset ("rotated") by `RAMInitValueOffset` bytes. This step creates the typical 00 00 ff ff ff ff 00 00 etc patterns.

2. Every `RAMInitPatternInvert` bytes the base pattern is inverted ("xored") by `RAMInitPatternInvertValue`. This step creates the inverted pattern every $4000 or so bytes that is typical for some boards/ICs. It may also create the 0x99, 0x66 based patterns that show up on other setups.

3. Every `RAMInitRepeatRandom` bytes `RAMInitStartRandom` bytes are randomly inverted ("xored"). Additionally every bit is randomly inverted with a `RAMInitRandomChance` : 4096 chance. This step creates the typical random bytes that have been observed with some boards/ICs.

6.16.1 RAM init pattern resources

`RAMInitStartValue`
: Integer specifying the first value of the base pattern (all emulators except vsid). (0..255)

`RAMInitValueInvert`
: Integer specifying the number of bytes until the base pattern is inverted (all emulators except vsid).

`RAMInitValueOffset`
: Integer specifying the number of bytes to offset the base pattern (all emulators except vsid).

`RAMInitPatternInvert`
: Integer specifying the length of the memory block when inverting the base pattern (all emulators except vsid).

`RAMInitPatternInvertValue`
: Integer specifying the value to use for inverting the base pattern with (all emulators except vsid).

`RAMInitStartRandom`
: Integer specifying the number of random bytes in the random pattern (all emulators except vsid).

`RAMInitRepeatRandom`
: Integer specifying the number of bytes after which to repeat the random pattern (all emulators except vsid).

`RAMInitRandomChance`
: Integer specifying the chance of any bit to randomly toggle (0-4096) (all emulators except vsid).

6.16.2 RAM init pattern command-line options

`-raminitstartvalue <value>`
: Set the first value of the base pattern (`RAMInitStartValue`) (all emulators except vsid). (0..255)

`-raminitvalueinvert <num of bytes>`
: Set the number of bytes until the base pattern is inverted (`RAMInitValueInvert`) (all emulators except vsid).

`-raminitvalueoffset <num of bytes>`
: Offset the first pattern by <num> of bytes (`RAMInitValueOffset`) (all emulators except vsid).

`-raminitpatterninvert <num of bytes>`
: Set the length of the memory block when inverting the base pattern (`RAMInitPatternInvert`) (all emulators except vsid).

`-raminitpatterninvertvalue <value>`
: Set the length of the memory block when inverting the base pattern (`RAMInitPatternInvertValue`) (all emulators except vsid).

`-raminitstartrandom <num of bytes>`
: Set the number of random bytes in the random pattern (`RAMInitStartRandom`) (all emulators except vsid).

`-raminitrepeatrandom <num of bytes>`
: Set the number of bytes after which to repeat the random pattern (`RAMInitRepeatRandom`) (all emulators except vsid).

`-raminitrandomchance <value>`
: Set the chance of any bit to randomly toggle (0-4096) (`RAMInitRandomChance`) (all emulators except vsid).

6.17 Debug settings

6.17.1 Debug resources

`TapeLog` Boolean specifying whether the tape log device is enabled.

`TapeLogDestination`
: Integer specifying where the tape log goes to (0=Enable logging to the emulator log file 1=Enable logging to a file).

`TapeLogfilename`
: String that specifies the tape log file name

`DebugCartEnable`
: Boolean specifying whether the debug "cartridge" used for the test suite is enabled.

  The following are only available when the emulators were compiled in DEBUG mode:

`TraceMode`
: Integer specifying the trace mode (0=normal 1=small 2=history)

`AutoPlaybackFrames`
: Amount of automatic playback frames

`MainCPU_TRACE`
: Trace the main CPU / Do not trace the main CPU

`Drive0CPU_TRACE`
: Trace the drive 0 CPU / Do not trace the drive 0 CPU

`Drive1CPU_TRACE`
: Trace the drive 1 CPU / Do not trace the drive 1 CPU

`Drive2CPU_TRACE`
: Trace the drive 2 CPU / Do not trace the drive 2 CPU

`Drive3CPU_TRACE`
: Trace the drive 3 CPU / Do not trace the drive 3 CPU

`IEC_TRACE`
: Trace IEC bus activity / Do not trace IEC bus activity

6.17.2 Debug command-line options

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

`-debug`
`+debug`
: Disable/enable calling of the exception handler (`DoCoreDump=1`, `DoCoreDump=0`) (all emulators except vsid).

`-debugcart`
`+debugcart`
: Enable/disable the debug "cartridge" used for the test suite.

`-tapelog`
`+tapelog`
: Enable/Disable the tape log device. (`TapeLog=1`, `TapeLog=0`).

`-tapelogtofile`
: Enable logging to a file (`TapeLogDestination=1`).

`-tapelogtolog`
: Enable logging to the emulator log file (`TapeLogDestination=0`).

`-tapelogimage <name>`
: Specify tape log file name (`TapeLogfilename`).

The following are only available when the emulators were compiled in DEBUG mode:

`-trace_maincpu`
`+trace_maincpu`
: Trace the main CPU / Do not trace the main CPU (`MainCPU_TRACE=1`, `MainCPU_TRACE=0`)

`-trace_drive0`
`+trace_drive0`
: Trace the drive 0 CPU / Do not trace the drive 0 CPU (`Drive0CPU_TRACE=1`, `Drive0CPU_TRACE=0`)

`-trace_drive1`
`+trace_drive1`
: Trace the drive 1 CPU / Do not trace the drive 1 CPU (`Drive1CPU_TRACE=1`, `Drive1CPU_TRACE=0`)

`-trace_drive2`
`+trace_drive2`
: Trace the drive 2 CPU / Do not trace the drive 2 CPU (`Drive2CPU_TRACE=1`, `Drive2CPU_TRACE=0`)

`-trace_drive3`
`+trace_drive3`
: Trace the drive 3 CPU / Do not trace the drive 3 CPU (`Drive3CPU_TRACE=1`, `Drive3CPU_TRACE=0`)

`-trace_iec`
`+trace_iec`
: Trace IEC bus activity / Do not trace IEC bus activity (`IEC_TRACE=1`, `IEC_TRACE=0`)

`-trace_mode <value>`
: Trace mode (0=normal 1=small 2=history)

`-autoplaybackframes <frames>`
: Set the amount of automatic playback frames

6.18 Network Play settings

6.18.1 Network Play resources

`NetworkServerName`
: String specifying the name of the remote server.

`NetworkServerBindAddress`
: String specifying the IP of the remote server.

`NetworkServerPort`
: Integer specifying the port used for network play.

`NetworkControl`
: Integer specifying whether the emulator is running as server or client (0: client, 1: server)

6.18.2 Network Play command-line options

`-netplayserver <name>`
: Set the name of the remote server.

`-netplaybind <ip>`
: Set the IP of the remote server.

`-netplayport <port>`
: Set the port used for network play.

`-netplayctrl <flag>`
: Specify whether the emulator is running as server or client (0: client, 1: server)

6.19 Miscellaneous settings

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

6.19.1 Miscellaneous resources

`JAMAction`
: Integer specifying the action to take when the CPU encounters a 'JAM' opcode. (0: show dialog, 1: continue emulation, 2: start monitor, 3: soft reset, 4: hard reset, 5: quit emulator)

`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.

`DoCoreDump`
: Boolean specifying whether the emulator should dump core when it gets a signal (all emulators except vsid).

`LogFileName`
: String specifying the filename of the current log file.

`ExitScreenshotName`
: String specifying the filename of a screenshot file that will be written when the emulator exits.

`ExitScreenshotName1`
: String specifying the filename of a screenshot file that will be written when the emulator exits. (x128)

`FliplistName`
: String specifying the filename of the current flip list. (Drive 8 only) (all emulators except vsid).

`AttachDevice8Readonly`
`AttachDevice9Readonly`
`AttachDevice10Readonly`
`AttachDevice11Readonly`
: Booleans that specify whether to attach images on drives 8 to 11 read-only or not (all emulators except vsid).

`AttachDevice8d1Readonly`
`AttachDevice9d1Readonly`
`AttachDevice10d1Readonly`
`AttachDevice11d1Readonly`
: Booleans that specify whether to attach images on the second drive of dual-drives 8 to 11 read-only or not (all emulators except vsid).

6.19.2 Miscellaneous command-line options

`-jamaction <Type>`
: Specify the action to take when the CPU encounters a 'JAM' opcode (`JAMAction`) (0: Show dialog, 1: continue emulation, 2: start monitor, 3: soft reset, 4: hard reset, 5: quit emulator).

`-directory <Path>`
: Specify the system file search path (`Directory`).

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 Håkan 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:

7.1.1.1 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

7.1.1.2 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

7.1.1.3 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 4KiB, 8KiB and 16KiB game- and ultimax cartridges
- Action Replay V5
- Action Replay MK2
- Action Replay MK3
- Action Replay MK4
- Atomic Power
- BIS-Plus
- Blackbox V3
- Blackbox V4
- Blackbox V8
- Blackbox V9
- C64 Games System
- Capture
- Comal 80
- Dela EP64
- Dela EP7x8
- Dela EP256
- Diashow-Maker
- Dinamic
- Easy Calc
- EasyFlash
- Epyx FastLoad
- EXOS

- The Final Cartridge
- The Final Cartridge III
- Final Cartridge Plus
- Formel 64
- Freeze Frame
- Freeze Machine
- Fun Play / Powerplay
- Game Killer
- GMod2
- GMod3
- H.E.R.O. bootleg
- IDE64 (`http://www.ide64.org`)
- KCS Power Cartridge
- Kingsoft
- Lt. Kernal Host Adaptor
- MACH 5
- Magic Desk
- Magic Formel
- MAX Basic
- Mikro Assembler
- MMC Replay
- Ocean
- Pagefox
- Prophet64
- RAMLink
- REX 256KiB EPROM Cart
- REX RAM-Floppy
- REX Utility
- Retro Replay
- RGCD
- RR-Net MK3
- ROSS
- SD-BOX
- Silverrock 128
- Simons' BASIC
- Snapshot 64
- Stardos
- Structured BASIC
- Super Explode V5.0

- Super Games
- Super Snapshot V4
- Super Snapshot V5
- Warp Speed
- Westermann Learning
- Zaxxon
- ZIPP-CODE 48

7.1.1.4 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
- DS12C887 RTC
- Ethernet (The Final Ethernet, RR-Net)
- GEO-RAM
- MIDI (Passport/Syntech, Datel/Siel/JMS/C-Lab, Maplin, Namesoft, Sequential)
- RAM Expansion Module (REU)
- SFX Sound Expander
- SFX Sound Sampler

7.1.1.5 Expected behaviour

When the emulator is run without arguments, all settings from the config file should be applied and arguments override settings from the config file.

When saving the settings to the config file it is expected that on the next run of the emulator all settings will be in the same state as they were when saved.

There is an exception to this rule: the cartridge in the "Main Slot" must be explicitly set as default before it gets saved to the config file.

`+cart` should disable ALL cartridges, including eventually activated REU, Swithlink and all similar expansionport devices.

`-cartXYZ` options should generally attach AND activate a cart of type XYZ. As a consequence, attaching carts this way which are NOT in the "Main Slot" will also enable the cart permanently.

7.1.1.6 Common problems

If attaching a cartridge does not work as expected, this may be because of various reasons:

- Not seldomly the CRT type is incorrectly set in `.crt` files found "in the wild". Make sure this is not the case (if in doubt use `cartconv` to verify and/or fix).

- You may have unintentionally enabled more than one cartridge at once, for example by saving the settings with REU enabled, and then later attaching a game cartridge from the command-line. The cartridge system will allow certain combinations, but (as on the real thing) not all do (can) actually work. To make sure this is not the case, either detach all cartridges from the menus, or use `+cart` on the command-line.
- The cartridge image might be broken. Try one from a different source. If you are sure the dump is ok (for example because you dumped it yourself) then make sure it is in proper linear order (on some cartridges, for example "capture", address and/or data lines at the eprom are shuffled around so a dump made with an eprom burner can not be used as is).
- Last not least you might have encountered a bug in the emulation. If you suspect this is the case, and you can still reproduce the bug after checking the things above, please file a bug report including the following information:
  - attach your vicerc and a reference to the cartridge binaries
  - if you can, comment in the respective DEBUGXYZ macros prominently defined at the top of these files: src/c64/cart/c64cart.c src/c64/cart/c64cartmem.c src/c64/c64io.c src/c64/c64export.c and then recompile. this will add debug output that might make it much easier to locate certain problems.

7.1.1.7 IEEE-488 interface

To be able to use an IEEE drive, you need to enable IEEE emulation for the emulator. To do this, follow the following steps:

Download the IEEE 488 ROM image from the CBM archives (formerly known as FUNET) Attach that image with File/Attach cartridge image/IEEE488 interface image.

Make sure you have a one-drive system only (that is, go to Settings/Peripheral Setting, uncheck "use IEC device" for all devices, go to Settings/Drive Settings and select "Floppy type" as "none" for all drives other than drive 8.

After this, all drives can be selected in x64 and x128.

7.1.1.8 The Final Cartridge 3

The Final Cartridge 3 detects whether 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.

7.1.1.9 CMD RAMLink

The CMD RAMLink can operate standalone using its own RAMCard or with an REU, GEORAM, or RAMDrive (only one) which is inserted into the RAM port. The RAMLink is essentially a builtin high capacity RAMDOS with battery backup support. This "backup" support can be emulated through the use of images. Initially it will "format" the memory so each has their own native parition on drive 16. Paritions and the device number can be changed with the CMD "RAM-tools". The latest RAMLink firmware (2.01) can not handle REU sizes other than 128 and 512 KiB, or GEORAM sizes beyond 2 MiB in VICE, as the firmware will hang on size detection.

The RAMLink also features a parallel port (similar to IEEE-488 interface) which allows faster transfers to the CMD HD series drives. The "cabling" is automatically performed by

VICE and the RAMLink firmware will detect the cable presence and use it automatically. However, some programs may have compatibility issues (notably the CMD HD-tools with older HD boot roms). To disable this, issue the JiffyDOS command "@P0" to disable, or "@P1" to enable parallel transfers.

The RAMLink features an "Enable/Disable" switch that can be changed at anytime using the GUI options. Prior to disabling, the JiffyDOS "@Q" command should be issued to remove the JiffyDOS system vectors. See the manual for how to reactivate JiffyDOS. It is not recommended to change the REU, GEORAM, or RAMCard size as the CMD RAMDOS was not designed to detect this. Any changes should be followed by a machine reset.

The RAMLink also has a "Normal/Disable" switch which can also be changed at anytime. Normal, the default, only allows RAMLink access to the RAM port (REU, GEORAM, and RAMDrive); their registers are hidden otherwise. Direct exposes their registers all the time. This can lead to potential data corruption of the data on the RAM expansion devices as RAMLink can not guarantee its access is exclusive.

RAMLink is also compatible with the CMD SuperCPU 64, however it requires the official SuperCPU ROM (2.04) along with the RAMLink ROM to properly detect and utilize all the RAMLink features.

7.1.2 C64 cartridge settings

7.1.2.1 C64 cartridge resources

`IOCollisionHandling`
: Integer specifying the way the I/O collisions should be handled. (0: error message and detach all involved carts, 1: error message and detach last attached involved carts, 2: warning in log and 'AND' the valid return values)

`CartridgeReset`
: Boolean specifying whether the machine should be reset when a cartridge is changed.

`CartridgeType`
: Integer specifying the type of cartridge emulated in the "main" slot.

  The following cartridge types are valid:

  - - 6: Ultimax
  - - 3: Generic 8KiB
  - - 2: Generic 16KiB
  - - 1: None
  - 0: CRT
  - 1: Action Replay V5
  - 2: KCS Power Cartridge
  - 3: The Final Cartridge III
  - 4: Simons' BASIC
  - 5: Ocean
  - 6: Expert Cartridge

- 7: Fun Play
- 8: Super Games
- 9: Atomic Power / Nordic Power
- 10: Epyx FastLoad
- 11: Westermann Learning
- 12: REX Utility
- 13: The Final Cartridge
- 14: Magic Formel
- 15: C64 Games System
- 16: Warp Speed
- 17: Dinamic
- 18: Zaxxon
- 19: Magic Desk
- 20: Super Snapshot V5
- 21: Comal 80
- 22: Structured BASIC
- 23: ROSS
- 24: Dela EP64
- 25: Dela EP7x8
- 26: Dela EP256
- 27: REX 256KiB EPROM Cart
- 28: Mikro Assembler
- 29: Final Cartridge Plus
- 30: Action Replay MK4
- 31: Stardos
- 32: EasyFlash
- 33: EasyFlash Xbank
- 34: Capture
- 35: Action Replay MK3
- 36: Retro Replay
- 37: MMC64
- 38: MMC Replay
- 39: IDE64
- 40: Super Snapshot V4
- 41: IEEE-488 Interface
- 42: Game Killer
- 43: Prophet64
- 44: EXOS
- 45: Freeze Frame

- 46: Freeze Machine
- 47: Snapshot 64
- 48: Super Explode V5.0
- 49: Magic Voice
- 50: Action Replay MK2
- 51: MACH 5
- 52: Diashow-Maker
- 53: Pagefox
- 54: Kingsoft
- 55: Silverrock 128KiB Cartridge
- 56: Formel 64
- 57: RGCD
- 58: RR-Net MK3
- 59: EasyCalc
- 60: GMod2
- 61: MAX Basic
- 62: GMod3
- 63: ZIPP-CODE 48
- 64: Blackbox V8
- 65: Blackbox V3
- 66: Blackbox V4
- 67: REX RAM-Floppy
- 68: BIS-Plus
- 69: SD-BOX
- 70: MultiMAX
- 71: Blackbox V9
- 72: Lt. Kernal Host Adaptor
- 73: RAMLink
- 74: H.E.R.O. bootleg

`CartridgeFile`
: String specifying the filename of the image for the cartridge emulated in the "main" slot.

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

`DQBBfilename`
: String specifying the filename of the DQBB RAM image.

`DQBBImageWrite`
: Boolean, if true write back the DQBB image file automatically, incase the RAM contents changed, when detaching or quitting the emulator.

`EasyFlashJumper`
: Boolean specifying whether the Easy Flash jumper is set.

`EasyFlashWriteCRT`
: Boolean, if true write back the Easy Flash image file automatically, incase the contents changed, when detaching or quitting the emulator.

`EasyFlashOptimizeCRT`
: Boolean, if true omit empty (filled with $ff) banks from the .crt image when writing.

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

`Expertfilename`
: String specifying the filename of the Expert Cartridge RAM image.

`ExpertImageWrite`
: Boolean, if true write back the Expert Cartridge image file automatically, incase the RAM contents changed, when detaching or quitting the emulator.

`ExpertCartridgeMode`
: Integer specifying the state of the expert cartridge switch. (0: off, 1: prg, 2: on)

`GMod2EEPROMImage`
: String that specifies the name of the raw GMod2 EEPROM image.

`GMod2FlashWrite`
: Boolean that specifies wether writes to GMod2 EEPROM image are enabled.

`GMod2EEPROMRW`
: Boolean that specifies wether the GMod2 ROM is saved at exit

`GMod3FlashWrite`
: Boolean that specifies wether changes to GMod3 EEPROM image are written back to the cartridge image.

`IDE64version`
: Integer specifying whether the emulated card version is V3.4, V4.1 or V4.2. This is automatically detected most of the time for .crt cartridge images.

`IDE64Image1`
`IDE64Image2`
`IDE64Image3`
`IDE64Image4`
: Strings specifying the full path to the four harddisk images. If a file is non-existing the drive is not emulated. Some older IDEDOS versions only support the first two harddisks.

`IDE64Cylinders1`
`IDE64Cylinders2`
`IDE64Cylinders3`
`IDE64Cylinders4`
: Integers specifying the number of cylinders for the four harddisk images. (1..65535)

`IDE64Heads1`
`IDE64Heads2`
`IDE64Heads3`
`IDE64Heads4`
: Integers specifying the number of heads for the four harddisk images. (1..16)

`IDE64Sectors1`
`IDE64Sectors2`
`IDE64Sectors3`
`IDE64Sectors4`
: Integers specifying the number of sectors for the four harddisk images. (1..63)

`IDE64AutodetectSize1`
`IDE64AutodetectSize2`
`IDE64AutodetectSize3`
`IDE64AutodetectSize4`
: Booleans specifying whether the disk geometry should be auto detected based on the disk image for the respective harddisk, or the cylinder/head/sector resources above should be used.

`IDE64USBServerAddress`
: String specifying the address the IDE64 USB server listens to (ip4://127.0.0.1:64245)

`IDE64USBServer`
: Boolean specifying whether the IDE64 USB server is enabled.

`IDE64RTCSave`
: Boolean specifying whether the IDE64 RTC data should be saved when changed or not.

`IDE64ClockPort`
: Integer that specifies the enabled IDE64 Clockport device. (0: None, 2: RRNet, 4: MP3@64)

`SBDIGIMAX`
: Boolean that specifies whether the Short Bus DigiMAX expansion is enabled.

`SBDIGIMAXbase`
: Integer specifying the Base address of the Short Bus DigiMAX expansion. (0xDE40/0xDE48)

`SBETFE` Boolean specifying whether the Short Bus ETFE expansion is enabled

`SBETFEbase`
: Integer specifying the Base address of the Short Bus ETFE expansion. ($de00, $de10, $df00)

`IEEE488` Boolean specifying whether the IEEE488 interface should be emulated or not.

`IEEE488Image`
: String specifying the filename of the IEEE488 ROM image.

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

`Isepicfilename`
: String specifying the filename of the ISEPIC RAM image.

`IsepicSwitch`
: Boolean specifying the status of the ISEPIC switch. (0: off, 1: on)

`IsepicImageWrite`
: Boolean, if true write back the ISEPIC image file automatically, incase the RAM contents changed, when detaching or quitting the emulator.

`LTKimage0`
`LTKimage1`
`LTKimage2`
`LTKimage3`
`LTKimage4`
`LTKimage5`
`LTKimage6`
: Strings specifying the full path to up to seven harddisk images. If a file is non-existing the drive is not emulated. The first drive is necessary.

`LTKio` Integer specifying the Base address of the I/O interface. (1: $de00, 2: $df00) Default is $df00.

`LTKport` Integer specifying the port number in multiplexed multi-computer/single-disk configurations. (0 to 15) Default is 0, the master.

`LTKserial`
: String specifying the serial number. It must be in the form `"00000000"` with all numeric digits. Default is 87000000. LTK DOS install disks are locked to their respective LTK host adapor boot ROM. This setting will override the boot ROM value. The floppy disk serial number can be changed with a sector editor on track 18, sector 18.

`MagicVoiceCartridgeEnabled`
: Boolean specifying whether the Magic Voice should be emulated or not.

`MagicVoiceImage`
: String specifying the filename of the Magic Voice ROM image.

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

`MMC64BIOSfilename`
: String specifying the filename of the MMC64 Flash ROM image.

`MMC64_bios_write`
: Boolean, if true write back the MMC64 Flash ROM image file automatically, incase the contents changed, when detaching or quitting the emulator.

`MMC64_flashjumper`
: Boolean that specifies whether the MMC64 flash jumper is set.

`MMC64_revision`
: Integer specifying the MMC64 hardware revision. (0: Revision A, 1: Revision B)

`MMC64imagefilename`
: String specifying the filename of the SD-Card image used by the MMC64 emulation.

`MMC64_RO` Boolean, if true the SD-Card image is mounted read-only.

`MMC64_sd_type`
: Integer that specifies the reported type for the emulated SD-Card. (0: Auto, 1: MMC, 2: SD, 3: SDHC)

`MMC64ClockPort`
: Integer that specifies the clockport device used. (0: None, 1: ETH64-II, 2: RRNet, 3: Silver Surfer, 4: MP3@64, 5: Catweasel MkIII SID)

`MMCRCardImage`
: String specifying the filename of the SD-Card image used by the MMCR emulation.

`MMCREEPROMImage`
: String specifying the filename of the MMCR EEPROM image.

`MMCRRescueMode`
: Boolean specifying if the rescue mode (both buttons pressed during powerup) of the MMCR is active.

`MMCRImageWrite`
: Boolean, if true write back the MMCR Flash ROM image file automatically, incase the contents changed, when detaching or quitting the emulator.

`MMCRCardRW`
: Boolean specifying if the SD-Card image used by the MMCR emulation is writeable.

`MMCRSDType`
: Integer that specifies the reported type for the emulated SD-Card. (0: Auto, 1: MMC, 2: SD, 3: SDHC)

`MMCREEPROMRW`
: Boolean specifying if the MMCR EEPROM image is writeable.

`MMCRClockPort`
: Integer that specifies the clockport device. (0: None, 1: ETH64-II, 2: RRNet, 3: Silver Surfer, 4: MP3@64, 5: Catweasel MkIII SID)

`RAMCART` Boolean specifying whether the RAMCart should be emulated or not.

`RAMCARTfilename`
: String specifying the filename of the RAMCart RAM image.

`RAMCARTImageWrite`
: Boolean, if true write back the RAMCart image file automatically, incase the RAM contents changed, when detaching or quitting the emulator.

`RAMCART_RO`
: Boolean, if true the RAMCart contents are read only.

`RAMCARTsize`
: Integer specifying the size of the RAMCart in KiB. (64, 128)

`RAMLINK`
: Boolean specifying whether the RAMLink module should be emulated or not. This option emulates the behavior of the enable/disable switch on the unit.

`RAMLINKfilename`
: String specifying the filename of the RAMCard image.

`RAMLINKImageWrite`
: Boolean, if true write back the RAMCard image file automatically, in case the RAM contents changed, when detaching or quitting the emulator.

`RAMLINKsize`
: Integer specifying the size of the RAMCard in MiB. (0..16; default is 0)

`RAMLINKmode`
: Integer specifying the mode of the RAMLink. `"1"`, the default, (Normal) only allows RAMLink access to the RAM port (REU, GEORAM, and RAMDrive); their registers are hidden otherwise. `"0"` (Direct) exposes their registers all the time. This option emulates the behavior of the Normal/Direct switch on the unit.

`RAMLINKRTCSave`
: Boolean specifying whether the RTC data should be saved when changed or not.

`RRrevision`
: Integer specifying the RR hardware revision. (0: Retro Replay, 1: Nordic Replay)

`RRFlashJumper`
: Boolean specifying whether the RR flash jumper is set or not.

`RRBankJumper`
: Boolean specifying whether the RR bank jumper is set or not.

`RRBiosWrite`
: Boolean, if true write back the RR Flash ROM image file automatically, incase the contents changed, when detaching or quitting the emulator.

`RRClockPort`
: Integer that specifies the clockport device. (0: None, 1: ETH64-II, 2: RRNet, 3: Silver Surfer, 4: MP3@64, 5: Catweasel MkIII SID)

`RRNETMK3_flashjumper`
: Boolean specifying whether the RRNETMK3 Flash Jumper is set.

`RRNETMK3_bios_write`
: Boolean specifying whether to save the RRNETMK3 bios when changed.

`SSRamExpansion`
: Boolean, if true enable the 32KiB addon RAM of the Supersnapshot V5

7.1.2.2 C64 cartridge command-line options

`-iocollision <method>`
: Select the way the I/O collisions should be handled (`IOCollisionHandling`). (0: error message and detach all involved carts, 1: error message and detach last attached involved carts, 2: warning in log and 'AND' the valid return values

`+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 (`CartridgeReset=1`, `CartridgeReset=0`).

`-cart8 <name>`
: Attach generic 8KiB cartridge image.

`-cart16 <name>`
: Attach generic 16KiB cartridge image.

`-cartultimax <name>`
: Attach generic 16KiB Ultimax cartridge image.

`-cartcrt <name>`
: Attach CRT cartridge image.

`-cartap <name>`
: Attach raw 32KiB Atomic Power cartridge image.

`-cartar2 <name>`
: Attach raw 16KiB Action Replay MK2 cartridge image.

`-cartar3 <name>`
: Attach raw 16KiB Action Replay MK3 cartridge image.

`-cartar4 <name>`
: Attach raw 32KiB Action Replay MK4 cartridge image.

`-cartar5 <name>`
: Attach raw 32KiB Action Replay cartridge image.

`-cartbis <name>`
: Attach raw 8KiB BIS-Plus cartridge image.

`-cartbb3 <name>`
: Attach raw 8KiB Blackbox V3 cartridge image.

`-cartbb4 <name>`
: Attach raw 16KiB Blackbox V4 cartridge image.

`-cartbb8 <name>`
: Attach raw 32/64KiB Blackbox V8 cartridge image.

`-cartbb9 <name>`
: Attach raw 32KiB Blackbox V9 cartridge image.

`-cartcap <name>`
: Attach raw 8KiB Capture cartridge image.

`-cartcomal <name>`
: Attach raw 64KiB 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 128KiB Dinamic cartridge image.

`-cartdsm <name>`
: Attach raw 8KiB Diashow-Maker cartridge image.

`-cartdqbb <name>`
: Attach raw 16KiB Double Quick Brown Box cartridge image.

`-dqbb`
`+dqbb`
: Enable/disable Double Quick Brown Box (`DQBB=1`, `DQBB=0`).

`-dqbbimage <name>`
: Specify Double Quick Brown Box filename (`DQBBfilename`).

`-dqbbimagerw`
`+dqbbimagerw`
: Allow/disallow writing to DQBB image (`DQBBImageWrite=1`, `DQBBImageWrite=0`).

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

`-carteasycalc <name>`
: Attach raw 24KiB Easy Calc Result cartridge image

`-easyflashjumper`
`+easyflashjumper`
: Enable/disable EasyFlash jumper (`EasyFlashJumper=1`, `EasyFlashJumper=0`).

`-easyflashcrtwrite`
`+easyflashcrtwrite`
: Allow/Disallow writing to EasyFlash .crt image (`EasyFlashWriteCRT=1`, `EasyFlashWriteCRT=0`).

`-easyflashcrtoptimize`
`+easyflashcrtoptimize`
: Allow/Disallow EasyFlash .crt image optimizing (omitting of empty banks) on write (`EasyFlashOptimizeCRT=1`, `EasyFlashOptimizeCRT=0`).

`-cartepyx <name>`
: Attach raw 8KiB Epyx FastLoad cartridge image.

`-cartexos <name>`
: Attach raw 8KiB EXOS cartridge image.

`-cartexpert <name>`
: Attach raw 8KiB Expert Cartridge image.

`-expert`
`+expert`
: Enable/Disable the Expert Cartridge (`ExpertCartridgeEnabled=1`, `ExpertCartridgeEnabled=0`).

`-expertimagename <name>`
: Set Expert Cartridge image name (`Expertfilename`).

`-expertimagerw`
`+expertimagerw`
: Allow/Disallow writing to Expert Cartridge image (`ExpertImageWrite=1`, `ExpertImageWrite=0`).

`-expertmode <mode>`
: Set Expert Cartridge mode (`ExpertCartridgeMode`). (0: off, 1: prg, 2: on)

`-cartf64 <Name>`
: Attach raw 32KiB Formel 64 image.

`-cartfc1 <name>`
: Attach raw 16KiB Final Cartridge image.

`-cartfc3 <name>`
: Attach raw 64KiB Final Cartridge III image.

`-cartfcplus <name>`
: Attach raw 32KiB Final Cartridge Plus image.

`-cartff <name>`
: Attach raw 8KiB Freeze Frame image.

`-cartfm <name>`
: Attach raw 32KiB Freeze Machine image.

`-cartfp <name>`
: Attach raw 128KiB Fun Play/Power Play cartridge image.

`-cartgmod2 <name>`
: Attach raw GMod2 cartridge image.

`-gmod2eepromimage <name>`
: Attach raw GMod2 EEPROM image (`GMod2EEPROMImage`).

`-gmod2eepromrw`
`+gmod2eepromrw`
: Enable/Disable writes to GMod2 EEPROM image (`GMod2EEPROMRW=1`, `GMod2EEPROMRW=0`).

`-gmod2flashwrite`
`+gmod2flashwrite`
: Enable/Disable saving of the GMod2 ROM at exit (`GMod2FlashWrite=1`, `GMod2FlashWrite=0`).

`-cartgmod3 <name>`
: Attach raw GMod3 cartridge image.

`-gmod3flashwrite`
`+gmod3flashwrite`
: Enable/Disable saving of the GMod3 ROM at exit (`GMod3FlashWrite=1`, `GMod3FlashWrite=0`).

`-cartgk <name>`
: Attach raw 8KiB Game Killer cartridge image.

`-cartgs <name>`
: Attach raw 512KiB Game System cartridge image.

`-carthero <name>`
: Attach raw 32KiB H.E.R.O. bootleg cartridge image

`-cartide64 <name>`
: Attach raw 64KiB or 128KiB IDE64 cartridge image.

`-IDE64image1 <name>`
`-IDE64image2 <name>`
`-IDE64image3 <name>`
`-IDE64image4 <name>`
: Specify path to the image files for IDE64 harddisks (`IDE64Image1`, `IDE64Image2`, `IDE64Image3`, `IDE64Image4`).

`-IDE64cyl1 <value>`
`-IDE64cyl2 <value>`
`-IDE64cyl3 <value>`
`-IDE64cyl4 <value>`
: Set number of cylinders for the IDE64 harddisk emulation (`IDE64Cylinders1`, `IDE64Cylinders2`, `IDE64Cylinders3`, `IDE64Cylinders1`). (1..65535)

`-IDE64hds1 <value>`
`-IDE64hds2 <value>`
`-IDE64hds3 <value>`
`-IDE64hds4 <value>`
: Set number of heads for the IDE64 harddisk emulation (`IDE64Heads1`, `IDE64Heads2`, `IDE64Heads3`, `IDE64Heads4`). (1..16)

`-IDE64sec1 <value>`
`-IDE64sec2 <value>`
`-IDE64sec3 <value>`
`-IDE64sec4 <value>`
: Set number of sectors for the IDE64 harddisk emulation (`IDE64Sectors1`, `IDE64Sectors2`, `IDE64Sectors3`, `IDE64Sectors4`). (1..63)

`-IDE64autosize1`
`+IDE64autosize1`
: Autodetect geometry of formatted image or do not autodetect and use specified geometry (`IDE64AutodetectSize1=1`, `IDE64AutodetectSize1=0`).

`-IDE64autosize2`
`+IDE64autosize2`
: Autodetect geometry of formatted image or do not autodetect and use specified geometry (`IDE64AutodetectSize2=1`, `IDE64AutodetectSize2=0`).

`-IDE64autosize3`
`+IDE64autosize3`
: Autodetect geometry of formatted image or do not autodetect and use specified geometry (`IDE64AutodetectSize3=1`, `IDE64AutodetectSize3=0`).

`-IDE64autosize4`
`+IDE64autosize4`
: Autodetect geometry of formatted image or do not autodetect and use specified geometry (`IDE64AutodetectSize4=1`, `IDE64AutodetectSize4=0`).

`-IDE64version <value>`
: Select IDE64 version V3 (0), V4.1 (1) or V4.2 (2). (`IDE64version`). (0..2)

`-IDE64USB`
`+IDE64USB`
: Enable/Disable IDE64 USB server

`-IDE64USBAddress <name>`
: The local address the IDE64 USB server should bind to

`-IDE64rtcsave`
`+IDE64rtcsave`
: Enable/disable saving of IDE64 RTC data when changed (`IDE64RTCSave=1`, `IDE64RTCSave=0`).

`-ide64clockportdevice`
: Enable IDE64 Clockport device (0: None, 2: RRNet, 4: MP3@64) (`IDE64ClockPort`).

`-sbdigimax`
`+sbdigimax`
: Enable/Disable the Short Bus DigiMAX expansion (`SBDIGIMAX`).

`-sbdigimaxbase`
: Set Base address of the Short Bus DigiMAX expansion (0xDE40/0xDE48) (`SBDIGIMAXbase`).

`-sbetfe`
`+sbetfe` Enable/Disable the Short Bus ETFE expansion (`SBETFE`).

`-sbetfebase`
: Set Base address of the Short Bus ETFE expansion (56832: $de00, 56848: $de10, 57088: $df00) (`SBETFEbase`).

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

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

`-ieee488image <name>`
: Set IEEE488 interface image name (`IEEE488Image`).

`-isepic`
`+isepic`
: Enable/disable the ISEPIC cart (`IsepicCartridgeEnabled=1`, `IsepicCartridgeEnabled=0`).

`-isepicswitch`
`+isepicswitch`
: Enable/disable the ISEPIC switch (`IsepicSwitch=1`, `IsepicSwitch=0`).

`-cartisepic <name>`
: Attach raw 2KiB ISEPIC cartridge image.

`-isepicimagename <name>`
: Set ISEPIC image name (`Isepicfilename`).

`-isepicimagerw`
`+isepicimagerw`
: Allow/disallow writing to ISEPIC image (`IsepicImageWrite=1`, `IsepicImageWrite=0`).

`-cartkcs <name>`
: Attach raw 16KiB KCS Power cartridge image.

`-cartks <name>`
: Attach raw 24KiB Kingsoft cartridge image.

`-cartltk <name>`
: Attach raw Lt. Kernal Host Adaptor cartridge image.

`-ltkimage0 <name>`
`-ltkimage1 <name>`
`-ltkimage2 <name>`
`-ltkimage3 <name>`
`-ltkimage4 <name>`
`-ltkimage5 <name>`
`-ltkimage6 <name>`
: Strings specifying the full path to up to seven harddisk images. If a file is non-existing the drive is not emulated. The first drive is necessary. (`ltkimage0`, `ltkimage1`, `ltkimage2`, `ltkimage3`, `ltkimage4`, `ltkimage5`, `ltkimage6`).

`-ltkio <value>`
: Integer specifying the Base address of the I/O interface. (1: $de00, 2: $df00) Default is $df00.

`-ltkport <value>`
: Integer specifying the port number in multiplexed multi-computer/single-disk configurations. (0..15) Default is 0, the master.

`-ltkserial <name>`
: String specifying the serial number. It must be in the form "00000000" with all numeric digits. Default is 87000000. LTK DOS install disks are locked to their respective LTK host adapor boot ROM. This setting will override the boot

ROM value. The floppy disk serial number can be changed with a sector editor on track 18, sector 18.

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

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

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

`-cartmax <name>`
Attach raw MAX Basic cartridge image.

`-cartmm <name>`
Attach raw MultiMAX cartridge image.

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

`-mmc64`
`+mmc64` Enable/disable the MMC64 expansion (`MMC64=1`, `MMC64=0`).

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

`-mmc64bios <name>`
Specify name of MMC64 BIOS image (`MMC64BIOSfilename`).

`-mmc64image <name>`
Specify name of MMC64 image (`MMC64imagefilename`).

`-mmc64readonly`
Set the MMC64 card to read-only (`MMC64_RO=1`).

`-mmc64readwrite`
Set the MMC64 card to read/write (`MMC64_RO=0`).

`-mmc64flash`
`+mmc64flash`
Enable/Disable the MMC64 flash jumper (`MMC64_flashjumper=1`, `MMC64_flashjumper=0`).

`-mmc64bioswrite`
Save the MMC64 bios when changed (`MMC64_bios_write`=1).

`-mmc64biosreadonly`
Do not save the MMC64 bios when changed (`MMC64_bios_write`=0).

`-mmc64rev <revision>`
Specify MMC64 revision (`MMC64_revision`). (0: Revision A, 1: Revision B)

`-mmc64sdtype <type>`
Specify MMC64 SD type (`MMC64_sd_type`). (0: Auto, 1: MMC, 2: SD, 3: SDHC)

`-mmc64clockportdevice <device>`
: Set MMC64 clockport device (`MMC64ClockPort`) (0: None, 1: ETH64-II, 2: RRNet, 3: Silver Surfer, 4: MP3@64, 5: Catweasel MkIII SID)

`-cartmmcr <name>`
: Attach raw 512KiB MMC Replay cartridge image.

`-mmcrrescue`
`+mmcrrescue`
: Enable/disable MMC Replay rescue mode (`MMCRRescueMode=1`, `MMCRRescueMode=0`).

`-mmcrimagerw`
`+mmcrimagerw`
: Allow/disallow writing to MMC Replay image (`MMCRImageWrite=1`, `MMCRImageWrite=0`).

`-mmcrsdtype <type>`
: Specify MMC Replay SD type (`MMCRSDType`). (0: Auto, 1: MMC, 2: SD, 3: SDHC)

`-mmcrcardimage <filename>`
: Specify MMC Replay card image filename (`MMCRCardImage`).

`-mmcrcardrw`
`+mmcrcardrw`
: Allow/disallow writes to MMC Replay card image (`MMCRCardRW=1`, `MMCRCardRW=0`).

`-mmcreepromimage <filename>`
: Specify MMC Replay EEPROM image filename (`MMCREEPROMImage`).

`-mmcreepromrw`
`+mmcreepromrw`
: Allow/disallow writes to MMC Replay EEPROM image (`MMCREEPROMRW=1`, `MMCREEPROMRW=0`).

`-mmcrclockportdevice <id>`
: Set MMC Replay clockport device (`MMCRClockPort`). (0: None, 1: ETH64-II, 2: RRNet, 3: Silver Surfer, 4: MP3@64, 5: Catweasel MkIII SID)

`-cartmv <name>`
: Attach raw 16KiB Magic Voice cartridge image.

`-magicvoiceimage <name>`
: Specify Magic Voice cartridge ROM image filename (`MagicVoiceImage`).

`-magicvoice`
`+magicvoice`
: Enable/disable Magic Voice cartridge (`MagicVoiceCartridgeEnabled=1`, `MagicVoiceCartridgeEnabled=0`).

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

`-cartp64 <name>`
: Attach raw 256KiB Prophet 64 cartridge image.

`-cartpf <name>`
: Attach raw 64KiB Pagefox cartridge image.

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

`-ramcart`
`+ramcart` Enable/disable the RAMCART expansion (`RAMCART=1`, `RAMCART=0`).

`-ramcartsize <size in KiB>`
: Size of the RAMCART expansion (`RAMCARTsize`). (64, 128)

`-ramcartimage <name>`
: Specify name of RAMCART image (`RAMCARTfilename`).

`-ramcartimagerw`
`+ramcartimagerw`
: Allow/disallow writing to RAMCart image (`RAMCARTImageWrite=1`, `RAMCARTImageWrite=0`).

`-ramcartro`
: Set the RamCart switch to read-only (`RAMCART_RO=1`).

`-ramcartrw`
: Set the RamCart switch to read-only (`RAMCART_RO=0`).

`-ramlink`
`+ramlink` Enable/disable emulation of the RAMLink module. This option emulates the behavior of the enable/disable switch on the unit (`RAMLINK=1`, `RAMLINK=0`).

`-cartramlink <name>`
: Attach the raw 64 KiB ROM image.

`-ramlinkimage <name>`
: Specify name of RAMLink RAMCard image (`RAMLINKfilename`).

`-ramlinkimagerw`
`+ramlinkimagerw`
: Allow/disallow writing to RAMCard image on detach (`RAMLINKImageWrite=1`, `RAMLINKImageWrite=0`).

`-ramlinksize <size in MiB>`
: Size of the RAMCard unit (`RAMLINKsize`). (0..16; default is 0)

`-ramlinkmode <value>`
: Sets the RAM port mode of the unit. "1", the default, (Normal) only allows RAMLink access to the RAM port (REU, GEORAM, and RAMDrive); their registers are hidden otherwise. "0" (Direct) exposes their registers all the time. This option emulates the behavior of the Normal/Direct switch on the unit. (`RAMLINKmode`). (1:normal (default), 0:direct)

`-ramlinkrtcsave`
`+ramlinkrtcsave`
: Enable/disable saving of the RTC data when changed (`RAMLINKRTCSave=1`, `RAMLINKRTCSave=0`).

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

`-cartrgcd <Name>`
: Attach raw 64KiB RGCD cartridge image.

`-cartross <name>`
: Attach raw 16/32KiB ROSS cartridge image.

`-cartrrnet <name>`
: Attach raw 8KiB RR-Net MK3 cartridge image.

`-cartrr <name>`
: Attach raw 64KiB Retro Replay cartridge image.

`-rrbioswrite`
`+rrbioswrite`
: Enable/disable saving of the RR ROM at exit (`RRBiosWrite=1`, `RRBiosWrite=0`).

`-rrbankjumper`
`+rrbankjumper`
: Set/unset RR Bank Jumper (`RRBankJumper=1`, `RRBankJumper=0`).

`-rrflashjumper`
`+rrflashjumper`
: Set/unset RR Flash Jumper (`RRFlashJumper=1`, `RRFlashJumper=0`).

`-rrrev <Revision>`
: Set the RR revision (`RRrevision`). (0: Retro Replay, 1: Nordic Replay)

`-rrclockportdevice <device>`
: Set the RR clockport device (`RRClockPort`) (0: None, 1: ETH64-II, 2: RRNet, 3: Silver Surfer, 4: MP3@64, 5: Catweasel MkIII SID)

`-rrnetmk3flash`
`+rrnetmk3flash`
: SetRemove the RRNETMK3 Flash Jumper (`RRNETMK3_flashjumper=1`, `RRNETMK3_flashjumper=0`).

`-rrnetmk3bioswrite`
`+rrnetmk3bioswrite`
: Save/Do not save the RRNETMK3 bios when changed (`RRNETMK3_bios_write=1`, `RRNETMK3_bios_write=0`).

`-cartrrf <name>`
: Attach raw 8KiB REX RAM-Floppy cartridge image.

`-cartru <name>`
: Attach raw 8KiB REX Utility cartridge image.

`-cartsdbox <name>`
: Attach raw 128KiB SD-BOX cartridge image.

`-carts64 <name>`
: Attach raw 4KiB Snapshot 64 cartridge image.

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

`-cartse5 <name>`
: Attach raw 16KiB Super Explode V5 cartridge image.

`-cartsg <name>`
: Attach raw 64KiB Super Games cartridge image.

`-cartsilver <Name>`
: Attach raw Silverrock 128 cartridge image.

`-cartsimon <name>`
: Attach raw 16KiB Simons' Basic cartridge image.

`-cartss4 <name>`
: Attach raw 32KiB Super Snapshot V4 cartridge image.

`-cartss5 <name>`
: Attach raw 64KiB Super Snapshot V5 cartridge image.

`-ssramexpansion`
`+ssramexpansion`
: Enable/disable 32KiB addon RAM. (`SSRamExpansion=1`, `SSRamExpansion=0`).

`-cartstar <name>`
: Attach raw 16KiB Stardos cartridge image.

`-cartwl <name>`
: Attach raw 16KiB Westermann Learning cartridge image.

`-cartws <name>`
: Attach raw 8KiB Warp Speed cartridge image.

`-cartzaxxon <name>`
: Attach raw 16KiB Zaxxon cartridge image.

`-cartzipp <name>`
: Attach raw 8KiB ZIPP-CODE 48 cartridge image.

7.1.3 CIA settings

7.1.3.1 CIA resources

`CIA1Model`
: Integer specifying CIA1 model (all emulators except x64dtv, xpet, xplus4, xvic). (0: old 6526, 1: new 6526A)

`CIA2Model`
: Integer specifying CIA2 model (all emulators except x64dtv, xcbm2, xcbm5x0, xpet, xplus4, xvic). (0: old 6526, 1: new 6526A)

7.1.3.2 CIA command-line options

`-ciamodel <model>`
: Set both CIA models (`CIA1Model`, `CIA2Model`) (all emulators except x64dtv, xcbm2, xcbm5x0, xpet, xplus4, xvic). (0: old 6526, 1: new 6526A)

`-cia1model <model>`
: Set CIA1 model (`CIA1Model`) (all emulators except x64dtv, xpet, xplus4, xvic). (0: old 6526, 1: new 6526A)

`-cia2model <model>`
: Set CIA2 model (`CIA2Model`) (all emulators except x64dtv, xcbm2, xcbm5x0, xpet, xplus4, xvic). (0: old 6526, 1: new 6526A)

7.1.4 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, for example:
  - `pepto-pal.vpl` ("Pepto PAL"), a Palette calculated by Philip "Pepto" Timmermann (`http://www.pepto.de/projects/colorvic/`). (This is the default)
  - `vice.vpl` ("VICE"), the old 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 Håkan Sundell.
  - `frodo.vpl` ("Frodo"), palette taken from the free Frodo emulator by Christian Bauer (`https://frodo.cebix.net/`).
  - `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 (`https://www.godot64.de`).

7.1.4.1 VIC-II resources

`VICIIModel`
: Integer that specifies VIC-II model (x64sc, xscpu64 only). (6569, 6569r1, 8565, 6567, 8562, 6567r56a, 6572)

`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.

`VICIIVSPBug`
: Boolean specifying whether the "VSP Bug" must be emulated (x64sc, xscpu64 only).

`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.

`VICIINewLuminances`
: Boolean specifying whether to use new (9 steps) luminances.

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

`VICIIExternalPalette`
: Boolean specifying whether to use external palette file or not.

`VICIIColorSaturation`
: Integer specifying saturation of internal calculated palette. (0..2000)

`VICIIColorContrast`
: Integer specifying contrast of internal calculated palette. (0..2000)

`VICIIColorBrightness`
: Integer specifying brightness of internal calculated palette. (0..2000)

`VICIIColorGamma`
: Integer specifying gamma of internal calculated palette. (0..4000)

`VICIIColorTint`
: Integer specifying tint of internal calculated palette. (0..2000)

`VICIIPALScanLineShade`
: Integer specifying amount of scan line shading for the CRT emulation. (0..1000)

`VICIIPALBlur`
: Integer specifying amount of horizontal blur for the CRT emulation. (0..1000)

`VICIIPALOddLinePhase`
: Integer specifying phase for color carrier in odd lines. (0..2000)

`VICIIPALOddLineOffset`
: Integer specifying phase offset for color carrier in odd lines. (0..2000)

`VICIIAudioLeak`
: Boolean specifying whether to enable/disable video to audio leak emulation.

`VICIIFilter`
: Integer specifying rendering filter. (0: None, 1: CRT emulation, 2: Scale2x)

`VICIIBorderMode`
: Integer specifying border display mode. (0: normal, 1: full, 2: debug, 3: none)

7.1.4.2 VIC-II command-line options

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

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

`-VICIIvspbug`
`+VICIIvspbug`
: Enable/disable emulation of the "VSP bug" (`VICIIVSPBug=1`, `VICIIVSPBug=0`) (x64sc, xscpu64 only).

`-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`).

`-VICIIfilter <Mode>`
: Select rendering filter (`VICIIFilter`). (0: None, 1: CRT emulation, 2: Scale2x)

`-VICIIintpal`
: Use an internal calculated palette (`VICIIExternalPalette=0`).

`-VICIIextpal`
: Use an external palette (file) (`VICIIExternalPalette=1`).

`-VICIIpalette <Name>`
: Specify the name of the palette file (`VICIIPaletteFile`).

`-VICIIborders <mode>`
: Set VIC-II border display mode (`VICIIBorderMode`). (0: normal, 1: full, 2: debug, 3: none)

`-VICIImodel <model>`
: Set VIC-II model (`VICIIModel`) (x64sc and xscpu64 only). (6569, 6569r1, 8565, 6567, 8562, 6567r56a, 6572)

`-VICIInewluminance`
`+VICIInewluminance`
: Enable/disable new luminances (`VICIINewLuminances=1`, `VICIINewLuminances=0`).

`-VICIIsaturation <0-2000>`
: Set saturation of internal calculated palette (`VICIIColorSaturation`).

`-VICIIcontrast <0-2000>`
: Set contrast of internal calculated palette (`VICIIColorContrast`).

`-VICIIbrightness <0-2000>`
: Set brightness of internal calculated palette (`VICIIColorBrightness`).

`-VICIIgamma <0-4000>`
: Set gamma of internal calculated palette (`VICIIColorGamma`).

`-VICIItint <0-2000>`
: Set tint of internal calculated palette (`VICIIColorTint`).

`-VICIIoddlinesphase <0-2000>`
: Set phase for color carrier in odd lines (`VICIIPALOddLinePhase`).

`-VICIIoddlinesoffset <0-2000>`
: Set phase offset for color carrier in odd lines (`VICIIPALOddLineOffset`).

`-VICIIcrtblur <0-1000>`
: Amount of horizontal blur for the CRT emulation (`VICIIPALBlur`).

`-VICIIcrtscanlineshade <0-1000>`
: Amount of scan line shading for the CRT emulation (`VICIIPALScanLineShade`).

`-VICIIaudioleak`
`+VICIIaudioleak`
: Enable/disable video to audio leak emulation (`VICIIAudioLeak=1`, `VICIIAudioLeak=0`).

7.1.5 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. Additionally there is "8580D", which refers to MOS8580 with added "Digifix" modification, which adds a DC offset to audio-in, which makes classic samples audible also on the 8580.
- “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).

7.1.5.1 SID resources

`SidStereo`

Integer specifying the amount of emulated extra SIDs. (0: off, 1: 1 extra sid, 2: 2 extra sids, 3: three extra sids, 4: four extra sids, 5: five extra sids, 6: six extra sids, 7: seven extra sids)

`Sid2AddressStart`

Integer specifying the base address of the second SID (x64, x64sc, xscpu64, x128 and vsid only). (x128: 0xD420, 0xD440, 0xD460, 0xD480, 0xD4A0, 0xD4C0, 0xD4E0, 0xD700, 0xD720, 0xD740, 0xD760, 0xD780, 0xD7A0, 0xD7C0, 0xD7E0, 0xDE00, 0xDE20, 0xDE40, 0xDE60, 0xDE80, 0xDEA0, 0xDEC0, 0xDEE0, 0xDF00, 0xDF20, 0xDF40, 0xDF60, 0xDF80, 0xDFA0, 0xDFC0, 0xDFE0) (x64, x64sc, xscpu64, vsid: 0xD420, 0xD440, 0xD460, 0xD480, 0xD4A0, 0xD4C0, 0xD4E0, 0xD500, 0xD520, 0xD540, 0xD560, 0xD580, 0xD5A0, 0xD5C0, 0xD5E0, 0xD600, 0xD620, 0xD640, 0xD660, 0xD680, 0xD6A0, 0xD6C0, 0xD6E0, 0xD700, 0xD720, 0xD740, 0xD760, 0xD780, 0xD7A0, 0xD7C0, 0xD7E0, 0xDE00, 0xDE20, 0xDE40, 0xDE60,

0xDE80, 0xDEA0, 0xDEC0, 0xDEE0, 0xDF00, 0xDF20, 0xDF40, 0xDF60, 0xDF80, 0xDFA0, 0xDFC0, 0xDFE0)

`Sid3AddressStart`
: Integer specifying the base address of the third SID (x64, x64sc, xscpu64, x128 and vsid only). (x128: 0xD420, 0xD440, 0xD460, 0xD480, 0xD4A0, 0xD4C0, 0xD4E0, 0xD700, 0xD720, 0xD740, 0xD760, 0xD780, 0xD7A0, 0xD7C0, 0xD7E0, 0xDE00, 0xDE20, 0xDE40, 0xDE60, 0xDE80, 0xDEA0, 0xDEC0, 0xDEE0, 0xDF00, 0xDF20, 0xDF40, 0xDF60, 0xDF80, 0xDFA0, 0xDFC0, 0xDFE0) (x64, x64sc, xscpu64, vsid: 0xD420, 0xD440, 0xD460, 0xD480, 0xD4A0, 0xD4C0, 0xD4E0, 0xD500, 0xD520, 0xD540, 0xD560, 0xD580, 0xD5A0, 0xD5C0, 0xD5E0, 0xD600, 0xD620, 0xD640, 0xD660, 0xD680, 0xD6A0, 0xD6C0, 0xD6E0, 0xD700, 0xD720, 0xD740, 0xD760, 0xD780, 0xD7A0, 0xD7C0, 0xD7E0, 0xDE00, 0xDE20, 0xDE40, 0xDE60, 0xDE80, 0xDEA0, 0xDEC0, 0xDEE0, 0xDF00, 0xDF20, 0xDF40, 0xDF60, 0xDF80, 0xDFA0, 0xDFC0, 0xDFE0)

`Sid4AddressStart`
: Integer specifying the base address of the fourth SID (x64, x64sc, xscpu64, x128 and vsid only). (x128: 0xD420, 0xD440, 0xD460, 0xD480, 0xD4A0, 0xD4C0, 0xD4E0, 0xD700, 0xD720, 0xD740, 0xD760, 0xD780, 0xD7A0, 0xD7C0, 0xD7E0, 0xDE00, 0xDE20, 0xDE40, 0xDE60, 0xDE80, 0xDEA0, 0xDEC0, 0xDEE0, 0xDF00, 0xDF20, 0xDF40, 0xDF60, 0xDF80, 0xDFA0, 0xDFC0, 0xDFE0) (x64, x64sc, xscpu64, vsid: 0xD420, 0xD440, 0xD460, 0xD480, 0xD4A0, 0xD4C0, 0xD4E0, 0xD500, 0xD520, 0xD540, 0xD560, 0xD580, 0xD5A0, 0xD5C0, 0xD5E0, 0xD600, 0xD620, 0xD640, 0xD660, 0xD680, 0xD6A0, 0xD6C0, 0xD6E0, 0xD700, 0xD720, 0xD740, 0xD760, 0xD780, 0xD7A0, 0xD7C0, 0xD7E0, 0xDE00, 0xDE20, 0xDE40, 0xDE60, 0xDE80, 0xDEA0, 0xDEC0, 0xDEE0, 0xDF00, 0xDF20, 0xDF40, 0xDF60, 0xDF80, 0xDFA0, 0xDFC0, 0xDFE0)

`Sid5AddressStart`
: Integer specifying the base address of the fifth SID (x64, x64sc, xscpu64, x128 and vsid only). (x128: 0xD420, 0xD440, 0xD460, 0xD480, 0xD4A0, 0xD4C0, 0xD4E0, 0xD700, 0xD720, 0xD740, 0xD760, 0xD780, 0xD7A0, 0xD7C0, 0xD7E0, 0xDE00, 0xDE20, 0xDE40, 0xDE60, 0xDE80, 0xDEA0, 0xDEC0, 0xDEE0, 0xDF00, 0xDF20, 0xDF40, 0xDF60, 0xDF80, 0xDFA0, 0xDFC0, 0xDFE0) (x64, x64sc, xscpu64, vsid: 0xD420, 0xD440, 0xD460, 0xD480, 0xD4A0, 0xD4C0, 0xD4E0, 0xD500, 0xD520, 0xD540, 0xD560, 0xD580, 0xD5A0, 0xD5C0, 0xD5E0, 0xD600, 0xD620, 0xD640, 0xD660, 0xD680, 0xD6A0, 0xD6C0, 0xD6E0, 0xD700, 0xD720, 0xD740, 0xD760, 0xD780, 0xD7A0, 0xD7C0, 0xD7E0, 0xDE00, 0xDE20, 0xDE40, 0xDE60, 0xDE80, 0xDEA0, 0xDEC0, 0xDEE0, 0xDF00, 0xDF20, 0xDF40, 0xDF60, 0xDF80, 0xDFA0, 0xDFC0, 0xDFE0)

`Sid6AddressStart`
: Integer specifying the base address of the sixth SID (x64, x64sc, xscpu64, x128 and vsid only). (x128: 0xD420, 0xD440, 0xD460, 0xD480, 0xD4A0, 0xD4C0, 0xD4E0, 0xD700, 0xD720, 0xD740, 0xD760, 0xD780, 0xD7A0, 0xD7C0,

0xD7E0, 0xDE00, 0xDE20, 0xDE40, 0xDE60, 0xDE80, 0xDEA0, 0xDEC0, 0xDEE0, 0xDF00, 0xDF20, 0xDF40, 0xDF60, 0xDF80, 0xDFA0, 0xDFC0, 0xDFE0) (x64, x64sc, xscpu64, vsid: 0xD420, 0xD440, 0xD460, 0xD480, 0xD4A0, 0xD4C0, 0xD4E0, 0xD500, 0xD520, 0xD540, 0xD560, 0xD580, 0xD5A0, 0xD5C0, 0xD5E0, 0xD600, 0xD620, 0xD640, 0xD660, 0xD680, 0xD6A0, 0xD6C0, 0xD6E0, 0xD700, 0xD720, 0xD740, 0xD760, 0xD780, 0xD7A0, 0xD7C0, 0xD7E0, 0xDE00, 0xDE20, 0xDE40, 0xDE60, 0xDE80, 0xDEA0, 0xDEC0, 0xDEE0, 0xDF00, 0xDF20, 0xDF40, 0xDF60, 0xDF80, 0xDFA0, 0xDFC0, 0xDFE0)

`Sid7AddressStart`
: Integer specifying the base address of the seventh SID (x64, x64sc, xscpu64, x128 and vsid only). (x128: 0xD420, 0xD440, 0xD460, 0xD480, 0xD4A0, 0xD4C0, 0xD4E0, 0xD700, 0xD720, 0xD740, 0xD760, 0xD780, 0xD7A0, 0xD7C0, 0xD7E0, 0xDE00, 0xDE20, 0xDE40, 0xDE60, 0xDE80, 0xDEA0, 0xDEC0, 0xDEE0, 0xDF00, 0xDF20, 0xDF40, 0xDF60, 0xDF80, 0xDFA0, 0xDFC0, 0xDFE0) (x64, x64sc, xscpu64, vsid: 0xD420, 0xD440, 0xD460, 0xD480, 0xD4A0, 0xD4C0, 0xD4E0, 0xD500, 0xD520, 0xD540, 0xD560, 0xD580, 0xD5A0, 0xD5C0, 0xD5E0, 0xD600, 0xD620, 0xD640, 0xD660, 0xD680, 0xD6A0, 0xD6C0, 0xD6E0, 0xD700, 0xD720, 0xD740, 0xD760, 0xD780, 0xD7A0, 0xD7C0, 0xD7E0, 0xDE00, 0xDE20, 0xDE40, 0xDE60, 0xDE80, 0xDEA0, 0xDEC0, 0xDEE0, 0xDF00, 0xDF20, 0xDF40, 0xDF60, 0xDF80, 0xDFA0, 0xDFC0, 0xDFE0)

`Sid8AddressStart`
: Integer specifying the base address of the eigth SID (x64, x64sc, xscpu64, x128 and vsid only). (x128: 0xD420, 0xD440, 0xD460, 0xD480, 0xD4A0, 0xD4C0, 0xD4E0, 0xD700, 0xD720, 0xD740, 0xD760, 0xD780, 0xD7A0, 0xD7C0, 0xD7E0, 0xDE00, 0xDE20, 0xDE40, 0xDE60, 0xDE80, 0xDEA0, 0xDEC0, 0xDEE0, 0xDF00, 0xDF20, 0xDF40, 0xDF60, 0xDF80, 0xDFA0, 0xDFC0, 0xDFE0) (x64, x64sc, xscpu64, vsid: 0xD420, 0xD440, 0xD460, 0xD480, 0xD4A0, 0xD4C0, 0xD4E0, 0xD500, 0xD520, 0xD540, 0xD560, 0xD580, 0xD5A0, 0xD5C0, 0xD5E0, 0xD600, 0xD620, 0xD640, 0xD660, 0xD680, 0xD6A0, 0xD6C0, 0xD6E0, 0xD700, 0xD720, 0xD740, 0xD760, 0xD780, 0xD7A0, 0xD7C0, 0xD7E0, 0xDE00, 0xDE20, 0xDE40, 0xDE60, 0xDE80, 0xDEA0, 0xDEC0, 0xDEE0, 0xDF00, 0xDF20, 0xDF40, 0xDF60, 0xDF80, 0xDFA0, 0xDFC0, 0xDFE0)

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

`SidModel` Integer specifying what model of the SID must be emulated. (0: 6581, 1: 8580, 2: 8580D, 3: DTVSID)

`SidEngine`
: Integer specifying what SID engine will be used. (0: FastSID, 1: ReSID, 2: Catweasel MKIII, 3: HardSID, 4: ParSID Port 1, 5: ParSID Port 2, 6: ParSID Port 3)

`SidResidSampling`
: Integer specifying the sampling method (`0`: Fast, `1`: Interpolation, `2`: Resampling, `3`: Fast Resampling)

`SidResidPassband`
: Integer specifying the resampling filter passband in percentage of the total bandwidth (`0 - 90`) for 6581.

`SidResidGain`
: Integer that specifies reSID gain in percent [97] (90..100) for 6581.

`SidResidFilterBias`
: Integer that specifies reSID filter bias for 6581, which can be used to adjust DAC bias in millivolts. [0] (-5000..5000)

`SidResid8580Passband`
: Integer specifying the resampling filter passband in percentage of the total bandwidth (`0 - 90`) for 8580.

`SidResid8580Gain`
: Integer that specifies reSID gain in percent [97] (90..100) for 8580.

`SidResid8580FilterBias`
: Integer that specifies reSID filter bias for 8580, which can be used to adjust DAC bias in millivolts. [0] (-5000..5000)

7.1.5.2 SID command-line options

`-sidextra`
: Specify the amount of extra SID chips to emulate (`SidStereo`). (0: off, 1: 1 extra sid, 2: 2 extra sids, 3: 3 extra sids, 4: 4 extra sids, 5: 5 extra sids, 6: 6 extra sids, 7: 7 extra sids)

`-sid2address <Base address>`
: Specifies the start address for the second SID chip (`Sid2AddressStart`) (x64, x64sc, xscpu64, x128 and vsid only). (x128: 0xD420, 0xD440, 0xD460, 0xD480, 0xD4A0, 0xD4C0, 0xD4E0, 0xD700, 0xD720, 0xD740, 0xD760, 0xD780, 0xD7A0, 0xD7C0, 0xD7E0, 0xDE00, 0xDE20, 0xDE40, 0xDE60, 0xDE80, 0xDEA0, 0xDEC0, 0xDEE0, 0xDF00, 0xDF20, 0xDF40, 0xDF60, 0xDF80, 0xDFA0, 0xDFC0, 0xDFE0) (x64, x64sc, xscpu64, vsid: 0xD420, 0xD440, 0xD460, 0xD480, 0xD4A0, 0xD4C0, 0xD4E0, 0xD500, 0xD520, 0xD540, 0xD560, 0xD580, 0xD5A0, 0xD5C0, 0xD5E0, 0xD600, 0xD620, 0xD640, 0xD660, 0xD680, 0xD6A0, 0xD6C0, 0xD6E0, 0xD700, 0xD720, 0xD740, 0xD760, 0xD780, 0xD7A0, 0xD7C0, 0xD7E0, 0xDE00, 0xDE20, 0xDE40, 0xDE60, 0xDE80, 0xDEA0, 0xDEC0, 0xDEE0, 0xDF00, 0xDF20, 0xDF40, 0xDF60, 0xDF80, 0xDFA0, 0xDFC0, 0xDFE0)

`-sid3address ADDRESS`
: Specifies the start address for the third SID chip (`Sid3AddressStart`) (x64, x64sc, xscpu64, x128 and vsid only). (x128: 0xD420, 0xD440, 0xD460, 0xD480, 0xD4A0, 0xD4C0, 0xD4E0, 0xD700, 0xD720, 0xD740, 0xD760, 0xD780, 0xD7A0, 0xD7C0, 0xD7E0, 0xDE00, 0xDE20, 0xDE40, 0xDE60,

0xDE80, 0xDEA0, 0xDEC0, 0xDEE0, 0xDF00, 0xDF20, 0xDF40, 0xDF60, 0xDF80, 0xDFA0, 0xDFC0, 0xDFE0) (x64, x64sc, xscpu64, vsid: 0xD420, 0xD440, 0xD460, 0xD480, 0xD4A0, 0xD4C0, 0xD4E0, 0xD500, 0xD520, 0xD540, 0xD560, 0xD580, 0xD5A0, 0xD5C0, 0xD5E0, 0xD600, 0xD620, 0xD640, 0xD660, 0xD680, 0xD6A0, 0xD6C0, 0xD6E0, 0xD700, 0xD720, 0xD740, 0xD760, 0xD780, 0xD7A0, 0xD7C0, 0xD7E0, 0xDE00, 0xDE20, 0xDE40, 0xDE60, 0xDE80, 0xDEA0, 0xDEC0, 0xDEE0, 0xDF00, 0xDF20, 0xDF40, 0xDF60, 0xDF80, 0xDFA0, 0xDFC0, 0xDFE0)

`-sid4address ADDRESS`
Specifies the start address for the fourth SID chip (`Sid4AddressStart`) (x64, x64sc, xscpu64, x128 and vsid only). (x128: 0xD420, 0xD440, 0xD460, 0xD480, 0xD4A0, 0xD4C0, 0xD4E0, 0xD700, 0xD720, 0xD740, 0xD760, 0xD780, 0xD7A0, 0xD7C0, 0xD7E0, 0xDE00, 0xDE20, 0xDE40, 0xDE60, 0xDE80, 0xDEA0, 0xDEC0, 0xDEE0, 0xDF00, 0xDF20, 0xDF40, 0xDF60, 0xDF80, 0xDFA0, 0xDFC0, 0xDFE0) (x64, x64sc, xscpu64, vsid: 0xD420, 0xD440, 0xD460, 0xD480, 0xD4A0, 0xD4C0, 0xD4E0, 0xD500, 0xD520, 0xD540, 0xD560, 0xD580, 0xD5A0, 0xD5C0, 0xD5E0, 0xD600, 0xD620, 0xD640, 0xD660, 0xD680, 0xD6A0, 0xD6C0, 0xD6E0, 0xD700, 0xD720, 0xD740, 0xD760, 0xD780, 0xD7A0, 0xD7C0, 0xD7E0, 0xDE00, 0xDE20, 0xDE40, 0xDE60, 0xDE80, 0xDEA0, 0xDEC0, 0xDEE0, 0xDF00, 0xDF20, 0xDF40, 0xDF60, 0xDF80, 0xDFA0, 0xDFC0, 0xDFE0)

`-sid5address ADDRESS`
Specifies the start address for the fifth SID chip (`Sid5AddressStart`) (x64, x64sc, xscpu64, x128 and vsid only). (x128: 0xD420, 0xD440, 0xD460, 0xD480, 0xD4A0, 0xD4C0, 0xD4E0, 0xD700, 0xD720, 0xD740, 0xD760, 0xD780, 0xD7A0, 0xD7C0, 0xD7E0, 0xDE00, 0xDE20, 0xDE40, 0xDE60, 0xDE80, 0xDEA0, 0xDEC0, 0xDEE0, 0xDF00, 0xDF20, 0xDF40, 0xDF60, 0xDF80, 0xDFA0, 0xDFC0, 0xDFE0) (x64, x64sc, xscpu64, vsid: 0xD420, 0xD440, 0xD460, 0xD480, 0xD4A0, 0xD4C0, 0xD4E0, 0xD500, 0xD520, 0xD540, 0xD560, 0xD580, 0xD5A0, 0xD5C0, 0xD5E0, 0xD600, 0xD620, 0xD640, 0xD660, 0xD680, 0xD6A0, 0xD6C0, 0xD6E0, 0xD700, 0xD720, 0xD740, 0xD760, 0xD780, 0xD7A0, 0xD7C0, 0xD7E0, 0xDE00, 0xDE20, 0xDE40, 0xDE60, 0xDE80, 0xDEA0, 0xDEC0, 0xDEE0, 0xDF00, 0xDF20, 0xDF40, 0xDF60, 0xDF80, 0xDFA0, 0xDFC0, 0xDFE0)

`-sid6address ADDRESS`
Specifies the start address for the sixth SID chip (`Sid6AddressStart`) (x64, x64sc, xscpu64, x128 and vsid only). (x128: 0xD420, 0xD440, 0xD460, 0xD480, 0xD4A0, 0xD4C0, 0xD4E0, 0xD700, 0xD720, 0xD740, 0xD760, 0xD780, 0xD7A0, 0xD7C0, 0xD7E0, 0xDE00, 0xDE20, 0xDE40, 0xDE60, 0xDE80, 0xDEA0, 0xDEC0, 0xDEE0, 0xDF00, 0xDF20, 0xDF40, 0xDF60, 0xDF80, 0xDFA0, 0xDFC0, 0xDFE0) (x64, x64sc, xscpu64, vsid: 0xD420, 0xD440, 0xD460, 0xD480, 0xD4A0, 0xD4C0, 0xD4E0, 0xD500, 0xD520, 0xD540, 0xD560, 0xD580, 0xD5A0, 0xD5C0, 0xD5E0, 0xD600, 0xD620, 0xD640, 0xD660, 0xD680, 0xD6A0, 0xD6C0, 0xD6E0, 0xD700, 0xD720, 0xD740, 0xD760, 0xD780, 0xD7A0, 0xD7C0, 0xD7E0, 0xDE00, 0xDE20,

0xDE40, 0xDE60, 0xDE80, 0xDEA0, 0xDEC0, 0xDEE0, 0xDF00, 0xDF20, 0xDF40, 0xDF60, 0xDF80, 0xDFA0, 0xDFC0, 0xDFE0)

`-sid7address ADDRESS`

Specifies the start address for the seventh SID chip (`Sid7AddressStart`) (x64, x64sc, xscpu64, x128 and vsid only). (x128: 0xD420, 0xD440, 0xD460, 0xD480, 0xD4A0, 0xD4C0, 0xD4E0, 0xD700, 0xD720, 0xD740, 0xD760, 0xD780, 0xD7A0, 0xD7C0, 0xD7E0, 0xDE00, 0xDE20, 0xDE40, 0xDE60, 0xDE80, 0xDEA0, 0xDEC0, 0xDEE0, 0xDF00, 0xDF20, 0xDF40, 0xDF60, 0xDF80, 0xDFA0, 0xDFC0, 0xDFE0) (x64, x64sc, xscpu64, vsid: 0xD420, 0xD440, 0xD460, 0xD480, 0xD4A0, 0xD4C0, 0xD4E0, 0xD500, 0xD520, 0xD540, 0xD560, 0xD580, 0xD5A0, 0xD5C0, 0xD5E0, 0xD600, 0xD620, 0xD640, 0xD660, 0xD680, 0xD6A0, 0xD6C0, 0xD6E0, 0xD700, 0xD720, 0xD740, 0xD760, 0xD780, 0xD7A0, 0xD7C0, 0xD7E0, 0xDE00, 0xDE20, 0xDE40, 0xDE60, 0xDE80, 0xDEA0, 0xDEC0, 0xDEE0, 0xDF00, 0xDF20, 0xDF40, 0xDF60, 0xDF80, 0xDFA0, 0xDFC0, 0xDFE0)

`-sid8address ADDRESS`

Specifies the start address for the eight SID chip (`Sid8AddressStart`) (x64, x64sc, xscpu64, x128 and vsid only). (x128: 0xD420, 0xD440, 0xD460, 0xD480, 0xD4A0, 0xD4C0, 0xD4E0, 0xD700, 0xD720, 0xD740, 0xD760, 0xD780, 0xD7A0, 0xD7C0, 0xD7E0, 0xDE00, 0xDE20, 0xDE40, 0xDE60, 0xDE80, 0xDEA0, 0xDEC0, 0xDEE0, 0xDF00, 0xDF20, 0xDF40, 0xDF60, 0xDF80, 0xDFA0, 0xDFC0, 0xDFE0) (x64, x64sc, xscpu64, vsid: 0xD420, 0xD440, 0xD460, 0xD480, 0xD4A0, 0xD4C0, 0xD4E0, 0xD500, 0xD520, 0xD540, 0xD560, 0xD580, 0xD5A0, 0xD5C0, 0xD5E0, 0xD600, 0xD620, 0xD640, 0xD660, 0xD680, 0xD6A0, 0xD6C0, 0xD6E0, 0xD700, 0xD720, 0xD740, 0xD760, 0xD780, 0xD7A0, 0xD7C0, 0xD7E0, 0xDE00, 0xDE20, 0xDE40, 0xDE60, 0xDE80, 0xDEA0, 0xDEC0, 0xDEE0, 0xDF00, 0xDF20, 0xDF40, 0xDF60, 0xDF80, 0xDFA0, 0xDFC0, 0xDFE0)

`-sidenginemodel <engine and model>`

Specify engine and model for the emulated SID chip (`SidEngine`, `SidModel`). (FastSID 6581: 0/fast/fastold/fast6581, FastSID 8580: 1/fastnew/fast8580, ReSID 6581: 256/resid/residold/resid6581, ReSID 8580: 257/residnew/resid8580, ReSID 8580 + digiboost: 258/residdigital/residd/residnewd/resid8580d, DTVSID: 260/dtv/c64dtv/dtvsid, Catweasel MKIII: 512/catweaselmkiii/catweasel3/catweasel/cwmkiii/cw3/cw, HardSID: 768/hardsid/hard/hs, ParSID Port 1: 1024/parsid/parsid1/par1/lpt1, ParSID Port 2: 1280/parsid2/par2/lpt2 ParSID Port 3: 1536/parsid3/par3/lpt3)

`-sidfilters`
`+sidfilters`

Enable/disable emulation of the built-in SID filters (`SidFilters=1`, `SidFilters=0`).

`-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`) for 6581.

`-residgain PERCENTAGE`
: Specifies reSID gain in percent (90 - 100) for 6581.

`-residfilterbias <number>`
: reSID filter bias setting for 6581, which can be used to adjust DAC bias in millivolts.

`-residpass PERCENTAGE`
: Specifies the resampling filter passband in percentage of the total bandwidth (`SidResid8580Passband=0-90`) for 8580.

`-residgain PERCENTAGE`
: Specifies reSID gain in percent (90 - 100) for 8580.

`-residfilterbias <number>`
: reSID filter bias setting for 8580, which can be used to adjust DAC bias in millivolts.

7.1.6 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
- DS12C887 RTC
- 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

7.1.6.1 C64 I/O extension resources

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

`DIGIMAXbase`
: Integer specifying the DigiMAX base address. (0xDD00: useport, 0xDE00, 0xDE20, 0xDE40, 0xDE60, 0xDE80, 0xDEA0, 0xDEC0, 0xDEE0, 0xDF00, 0xDF20, 0xDF40, 0xDF60, 0xDF80, 0xDFA0, 0xDFC0, 0xDFE0)

`DS12C887RTC`
: Boolean specifying whether the DS12C887 RTC cartridge should be emulated or not.

`DS12C887RTCbase`
: Integer specifying the DS12C887 RTC base address. x128: (0xD700, 0xDE00, 0xDF00) x64, x64sc, xscpu64: (0xD500, 0xD600, 0xD700, 0xDE00, 0xDF00)

`DS12C887RTCRunMode`
: Boolean specifying whether the DS12C887 RTC cartridge starts out running or halted. (0: halted, 1: running)

`DS12C887RTCSave`
: Boolean specifying whether the DS12C887 RTC data should be saved when changed or not.

`ETHERNETCART_ACTIVE`
: Boolean that specifies whether the CS8900 ethernet interface emulation is active.

`ETHERNET_INTERFACE`
: String specifying the device name of the ethernet device to use for the emulation.

`ETHERNET_DISABLED`
: Boolean that specified whether ethernet emulation has been disabled because it is not available in the current configuration.

`ETHERNETCARTMode`
: Boolean that specifies whether RR-Net compatible mapping is enabled.

`ETHERNETCARTBase`
: Integer specifying the I/O base address of the emulated ethernet cartridge,

`GEORAM`
: Boolean specifying whether the GEO-RAM cartridge should be emulated or not. (x64, x64sc, x128).

`GEORAMfilename`
: String specifying the filename of the GEORAM image. (x64, x64sc, x128).

`GEORAMImageWrite`
: Boolean, if true write back the GEO-RAM image file automatically, incase the RAM contents changed, when detaching or quitting the emulator. (x64, x64sc, x128).

`GEORAMsize`
: Integer specifying the size of the emulated GEO-RAM in KiB. (64, 128, 256, 512, 1024, 2048, 4096). (x64, x64sc, x128).

`MIDIEnable`
: Boolean specifying whether the MIDI cartridge should be emulated or not (x64, x64sc, xscpu64, x128 only, and only if MIDI support is enabled and available at compile time).

`MIDIMode` Integer specifying the type of emulated MIDI interface (x64, x64sc, xscpu64 and x128 only, and only if MIDI support is enabled and available at compile time). (0: Sequential, 1: Passport/Syntech, 2: DATEL/Siel/JMS, 3: Namesoft, 4: Maplin)

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

`REUfilename`
String specifying the filename of the REU image.

`REUImageWrite`
Boolean, if true write back the REU image file automatically, in case the RAM contents changed, when detaching or quitting the emulator.

`REUsize` Integer specifying the size of the emulated REU in KiB. (128, 256, 512, 1024, 2048, 4096, 8192, 16384)

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

`SFXSoundExpanderChip`
Integer specifying which YM chip is emulated. (3526, 3812)

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

7.1.6.2 C64 I/O extension command-line options

`-digimax`
`+digimax` Enable/disable the DigiMAX cartridge (`DIGIMAX=1`, `DIGIMAX=0`).

`-digimaxbase <base address>`
Base address of the DigiMAX cartridge (`DIGIMAXbase`). (0xDD00: userport, 0xDE00, 0xDE20, 0xDE40, 0xDE60, 0xDE80, 0xDEA0, 0xDEC0, 0xDEE0, 0xDF00, 0xDF20, 0xDF40, 0xDF60, 0xDF80, 0xDFA0, 0xDFC0, 0xDFE0)

`-ds12c887rtc`
`+ds12c887rtc`
Enable/disable the DS12C887 RTC cartridge (`DS12C887RTC=1`, `DS12C887RTC=0`).

`-ds12c887rtcbase <base address>`
Base address of the DS12C887 RTC cartridge (`DS12C887RTCbase`). x128: (0xD700, 0xDE00, 0xDF00) x64, x64sc, xscpu64: (0xD500, 0xD600, 0xD700, 0xDE00, 0xDF00)

`-ds12c887rtchalted`
Set the DS12C887 RTC oscillator to 'halted' (`DS12C887RTCRunMode=0`).

`-ds12c887rtcrunning`
Set the DS12C887 RTC oscillator to 'running' (`DS12C887RTCRunMode=1`).

`-ds12c887rtcsave`
`+ds12c887rtcsave`
Enable/disable saving of the DS12C887 RTC data when changed (`DS12C887RTCSave=1`, `DS12C887RTCSave=0`).

`-miditype <0-4>`
Set MIDI interface type (`MIDIMode`) (x64, x64sc, xscpu64 and x128 only, and only if MIDI support is enabled and available at compile time). (0: Sequential, 1: Passport/Syntech, 2: DATEL/Siel/JMS, 3: Namesoft, 4: Maplin)

`-midi`
`+midi` Enable/disable MIDI emulation (`MIDIEnable=1`, `MIDIEnable=0`) (x64, x64sc, xscpu64 and x128 only, and only if MIDI support is enabled and available at compile time). (0: Sequential, 1: Passport/Syntech, 2: DATEL/Siel/JMS, 3: Namesoft, 4: Maplin)

`-georam`
`+georam` Enable/disable the GEORAM expansion unit (`GEORAM=1`, `GEORAM=0`).

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

`-georamimage <name>`
Specify name of GEORAM image (`GEORAMfilename`).

`-georamimagerw`
`+georamimagerw`
Allow/disallow writing to GEORAM image (`GEORAMImageWrite=1`, `GEORAMImageWrite=0`).

`-georamsize <size in KiB>`
Size of the GEORAM expansion unit (`GEORAMsize`). (64, 128, 256, 512, 1024, 2048, 4096)

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

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

`-reuimage <name>`
Specify name of REU image (`REUfilename`).

`-reuimagerw`
`+reuimagerw`
Allow/disallow writing to REU image (`REUImageWrite=1`, `REUImageWrite=0`).

`-reusize <size in KiB>`
Size of the RAM expansion unit (`REUsize`). (128, 256, 512, 1024, 2048, 4096, 8192, 16384)

`-sfxse`
`+sfxse` Enable/disable the SFX soundexpander cartridge (`SFXSoundExpander=1`, `SFXSoundExpander=0`).

`-sfxsetype <type>`
Set YM chip type (`SFXSoundExpanderChip`). (3526, 3812)

`-sfxss`
`+sfxss` Enable/disable the SFX Sound Sampler cartridge (`SFXSoundSampler=1`, `SFXSoundSampler=0`).

`-cs8900ioif <name>`
Set the system ethernet interface for Ethernet Cartridge emulation

`-ethernetcart`
`+ethernetcart`
Disable/Enable the Ethernet Cartridge (TFE/RR-Net/64NIC/FB-NET)

`-ethernetcartmode <Mode>`
Mode of Ethernet Cartridge (0: TFE, 1: RR-Net)

`-ethernetcartbase <Base address>`
Base address of the Ethernet Cartridge. (0xDE00, 0xDE10, 0xDE20, 0xDE30, 0xDE40, 0xDE50, 0xDE60, 0xDE70, 0xDE80, 0xDE90, 0xDEA0, 0xDEB0, 0xDEC0, 0xDED0, 0xDEE0, 0xDEF0, 0xDF00, 0xDF10, 0xDF20, 0xDF30, 0xDF40, 0xDF50, 0xDF60, 0xDF70, 0xDF80, 0xDF90, 0xDFA0, 0xDFB0, 0xDFC0, 0xDFD0, 0xDFE0, 0xDFF0)

`-tfe` Enable the Ethernet Cartridge in TFE ("The Final Ethernet") compatible mode and set default I/O address

`-rrnet` Enable the Ethernet Cartridge in RR-Net compatible mode and set default I/O address

`-burstmod <value>`
Set the kind of burst modification. This emulates the fast serial bus connection as described at `https://a1bert.kapsi.fi/Dev/burst/`, with the wire to the tape port cut (`BurstMod`). (0: None, 1: CIA-1, 2: CIA-2)

7.1.7 C64 system ROM settings

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

7.1.7.1 C64 system ROM resources

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

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

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

`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.

7.1.7.2 C64 system ROM command-line options

`-basic <name>`
Specify filename of the Basic ROM file (`BasicName`).

`-chargen <name>`
Specify filename of the character generator ROM file (`ChargenName`).

`-kernal <name>`
Specify filename of the Kernal ROM file (`KernalName`).

`-kernalrev <revision>`
Specify Kernal revision (`KernalRev`). (1/2/3, 67/sx, 100/4064)

7.1.8 C64 settings

7.1.8.1 C64 resources

`GlueLogic`
Integer specifying the type of emulated glue-logic. (0: discrete, 1: custom IC)

`BurstMod` Integer specifying the kind of Burst-Mode modification. (0: None, 1: CIA-1, 2: CIA-2)

`BoardType`
Integer specifying the type of emulated board (not available in xscpu64). (0: C64, 1: MAX)

`IECReset` Integer specifying if the IEC bus resets when the CPU resets. (0: No, 1: Yes)

`MemoryHack`
Integer specifying what memory expansion hack is active. (0: None, 1: C64 256K, 2: PLUS60K, 3: PLUS256K)

`PLUS60Kfilename`
String specifying the filename of the PLUS60K RAM image.

`PLUS60Kbase`
Integer that specifies the base address of the PLUS60K RAM expansion. (0xD040, 0xD100)

`PLUS256Kfilename`
: String specifying the filename of the PLUS256K RAM image.

`C64_256Kfilename`
: String specifying the filename of the 256KiB RAM image.

`C64_256Kbase`
: Integer that specifies the base address of the 256KiB RAM expansion. (0xDE00/0xDE80/0xDF00/0xDF80)

`MachineVideoStandard`
: Integer that specifies the video standard of the emulated machine (0: PAL, 1: Old PAL, 2: NTSC, 3: Old NTSC, 4: PAL-N).

`CPMCart`
: Boolean that specifies if a CP/M cartridge is attached.

7.1.8.2 C64 command-line options

`-gluelogictype <type>`
: Set glue logic type (`GlueLogic`). (0: discrete, 1: 252535-01)

`-iecreset <value>`
: Set IEC reset bahaviour (`IECReset`). (0: Do not reset with CPU reset, 1: Reset with CPU reset)

`-memoryexphack <device>`
: Set active memory expansion hack (`MemoryHack`). (0: None, 1: C64 256K, 2: PLUS60K, 3: PLUS256K)

`-plus60kimage <name>`
: Specify name of PLUS60K image (`PLUS60Kfilename`).

`-plus60kbase <base address>`
: Base address of the PLUS60K expansion (`PLUS60Kbase`). (0xD040/0xD100)

`-plus256kimage <name>`
: Specify name of PLUS256K image (`PLUS256Kfilename`).

`-256kimage <name>`
: Specify name of 256K image (`C64_256Kfilename`).

`-256kbase <base address>`
: Base address of the 256KiB expansion (`C64_256Kbase`). (0xDE00, 0xDE80, 0xDF00, 0xDF80)

`-pal`
: Use PAL sync factor (`MachineVideoStandard=1`).

`-ntsc`
: Use NTSC sync factor (`MachineVideoStandard=2`).

`-ntscold`
: Use old NTSC sync factor (`MachineVideoStandard=3`).

`-paln`
: Use PAL-N sync factor (`MachineVideoStandard=4`).

`-model <Model>`
: Set the C64 model (`VICIIModel`, `CIA1Model`, `CIA2Model`, `GlueLogic`, `BoardType`, `IECReset`, `KernalName`, `ChargenName`, `SidEngine`, `SidModel`) (x64 and x64sc only). (c64/c64c/c64old, ntsc/newntsc/oldntsc, drean, jap, c64gs, pet64, ultimax)

`-cpmcart`
`+cpmcart` Enable/disable the CP/M cartridge (`CPMCart=1`, `CPMCart=0`)

7.2 C128-specific commands and settings

7.2.1 VDC settings

7.2.1.1 VDC resources

`VDC64KB` Boolean to enabled/disable full 64KiB video ram.

`VDCRevision`
Integer specifying the VDC hardware revision (0: Rev 0, 1: Rev 1, 2: Rev 2).

`VDCVideoCache`
Boolean specifying whether the video cache is turned on.

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

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

`VDCStretchVertical`
Boolean specifying whether vertical stretching is turned on.

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

`VDCExternalPalette`
Boolean specifying whether to use external palette file or not.

`VDCColorSaturation`
Integer specifying saturation of internal calculated palette. (0..2000)

`VDCColorContrast`
Integer specifying contrast of internal calculated palette. (0..2000)

`VDCColorBrightness`
Integer specifying brightness of internal calculated palette. (0..2000)

`VDCColorGamma`
Integer specifying gamma of internal calculated palette. (0..4000)

`VDCColorTint`
Integer specifying tint of internal calculated palette. (0..2000)

`VDCPALScanLineShade`
Integer specifying amount of scan line shading for the CRT emulation. (0..1000)

`VDCPALBlur`
Integer specifying amount of horizontal blur for the CRT emulation. (0..1000)

`VDCPALOddLinePhase`
Integer specifying phase for color carrier in odd lines. (0..2000)

`VDCPALOddLineOffset`
: Integer specifying phase offset for color carrier in odd lines. (0..2000)

`VDCAudioLeak`
: Boolean specifying whether to enable/disable video to audio leak emulation.

`VDCFilter`
: Integer specifying rendering filter (0: None, 1: CRT emulation, 2: Scale2x)

7.2.1.2 VDC command-line options

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

`-VDCdsize`
`+VDCdsize`
: Enable/disable double size (`VDCDoubleSize=1`, `VDCDoubleSize=0`).

`-VDCstretchvertical`
`+VDCstretchvertical`
: Enable/Disable vertical stretching (`VDCStretchVertical=1`, `VDCStretchVertical=0`).

`-VDCdscan`
`+VDCdscan`
: Enable/disable double scan (`VDCDoubleScan=1`, `VDCDoubleScan=0`).

`-VDCintpal`
: Use an internal calculated palette (`VDCExternalPalette=0`).

`-VDCextpal`
: Use an external palette (file) (`VDCExternalPalette=1`).

`-VDCpalette <name>`
: Specify name of file of external palette (`VDCPaletteFile`).

`-VDC16KB` Set the VDC memory size to 16KiB (`VDC64KB=0`).

`-VDC64KB` Set the VDC memory size to 64KiB (`VDC64KB=1`).

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

`-VDCsaturation <0-2000>`
: Set saturation of internal calculated palette (`VDCColorSaturation`).

`-VDCcontrast <0-2000>`
: Set contrast of internal calculated palette (`VDCColorContrast`).

`-VDCbrightness <0-2000>`
: Set brightness of internal calculated palette (`VDCColorBrightness`).

`-VDCgamma <0-4000>`
: Set gamma of internal calculated palette (`VDCColorGamma`).

`-VDCtint <0-2000>`
: Set tint of internal calculated palette (`VDCColorTint`).

`-VDCoddlinesphase <0-2000>`
: Set phase for color carrier in odd lines (`VDCPALOddLinePhase`).

`-VDCoddlinesoffset <0-2000>`
: Set phase offset for color carrier in odd lines (`VDCPALOddLineOffset`).

`-VDCcrtblur <0-1000>`
: Amount of horizontal blur for the CRT emulation (`VDCPALBlur`).

`-VDCcrtscanlineshade <0-1000>`
: Amount of scan line shading for the CRT emulation (`VDCPALScanLineShade`).

`-VDCaudioleak`
`+VDCaudioleak`
: Enable/disable video to audio leak emulation (`VDCAudioLeak=1`, `VDCAudioLeak=0`).

`-VDCfilter <Mode>`
: Select rendering filter (`VDCFilter`). (0: None, 1: CRT emulation, 2: Scale2x)

7.2.2 C128 system ROM settings

7.2.2.1 C128 system ROM resources

`ChargenIntName`
: String specifying the filename of the international character generator ROM image.

`ChargenDEName`
: String specifying the filename of the German character generator ROM image.

`ChargenFRName`
: String specifying the filename of the French character generator ROM image.

`ChargenSEName`
: String specifying the filename of the Swedish character generator ROM image.

`ChargenCHName`
: String specifying the filename of the Swiss character generator ROM image.

`ChargenNOName`
: String specifying the filename of the Norwegian character generator ROM image.

`KernalIntName`
: String specifying the filename of the international Kernal ROM image.

`KernalDEName`
: String specifying the filename of the German Kernal ROM image.

`KernalFIName`
: String specifying the filename of the Finnish Kernal ROM image.

`KernalFRName`
: String specifying the filename of the French Kernal ROM image.

`KernalITName`
: String specifying the filename of the Italian Kernal ROM image.

`KernalNOName`
: String specifying the filename of the Norwegian Kernal ROM image.

`KernalSEName`
: String specifying the filename of the Swedish Kernal ROM image.

`KernalCHName`
: String specifying the filename of the Swiss Kernal ROM image.

`BasicLoName`
`BasicHiName`
: Strings specifying the filename of the Basic ROM images.

`Kernal64Name`
: String specifying the filename of the C64 Kernal ROM image.

`Basic64Name`
: String specifying the filename of the C64 Basic ROM image.

`InternalFunctionROM`
: Integer specifying the internal function ROM type (0: None, 1: ROM, 2: RAM, 3: RTC).

`InternalFunctionName`
: String specifying the filename of the ROM image for the internal function ROM.

`InternalFunctionROMRTCSave`
: Boolean to enable/disable the saving of the Internal Function RTC data when changed.

`ExternalFunctionROM`
: Integer specifying the external function ROM type (0: None, 1: ROM, 2: RAM, 3: RTC)

`ExternalFunctionName`
: String specifying the filename of the ROM image for the external function ROM.

`ExternalFunctionROMRTCSave`
: Boolean to enable/disable the saving of the External Function RTC data when changed.

7.2.2.2 C128 system ROM command-line options

`-basic64 <name>`
: Specify name of C64 mode BASIC ROM image (`Basic64Name`).

`-kernal64 <name>`
: Specify name of C64 mode Kernal ROM image (`Kernal64Name`).

`-basiclo <name>`
: Specify name of BASIC ROM image (lower part) (`BasicLoName`).

`-basichi <name>`
: Specify name of BASIC ROM image (higher part) (`BasicHiName`).

`-kernal <name>`
: Specify name of international Kernal ROM image (`KernalIntName`).

`-kernalde <name>`
: Specify name of German Kernal ROM image (`KernalDEName`).

`-kernalfi <name>`
: Specify name of Finnish Kernal ROM image (`KernalFIName`).

`-kernalfr <name>`
: Specify name of French Kernal ROM image (`KernalFRName`).

`-kernalit <name>`
: Specify name of Italian Kernal ROM image (`KernalITName`).

`-kernalno <name>`
: Specify name of Norwegian Kernal ROM image (`KernalNOName`).

`-kernalse <name>`
: Specify name of Swedish Kernal ROM image (`KernalSEName`).

`-kernalch <name>`
: Specify name of Swiss Kernal ROM image (`KernalCHName`).

`-chargen <name>`
: Specify name of international character generator ROM image (`ChargenIntName`).

`-chargde <name>`
: Specify name of German character generator ROM image (`ChargenDEName`).

`-chargfr <name>`
: Specify name of French character generator ROM image (`ChargenFRName`).

`-chargse <name>`
: Specify name of Swedish character generator ROM image (`ChargenSEName`).

`-chargch <name>`
: Specify name of Swiss character generator ROM image (`ChargenCHName`).

`-chargno <name>`
: Specify name of Norwegian character generator ROM image (`ChargenNOName`).

`-intfunc <type>`
: Set the internal Function ROM type (`InternalFunctionROM`). (0: None, 1: ROM, 2: RAM, 3: RTC)

`-intfrom <name>`
: Specify name of internal Function ROM image (`InternalFunctionName`).

`-intfuncrtcsave`
`+intfuncrtcsave`
: Enable/disable the saving of the Internal Function RTC data when changed (`InternalFunctionROMRTCSave=1`, `InternalFunctionROMRTCSave=0`).

`-extfunc <type>`
: Set the external Function ROM type (`ExternalFunctionROM`). (0: None, 1: ROM, 2: RAM, 3: RTC)

`-extfrom <name>`
: Specify name of external Function ROM image (`ExternalFunctionName`).

`-extfuncrtcsave`
`+extfuncrtcsave`
: Enable/disable the saving of the External Function RTC data when changed (`ExternalFunctionROMRTCSave=1`, `ExternalFunctionROMRTCSave=0`).

7.2.3 C128 settings

7.2.3.1 C128 resources

`C128ColumnKey`
: Boolean specifying the status of the 40/80 columns key. (0: Not pressed, 1: Pressed)

`Go64Mode` Boolean, if true enter C64 mode on reset.

`C128FullBanks`
: Boolean to enable/disable RAM banks 2 and 3.

`MachineType`
: Integer specifying the C128 machine type. (0: International, 1: Finnish, 2: French, 3: German, 4: Italian, 5: Norwegian, 6: Swedish)

`MachineVideoStandard`
: Integer that specifies the video standard of the emulated machine (1: PAL, 2: NTSC).

`C128HideVDC`
: Boolean to enable/disable the VDC display window.

7.2.3.2 C128 command-line options

`-40col` Activate 40 column mode (`C128ColumnKey=1`).

`-80col` Activate 80 column mode (`C128ColumnKey=0`).

`-go64` Always switch to C64 mode on reset (`Go64Mode=1`.

`+go64` Always switch to C128 mode on reset (`Go64Mode=0`.

`-pal` Use PAL sync factor (`MachineVideoStandard=1`).

`-ntsc` Use NTSC sync factor (`MachineVideoStandard=2`).

`-model <Model>`
: Set the C128 model (`MachineVideoStandard`, `CIA1Model`, `CIA2Model`, `VICIINewLuminances`, `VDCRevision`, `VDC64KB`, `SidEngine`, `SidModel`). (c128/c128dcr, pal/ntsc)

`-c128fullbanks`
`+c128fullbanks`
: Enable/disable RAM banks 2 and 3 (`C128FullBanks=1`, `C128FullBanks=0`).

`-machinetype <Type>`
: Set the C128 machine type (`MachineType`). (0: International, 1: Finnish, 2: French, 3: German, 4: Italian, 5: Norwegian, 6: Swedish)

`-hidevdcwindow`
`+hidevdcwindow`
: Hide/show the VDC display window (`C128HideVDC=1`, `C128HideVDC=0`).

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 2MiB 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 2MiB (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 filename of the C64DTV ROM image (`c64dtvromfilename`).

`-c64dtvromrw`
`+c64dtvromrw`
: Enable/disable writing to C64DTV ROM image (`c64dtvromrw=1`, `c64dtvromrw=0`).

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/disable true hardware flash file system (`FlashTrueFS=1`, `FlashTrueFS=0`).

`-fsflash <name>`
: Specify the directory for the flash file system device (`FSFlashDir`).

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 (`DtvRevision`). (2: DTV2, 3: DTV3)

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 (`HummerADC=1`, `HummerADC=0`).

`-ps2mouse`
`+ps2mouse`
: Enable/disable PS/2 mouse on userport (`ps2mouse=1`, `ps2mouse=0`).

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 2MiB of RAM and 2MiB of Flash, but only 64 KiB at a time. The 64KiB bank can be selected with "bank ram00".."ram1f" for RAM and "bank rom00".."rom1f" for Flash.

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

7.3.7 DTV resources

`DtvRevision`
: Integer specifying the emulated DTV revision. (2: DTV2, 3: DTV3)

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

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

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

`c64dtvromfilename`
: String specifying the filename of the DTV Flash ROM image.

`c64dtvromrw`
: Boolean that specifies whether the emulated Flash ROM is writeable.

`FSFlashDir`
: String specifying the working directory for the flash file system.

`FlashTrueFS`
: Boolean, enables true hardware flash file system.

`HummerADC`
: Boolean to enable/disable the Hummer ADC emulation.

`ps2mouse` Boolean to enable/disable PS/2 Mouse emulation.

`DtvBlitterLog`
: Boolean, enables Blitter logging.

`DtvDMALog`
: Boolean, enables DMA logging.

`DtvFlashLog`
: Boolean, enables Flash ROM logging.

`MachineVideoStandard`
: Integer that specifies the video standard of the emulated machine. (1: PAL, 2: NTSC)

7.3.8 DTV command-line options

`-model <Model>`
: Set the DTV model (`MachineVideoStandard`, `DtvRevision`, `HummerADC`). (v2/v2pal/v2ntsc, v3/v3pal/v3ntsc, hummer)

`-pal` Use PAL sync factor (`MachineVideoStandard=1`).

`-ntsc` Use NTSC sync factor (`MachineVideoStandard=2`).

`-chargen <name>`
: Specify filename of the character generator ROM file (`ChargenName`).

`-kernal <name>`
: Specify filename of the Kernal ROM file (`KernalName`).

`-basic <name>`
: Specify the filename of the Basic ROM file (`BasicName`).

7.4 SCPU64-specific commands and settings

7.4.1 SCPU64 resources

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

`SCPU64Name`
: String specifying the name of the SCPU64 ROM (default `scpu64`).

`MachineVideoStandard`
: Integer that specifies the video standard of the emulated machine (4: PAL-N, 3: Old NTSC, 1: PAL, 2: NTSC).

`IECReset` Integer specifying if the IEC bus resets when the CPU resets. (0: No, 1: Yes)

`BurstMod` Integer specifying the kind of Burst-Mode modification. (0: None, 1: CIA-1, 2: CIA-2)

`SIMMSize` Integer specifying the size of the SIMM RAM. (0, 1, 4, 8, 16)

`JiffySwitch`
: Boolean to enable/disable the jiffy switch.

`SpeedSwitch`
: Boolean to enable/disable the speed switch.

`GlueLogic`
: Integer specifying the type of emulated glue-logic. (0: discrete, 1: custom IC)

7.4.2 SCPU64 command-line options

`-chargen <name>`
: Specify filename of the character generator ROM file (`ChargenName`).

`-scpu64 <Name>`
: Specify filename of the SCPU64 ROM file (`SCPU64Name`).

`-model <Model>`
: Set the C64 model (`MachineVideoStandard`, `CIA1Model`, `CIA2Model`, `VICIINewLuminances`, `IECReset`, `ChargenName`, `SidEngine`, `SidModel`). (c64/c64c/c64old, ntsc/newntsc/oldntsc, drean, jap, c64gs)

`-pal`
: Use PAL sync factor (`MachineVideoStandard=1`).

`-ntsc`
: Use NTSC sync factor (`MachineVideoStandard=2`).

`-ntscold`
: Use old NTSC sync factor (`MachineVideoStandard=3`).

`-paln`
: Use PAL-N sync factor (`MachineVideoStandard=4`).

`-iecreset <value>`
: Set IEC reset bahaviour (`IECReset`). (0: Do not reset with CPU reset, 1: Reset with CPU reset)

`-burstmod <value>`
: Set the kind of burst modification. This emulates the fast serial bus connection as described at `https://a1bert.kapsi.fi/Dev/burst/`, with the wire to the tape port cut (`BurstMod`). (0: None, 1: CIA-1, 2: CIA-2)

`-simmsize <number>`
: Set the size of the SIMM RAM (`SIMMSize`). (0, 1, 4, 8, 16)

`-jiffyswitch`
`+jiffyswitch`
: Enable/disable the jiffy switch (`JiffySwitch=1`, `JiffySwitch=0`).

`-speedswitch`
`+speedswitch`
: Enable/disable the speed switch (`SpeedSwitch=1`, `SpeedSwitch=0`).

`-gluelogictype <type>`
: Set glue logic type (`GlueLogic`). (0: discrete, 1: 252535-01)

7.5 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.

7.5.1 Using cartridge images

As with the C64 (see Section 7.1.1 [C64 cartridges], page 78), it is possible to attach several types of cartridge images:

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

This can all be done via the “Attach cartridge image. . .” command in the 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 16KiB image. Vice can attach such concatenated files at the start address $2000, $4000, and $6000. The second half of such an image is moved to the memory block following the first block. It is also possible to concatenate four 8KiB files, the resulting file will load into all 4 cartridge blocks.

If you encounter 16KiB images that have the second half at eg $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. Alternatively you can create a 32KiB file with padding blocks in between.

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.

7.5.2 VIC20 cartridge settings

7.5.2.1 VIC20 cartridge resources

`CartridgeReset`
: Boolean specifying whether the machine should be reset when a cartridge is changed.

`CartridgeType`
: Integer specifying the type of cartridge emulated.

  The following cartridge types are valid:

  - - 1: None
  - 1: Generic
  - 2: Megacart
  - 3: Final Expansion
  - 4: Vic Flash Plugin
  - 5: IEEE-4888
  - 6: SIDCart
  - 7: Ultimem
  - 8: IO ram cart (IO2)
  - 9: IO ram cart (IO3)
  - 10: BehrBonz
  - 0x8000: Auto Detect
  - 0x8002: 4KiB at $2000-$2FFF
  - 0x8003: 8KiB at $2000-$3FFF
  - 0x8004: 4KiB at $6000-$6FFF
  - 0x8005: 8KiB at $6000-$7FFF
  - 0x8006: 4KiB at $A000-$AFFF

- 0x8007: 8KiB at $A000-$BFFF
- 0x8008: 4KiB at $B000-$BFFF
- 0x8009: 8KiB at $4000-$5FFF
- 0x800A: 4KiB at $4000-$4FFF
- 0x8013: 16KiB at $2000-$5FFF
- 0x8015: 16KiB at $6000-$9FFF
- 0x8019: 16KiB at $4000-$7FFF

`CartridgeFile`
: String specifying the filename of the image for the current cartridge.

`IOCollisionHandling`
: Integer specifying the way the I/O collisions should be handled. (0: error message and detach all involved carts, 1: error message and detach last attached involved carts, 2: warning in log and 'AND' the valid return values)

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

`FinalExpansionWriteBack`
: Boolean, if true write back the Flash ROM image file automatically, incase the contents changed, when detaching or quitting the emulator.

`VicFlashPluginWriteBack`
: Boolean, if true write back the Flash ROM image file automatically, incase the contents changed, when detaching or quitting the emulator.

`MegaCartNvRAMfilename`
: String specifying the filename of the MegaCart NvRAM image.

`MegaCartNvRAMWriteBack`
: Boolean, if true write back the NvRAM image file automatically, incase the RAM contents changed, when detaching or quitting the emulator.

`UltiMemWriteBack`
: Boolean, if true write back the Flash ROM image file automatically, incase the contents changed, when detaching or quitting the emulator.

`IO2RAM` Boolean specifying whether the I/O-2 ($9800-$9BFF) RAM cartridge should be emulated or not.

`IO3RAM` Boolean specifying whether the I/O-3 ($9C00-$9FFF) RAM cartridge should be emulated or not.

`VFLImod` Boolean specifying whether the VFLI modification should be enabled.

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

`SFXSoundExpanderChip`
: Integer specifying which YM chip is emulated. (3526, 3812)

`SFXSoundExpanderIOSwap`
: Boolean, swap io mapping (map cart I/O to VIC20 I/O-2) or don't swap io mapping (map cart I/O to VIC20 I/O-3).

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

`SFXSoundSamplerIOSwap`
: Boolean, swap io mapping (map cart I/O to VIC20 I/O-2) or don't swap io mapping (map cart I/O to VIC20 I/O-3).

`GEORAMfilename`
: String specifying the filename of the GEORAM image.

`GEORAM`
: Boolean specifying whether the GEO-RAM cartridge (using the MasC=uerade cartridge adapter) should be emulated or not.

`GEORAMsize`
: Integer specifying the size of the emulated GEO-RAM in KiB. (64, 128, 256, 512, 1024, 2048, 4096).

`GEORAMImageWrite`
: Boolean, if true write back the GEO-RAM image file automatically, incase the RAM contents changed, when detaching or quitting the emulator.

`GEORAMIOSwap`
: Boolean specifying whether the io mapping should be swapped (map cart I/O-1 to VIC20 I/O-3 and cart I/O-2 to VIC20 I/O-2) or not (map cart I/O-2 to VIC20 I/O-2 and cart I/O-2 to VIC20 I/O-3).

`SidCart`
: Boolean specifying whether SID-Cart emulation is enabled or not.

`SidAddress`
: Integer that specifies the base address of the emulated SID chip. (0x9800, 0x9C00)

`SidClock`
: Integer specifying the clock rate used for the emulated SID chip (0: C64, 1: VIC20)

`DIGIMAX`
: Boolean specifying whether the DigiMAX cartridge (using the MasC=uerade cartridge adapter) should be emulated or not.

`DIGIMAXbase`
: Integer specifying the DigiMAX base address. (0x9800, 0x9820, 0x9840, 0x9860, 0x9880, 0x98A0, 0x98C0, 0x98E0, 0x9C00, 0x9C20, 0x9C40, 0x9C60, 0x9C80, 0x9CA0, 0x9CC0, 0x9CE0)

`DS12C887RTC`
: Boolean specifying whether the DS12C887 RTC cartridge (using the MasC=uerade cartridge adapter) should be emulated or not.

`DS12C887RTCbase`
: Integer specifying the DS12C887 RTC base address. (0x9800, 0x9C00)

`DS12C887RTCRunMode`
: Boolean specifying whether the DS12C887 RTC cartridge starts out running or halted. (0: halted, 1: running)

`DS12C887RTCSave`
: Boolean specifying whether the DS12C887 RTC data should be saved when changed or not.

`IEEE488` Boolean specifying whether the IEEE488 interface should be emulated or not.

`MachineVideoStandard`
: Integer that specifies the video standard of the emulated machine (1: PAL, 2: NTSC).

7.5.2.2 VIC20 cartridge command-line options

`-iocollision <method>`
: Select the way the I/O collisions should be handled (`IOCollisionHandling`). (0: error message and detach all involved carts, 1: error message and detach last attached involved carts, 2: warning in log and 'AND' the valid return values

`-cartreset`
`+cartreset`
: Do/don't reset machine if a cartridge is attached or detached (`CartridgeReset=1`, `CartridgeReset`).

`-cart2 <name>`
: Specify 4/8/16KiB extension ROM name at $2000

`-cart4 <name>`
: Specify 4/8/16KiB extension ROM name at $4000

`-cart6 <name>`
: Specify 4/8/16KiB extension ROM name at $6000

`-cartA <name>`
: Specify 4/8KiB extension ROM name at $A000

`-cartB <name>`
: Specify 4KiB extension ROM name at $B000

`-cartgeneric <name>`
: Specify generic extension ROM name

`-cartbb <name>`
: Specify Behr-Bonz extension ROM name

`-cartmega <name>`
: Specify Mega-Cart extension ROM name

`-mcnvramfile <name>`
: Set Mega-Cart NvRAM filename (`MegaCartNvRAMfilename`).

`-mcnvramwriteback`
`+mcnvramwriteback`
: Enable/Disable Mega-Cart NvRAM writeback (`MegaCartNvRAMWriteBack=1`, `MegaCartNvRAMWriteBack=0`).

`-cartfe <name>`
Specify Final Expansion extension ROM name

`-fewriteback`
`+fewriteback`
Enable/disable Final Expansion write back to ROM file (`FinalExpansionWriteBack=1`, `FinalExpansionWriteBack=0`).

`-cartfp <name>`
Specify Vic Flash Plugin extension ROM name

`-fpwriteback`
`+fpwriteback`
Enable/Disable Vic Flash Plugin write back to ROM file (`VicFlashPluginWriteBack=1`, `VicFlashPluginWriteBack=0`).

`-ultimem <name>`
Specify Ultimem extension ROM name

`-umwriteback`
`+umwriteback`
Enable/disable UltiMem write back to ROM file (`UltiMemWriteBack=1`, `UltiMemWriteBack=0`).

`-io2ram`
`+io2ram` Enable/disable the I/O-2 ($9800-$9BFF) RAM cartridge (`IO2RAM=1`, `IO2RAM=0`).

`-io3ram`
`+io3ram` Enable/disable the I/O-3 ($9C00-$9FFF) RAM cartridge (`IO3RAM=1`, `IO3RAM=0`).

`-ieee488`
`+ieee488` Enable/disable VIC-1112 IEEE488 interface (`IEEE488=1`, `IEEE488=0`).

`-vflimod`
`+vflimod` Enable/disable VIC-20 VFLI modification (`VFLImod=1`, `VFLImod=0`)

`-sidcart`

`+sidcart` Enable/disable SID Cartridge (`SidCart=1`, `SidCart=0`).

`-sidcartaddress <address>`
Specify address of the SID Cartridge (`SidAddress`). (0x9800, 0x9C00)

`-sidcartclock <clock>`
Specify clock of the SID Cartridge (`SidClock`). (0: C64, 1: VIC20)

`-cs8900ioif <name>`
Set the system ethernet interface for Ethernet Cartridge emulation

`-ethernetcart`
`+ethernetcart`
Disable/Enable the Ethernet Cartridge (TFE/RR-Net/64NIC/FB-NET)

`-ethernetcartmode <Mode>`
Mode of Ethernet Cartridge (0: TFE, 1: RR-Net)

`-ethernetcartbase <Base address>`
: Base address of the Ethernet Cartridge. (0x9800, 0x9810, 0x9820, 0x9830, 0x9840, 0x9850, 0x9860, 0x9870, 0x9880, 0x9890, 0x98A0, 0x98B0, 0x98C0, 0x98D0, 0x98E0, 0x98F0, 0x9C00, 0x9C10, 0x9C20, 0x9C30, 0x9C40, 0x9C50, 0x9C60, 0x9C70, 0x9C80, 0x9C90, 0x9CA0, 0x9CB0, 0x9CC0, 0x9CD0, 0x9CE0, 0x9CF0)

`-tfe`
: Enable the Ethernet Cartridge in TFE ("The Final Ethernet") compatible mode and set default I/O address

`-rrnet`
: Enable the Ethernet Cartridge in RR-Net compatible mode and set default I/O address

`-digimax`
`+digimax`
: Enable/disable the DigiMAX cartridge (using the MasC=uerade cartridge adapter) (`DIGIMAX=1`, `DIGIMAX=0`).

`-digimaxbase <base address>`
: Base address of the DigiMAX cartridge (`DIGIMAXbase`). (0x9800, 0x9820, 0x9840, 0x9860, 0x9880, 0x98A0, 0x98C0, 0x98E0, 0x9C00, 0x9C20, 0x9C40, 0x9C60, 0x9C80, 0x9CA0, 0x9CC0, 0x9CE0)

`-ds12c887rtc`
`+ds12c887rtc`
: Enable/disable the DS12C887 RTC cartridge (using the MasC=uerade cartridge adapter) (`DS12C887RTC=1`, `DS12C887RTC=0`).

`-ds12c887rtcbase <base address>`
: Base address of the DS12C887 RTC cartridge (`DS12C887RTCbase`). (0x9800, 0x9C00)

`-ds12c887rtchalted`
: Set the DS12C887 RTC oscillator to 'halted' (`DS12C887RTCRunMode=0`).

`-ds12c887rtcrunning`
: Set the DS12C887 RTC oscillator to 'running' (`DS12C887RTCRunMode=1`).

`-ds12c887rtcsave`
`+ds12c887rtcsave`
: Enable/disable saving of the DS12C887 RTC data when changed (`DS12C887RTCSave=1`, `DS12C887RTCSave=0`).

`-sfxse`
`+sfxse`
: Enable/disable the SFX soundexpander cartridge (using the MasC=uerade cartridge adapter) (`SFXSoundExpander=1`, `SFXSoundExpander=0`).

`-sfxsetype <type>`
: Set YM chip type (`SFXSoundExpanderChip`). (3526, 3812)

`-sfxseioswap`
: Swap io mapping (map cart I/O to VIC20 I/O-2) (`SFXSoundExpanderIOSwap=1`).

`+sfxseioswap`
: Swap io mapping (map cart I/O to VIC20 I/O-3) (`SFXSoundExpanderIOSwap=0`).

`-sfxssioswap`
: Swap io mapping (map cart I/O to VIC20 I/O-2) (`SFXSoundSamplerIOSwap=1`).

`+sfxssioswap`
: Don't swap io mapping (map cart I/O to VIC20 I/O-3) (`SFXSoundSamplerIOSwap=0`).

`-sfxss`
`+sfxss`
: Enable/disable the SFX Sound Sampler cartridge (`SFXSoundSampler=1`, `SFXSoundSampler=0`).

`-georamioswap`
: Swap the io mapping (map cart I/O-1 to VIC20 I/O-3 and cart I/O-2 to VIC20 I/O-2) (`GEORAMIOSwap=1`).

`+georamioswap`
: Do not swap the io mapping (map cart I/O-2 to VIC20 I/O-2 and cart I/O-2 to VIC20 I/O-3) (`GEORAMIOSwap=0`).

`-georam`
`+georam`
: Enable/disable the GEORAM expansion unit (using the MasC=uerade cartridge adapter) (`GEORAM=1`, `GEORAM=0`).

`-georamimage <name>`
: Specify name of GEORAM image (`GEORAMfilename`).

`-georamimagerw`
`+georamimagerw`
: Allow/disallow writing to GEORAM image (`GEORAMImageWrite=1`, `GEORAMImageWrite=0`).

`-georamsize <size in KiB>`
: Size of the GEORAM expansion unit (`GEORAMsize`). (64, 128, 256, 512, 1024, 2048, 4096)

`-model <model>`
: Specify the VIC20 model you want to emulate (`MachineVideoStandard`, `RamBlock0`, `RamBlock1`, `RamBlock2`, `RamBlock3` and `RamBlock5`). (vic20/vic20pal/vic20ntsc, vic21)

`-pal`
: Use PAL sync factor (`MachineVideoStandard=1`).

`-ntsc`
: Use NTSC sync factor (`MachineVideoStandard=2`).

7.5.3 VIC settings

7.5.3.1 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.

`VICExternalPalette`
: Boolean specifying whether to use external palette file or not.

`VICColorSaturation`
: Integer specifying saturation of internal calculated palette. (0..2000)

`VICColorContrast`
: Integer specifying contrast of internal calculated palette, (0..2000)

`VICColorBrightness`
: Integer specifying brightness of internal calculated palette. (0..2000)

`VICColorGamma`
: Integer specifying gamma of internal calculated palette. (0..4000)

`VICColorTint`
: Integer specifying tint of internal calculated palette. (0..2000)

`VICPALScanLineShade`
: Integer specifying amount of scan line shading for the CRT emulation. (0..1000)

`VICPALBlur`
: Integer specifying amount of horizontal blur for the CRT emulation. (0..1000)

`VICPALOddLinePhase`
: Integer specifying phase for color carrier in odd lines. (0..2000)

`VICPALOddLineOffset`
: Integer specifying phase offset for color carrier in odd lines. (0..2000)

`VICAudioLeak`
: Boolean specifying whether to enable/disable video to audio leak emulation.

`VICFilter`
: Integer specifying the rendering filter. (0: None, 1: CRT emulation, 2: Scale2x)

`VICBorderMode`
: Integer specifying border display mode (0: normal, 1: full, 2: debug, 3: none)

7.5.3.2 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`).

`-VICfilter <Mode>`
: Select rendering filter (`VICFilter`). (0: None, 1: CRT emulation, 2: Scale2x)

`-VICpalette NAME`
: Specify the name of the palette file (`VICPaletteFile`).

`-VICintpal`
: Use an internal calculated palette (`VICExternalPalette=0`).

`-VICextpal`
: Use an external palette (file) (`VICExternalPalette=1`).

`-VICborders <mode>`
: Set VIC border display mode (`VICBorderMode`). (0: normal, 1: full, 2: debug, 3: none)

`-VICsaturation <0-2000>`
: Set saturation of internal calculated palette (`VICColorSaturation`).

`-VICcontrast <0-2000>`
: Set contrast of internal calculated palette (`VICColorContrast`).

`-VICbrightness <0-2000>`
: Set brightness of internal calculated palette (`VICColorBrightness`).

`-VICgamma <0-4000>`
: Set gamma of internal calculated palette (`VICColorGamma`).

`-VICtint <0-2000>`
: Set tint of internal calculated palette (`VICColorTint`).

`-VICoddlinesphase <0-2000>`
: Set phase for color carrier in odd lines (`VICPALOddLinePhase`).

`-VICoddlinesoffset <0-2000>`
: Set phase offset for color carrier in odd lines. (`VICPALOddLineOffset`).

`-VICcrtblur <0-1000>`
: Amount of horizontal blur for the CRT emulation (`VICPALBlur`).

`-VICcrtscanlineshade <0-1000>`
: Amount of scan line shading for the CRT emulation (`VICPALScanLineShade`).

`-VICaudioleak`
`+VICaudioleak`
: Enable/disable video to audio leak emulation (`VICAudioLeak=1`, `VICAudioLeak=0`).

7.5.4 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 KiB at $0400-$0FFF);

- block 1 (8 KiB at $2000-$3FFF);
- block 2 (8 KiB at $4000-$5FFF);
- block 3 (8 KiB at $6000-$7FFF);
- block 5 (8 KiB 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 select:

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

7.5.4.1 VIC20 memory configuration resources

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

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

7.5.4.2 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` (3KiB space in block 0);
- `8k` (first 8KiB extension block);
- `16k` (first and second 8KiB 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 16KiB ROM modules.

7.5.5 VIC20 system ROM settings

These settings can be used to control what system ROMs are loaded in the VIC20 emulator at startup.

7.5.5.1 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`).

7.5.5.2 VIC20 system ROM command-line options

`-kernal <name>`
: Specify the filename of the Kernal ROM file (`KernalName`).

`-basic <name>`
: Specify the filename of the Basic ROM file (`BasicName`).

`-chargen <name>`
: Specify the filename of 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`).

7.5.6 VIC20 settings

7.5.6.1 VIC20 command-line options

7.6 PLUS4-specific commands and settings

7.6.1 TED settings

7.6.1.1 TED resources

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

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

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

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

`TEDExternalPalette`
: Boolean specifying whether to use external palette file or not.

`TEDColorSaturation`
: Integer specifying saturation of internal calculated palette. (0..2000)

`TEDColorContrast`
: Integer specifying contrast of internal calculated palette. (0..2000)

`TEDColorBrightness`
: Integer specifying brightness of internal calculated palette. (0..2000)

`TEDColorGamma`
: Integer specifying gamma of internal calculated palette. (0..4000)

`TEDColorTint`
: Integer specifying tint of internal calculated palette. (0..2000)

`TEDPALScanLineShade`
: Integer specifying amount of scan line shading for the CRT emulation. (0..1000)

`TEDPALBlur`
: Integer specifying amount of horizontal blur for the CRT emulation. (0..1000)

`TEDPALOddLinePhase`
: Integer specifying phase for color carrier in odd lines. (0..2000)

`TEDPALOddLineOffset`
: Integer specifying phase offset for color carrier in odd lines. (0..2000)

`TEDAudioLeak`
: Boolean specifying whether to enable/disable video to audio leak emulation.

`TEDFilter`
: Integer specifying rendering filter. (0: None, 1: CRT emulation, 2: Scale2x)

`TEDBorderMode`
: Integer specifying border display mode. (0: normal, 1: full, 2: debug, 3: none)

7.6.1.2 TED command-line options

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

`-TEDdsize`
`+TEDdsize`
: Enable/disable double size (`TEDDoubleSize=1`, `TEDDoubleSize=0`).

`-TEDdscan`
`+TEDdscan`
: Enable/disable double scan (`TEDDoubleScan=1`, `TEDDoubleScan=0`).

`-TEDfilter <Mode>`
: Select rendering filter (`TEDFilter`). (0: None, 1: CRT emulation, 2: Scale2x)

`-TEDintpal`
: Use an internal calculated palette (`TEDExternalPalette=0`).

`-TEDextpal`
: Use an external palette (file) (`TEDExternalPalette=1`).

`-TEDpalette <name>`
: Specify name of the external palette file (`TEDPaletteFile`).

`-TEDborders <mode>`
: Set TED border display mode (`TEDBorderMode`). (0: normal, 1: full, 2: debug, 3: none)

`-TEDsaturation <0-2000>`
: Set saturation of internal calculated palette (`TEDColorSaturation`).

`-TEDcontrast <0-2000>`
: Set contrast of internal calculated palette (`TEDColorContrast`).

`-TEDbrightness <0-2000>`
: Set brightness of internal calculated palette (`TEDColorBrightness`).

`-TEDgamma <0-4000>`
: Set gamma of internal calculated palette (`TEDColorGamma`).

`-TEDtint <0-2000>`
: Set tint of internal calculated palette (`TEDColorTint`).

`-TEDoddlinesphase <0-2000>`
: Set phase for color carrier in odd lines (`TEDPALOddLinePhase`).

`-TEDoddlinesoffset <0-2000>`
: Set phase offset for color carrier in odd lines (`TEDPALOddLineOffset`).

`-TEDcrtblur <0-1000>`
: Amount of horizontal blur for the CRT emulation (`TEDPALBlur`).

`-TEDcrtscanlineshade <0-1000>`
: Amount of scan line shading for the CRT emulation (`TEDPALScanLineShade`).

`-TEDaudioleak`
`+TEDaudioleak`
: Enable/disable video to audio leak emulation (`TEDAudioLeak=1`, `TEDAudioLeak=0`).

7.6.2 PLUS4 I/O extension settings

7.6.2.1 PLUS4 I/O extension resources

`CartridgeReset`
: Boolean specifying whether the machine should be reset when a cartridge is changed.

`DIGIBLASTER`
: Boolean to enable/disable the Digiblaster emulation.

`SpeechEnabled`
: Boolean to enable/disable the emulation of the V364 speech chip.

`SidCart` Boolean specifying whether SID-Cart emulation is enabled or not.

`SidAddress`
: Integer that specifies the base address of the emulated SID chip. (0xFD40, 0xFE80)

`SidClock` Integer specifying the clock rate used for the emulated SID chip (0: C64, 1: Plus4)

`SIDCartJoy`
: Boolean to enable/disable the emulation of the SID-Cart control port.

7.6.2.2 PLUS4 I/O extension command-line options

`-cartreset`
`+cartreset`
: Do/don't reset machine if a cartridge is attached or detached (`CartridgeReset=1`, `CartridgeReset`).

`-cart <Name>`
`+cart` Smart-attach cartridge image / Start without cartridge

`-digiblaster`
`+digiblaster`
: Enable/disable the digiblaster add-on (`DIGIBLASTER=1`, `DIGIBLASTER=0`).

`-sidcart`

`+sidcart` Enable/disable SID Cartridge (`SidCart=1`, `SidCart=0`).

`-sidcartjoy`
`+sidcartjoy`
: Enable/disable SID cartridge control port (`SIDCartJoy=1`, `SIDCartJoy=0`).

`-sidcartaddress <address>`
: Specify address of the SID Cartridge (`SidAddress`). (0xFD40, 0xFE80)

`-sidcartclock <clock>`
: Specify clock of the SID Cartridge (`SidClock`). (0: C64, 1: Plus4)

`-speech`
`+speech` Enable/Disable the v364 speech add-on (`SpeechEnabled=1`, `SpeechEnabled=0`).

7.6.3 PLUS4 system ROM settings

7.6.3.1 PLUS4 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`).

`FunctionLowName`
: String specifying the filename of the Lo Function ROM.

`FunctionHighName`
: String specifying the filename of the High Function ROM.

`c1loName` String specifying the filename of cartridge 1 Lo ROM.

`c1hiName` String specifying the filename of cartridge 1 High ROM.

`c2loName` String specifying the filename of cartridge 2 Lo ROM.

`c2hiName` String specifying the filename of cartridge 2 High ROM.

7.6.3.2 PLUS4 system ROM command-line options

`-kernal <Name>`
: Specify name of the Kernal ROM image (`KernalName`).

`-basic <Name>`
: Specify name of the BASIC ROM image (`BasicName`).

`-functionlo <name>`
: Specify name of Function low ROM image (`FunctionLowName`).

`-functionhi <name>`
: Specify name of Function high ROM image (`FunctionHighName`).

`-c1lo <name>`
: Specify name of Cartridge 1 low ROM image (`c1loName`).

`-c1hi <name>`
: Specify name of Cartridge 1 high ROM image (`c1HiName`).

`-c2lo <name>`
: Specify name of Cartridge 2 low ROM image (`c2loName`).

`-c2hi <name>`
: Specify name of Cartridge 2 high ROM image (`c2HiName`).

7.6.4 PLUS4 settings

7.6.4.1 PLUS4 resources

`MachineVideoStandard`
: Integer that specifies the video standard of the emulated machine (1: PAL, 2: NTSC).

`RamSize` Integer specifying the size of the memory in KiB. (16, 32, 64)

`MemoryHack`
Integer specifying what memory expansion hack is active. (0: None, 1: C256K, 2: H256K, 3: H1024K, 4: H4096K)

`Acia1Enable`
Boolean specifying whether the ACIA should be emulated or not.

7.6.4.2 PLUS4 command-line options

`-model <model>`
Specify the PLUS4 model you want to emulate (`MachineVideoStandard`, `RamSize`, `KernalName`, `BasicName`, `FunctionLowName`, `FunctionHighName`, `c2loName`, `Acia1Enable`, `SpeechEnabled`). (c16/c16pal/c16ntsc, plus4/plus4pal/plus4ntsc, v364/cv364, c232)

`-pal` Use PAL sync factor (`MachineVideoStandard=1`).

`-ntsc` Use NTSC sync factor (`MachineVideoStandard=2`).

`-ramsize <ramsize>`
Specify size of RAM installed in KiB (`RamSize`). (16, 32, 64)

`-memoryexphack <device>`
Set active memory expansion hack (`MemoryHack`). (0: None, 1: C256K, 2: H256K, 3: H1024K, 4: H4096K)

`-acia`
`+acia` Enable/Disable the ACIA emulation

`-userportdac`
`+userportdac`
Enable/disable the userport DAC (`UserportDAC=1`, `UserportDAC=0`).

7.7 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.

7.7.1 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` Integer specifying the size of the memory in KiB. 96KiB denotes an 8096, 128KiB an 8296. (4, 8, 16, 32, 96, 128)

`IOSize` Integer specifying the size of the I/O area in Bytes for the 8296 model. (256, 2048)

`VideoSize`
: Integer specifying the number of columns on the screen. (0: Automatic from ROM, 40: 40 columns, 80: 80 columns)

`Ram9`
: Boolean specifying if the 8296 can map RAM into the address range $9*** or not.

`RamA`
: Boolean specifying if the 8296 can map RAM into the address range $A*** or not.

`SuperPET`
: Boolean that enables/disables the SuperPET (MicroMainFrame 9000) I/O and 6809 CPU, and disables/enables the 8x96 mappings.

`Basic1`
: Boolean to enable/disable the patching of a version 1 kernal ROM to make the IEEE488 interface work.

`Basic1Chars`
: Boolean to enable/disable the exchanges of some characters in the character ROM that have changed between the first PET 2001 and all newer versions.

`EoiBlank`
: Boolean to enable/disable the "blank screen on EOI" feature of the oldest PET 2001.

`DiagPin`
: Boolean to enable/disable the diagnostic pin on the PET userport (see below).

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

`KernalName`
: String specifying the name of the Kernal ROM (default `kernal4`).

`BasicName`
: String specifying the name of the Basic ROM (default `basic4`).

`EditorName`
: String specifying the filename of the editor ROM file. This file contains an overlay for the editor ROM at $E000-$E7FF if necessary.

`RomModule9Name`
: String specifying the filename for the $9*** Expansion ROM file. This file contains an expansion ROM image of 4KiB.

`RomModuleAName`
: String specifying the filename for the $A*** Expansion ROM file. This file contains an expansion ROM image of 4KiB.

`RomModuleBName`
: String specifying the filename for the $B*** Expansion ROM file. This file contains an expansion ROM image of 4KiB. This file overlays the lowest 4KiB of a BASIC 4 ROM.

Choosing a common PET model is done from the setting dialog, by choosing an item from the “Model” selection. 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.

7.7.2 CRTC Settings

7.7.2.1 CRTC resources

`Crtc` Boolean specifying whether to enable/disable 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.

`CrtcStretchVertical`
Boolean specifying whether vertical stretching is turned on.

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

`CrtcExternalPalette`
Boolean specifying whether to use external palette file or not.

`CrtcColorSaturation`
Integer specifying saturation of internal calculated palette. (0..2000)

`CrtcColorContrast`
Integer specifying contrast of internal calculated palette. (0..2000)

`CrtcColorBrightness`
Integer specifying brightness of internal calculated palette. (0..2000)

`CrtcColorGamma`
: Integer specifying gamma of internal calculated palette. (0..4000)

`CrtcColorTint`
: Integer specifying tint of internal calculated palette. (0..2000)

`CrtcPALScanLineShade`
: Integer specifying amount of scan line shading for the CRT emulation. (0..1000)

`CrtcPALBlur`
: Integer specifying amount of horizontal blur for the CRT emulation. (0..1000)

`CrtcPALOddLinePhase`
: Integer specifying phase for color carrier in odd lines. (0..2000)

`CrtcPALOddLineOffset`
: Integer specifying phase offset for color carrier in odd lines. (0..2000)

`CrtcAudioLeak`
: Boolean specifying whether to enable/disable video to audio leak emulation.

`CrtcFilter`
: Integer specifying rendering filter. (0: None, 1: CRT emulation, 2: Scale2x)

7.7.2.2 CRTC command-line options

`-crtc`
`+crtc`
: Enable/disable the 6545 CRTC emulation (`Crtc=1`, `Crtc=0`).

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

`-Crtcdsize`
`+Crtcdsize`
: Enable/disable double size (`CrtcDoubleSize=1`, `CrtcDoubleSize=0`).

`-CRTCstretchvertical`
`+CRTCstretchvertical`
: Enable/disable vertical stretching (`CrtcStretchVertical=1`, `CrtcStretchVertical=0`).

`-Crtcdscan`
`+Crtcdscan`
: Enable/disable double scan (`CrtcDoubleScan=1`, `CrtcDoubleScan=0`).

`-Crtcfilter <Mode>`
: Select rendering filter (`CrtcFilter`). (0: None, 2: CRT emulation, 3: Scale2x)

`-Crtcintpal`
: Use an internal calculated palette (`CrtcExternalPalette=0`).

`-Crtcextpal`
: Use an external palette (file) (`CrtcExternalPalette=1`).

`-Crtcpalette <name>`
: Specify the name of the palette file (`CrtcPaletteFile`).

`-Crtcsaturation <0-2000>`
Set saturation of internal calculated palette (`CrtcColorSaturation`).

`-Crtccontrast <0-2000>`
Set contrast of internal calculated palette (`CrtcColorContrast`).

`-Crtcbrightness <0-2000>`
Set brightness of internal calculated palette (`CrtcColorBrightness`).

`-Crtcgamma <0-4000>`
Set gamma of internal calculated palette (`CrtcColorGamma`).

`-Crtctint <0-2000>`
Set tint of internal calculated palette (`CrtcColorTint`).

`-Crtcoddlinesphase <0-2000>`
Set phase for color carrier in odd lines (`CrtcPALOddLinePhase`).

`-Crtcoddlinesoffset <0-2000>`
Set phase offset for color carrier in odd lines (`CrtcPALOddLineOffset`).

`-Crtccrtblur <0-1000>`
Amount of horizontal blur for the CRT emulation (`CrtcPALBlur`).

`-Crtccrtscanlineshade <0-1000>`
Amount of scan line shading for the CRT emulation (`CrtcPALScanLineShade`).

`-Crtcaudioleak`
`+Crtcaudioleak`
Enable/disable video to audio leak emulation (`CrtcAudioLeak=1`, `CrtcAudioLeak=0`).

7.7.3 PET I/O extension settings

7.7.3.1 PET I/O extension resources

`PETREU` Boolean specifying whether PET REU emulation is enabled or not.

`PETREUfilename`
String specifying the filename of the PET REU image.

`PETREUsize`
Integer specifying the size of the emulated PET REU in KiB. (128, 512, 1024, 2048)

`PETDWW` Boolean specifying whether DWW emulation is enabled or not.

`PETDWWfilename`
String specifying the filename of the DWW image RAM image.

`PETHRE` Boolean specifying whether HRE emulation is enabled or not.

`PETColour`
Integer selecting the colour extension. (0: off, 1: RGBI, 2: Analog)

`PETColourBG`
Iteger specifying the analog colour background on PET 4032. (0..255)

`UserportDAC`
: Boolean specifying whether userport DAC emulation is enabled.

`SidCart` Boolean specifying whether SID Cart emulation is enabled or not.

`SidAddress`
: Integer that specifies the base address of the emulated SID chip. (0x8F00, 0xE900)

`SidClock` Integer specifying the clock rate used for the emulated SID chip. (0: C64, 1: PET)

7.7.3.2 PET I/O extension command-line options

`-petreu`
`+petreu` Enable/disable the PET Memory Expansion Unit (`PETREU=1`, `PETREU=0`).

`-petreuimage <name>`
: Specify name of PET Ram and Expansion Unit image (`PETREUfilename`).

`-petreuramsize <size in KiB>`
: Size of the PET Ram and Expansion Unit (`PETREUsize`). (128, 512, 1024, 2048)

`-userportdac`
`+userportdac`
: Enable/disable the userport DAC (`UserportDAC=1`, `UserportDAC=0`).

`-petdww`
`+petdww` Enable/disable the PET DWW hi-res board (30xx models only) (`PETDWW=1`, `PETDWW=0`).

`-petdwwimage <name>`
: Specify name of PET DWW RAM image (`PETDWWfilename`).

`-pethre`
`+pethre` Enable/disable the PET HRE extension (model 8296 only) (`PETHRE=1`, `PETHRE=0`).

`-colour-rgbi`
: Enable the RGBI colour extension to PET 4032 (`PETColour=1`).

`-colour-analog`
: Enable the analog colour extension to PET 4032 (`PETColour=2`).

`-colour-analog-bg <Colour 0-255>`
: Select the analog colour background on PET 4032 (`PETColourBG`).

`-sidcart`
`+sidcart` Enable/disable the SID Cartridge (`SidCart=1`, `SidCart=0`).

`-sidcartaddress <address>`
: Specify address of the SID Cartridge (`SidAddress`). (0x8F00, 0xE900)

`-sidcartclock <clock>`
: Specify clock of the SID Cartridge (`SidClock`). (0: C64, 1: PET)

7.7.4 PET system ROM settings

7.7.4.1 PET system ROM resources

`H6809RomAName`
`H6809RomBName`
`H6809RomCName`
`H6809RomDName`
`H6809RomEName`
`H6809RomFName`

Strings specifying the filenames of the respective H6809 ROM images, relevant for the SuperPET.

7.7.4.2 PET system ROM command-line options

`-kernal <name>`

Specify filename of the Kernal ROM file (`KernalName`).

`-basic <name>`

Specify filename of the BASIC ROM file (`BasicName`).

`-editor NAME`

Specify the filename for the editor ROM file (`EditorName`).

`-chargen <name>`

Specify filename of the character generator ROM file (`ChargenName`).

`-rom9 <name>`

Specify the filename for the ROM image file for the $9*** cartridge area (`RomModule9Name`).

`-romA <name>`

Specify the filename for the ROM image file for the $A*** cartridge area (`RomModuleAName`).

`-romB <name>`

Specify the filename for the ROM image file for the $B*** cartridge area (`RomModuleBName`).

`-6809romA <Name>`

Specify 4KiB to 24KiB ROM file name at $A000 for 6809 (`H6809RomAName`).

`-6809romB <Name>`

Specify 4KiB to 20KiB ROM file name at $B000 for 6809 (`H6809RomBName`).

`-6809romC <Name>`

Specify 4KiB to 16KiB ROM file name at $C000 for 6809 (`H6809RomCName`).

`-6809romD <Name>`

Specify 4KiB to 12KiB ROM file name at $D000 for 6809 (`H6809RomDName`).

`-6809romE <Name>`

Specify 2KiB or 8KiB ROM file name at $E000 for 6809 (`H6809RomEName`).

`-6809romF <Name>`

Specify 4KiB ROM file name at $F000 for 6809 (`H6809RomFName`).

7.7.5 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.

7.7.6 PET settings

7.7.6.1 PET resources

`MachineVideoStandard`
: Integer that specifies the video standard of the emulated machine (1: PAL, 2: NTSC).

`CPUswitch`
: Integer specifying the status of the SuperPET CPU switch (0: 6502, 1: 6809, 2: Prog.)

7.7.6.2 PET command line options

These are the commandline options specific for the PET models.

`-pal` Use PAL sync factor (`MachineVideoStandard=1`).

`-ntsc` Use NTSC sync factor (`MachineVideoStandard=2`).

`-model <model>`
: Specify the PET model you want to emulate (`Basic1`, `Basic1Chars`, `ChargenName`, `KernalName`, `BasicName`, `EditorName`, `RomModule9Name`, `RomModuleAName`, `RomModuleBName`, `H6809RomAName`, `H6809RomBName`, `H6809RomCName`, `H6809RomDName`, `H6809RomEName`, `H6809RomFName`). (2001, 3008, 3016, 3032, 3032B, 4016, 4032, 4032B, 8032, 8096, 8296, SuperPET)

`-iosize <size>`
: Specify size of I/O in Bytes (`IOSize`). (256, 2048)

`-ramsize <size>`
: Specify size of RAM in KiB (`RamSize`). (8, 16, 32, 96, 128)

`-videosize <size>`
: Specify video size (`VideoSize`). (0: Automatic from ROM, 40: 40 columns, 80: 80 columns)

`-petram9`
`+petram9` Switch on/off RAM mapping at address $9000-$9FFF (`Ram9=1`, `Ram9=0`).

`-petramA`
`+petramA` Switch on/off RAM mapping at address $A000-$AFFF (`RamA=1`, `RamA=0`).

`-superpet`
`+superpet`
: Enable/disable SuperPET I/O and CPU emulation (`SuperPET`).

`-cpu6502` Set SuperPET CPU switch to '6502' (`CPUswitch=0`).

`-cpu6809` Set SuperPET CPU switch to '6809' (`CPUswitch=1`).

`-cpuprog` Set SuperPET CPU switch to 'Prog' (`CPUswitch=2`).

`-basic1`
`+basic1` Enable/disable patching the IEEE488 section of the PET2001 ROM when detected (`Basic1=1`, `Basic1=0`).

`-basic1char`
`+basic1char`
Enable/disable PET 2001 character generator (`Basic1Chars=1`, `Basic1Chars=0`).

`-eoiblank`
`+eoiblank`
Enable/disable EOI blanking the screen (`EoiBlank=1`, `EoiBlank=0`).

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

7.7.7 Colour PET

The Colour PET is a simple extension by Steve Gray `http://www.6502.org/users/sjgray/projects/colourpet/`. It exploits the similarities and differences between the 4032 and 8032 models, to use memory at $8800 as colour RAM.

There are two versions of the extension:

with digital colour (RGBI), which can select 16 foreground and 16 background colours for each character cell. The 8 colour bits are used as RGBI RGBI, with the most significant bits for the background, and the least significant bits for the foreground.

With analog colour, which can use 256 foreground colours for each character cell, the 8 colour bits are used as RRR GGG BB.

7.7.8 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 24) 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.

7.7.9 DWW high resolution graphics

The DWW, or Double-W[1], board is a high resolution graphics board for PET models 30xx. It attaches to the internal expansion connector. It would not physically fit in 20xx, 40xx or 80xx models because their connectors are physically and logically different. Apart from this, it requires address space at $EC00-$EFFF, which makes it unfit to work in 40xx and 80xx models, which have only 256 bytes of I/O space at $E800.

[1] Dubbel-W bord, designed by Ben de Winter and Pieter Wolvekamp

The DWW board consists of 8 KiB of RAM and a PIA to control various options. The RAM can be mapped either linearly at $9000-$AFFF, or bank-switched in 8 banks of 1 KiB each at $EC00-$EFFF.

It seems that in the demo programs, the BASIC versions like to POKE in 60200, where the machine language programs use $EB00.

```
60200 Port A or DDR A           $EB28
$EBx0 1  0 \
      2  1 - RAM block at $EC00 (0-7)
      4  2 /
      8  3 charrom         0 = off 1 = on
      16 4 hires           0 = on  1 = off
      32 5 extra charrom   0 = on  1 = off

60201 Control Register A: bit #3 (worth 4) controls if 60200
$EBx1 accesses the Data Direction Register A (0) or Port A (1).

60202 Port B or DDR B
$EBx2 0 = RAM is visible from $9000 - $AFFF
      1 = RAM is bank-switched in blocks of 1 K in $EC00 - $EFFF

      [Control Register B is never mentioned, so putting 1 in this
       address would access the DDR, creating an output line, which
       after RESET is default 0...]
```

Typical initialisation sequence:

```
    poke 60201,0        poke 60200,255          (all outputs)
    poke 60201,4        poke 60200,24 or 25 (16 + 8 + 1)
```

Demo programs on disk PBE-110A, 110B, 111A, and 111B. (PBE = PET Benelux Exchange, the Dutch PET user group)

The memory mapping is a bit strange. It seems each 1 K block contains the pixeldata for 1 bit-line of each text line. This is probably so that the addressing of the RAM can borrow part of the addressing logic/signals of the text screen. (The screen addressing cycles through 0-39, then increases the line (= byte offset) which is fetched from the character ROM; for the graphics, the screen position selects the byte in a KiB and the char ROM offset selects which KiB of graphics RAM).

My notes say: to set a pixel:

```
RE = INT(Y/8): LY = Y - 8*RE    (or Y AND 7)
BY = INT(X/8): BI = X - 8*BY    (or X AND 7)
```

when memory mapped to $9000:

```
    L = 36864 + 1024 * LY + 40 * RE + BY
    POKE L, PEEK(L) OR 2^BI
```

when memory mapped to $EC00:

```
    POKE 60200,LY + 40 (or 8?)
    L = 60416 + RE * 40 + BY
    POKE L, PEEK(L) OR 2^BI
```

Unfortunately there is no logical means of expanding the memory to 16 K, so even in an 80 columns PET the resolution will be the same.

7.7.10 HRE high resolution graphics

This information comes from Michal Pleban, `http://www.cbm-ii.com`.

This is a short information about the HRE (HiRes Emulator) graphics board for PET 8296. This is a small board that is installed on the mainboard into sockets for the CRTC and character ROM. It adds the ability to display 512x256 graphics on the built-in monitor.

The board is accompanied with two extension ROMs:

- 324992-02 HiRes Emulator (at $9000)
- 324993-02 HiRes BASIC (at $A000)

The ROMs are initialized by the command `SYS 36864`.

After that, the computer recognizes additional BASIC commands like DISPLAY, PLOT and so on; these commands are identical to those of the HiRes Graphics boards based on Thomson chips. All BASIC programs written for the HRG boards should also work on the HRE (albeit much slower).

Upon initialization, the ROM code writes some routines into expansion RAM as $8800 - they are used to manipulate the bitmap data. The actual bitmap is located in expansion RAM at addresses $A000-$DFFF, and is organized linearly (first 64 bytes form the first display line, then comes the next line and so on).

The RAM is accessed for reading by writing the value #$83 into $E888. This is a register in the CRTC memory space that is intercepted by the board and serves as a latch to drive jumpers on the 8296D mainboard. Because the ROMs are banked out this way, all video memory manipulation must happen with interrupts disabled. Normal ROM operation is restored by writing #$0F into $E888.

As usual with the 8296, the RAM can also be written even when the ROMs are mapped normally.

The board is initialized into graphics mode by modifying the following values in CRTC registers (it is done by the routine located at $8C1A):

- Register $01 (horizontal displayed) = #$20
- Register $02 (horizontal sync position) = original `+` #$02
- Register $06 (vertical displayed) = #$20
- Register $07 (vertical sync position) = original `+` #$04
- Register $0C (RAM address high) = #$02

The last value causes the board to switch into graphics mode; former values set up proper screen dimensions for 512x256 display.

7.8 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.

7.8.1 Changing CBM-II model

With `xcbm2` and `xcbm5x0`, 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):

`RamSize` Size of memory in KiB. xcbm2: (128, 256, 512, 1024) xcbm5x0: (64, 128, 256, 512, 1024)

`Ram08` Boolean specifying whether the RAM should be mapped to the expansion ROM area at $0800-$0FFF or not.

`Ram1` Boolean specifying whether the RAM should be mapped to the expansion ROM area at $1000-$1FFF or not.

`Ram2` Boolean specifying whether the RAM should be mapped to the expansion ROM area at $2000-$3FFF or not.

`Ram4` Boolean specifying whether the RAM should be mapped to the expansion ROM area at $4000-$5FFF or not.

`Ram6` Boolean specifying whether the RAM should be mapped to the expansion ROM area at $6000-$7FFF or not.

`RamC` Boolean specifying whether the RAM should be mapped to the expansion ROM area at $C000-$CFFF or not.

`CartridgeReset`
Boolean specifying whether the machine should be reset when a cartridge is changed.

`Cart1Name`
String specifying the filename for the $1000-$1FFF expansion ROM file. This file contains a 4KiB ROM dump.

`Cart2Name`
String specifying the filename for the $2000-$3FFF expansion ROM file. This file contains an 8KiB ROM dump.

`Cart4Name`
String specifying the filename for the $4000-$5FFF expansion ROM file. This file contains an 8KiB ROM dump.

`Cart6Name`
String specifying the filename for the $6000-$7FFF expansion ROM file. This file contains an 8KiB 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. xcbm2: (0: CBM 7x0 (50Hz), 1: 60Hz C6x0, 2: 50Hz C6x0) xcbm5x0: (1: 60Hz C5x0, 2: 50Hz C5x0)

`ChargenName`
: String specifying the name of the character generator ROM (default for xcbm2: `chargen.600`, default for xcbm5x0: `chargen.500`).

`KernalName`
: String specifying the name of the Kernal ROM (default for xcbm2: `kernal`, default for xcbm5x0: `kernal.500`).

`BasicName`
: String specifying the name of the Basic ROM (default for xcbm2: `basic.128`, default for xcbm5x0: `basic.500`).

`MachineVideoStandard`
: Integer that specifies the video standard of the emulated machine (1: PAL, 2: NTSC).

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 PAL or NTSC (128KiB RAM)
- C610 PAL or NTSC (128KiB RAM)
- C620 (256KiB RAM)
- C620+ (1024KiB RAM, expanded) PAL or NTSC
- C710 (128KiB RAM) NTSC
- C720 (256KiB RAM) NTSC
- C720+ (1024KiB RAM, expanded) NTSC

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

Warning: At this time the 5x0 and other machines are implemented in different executables, so switching between those models is not possible.

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.

7.8.2 CBM-II system ROM settings

7.8.2.1 CBM-II system ROM resources

7.8.2.2 CBM-II system ROM command line options

`-kernal <name>`
: Specify the name of the Kernal ROM file (`KernalName`).

`-basic <name>`
: Specify the name of the Basic ROM file (`BasicName`).

`-chargen <name>`
: Specify the filename for the character generator ROM file (`ChargenName`).

`+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 (`CartridgeReset=1`, `CartridgeReset=0`).

`-cart1 <name>`
: Specify the filename for the ROM image file for the cartridge area $1000-$1FFF (`Cart1Name`).

`-cart2 <name>`
: Specify the filename for the ROM image file for the cartridge area $2000-$2FFF (`Cart2Name`).

`-cart4 <name>`
: Specify the filename for the ROM image file for the cartridge area $4000-$5FFF (`Cart4Name`).

`-cart6 <name>`
: Specify the filename for the ROM image file for the cartridge area $6000-$7FFF (`Cart6Name`).

7.8.3 CBM-II command line options

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

`-pal` Use PAL sync factor (`MachineVideoStandard=1`).

`-ntsc` Use NTSC sync factor (`MachineVideoStandard=2`).

`-ramsize <ramsize>`
: Specify size of RAM (`RamSize`). xcbm2: (128, 256, 512, 1024) xcbm5x0: (64, 128, 256, 512, 1024)

`-ram08`
`+ram08` Enable/disable RAM mapping in bank 15 at address $0800-$0FFF (`Ram08=1`, `Ram08=0`).

`-ram1`
`+ram1` Enable/disable RAM mapping in bank 15 at address $1000-$1FFF (`Ram1=1`, `Ram1=0`).

`-ram2`
`+ram2` Enable/disable RAM mapping in bank 15 at address $2000-$3FFF (`Ram2=1`, `Ram2=0`).

`-ram4`
`+ram4` Enable/disable RAM mapping in bank 15 at address $4000-$5FFF (`Ram4=1`, `Ram4=0`).

`-ram6`
`+ram6` Enable/disable RAM mapping in bank 15 at address $6000-$7FFF (`Ram6=1`, `Ram6=0`).

`-ramC`
`+ramC` Enable/disable RAM mapping in bank 15 at address $C000-$CFFF (`RamC=1`, `RamC=0`).

`-model <modelnumber>`
: Specify CBM-II model to emulate (`ModelLine`, `MachineVideoStandard`, `RamSize`, `KernalName`, `BasicName`, `ChargenName`). xcbm2: (610, 620, 620+, 710, 720, 720+) xcbm5x0: (510)

7.8.4 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 24) 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.

7.9 VSID-specific commands and settings

7.9.1 VSID settings

7.9.1.1 VSID resources

`PSIDKeepEnv`
: Boolean that specifies whether to override PSID settings for Video standard and SID model.

`PSIDTune` Integer that specifies the currently played sub tune.

`HVSCRoot` String specifying the location of the HVSC `"C64Music"` directory.

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

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

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

`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.

`MachineVideoStandard`
Integer that specifies the video standard of the emulated machine (4: PAL-N, 3: Old NTSC, 1: PAL, 2: NTSC).

7.9.1.2 VSID command-line options

`-keepenv` Override PSID settings for Video standard and SID model (`PSIDKeepEnv=1`).

`-tune <number>`
Specify PSID tune <number> (`PSIDTune`).

`-hvsc-root <path>`
Specify the location of the HVSC "C64Music" directory. (`HVSCRoot`).

`-chargen <name>`
Specify name of character generator ROM image (`ChargenName`).

`-kernal <name>`
Specify name of the Kernal ROM image (`KernalName`).

`-basic <name>`
Specify name of the Basic ROM image (`BasicName`).

`-pal` Use PAL sync factor (`MachineVideoStandard=1`).

`-ntsc` Use NTSC sync factor (`MachineVideoStandard=2`).

`-ntscold` Use old NTSC sync factor (`MachineVideoStandard=3`).

`-paln` Use PAL-N sync factor (`MachineVideoStandard=4`).

8 Platform-specific features

This section lists the settings and commands that are platform specific.

8.1 SDL-specific features

This section lists the settings and commands that are common and specific to SDL, and are thus not present in other platforms.

8.1.1 SDL specific resources

`HotkeyFile`
: String specifying the name of the hotkey file (all emulators except vsid).

`MenuKey`
: Integer specifying the keycode for activating the SDL menu.

`MenuKeyUp`
: Integer specifying the keycode for 'up' in the SDL menu.

`MenuKeyDown`
: Integer specifying the keycode for 'down' in the SDL menu.

`MenuKeyLeft`
: Integer specifying the keycode for 'left' in the SDL menu.

`MenuKeyRight`
: Integer specifying the keycode for 'right' in the SDL menu.

`MenuKeyPageUp`
: Integer specifying the keycode for 'page up' in the SDL menu.

`MenuKeyPageDown`
: Integer specifying the keycode for 'page down' in the SDL menu.

`MenuKeyHome`
: Integer specifying the keycode for 'home' in the SDL menu.

`MenuKeyEnd`
: Integer specifying the keycode for 'end' in the SDL menu.

`MenuKeySelect`
: Integer specifying the keycode for 'select' in the SDL menu.

`MenuKeyCancel`
: Integer specifying the keycode for 'cancel' in the SDL menu.

`MenuKeyExit`
: Integer specifying the keycode for 'exit' in the SDL menu.

`MenuKeyMap`
: Integer specifying the keycode for 'map' in the SDL menu.

`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.

`ConfirmOnExit`
: Boolean specifying whether to show a confirmation dialog on exit.

`PauseOnSettings`
: Boolean specifying wheter to pause the emulation when triggering the settings dialog. If set, the emulation will pause when triggering the settings dialog. When closing the settings dialog the pause state will revert to its state before triggering the settings UI.

`SDLStatusbar`
: Boolean to enable/disable the statusbar (all emulators except vsid).

`SDLBitdepth`
: Integer specifying the bitdepth used. (0: current, 8, 15, 16, 24, 32)

`SDLLimitMode`
: Integer specifying the resolution limit mode. (0: off, 1: max, 2: fixed)

`SDLCustomWidth`
: Integer specifying the custom resolution width.

`SDLCustomHeight`
: Integer specifying the custom resolution height.

`KbdStatusbar`
: Boolean specifying whether the keyboard-status bar is enabled.

`SDL2Renderer`
: String specifying the preferred SDL2 renderer.

`SDLWindowWidth`
: Integer specifying the SDL window width.

`SDLWindowHeight`
: Integer specifying the SDL window height.

`SDLGLFilter`
: Integer specifying the OpenGL filtering mode.

`JoyDevice1`
: Integer specifying which joystick device the emulator should use for the emulation of joystick 1 (all emulators except xcbm2, xpet and vsid). (0: None, 1: Numpad, 2: Keyset 1, 3: Keyset 2, 4: Joystick (only if joystick support was present in the compile time SDL library))

`JoyDevice2`
: Integer specifying which joystick device the emulator should use for the emulation of joystick 2 (all emulators except xcbm2, xpet, xvic and vsid). (0: None, 1: Numpad, 2: Keyset 1, 3: Keyset 2, 4: Joystick (only if joystick support was present in the compile time SDL library))

`JoyDevice3`
: Integer specifying which joystick device the emulator should use for the emulation of extra joystick 1 (all emulators except xcbm5x0 and vsid). (0: None, 1: Numpad, 2: Keyset 1, 3: Keyset 2, 4: Joystick (only if joystick support was present in the compile time SDL library))

`JoyDevice4`
: Integer specifying which joystick device the emulator should use for the emulation of extra joystick 2 (all emulators except xcbm5x0, xplus4 and vsid). (0: None, 1: Numpad, 2: Keyset 1, 3: Keyset 2, 4: Joystick (only if joystick support was present in the compile time SDL library))

`CrtcSDLFullscreenMode`
: Integer specifying the fullscreen mode (xcbm2 and xpet only).

`CrtcHwScale`
: Boolean specifying whether to enable or disable hardware scaling (xcbm2 and xpet only).

`CrtcFullscreenDevice`
: String specifying the fullscreen device (xcbm2 and xpet only).

`CrtcFullscreen`
: Boolean specifying whether to use fullscreen mode or not (xcbm2 and xpet only).

`CrtcFullscreenStatusbar`
: Boolean specifying whether to show the status bar in fullscreen mode or not (xcbm2 and xpet only).

`TEDSDLFullscreenMode`
: Integer specifying the fullscreen mode (xplus4 only).

`TEDHwScale`
: Boolean specifying whether to enable or disable hardware scaling (xplus4 only).

`TEDFullscreenDevice`
: String specifying the fullscreen device (xplus4 only).

`TEDFullscreen`
: Boolean specifying whether to use fullscreen mode or not (xplus4 only).

`TEDFullscreenStatusbar`
: Boolean specifying whether to show the status bar in fullscreen mode or not (xplus4 only).

`VDCSDLFullscreenMode`
: Integer specifying the fullscreen mode (x128 only).

`VDCHwScale`
: Boolean specifying whether to enable or disable hardware scaling (x128 only).

`VDCFullscreenDevice`
: String specifying the fullscreen device (x128 only).

`VDCFullscreen`
: Boolean specifying whether to use fullscreen mode or not (x128 only).

`VDCFullscreenStatusbar`
: Boolean specifying whether to show the status bar in fullscreen mode or not (x128 only).

`VICSDLFullscreenMode`
: Integer specifying the fullscreen mode (xvic only).

`VICHwScale`
: Boolean specifying whether to enable or disable hardware scaling (xvic only).

`VICFullscreenDevice`
: String specifying the fullscreen device (xvic only).

`VICFullscreen`
: Boolean specifying whether to use fullscreen mode or not (xvic only).

`VICFullscreenStatusbar`
: Boolean specifying whether to show the status bar in fullscreen mode or not (xvic only).

`VICIISDLFullscreenMode`
: Integer specifying the fullscreen mode (all emulators except xcbm2, xpet, xplus4, xvic and vsid).

`VICIIHwScale`
: Boolean specifying whether to enable or disable hardware scaling (all emulators except xcbm2, xpet, xplus4, xvic and vsid).

`VICIIFullscreenDevice`
: String specifying the fullscreen device (all emulators except xcbm2, xpet, xplus4, xvic and vsid).

`VICIIFullscreen`
: Boolean specifying whether to use fullscreen mode or not (all emulators except xcbm2, xpet, xplus4, xvic and vsid).

`VICIIFullscreenStatusbar`
: Boolean specifying whether to show the status bar in fullscreen mode or not (all emulators except xcbm2, xpet, xplus4, xvic and vsid).

The following resources are only present if the OpenGL library is present and used at compile time.

`AspectRatio`
: String/float specifying the aspect ratio. (0.5-2.0)

`SDLGLAspectMode`
: Integer specifying the OpenGL aspect mode. (0: off, 1: custom, 2: true)

`SDLGLFlipX`
: Boolean to enable/disable OpenGL horizontal screen flip.

`SDLGLFlipY`
: Boolean to enable/disable OpenGL vertical screen flip.

The following resources are only present if the SDL library has joystick support at compile time and applies to all emulators except vsid.

`JoyMapFile`
: String specifying the name of the joystick map file.

`JoyThreshold`
: Integer specifying the joystick threshold. (0..32767)

`JoyFuzz` Integer specifying the joystick fuzz. (0..32767)

8.1.2 SDL specific command-line options

`-hotkeyfile <name>`
: Set the hotkey file name (`HotkeyFile`) (all emulators except vsid).

`-menukey <key>`
: Set the keycode of the SDL menu activation key (`MenuKey`).

`-menukeyup <key>`
: Set the keycode of the 'up' key in the SDL menu (`MenuKeyUp`).

`-menukeydown <key>`
: Set the keycode of the 'down' key in the SDL menu (`MenuKeyDown`).

`-menukeyleft <key>`
: Set the keycode of the 'left' key in the SDL menu (`MenuKeyLeft`).

`-menukeyright <key>`
: Set the keycode of the 'right' key in the SDL menu (`MenuKeyRight`).

`-menukeypageup <key>`
: Set the keycode of the 'page up' key in the SDL menu (`MenuKeyPageUp`).

`-menukeypagedown <key>`
: Set the keycode of the 'page down' key in the SDL menu (`MenuKeyPageDown`).

`-menukeyhome <key>`
: Set the keycode of the 'home' key in the SDL menu (`MenuKeyHome`).

`-menukeyend <key>`
: Set the keycode of the 'end' key in the SDL menu (`MenuKeyEnd`).

`-menukeyselect <key>`
: Set the keycode of the 'select' key in the SDL menu (`MenuKeySelect`).

`-menukeycancel <key>`
: Set the keycode of the 'cancel' key in the SDL menu (`MenuKeyCancel`).

`-menukeyexit <key>`
: Set the keycode of the 'exit' key in the SDL menu (`MenuKeyExit`).

`-menukeymap <key>`
: Set the keycode of the 'map' key in the SDL menu (`MenuKeyMap`).

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

`-confirmonexit`
: Confirm quitting VICE (`ConfirmOnExit=1`).

`+confirmonexit`
: Never confirm quitting VICE (`ConfirmOnExit=0`).

`-pauseonsettings`
: Pause emulation when enabling the settings dialog. (`PauseOnSettings=1`).

`+pauseonsettings`
: Do not pause emulation when enabling the settings dialog. (`PauseOnSettings=0`).

`-statusbar`
`+statusbar`
: Enable/disable the statusbar (`SDLStatusbar=1`, `SDLStatusbar=0`) (all emulators except vsid).

`-sdlbitdepth <bpp>`
: Set the display bitdepth (`SDLBitdepth`). (0: current, 8, 15, 16, 24, 32)

`-sdllimitmode <mode>`
: Set the resolution limit mode (`SDLLimitMode`). (0: off, 1: max, 2: fixed)

`-sdlcustomw <width>`
: Set the custom resolution width (`SDLCustomWidth`).

`-sdlcustomh <height>`
: Set the custom resolution height (`SDLCustomHeight`).

`-sdlinitialw <width>`
: Set initial window width.

`-sdlinitialh <height>`
: Set initial window height.

`-kbdstatusbar`
`+kbdstatusbar`
: Enable/Disable keyboard-status bar (when status bar is enabled) (`KbdStatusbar=1`, `KbdStatusbar=0`).

`-sdlglfilter <mode>`
: Set OpenGL filtering mode (0 = nearest, 1 = linear) (`SDLGLFilter`).

`-sdl2renderer <renderer name>`
: Set the preferred SDL2 renderer (`SDL2Renderer`).

`-joydev1 <0-3> / <0-4>`
: Set the device for joystick emulation of port 1 (`JoyDevice1`) (all emulators except xcbm2, xpet and vsid). (0: None, 1: Numpad, 2: Keyset 1, 3: Keyset 2, 4: Joystick (only if joystick support was present in the compile time SDL library))

`-joydev2 <0-3> / <0-4>`
: Set the device for joystick emulation of port 2 (`JoyDevice2`) (all emulators except xcbm2, xpet, xvic and vsid). (0: None, 1: Numpad, 2: Keyset 1, 3: Keyset 2, 4: Joystick (only if joystick support was present in the compile time SDL library))

`-extrajoydev1 <0-3> / <0-4>`
: Set device for extra joystick port 1 (`JoyDevice3`) (all emulators except xcbm5x0 and vsid). (0: None, 1: Numpad, 2: Keyset 1, 3: Keyset 2, 4: Joystick (only if joystick support was present in the compile time SDL library))

`-extrajoydev2 <0-3> / <0-4>`
: Set device for extra joystick port 2 (`JoyDevice4`) (all emulators except xcbm5x0, xplus4 and vsid). (0: None, 1: Numpad, 2: Keyset 1, 3: Keyset 2, 4: Joystick (only if joystick support was present in the compile time SDL library))

`-CRTCSDLfullmode <Mode>`
: Set the fullscreen mode (`CrtcSDLFullscreenMode`) (xcbm2 and xpet only).

`-CRTChwscale`
`+CRTChwscale`
: Enable/Disable hardware scaling (`CrtcHwScale=1`, `CrtcHwScale=0`) (xcbm2 and xpet only).

`-CRTCfulldevice <device>`
: Select fullscreen device (`CrtcFullscreenDevice`) (xcbm2 and xpet only).

`-CRTCfull`
`+CRTCfull`
: Enable/Disable fullscreen (`CrtcFullscreen=1`, `CrtcFullscreen=0`) (xcbm2 and xpet only).

`-TEDSDLfullmode <Mode>`
: Set the fullscreen mode (`TEDSDLFullscreenMode`) (xplus4 only).

`-TEDhwscale`
`+TEDhwscale`
: Enable/Disable hardware scaling (`TEDHwScale=1`, `TEDHwScale=0`) (xplus4 only).

`-TEDfulldevice <device>`
: Select fullscreen device (`TEDFullscreenDevice`) (xplus4 only).

`-TEDfull`
`+TEDfull` Enable/Disable fullscreen (`TEDFullscreen=1`, `TEDFullscreen=0`) (xplus4 only).

`-VDCSDLfullmode <Mode>`
: Set the fullscreen mode (`VDCSDLFullscreenMode`) (x128 only).

`-VDChwscale`
`+VDChwscale`
: Enable/Disable hardware scaling (`VDCHwScale=1`, `VDCHwScale=1`) (x128 only).

`-VDCfulldevice <device>`
: Select fullscreen device (`VDCFullscreenDevice`) (x128 only).

`-VDCfull`
`+VDCfull` Enable/Disable fullscreen (`VDCFullscreen=1`, `VDCFullscreen=0`) (x128 only).

`-VICSDLfullmode <Mode>`
: Set the fullscreen mode (`VICSDLFullscreenMode`) (xvic only).

`-VIChwscale`
`+VIChwscale`
: Enable/Disable hardware scaling (`VICHwScale=1`, `VICHwScale=0`) (xvic only).

`-VICfulldevice <device>`
: Select fullscreen device (`VICFullscreenDevice`) (xvic only).

`-VICfull`
`+VICfull` Enable/Disable fullscreen (`VICFullscreen=1`, `VICFullscreen=0`) (xvic only).

`-VICIISDLfullmode <Mode>`
: Set the fullscreen mode (`VICIISDLFullscreenMode`) (all emulators except xcbm2, xpet, xplus4, xvic and vsid).

`-VICIIhwscale`
`+VICIIhwscale`
: Enable/Disable hardware scaling (`VICIIHwScale=1`, `VICIIHwScale=0`) (all emulators except xcbm2, xpet, xplus4, xvic and vsid).

`-VICIIfulldevice <device>`
: Select fullscreen device (`VICIIFullscreenDevice`) (all emulators except xcbm2, xpet, xplus4, xvic and vsid).

`-VICIIfull`
`+VICIIfull`
: Enable/Disable fullscreen (`VICIIFullscreen=1`, `VICIIFullscreen=0`) (all emulators except xcbm2, xpet, xplus4, xvic and vsid).

The following command-line options are only present if the OpenGL library is present and used at compile time.

`-aspect <aspect ratio>`
: Set the aspect ratio (`AspectRatio`). (0.5-2.0)

`-sdlaspectmode <mode>`
: Set the aspect mode (`SDLGLAspectMode`). (0: off, 1: custom, 2: true)

`-sdlflipx`
`+sdlflipx`
: Enable/disable OpenGL horizontal screen flip (`SDLGLFlipX=1`, `SDLGLFlipX=0`).

`-sdlflipy`
`+sdlflipy`
: Enable/disable OpenGL vertical screen flip (`SDLGLFlipY=1`, `SDLGLFlipY=0`).

The following command-line options are only present if the SDL library has joystick support at compile time and applies to all emulators except vsid.

`-joymap <name>`
: Set the joystick map file name (`JoyMapFile`).

`-joythreshold <0-32767>`
: Set the joystick threshold (`JoyThreshold`).

`-joyfuzz <0-32767>`
: Set the joystick fuzz (`JoyFuzz`).

8.2 GTK3 specific features

This section lists the settings and commands that are common and specific to Unix, and are thus not present in other platforms.

8.2.1 GTK3 specific resources

`GTKBackend`
: Rendering backend to use for the Gtk3 UI (0: Cairo, 1: OpenGL).

`GTKFilter`
: Rendering filter to use for the Gtk3 UI (0: Nearest neighbor, 1: Bilinear interpolation).

`AspectRatio`
: String/float specifying the aspect ratio (0.5-2.0).

`KeepAspectRatio`
: Boolean specifying whether the aspect ratio of the output window should be preserved. (0: free scaling, 1: scale with fixed aspect ratio)

`TrueAspectRatio`
: Boolean specifying whether to use the true (non square pixels) aspect ratio.

`VSync` Boolean specifying whether to enable vsync to prevent screen tearing. (0: vsync off, 1: vsync on)

`StartMinimized`
: Boolean specifying whether the emulator should start minimized

`RestoreWindowGeometry`
: Boolean specifying whether to restore an emulator to its last-saved position and size.

`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.

`ConfirmOnExit`
: Boolean specifying whether to show a confirmation dialog on exit.

`DisplayDepth`
: Integer specifying the depth of the host display. The value '`0`' (the default) causes the emulator to autodetect it (all emulators except vsid). (0..32)

`Window0Width`
`Window0Height`
`Window0Xpos`
`Window0Ypos`
: Integers specifying the position and size of the (first) emulator window (all emulators except vsid).

`Window1Width`
`Window1Height`
`Window1Xpos`
`Window1Ypos`
: Integers specifying the position and size of the (second) emulator window (x128 only).

`CrtcHwScale`
: Boolean specifying whether to enable or disable hardware scaling (xcbm2 and xpet only).

`CrtcFullscreenDevice`
: String specifying the fullscreen device (xcbm2 and xpet only).

`CrtcFullscreen`
: Boolean specifying whether to use fullscreen mode or not (xcbm2 and xpet only).

`CrtcFullscreenStatusbar`
: Boolean specifying whether to show the status bar in fullscreen mode or not (xcbm2 and xpet only).

`TEDHwScale`
: Boolean specifying whether to enable or disable hardware scaling (xplus4 only).

`TEDFullscreenDevice`
: String specifying the fullscreen device (xplus4 only).

`TEDFullscreen`
: Boolean specifying whether to use fullscreen mode or not (xplus4 only).

`TEDFullscreenStatusbar`
: Boolean specifying whether to show the status bar in fullscreen mode or not (xplus4 only).

`VDCHwScale`
: Boolean specifying whether to enable or disable hardware scaling (x128 only).

`VDCFullscreenDevice`
: String specifying the fullscreen device (x128 only).

`VDCFullscreen`
: Boolean specifying whether to use fullscreen mode or not (x128 only).

`VDCFullscreenStatusbar`
: Boolean specifying whether to show the status bar in fullscreen mode or not (x128 only).

`VICHwScale`
: Boolean specifying whether to enable or disable hardware scaling (xvic only).

`VICFullscreenDevice`
: String specifying the fullscreen device (xvic only).

`VICFullscreen`
: Boolean specifying whether to use fullscreen mode or not (xvic only).

`VICFullscreenStatusbar`
: Boolean specifying whether to show the status bar in fullscreen mode or not (xvic only).

`VICIIHwScale`
: Boolean specifying whether to enable or disable hardware scaling (all emulators except xcbm2, xpet, xplus4, xvic and vsid).

`VICIIFullscreenDevice`
: String specifying the fullscreen device (all emulators except xcbm2, xpet, xplus4, xvic and vsid).

`VICIIFullscreen`
: Boolean specifying whether to use fullscreen mode or not (all emulators except xcbm2, xpet, xplus4, xvic and vsid).

`VICIIFullscreenStatusbar`
: Boolean specifying whether to show the status bar in fullscreen mode or not (all emulators except xcbm2, xpet, xplus4, xvic and vsid).

`JoyDevice1`
: Integer specifying which joystick device the emulator should use for the emulation of joystick 1 (all emulators except xcbm2, xpet and vsid). (0: None, 1: Numpad, 2: Keyset 1, 3: Keyset 2, 4: Analog joystick 1, 5: Analog joystick 2, 6: Analog joystick 3, 7: Analog joystick 4, 8: Analog joystick 5, 9: Analog joystick 6, 10: Digital joystick 1, 11: Digital joystick 2, 12: USB joystick 1, 13: USB joystick 2)

`JoyDevice2`
: Integer specifying which joystick device the emulator should use for the emulation of joystick 2 (all emulators except xcbm2, xpet, xvic and vsid). (0: None, 1: Numpad, 2: Keyset 1, 3: Keyset 2, 4: Analog joystick 1, 5: Analog joystick 2, 6: Analog joystick 3, 7: Analog joystick 4, 8: Analog joystick 5, 9: Analog joystick 6, 10: Digital joystick 1, 11: Digital joystick 2, 12: USB joystick 1, 13: USB joystick 2)

`JoyDevice3`
: Integer specifying which joystick device the emulator should use for the emulation of extra joystick 1 (all emulators except xcbm5x0 and vsid). (0: None, 1: Numpad, 2: Keyset 1, 3: Keyset 2, 4: Analog joystick 1, 5: Analog joystick 2, 6: Analog joystick 3, 7: Analog joystick 4, 8: Analog joystick 5, 9: Analog joystick 6, 10: Digital joystick 1, 11: Digital joystick 2, 12: USB joystick 1, 13: USB joystick 2)

`JoyDevice4`
: Integer specifying which joystick device the emulator should use for the emulation of extra joystick 2 (all emulators except xcbm5x0, xplus4 and vsid). (0: None, 1: Numpad, 2: Keyset 1, 3: Keyset 2, 4: Analog joystick 1, 5: Analog joystick 2, 6: Analog joystick 3, 7: Analog joystick 4, 8: Analog joystick 5, 9: Analog joystick 6, 10: Digital joystick 1, 11: Digital joystick 2, 12: USB joystick 1, 13: USB joystick 2)

The available joysticks might differ depending on operating system and joystick support in the OS.

`Devices 4..9`
: Only available if joystick support is available at compile time.

`Devices 10 and 11`
: Only available if digital joystick support is available at compile time.

`Devices 12 and 13`
: Only available if USB joystick support is available at compile time.

The following resources are only available if MIDI support is available at compile time.

`MIDIInDev`
: String specifying the MIDI input device (x64, x64sc, xscpu64, x128 and xvic only).

`MIDIOutDev`
: String specifying the MIDI output device (x64, x64sc, xscpu64, x128 and xvic only).

The following resource is only available if MIDI support and ALSA support is available at compile time.

`MIDIDriver`
: Enum specifying the MIDI driver (x64, x64sc, xscpu64, x128 and xvic only). (0: OSS, 1: ALSA)

8.2.2 GTK3 specific command-line options

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

`-minimized`
`+minimized`
: Start/Do not start with minimized window (`StartMinimized=1`, `StartMinimized=0`).

`-restore-window-geometry`
`+restore-window-geometry`
: Restore/ignore Window position(s) and dimension(s) stored in vicerc (`RestoreWindowGeometry=1`, `RestoreWindowGeometry=0`).

`-aspect <aspect ratio>`
: Set the aspect ratio (`AspectRatio`). (0.5-2.0)

`-gtkbackend <backend>`
: Set the Gtk3 rendering backend (`GTKBackend`) (0: Cairo, 1: OpenGL)

`-gtkfilter <filter>`
: (`GTKFilter`) (0: Nearest neighbor, 1: Bilinear interpolation)

`-keepaspect`
: Enable keeping of the aspect ratio when scaling (`KeepAspectRatio=1`).

`+keepaspect`
: Disable keeping of the aspect ratio when scaling (freescaling) (`KeepAspectRatio=0`).

`-trueaspect`
`+trueaspect`
: Enable/disable whether to use the true (non square pixels) aspect ratio (`TrueAspectRatio=1`, (`TrueAspectRatio=0`).

`-vsync` Enable vsync to prevent screen tearing (`VSync=1`).

`+vsync` Disable vsync to allow screen tearing (`VSync=0`).

`-confirmonexit`
: Confirm quitting VICE (`ConfirmOnExit=1`).

`+confirmonexit`
: Never confirm quitting VICE (`ConfirmOnExit=0`).

`-fullscreen`
`+fullscreen`
: Enable/disable fullscreen mode (`UseFullscreen=1`, (`UseFullscreen=0`) (all emulators except vsid).

`-CRTChwscale`
`+CRTChwscale`
: Enable/Disable hardware scaling (`CrtcHwScale=1`, `CrtcHwScale=0`) (xcbm2 and xpet only).

`-CRTCfulldevice <device>`
: Select fullscreen device (`CrtcFullscreenDevice`) (xcbm2 and xpet only).

`-TEDhwscale`
`+TEDhwscale`
: Enable/Disable hardware scaling (`TEDHwScale=1`, `TEDHwScale=0`) (xplus4 only).

`-TEDfulldevice <device>`
: Select fullscreen device (`TEDFullscreenDevice`) (xplus4 only).

`-VDChwscale`
`+VDChwscale`
: Enable/Disable hardware scaling (`VDCHwScale=1`, `VDCHwScale=1`) (x128 only).

`-VIChwscale`
`+VIChwscale`
: Enable/Disable hardware scaling (`VICHwScale=1`, `VICHwScale=0`) (xvic only).

`-VICfulldevice <device>`
: Select fullscreen device (`VICFullscreenDevice`) (xvic only).

`-VICIIhwscale`
`+VICIIhwscale`
: Enable/Disable hardware scaling (`VICIIHwScale=1`, `VICIIHwScale=0`) (all emulators except xcbm2, xpet, xplus4, xvic and vsid).

`-VICIIfulldevice <device>`
: Select fullscreen device (`VICIIFullscreenDevice`) (all emulators except xcbm2, xpet, xplus4, xvic and vsid).

`-joydev1 <0-3> / <0-9> / <0-11> / <0-13>`
: Set the device for joystick emulation of port 1 (`JoyDevice1`) (all emulators except xcbm2, xpet and vsid). (0: None, 1: Numpad, 2: Keyset 1, 3: Keyset 2, 4: Analog joystick 1, 5: Analog joystick 2, 6: Analog joystick 3, 7: Analog joystick 4, 8: Analog joystick 5, 9: Analog joystick 6, 10: Digital joystick 1, 11: Digital joystick 2, 12: USB joystick 1, 13: USB joystick 2)

`-joydev2 <0-3> / <0-9> / <0-11> / <0-13>`
: Set the device for joystick emulation of port 2 (`JoyDevice2`) (all emulators except xcbm2, xpet, xvic and vsid). (0: None, 1: Numpad, 2: Keyset 1, 3: Keyset 2, 4: Analog joystick 1, 5: Analog joystick 2, 6: Analog joystick 3, 7: Analog joystick 4, 8: Analog joystick 5, 9: Analog joystick 6, 10: Digital joystick 1, 11: Digital joystick 2, 12: USB joystick 1, 13: USB joystick 2)

`-extrajoydev1 <0-3> / <0-9> / <0-11> / <0-13>`
: Set device for extra joystick port 1 (`JoyDevice3`) (all emulators except xcbm5x0 and vsid). (0: None, 1: Numpad, 2: Keyset 1, 3: Keyset 2, 4: Analog joystick 1, 5: Analog joystick 2, 6: Analog joystick 3, 7: Analog joystick 4, 8: Analog joystick 5, 9: Analog joystick 6, 10: Digital joystick 1, 11: Digital joystick 2, 12: USB joystick 1, 13: USB joystick 2)

`-extrajoydev2 <0-3> / <0-9> / <0-11> / <0-13>`
: Set device for extra joystick port 2 (`JoyDevice4`) (all emulators except xcbm5x0, xplus4 and vsid). (0: None, 1: Numpad, 2: Keyset 1, 3: Keyset 2, 4: Analog joystick 1, 5: Analog joystick 2, 6: Analog joystick 3, 7: Analog joystick 4, 8: Analog joystick 5, 9: Analog joystick 6, 10: Digital joystick 1, 11: Digital joystick 2, 12: USB joystick 1, 13: USB joystick 2)

The available joysticks might differ depending on operating system and joystick support in the OS.

`Devices 4..9`
: Only available if joystick support is available at compile time.

`Devices 10 and 11`
: Only available if digital joystick support is available at compile time.

`Devices 12 and 13`
: Only available if USB joystick support is available at compile time.

The following command-line options are only available if MIDI support is available at compile time.

`-midiin <name>`
: Specify MIDI-In device (`MIDIInDev`) (x64, x64sc, xscpu64, x128 and xvic only).

`-midiout <name>`
: Specify MIDI-Out device (`MIDIOutDev`) (x64, x64sc, xscpu64, x128 and xvic only).

The following command-line option is only available if MIDI support and ALSA support is available at compile time.

`-mididrv <driver>`
: Specify MIDI driver (`MIDIDriver`) (x64, x64sc, xscpu64, x128 and xvic only). (0: OSS, 1: ALSA)

9 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.

Important note: In the past the idea was that snapshots with the same major version can be exchanged with later versions of the emulator. However, this proved to be too difficult to maintain, and insanely hard to test for correctness, which is why now the emulator will reject the snapshot when it finds a module that is too old. **Because of this you should not use snapshots for permanent/long term storage.**

9.1 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.

9.2 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.

9.2.1 Emulator modules

This section lists the modules that are contained in each of the emulators snapshot files.

9.2.1.1 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 9.2.1.6 [Drive modules], page 179,

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.

9.2.1.2 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 9.2.1.6 [Drive modules], page 179,

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.

9.2.1.3 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 9.2.1.6 [Drive modules], page 179,

9.2.1.4 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.
DWWPIA	6520	The PIA for the DWW hires board.
CPU6809	6809	The extra CPU in the SuperPET. This module is optional.
*	Drive modules	The emulated drive(s) have their own modules see Section 9.2.1.6 [Drive modules], page 179,

9.2.1.5 xcbm2 and xcbm5x0 modules

The modules in the xcbm2 and xcbm5x0 emulators 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 9.2.1.6 [Drive modules], page 179,

9.2.1.6 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
*	*	*

9.2.2 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, and sometimes additional state of the emulator. In the past we tried to make the format as implementation-independent as possible, to allow reuse of snapshots in later versions of the emulator, that however proved to be very impractical and insanely hard to maintain.

9.2.2.1 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.

9.2.2.2 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.
13 BYTE	VERSION MAGIC	"VICE Version\032", padded with 0
4 BYTE	VERSION	Release version (major, minor, micro). Byte 4 is always zero.
DWORD	SVNVERSION	SVN revision (or 0 if not available)

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. In the past the idea was that modules with the same VMAJOR should be able to be exchanged. This however proved to be too difficult to maintain, and insanely hard to test for correctness, which is why now the emulator will reject the snapshot when it finds a module that is too old. **Because of this you should not use snapshots for permanent/long term storage.**

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

9.2.2.3 CPU 6502 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.

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	?	?

9.2.2.4 CPU 6809 module

This module saves the core 6809 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.

Type	Name	Description
DWORD	CLK	the current CPU clock value. All other clock values are relative to this.
WORD	X	The X register
WORD	Y	The Y register
WORD	U	The U register
WORD	PC	The Program Counter register
BYTE	DP	The Direct Page register
BYTE	CC	The Condition Code register
BYTE	A	The A register
BYTE	B	The B register
		The following are for 6309 compatibility:
WORD	V	The V register
BYTE	E	The E register
BYTE	F	The F register

9.2.2.5 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`.

9.2.2.6 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)

9.2.2.7 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

9.2.2.8 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

9.2.2.9 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.

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

9.2.2.10 SID module

9.2.2.11 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

9.2.2.12 VIC-I module

9.2.2.13 VIC-II module

9.2.2.14 CRTC module

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 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

9.2.2.15 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.

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	64KiB RAM dump

The C64ROM module

Type	Name	Description
ARRAY	KERNAL	8KiB dump of the kernal ROM
ARRAY	BASIC	8KiB dump of the basic ROM
ARRAY	CHARGEN	4KiB dump of the chargen ROM

9.2.2.16 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	128KiB RAM dump banks 0 and 1

The C128ROM module

Type	Name	Description
ARRAY	KERNAL	8KiB dump of the kernal ROM
ARRAY	BASIC	32KiB dump of the basic ROM
ARRAY	EDITOR	4KiB dump of the editor ROM
ARRAY	4KiB CHARGEN	dump of the chargen ROM

9.2.2.17 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	1KiB RAM dump $0000-$03ff
ARRAY	RAM1	4KiB RAM dump $1000-$1fff
ARRAY	COLORRAM	2KiB Color RAM, $9400-$9bff
ARRAY	BLK0	if CONFIG & 1 then: 3KiB RAM dump $0400-$0fff

ARRAY	BLK1	if CONFIG & 2 then: 8KiB RAM dump $2000-$3fff
ARRAY	BLK2	if CONFIG & 4 then: 8KiB RAM dump $4000-$5fff
ARRAY	BLK3	if CONFIG & 8 then: 8KiB RAM dump $6000-$7fff
ARRAY	BLK5	if CONFIG & 32 then: 8KiB 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	8KiB KERNAL ROM image $e000-$ffff
ARRAY	BASIC	16KiB BASIC ROM image $c000-$dfff
ARRAY	CHARGEN	4KiB CHARGEN ROM image
ARRAY	BLK1A	4KiB ROM image $2*** (if CONFIG & 1)
ARRAY	BLK1B	4KiB ROM image $3*** (if CONFIG & 2)
ARRAY	BLK3A	4KiB ROM image $6*** (if CONFIG & 16)
ARRAY	BLK3B	4KiB ROM image $7*** (if CONFIG & 32)
ARRAY	BLK5A	4KiB ROM image $A*** (if CONFIG & 64)
ARRAY	BLK5B	4KiB ROM image $B*** (if CONFIG & 128)
ARRAY	BLK2A	4KiB ROM image $4*** (if CONFIG & 4; added in V1.1)
ARRAY	BLK2B	4KiB ROM image $5*** (if CONFIG & 8; added in V1.1)

9.2.2.18 PET memory module

The PET memory module actually consists of three 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 "PETDWW" module is also optional and contains the image of the hires expansion board (if enabled).

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 3

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 32KiB 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	4-32KiB RAM (not 8296, size depends on MEMSIZE)
ARRAY	VRAM	2/4KiB RAM (not 8296, size depends on CONFIG)
ARRAY	EXTRAM	64KiB expansion RAM (SuperPET and 8096 only)
ARRAY	RAM	128KiB 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 V1.2
BYTE	EOIBLANK	bit 0=0 = EOI does not blank screen, bit 0=1 = EOI blanks screen.
–	–	The following items have been added in V1.3
WORD	CPU_SWITCH	6502 / 6809 / PROG
BYTE	VAL, PREVODD, WANTODD	6702 dongle state information
WORD[8]	SHIFT	
BYTE	SuperPET config 2	Extra bits due to the Super-OS-9 MMU. Bit 5: FIRQ disabled. Bit 6: expansion memory in OS-9 flat mode.

The POSITIONAL 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.

In V1.3 the state for SuperPET has been added.

The PETROM module

Version numbers: Major 1, Minor 1

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. Bit 4: 1= SuperPET ROMs included.
ARRAY	KERNAL	4KiB KERNAL ROM image $f000-$ffff
ARRAY	EDITOR	2KiB EDITOR ROM image $e000-$e7ff
ARRAY	CHARGEN	2KiB CHARGEN ROM image
ARRAY	ROM9	4KiB $9*** ROM image (if CONFIG & 1)
ARRAY	ROMA	4KiB $A*** ROM image (if CONFIG & 2)
ARRAY	ROMB	4KiB $B*** ROM image (if CONFIG & 4)
ARRAY	ROMC	4KiB $C*** ROM image
ARRAY	ROMD	4KiB $D*** ROM image
ARRAY	ROME9	7 blocks $e900-$efff ROM image (if CONFIG & 8)
–	–	The following items have been added in V1.1
ROM6809	ROM6809	24KiB $A000-$FFFF ROM (if CONFIG & 16)
ARRAY	CHARGEN(2)	upper half of CHARGEN (if CONFIG & 16)

The PETDWW module

For storing the state of the DWW hires expansion board, there is a PETDWWPIA module, and a DWWMEM module.

The former has the same format as the PIA1.

Type	Name	Description
WORD	SIZE	The size of the memory dump that follows, or 0 if DWW disabled.
ARRAY	MEM	The memory in the DWW card, SIZE bytes.

9.2.2.19 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 128KiB blocks (1=128KiB, 2=256KiB, 4=512KiB, 8=1024KiB)

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	2KiB system RAM $f0000-$f07ff
ARRAY	VIDEO	2KiB video RAM $fd000-$fd7ff
ARRAY	RAM	RAM dump, size according to MEMSIZE
ARRAY	RAM08	if memsize < 1MiB and CONFIG & 1 : 2KiB RAM $f0800-$f0fff
ARRAY	RAM1	if memsize < 1MiB and CONFIG & 2 : 4KiB RAM $f1000-$f1fff
ARRAY	RAM2	if memsize < 1MiB and CONFIG & 4 : 8KiB RAM $f2000-$f3fff
ARRAY	RAM4	if memsize < 1MiB and CONFIG & 8 : 8KiB RAM $f4000-$f5fff
ARRAY	RAM6	if memsize < 1MiB and CONFIG & 16 : 8KiB RAM $f6000-$f7fff
ARRAY	RAMC	if memsize < 1MiB and CONFIG & 32 : 4KiB RAM $fc000-$fcfff

The RAM* arrays are only saved if the RAM itself is less than 1MiB. If the memory size is 1MiB 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 512KiB, or at $00000 if 512KiB 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	4KiB CHARGEN ROM image
ARRAY	ROM1	4KiB cartridge ROM image for $1*** (if CONFIG & 2)
ARRAY	ROM2	8KiB cartridge ROM image for $2000-$3fff (if CONFIG & 4)
ARRAY	ROM4	8KiB cartridge ROM image for $4000-$5fff (if CONFIG & 8)

ARRAY	ROM6	8KiB cartridge ROM image for $6000-$7fff (if CONFIG & 16)

9.2.2.20 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

10 Media images

10.1 Media images resources

`OCPOversizeHandling`
: Integer specifying the way the oversized input should be handled (all emulators except vsid). (0: scale down, 1: crop left top, 2: crop center top, 3: crop right top, 4: crop left center, 5: crop center, 6: crop right center, 7: crop left bottom, 8: crop center bottom, 9: crop right bottom)

`OCPUndersizeHandling`
: Integer specifying the way the undersized input should be handled (all emulators except vsid). (0: scale, 1: borderize)

`OCPMultiColorHandling`
: Integer specifying the way the multicolor to hires should be handled (all emulators except vsid). (0: b&w, 1: 2 colors, 2: 4 colors, 3: gray scale, 4: best cell colors)

`OCPTEDLumHandling`
: Integer specifying the way the TED luminosity should be handled (all emulators except vsid). (0: ignore, 1: dither)

`KoalaOversizeHandling`
: Integer specifying the way the oversized input should be handled (all emulators except vsid). (0: scale down, 1: crop left top, 2: crop center top, 3: crop right top, 4: crop left center, 5: crop center, 6: crop right center, 7: crop left bottom, 8: crop center bottom, 9: crop right bottom)

`KoalaUndersizeHandling`
: Integer specifying the way the undersized input should be handled (all emulators except vsid). (0: scale, 1: borderize)

`KoalaTEDLumHandling`
: Integer specifying the way the TED luminosity should be handled (all emulators except vsid). (0: ignore, 1: dither)

`FFMPEGFormat`
: String specifying the current FFMPEG output driver.

`FFMPEGAudioBitrate`
: Integer specifying the current FFMPEG audio bitrate.

`FFMPEGVideoBitrate`
: Integer specifying the current FFMPEG video bitrate.

`FFMPEGAudioCodec`
: Integer specifying the current FFMPEG audio codec.

`FFMPEGVideoCodec`
: Integer specifying the current FFMPEG video codec.

`FFMPEGVideoHalveFramerate`
: Boolean, if true record only every other frame.

10.2 Media images command-line options

`-ocpoversize <method>`
: Select the way the oversized input should be handled (`OCPOversizeHandling`) (all emulators except vsid). (0: scale down, 1: crop left top, 2: crop center top, 3: crop right top, 4: crop left center, 5: crop center, 6: crop right center, 7: crop left bottom, 8: crop center bottom, 9: crop right bottom)

`-ocpundersize <method>`
: Select the way the undersized input should be handled (`OCPUndersizeHandling`) (all emulators except vsid). (0: scale, 1: borderize)

`-ocpmc <method>`
: Select the way the multicolor to hires should be handled (`OCPMultiColorHandling`) (all emulators except vsid). (0: b&w, 1: 2 colors, 2: 4 colors, 3: gray scale, 4: best cell colors)

`-ocptedlum <method>`
: Select the way the TED luminosity should be handled (`OCPTEDLumHandling`) (all emulators except vsid). (0: ignore, 1: dither)

`-koalaoversize <method>`
: Select the way the oversized input should be handled (`KoalaOversizeHandling`) (all emulators except vsid). (0: scale down, 1: crop left top, 2: crop center top, 3: crop right top, 4: crop left center, 5: crop center, 6: crop right center, 7: crop left bottom, 8: crop center bottom, 9: crop right bottom)

`-koalaundersize <method>`
: Select the way the undersized input should be handled (`KoalaUndersizeHandling`) (all emulators except vsid). (0: scale, 1: borderize)

`-koalatedlum <method>`
: Select the way the TED luminosity should be handled (`KoalaTEDLumHandling`) (all emulators except vsid). (0: ignore, 1: dither)

`-ffmpegaudiobitrate <value>`
: Set bitrate for audio stream in media file

`-ffmpegvideobitrate <value>`
: Set bitrate for video stream in media file

11 Event history

VICE supports recording an arbitrary session on the emulated machine and playing back this session later. This is done by saving a snapshot at the beginning of the recording session and then remembering all the user interaction such as keyboard and joystick input. We call this an 'event history'. The main purpose for having an event history is to create game sessions, e.g. recording how to solve a game. An example walkthrough for the well known game "Fort Apocalypse" is available.

11.1 Recommended Settings

When using the event history feature it is possible that the playback session differs from what was done at recording time. This might arise due to a problem in the initial snapshot or settings. Here are some suggestions to minimize the chance of failures in the session: a. Reset to default settings. b. Choose refresh rate 1/1. c. Choose joystick and Video/Doublesize settings as desired or needed. d. SID engine must be reSID (which is the default). e. Choose Drive settings/Idle method: None Do not change any settings during recording or playback!

11.2 Recorded Events

The following is a list of the user interaction that will be recorded: - Joystick movement and button - Keyboard - Resetting the machine (hard and soft) - Attaching or detaching disk/tape images (see 8. Limitations) - Datasette controls

11.3 Recording an Event History

Recording an event history will create one or two files for a snapshot and the list of the user events. First, create an empty directory in which these files are to be saved and then select this directory and the name of the snapshot files via 'Snapshot//Select History files/directory'. Next, attach the disk or tape image with the game you want to record and load and run the game.

Start recording via 'Snapshot//Start/Stop Recording History'. Play the game. All actions will be recorded. After the game is finished, stop recording via 'Snapshot//Start/Stop Recording History'. The selected directory should now contain the two snapshot files (default ist start.vsf and end.vsf).

11.4 Setting and Returning to Milestones

It is difficult to finish a game in one sitting and noone wants to record all their mistakes and lost lives. Use the milestone feature in a recording session in the following way:

Set a milestone when you have finished a level or completed a task ('Snapshot//Set recording milestone' or ALT-E). This will save the event history and a snapshot of the machine to the file end.vsf but recording will continue.

Return to the last milestone when you have made a mistake or lost a life ('Snapshot//Return to milestone' or ALT-U). This will reset the game and the event history to the last milestone snapshot so that you can try again.

11.5 Continuing an Event History

If you have stopped a recording session and want to continue it later, you should create a backup of your start.vsf and end.vsf files first to avoid overwriting them by accident.

Then change the event history start mode: 'Snapshot//Recording start mode//Load existing snapshot'. When you start recording now, you will continue where the session was stopped last time.

Another way of continuing an existing history is to start somewhere inside the history (e.g. you have recorded 10 minutes of a game and later recognize that you made a mistake after 6 minutes that makes it impossible to finish the game). For this you have to select the start mode 'Overwrite playback'. Now you can start the playback with 'Snapshot//Start/Stop Playback History' and when you reach the point where you want to change the history you can directly switch to recording via 'Snapshot//Start/Stop Recording History'.

11.6 Playing Back an Event History

To play back an event history, select the directory with the history files start.vsf and end.vsf via 'Snapshot//Select History directory' and start the playback with 'Snapshot//Start/Stop Playback History'. Enjoy! During playback any user interaction is disabled. The playback stops when the end of the session is reached or if 'Snapshot//Select History directory' is selected again.

11.7 Limitations and Suggestions

a. Snapshot files will be quite big (>1MiB) if a disk image has been attached. If possible, use PRG or T64 images to reduce the size of snapshot files. b. Snapshots may not be 100% accurate even with all the recommended settings.

11.8 Event history resources

`EventSnapshotDir`
: String specifying the directory used for loading and saving snapshots (all emulators except vsid).

`EventStartSnapshot`
: String specifying the filename for the start snapshot (all emulators except vsid).

`EventEndSnapshot`
: String specifying the filename for the end snapshot (all emulators except vsid).

`EventStartMode`
: Integer specifying how to start event recording (all emulators except vsid). (0: save new snapshot, 1: load existing snapshot, 2: reset, 3: playback)

`EventImageInclude`
: Boolean specifying whether to include ROM and Disk images in the snapshots (all emulators except vsid).

11.9 Event history command-line options

`-playback`
: Playback recorded events (all emulators except vsid).

`-eventsnapshotdir <Name>`
: Set event snapshot directory (`EventSnapshotDir`) (all emulators except vsid).

`-eventstartsnapshot <Name>`
: Set event start snapshot filename (`EventStartSnapshot`) (all emulators except vsid).

`-eventendsnapshot <Name>`
: Set event end snapshot filename (`EventEndSnapshot`) (all emulators except vsid).

`-eventstartmode <Mode>`
: Set event start mode (`EventStartMode`) (all emulators except vsid). (0: save new snapshot, 1: load existing snapshot, 2: reset, 3: playback)

`-eventimageinc`
`+eventimageinc`
: Enable/disable the inclusion of disk images in the event (`EventImageInclude=1`, `EventImageInclude=0`) (all emulators except vsid).

12 Monitor

The 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 "Activate monitor" from the menu. The monitor will either run in a seperate window, or in a terminal emulation program (such as `rxvt` or `xterm`) when "native monitor" has been enabled in the settings.

12.1 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 64KiB 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.

'`label`' `label` is the name of a label. It must start with a dot (`"."`) in order for the monitor to recognize it as a label. Register names preceeded by a dot (for example `.PC`) are special labels that evaluate to value of the respective register at the time it is used, and thus can not be used as a regular label.

'`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-cpuhistory" 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.

12.2 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-cpuhistory 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 (use FL for status flags). 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.

`stopwatch [reset]`
: Print the CPU cycle counter of the current device. 'reset' sets the counter to 0.

`undump "<filename>"`
: Read a snapshot of the machine from the file specified.

12.3 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-cpuhistory to enable)

`memmapzap`
`mmzap` Clear the memmap. (disabled by default; configure with –enable-cpuhistory 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-cpuhistory 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`.

12.4 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.

12.5 Checkpoint commands

`break [load|store|exec] [address [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. The "load|store|exec" parameter can be either "load", "store" or "exec" (or any combination of these) to determine on which operation the monitor breaks. If not specified, the monitor breaks on "exec". 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 ignored. 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.

The condition can use registers (A, X, Y, PC, SP, FL and other cpu specific registers (see manual)) and compare them (==, !=, <, >, <=, >=) against other registers or constants. RL can be used to refer to the current rasterline, and CY refers to the current cycle in the line.

Full expressions are also supported (+, -, *, /, &&, ||). This let's you f.e. to check specific bits in the FL register using the bitwise boolean operators. Paranthises are also supported in the expression. Registers can be the registers of other devices; this is denoted by a memspace prefix (i.e., c:, 8:, 9:, 10:, 11:

Examples: A == $0, X == Y, 8:X == X) You can also compare against the value of a memory location in a specific bank, i.e you can break only if the vic register $d020 is $f0. use the form @[bankname]:[$<address>] | [.label]. Note this is for the C : memspace only. Examples : if @io:$d020 == $f0, if @io:.vicBorder == $f0

`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 [load|store|exec] [address [address] [if <cond_expr>]]`
`tr [load|store|exec] [address [address] [if <cond_expr>]]`
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. If no addresses are given, a list of all the checkpoints is printed. The "load|store|exec" parameter can be either "load", "store" or "exec" (or any combination of these) to determine which operation the monitor traces. If not specified, the monitor traces all operations. A conditional expression can also be specified for the tracepoint. For more information on conditions, see the CONDITION command.

`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 [load|store|exec] [address [address] [if <cond_expr>]]`
`w [load|store|exec] [address [address] [if <cond_expr>]]`
This command is similar to the `break` command except that it operates on watchpoints. A watchpoint differs from a breakpoint 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 checkpoints is printed. The "load|store|exec" parameter can be either "load", "store" or "exec" (or any combination of these) to determine on which operation the monitor breaks. If not specified, the monitor breaks on "load" and "store" operations. A conditional expression can also be specified for the watchpoint. For more information on conditions, see the CONDITION command.

`dummy [on|off|toggle]`
Control whether the checkpoints will trigger on dummy accesses. If the argument is 'on' then dummy accesses will cause checkpoints to trigger. If the argument is 'off' then dummy accesses will not trigger any checkpoints. If the

argument is 'toggle' then the current mode is switched. No argument displays the current state.

12.6 General commands

`cd <directory>`
Change the working directory.

`device [c:|8:|9:]`
`dev [c:|8:|9:]`
Set the default address space to either the computer (`c:`) or the disk (`8:|9:`).

`dir [<directory>]`
`ls [<directory>]`
Display the directory contents.

`pwd` Show current working directory.

`mkdir <directory>`
Create directory.

`rmdir <directory>`
Remove 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.

`log [on|off|toggle]`
Control whether the monitor output is logged into a logfile. If the argument is 'on' then all output will be written into the logfile. If the argument is 'off' then no log is produced. If the argument is 'toggle' then the current mode is switched. No argument displays the current state.

`logname "<filename>"`
Sets the filename of the logfile.

12.7 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.

`list [<directory>]`
: List disk contents.

`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.

`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.

`verify "<filename>" <device> [<address>]`
`v "<filename>" <device> [<address>]`
: Compare the specified file with memory. If no address is given, the address is specified by the first two bytes read from the file. If address is given, the specified address is used and the first two bytes read from the file are skipped. If device is 0, the file is read from the file system.

`bverify "<filename>" <device> <address>`
`bv "<filename>" <device> <address>`
: Compare the specified file with memory at the specified address. If device is 0, the file is read from the file system.

12.8 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`.

12.9 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 [<address_space>] <label>`
`dl [<address_space>] <label>`
: Remove the specified label from the label tables. If no address space is checked, all tables are checked.

`load_labels [<address_space>] "<filename>"`
`ll [<address_space>] "<filename>"`
: Load a file containing a mapping of labels to addresses. If no address 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 [<address_space>] "<filename>"`
`sl [<address_space>] "<filename>"`
: Save labels to a file. If no address space is specified, all of the labels are saved.

`show_labels [<address_space>]`
`shl [<address_space>]`
: Display current label mappings. If no address space is specified, show all labels from default address space.

`clear_labels [<address_space>]`
`cl [<address_space>]`
: Clear current label mappings. If no address space is specified, clear all labels from default address space.

12.10 Miscellaneous commands

`; <text>` Add a comment to a file to be read by the monitor. For example:

```
; Set border and background to black
> d020 00 00
```

`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. Note that you can specify specific keycodes by using C-style escaped hexcodes (`"\x0a"`).

`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`.

`load_resources "<file>"`
`resload "<file>"`
: Load resources from `file`.

`save_resources "<file>"`
`ressave "<file>"`
: Save resources to `file`.

`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`
`q` Exit the emulator immediately.

`~ <number>`
Display the specified number in decimal, hex, octal and binary.

13 Binary monitor

The binary remote monitor commands are sent over a dedicated connection, specified with the command line options `-binarymonitor` & `-binarymonitoraddress`. See Section 6.15 [Monitor settings], page 70. The remote monitor detects a binary command because it starts with ASCII STX (0x02). Note that there is no termination character. The command length acts as synchronisation point.

All multibyte values are in little endian order unless otherwise specified.

13.1 Command Structure

byte 0: 0x02 (STX)

byte 1: API version ID (currently 0x01)

The API version identifies incompatible changes, such as modifying the header structure, or rearranging or changing the meaning of existing response fields. The API version does not need to be incremented for additional fields. If all the variable length fields are prefixed with their lengths then you should be able to add new ones to any response. The server can assume default values for older clients, and for newer clients with longer commands it should be able to ignore the extra fields safely.

byte 2-5: length

Note that the command length does *not* count the STX, the command length, the command byte, or the request ID. Basically nothing in the header, just the body.

byte 6-9: request id

In little endian order. All multibyte values are in little endian order, unless otherwise specified. There is no requirement for this to be unique, but it makes it easier to match up the responses if you do.

byte 10: The numeric command type

See Section 13.4 [Binary Commands], page 213.

byte 11+: The command body.

See Section 13.4 [Binary Commands], page 213.

13.2 Response Structure

byte 0: 0x02 (STX)

byte 1: API version ID (currently 0x01)

The API version identifies incompatible changes, such as modifying the header structure, or rearranging or changing the meaning of existing response fields. The API version does not need to be incremented for additional fields. If all the variable length fields are prefixed with their lengths then you should be able to add new ones to any response. The client can assume default values for older versions of VICE, and for newer versions of VICE with longer responses it should be able to ignore the extra fields safely.

byte 2-5: response body length. Does not include any header fields
byte 6: response type
This is usually the same as the command ID

byte 7: error code

`0x00`	OK, everything worked
`0x01`	The object you are trying to get or set doesn't exist.
`0x02`	The memspace is invalid
`0x80`	Command length is not correct for this command
`0x81`	An invalid parameter value was present
`0x82`	The API version is not understood by the server
`0x83`	The command type is not understood by the server
`0x8f`	The command had parameter values that passed basic checks, but a general failure occurred See Section 13.4 [Binary Commands], page 213, for other error codes

byte 8-11: request ID
This is the request ID given to initiate this response. If the value is 0xffffffff, Then the response was initiated by an event, such as hitting a checkpoint.

byte 12+: response body.
See Section 13.4 [Binary Commands], page 213.

13.3 Example Exchange

1. Client connects to ip4://127.0.0.1:6502
2. Client sends a command to set a temporary checkpoint:

```
02 | 01 | 08 00 00 00 | ad de 34 12 | 12 | e2 fc | e3 fc | 01 | 01 | 04 | 01
```

`0x02` Begin command

`0x01` API version 1

`0x00000008`
The command excluding the header is 8 bytes long.

`0x1234dead`
The request ID is 0x1234dead. The response will contain this ID.

`0x12` See Section 13.4.4 [MON_CMD_CHECKPOINT_SET], page 214.

`0xfce2` The address range of the checkpoint starts at 0xfce2.

`0xfce3` The address range of the checkpoint ends at 0xfce3.

`0x01` The checkpoint will cause the emulator to stop.

`0x01` The checkpoint is enabled.

`0x04` The checkpoint will trigger on exec from 0xfce2 - 0xfce3.

`0x01` The checkpoint is temporary.

3. The transmission of any command causes the emulator to stop, similar to the regular monitor. This causes the server to respond with a list of register values.

```
02 | 01 | 26 00 00 00 | 31 | 00 | ff ff ff ff | 09 00 [ 03 { 03 | cf e5 } 03 { 00
| 00 00 } ... ]
```

`0x02` Begin response

`0x01` API Version 1

`0x00000026`
Response length is 38

`0x31` See Section 13.5.2 [MON_RESPONSE_REGISTER_INFO], page 223.

`0x00` No error occurred

`0xffffffff`
This response was not directly triggered by a command from the client.

`0x0009` The register array is 9 items long

PC:

`0x03` The register array item is 3 bytes long

`0x03` The register is the PC (ID 3) Note: you should find the names to these IDs using the MON_CMD_REGISTERS_AVAILABLE command. See Section 13.4.20 [MON_CMD_REGISTERS_AVAILABLE], page 219. Do not rely on them being consistent.

`0xe5cf` The register value is 0xe5cf

A:

`0x03` The register array item is 3 bytes long

`0x00` The register is A (ID 0) Note: you should find the names to these IDs using the MON_CMD_REGISTERS_AVAILABLE command. See Section 13.4.20 [MON_CMD_REGISTERS_AVAILABLE], page 219. Do not rely on them being consistent.

`0x0000` The register value is 0x0000

4. After the register information, the server sends a stopped event to indicate that the emulator is stopped.

```
02 | 01 | 02 00 00 00 | 62 | 00 | ff ff ff ff | cf e5
```

`0x02` Begin response

`0x01` API Version 1

`0x00000002`
Response is two bytes long.

`0x62` Response type is 0x62, MON_RESPONSE_STOPPED.

`0xffffffff`
This response was not directly triggered by a command from the client.

`0xe5cf` The current program counter

5. The server processes the checkpoint set command, and sends a response to the client.

```
... | 11 | ... | 02 00 00 00 | 00 | e2 fc | e3 fc | 01 | 01 | 04 | 01 | 00 00 00
00 | 00 00 00 00 | 00
```
(Some response header fields are omitted (...) for brevity.)

`0x11` See Section 13.5.1 [MON_RESPONSE_CHECKPOINT_INFO], page 222.

`0x00000002`
Checkpoint number is 2

`0x00` Checkpoint was not hit (as it was just created)

`0xfce2` Checkpoint start address

`0xfce3` Checkpoint end address

`0x01` The checkpoint will cause the emulator to stop.

`0x01` The checkpoint is enabled.

`0x04` The checkpoint will trigger on exec from 0xfce2 - 0xfce3.

`0x01` The checkpoint is temporary.

`0x00000000`
The checkpoint has been hit zero times.

`0x00000000`
The checkpoint has been ignored zero times.

6. Client sends a command to continue:

```
... | aa
```
(Some command header fields are omitted (...) for brevity.)

`0xaa` See Section 13.4.22 [MON_CMD_EXIT], page 221.

7. Server acknowledges the command:

   ```
   ... | aa | ...
   ```
   (Some response header fields are omitted (...) for brevity.)

   `0xaa` See Section 13.4.22 [MON_CMD_EXIT], page 221.
8. Server resumes execution and sends a resume event:

   ```
   ... | 63 | ... | cf e5
   ```
   (Some response header fields are omitted (...) for brevity.)

   `0x63` See Section 13.5.5 [MON_RESPONSE_RESUMED], page 223.

   `0xe5cf` Program counter is currently at 0xe5cf
9. Some time later, the server hits the breakpoint. This causes it to emit a checkpoint response. This is identical to the previous checkpoint response, except that it is marked as "hit" and the hit and ignore counts are updated.
10. The server emits the register information and the stopped event when reentering the monitor, as seen previously.

13.4 Commands

These are the possible command types and responses, without the header portions mentioned above.

13.4.1 Memory get (0x01)

Reads a chunk of memory from a start address to an end address (inclusive).

Command body:

byte 0: side effects?
: Should the read cause side effects?

byte 1-2: start address
byte 3-4: end address
byte 5: memspace
: Describes which part of the computer you want to read:
  - 0x00: main memory
  - 0x01: drive 8
  - 0x02: drive 9
  - 0x03: drive 10
  - 0x04: drive 11

byte 6-7: bank ID
: Describes which bank you want. This is dependent on your machine. See Section 13.4.19 [MON_CMD_BANKS_AVAILABLE], page 219. If the memspace selected doesn't support banks, this value is ignored.

Response type:

0x01: MON_RESPONSE_MEM_GET

Response body:

byte 0-1: The length of the memory segment.
byte 2+: The memory at the address.

13.4.2 Memory set (0x02)

Writes a chunk of memory from a start address to an end address (inclusive).

Command body:

byte 0: side effects?
Should the write cause side effects?

byte 1-2: start address
byte 3-4: end address
byte 5: memspace
Describes which part of the computer you want to write:

- 0x00: main memory
- 0x01: drive 8
- 0x02: drive 9
- 0x03: drive 10
- 0x04: drive 11

byte 6-7: bank ID
Describes which bank you want. This is dependent on your machine. See Section 13.4.19 [MON_CMD_BANKS_AVAILABLE], page 219. If the memspace selected doesn't support banks, this byte is ignored.

byte 8+: Memory contents to write

Response type:

0x02: MON_RESPONSE_MEM_SET

Response body:

Currently empty.

13.4.3 Checkpoint get (0x11)

Gets any type of checkpoint. (break, watch, trace)

Command body:

byte 0-3: checkpoint number

See Section 13.5.1 [MON_RESPONSE_CHECKPOINT_INFO], page 222.

13.4.4 Checkpoint set (0x12)

Sets any type of checkpoint. This combines the functionality of several textual commands (break, watch, trace) into one, as they are all the same with only minor variations. To set conditions, see Section 13.4.8 [MON_CMD_CONDITION_SET], page 216, after executing this one.

Command body:

byte 0-1: start address
byte 2-3: end address
byte 4: stop when hit
0x01: true, 0x00: false

byte 5: enabled
0x01: true, 0x00: false

byte 6: CPU operation
0x01: load, 0x02: store, 0x04: exec

byte 7: temporary
Deletes the checkpoint after it has been hit once. This is similar to "until" command, but it will not resume the emulator.

See Section 13.5.1 [MON_RESPONSE_CHECKPOINT_INFO], page 222.

13.4.5 Checkpoint delete (0x13)

Deletes any type of checkpoint. (break, watch, trace)

Command body:

byte 0-3: checkpoint number

Response type:

0x13: MON_RESPONSE_CHECKPOINT_DELETE

Response body:

Currently empty.

13.4.6 Checkpoint list (0x14)

Command body:

Currently empty.

Response type:

Emits a series of MON_RESPONSE_CHECKPOINT_INFO responses (see Section 13.5.1 [MON_RESPONSE_CHECKPOINT_INFO], page 222) followed by

0x14: MON_RESPONSE_CHECKPOINT_LIST

Response body:

byte 0-3: The total number of checkpoints

13.4.7 Checkpoint toggle (0x15)

Command body:

byte 0-3: Checkpoint number
byte 4: Enabled?
0x00: disabled, 0x01: enabled

Response type:

0x15: MON_RESPONSE_CHECKPOINT_TOGGLE

Response body:

Currently empty.

13.4.8 Condition set (0x22)

Sets a condition on an existing checkpoint. It is not currently possible to retrieve conditions after setting them.

Command body:

byte 0-3: checkpoint number
byte 4: condition expression length
byte 5+: condition expression string
This is the same format used on the command line. Not null terminated.

Response type:

0x22: MON_RESPONSE_CONDITION_SET

Response body:

Currently empty.

13.4.9 Registers get (0x31)

Get details about the registers

Command body:

byte 0: memspace
Describes which part of the computer you want to read:

- 0x00: main memory
- 0x01: drive 8
- 0x02: drive 9
- 0x03: drive 10
- 0x04: drive 11

See Section 13.5.2 [MON_RESPONSE_REGISTER_INFO], page 223.

13.4.10 Registers set (0x32)

Set the register values

Command body:

byte 0: memspace
Describes which part of the computer you want to write:

- 0x00: main memory
- 0x01: drive 8
- 0x02: drive 9
- 0x03: drive 10
- 0x04: drive 11

byte 1-2: The count of the array items
byte 2+: An array with items of structure:
byte 0: Size of the item, excluding this byte byte 1: ID of the register byte 2-3: register value

See Section 13.5.2 [MON_RESPONSE_REGISTER_INFO], page 223.

13.4.11 Dump (0x41)

Saves the machine state to a file.

Command body:

byte 0: Save ROMs to snapshot file?
0x01: true, 0x00: false

byte 1: Save disks to snapshot file?
0x01: true, 0x00: false

byte 2: Length of filename
byte 3+: Filename
The filename to save the snapshot to.

Response type:

0x41: MON_RESPONSE_DUMP

Response body:

Currently empty.

13.4.12 Undump (0x42)

Loads the machine state from a file.

Command body:

byte 0: Length of filename
byte 1+: Filename
The filename to load the snapshot from.

Response type:

0x42: MON_RESPONSE_UNDUMP

Response body:

byte 0-1: The current program counter position

13.4.13 Resource Get (0x51)

Get a resource value from the emulator. See Section “Settings and resources” in `Settings and resources`.

Command body:

byte 0: Length of resource name
byte 1+: Resource name

Response type:

0x51: MON_RESPONSE_RESOURCE_GET

Response body:

byte 0: Type of the resource
0x00: String, 0x01: Integer

byte 1: Length of the value
byte 2+: The value

13.4.14 Resource Set (0x52)

Set a resource value in the emulator. See Section "Settings and resources" in `Settings and resources`.

Command body:

1 byte: Type of the resource value
: 0x00: String, 0x01: Integer
Strings will be interpreted if the destination is an Integer.

1 byte: Resource name length = (&name)
(*name) bytes: The resource name
1 byte: Resource value length = (&value)
(*value) bytes: The resource value

Response type:

0x52: MON_RESPONSE_RESOURCE_SET

Response body:

Currently empty.

13.4.15 Advance Instructions (0x71)

Step over a certain number of instructions.

Command body:

byte 0: Step over subroutines?
: Should subroutines count as a single instruction?

byte 1-2: How many instructions to step over.

Response type:

0x71: MON_RESPONSE_ADVANCE_INSTRUCTIONS

Response body:

Currently empty.

13.4.16 Keyboard feed (0x72)

Add text to the keyboard buffer.

Command body:

byte 0: Length of text
byte 1+: The text
: Special characters such as return are escaped with backslashes.

Response type:

0x72: MON_RESPONSE_KEYBOARD_FEED

Response body:

Currently empty.

13.4.17 Execute until return (0x73)

Continues execution and returns to the monitor just after the next RTS or RTI is executed.

This command is the same as "return" in the text monitor.

Command body:

Currently empty.

Response type:

0x73: MON_RESPONSE_EXECUTE_UNTIL_RETURN

Response body:

Currently empty.

13.4.18 Ping (0x81)

Get an empty response

Command body:

Always empty

Response type:

0x81: MON_RESPONSE_PING

Response body:

Always empty

13.4.19 Banks available (0x82)

Gives a listing of all the bank IDs for the running machine with their names.

Command body:

Currently empty.

Response type:

0x82: MON_RESPONSE_BANKS_AVAILABLE

Response body:

byte 0-1: The count of the array items
byte 2+: An array with items of structure:

> **byte 0: Size of the item, excluding this byte**
> **byte 1-2: bank ID**
> **byte 3: Name length**
> **byte 4+: Name**

13.4.20 Registers available (0x83)

Gives a listing of all the registers for the running machine with their names.

Command body:

byte 0: memspace

> Describes which part of the computer you want to read:
>
> - 0x00: main memory
> - 0x01: drive 8

- 0x02: drive 9
- 0x03: drive 10
- 0x04: drive 11

Response type:

0x82: MON_RESPONSE_REGISTERS_AVAILABLE

Response body:

byte 0-1: The count of the array items
byte 2+: An array with items of structure:

byte 0: Size of the item, excluding this byte
byte 1: ID of the register
byte 2: Size of the register in bits
byte 3: Length of name
byte 4+: Name

13.4.21 Display Get (0x84)

Gets the current screen in a requested bit format.

Command body:

byte 0: USE VIC-II?

Must be included, but ignored for all but the C128. If true, (0x01) the screen returned will be from the VIC-II. If false (0x00), it will be from the VDC.

byte 1: Format

0x00: Indexed, 8 bit
0x01: RGB, 24 bit
0x02: BGR, 24 bit
0x03: RGBA, 32 bit
0x04: BGRA, 32 bit

Response type:

0x84: MON_RESPONSE_DISPLAY_GET

Response body:

The length-at-the-beginning format of this object is similar to other response types, except for being four bytes. Also the display buffer length is contained inside another object instead of before the buffer.

4 bytes: Length of the fields before the display buffer

4 bytes: Length of fields before reserved area

4 bytes: Length of display buffer = (&buffer)
2 bytes: Debug width of display buffer (uncropped)

The largest width the screen gets.

2 bytes: Debug height of display buffer (uncropped)

The largest height the screen gets.

2 bytes: X offset
X offset to the inner part of the screen.

2 bytes: Y offset
Y offset to the inner part of the screen.

2 bytes: Width of the inner part of the screen.
2 bytes: Height of the inner part of the screen.
1 byte: Bits per pixel of display buffer, 8, 24 or 32

4 bytes: Length of the reserved area = (&reserved)

(*reserved) bytes: Reserved TGA area

(*buffer) bytes: Display buffer data

13.4.22 Exit (0xaa)

Exit the monitor until the next breakpoint.

Command body:

Currently empty.

Response type:

0xaa: MON_RESPONSE_EXIT

Response body:

Currently empty.

13.4.23 Quit (0xbb)

Quits VICE.

Command body:

Currently empty.

Response type:

0xbb: MON_RESPONSE_QUIT

Response body:

Currently empty.

13.4.24 Reset (0xcc)

Reset the system or a drive

Command body:

byte 0: What to reset

- 0x00: Soft reset system
- 0x01: Hard reset system
- 0x08 - 0x0b: Reset drives 8 - 11

Response type:

0xcc: MON_RESPONSE_RESET

Response body:

Currently empty.

13.4.25 Autostart / autoload (0xdd)

Load a program then return to the monitor

Command body:

byte 0: Run after loading?
0x01: true, 0x00: false

byte 1-2: File index
The index of the file to execute, if a disk image. 0x00 is the default value.

byte 3: Length of filename
byte 4+: Filename
The filename to autoload.

Response type:

0xdd: MON_RESPONSE_AUTOSTART

Response body:

Currently empty.

13.5 Responses

These responses are generated by many different commands, or by certain events. Events are generated with a request ID of 0xffffffff, so that they can be easily distinguished from regular requests.

13.5.1 Checkpoint Response (0x11)

This response is generated by hitting a checkpoint, or by many of the checkpoint commands.

See Section 13.4.3 [MON_CMD_CHECKPOINT_GET], page 214.

See Section 13.4.4 [MON_CMD_CHECKPOINT_SET], page 214.

See Section 13.4.6 [MON_CMD_CHECKPOINT_LIST], page 215.

Response type:

0x11: MON_RESPONSE_CHECKPOINT_INFO

Response body:

byte 0-3: Checkpoint number
byte 4: Currently hit?
0x01: true, 0x00: false

byte 5-6: start address
byte 7-8: end address
byte 9: stop when hit
0x01: true, 0x00: false

byte 10: enabled
0x01: true, 0x00: false

byte 11: CPU operation
0x01: load, 0x02: store, 0x04: exec

byte 12: temporary

Deletes the checkpoint after it has been hit once. This is similar to "until" command, but it will not resume the emulator.

byte 13-16: hit count
byte 17-20: ignore count
byte 21: Has condition?

0x01: true, 0x00: false

13.5.2 Register Response (0x31)

Response type:

0x31: MON_RESPONSE_REGISTER_INFO

Response body:

byte 0-1: The count of the array items
byte 2+: An array with items of structure:

byte 0: Size of the item, excluding this byte
byte 1: ID of the register
byte 2-3: register value

13.5.3 JAM Response (0x61)

When the CPU jams

Response type:

0x61: MON_RESPONSE_JAM

Response body:

byte 0-1: The current program counter position

13.5.4 Stopped Response (0x62)

When the machine stops for the monitor, either due to hitting a checkpoint or stepping.

Response type:

0x62: MON_RESPONSE_STOPPED

Response body:

byte 0-1: The current program counter position

13.5.5 Resumed Response (0x63)

When the machine resumes execution for any reason.

Response type:

0x63: MON_RESPONSE_RESUMED

Response body:

byte 0-1: The current program counter position

14 c1541

VICE is provided with a complete stand-alone disk image maintenance utility, called `c1541`.

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.

14.1 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.

14.2 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).

14.3 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).

`bam [<unit>] | <track-min> <track-max> [<unit>]`
: Show the BAM of `unit`, optionally displaying only the entries for `track-min` to `track-max`

`bcopy <src-trk> <src-sec> <dst-trk> <dst-sec> [<src-unit> [<dst-unit>]]`
: Copy a block to another block, optionally specifying different source and destination units. The block is copied using all 256 bytes.

`bfill <track> <sector> <value> [<unit>]`
: Fill a block with a single value.

`block <track> <sector> [<offset> [<unit>]]`
: Show specified disk block in hex form.

`bpoke [@unit:<unit>] <track> <sector> <offset> <data ...>`
: Store one or more bytes of `data` into a block. The `data` can be specified using prefixes:

`0b or %`	binary value (‘%11111111‘)
`&`	octal value (‘&377‘)
`0x or $`	hexadecimal value (‘$ff‘)

  The `<unit>` is optional, and when used must use the CBM DOS notation for the unit number, for example ‘@9:‘.

`bread <filename> <track> <sector> [<unit>]`
: Read a block from a disk image and write it to the host filesystem as `filename`.

`bwrite <filename> <track> <sector> [<unit>]`
: Write data from the host filesystem using `filename` as input. At most 256 bytes are written to the disk image.

`chain <track> <sector> [<unit>]`
: Show block chain starting at (`track`, `sector`). The last number shown is the number of bytes used in the final block.

`copy <source1> [<source2> ... <sourceN>] <destination>`
: Copy `source1 ... sourceN` into destination. If N > 1, `destination` must be a simple drive specifier (`@n:`). To copy a REL file, specify the `<source>` with a `,L` file type suffix.

`delete <file1> [<file2> ... <fileN>]`
: Delete the specified files.

`exit`
: Exit (same as `quit`).

`entry [+side] <file1> [<file2> ... <fileN>]`
: Show the directory entries of the given files in full detail. If the `+side` option is present, and it is a RELative file, it also shows all side sectors.

`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), `g71` (VC1571 but in GCR coding), `d81` (VC1581), `d80` (CBM8050), `d82` (CBM8250/1001), or `d90` (CBM D9090). Otherwise, format the disk in the current unit, if any.

`geosread <source> [<destination>]`
: Read GEOS <source> from the disk image and copy it as a Convert file into <destination> in the file system. If <destination> is not specified, copy it into a file with the same name as <source>.

`geoswrite <source>`
: Write GOES Convert file <source> from the file system on a disk image.

`geosextract <source>`
: Extract all the files to the file system and GEOS Convert them.

`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>]`
`dir [<pattern>]`
: List files matching `pattern` (default is all files).

`name <diskname>[,<id>] <unit>`
: Change image name.

`p00save <enable> [<unit>]`
: Save P00 files to the file system.

`pwd`
: Print current working directory.

`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`. By default PRG files are copied. To copy SEQ files add `,s` (i.e. the usual DOS file specification syntax). REL files can be copied by adding `,l`. Don't specify the record length, `c1541` will determine it automatically.

`rename <oldname> <newname>`
: Rename `oldname` into `newname`. The files must be on the same drive.

`show [copying | warranty]`
: Show conditions for redistributing copies of C1541 ('copying') or the various kinds of warranty you do not have with C1541 ('warranty').

`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.

`verbose ["off"]`
: Enable or disable verbose output.

`version` Show C1541 version string

`write <source> [<destination>]`
: Write `source` from the file system into `destination` on a disk image.

  To create a REL file, you must specify the `destination` including the record length. As a special convenience, you can give it in decimal. Example: `write fsname imgname,l,100` Note that the size of the file may be rounded up to fill the last sector.

`unzip <d64name> <zipname> [<label,id>]`
: Create a D64 disk image out of a set of four Zipcoded files named `1!zipname`, `2!zipname`, `3!zipname` and `4!zipname`.

14.4 Executing shell commands

If you want to execute a shell command from within `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.

14.5 c1541 examples

`c1541 -attach test.d64 -list`
: Attach `test.d64` and show directory.

`c1541 -attach test.d64 -write test.prg testfile`
: Write `test.prg` to `test.d64` as `testfile`.

`c1541 -format "name,id" d64 disk.d64`

Create a disk image in `d64` format, format it with the `name` and `id` and save it to `disk.d64`.

15 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.

15.1 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.

`-s revision`
: This parameter is optional. It is only needed when converting to a .crt file. See the detailed description of the .crt format for which types support sub-types/revisions.

`-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)

`-p` This parameter is optional, when it is given cartconv will accept input files that do not match the cartridge sizes (useful for development).

`-b` This parameter is optional, when it is given cartconv will not omit banks that are empty (filled with $ff).

`-q` This parameter is optional, it disables all non-error messages

`--types` This parameter is optional. It shows all supported cartridge types (see below).

`--version`
: Show cartconv version string and exit

The following cartridge types are supported:

`bin`	Binary .bin file (Default crt->bin)
`normal`	Generic 8KiB/12KiB/16KiB .crt file (Default bin->crt)
`prg`	Binary C64 .prg file with load-address
`ulti`	Ultimax mode 4KiB/8KiB/16KiB .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
`bis`	BIS-Plus .crt file
`bb3`	Blackbox V3 .crt file
`bb4`	Blackbox V4 .crt file
`bb8`	Blackbox V8 .crt file
`bb9`	Blackbox V9 .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
`ecr`	Easy Calc Result .crt file
`epyx`	Epyx FastLoad .crt file
`exos`	EXOS .crt file
`expert`	Expert Cartridge .crt file
`f64`	Formel 64 .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
`gmod2`	GMod2 .crt file
`gmod3`	GMod3 .crt file
`gs`	C64 Games System .crt file
`ide64`	IDE64 .crt file
`ieee`	IEEE-488 Interface .crt file
`kcs`	KCS Power Cartridge .crt file
`ks`	Kingsoft .crt file
`ltk`	Lt. Kernal Host Adaptor .crt file
`mach5`	MACH 5 .crt file
`md`	Magic Desk .crt file
`mf`	Magic Formel .crt file
`max`	MAX Basic .crt file
`mikro`	Mikro Assembler .crt file
`mmc64`	MMC64 .crt file
`mmcr`	MMC Replay .crt file
`mv`	Magic Voice .crt file
`mm`	MultiMAX .crt file
`ocean`	Ocean .crt file
`p64`	Prophet64 .crt file
`pf`	Pagefox .crt file
`rep256`	REX 256KiB EPROM Cart .crt file, extra files can be inserted (1)(2)(3)
`rgcd`	RGCD .crt file
`ross`	ROSS .crt file
`rr`	Retro Replay .crt file
`rrnet`	RR-Net MK3 .crt file
`rrf`	REX RAM-Floppy .crt file
`ru`	REX Utility .crt file
`sdbox`	SD-BOX .crt file
`s64`	Snapshot 64 .crt file
`sb`	Structured BASIC .crt file
`se5`	Super Explode V5.0 .crt file

`sg` Super Games .crt file

`silver` Silverrock 128KiB Cartridge .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

`zipp` ZIPP-CODE 48 .crt file

- (1) insertion of 32KiB EPROM files supported.
- (2) insertion of 8KiB .crt/binary files supported.
- (3) insertion of 16KiB .crt/binary files supported.

15.2 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 32KiB EPROM file into an dep64 type cartridge.

- step 1 : use the dep64 binary file in VICE as a generic 8KiB 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 8KiB .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 8KiB .crt files into a rep256 type cartridge.

`cartconv -f foo.crt`
Get information about a .crt file.

16 petcat

The petcat program is a text conversion utility, it can convert between ASCII, PETSCII and tokenized BASIC.

16.1 petcat command line options

`-help` Output help text

`-?` Same as above

`-version` show petcat version string and exit

`-v` verbose output

`-c` controls (interpret also control codes) (default if textmode)

`-nc` no controls (suppress control codes in printout) (default if non-textmode)

`-qc` convert all non alphanumeric characters inside quotes into controlcodes

`-ic` interpret control codes case-insensitive

`-d` output raw codes in decimal

`-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:

`10` Basic v10.0 (C65/C64DX)

`1p` Basic v1.0 (PET)

`2` Basic v2.0 (C64/VIC20/PET)

`3` Basic v3.5 (C16)

`40` Basic v4.0 (PET/CBM2)

`4e`	Basic v2.0 with Basic v4.0 extension (C64)
`4v`	Basic v2.0 with Basic v4.0 extension (VIC20)
`5`	Basic v2.0 with Basic v5.0 extension (VIC20)
`70`	Basic v7.0 (C128)
`71`	Basic v7.0 with Basic v7.1 extension (C128)
`a`	Basic v2.0 with AtBasic (C64)
`bk`	Basic v2.0 with Kipper Basic (C64)
`blarg`	Basic v2.0 with Blarg (C64)
`bob`	Basic v2.0 with Basic on Bails (C64)
`bsx`	Basic v2.0 with Basex (C64)
`bwarsaw`	Basic v2.0 with Warsaw Basic (C64)
`bws`	Basic v2.0 with WS Basic (C64)
`bwsf`	Basic v2.0 with WS Basic final (C64)
`drago`	Basic v2.0 with Drago basic v2.2 (C64)
`easy`	Basic v2.0 with Easy Basic (VIC20)
`eve`	Basic v2.0 with Eve Basic (C64)
`exp20`	Basic v2.0 with Expanded Basic (VIC20)
`exp64`	Basic v2.0 with Expanded Basic (C64)
`f`	Basic v2.0 with Final Cartridge III (C64)
`game`	Basic v2.0 with Game Basic (C64)
`graph`	Basic v2.0 with Graphics basic (C64)
`lightning`	Basic v2.0 with Basic Lightning (C64)
`magic`	Basic v2.0 with Magic Basic (C64)
`mighty`	Basic v2.0 with Mighty Basic by Craig Bruce (VIC20)
`pegasus`	Basic v2.0 with Pegasus Basic v4.0 (C64)
`reu`	Basic v2.0 with REU-Basic (C64)
`simon`	Basic v2.0 with Simons' Basic extension (C64)
`speech`	Basic v2.0 with Speech Basic v2.7 (C64)
`superbas`	Basic v2.0 with Super Basic (C64)
`superexp`	Basic v2.0 with Super Expander (VIC20)
`supergra`	Basic v2.0 with Supergrafik 64 (C64)
`sxc`	Basic v2.0 with Super Expander Chip (C64)

`tt64` Basic v2.0 with The Tool 64 (C64)

`turtle` Basic v2.0 with Turtle Basic by Craig Bruce (VIC20)

`ultra` Basic v2.0 with Ultrabasic-64 (C64)

`xbasic` Basic v2.0 with Xbasic (C64)

16.2 petcat examples

`petcat -2 -o outputfile.txt -- inputfile.prg`
: De-tokenize, Convert inputfile.prg to a text file in outputfile.txt, using BASIC V2 only

`petcat -wsimon -o outputfile.prg -- inputfile.txt`
: Tokenize, Convert inputfile.txt to a PRG file in outputfile.prg, using Simons' BASIC

`petcat -text -o outputfile.txt -- inputfile.seq`
: Convert inputfile.seq to a Ascii text file in outputfile.txt.

`petcat -text -w2 -o outputfile.seq -- inputfile.txt`
: Convert inputfile.txt to a Petscii text SEQ file in outputfile.seq.

17 The emulator file formats

This chapter gives a technical description of the various files supported by the emulators.

17.1 The raw tape image format

(This section is based on v1.1 TAP specification by Per Hakan Sundell and Markus Brenner, additional format v2 expansion by Markus Brenner, plus some editing by groepaz)

Designed by Per Hakan Sundell (author of the CCS64 C64 emulator) in 1997, this format attempts to duplicate the data stored on a C64 cassette tape, bit for bit. Since it is simply a representation of the raw serial data from a tape, it should handle *any* custom tape loaders that exist.

The TAP images are generally very large, being a minimum of eight times, and up to sixteen times as large as what a raw PRG file would be. This is due to the way the data is stored, with each bit of the original file now being one byte large in the TAP file. The layout is fairly simple, with a small 14-byte header followed by file data.

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F        ASCII
      -----------------------------------------------   ----------------
0000: 43 36 34 2D 54 41 50 45 2D 52 41 57 00 00 00 00   C64-TAPE-RAW????
0010: 51 21 08 00 2F 0F 0D 31 64 1D 26 0D 07 21 0A 12   Q!??/??1d?&??!??
0020: 4A 2F 2C 34 07 18 0D 31 07 04 23 04 0D 42 0D 1E   J/,4???1??#??B??
0030: 34 04 42 0D 20 15 5E 04 0D 18 61 0D 26 29 34 0D   4?B???^???a?&)4?
0040: 23 0D 07 0A 3F 55 04 0A 13 3F 07 0D 12 2B 18 0A   #????U???????+??
```

Offset	Description
$0000-000B	File signature (`C64-TAPE-RAW` or `C16-TAPE-RAW`)
$000C	`G64` TAP version (see below for description), $00 - Original layout, $01 - Updated, $02 - Halfwave Extension
$000D	Computer Platform (0 = C64, 1 = VIC-20, 2 = C16, Plus/4)
$000E	Video Standard (0 = PAL, 1 = NTSC1, 2 = NTSC2)
000F	reserved for future expansion
$0010-0013	File data size (not including this header, in LOW/HIGH format) i.e. This image is $00082151 bytes long.
$0014-xxxx	File data

In TAP version $00 files, each data byte in the file data area represents the length of a pulse, when the C64's hardware will trigger again. This pulse length is determined by the following formula:

```
pulse length (in seconds) = (8 * data byte) / (clock cycles)
```

Therefore, a data value of $2F (47 in decimal) would be:

```
(47 * 8) / 985248 = .00038975 seconds.
```

A data value of $00 represents an "overflow" condition, any pulse length which is more that 255 * 8 in length.

The value of "clock cyvles" from above (985248) is based on the PAL value. Since this file format was developed in Europe, which is predominantly PAL video, this is only logical. The NTSC value would be 1022730, which is very close to the PAL, and therefore won't

cause a compatability problem converting European and NTSC tapes. I would stick to using the PAL value just in case.

In TAP version $01 files, the data value of $00 has been re-coded to represent values greater than 255 * 8. When a $00 is encountered, three bytes will follow which are the actual time (in cycles) of a pulse, and the above formula does not apply. The three bytes are stored in LOW/HIGH format.

Version 2 is an extension made by Markus Brenner for C16 tapes. It is version 1, but each value represents a halfwave, starting with a '0'->'1' transition. The time encoding doesn't change.

The actual interpretation of the serial data takes a little more work to explain. The typical ROM tape loader (and the turbo loaders) will initialize a timer with a specified value and start it counting down. If either the tape data changes or the timer runs out, an IRQ will occur. The loader will determine which condition caused the IRQ. If the tape data changed before the timer ran out, we have a short pulse, or a "0" bit. If the timer ran out first, we have a long pulse, or a "1" bit. Doing this continuously and we decode the entire file.

17.2 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.

17.2.1 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

17.2.2 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

17.2.3 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).

17.3 The G64 GCR-encoded disk image format

17.3.1 The original 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 Håkan 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.

17.3.2 An extension for double sided disks (Commodore VIC 1571)

Given that a Commodore VIC 1541/1570/1571 can read/write at most 42 tracks per side, it is quite natural to use the half-tracks from 1 to 84 for the first side and the following 84 half-tracks (i. e. halg.track 85 to 168) for the second size.

In order for this to work, byte 9 of the image (i. e. the maximum number of tracks) usually contains 168 for double sided disks.

To make identifying such images easy, the file signature in the first 8 bytes of such files will be `GCR-1571`).

17.3.3 Extra mastering info (SPS extension)

G64s created by the DTC tool from Kryoflux dumps contain extra info for each track, usually at offset $02ac right after the speedzone entries.

First comes the extra info tag:

Offset	Description
$00-$02	extra info tag (**EXT**) $45,$58,$54
$03	extra info version ($01)

Followed by 16 bytes of extra info for each track in the image:

Offset	Description
$00	write splice position
$04	write area size
$08	bitcell size in nanoseconds (eg 3500 means 3.5us)
$0c	track fill value
$0d	n/a
$0e	format code
$0f	format extension

A zero value in any of these fields means that the respective default/standard value applies.

Format codes:

ID	Description
0	Unknown format
1	GCR Data
2	CBM DOS
3	CBM DOS Extended
4	MicroProse
5	RapidLok
6	Datasoft
7	Vorpal
8	V-MAX!
9	Teque

10	TDP
11	Big Five
12	OziSoft

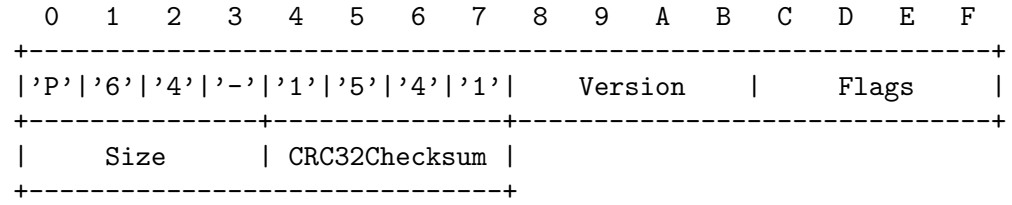
Format Extensions:

ID	Description
0	Unknown protection
1	Datasoft with Weak bits
2	CBM DOS with Cyan loader, Weak bits
3	CBM DOS with Datasoft, Weak bits
4	RapidLok Key
5	Data Duplication
6	Melbourne House
7	Melbourne House, Weak bits
8	PirateBusters v1.0
9	PirateBusters v2.0, Track A
10	PirateBusters v2.0, Track B
11	PirateSlayer
12	CBM DOS, XEMAG

17.4 The P64 NRZI flux pulse disk image format

This section is taken from "P64 file format specification" by Benjamin 'BeRo' Rosseaux.
All values are in little endian order !

17.4.1 P64 Header Layout



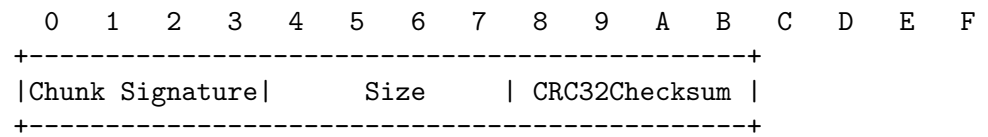
Version: File format version, current is 0x00000000

Size Size of the following whole chunk content stream

Flags: Bit 0 = Write protect Bit 1 = Number of sides (0 for single sided, 1 for double sided) Bit 2-31 = Reserved, all set to 0 when creating a file, preserve existing value when updating

CRC32CheckSum: CRC32 checksum of the following whole chunk content stream

17.4.2 P64 Chunk Header Layout



Chunk signature: Signature of chunk

Size: Size of the chunk data

CRC32CheckSum: CRC32 checksum of the chunk data

17.4.3 P64 Chunk 'HTPx' Layout

| x = half track index byte | +——————————+ Bit 7 of the half track index byte contains the side.

Track 18 = Half track 36 = Half track index byte decimal value 36

Half track NRZI transition flux pulse data chunk block

```
       0   1   2   3   4   5   6   7   8   9   A   B   C   D   E   F
     +-----------------------------------------------------------------+
0000: |  Count pulses |     Size      | ..... Range-encoded data .... |
     +-----------------------------------------------------------------+
```

Count pulses: Count of the NRZI transition flux pulses in half track

Size: Size of the range-encoded data

17.4.4 'HTPx' Range encoded data format

Hint: For a working C implememtation see p64.c and p64.h

The range coder is a FPAQ0-style range coder combined with 12-bit 0-order models, one model per byte with one bit per byte processing.

Sub stream	Count of models	Size per model	Total value bits
Position	4	65536	32
Strength	4	65536	32
Position flag	1	2	1
Strenth flag	1	2	1
Total models	10		

All initial model state values are initialized with zero.

All initial model probability values are initialized with 2048.

These model probability values will be updating in a adaptive way on the fly and not precalculated before the encoding and even not loaded before the decoding, see pseudo code below.

16000000 Hz / 5 rotations per second at 300 RPM = maximal 3200000 flux pulses

So NRZI transition flux pulse positions are in the 0 .. 3199999 value range, which is also a exact single rotation, where each time unit is a cycle at 16 MHz with 300 RPM as a mapping for the ideal case.

The NRZI transition flux pulse stength are in the 0x00000000 .. 0xffffffff value range, where 0xffffffff indices a strong flux pulse, that always triggers, and 0x00000001 indices a weak flux pulse, that almost never triggers, and 0x00000000 indices a flux pulse, that absolutly never triggers.

For 32-bit values, the model sub streams are subdivided byte wide in a little-endian manner, and each byte is processed bitwise with model probability shifting of 4 bits, just as:

Pascal-Style pseudo code:

```
procedure WriteDWord(Model, Value : longword);
var ByteValue, ByteIndex, Context, Bit : longword;
begin
  for ByteIndex := 0 to 3 do begin
    ByteValue := (Value shr (ByteIndex shl 3)) and $ff;
    Context := 1;
    for Bit := 7 downto 0 do begin
      Context := (Context shl 1) or RangeCoderEncodeBit(
                    RangeCoderProbabilities[
                     RangeCoderProbabilityOffsets[Model + ByteIndex] +
                     (((RangeCoderProbabilityStates[Model + ByteIndex]
                     shl 8) or Context) and $ffff)], 4, (ByteValue shr
                     Bit) and 1);
    end;
    RangeCoderProbabilityStates[Model+ByteIndex] := ByteValue;
  end;
end;
```

And for 1-bit flag values it is much simpler, but also with model probability shifting of 4 bits, just as:

Pascal-Style pseudo code:

```
procedure WriteBit(Model, Value : longword);
begin
   RangeCoderProbabilityStates[Model] :=
     RangeCoderEncodeBit(RangeCoderProbabilities[
       RangeCoderProbabilityOffsets[Model] +
         RangeCoderProbabilityStates[Model]], 4, Value and 1);
end;
```

The position and strength values are delta-encoded. If a value is equal to the last previous value, then the value will not encoded, instead, a flag for this will encoded. First the position value will encoded, then the stength value. If the last position delta is 0, then it is a track stream end marker.

Pascal-Style pseudo code:

```
LastPosition := 0;
PreviousDeltaPosition := 0

LastStrength := 0;

for PulseIndex := 0 to PulseCount - 1 do begin

  DeltaPosition := Pulses[PulseIndex].Position - LastPosition;
  if PreviousDeltaPosition <> DeltaPosition then begin
    PreviousDeltaPosition := DeltaPosition;
    WriteBit(ModelPositionFlag, 1)
    WriteDWord(ModelPosition, DeltaPosition);
```

```
  end else begin
    WriteBit(ModelPositionFlag, 0);
  end;
  LastPosition := Pulses[PulseIndex].Position;

  if LastStrength <> Pulses[PulseIndex].Strength then begin
    WriteBit(ModelStrengthFlag, 1)
    WriteDWord(ModelStrength, Pulses[PulseIndex].Strength - LastStrength);
  end else begin
    WriteBit(ModelStrengthFlag, 0);
  end;
  LastStrength := Pulses^[PulseIndex].Strength;

end;

// End code
WriteBit(ModelPositionFlag, 1);
WriteDWord(ModelPosition, 0);
```

The decoding is simply just in the another direction way.

Pseudo code for a FPAQ0-style carryless range coder:

Pascal-Style pseudo code:

```
procedure RangeCoderInit; // At encoding and decoding start
begin
  RangeCode := 0;
  RangeLow := 0;
  RangeHigh := $ffffffff;
end;

procedure RangeCoderStart; // At decoding start
var Counter : longword;
begin
  for Counter := 1 to 4 do begin
   RangeCode := (RangeCode shl 8) or ReadByteFromInput;
  end;
end;

procedure RangeCoderFlush; // At encoding end
var Counter : longword;
begin
  for Counter := 1 to 4 do begin
    WriteByteToOutput(RangeHigh shr 24);
    RangeHigh := RangeHigh shl 8;
  end;
end;

procedure RangeCoderEncodeNormalize;
```

```
begin
  while ((RangeLow xor RangeHigh) and $ff000000) = 0 do begin
   WriteByteToOutput(RangeHigh shr 24);
   RangeLow := RangeLow shl 8;
   RangeHigh := (RangeHigh shl 8) or $ff;
  end;
end;

function RangeCoderEncodeBit(var Probability : longword; Shift,
                             BitValue : longword) : longword;
begin
  RangeMiddle := RangeLow + (((RangeHigh - RangeLow) shr 12) *
                              Probability);
  if BitValue <> 0 then begin
    inc(Probability, ($fff - Probability) shr Shift);
    RangeHigh := RangeMiddle;
  end else begin
    dec(Probability, Probability shr Shift);
    RangeLow := RangeMiddle + 1;
  end;
  RangeCoderEncodeNormalize;
  result := BitValue;
end;

procedure RangeCoderDecodeNormalize;
begin
  while ((RangeLow xor RangeHigh) and $ff000000) = 0 do begin
    RangeLow := RangeLow shl 8;
    RangeHigh := (RangeHigh shl 8) or $ff;
    RangeCode := (RangeCode shl 8) or ReadByteFromInput;
  end;
end;

function RangeCoderDecodeBit(var Probability : longword;
                             Shift : longword) : longword;
begin
  RangeMiddle := RangeLow + (((RangeHigh - RangeLow) shr 12) *
                             Probability);
  if RangeCode <= RangeMiddle then begin
    inc(Probability, ($fff - Probability) shr Shift);
    RangeHigh := RangeMiddle;
    result := 1;
  end else begin
    dec(Probability, Probability shr Shift);
    RangeLow := RangeMiddle + 1;
    result := 0;
  end;
```

```
  RangeCoderDecodeNormalize;
end;
```

The probability may be never zero! But that can't happen here with this adaptive model in this P64 file format, since the adaptive model uses a shift factor of 4 bits and initial probabilities value of 2048, so the probability has a value range from 15 up to 4080 here. If you do want to use the above range coder routines for other stuff with other probability models, then you must to ensure that the probability output value is never zero, for example with "probability |= (probability < 1); " in C.

17.4.5 P64 Chunk 'DONE' Layout

This is the last empty chunk for to signalize that the correct file end is reached.

17.5 The D64 disk image format

(This section was contributed by Peter Schepers and slightly edited by Marco van den Heuvel. Added 42 track info by groepaz)

First and foremost we have D64, which is basically a sector-for-sector copy of a 1540/1541 disk. There are several versions of these which I will cover shortly. The standard D64 is a 174848 byte file comprised of 256 byte sectors arranged in 35 tracks with a varying number of sectors per track for a total of 683 sectors. Track counting starts at 1, not 0, and goes up to 35. Sector counting starts at 0, not 1, for the first sector, therefore a track with 21 sectors will go from 0 to 20.

The original media (a 5.25" disk) has the tracks laid out in circles, with track 1 on the very outside of the disk (closest to the sides) to track 35 being on the inside of the disk (closest to the inner hub ring). Commodore, in their infinite wisdom, varied the number of sectors per track and data densities across the disk to optimize available storage, resulting in the chart below. It shows the sectors/track for a standard D64. Since the outside diameter of a circle is the largest (versus closer to the center), the outside tracks have the largest amount of storage.

Track	**Sectors/track**	**# Sectors**
1-17	21	357
18-24	19	133
25-30	18	108
31-35	17	85
36-40(*)	17	85
41-42(*)	17	34

Track	**#Sect**	**#SectorsIn**	**D64 Offset**
1	21	0	$00000
2	21	21	$01500
3	21	42	$02A00
4	21	63	$03F00
5	21	84	$05400
6	21	105	$06900
7	21	126	$07E00
8	21	147	$09300
9	21	168	$0A800

10	21	189	$0BD00
11	21	210	$0D200
12	21	231	$0E700
13	21	252	$0FC00
14	21	273	$11100
15	21	294	$12600
16	21	315	$13B00
17	21	336	$15000
18	19	357	$16500
19	19	376	$17800
20	19	395	$18B00
21	19	414	$19E00
22	19	433	$1B100
23	19	452	$1C400
24	19	471	$1D700
25	18	490	$1EA00
26	18	508	$1FC00
27	18	526	$20E00
28	18	544	$22000
29	18	562	$23200
30	18	580	$24400
31	17	598	$25600
32	17	615	$26700
33	17	632	$27800
34	17	649	$28900
35	17	666	$29A00
36(*)	17	683	$2AB00
37(*)	17	700	$2BC00
38(*)	17	717	$2CD00
39(*)	17	734	$2DE00
40(*)	17	751	$2EF00
41(*)	17	768	$30000
42(*)	17	785	$31100

(*) Tracks 36-40 apply to 40- and 42-track images only. (*) Tracks 41-42 apply to 42-track images only.

The directory track should be contained totally on track 18. Sectors 1-18 contain the entries and sector 0 contains the BAM (Block Availability Map) and disk name/ID. Since the directory is only 18 sectors large (19 less one for the BAM), and each sector can contain only 8 entries (32 bytes per entry), the maximum number of directory entries is 18 * 8 = 144. The first directory sector is always 18/1, even though the t/s pointer at 18/0 (first two bytes) might point somewhere else. It then follows the same chain structure as a normal file, using a sector interleave of 3. This makes the chain links go 18/1, 18/4, 18/7 etc.

Note that you can extend the directory off of track 18, but only when reading the disk or image. Attempting to write to a directory sector not on track 18 will cause directory corruption. Each directory sector has the following layout (18/1 partial dump):

```
00: 12 04 81 11 00 4E 41 4D 45 53 20 26 20 50 4F 53 <- notice the T/S link
```

```
10: 49 54 A0 A0 A0 00 00 00 00 00 00 00 00 00 15 00 <- to 18/4 ($12/$04)
20: 00 00 84 11 02 41 44 44 49 54 49 4F 4E 41 4C 20 <- and how its not here
30: 49 4E 46 4F A0 11 0C FE 00 00 00 00 00 00 61 01 <- ($00/$00)
```

The first two bytes of the sector ($12/$04) indicate the location of the next track/sector of the directory (18/4). If the track is set to $00, then it is the last sector of the directory. It is possible, however unlikely, that the directory may *not* be competely on track 18 (some disks do exist like this). Just follow the chain anyhow.

When the directory is done, the track value will be $00. The sector link should contain a value of $FF, meaning the whole sector is allocated, but the actual value doesn't matter. The drive will return all the available entries anyways.

This is a breakdown of a standard directory sector:

Bytes	**Description**
$00-$1F	First directory entry
$20-$3F	Second dir entry
$40-$5F	Third dir entry
$60-$7F	Fourth dir entry
$80-$9F	Fifth dir entry
$A0-$BF	Sixth dir entry
$C0-$DF	Seventh dir entry
$E0-$FF	Eighth dir entry

This is a breakdown of a standard directory entry:

Bytes	**Description**
$00-$01	Track/Sector location of next directory sector ($00 $00 if not the first entry in the sector)
$02	File type
$03-$04	Track/sector location of first sector of file
$05-$14	16 character filename (in PETASCII, padded with $A0)
$15-$16	Track/Sector location of first side-sector block (REL file only)
$17	REL file record length (REL file only, max. value 254)
$18-$1D	Unused (except with GEOS disks)
$1E-$1F	File size in sectors, low/high byte order ($1E+$1F*256). The approx. filesize in bytes is <= #sectors * 254

The file type field is used as follows:

Bits	**Description**
0-3	The actual file type
4	Unused
5	Used only during SAVE-@ replacement
6	Locked flag (Set produces ">" locked files)
7	Closed flag (Not set produces "*", or "splat" files)

The actual file type can be one of the following:

Binary	**Decimal**	**File type**
0000	0	DEL
0001	1	SEQ
0010	2	PRG
0011	3	USR

0100	4	REL

Values 5-15 are illegal, but if used will produce very strange results. The 1541 is inconsistent in how it treats these bits. Some routines use all 4 bits, others ignore bit 3, resulting in values from 0-7.

Files, on a standard 1541, are stored using an interleave of 10. Assuming a starting track/sector of 17/0, the chain would run 17/0, 17/10, 17/20, 17/8, 17/18, etc.

17.5.1 Non-Standard & Long Directories

Most Commdore floppy disk drives use a single dedicated directory track where all filenames are stored. This limits the number of files stored on a disk based on the number of sectors on the directory track. There are some disk images that contain more files than would normally be allowed. This requires extending the directory off the default directory track by changing the last directory sector pointer to a new track, allocating the new sectors in the BAM, and manually placing (or moving existing) file entries there. The directory of an extended disk can be read and the files that reside there can be loaded without problems on a real drive. However, this is still a very dangerous practice as writing to the extended portion of the directory will cause directory corruption in the non-extended part. Many of the floppy drives core ROM routines ignore the track value that the directory is on and assume the default directory track for operations.

To explain: assume that the directory has been extended from track 18 to track 19/6 and that the directory is full except for a few slots on 19/6. When saving a new file, the drive DOS will find an empty file slot at 19/6 offset $40 and correctly write the filename and a few other things into this slot. When the file is done being saved the final file information will be written to 18/6 offset $40 instead of 19/6 causing some directory corruption to the entry at 18/6. Also, the BAM entries for the sectors occupied by the new file will not be saved and the new file will be left as a SPLAT (*) file.

Attempts to validate the disk will result in those files residing off the directory track to not be allocated in the BAM, and could also send the drive into an endless loop. The default directory track is assumed for all sector reads when validating so if the directory goes to 19/6, then the validate code will read 18/6 instead. If 18/6 is part of the normal directory chain then the validate routine will loop endlessly.

17.5.2 BAM layout

The layout of the BAM area (sector 18/0) is a bit more complicated. . .

```
     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
     -----------------------------------------------
 00: 12 01 41 00 12 FF F9 17 15 FF FF 1F 15 FF FF 1F
 10: 15 FF FF 1F 12 FF F9 17 00 00 00 00 00 00 00 00
 20: 00 00 00 00 0E FF 74 03 15 FF FF 1F 15 FF FF 1F
 30: 0E 3F FC 11 07 E1 80 01 15 FF FF 1F 15 FF FF 1F
 40: 15 FF FF 1F 15 FF FF 1F 0D C0 FF 07 13 FF FF 07
 50: 13 FF FF 07 11 FF CF 07 13 FF FF 07 12 7F FF 07
 60: 13 FF FF 07 0A 75 55 01 00 00 00 00 00 00 00 00
 70: 00 00 00 00 00 00 00 00 01 08 00 00 03 02 48 00
 80: 11 FF FF 01 11 FF FF 01 11 FF FF 01 11 FF FF 01
```

```
90: 53 48 41 52 45 57 41 52 45 20 31 20 20 A0 A0 A0
A0: A0 A0 56 54 A0 32 41 A0 A0 A0 A0 00 00 00 00 00
B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Bytes	**Description**
$00-$01	Track/Sector location of the first directory sector (should be set to 18/1 but it doesn't matter, and don't trust what is there, always go to 18/1 for first directory entry)
$02	Disk DOS version type (see note below) $41 ("A")
$03	Unused
$04-$8F	BAM entries for each track, in groups of four bytes per track, starting on track 1 (see below for more details)
$90-$9F	Disk Name (padded with $A0)
$A0-$A1	Filled with $A0
$A2-$A3	Disk ID
$A4	Usually $A0
$A5-$A6	DOS type, usually "2A"
$A7-$AA	Filled with $A0
$AB	Unused ($00)
$AC-$BF	For DOLPHIN DOS track 36-40 BAM entries, otherwise unused ($00)
$C0-$D3	For SPEED DOS track 36-40 BAM entries, otherwise unused ($00)
$D4-$FF	Unused ($00)

Note: The BAM entries for SPEED, DOLPHIN and ProLogic DOS use the same layout as standard BAM entries. One of the interesting things from the BAM sector is the byte at offset $02, the DOS version byte. If it is set to anything other than $41 or $00, then we have what is called "soft write protection". Any attempt to write to the disk will return the "DOS Version" error code 73 ,"CBM DOS V 2.6 1541". The 1541 is simply telling you that it thinks the disk format version is incorrect. This message will normally come up when you first turn on the 1541 and read the error channel. If you write a $00 or a $41 into 1541 memory location $00FF (for device 0), then you can circumvent this type of write-protection, and change the DOS version back to what it should be.

The BAM entries require a bit (no pun intended) more of a breakdown. Take the first entry at bytes $04-$07 ($12 $FF $F9 $17). The first byte ($12) is the number of free sectors on that track. Since we are looking at the track 1 entry, this means it has 18 (decimal) free sectors. The next three bytes represent the bitmap of which sectors are used/free. Since it is 3 bytes (8 bits/byte) we have 24 bits of storage. Remember that at most, each track only has 21 sectors, so there are a few unused bits.

Bytes	**Data**	**Description**
$04-$07	$12 $FF $F9 $17	Track 1 BAM
$08-$0B	$15 $FF $FF $FF	Track 2 BAM
$0C-$0F	$15 $FF $FF $1F	Track 3 BAM
...	...	...

$8C-$8F	$11 $FF $FF $01	Track 35 BAM

These entries must be viewed in binary to make any sense. We will use the first entry (track 1) at bytes 04-07:

```
FF=11111111, F9=11111001, 17=00010111
```

In order to make any sense from the binary notation, flip the bits around.

```
            111111 11112222
01234567 89012345 67890123
-------------------------
11111111 10011111 11101000
^                     ^
sector 0           sector 20
```

Since we are on the first track, we have 21 sectors, and only use up to the bit 20 position. If a bit is on (1), the sector is free. Therefore, track 1 has sectors 9, 10 and 19 used, all the rest are free. Any leftover bits that refer to sectors that don't exist, like bits 21-23 in the above example, are set to allocated.

Each filetype has its own unique properties, but most follow one simple structure. The first file sector is pointed to by the directory and follows a t/s chain, until the track value reaches $00. When this happens, the value in the sector link location indicates how much of the sector is used. For example, the following chain indicates a file 6 sectors long, and ends when we encounter the $00/$34 chain. At this point the last sector occupies from bytes $02-$34.

1	2	3	4	5	6
—-	—-	—-	—-	—-	—-
17/0	17/10	17/20	17/1	17/11	0/52
(11/00)	(11/0A)	(11/14)	(11/01)	(11/0B)	(0/34)

17.5.3 Variations on the D64 layout

These are some variations of the D64 layout:

1. Standard 35 track layout but with 683 error bytes added on to the end of the file. Each byte of the error info corresponds to a single sector stored in the D64, indicating if the sector on the original disk contained an error. The first byte is for track 1/0, and the last byte is for track 35/16.

2. A 40 track layout, following the same layout as a 35 track disk, but with 5 extra tracks. These contain 17 sectors each, like tracks 31-35. Some of the PC utilities do allow you to create and work with these files. This can also have error bytes attached like variant #1.

3. A 42 track layout, with two extra tracks of 17 sectors each. This is extremely uncommon, since real drives often have problems with accessing these tracks, software that uses them is very rare.

4. The Commodore 128 allowed for "auto-boot" disks. With this, t/s 1/0 holds a specific byte sequence which the computer recognizes as boot code.

Below is a small chart detailing the standard file sizes of D64 images, 35, 40 or 42 tracks, with or without error bytes.

Disk type	**Size**
35 track, no errors	174848

35 track, 683 error bytes	175531
40 track, no errors	196608
40 track, 768 error bytes	197376
42 track, no errors	205312
42 track, 802 error bytes	206114

The following table (provided by Wolfgang Moser) outlines the differences between the standard 1541 DOS and the various "speeder" DOS's that exist. The 'header 7/8' category is the 'fill bytes' as the end of the sector header of a real 1541 disk.

Disk format	**tracks**	**header 7/8**	**Dos type**	**Diskdos type**	**vs.**
Original CBM DOS v2.6	35	$0f $0f	"2A"	$41/'A'	
*SpeedDOS+	40	$0f $0f	"2A"	$41/'A'	
Professional DOS Initial	35	$0f $0f	"2A"	$41/'A'	
Professional DOS Version 1/Prototype	40	$0f $0f	"2A"	$41/'A'	
ProfDOS Release	40	$0f $0f	"4A"	$41/'A'	
Dolphin-DOS 2.0/3.0	35	$0f $0f	"2A"	$41/'A'	
Dolphin-DOS 2.0/3.0	40	$0d $0f	"2A"	$41/'A'	
PrologicDOS 1541	35	$0f $0f	"2A"	$41/'A'	
PrologicDOS 1541	40	$0f $0f	"2P"	$50/'P'	
ProSpeed 1571 2.0	35	$0f $0f	"2A"	$41/'A'	
ProSpeed 1571 2.0	40	$0f $0f	"2P"	$50/'P'	

*Note: There are also clones of SpeedDOS that exist, such as RoloDOS and DigiDOS. Both are just a change of the DOS startup string.

The location of the extra BAM information in sector 18/0, for 40 track images, will be different depending on what standard the disks have been formatted with. SPEED DOS stores them from $C0 to $D3, DOLPHIN DOS stores them from $AC to $BF and PrologicDOS stored them right after the existing BAM entries from $90-A3. PrologicDOS also moves the disk label and ID forward from the standard location of $90 to $A4. 64COPY and Star Commander let you select from several different types of extended disk formats you want to create/work with.

All three of the speeder DOS's mentioned above don't alter the standard sector interleave of 10 for files and 3 for directories. The reason is that they use a memory cache installed in the drive which reads the entire track in one pass. This alleviates the need for custom interleave values. They do seem to alter the algorithm that finds the next available free sector so that the interleave value can deviate from 10 under certain circumstances, but I don't know why they would bother.

Below is a HEX dump of a Speed DOS BAM sector. Note the location of the extra BAM info from $C0-D3.

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
      -----------------------------------------------
0070: 12 FF FF 03 12 FF FF 03 12 FF FF 03 11 FF FF 01
0080: 11 FF FF 01 11 FF FF 01 11 FF FF 01 11 FF FF 01
0090: A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0
00A0: A0 A0 30 30 A0 32 41 A0 A0 A0 A0 00 00 00 00 00
```

```
00B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00C0: 11 FF FF 01 11 FF FF 01 11 FF FF 01 11 FF FF 01
00D0: 11 FF FF 01 00 00 00 00 00 00 00 00 00 00 00 00
```

Below is a HEX dump of a Dolphin DOS BAM sector. Note the location of the extra BAM info from $AC-BF.

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
      -----------------------------------------------
0070: 12 FF FF 03 12 FF FF 03 12 FF FF 03 11 FF FF 01
0080: 11 FF FF 01 11 FF FF 01 11 FF FF 01 11 FF FF 01
0090: A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0
00A0: A0 A0 30 30 A0 32 41 A0 A0 A0 A0 00 11 FF FF 01
00B0: 11 FF FF 01 11 FF FF 01 11 FF FF 01 11 FF FF 01
00C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Below is a HEX dump of a PrologicDOS BAM sector. Note that the disk name and ID are now located at $A4 instead of starting at $90.

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
      -----------------------------------------------
0070: 12 FF FF 03 12 FF FF 03 12 FF FF 03 11 FF FF 01
0080: 11 FF FF 01 11 FF FF 01 11 FF FF 01 11 FF FF 01
0090: 11 FF FF 01 11 FF FF 01 11 FF FF 01 11 FF FF 01
00A0: 11 FF FF 01 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0
00B0: A0 A0 A0 A0 A0 A0 30 30 A0 32 50 A0 A0 A0 A0 00
00C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

17.5.4 Error codes

Here is the meaning of the error bytes added onto the end of any extended D64. The CODE is the same as that generated by the 1541 drive controller... it reports these numbers, not the error code we usually see when an error occurs.

Some of what comes below is taken from Immers/Neufeld book "Inside Commodore DOS". Note the descriptions are not completely accurate as to what the drive DOS is actually doing to seek/read/decode/write sectors, but serve as simple examples only. The "type" field is where the error usually occurs, whether it's searching for any SYNC mark, any header ID, any valid header, or reading a sector.

Code	Error	Type	D64	Description
$01	00	N/A	*	No error.
$02	20	Seek	*	Header block not found / Header descriptor byte not found
$03	21	Seek	*	No SYNC sequence found
$04	22	Read	*	Data descriptor byte not found
$05	23	Read	*	Checksum error in data block
$06	24	Write		Write verify on format (never occurs on 1541)
$07	25	Write		Write verify error

$08	26	Write		Write protect on
$09	27	Seek	*	Checksum error in header block
$0A	28	Write		Write error (never occurs on 1541)
$0B	29	Seek	*	Disk sector ID mismatch
$0F	74	Read		Drive Not Ready (no disk in drive or no device 1)

Codes $0 and $C to $E are unused and never occur.

These first errors are "seek" errors, where the disk controller is simply reading headers and looking at descriptor bytes, checksums, format ID's and reporting what errors it sees. These errors do *not* necessarily apply to the exact sector being looked for. This fact makes duplication of these errors very unreliable.

Code : $03 Error : 21 Type : Seek Message : No SYNC sequence found.

Each sector data block and header block are preceeded by SYNC marks. If *no* sync sequence is found within 20 milliseconds (only ~1/10 of a disk rotation!) then this error is generated. This error used to mean the entire track is bad, but it does not have to be the case. Only a small area of the track needs to be without a SYNC mark and this error will be generated.

Converting this error to a D64 is very problematic because it depends on where the physical head is on the disk when a read attempt is made. If it is on valid header/sectors then it won't occur. If it happens over an area without SYNC marks, it will happen.

Code : $02 Error : 20 Type : Seek Message : Header descriptor byte not found (HEX $08, GCR $52)

Each sector is preceeded by an 8-byte GCR header block, which starts with the value $52 (GCR). If this value is not found after 90 attempts, this error is generated.

Basically, what a track has is SYNC marks, and possibly valid data blocks, but no valid header descriptors.

Code : $09 Error : 27 Type : Seek Message : Checksum error in header block

The header block contains a checksum value, calculated by XOR'ing the TRACK, SECTOR, ID1 and ID2 values. If this checksum is wrong, this error is generated.

Code : $0B Error : 29 Type : Seek Message : Disk sector ID mismatch

The ID's from the header block of the currently read sector are compared against the ones from the low-level header of 18/0. If there is a mismatch, this error is generated.

Code : $02 Error : 20 Type : Seek Message : Header block not found

This error can be reported again when searching for the correct header block. An image of the header is built and searched for, but not found after 90 read attempts. Note the difference from the first occurance. The first one only searches for a valid ID, not the whole header.

Note that error 20 occurs twice during this phase. The first time is when a header ID is being searched for, the second is when the proper header pattern for the sector being searched for is not found.

From this point on, all the errors apply to the specific sector you are looking for. If a read passed all the previous checks, then we are at the sector being searched for.

Note that the entire sector is read before these errors are detected. Therefore the data, checksum and off bytes are available.

Code : $04 Error : 22 Type : Read Message : Data descriptor byte not found (HEX $07, GCR $55)

Each sector data block is preceeded by the value $07, the "data block" descriptor. If this value is not there, this error is generated. Each encoded sector has actually 260 bytes. First is the descriptor byte, then follows the 256 bytes of data, a checksum, and two "off" bytes.

Code : $05 Error : 23 Type : Read Message : Checksum error in data block

The checksum of the data read of the disk is calculated, and compared against the one stored at the end of the sector. If there's a discrepancy, this error is generated.

Code : $0F Error : 74 Type : Read Message : Drive Not Ready (no disk in drive or no device 1)

These errors only apply when writing to a disk. I don't see the usefulness of having these as they cannot be present when only *reading* a disk.

Code : $06 Error : 24 Type : Write Message : Write verify (on format)

Code : $07 Error : 25 Type : Write Message : Write verify error

Once the GCR-encoded sector is written out, the drive waits for the sector to come around again and verifies the whole 325-byte GCR block. Any errors encountered will generate this error.

Code : $08 Error : 26 Type : Write Message : Write protect on

Self explanatory. Remove the write-protect tab, and try again.

Code : $0A Error : 28 Type : Write Message : Write error

In actual fact, this error never occurs, but it is included for completeness.

This is not an error at all, but it gets reported when the read of a sector is ok.

Code : $01 Error : 00 Type : N/A Message : No error.

Self explanatory. No errors were detected in the reading and decoding of the sector.

The advantage with using the 35 track D64 format, regardless of error bytes, is that it can be converted directly back to a 1541 disk by either using the proper cable and software on the PC, or send it down to the C64 and writing it back to a 1541. It is the best documented format since it is also native to the C64, with many books explaining the disk layout and the internals of the 1541.

17.6 The X64 disk image format

(This section was contributed by Peter Schepers and slightly edited by Marco van den Heuvel.)

This file type, created by Teemu Rantanen, was used on the X64 emulator (a UNIX-based emulator) which has been superceeded by VICE. Both VICE and X64 support the X64 file standard, with VICE also supporting the regular D64 and T64 files.

Note that this ancient format is deprecated and subject for removal. It never got any momentum in the emulation community, and VICE never supported it for anything but 1541 disks.

X64 is not a specific type of file, but rather encompasses *all* known C64 disk types (hard disk, floppies, etc). An X64 is created by prepending a 64-byte header to an existing image (1541, 1571, etc) and setting specific bytes which describe what type of image follows. This header has undergone some revision, and this description is based on the 1.02 version, which was the last known at the time of writing.

The most common X64 file you will see is the D64 variety, typically 174912 bytes long (174848 for the D64 and 64 bytes for the header, assuming no error bytes are appended). The header layout (as used in 64COPY) is as follows:

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
      -----------------------------------------------
0000: 43 15 41 64 01 02 01 23 00 00 00 00 00 00 00 00
0010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0040: XX XX XX <- standard C64 image starts here....
```

Bytes	**Description**
$00-$03	This is the "Magic header" ($43 $15 $41 $64)
$04	Header version major ($01)
$05	Header version minor ($01, now its up to $02)
$06	Device type represented
$07	Maximum tracks in image (only in version 1.02 or greater) 1540/41/70: 35 1571: 35 1581: 80 (Logical single-sided disk)
$08	Number of disk sides in image. This value must be $00 for all 1541 and 1581 formats. $00=No second side $01=Second side
$09	Error data flag.
$0A-$1F	Unused, set to $00
$20-$3E	Disk image description (in ASCII or ISO Latin/1)
$3F	Always set to $00
$40-	Standard C64 file begins here.

The device types are:

Value	**Drive type**
$00	1540 See note below. . .
$01	1541 (Default)
$02	1542
$03	1551
$04	1570
$05	1571
$06	1572
$08	1581
$10	2031 or 4031
$11	2040 or 3040
$12	2041
$18	4040
$20	8050
$21	8060
$22	8061

$30	SFD-1001
$31	8250
$32	8280

The first four bytes used for the device type at position $06 ($00 to $03) are functionally the same, and are compatible with older version of X64 files. Some old X64 files might have $00 for the device type (instead of $01), but it makes no real difference.

As most instances of X64 files will be strictly 1541 images, bytes $08-$3F are set to zero, and some versions of the X64 emulator don't use bytes $08-$3F.

17.7 The D71 disk image format

(This section was contributed by Peter Schepers and slightly edited by Marco van den Heuvel.)

Similar to the D64 (1541), the 1571 drive can operate in either single-sided (1541 compatible) mode or double-sided (1571) mode. In this section I will be dealing with the double-sided mode only. For the breakdown of the single-sided mode, see the D64 section.

The D71 has 70 tracks, double that of the 1541, with a DOS file size of 349696 bytes. If the error byte block (1366 bytes) is attached, this makes the file size 351062 bytes. The track range and offsets into the D71 files are as follows:

Track	**Sec/trk**	**# Sectors**
1-17 (side 0)	21	357
18-24 (side 0)	19	133
25-30 (side 0)	18	108
31-35 (side 0)	17	85
36-52 (side 1)	21	357
53-59 (side 1)	19	133
60-65 (side 1)	18	108
66-70 (side 1)	17	85

Track	**#Sect**	**#SectorsIn**	**D71 Offset**
1	21	0	$00000
2	21	21	$01500
3	21	42	$02A00
4	21	63	$03F00
5	21	84	$05400
6	21	105	$06900
7	21	126	$07E00
8	21	147	$09300
9	21	168	$0A800
10	21	189	$0BD00
11	21	210	$0D200
12	21	231	$0E700
13	21	252	$0FC00
14	21	273	$11100
15	21	294	$12600
16	21	315	$13B00
17	21	336	$15000

18	19	357	$16500
19	19	376	$17800
20	19	395	$18B00
21	19	414	$19E00
22	19	433	$1B100
23	19	452	$1C400
24	19	471	$1D700
25	18	490	$1EA00
26	18	508	$1FC00
27	18	526	$20E00
28	18	544	$22000
29	18	562	$23200
30	18	580	$24400
31	17	598	$25600
32	17	615	$26700
33	17	632	$27800
34	17	649	$28900
35	17	666	$29A00
36	21	683	$2AB00
37	21	704	$2C000
38	21	725	$2D500
39	21	746	$2EA00
40	21	767	$2FF00
41	21	788	$31400
42	21	809	$32900
43	21	830	$33E00
44	21	851	$35300
45	21	872	$36800
46	21	893	$37D00
47	21	914	$39200
48	21	935	$3A700
49	21	956	$3BC00
50	21	977	$3D100
51	21	998	$3E600
52	21	1019	$3FB00
53	19	1040	$41000
54	19	1059	$42300
55	19	1078	$43600
56	19	1097	$44900
57	19	1116	$45C00
58	19	1135	$46F00
59	19	1154	$48200
60	18	1173	$49500
61	18	1191	$4A700
62	18	1209	$4B900
63	18	1227	$4CB00
64	18	1245	$4DD00

65	18	1263	$4EF00
66	17	1281	$50100
67	17	1298	$51200
68	17	1315	$52300
69	17	1332	$53400
70	17	1349	$54500

The directory structure is the same as a D64/1541. All the same filetypes apply, the directory still only holds 144 files per disk and should only exist on track 18.

The first two bytes of the sector ($12/$04 or 18/4) indicate the location of the next track/sector of the directory. If the track value is set to $00, then it is the last sector of the directory. It is possible, however unlikely, that the directory may *not* be competely on track 18 (some disks do exist like this). Just follow the chain anyhow.

When the directory is done, the track value will be $00. The sector link should contain a value of $FF, meaning the whole sector is allocated, but the actual value doesn't matter. The drive will return all the available entries anyways. This is a breakdown of a standard directory sector and entry:

```
    00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
    -----------------------------------------------
00: 12 04 82 11 00 4A 45 54 20 53 45 54 20 57 49 4C
10: 4C 59 A0 A0 A0 00 00 00 00 00 00 00 00 00 2B 00
20: 00 00 82 0F 01 4A 53 57 20 31 A0 A0 A0 A0 A0 A0
30: A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 BF 00
40: 00 00 82 06 03 53 4F 4E 20 4F 46 20 42 4C 41 47
50: 47 45 52 A0 A0 00 00 00 00 00 00 00 00 00 AE 00
60: 00 00 82 15 0D 50 4F 54 54 59 20 50 49 47 45 4F
70: 4E A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 A2 00
80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Bytes	Description
$00-$1F	First directory entry
$20-$3F	Second dir entry
$40-$5F	Third dir entry
$60-$7F	Fourth dir entry
$80-$9F	Fifth dir entry
$A0-$BF	Sixth dir entry
$C0-$DF	Seventh dir entry
$E0-$FF	Eighth dir entry

This is a breakdown of a standard directory entry:

Bytes	Description

$00-$01	Track/Sector location of next directory sector ($00/$FF if its the last sector)
$02	File type
$03-$04	Track/sector location of first sector of file
$05-$14	16 character filename (in PETASCII, padded with $A0)
$15-$16	Track/Sector location of first side-sector block (REL file only)
$17	REL file record length (REL file only, max. value 254)
$18-$1D	Unused (except with GEOS disks)
$1E-$1F	File size in sectors, low/high byte order ($1E+$1F*256). The approx. filesize in bytes is <= #sectors * 254

The file type field is used as follows:

Bits	**Description**
0-3	The actual file type
4	Unused
5	Used only during SAVE-@ replacement
6	Locked flag (Set produces ">" locked files)
7	Closed flag (Not set produces "*", or "splat" files)

The actual file type can be one of the following:

Binary	**Decimal**	**File type**
0000	0	DEL
0001	1	SEQ
0010	2	PRG
0011	3	USR
0100	4	REL

Values 5-15 are illegal, but if used will produce very strange results. The 1571 is inconsistent in how it treats these bits. Some routines use all 4 bits, others ignore bit 3, resulting in values from 0-7.

When the 1571 is in is native ("1571") mode, files are stored with a sector interleave of 6, rather than 10 which the 1541 (and the 1571 in "1541" mode) uses. The directory still uses an interleave of 3.

17.7.1 Non-Standard & Long Directories

Most Commodore floppy disk drives use a single dedicated directory track where all filenames are stored. This limits the number of files stored on a disk based on the number of sectors on the directory track. There are some disk images that contain more files than would normally be allowed. This requires extending the directory off the default directory track by changing the last directory sector pointer to a new track, allocating the new sectors in the BAM, and manually placing (or moving existing) file entries there. The directory of an extended disk can be read and the files that reside there can be loaded without problems on a real drive. However, this is still a very dangerous practice as writing to the extended portion of the directory will cause directory corruption in the non- extended part. Many of the floppy drives core ROM routines ignore the track value that the directory is on and assume the default directory track for operations.

To explain: assume that the directory has been extended from track 18 to track 19/6 and that the directory is full except for a few slots on 19/6. When saving a new file, the drive

DOS will find an empty file slot at 19/6 offset $40 and correctly write the filename and a few other things into this slot. When the file is done being saved the final file information will be written to 18/6 offset $40 instead of 19/6 causing some directory corruption to the entry at 18/6. Also, the BAM entries for the sectors occupied by the new file will not be saved and the new file will be left as a SPLAT (*) file.

Attempts to validate the disk will result in those files residing off the directory track to not be allocated in the BAM, and could also send the drive into an endless loop. The default directory track is assumed for all sector reads when validating so if the directory goes to 19/6, then the validate code will read 18/6 instead. If 18/6 is part of the normal directory chain then the validate routine will loop endlessly.

17.7.2 Bam layout

The BAM is somewhat different as it now has to take 35 new tracks into account. In order to do this, most of the extra BAM information is stored on track 53/0, and the remaining sectors on track 53 are marked in the BAM as allocated. This does mean that except for one allocated sector on track 53, the rest of the track is unused and wasted. (Track 53 is the equivalent to track 18, but on the flip side of the disk). Here is a dump of the first BAM sector. . .

```
    00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
    -----------------------------------------------
00: 12 01 41 80 12 FF F9 17 15 FF FF 1F 15 FF FF 1F
10: 15 FF FF 1F 15 FF FF 1F 15 FF FF 1F 15 FF FF 1F
20: 15 FF FF 1F 15 FF FF 1F 15 FF FF 1F 15 FF FF 1F
30: 15 FF FF 1F 15 FF FF 1F 15 FF FF 1F 15 FF FF 1F
40: 15 FF FF 1F 15 FF FF 1F 11 FC FF 07 13 FF FF 07
50: 13 FF FF 07 13 FF FF 07 13 FF FF 07 13 FF FF 07
60: 13 FF FF 07 12 FF FF 03 12 FF FF 03 12 FF FF 03
70: 12 FF FF 03 12 FF FF 03 12 FF FF 03 11 FF FF 01
80: 11 FF FF 01 11 FF FF 01 11 FF FF 01 11 FF FF 01
90: A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0
A0: A0 A0 30 30 A0 32 41 A0 A0 A0 A0 00 00 00 00 00
B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 15 15 15
E0: 15 15 15 15 15 15 15 15 15 15 15 15 15 15 00 13
F0: 13 13 13 13 13 12 12 12 12 12 12 11 11 11 11 11
```

Bytes	**Description**
$00-$01	Track/Sector location of the first directory sector (should be set to 18/1 but it doesn't matter, and don't trust what is there, always go to 18/1 for first directory entry)
$02	Disk DOS version type (see note below) $41 ('A') = 1541
$03	Double-sided flag $00 - Single sided disk $80 - Double sided disk

$04-8F	BAM entries for each track, in groups of four bytes per track, starting on track 1.
$90-$9F	Disk Name (padded with $A0)
$A0-$A1	Filled with $A0
$A2-$A3	Disk ID
$A4	Usually $A0
$A5-$A6	DOS type, usually "2A"
$A7-$AA	Filled with $A0
$AB-$DC	Not used ($00's)
$DD-$FF	Free sector count for tracks 36-70 (1 byte/track).

The "free sector" entries for tracks 36-70 are likely included here in the first BAM sector due to some memory restrictions in the 1571 drive. There is only enough memory available for one BAM sector, but in order to generate the "blocks free" value at the end of a directory listing, the drive needs to know the extra track "free sector" values. It does make working with the BAM a little more difficult, though.

These are the values that would normally be with the 4-byte BAM entry, but the rest of the entry is contained on 53/0.

Note: If the DOS version byte is set to anything other than $41 or $00, then we have what is called "soft write protection". Any attempt to write to the disk will return the "DOS Version" error code 73. The 1571 is simply telling you that it thinks the disk format version is incorrect.

The BAM entries require some explanation. Take the first entry at bytes $04-$07 ($12 $FF $F9 $17). The first byte ($12) is the number of free sectors on that track. Since we are looking at the track 1 entry, this means it has 18 (decimal) free sectors.

The next three bytes represent the bitmap of which sectors are used/free. Since it is 3 bytes (8 bits/byte) we have 24 bits of storage. Remember that at most, each track only has 21 sectors, so there are a few unused bits. These entries must be viewed in binary to make any sense. We will use the first entry (track 1) at bytes 04-07:

```
FF=11111111, F9=11111001, 17=00010111
```

In order to make any sense from the binary notation, flip the bits around.

```
           111111 11112222
01234567 89012345 67890123
-------------------------
11111111 10011111 11101000
^                    ^
sector 0           sector 20
```

Since we are on the first track, we have 21 sectors, and only use up to the bit 20 position. If a bit is on (1), the sector is free. Therefore, track 1 has sectors 9,10 and 19 used, all the rest are free.

In order to complete the BAM, we must check 53/0.

```
    00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
    -----------------------------------------------
00: FF FF 1F FF FF 1F FF FF 1F FF FF 1F FF FF 1F FF
```

```
10: FF 1F FF FF 1F FF FF 1F FF FF 1F FF FF 1F FF FF
20: 1F FF FF 1F FF FF 1F FF FF 1F FF FF 1F FF FF 1F
30: FF FF 1F 00 00 00 FF FF 07 FF FF 07 FF FF 07 FF
40: FF 07 FF FF 07 FF FF 07 FF FF 03 FF FF 03 FF FF
50: 03 FF FF 03 FF FF 03 FF FF 03 FF FF 01 FF FF 01
60: FF FF 01 FF FF 01 FF FF 01 00 00 00 00 00 00 00
70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Each track from 36-70 has 3 byte entries, starting at address $00.

```
Byte: $00-$02: $FF $FF $1F - BAM map for track 36
      $03-$05: $FF $FF $1F - BAM map for track 37
      ...
      $33-$35: $00 $00 $00 - BAM map for track 53
      ...
      $66-$68: $FF $FF $01 - BAM map for track 70
      $69-$FF:             - Not used
```

You can break down the entries for tracks 36-70 the same way as track 1, just combine the free sector bytes from 18/0 and the BAM usage from 53 to get the full 4-byte entry.

Just like a D64, you can attach error bytes to the file, for sector error information. This block is 1366 bytes long, 1 byte for each of the 1366 sectors in the image. With the error bytes, the file size is 351062 bytes.

17.8 The D81 disk image format

(This section was contributed by Peter Schepers and slightly edited by Marco van den Heuvel.)

Like D64 and D71, this is a byte for byte copy of a physical 1581 disk. It consists of 80 tracks, 40 sectors each (0 to 39) for a size of 819200 bytes, or 3200 sectors. If the error byte block is attached, this makes the file size 822400 bytes.

There are three sectors on the directory track used for disk internals (header and BAM), leaving 37 sectors for filename entries, thus allowing for 296 files (37 * 8) to be stored at the root level of the disk.

The actual physical layout on the disk is quite different from what the user sees, but this is unimportant to the scope of this section. One important difference from the D64 and D71 is all the sector interleaves are now 1 for both files and directory storage (rather than 3 for directory and 10 for file on a D64/D71). This is due to the built-in buffering in the 1581. When reading a sector, the whole track will be buffered in memory, and any sectors being modified will be done in memory. Once it has to be written, the whole track will be written out in one step.

The track range and offsets into the D81 files are as follows:

Track	**#Sect**	**#SectorsIn**	**D81 Offset**
1	40	0	$00000
2	40	40	$02800
3	40	80	$05000
4	40	120	$07800
5	40	160	$0A000
6	40	200	$0C800
7	40	240	$0F000
8	40	280	$11800
9	40	320	$14000
10	40	360	$16800
11	40	400	$19000
12	40	440	$1B800
13	40	480	$1E000
14	40	520	$20800
15	40	560	$23000
16	40	600	$25800
17	40	640	$28000
18	40	680	$2A800
19	40	720	$2D000
20	40	760	$2F800
21	40	800	$32000
22	40	840	$34800
23	40	880	$37000
24	40	920	$39800
25	40	960	$3C000
26	40	1000	$3E800
27	40	1040	$41000
28	40	1080	$43800
29	40	1120	$46000
30	40	1160	$48800
31	40	1200	$4B000
32	40	1240	$4D800
33	40	1280	$50000
34	40	1320	$52800
35	40	1360	$55000
36	40	1400	$57800
37	40	1440	$5A000
38	40	1480	$5C800
39	40	1520	$5F000
40	40	1560	$61800
41	40	1600	$64000
42	40	1640	$66800
43	40	1680	$69000
44	40	1720	$6B800

45	40	1760	$6E000
46	40	1800	$70800
47	40	1840	$73000
48	40	1880	$75800
49	40	1920	$78000
50	40	1960	$7A800
51	40	2000	$7D000
52	40	2040	$7F800
53	40	2080	$82000
54	40	2120	$84800
55	40	2160	$87000
56	40	2200	$89800
57	40	2240	$8C000
58	40	2280	$8E800
59	40	2320	$91000
60	40	2360	$93800
61	40	2400	$96000
62	40	2440	$98800
63	40	2480	$9B000
64	40	2520	$9D800
65	40	2560	$A0000
66	40	2600	$A2800
67	40	2640	$A5000
68	40	2680	$A7800
69	40	2720	$AA000
70	40	2760	$AC800
71	40	2800	$AF000
72	40	2840	$B1800
73	40	2880	$B4000
74	40	2920	$B6800
75	40	2960	$B9000
76	40	3000	$BB800
77	40	3040	$BE000
78	40	3080	$C0800
79	40	3120	$C3000
80	40	3160	$C5800

The header sector is stored at 40/0, and contains the disk name, ID and DOS version bytes, but the BAM is no longer contained here (like the D64).

```
    00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
    -----------------------------------------------
00: 28 03 44 00 31 35 38 31 20 55 54 49 4C 49 54 59
10: 20 56 30 31 A0 A0 47 42 A0 33 44 A0 A0 00 00 00
20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Bytes	**Description**
$00-$01	Track/Sector location of the first directory sector (should be set to 40/3 but it doesn't matter, and don't trust what is there, always go to 40/3 for first directory entry)
$02	Disk DOS version type (see note below) $44 ('D')=1581
$03	$00
$04-$13	16 character Disk Name (padded with $A0)
$14-$15	$A0
$16-$17	Disk ID
$18	$A0
$19	DOS Version ("3")
$1A	Disk version ("D")
$1B-$1C	$A0
$1D-$FF	Unused (usually $00)

The following might be set if the disk is a GEOS format (this info is based on the D64 layout, and might not prove to be true)

Bytes	**Description**
$AB-$AC	Border sector (GEOS only, else set to $00)
$AD-$BC	GEOS ID string ("geos FORMAT V1.x" GEOS only, else $00)
$BD-$FF	Unused (usually $00)

Note: If the DOS version byte is changed to anything other than a $44 (or $00), then we have what is called "soft write protection". Any attempt to write to the disk will return the "DOS Version" error code 73. The drive is simply telling you that it thinks the disk format version is incompatible.

The directory track should be contained totally on track 40. Sectors 3-39 contain the entries and sector 1 and 2 contain the BAM (Block Availability Map). Sector 0 holds the disk name and ID. The first directory sector is always 40/3, even though the t/s pointer at 40/0 (first two bytes) might point somewhere else. It goes linearly up the sector count, 3-4-5-6-etc. Each sector holds up to eight entries.

The first two bytes of the sector ($28/$04) indicate the location of the next track/sector of the directory (40/4). If the track is set to $00, then it is the last sector of the directory. It is possible, however unlikely, that the directory may *not* be competely on track 40. Just follow the chain anyhow.

When the directory is done (track=$00), the sector should contain an $FF, meaning the whole sector is allocated. Theactual value doesn't matter as all the entries will be returned anyways. Each directory sector has the following layout:

```
    00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
    -----------------------------------------------
00: 28 04 81 2B 00 53 43 52 45 45 4E 20 20 33 A0 A0
10: A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 02 00
20: 00 00 81 2B 01 53 43 52 45 45 4E 20 20 34 A0 A0
30: A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 03 00
40: 00 00 81 2B 02 53 43 52 45 45 4E 20 20 35 A0 A0
50: A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 07 00
60: 00 00 81 2B 08 53 43 52 45 45 4E 20 20 36 A0 A0
70: A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 08 00
80: 00 00 81 2B 14 53 43 52 45 45 4E 20 20 37 A0 A0
90: A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 07 00
A0: 00 00 81 24 00 53 43 52 45 45 4E 20 20 38 A0 A0
B0: A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 0B 00
C0: 00 00 82 24 04 46 49 4C 45 34 32 39 33 36 39 30
D0: A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 07 00
E0: 00 00 82 24 06 46 49 4C 45 32 35 37 38 38 31 35
F0: A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 05 00
```

Bytes	**Description**
$00-$1F	First directory entry
$20-$3F	Second dir entry
$40-$5F	Third dir entry
$60-$7F	Fourth dir entry
$80-$9F	Fifth dir entry
$A0-$BF	Sixth dir entry
$C0-$DF	Seventh dir entry
$E0-$FF	Eighth dir entry

This is a breakdown of a standard directory entry:

Bytes	**Description**
$00-$01	Track/Sector location of next directory sector
$02	File type
$03-$04	Track/sector location of first sector of file or partition
$05-$14	16 character filename (in PETASCII, padded with $A0)
$15-$16	Track/Sector location of first SUPER SIDE SECTOR block (REL file only)
$17	REL file record length (REL file only)
$18-$1B	Unused (except with GEOS disks)
$1C-$1D	(Used during an SAVE or OPEN, holds the new t/s link)
$1E-$1F	File or partition size in sectors, low/high byte order ($1E+$1F*256). The approx. file size in bytes is <= #sectors * 254

The file type field is used as follows:

Bits	**Description**

0-3	The actual file type
4	Unused
5	Used only during SAVE-@ replacement
6	Locked flag (Set produces `">"` locked files)
7	Closed flag (Not set produces `"*"`, or `"splat"` files)

The actual file type can be one of the following:

Binary	**Decimal**	**File type**
0000	0	DEL
0001	1	SEQ
0010	2	PRG
0011	3	USR
0100	4	REL
0101	5	CBM (partition or sub-directory)

Values 6-15 are illegal, but if used will produce very strange results.

17.8.1 Non-Standard & Long Directories

Most Commdore floppy disk drives use a single dedicated directory track where all filenames are stored. This limits the number of files stored on a disk based on the number of sectors on the directory track. There are some disk images that contain more files than would normally be allowed. This requires extending the directory off the default directory track by changing the last directory sector pointer to a new track, allocating the new sectors in the BAM, and manually placing (or moving existing) file entries there. The directory of an extended disk can be read and the files that reside there can be loaded without problems on a real drive. However, this is still a very dangerous practice as writing to the extended portion of the directory will cause directory corruption in the non-extended part. Many of the floppy drives core ROM routines ignore the track value that the directory is on and assume the default directory track for operations.

17.8.2 BAM layout

The BAM is located on 40/1 (for side 0, tracks 1-40) and 40/2 (for side 1, tracks 41-80). Each entry takes up six bytes, one for the `"free sector"` count and five for the allocation bitmap.

```
    00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
    -----------------------------------------------
00: 28 02 44 BB 47 42 C0 00 00 00 00 00 00 00 00 00
10: 28 FF FF FF FF FF 28 FF FF FF FF FF 28 FF FF FF
20: FF FF 28 FF FF FF FF FF 28 FF FF FF FF FF 28 FF
30: FF FF FF FF 28 FF FF FF FF FF 28 FF FF FF FF FF
40: 28 FF FF FF FF FF 28 FF FF FF FF FF 28 FF FF FF
50: FF FF 28 FF FF FF FF FF 28 FF FF FF FF FF 28 FF
60: FF FF FF FF 28 FF FF FF FF FF 28 FF FF FF FF FF
70: 28 FF FF FF FF FF 28 FF FF FF FF FF 28 FF FF FF
80: FF FF 28 FF FF FF FF FF 28 FF FF FF FF FF 28 FF
90: FF FF FF FF 28 FF FF FF FF FF 28 FF FF FF FF FF
A0: 28 FF FF FF FF FF 28 FF FF FF FF FF 28 FF FF FF
B0: FF FF 28 FF FF FF FF FF 28 FF FF FF FF FF 28 FF
```

```
C0: FF FF FF FF 28 FF FF FF FF FF 28 FF FF FF FF FF
D0: 28 FF FF FF FF FF 28 FF FF FF FF FF 28 FF FF FF
E0: FF FF 28 FF FF FF FF FF 28 FF FF FF FF FF 28 FF
F0: FF FF FF FF 28 FF FF FF FF FF 24 F0 FF 2D FF FE
```

```
Bytes:
$00-$01: Track/sector of next bam sector (40/2)
    $02: Version # ('D')
    $03: One's complement of version# ($BB)
$04-$05: Disk ID bytes (same as 40/0 Disk ID)
    $06: I/O byte
         bit 7 set - Verify on
         bit 7 clear - Verify off
         bit 6 set - Check header CRC
         bit 6 clear - Don't check header CRC
    $07: Auto-boot-loader flag
$08-$0F: Reserved for future (set to $00)
$10-$15: BAM entry for track 1 (track 41, side 1)
$16-$1B: BAM entry for track 2 (track 42, side 1)
         ...
$46-$4B: BAM entry for track 10 (track 50, side 1)
         ...
$82-$87: BAM entry for track 20 (track 60, side 1)
         ...
$BE-$C3: BAM entry for track 30 (track 70, side 1)
         ...
$FA-$FF: BAM entry for track 40 (track 80, side 1)
```

The BAM entries require some explanation, so lets look at the track 40 entry at bytes $FA-FF ($24 $F0 $FF $2D $FF $FE). The first byte ($24, or 36 decimal) is the number of free sectors on that track. The next five bytes represent the bitmap of which sectors are used/free. Since it is five bytes (8 bits/byte) we have 40 bits of storage. Since this format has 40 sectors/track, the whole five bytes are used.

```
 F0: .. .. .. .. .. .. .. .. .. .. 24 F0 FF 2D FF FE
```

The last five bytes of any BAM entry must be viewed in binary to make any sense. We will once again use track 40 as our reference:

```
 F0=11110000, FF=11111111, 2D=00101101, FF=11111111, FE=11111110
```

In order to make any sense from the binary notation, flip the bits around.

```
                   111111 11112222 22222233 33333333
 Sector 01234567 89012345 67890123 45678901 23456789
        ------------------------ -------- --------
        00001111 11111111 10110100 11111111 01111111
```

Note that if a bit is on (1), the sector is free. Therefore, track 40 has sectors 0-3, 17, 20, 22, 23 and 32 used, all the rest are free.

The second BAM (for side 1) contains the entries for tracks 41-80.

```
    00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
    -----------------------------------------------
```

```
00: 00 FF 44 BB 47 42 C0 00 00 00 00 00 00 00 00
10: 28 FF FF FF FF FF 28 FF FF FF FF FF 28 FF FF FF
20: FF FF 28 FF FF FF FF FF 28 FF FF FF FF FF 28 FF
30: FF FF FF FF 28 FF FF FF FF FF 28 FF FF FF FF FF
40: 28 FF FF FF FF FF 28 FF FF FF FF FF 28 FF FF FF
50: FF FF 28 FF FF FF FF FF 28 FF FF FF FF FF 28 FF
60: FF FF FF FF 28 FF FF FF FF FF 28 FF FF FF FF FF
70: 28 FF FF FF FF FF 28 FF FF FF FF FF 28 FF FF FF
80: FF FF 28 FF FF FF FF FF 28 FF FF FF FF FF 28 FF
90: FF FF FF FF 28 FF FF FF FF FF 28 FF FF FF FF FF
A0: 28 FF FF FF FF FF 28 FF FF FF FF FF 28 FF FF FF
B0: FF FF 28 FF FF FF FF FF 28 FF FF FF FF FF 28 FF
C0: FF FF FF FF 28 FF FF FF FF FF 28 FF FF FF FF FF
D0: 28 FF FF FF FF FF 28 FF FF FF FF FF 28 FF FF FF
E0: FF FF 28 FF FF FF FF FF 28 FF FF FF FF FF 28 FF
F0: FF FF FF FF 28 FF FF FF FF FF 28 FF FF FF FF FF
```

It is laid out exactly as the side 0 BAM except for one difference. The track/sector reference for the next sector should be set to $00/$FF, indicating there is no next sector.

17.8.3 REL files

The REL filetype requires some extra explaining. It was designed to make access to data *anywhere* on the disk very fast. Take a look at this directory entry. . .

```
 00: 00 FF 84 27 00 41 44 44 49 54 49 4F 4E 41 4C 20
 10: 49 4E 46 4F A0 27 02 FE 00 00 00 00 00 00 D2 0B
```

The third byte ($84) indicates this entry is a REL file and that the three normally empty entries at offset $15, $16 and $17 are now used as they are explained above. It's the track/sector chain that this entry points to, called the SUPER SIDE SECTOR, which is of interest here (in this case, 39/2). The SUPER SIDE SECTOR is very different from the D64 format. If you check the D64 entry for a REL file and do the calculations, you will find that the maximum file size of the REL file is 720 data sectors. With the new SUPER SIDE SECTOR, you can now have 126 groups of these SIDE SECTORS chains, allowing for file sizes up to (theoretically) 90720 sectors, or about 22.15 Megabytes.

Here is a dump of the beginning of the SUPER SIDE SECTOR. . .

```
 00: 27 01 FE 27 01 15 09 03 0F 38 16 4A 1C 00 00 00
 10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
Bytes:
$00-$01: Track/sector of first side sector in group 0
    $02: Always $FE
$03-$04: Track/sector of first side sector in group 0 (again)
         ...
$FD-$FE: Track/sector of first side sector in group 125
    $FF: Unused (likely $00)
```

The side sector layout is the same as the D64/1571.

```
    00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
```

```
    ------------------------------------------------
00: 12 0A 00 FE 15 09 12 0A 0F 0B 0C 0C 09 0D 06 0E
10: 15 07 15 08 15 0A 15 0B 15 0C 15 0D 15 0E 15 0F
20: 15 10 15 11 15 12 15 13 15 14 15 15 15 16 15 17
30: 15 18 15 19 15 1A 15 1B 15 1C 15 1D 15 1E 15 1F
40: 15 20 15 21 15 22 15 23 15 24 15 25 15 26 15 27
50: 14 00 14 01 14 02 14 03 14 04 14 05 14 06 14 07
60: 14 08 14 09 14 0A 14 0B 14 0C 14 0D 14 0E 14 0F
70: 14 10 14 11 14 12 14 13 14 14 14 15 14 16 14 17
80: 14 18 14 19 14 1A 14 1B 14 1C 14 1D 14 1E 14 1F
90: 14 20 14 21 14 22 14 23 14 24 14 25 14 26 14 27
A0: 13 00 13 01 13 02 13 03 13 04 13 05 13 06 13 07
B0: 13 08 13 09 13 0A 13 0B 13 0C 13 0D 13 0E 13 0F
C0: 13 10 13 11 13 12 13 13 13 14 13 15 13 16 13 17
D0: 13 18 13 19 13 1A 13 1B 13 1C 13 1D 13 1E 13 1F
E0: 13 20 13 21 13 22 13 23 13 24 13 25 13 26 13 27
F0: 12 00 12 01 12 02 12 03 12 04 12 05 12 06 12 07
Bytes:
    $00: Track location of next side-sector ($00 if last sector)
    $01: Sector location of next side-sector
    $02: Side-sector block number (first sector is $00, the next is
         $01, then $02, etc)
    $03: REL file RECORD size (from directory entry)
$04-$0F: Track/sector locations of the six other side-sectors. Note
         the first entry is this very sector we have listed here.
         The next is the next t/s listed at the beginning of the
         sector. All of this information must be correct. If one of
         these chains is $00/$00, then we have no more side sectors.
         Also, all of these (up to six) side sectors must have the
         same values in this range.
$10-$FF: T/S chains of *each* sector of the data portion. When we
         get a $00/$00, we are at the end of the file.
```

17.8.4 1581 Partitions and Sub-directories

At the beginning of this section it was stated that the 1581 can hold 296 entries "at the root level". The 1581 also has the ability to partition areas of the disk. Under the right conditions these can become sub-directories, acting as a small diskette, complete with its own directory and BAM. When you are inside of a sub-directory, no other files except those in that directory are visible, or can be affected.

To the 1581, this file will show up as a "CBM" filetype in a directory. All this does is tell the disk that a file, starting at X/Y track/sector and Z sectors large exists. Doing a validate will not harm these files as they have a directory entry, and are fully allocated in the BAM.

There are two main uses for partitions. One is to simply allocate a section of the disk to be used for direct-access reads/writes, and lock it away from being overwritten after a VALIDATE. The second is as a sub-directory, basically a small "disk within a disk".

In order to use a partition as a sub-directory, it must adhere to the following four rules:

```
1. If must start on sector 0
2. It's size must be in multiples of 40 sectors
3. It must be a minimum of 120 sectors long (3 tracks)
4. If must not start on or cross track 40, which limits the
   biggest directory to 1600 sectors (tracks 1-39).
```

This is a dump of a sub-directory entry:

```
00: 00 FF 85 29 00 50 41 52 54 49 54 49 4F 4E 20 31
10: A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 40 06
```

It is a partition starting on track 41/0, extends for 1600 sectors, and has been formatted as a sub-directory. Note that when a partition is created, the area being allocated is not touched in any way. If you want it set up as a sub-directory, you must issue the FORMAT command to the 1581 to create the central directory and BAM. Also note that from the directory entry you can't tell whether it is a sub-directory or not, just that it fits the sub-directory parameters.

The BAM track for the sub-directory exists on the first track of the partition, and has the same layout as the disk BAM on track 40. The biggest difference is the "disk name" is what what given when the partition was formatted rather than what the actual disk name is. Also, except for the free sectors in the partition area, all other sectors in the BAM will be allocated.

If the partition size doesn't match the above rules for a sub-directory, it will simply exist as a "protected" area of the disk, and can't be used as a sub-directory. Either way, it still shows up as a "CBM" type in a directory listing. Below is a dump of a 10-sector partition starting on track 5/1, which does not qualify as a sub-directory. . .

```
00: 00 00 85 05 01 53 4D 41 4C 4C 50 41 52 54 20 32
10: A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 0A 00
```

The master BAM shows the entry for this partition on track 5. . .

```
00: 28 02 44 BB 43 44 C0 00 00 00 00 00 00 00 00 00
10: 23 C1 FF FF FF FF 28 FF FF FF FF FF 28 FF FF FF
20: FF FF 28 FF FF FF FF FF 1E 01 F8 FF FF FF 28 FF
                           ^^^^^^^^^^^^^^^^^
```

The breakdown of the BAM shows the allocation for this track, with sectors 1-10 allocated, as it should be.

```
10000000 00011111 11111111 11111111 11111111
^        ^        ^        ^        ^

0        10       20       30       39
```

Partitions and sub-directories share one very important trait. When created, the sub-directory entry simply has the starting track/sector and the size of the partition in sectors. Partitions are created linearly, meaning if one starts on 30/1 and is of size 15 sectors, then the sector range from 1 through 15 on track 30 will be allocated. If a partition size crosses a track boundary, the allocation will continue on the next track starting on sector 0, and going up.

The section allocated will *not* have a track/sector chain like a file would, but rather is dependant on the directory entry to keep it from being overwritten. You can store whatever you want to in the allocated area.

17.8.5 AUTO-BOOT LOADER

If byte $07 in the BAM is set, then when the drive is reset (and other circumstances) it will look for a USR file called "COPYRIGHT CBM 86". This file will then be loaded into the drive RAM and executed.

The format for this auto-loader file is fairly basic. It starts with a two-byte load address, a size byte, program data, and a checksum at the end.

```
Bytes:
       $00-$01: Load address, low/high format
           $02: Size of program (SZ) (smaller than 256 bytes)
$03-($03+SZ-1): Program data
        $03+SZ: Checksum byte
```

17.9 The D80 disk image format

(This section was contributed by Peter Schepers and slightly edited by Marco van den Heuvel.)

This is a sector-for-sector copy of an 8050 floppy disk. The file size for an 8050 image is 533248 bytes. It is comprised of 256-byte sectors arranged across 77 tracks, with a varying number of sectors per track for a total of 2083 sectors. Track counting starts at 1 (not 0) and sector counting starts at 0 (not 1), therefore a track with 29 sectors will go from 0 to 28.

The original media (a 5.25" disk) has the tracks laid out in circles, with track 1 on the very outside of the disk (closest to the sides) to track 77 being on the inside of the disk (closest to the inner hub ring). Commodore, in their infinite wisdom, varied the number of sectors per track and data densities across the disk to optimize available storage, resulting in the chart below. It shows the sectors/track for a D80. Since the outside diameter of a circle is the largest (versus closer to the center), the outside tracks have the largest amount of storage.

Track Range	**Sectors/track**	**# Sectors**
1-39	29	1131
40-53	27	378
54-64	25	275
65-77	23	299

Track	**#Sect**	**#SectorsIn**	**D8x Offset**
1	29	0	$00000
2	29	29	$01D00
3	29	58	$03A00
4	29	87	$05700
5	29	116	$07400
6	29	145	$09100
7	29	174	$0AE00
8	29	203	$0CB00
9	29	232	$0E800
10	29	261	$10500
11	29	290	$12200

12	29	319	$13F00
13	29	348	$15C00
14	29	377	$17900
15	29	406	$19600
16	29	435	$1B300
17	29	464	$1D000
18	29	493	$1ED00
19	29	522	$20A00
20	29	551	$22700
21	29	580	$24400
22	29	609	$26100
23	29	638	$27E00
24	29	667	$29B00
25	29	696	$2B800
26	29	725	$2D500
27	29	754	$2F200
28	29	783	$30F00
29	29	812	$32C00
30	29	841	$34900
31	29	870	$36600
32	29	899	$38300
33	29	928	$3A000
34	29	957	$3BD00
35	29	986	$3DA00
36	29	1015	$3F700
37	29	1044	$41400
38	29	1073	$43100
39	29	1102	$44E00
40	27	1131	$46B00
41	27	1158	$48600
42	27	1185	$4A100
43	27	1212	$4BC00
44	27	1239	$4D700
45	27	1266	$4F200
46	27	1293	$50D00
47	27	1320	$52800
48	27	1347	$54300
49	27	1374	$55E00
50	27	1401	$57900
51	27	1428	$59400
52	27	1455	$5AF00
53	27	1482	$5CA00
54	25	1509	$5E500
55	25	1534	$5FE00
56	25	1559	$61700
57	25	1584	$63000
58	25	1609	$64900

59	25	1634	$66200
60	25	1659	$67B00
61	25	1684	$69400
62	25	1709	$6AD00
63	25	1734	$6C600
64	25	1759	$6DF00
65	23	1784	$6F800
66	23	1807	$70F00
67	23	1830	$72600
68	23	1853	$73D00
69	23	1876	$75400
70	23	1899	$76B00
71	23	1922	$78200
72	23	1945	$79900
73	23	1968	$7B000
74	23	1991	$7C700
75	23	2014	$7DE00
76	23	2037	$7F500
77	23	2060	$80C00

The BAM (Block Availability Map) is on track 38. The D80 is only 77 tracks and so the BAM is contained on 38/0 and 38/3. The BAM interleave is 3.

The directory is on track 39, with 39/0 contains the header (DOS type, disk name, disk ID's) and sectors 1-28 contain the directory entries. Both files and the directory use an interleave of 1. Since the directory is only 28 sectors large (29 less one for the header), and each sector can contain only 8 entries (32 bytes per entry), the maximum number of directory entries is 28 * 8 = 224. The first directory sector is always 39/1. It then follows a chain structure using a sector interleave of 1 making the links go 39/1, 39/2, 39/3 etc.

When reading a disk, you start with 39/0 (disk label/ID) which points to 38/0 (BAM0), 38/3 (BAM1), and finally to 39/1 (first dir entry sector). When writing a file to a blank disk, it will start at 38/1 because 38/0 is already allocated.

Below is a dump of the header sector 39/0:

```
    00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
    -----------------------------------------------
00: 26 00 43 00 00 00 73 61 6D 70 6C 65 20 64 38 30
10: A0 A0 A0 A0 A0 A0 A0 A0 65 72 A0 32 43 A0 A0 A0
20: A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
...
F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Bytes	**Description**
$00-$01	T/S pointer to first BAM sector (38/0)
$02	$43 'C' is for DOS format version
$03	Reserved
$04-$05	Unused
$06-$16	Disk name, padded with 0xA0 ("sample d80")
$17	0xA0

$18-$19	Disk ID bytes "er"
$1A	0xA0
$1B-$1C	DOS version bytes "2C"
$1D-$20	0xA0
$21-$FF	Unused

Below is a dump of the first directory sector, 39/1

```
    00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
    -----------------------------------------------
00: 27 02 82 26 01 54 45 53 54 A0 A0 A0 A0 A0 A0 A0
10: A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 01 00
20: 00 00 82 26 02 54 45 53 54 32 A0 A0 A0 A0 A0 A0
30: A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 01 00
40: 00 00 82 26 04 54 45 53 54 33 A0 A0 A0 A0 A0 A0
50: A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 05 00
60: 00 00 82 26 0B 54 45 53 54 34 A0 A0 A0 A0 A0 A0
70: A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 09 00
80: 00 00 82 26 14 54 45 53 54 35 A0 A0 A0 A0 A0 A0
90: A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 0C 00
A0: 00 00 82 28 00 54 45 53 54 36 A0 A0 A0 A0 A0 A0
B0: A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 01 00
C0: 00 00 82 28 01 54 45 53 54 37 A0 A0 A0 A0 A0 A0
D0: A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 01 00
E0: 00 00 82 28 02 54 45 53 54 38 A0 A0 A0 A0 A0 A0
F0: A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 01 00
```

The first two bytes of the directory sector ($27/$02) indicate the location of the next track/sector of the directory (39/2). If the track is set to $00, then it is the last sector of the directory.

When the directory is done, the track value will be $00. The sector link should contain a value of $FF, meaning the whole sector is allocated, but the actual value doesn't matter. The drive will return all the available entries anyways. This is a breakdown of a standard directory sector:

Bytes	**Description**
$00-$1F	First directory entry
$20-$3F	Second dir entry
$40-$5F	Third dir entry
$60-$7F	Fourth dir entry
$80-$9F	Fifth dir entry
$A0-$BF	Sixth dir entry
$C0-$DF	Seventh dir entry
$E0-$FF	Eighth dir entry

This is a breakdown of a standard directory entry:

Bytes	**Description**
$00-$01	Track/Sector location of next directory sector ($00 $00 if not the first entry in the sector)
$02	File type

$03-$04	Track/sector location of first sector of file
$05-$14	16 character filename (in PETASCII, padded with $A0)
$15-$16	Track/Sector location of first side-sector block (REL file only)
$17	REL file record length (REL file only, max. value 254)
$18-$1D	Unused
$1E-$1F	File size in sectors, low/high byte order ($1E+$1F*256). The approx. filesize in bytes is <= #sectors * 254

The file type field is used as follows:

Bits	Description
0-3	The actual file type
4	Unused
5	Used only during SAVE-@ replacement
6	Locked flag (Set produces ">" locked files)
7	Closed flag (Not set produces "*", or "splat" files)

The actual file type can be one of the following:

Binary	Decimal	File type
0000	0	DEL
0001	1	SEQ
0010	2	PRG
0011	3	USR
0100	4	REL

Values 5-15 are illegal, but if used will produce very strange results.

17.9.1 Non-Standard & Long Directories

Most Commdore floppy disk drives use a single dedicated directory track where all filenames are stored. This limits the number of files stored on a disk based on the number of sectors on the directory track. There are some disk images that contain more files than would normally be allowed. This requires extending the directory off the default directory track by changing the last directory sector pointer to a new track, allocating the new sectors in the BAM, and manually placing (or moving existing) file entries there. The directory of an extended disk can be read and the files that reside there can be loaded without problems on a real drive. However, this is still a very dangerous practice as writing to the extended portion of the directory will cause directory corruption in the non-extended part. Many of the floppy drives core ROM routines ignore the track value that the directory is on and assume the default directory track for operations.

17.9.2 BAM layout

The BAM only occupies up to four sectors on track 38, so the rest of the track is empty and is available for file storage. Below is a dump of the first BAM block, 38/0. A D80 will only contain two BAM sectors, 38/0 and 38/3. Each entry takes 5 bytes, 1 for the free count on that track, and 4 for the BAM bits.

```
    00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
    -----------------------------------------------
00: 26 03 43 00 01 33 1D FF FF FF 1F 1D FF FF FF 1F
10: 1D FF FF FF 1F 1D FF FF FF 1F 1D FF FF FF 1F 1D
```

```
20: FF FF FF 1F 1D FF FF FF 1F 1D FF FF FF 1F 1D FF
30: FF FF 1F 1D FF FF FF 1F 1D FF FF FF 1F 1D FF FF
40: FF 1F 1D FF FF FF 1F 1D FF FF FF 1F 1D FF FF FF
50: 1F 1D FF FF FF 1F 1D FF FF FF 1F 1D FF FF FF 1F
60: 1D FF FF FF 1F 1D FF FF FF 1F 1D FF FF FF 1F 1D
70: FF FF FF 1F 1D FF FF FF 1F 1D FF FF FF 1F 1D FF
80: FF FF 1F 1D FF FF FF 1F 1D FF FF FF 1F 1D FF FF
90: FF 1F 1D FF FF FF 1F 1D FF FF FF 1F 1D FF FF FF
A0: 1F 1D FF FF FF 1F 1D FF FF FF 1F 1D FF FF FF 1F
B0: 1D FF FF FF 1F 1D FF FF FF 1F 1D FF FF FF 1F 1B
C0: F6 FF FF 1F 1B FC FF FF 1F 1B FF FF FF 07 1B FF
D0: FF FF 07 1B FF FF FF 07 1B FF FF FF 07 1B FF FF
E0: FF 07 1B FF FF FF 07 1B FF FF FF 07 1B FF FF FF
F0: 07 1B FF FF FF 07 1B FF FF FF 07 1B FF FF FF 07
```

Bytes	**Description**
$00-$01	T/S pointer to second BAM sector (38/3)
$02	DOS version byte (0x43='C')
$03	Reserved
$04	Lowest track covered by this BAM (0x01=1)
$05	Highest+1 track covered by this BAM (0x33=51)
$06-$0A	BAM for track 1. The first byte shows the "blocks free" for this track, the remaining 4 show the BAM for the track.
$0B-$0F	BAM for track 2
...	...
$FB-$FF	BAM for track 50

Being bit-based, the BAM entries need some explanation. The first track entry in the above BAM sector is at offset 06, "1D FF FF FF 1F". The first number is how many blocks are free on this track ($1D=29) and the remainder is the bit representation of the usage map for the track. These entries must be viewed in binary to make any sense. First convert the values to binary:

```
FF=11111111, FF=11111111, FF=11111111, 1F=00011111
```

In order to make any sense from the binary notation, flip the bits around.

```
           111111 11112222 222222
01234567 89012345 67890123 456789...
------------------------ ---------
11111111 11111111 11111111 11111000
^                             ^
sector 0                   sector 28
```

Since we are on the first track, we have 29 sectors, and only use up to the bit 28 position. If a bit is on (1), the sector is free. Therefore, track 1 is clean, all sectors are free. Any leftover bits that refer to sectors that don't exist, like bits 29-31 in the above example, are set to allocated.

Second BAM block 38/3.

```
    00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
    -----------------------------------------------
```

```
00: 27 01 43 00 33 4E 1B FF FF FF 07 1B FF FF FF 07
10: 1B FF FF FF 07 19 FF FF FF 01 19 FF FF FF 01 19
20: FF FF FF 01 19 FF FF FF 01 19 FF FF FF 01 19 FF
30: FF FF 01 19 FF FF FF 01 19 FF FF FF 01 19 FF FF
40: FF 01 19 FF FF FF 01 19 FF FF FF 01 17 FF FF 7F
50: 00 17 FF FF 7F 00 17 FF FF 7F 00 17 FF FF 7F 00
60: 17 FF FF 7F 00 17 FF FF 7F 00 17 FF FF 7F 00 17
70: FF FF 7F 00 17 FF FF 7F 00 17 FF FF 7F 00 17 FF
80: FF 7F 00 17 FF FF 7F 00 17 FF FF 7F 00 00 00 00
90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
B0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Bytes	**Description**
$00-$01	T/S pointer to second BAM sector (39/1)
$02	DOS version byte (0x43='C')
$03	Reserved
$04	Lowest track covered by this BAM (0x33=51)
$05	Highest+1 track covered by this BAM (0x43=78)
$06-$0A	BAM for track 51. The first byte shows the "blocks free" for this track, the remaining 4 show the BAM for the track.
$0B-$0F	BAM for track 52
...	...
$88-$8C	BAM for track 77
$8D-$FF	Not used

17.10 The D82 disk image format

(This section was contributed by Peter Schepers and slightly edited by Marco van den Heuvel.)

This is a sector-for-sector copy of an 8250 floppy disk. The file size for an 8250 image is 1066496 bytes. It is comprised of 256-byte sectors arranged across 154 tracks, with a varying number of sectors per track for a total of 4166 sectors. Track counting starts at 1 (not 0) and sector counting starts at 0 (not 1), therefore a track with 29 sectors will go from 0 to 28.

The original media (a 5.25" disk) has the tracks laid out in circles, with track 1 on the very outside of the disk (closest to the sides) to track 77 being on the inside of the disk (closest to the inner hub ring). Commodore, in their infinite wisdom, varied the number of sectors per track and data densities across the disk to optimize available storage, resulting in the chart below. It shows the sectors/track for a D82. Since the outside diameter of a circle is the largest (versus closer to the center), the outside tracks have the largest amount of storage.

Track Range	**Sectors/track**	**# Sectors**

1-39	29	1131
40-53	27	378
55-64	25	275
65-77	23	299
78-116	29	1131
117-130	27	378
131-141	25	275
142-154	23	299

Track	**#Sect**	**#SectorsIn**	**D82 Offset**
1	29	0	$000000
2	29	29	$001D00
3	29	58	$003A00
4	29	87	$005700
5	29	116	$007400
6	29	145	$009100
7	29	174	$00AE00
8	29	203	$00CB00
9	29	232	$00E800
10	29	261	$010500
11	29	290	$012200
12	29	319	$013F00
13	29	348	$015C00
14	29	377	$017900
15	29	406	$019600
16	29	435	$01B300
17	29	464	$01D000
18	29	493	$01ED00
19	29	522	$020A00
20	29	551	$022700
21	29	580	$024400
22	29	609	$026100
23	29	638	$027E00
24	29	667	$029B00
25	29	696	$02B800
26	29	725	$02D500
27	29	754	$02F200
28	29	783	$030F00
29	29	812	$032C00
30	29	841	$034900
31	29	870	$036600
32	29	899	$038300
33	29	928	$03A000
34	29	957	$03BD00
35	29	986	$03DA00
36	29	1015	$03F700
37	29	1044	$041400

38	29	1073	$043100
39	29	1102	$044E00
40	27	1131	$046B00
41	27	1158	$048600
42	27	1185	$04A100
43	27	1212	$04BC00
44	27	1239	$04D700
45	27	1266	$04F200
46	27	1293	$050D00
47	27	1320	$052800
48	27	1347	$054300
49	27	1374	$055E00
50	27	1401	$057900
51	27	1428	$059400
52	27	1455	$05AF00
53	27	1482	$05CA00
54	25	1509	$05E500
55	25	1534	$05FE00
56	25	1559	$061700
57	25	1584	$063000
58	25	1609	$064900
59	25	1634	$066200
60	25	1659	$067B00
61	25	1684	$069400
62	25	1709	$06AD00
63	25	1734	$06C600
64	25	1759	$06DF00
65	23	1784	$06F800
66	23	1807	$070F00
67	23	1830	$072600
68	23	1853	$073D00
69	23	1876	$075400
70	23	1899	$076B00
71	23	1922	$078200
72	23	1945	$079900
73	23	1968	$07B000
74	23	1991	$07C700
75	23	2014	$07DE00
76	23	2037	$07F500
77	23	2060	$080C00
78	29	2083	$082300
79	29	2112	$084000
80	29	2141	$085D00
81	29	2170	$087A00
82	29	2199	$089700
83	29	2228	$08B400
84	29	2257	$08D100

85	29	2286	$08EE00
86	29	2315	$090600
87	29	2344	$092800
88	29	2373	$094500
89	29	2402	$096200
90	29	2431	$097F00
91	29	2460	$099C00
92	29	2489	$09B900
93	29	2518	$09D600
94	29	2547	$09F300
95	29	2576	$0A1000
96	29	2605	$0A2D00
97	29	2634	$0A4A00
98	29	2663	$0A6700
99	29	2692	$0A8400
100	29	2721	$0AA100
101	29	2750	$0ABE00
102	29	2779	$0ADB00
103	29	2808	$0AF800
104	29	2837	$0B1500
105	29	2866	$0B3200
106	29	2895	$0B4F00
107	29	2924	$0B6C00
108	29	2953	$0B8900
109	29	2982	$0BA600
110	29	3011	$0BC300
111	29	3040	$0BE000
112	29	3069	$0BFD00
113	29	3098	$0C1A00
114	29	3137	$0C3700
115	29	3156	$0C5400
116	29	3185	$0C7100
117	27	3214	$0C8E00
118	27	3241	$0CA900
119	27	3268	$0CC400
120	27	3295	$0CDF00
121	27	3322	$0CFA00
122	27	3349	$0D1500
123	27	3376	$0D3000
124	27	3403	$0D4B00
125	27	3430	$0D6600
126	27	3457	$0D8100
127	27	3484	$0D9C00
128	27	3511	$0DB700
129	27	3538	$0DD200
130	27	3565	$0DED00
131	25	3592	$0E0800

132	25	3617	$0E2100
133	25	3642	$0E3A00
134	25	3667	$0E5300
135	25	3692	$0E6C00
136	25	3717	$0E8500
137	25	3742	$0E9E00
138	25	3767	$0EB700
139	25	3792	$0ED000
140	25	3817	$0EE900
141	25	3842	$0F0200
142	23	3867	$0F1B00
143	23	3890	$0F3200
144	23	3913	$0F4900
145	23	3936	$0F6000
146	23	3959	$0F7700
147	23	3982	$0F8E00
148	23	4005	$0FA500
149	23	4028	$0FBC00
150	23	4051	$0FD300
151	23	4074	$0FEA00
152	23	4097	$100100
153	23	4120	$101800
154	23	4143	$102F00

The BAM (Block Availability Map) is on track 38. The D82 is 154 tracks and so the BAM is contained on 38/0, 38/3, 38/6 and 38/9. The BAM interleave is 3.

The directory is on track 39, with 39/0 contains the header (DOS type, disk name, disk ID's) and sectors 1-28 contain the directory entries. Both files and the directory use an interleave of 1. Since the directory is only 28 sectors large (29 less one for the header), and each sector can contain only 8 entries (32 bytes per entry), the maximum number of directory entries is 28 * 8 = 224. The first directory sector is always 39/1. It then follows a chain structure using a sector interleave of 1 making the links go 39/1, 39/2, 39/3 etc.

When reading a disk, you start with 39/0 (disk label/ID) which points to 38/0 (BAM0), 38/3 (BAM1), 38/6 (BAM2), 38/9 (BAM3, and finally to 39/1 (first dir entry sector). When writing a file to a blank disk, it will start at 38/1 because 38/0 is already allocated.

Below is a dump of the header sector 39/0:

```
    00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
    -----------------------------------------------
00: 26 00 43 00 00 00 73 61 6D 70 6C 65 20 64 38 30
10: A0 A0 A0 A0 A0 A0 A0 A0 65 72 A0 32 43 A0 A0 A0
20: A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
...
F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Bytes	**Description**
$00-$01	T/S pointer to first BAM sector (38/0)
$02	$43 'C' is for DOS format version

$03	Reserved
$04-$05	Unused
$06-$16	Disk name, padded with 0xA0 ("sample d82")
$17	0xA0
$18-$19	Disk ID bytes "er"
$1A	0xA0
$1B-$1C	DOS version bytes "2C"
$1D-$20	0xA0
$21-$FF	Unused

Below is a dump of the first directory sector, 39/1

```
    00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
    -----------------------------------------------
00: 27 02 82 26 01 54 45 53 54 A0 A0 A0 A0 A0 A0 A0
10: A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 01 00
20: 00 00 82 26 02 54 45 53 54 32 A0 A0 A0 A0 A0 A0
30: A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 01 00
40: 00 00 82 26 04 54 45 53 54 33 A0 A0 A0 A0 A0 A0
50: A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 05 00
60: 00 00 82 26 0B 54 45 53 54 34 A0 A0 A0 A0 A0 A0
70: A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 09 00
80: 00 00 82 26 14 54 45 53 54 35 A0 A0 A0 A0 A0 A0
90: A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 0C 00
A0: 00 00 82 28 00 54 45 53 54 36 A0 A0 A0 A0 A0 A0
B0: A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 01 00
C0: 00 00 82 28 01 54 45 53 54 37 A0 A0 A0 A0 A0 A0
D0: A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 01 00
E0: 00 00 82 28 02 54 45 53 54 38 A0 A0 A0 A0 A0 A0
F0: A0 A0 A0 A0 A0 00 00 00 00 00 00 00 00 00 01 00
```

The first two bytes of the directory sector ($27/$02) indicate the location of the next track/sector of the directory (39/2). If the track is set to $00, then it is the last sector of the directory.

When the directory is done, the track value will be $00. The sector link should contain a value of $FF, meaning the whole sector is allocated, but the actual value doesn't matter. The drive will return all the available entries anyways. This is a breakdown of a standard directory sector:

Bytes	**Description**
$00-$1F	First directory entry
$20-$3F	Second dir entry
$40-$5F	Third dir entry
$60-$7F	Fourth dir entry
$80-$9F	Fifth dir entry
$A0-$BF	Sixth dir entry
$C0-$DF	Seventh dir entry
$E0-$FF	Eighth dir entry

This is a breakdown of a standard directory entry:

Bytes	Description
$00-$01	Track/Sector location of next directory sector ($00 $00 if not the first entry in the sector)
$02	File type
$03-$04	Track/sector location of first sector of file
$05-$14	16 character filename (in PETASCII, padded with $A0)
$15-$16	Track/Sector location of first side-sector block (REL file only)
$17	REL file record length (REL file only, max. value 254)
$18-$1D	Unused
$1E-$1F	File size in sectors, low/high byte order ($1E+$1F*256). The approx. filesize in bytes is <= #sectors * 254

The file type field is used as follows:

Bits	Description
0-3	The actual file type
4	Unused
5	Used only during SAVE-@ replacement
6	Locked flag (Set produces ">" locked files)
7	Closed flag (Not set produces "*", or "splat" files)

The actual file type can be one of the following:

Binary	Decimal	File type
0000	0	DEL
0001	1	SEQ
0010	2	PRG
0011	3	USR
0100	4	REL

Values 5-15 are illegal, but if used will produce very strange results.

17.10.1 Non-Standard & Long Directories

Most Commdore floppy disk drives use a single dedicated directory track where all filenames are stored. This limits the number of files stored on a disk based on the number of sectors on the directory track. There are some disk images that contain more files than would normally be allowed. This requires extending the directory off the default directory track by changing the last directory sector pointer to a new track, allocating the new sectors in the BAM, and manually placing (or moving existing) file entries there. The directory of an extended disk can be read and the files that reside there can be loaded without problems on a real drive. However, this is still a very dangerous practice as writing to the extended portion of the directory will cause directory corruption in the non-extended part. Many of the floppy drives core ROM routines ignore the track value that the directory is on and assume the default directory track for operations.

17.10.2 BAM layout

The BAM only occupies up to four sectors on track 38, so the rest of the track is empty and is available for file storage. Below is a dump of the first BAM block, 38/0. A D82 will contain four BAM sectors, 38/0, 38/3, 38/6 and 38/9. Each entry takes 5 bytes, 1 for the free count on that track, and 4 for the BAM bits.

```
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
```

```
    -------------------------------------------
00: 26 03 43 00 01 33 1D FF FF FF 1F 1D FF FF FF 1F
10: 1D FF FF FF 1F 1D FF FF FF 1F 1D FF FF FF 1F 1D
20: FF FF FF 1F 1D FF FF FF 1F 1D FF FF FF 1F 1D FF
30: FF FF 1F 1D FF FF FF 1F 1D FF FF FF 1F 1D FF FF
40: FF 1F 1D FF FF FF 1F 1D FF FF FF 1F 1D FF FF FF
50: 1F 1D FF FF FF 1F 1D FF FF FF 1F 1D FF FF FF 1F
60: 1D FF FF FF 1F 1D FF FF FF 1F 1D FF FF FF 1F 1D
70: FF FF FF 1F 1D FF FF FF 1F 1D FF FF FF 1F 1D FF
80: FF FF 1F 1D FF FF FF 1F 1D FF FF FF 1F 1D FF FF
90: FF 1F 1D FF FF FF 1F 1D FF FF FF 1F 1D FF FF FF
A0: 1F 1D FF FF FF 1F 1D FF FF FF 1F 1D FF FF FF 1F
B0: 1D FF FF FF 1F 1D FF FF FF 1F 1D FF FF FF 1F 1B
C0: F6 FF FF 1F 1B FC FF FF 1F 1B FF FF FF 07 1B FF
D0: FF FF 07 1B FF FF FF 07 1B FF FF FF 07 1B FF FF
E0: FF 07 1B FF FF FF 07 1B FF FF FF 07 1B FF FF FF
F0: 07 1B FF FF FF 07 1B FF FF FF 07 1B FF FF FF 07
```

Bytes	**Description**
$00-$01	T/S pointer to second BAM sector (38/3)
$02	DOS version byte (0x43='C')
$03	Reserved
$04	Lowest track covered by this BAM (0x01=1)
$05	Highest+1 track covered by this BAM (0x33=51)
$06-$0A	BAM for track 1. The first byte shows the "blocks free" for this track, the remaining 4 show the BAM for the track.
$0B-$0F	BAM for track 2
...	...
$FB-$FF	BAM for track 50

Being bit-based, the BAM entries need some explanation. The first track entry in the above BAM sector is at offset 06, "1D FF FF FF 1F". The first number is how many blocks are free on this track ($1D=29) and the remainder is the bit representation of the usage map for the track. These entries must be viewed in binary to make any sense. First convert the values to binary:

```
FF=11111111, FF=11111111, FF=11111111, 1F=00011111
```

In order to make any sense from the binary notation, flip the bits around.

```
           111111 11112222 222222
01234567 89012345 67890123 456789...
------------------------- ---------
11111111 11111111 11111111 11111000
^                               ^
sector 0                   sector 28
```

Since we are on the first track, we have 29 sectors, and only use up to the bit 28 position. If a bit is on (1), the sector is free. Therefore, track 1 is clean, all sectors are free. Any leftover bits that refer to sectors that don't exist, like bits 29-31 in the above example, are set to allocated.

Second BAM block 38/3

```
    00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
    -----------------------------------------------
00: 26 06 43 00 33 65 1B FF FF FF 07 1B FF FF FF 07
10: 1B FF FF FF 07 19 FF FF FF 01 19 FF FF FF 01 19
20: FF FF FF 01 19 FF FF FF 01 19 FF FF FF 01 19 FF
30: FF FF 01 19 FF FF FF 01 19 FF FF FF 01 19 FF FF
40: FF 01 19 FF FF FF 01 19 FF FF FF 01 17 FF FF 7F
50: 00 17 FF FF 7F 00 17 FF FF 7F 00 17 FF FF 7F 00
60: 17 FF FF 7F 00 17 FF FF 7F 00 17 FF FF 7F 00 17
70: FF FF 7F 00 17 FF FF 7F 00 17 FF FF 7F 00 17 FF
80: FF 7F 00 17 FF FF 7F 00 17 FF FF 7F 00 1D FF FF
90: FF 1F 1D FF FF FF 1F 1D FF FF FF 1F 1D FF FF FF
A0: 1F 1D FF FF FF 1F 1D FF FF FF 1F 1D FF FF FF 1F
B0: 1D FF FF FF 1F 1D FF FF FF 1F 1D FF FF FF 1F 1D
C0: FF FF FF 1F 1D FF FF FF 1F 1D FF FF FF 1F 1D FF
D0: FF FF 1F 1D FF FF FF 1F 1D FF FF FF 1F 1D FF FF
E0: FF 1F 1D FF FF FF 1F 1D FF FF FF 1F 1D FF FF FF
F0: 1F 1D FF FF FF 1F 1D FF FF FF 1F 1D FF FF FF 1F
```

Bytes	**Description**
$00-$01	T/S pointer to third BAM sector (38/6)
$02	DOS version byte (0x43='C')
$03	Reserved
$04	Lowest track covered by this BAM (0x33=51)
$05	Highest+1 track covered by this BAM (0x65=101)
$06-$0A	BAM for track 51. The first byte shows the "blocks free" for this track, the remaining 4 show the BAM for the track.
$0B-$0F	BAM for track 52
...	...
$FB-$FF	BAM for track 100

Third BAM block 38/6

```
    00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
    -----------------------------------------------
00: 26 09 43 00 65 97 1D FF FF FF 1F 1D FF FF FF 1F
10: 1D FF FF FF 1F 1D FF FF FF 1F 1D FF FF FF 1F 1D
20: FF FF FF 1F 1D FF FF FF 1F 1D FF FF FF 1F 1D FF
30: FF FF 1F 1D FF FF FF 1F 1D FF FF FF 1F 1D FF FF
40: FF 1F 1D FF FF FF 1F 1D FF FF FF 1F 1D FF FF FF
50: 1F 1D FF FF FF 1F 1B FF FF FF 07 1B FF FF FF 07
60: 1B FF FF FF 07 1B FF FF FF 07 1B FF FF FF 07 1B
70: FF FF FF 07 1B FF FF FF 07 1B FF FF FF 07 1B FF
80: FF FF 07 1B FF FF FF 07 1B FF FF FF 07 1B FF FF
90: FF 07 1B FF FF FF 07 1B FF FF FF 07 19 FF FF FF
A0: 01 19 FF FF FF 01 19 FF FF FF 01 19 FF FF FF 01
B0: 19 FF FF FF 01 19 FF FF FF 01 19 FF FF FF 01 19
C0: FF FF FF 01 19 FF FF FF 01 19 FF FF FF 01 19 FF
```

```
D0: FF FF 01 17 FF FF 7F 00 17 FF FF 7F 00 17 FF FF
E0: 7F 00 17 FF FF 7F 00 17 FF FF 7F 00 17 FF FF 7F
F0: 00 17 FF FF 7F 00 17 FF FF 7F 00 17 FF FF 7F 00
```

Bytes	Description
$00-$01	T/S pointer to fourth BAM sector (38/9)
$02	DOS version byte (0x43='C')
$03	Reserved
$04	Lowest track covered by this BAM (0x65=101)
$05	Highest+1 track covered by this BAM (0x97=151)
$06-$0A	BAM for track 101. The first byte shows the "blocks free" for this track, the remaining 4 show the BAM for the track.
$0B-$0F	BAM for track 102
...	...
$FB-$FF	BAM for track 150

Fourth BAM block 38/9

```
    00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
    -----------------------------------------------
00: 27 01 43 00 97 9B 17 FF FF 7F 00 17 FF FF 7F 00
10: 17 FF FF 7F 00 17 FF FF 7F 00 00 00 00 00 00 00
20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Bytes	Description
$00-$01	T/S pointer to first directory sector (39/1)
$02	DOS version byte (0x43='C')
$03	Reserved
$04	Lowest track covered by this BAM (0x97=151)
$05	Highest+1 track covered by this BAM (0x9B=155)
$06-$0A	BAM for track 151. The first byte shows the "blocks free" for this track, the remaining 4 show the BAM for the track.
$0B-$0F	BAM for track 152
...	...
$15-$19	BAM for track 154
$1A-$FF	Not used

17.11 The D90 disk image format

The D90 image is bit-for-bit copy of the hard drives in the D9090 and D9060. The specifications are as follows:

Specifications	D9060	D9090
Hard Disk Mechanism	Tandom TM602S	Tandom TM603S
Cylinders	153	153
Heads	4	6
Sectors	32	32
Sector Size	256	256
Disk Interface	ST-506	ST-506

Unformatted Capacity	6.4 MB	9.6 MB
Formatted Capacity	5.01 MB	7.52 MB
Available Physical Sectors	19584	29376
Usable CBM Blocks	19441	29162

Note that the unformatted capacity is simply the total raw storage and makes no accommodation for the management of the information. A disk needs to be low-level formatted to include sync bits, sector identifiers, error checking, and correcting bits to reliably find and hold data. This results in an overall loss of about 22% of space, which wasn't unusual for ST-506 MFM hard disks at the time. Modern drives use more advanced storage techniques to reduce this overhead. Earlier published information stating these drives didn't use their full capacity was incorrect.

It can be seen that the only difference between this two mechanisms was the number of heads. In fact the only difference between the two D9090/60 models is the hard drive mechanism and the setting of the J14 configuration jumper which reflects which head count to use. Jumper J13 is connected to the FDC, but in the latest ROM it has no function.

The slight difference between the physical sectors and available blocks is due to the overhead of the CBM filing system (BAM, directory sectors, and bad block list).

The D9090/60 hard disk is the only Commodore drive which uses a "track 0". Sector 0 of this track contains the track and sector information for the header, directory, beginning of the BAM, and the beginning of the bad sector list.

Below is a dump of the configuration sector (track 0, sector 0):

```
    00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
    -----------------------------------------------
00: 00 01 00 FF 4C 0A 4C 14 01 00 49 44 00 00 00 00
10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
...
```

Bytes	**Description**
$00-$01	T/S pointer to bad blocks list
$02	Always 0x00
$03	Always 0xff (DOS version?)
$04-$05	T/S pointer to first directory entry
$06-$07	T/S pointer to header
$08-$08	T/S pointer to first BAM sector
$0A-$0B	Disk ID, "ID" in this example
$0C-$FF	Always 0x00

This sector is the same for both the D9090 and D9060 as the two hard drives had the same number of cylinders. The the header and directory are placed in the middle of the disk (track 76) to minimize access times. The on-board DOS was designed to accommodate for different cylinder configurations so it was easy to install larger disks by only modifying the FDC ROM. The number of sectors can not exceed 32, and the number of heads can not exceed 7 as they would cause unhandled numeric overflow in the DOS. This variation in configuration values makes it difficult to determine them based solely on the D90 file size. At this point only the stock image sizes of 5,013,504, and 7,520,256 will be attachable in VICE.

Regardless of the size, access to the drive is translated from CHS (cylinder, head, sector) to LBA (logical block address) in the FDC. Although the drive mechanism interface is ST-506, the AM2910 bridges the SASI communication from the FDC.

Since most hard disks of that era had some type of physical defect, and no defect management system, a bad blocks list is included in the DOS. This list begins at track 0, sector 1 as seen from the configuration sector. Below is a dump of an empty bad blocks list (track 0, sector 1):

```
    00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
    -----------------------------------------------
00: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
...
```

The list begins as a typical T/S linking sector, however, it is terminated with a T/S of $FF/$FF unlike the typical $00/$xx normally seen in other standard files. The following is a more practical example:

```
    00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
    -----------------------------------------------
00: 00 02 10 20 FF 02 30 40 FF 08 50 60 FF 14 70 80
...
```

The actual information starts at offset $02. "Bad" blocks are simply marked as "allocated" in the BAM so the DOS doesn't try to put any new data in them. This file is processed by setting the "working" track to 0. Starting at the beginning (offset $02), a byte is read. If this value is not $FF, it is interpreted as the "working" sector. The "working" track/sector is marked as bad (used). We continue on by reading the next byte as the "working" sector and marking it until a $FF is encountered. In the above example, track $0/$10 and $0/$20 would be marked as bad. Once the $FF is found, the next byte is the new "working" track if it is not $FF; if it is, the list is finished. Otherwise we continue processing the list just as we did when the "working" track was 0. In the above example, track $2/$30, $2/$40, $8/$50, $8/$60, $14/70, and $14/$80 would be marked as bad. Note that the list in this example would continue onto track 0, sector 2 at offset $2 as the T/S link is set. This list is processed during a "validation" as the current BAM is erased and rebuilt. It also appears to be processed after a format, but it is not certain if the DOS generates the list or if it was generated in factory by a diagnostic system prior to the drives distribution.

The BAM (Block Availability Map) generally begins on track 1, sector 0, and continues on as necessary based on the disk configuration. Below is a dump of that first BAM sector with a configuration of 32 sectors per head, and 6 heads:

```
    00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
    -----------------------------------------------
00: 09 00 FF FF 00 08 00 00 00 00 00 00 00 00 00 00
10: 1E FC FF FF FF 20 FF FF FF FF 20 FF FF FF FF 20
20: FF FF FF FF 20 FF FF FF FF 20 FF FF FF FF 1F FE
30: FF FF FF 20 FF FF FF FF 20 FF FF FF FF 20 FF FF
...
F0: FF 20 FF FF FF FF 20 FF FF FF FF 20 FF FF FF FF
```

Bytes	**Description**
$00-$01	T/S pointer to next BAM sector ($9/$0)

$02-$03	T/S pointer to previous BAM sector ($FF/$FF)
$04	Lowest track covered by this BAM ($0)
$05	Highest+1 track covered by this BAM ($8)
$06	Possible start of bitmap data
$10-$14 (first entry based on this example)	BAM for track 0, head 0. The first byte shows the "blocks free" for this head, the remaining 4 show the BAM for this track and head.
$24-$08	BAM for track 0, head 1.
...	...
$2E-$32	BAM for track 1, head 0.
...	...
$FB-$FF	BAM for track 7, head 5.

If the T/S pointer is $FF/$FF, it means the end of the BAM. So in this case, there is no previous BAM sector, whereas in the last BAM sector, bytes $01-$02 will be $FF/$FF indicating that there is no next BAM sector.

Bytes $4 and $5 show the range of which tracks this BAM sector applies to. There are 250 bytes to hold this information. The size of each bitmap entry is calculated by "ceiling(sectors_per_head/8)+1"; the "1" is to include the block count. This value is then multipled by the head number and divided into 250. The remainder is added to 6 (the offset for the start of bitmap data). For this example, the bitmap entry size would be "ceiling(32/8)+1" which is 5 bytes. The offset is "(250%(5*6))+6" which is 16 or $10. The goal of this calculation is to place the maximum number of whole tracks inside each BAM. In this case, "int(250/(5*6))" is 8, so we can store 8 tracks per BAM sector. Therefore the lowest track is 0, and the highest is 0 + 8. The next BAM sector will handle tracks 8 to 15 (16 - 1). The position of the BAMs will start at track 1, and the second will be at track 9 (8 + 1), the third will be at track 17 (2 * 8 + 1), and so on. For this example with 153 tracks, we will need at least 20 (153 divided by 8) BAM sectors. BAM sectors are always placed on sector 0. The header and directory will never start on sector 0, so we don't have to worry if one of the BAM sectors will end up on track 76 for example.

Being bit-based, the BAM entries need some explanation. The first track (track 0) entry in the above BAM sector is at offset $10, "1E FC FF FF FF". The first number is how many blocks are free on this track and head ($1E=30) and the remainder is the bit representation of the usage map for the track. These entries must be viewed in binary to make any sense. First convert the values to binary:

```
FC=11111100, FF=11111111, FF=11111111, FF=11111111
```

In order to make any sense from the binary notation, flip the bits around.

```
           111111 11112222 22222233
01234567 89012345 67890123 45678901
------------------------- ---------
00111111 11111111 11111111 11111111
^                                 ^
sector 0                  sector 31
```

If a bit is on (1), the sector is free. We can see all but sectors 0 and 1 are free here as it holds the configuration sector, and one sector for the bad block list respectively.

At offset $24, "1F FE FF FF FF" describes the allocation at track 1, sector 0. The first value is the number of free blocks ($1F=31) and the next 4 indicate which sectors are used (or available); in this case sector 0 is allocated as it is used by the first BAM sector.

Below is the dump of the last BAM sector (track 152, sector 0):

```
    00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
    -----------------------------------------------
00: FF FF 91 00 98 99 00 00 00 00 00 00 00 00 00 00
10: 1F FE FF FF FF 20 FF FF FF FF 20 FF FF FF FF 20
20: FF FF FF FF 20 FF FF FF FF 20 FF FF FF FF 00 00
30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
...
```

Here we can see it only covers tracks 152 ($98) to 152 ($99-1). So only 6 entries are valid, and the rest are all zeros. The next T/S pointer is $FF/$FF indicating this is the last BAM sector.

When the blocks free is calculated, only track 0 is excluded. Other Commodore drives exclude the dedicated directory track.

The directory is generally on track 76 (152 divided by 2), with sector 20 containing the header (disk name, disk ID's). The directory entries begin at sector 10. The drives interleave is 10 sectors and it appears the directory entry is allocated before the header on a format. Once the directory grows, the interleave becomes 3, as on most Commodore drives. Unlike most Commodore drives, the directory is not limited to track 76, nor is file data limited to be else where. Once track 76 is filled with directory entries, it will look in other tracks to place more entries. Filling the disk to capacity will also place data into track 76. Be warned though as the directory listing can exceed the amount of memory available on the Commodore machine which is "loading" the directory. It is wise to use a DOS wedge which displays the directory as it is being read but doesn't store it in memory.

The D9090/60 emulation in VICE was tested to see how many one block files could be created on a D9090 image. The disk was filled with 25922 files which required 3241 blocks for the directory entries.

Below is a dump of the header at track 76, sector 20:

```
    00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
    -----------------------------------------------
00: 4C 0A 00 00 00 00 54 45 53 54 A0 A0 A0 A0 A0 A0
10: A0 A0 A0 A0 A0 A0 A0 A0 49 44 A0 33 41 A0 A0 A0
20: A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
...
F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

This is very similar to the D80/D82 header in placement but without the DOS version value. Also note that the dos version bytes are "3A". For the D90, it is believed that the $FF at offset $3 in the configuration sector dictates the DOS version.

The directory entry generally starts at track 76, sector 10. It has the identical layout as all other Commodore drives. There are no files specific to the D90 image other than super side sectors for larger REL files.

17.12 The DHD disk image format

The DHD image is bit-for-bit copy of a Creative Micro Designs (CMD) Hard Drive (HD). CMD designed their DOS so that it could co-exist with other operating systems from other computer platforms. At the time, SCSI HDs were very expensive, making this a likely scenario for some. In this section, a block refers to a SCSI HD data block which is 512 bytes. A SCSI HD refers to the location of a blocks on the disk as logical block address (LBA). A sector refers to a commodore disk sector which is 256 bytes. SCSI, being a smarter disk system (compared to earlier ones), translates the LBAs into physical head, sector, cylinder coordinates internally thus simplifying the host's interface. So, for example, a 40 MB HD of that era would have had 82332 (or $1419C) LBAs. As HD technology improved, the number of sectors per cylinder became variable depending on the cylinder which allowed for greater capacity. The host therefore has no information about the HD geometry other than the total number of LBAs. Most operating systems place their partition tables or boot code near the beginning of the HD. Because of this variability, CMD designed their DOS to begin at any location in increments of 128 blocks. It was therefore possible to configure foreign partitions on various operating system so that the CMD data was ignored. The starting location of the CMD whole disk partition will be referred to as X blocks.

Upon reset, the boot ROM looks for the CMD whole partition by examining LBA X+2 (starting at X=0) and checking for a special signature in the block. If it does not find it, X will increase by 128, and it will try again. It will do this until it finds the signature, or until it gets a read error from the HD signaling the end of disk space. The device will the switch to installation mode.

The following is an example configuration block of a 40 MB drive:

```
X *     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F        ASCII
$200 +  -----------------------------------------------   ----------------
000400: 94 6C 00 0E 7D E7 00 00 00 00 00 00 00 00 00 00
000410: 20 20 20 20 31 2E 39 32 30 33 2F 32 32 2F 39 36   ....1.9203/22/96
000420: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000430: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000440: 03 02 00 7E C9 55 00 00 00 00 00 00 00 00 00 00
000450: 20 20 20 20 32 2E 30 30 30 33 2F 32 32 2F 39 36   ....2.0003/22/96
000460: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
...
000500: 00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000510: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000530: FF FF FF FF FF FF FF FF 00 01 FF FF FF FF FF FF
000540: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000570: 00 41 FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000580: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0005A0: FF FF FF FF FF FF FF FF 00 9C FF FF FF FF FF FF
0005B0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
0005E0: 00 0C 01 01 0C 80 01 00 00 00 00 00 00 00 00 00
0005F0: 43 4D 44 20 48 44 20 20 8D 03 88 8E 02 88 EA 60   CMD HD.........
...
000E00: < start of first OS, n=0, CMD DOS, see $402-$403 >
007E00: < start of second OS, n=1, GEOS overlay, see $442-$443 >
```

```
010000: < start of partition table, see $5E6-$5E7 >
```

The first 256 bytes of the configuration block appear to be a list of loadable operating systems for the drive. Each entry, "n", uses 64 bytes, so there can be at most 4. The first is the standard CMD DOS, while the second is an overlay for GEOS. The boot ROM explicitly loads the first (n=0).

The CMD DOS supports up to 56 connected HDs: 7 devices with 8 logical units (LUNs) each. For that era, most HDs only supported a single LUN (0), whereas more exotic array devices would allow for multiple LUNs. The relative location of the disk, "d", in the whole drive map, is stored in the next 224 bytes. For example, the first disk would start at location 0. The next disk following the first, would start at LBA $01419C (the size of the first disk plus its starting point in the map). The 24-bit values used here for the placement are adequate since a fully populated CMD HD can only access 255 16 MB partitions, which his just under 4 GiB.

The last portion of the configuration block holds the device parameters (device number, partition location, default partition) as well as the CMD HD signature.

Bytes	**Description**
(X*$200)+($400-$4FF)	Operating System Table
(X*$200+$400+n*$40)+$00	Destination memory page
(X*$200+$400+n*$40)+$01	Number of pages
(X*$200+$400+n*$40)+$02-$03	Location of data offset from X in sectors (256 bytes); MSB format
(X*$200+$400+n*$40)+$04-$05	Checksum of data; MSB format
(X*$200+$400+n*$40)+$06-$0F	All 0x00
(X*$200+$400+n*$40)+$10-$17	ASCII of version number
(X*$200+$400+n*$40)+$18-$1F	ASCII of date
(X*$200+$400+n*$40)+$20-$3F	All 0x00
(X*$200+$500+d*$38)+$00	SCSI ID of HD in bits 7-4, LUN in bits 3-0; $FF means not present
(X*$200+$500+d*$38)+$38	MSB of disk location in drive map
(X*$200+$500+d*$38)+$70	Middle byte of disk location in drive map
(X*$200+$500+d*$38)+$A8	LSB of disk location in drive map
(X*$200+$500)+$E0	0x00 (unknown)
(X*$200+$500)+$E1	Device number (12 or 0x0C by default)
(X*$200+$500)+$E2	0x01 (unknown)
(X*$200+$500)+$E3	0x01 (unknown)
(X*$200+$500)+$E4	Device number (12 or 0x0C by default)
(X*$200+$500)+$E5	0x80: 2 to the power of the SCSI ID of host (which is always 7)
(X*$200+$500)+$E6-$E7	Location of CMD partition table offset from X in sectors (256 bytes); MSB format
(X*$200+$500)+$E8	Default partition number
(X*$200+$500)+$E9-$EF	All 0x00
(X*$200+$500)+$F0-$F7	0x43 0x4D 0x44 0x20 0x48 0x44 0x20 0x20: First half of signature

(X*$200+$500)+$F8-$FF	0x8D 0x03 0x88 0x8E 0x02 0x88 0xEA 0x60: Second half of signature; decodes to STA $8803, STX $8802, NOP, RTS

The CMD partition table is a special disk image with one track and 32 sectors. It is very similar the standard directory structure of most CBM drives. The first two bytes of the sector indicate the location of the next track/sector of the partition table; it is usually in sequence. If the next track is set to $00 (the next sector will be $ff), then it is the last sector of the partition table.

The standard partition table sectors:

Bytes	**Description**
$00-$1F	First partition entry for this sector
$20-$3F	Second partition entry for this sector
$40-$5F	Third partition entry for this sector
$60-$7F	Fourth partition entry for this sector
$80-$9F	Fifth partition entry for this sector
$A0-$BF	Sixth partition entry for this sector
$C0-$DF	Seventh partition entry for this sector
$E0-$FF	Eighth partition entry for this sector

The standard partition entry:

Bytes	**Description**
$00-$01	Track/Sector location of next partition sector ($00 $00 if not the first entry in the sector)
$02	Partition type: 0x00=none, 0x01=native, 0x02=1541, 0x03=1571, 0x04=1581, 0x05=1581CPM, 0x06=Print Queue, 0x07=Foreign, 0xFF=System. Only partition 0 can be a system type, and type 6 isn't known to be implemented on any CMD DOS.
$03-$04	0x00 0x00
$05-$14	16 character partition name (in PETASCII, padded with $A0)
$15-$17	Location of the partition data offset from X in sectors (256 bytes); MSB format; partition 0, the system partition always starts a 0.
$18-$1D	All 0x00
$1E-$1F	Size of partition (in blocks) in MSB format. Partition 0, the system partition is always 0x90 blocks, which includes blocks from X to the end of the partition table.

When a partition is deleted, its entry is removed and all others stay in their existing place. The partition data following the deleted partition is moved back to fill in the gap from the deletion; this can be very time consuming. So, when deleting a partition, it is suggested to start with the one with the highest start LBA.

The data at the starting LBA is simply the associated disk image of the partition type. It follows the formats (D64, D71, and D81) documented previously.

17.13 The P00 image format

(This section was contributed by Peter Schepers and slightly edited by Marco van den Heuvel.)

These files were created for use in the PC64 emulator, written by Wolfgang Lorenz. Each one has the same layout with the filetype being stored in the DOS extension (i.e. Pxx is a PRG, Sxx is a SEQ, Uxx is a USR and Rxx is a RELative file), and the header is only 26 bytes long.

This is a dump of a Pxx file (PRG)...

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
      -----------------------------------------------
0000: 43 36 34 46 69 6C 65 00 43 52 49 53 49 53 20 4D
0010: 4F 55 4E 54 41 49 4E 00 00 00
```

Bytes	Description
$00-$06	ASCII string "C64File"
$07	Always $00
$08-$17	Filename in PETASCII, padded with $00 (not $A0, like a D64)
$18	Always $00
$19	REL file record size ($00 if not a REL file)
$1A-??	Program data

The 'xx' in the extension of the file is usually 00, except when we have two DOS filenames which would be the same, but the C64 filenames are different! If we have two C64 filenames which are the same, they *cannot* co-exist in the same directory. If we have two files which do convert down to be the same DOS filename, the extension is incremented until an unused one is found (x01, x02, x03, up to x99). We can have up to 99 different C64 files with the same corresponding DOS names as that's all the extension will hold (from P00 to P99).

Each PC64 file only has one entry, there are no multi-file containers allowed. This could result in a large number of these files in a directory, even for only a few programs, as each C64 file will result in a PC64 file entry. The best use for a PC64 file is a single-file program, one which does not load anything else.

17.14 The CRT cartridge image format

This chapter is based on CRT.txt (rev1.14) compiled by Peter Schepers, with additional contributions from Per Hakan Sundell, Markus Brenner, Marco Van Den Heuvel, Groepaz.

Cartridge files were introduced in the CCS64 emulator, written by Per Hakan Sundell, and use the ".CRT" file extension. This format was created to handle the various ROM cartridges that exist, such as Action Replay, the Power cartridge, and the Final Cartridge.

Normal game cartridges can load into several different memory ranges ($8000-9FFF, $A000-BFFF or $E000-FFFF). Newer utility and freezer cartridges were less intrusive, hiding themselves until called upon, and still others used bank-switching techniques to allow much larger ROM's than normal. Because of these "stealthing" and bank-switching methods, a special cartridge format was necessary, to let the emulator know where the cartridge should reside, the control line states to enable it and any special hardware features it uses.

17.14.1 Header contents

Here is a dump of a sample 8KiB normal cartridge, "Attack Of The Mutant Camels"...

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F        ASCII
```

```
     ------------------------------------------------   ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20   C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 00 00 01 00 00 00 00 00 00   ...@............
0020: 41 54 54 41 43 4B 20 4F 46 20 54 48 45 20 4D 55   ATTACK OF THE MU
0030: 54 41 4E 54 20 43 41 4D 45 4C 53 00 00 00 00 00   TANT CAMELS.....
0040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00   CHIP.. ....... .
0050: D3 9B BC FE C3 C2 CD 38 30 EA EA EA A9 01 85 13   .......80.......
0060: 4C B3 9B A9 08 85 5A 88 D0 FD C6 5A D0 F9 60 D0   L.....Z....Z..‘.
```

```
  Bytes:$0000-000F - 16-byte cartridge signature "C64 CARTRIDGE" (padded
                     with space characters)
        0010-0013 - File header length ($00000040, in high/low format,
                     calculated from offset $0000). The default (also the
                     minimum) value is $40. Some cartridges exist which
                     show a value of $00000020 which is wrong.
        0014-0015 - Cartridge version (high/low, presently 01.01)
        0016-0017 - Cartridge hardware type ($0000, high/low)
                      0 - Normal cartridge
                      1 - Action Replay
                      2 - KCS Power Cartridge
                      3 - Final Cartridge III
                      4 - Simons' BASIC
                      5 - Ocean type 1*
                      6 - Expert Cartridge
                      7 - Fun Play, Power Play
                      8 - Super Games
                      9 - Atomic Power
                     10 - Epyx Fastload
                     11 - Westermann Learning
                     12 - Rex Utility
                     13 - Final Cartridge I
                     14 - Magic Formel
                     15 - C64 Game System, System 3
                     16 - Warp Speed
                     17 - Dinamic**
                     18 - Zaxxon, Super Zaxxon (SEGA)
                     19 - Magic Desk, Domark, HES Australia
                     20 - Super Snapshot V5
                     21 - Comal-80
                     22 - Structured BASIC
                     23 - Ross
                     24 - Dela EP64
                     25 - Dela EP7x8
                     26 - Dela EP256
                     27 - Rex EP256
                     28 - Mikro Assembler
                     29 - Final Cartridge Plus
```

```
        30 - Action Replay 4
        31 - Stardos
        32 - EasyFlash
        33 - EasyFlash Xbank
        34 - Capture
        35 - Action Replay 3
        36 - Retro Replay
        37 - MMC64
        38 - MMC Replay
        39 - IDE64
        40 - Super Snapshot V4
        41 - IEEE-488
        42 - Game Killer
        43 - Prophet64
        44 - EXOS
        45 - Freeze Frame
        46 - Freeze Machine
        47 - Snapshot64
        48 - Super Explode V5.0
        49 - Magic Voice
        50 - Action Replay 2
        51 - MACH 5
        52 - Diashow-Maker
        53 - Pagefox
        54 - Kingsoft
        55 - Silverrock 128K Cartridge
        56 - Formel 64
        57 - RGCD
        58 - RR-Net MK3
        59 - EasyCalc
        60 - GMod2
        61 - MAX Basic
        62 - GMod3
        63 - ZIPP-CODE 48
        64 - Blackbox V8
        65 - Blackbox V3
        66 - Blackbox V4
        67 - REX RAM-Floppy
        68 - BIS-Plus
        69 - SD-BOX
        70 - MultiMAX
        71 - Blackbox V9
        72 - Lt. Kernal Host Adaptor
        73 - RAMLink
        74 - H.E.R.O.
0018 - Cartridge port EXROM line status
        0 - active (lo)
```

```
                1 - inactive (hi)
          0019 - Cartridge port GAME line status
                0 - active (lo)
                1 - inactive (hi)
          001A - Cartridge Hardware Revision/Subtype (usually 0)
     001B-001F - Reserved for future use
     0020-003F - 32-byte cartridge name "CCSMON" (uppercase, padded
                 with null characters)
     0040-xxxx - Cartridge contents (called CHIP PACKETS, as there can
                 be more than one per CRT file). See below for a
                 breakdown of the CHIP format.
```

(*Note: Ocean type 1 includes Navy Seals, Robocop 2 & 3, Shadow of the Beast, Toki, Terminator 2 and more)

(**Note: Dinamic includes Narco Police and more)

17.14.2 CHIP Contents

The following is the contents of the CHIP packet, from position $0040 on in the CRT file. Note I have re-adjusted the starting address to be $0000, since we are now looking at a file contained in the .CRT file, and all size references are from where it starts.

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
0010: D3 9B BC FE C3 C2 CD 38 30 EA EA EA A9 01 85 13  .......80.......
0020: 4C B3 9B A9 08 85 5A 88 D0 FD C6 5A D0 F9 60 D0  L.....Z....Z..‘.
0030: F2 60 A9 04 85 49 A9 00 85 48 A2 00 A5 48 9D 40  .‘...I...H...H.@
0040: 03 A5 49 9D 60 03 A5 48 18 69 28 85 48 A5 49 69  ..I.‘..H.i(.H.Ii
0050: 00 85 49 E8 E0 18 D0 E4 60 A6 03 A4 02 BD 40 03  ..I.....‘.....@.
```

```
  Bytes:$0000-0003 - Contained ROM signature "CHIP" (note there can be more
                     than one image in a .CRT file)
          0004-0007 - Total packet length ($00002010, ROM image size and
                     header combined) (high/low format)
          0008-0009 - Chip type
                      0 - ROM
                      1 - RAM, no ROM data
                      2 - Flash ROM
          000A-000B - Bank number ($0000 - normal cartridge)
          000C-000D - Starting load address (high/low format)
          000E-000F - ROM image size in bytes (high/low format, typically
                     $2000 or $4000)
          0010-xxxx - ROM data
```

The following is a chart taken from the "Commodore Programmers Reference Guide". It details the state of various areas of memory depending on the state of the control lines.

Legend: L - ROML (low) H - ROMH (high) G - GAME E - EXROM

```
Addr       LHGE   LHGE   LHGE   LHGE   LHGE   LHGE   LHGE   LHGE   LHGE
Range
```

```
          1111   101X   1000   011X   001X   1110   0100   1100   XX01
        default               00X0                             Ultimax
-------------------------------------------------------------------------
E000-FFFF Kernal  RAM    RAM   Kernal  RAM   Kernal Kernal Kernal ROMH(*)
D000-DFFF IO/CHR IO/CHR IO/RAM IO/CHR  RAM   IO/CHR IO/CHR IO/CHR   I/O
C000-CFFF  RAM    RAM    RAM    RAM    RAM    RAM    RAM    RAM     -
A000-BFFF BASIC   RAM    RAM    RAM    RAM   BASIC   ROMH   ROMH    -
8000-9FFF  RAM    RAM    RAM    RAM    RAM    ROML   RAM    ROML  ROML(*)
4000-7FFF  RAM    RAM    RAM    RAM    RAM    RAM    RAM    RAM     -
1000-3FFF  RAM    RAM    RAM    RAM    RAM    RAM    RAM    RAM     -
0000-0FFF  RAM    RAM    RAM    RAM    RAM    RAM    RAM    RAM    RAM
```

(*) Internal memory does not respond to write accesses in these areas

From the above chart, the following table can be built. It shows standard cartridges, either 8KiB or 16KiB in size, and the memory ranges they load into.

```
Type     Size   Game   EXRom  Low Bank  High Bank
         in K   Line   Line    (ROML)    (ROMH)
-----------------------------------------------
Normal    8KiB    hi     lo     $8000      ----
Normal    16KiB   lo     lo     $8000     $A000
Ultimax   8KiB    lo     hi     $E000      ----
```

The ROMH and ROML lines are CPU-controlled status lines, used to bank in/out RAM, ROM or I/O, depending on what is needed at the time.

Ultimax cartridges typically are situated in the $E000-FFFF (8KiB) ROM address range. There are some cartridges which only use 4KiB of the 8KiB allocation. If the cartridge is 16KiB in size, then it will reside in both $8000-9FFF and $E000-FFFF.

17.14.3 Cartridge Specifics

17.14.3.1 0 - Normal cartridge

Size	8KiB
EXROM	active (lo) (0)
GAME	inactive (hi) (1)
Load address	$8000-9FFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 00 00 01 00 00 00 00 00 00  ...@............
0020: 41 54 54 41 43 4B 20 4F 46 20 54 48 45 20 4D 55  ATTACK OF THE MU
0030: 54 41 4E 54 20 43 41 4D 45 4C 53 00 00 00 00 00  TANT CAMELS.....
0040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
0050: D3 9B BC FE C3 C2 CD 38 30 EA EA EA A9 01 85 13  .......80.......
```

The second sample below is a dump of "Music Machine", a 4KiB ULTIMAX mode cartridge. It is still identified as a "standard cartridge" according to the ID.

Normal cartridge

Size	4KiB (ULTIMAX mode)

EXROM inactive (hi) (1)
GAME active (lo) (0)
Load address $F000-F7FF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 00 01 00 00 00 00 00 00 00  ...@............
0020: 4D 55 53 49 43 20 4D 41 43 48 49 4E 45 00 00 00  MUSIC MACHINE...
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 10 10 00 00 00 00 F0 00 10 00  CHIP............
0050: 3C 66 C3 C3 66 3C FF FF 18 3C 66 7E 66 66 66 00  <f..f<...<f~fff.
```

The third sample is a dump of "Adventure Creator", a 16KiB standard cartridge.

Normal cartridge

Size 16KiB
EXROM active (lo) (0)
GAME active (lo) (0)
Load address $8000-BFFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 00 00 00 00 00 00 00 00 00  ...@............
0020: 41 64 76 65 6E 74 75 72 65 20 43 72 65 61 74 6F  Adventure Creato
0030: 72 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  r...............
0040: 43 48 49 50 00 00 40 10 00 00 00 00 80 00 40 00  CHIP..@.......@.
0050: 09 80 81 EA C3 C2 CD 38 30 A2 00 78 D8 8E 11 D0  .......80..x....
```

17.14.3.2 1 - Action Replay

Size 32KiB (4 banks of 8KiB each)
EXROM active (lo) (0)
GAME inactive (hi) (1)
Load address $8000-9FFF (all modules)

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 01 00 01 00 00 00 00 00 00  ...@............
0020: 41 63 74 69 6F 6E 20 52 65 70 6C 61 79 20 56 00  Action Replay V.
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
0050: 09 80 0C 80 C3 C2 CD 38 30 4C 60 80 4C 63 80 4C  .......80L`.Lc.L
```

This cart has 32KiB of ROM, and 8KiB of RAM. The bank switching is done by writing to the I/O-1 range as follows:

```
bit  meaning
---  -------
7    extra ROM bank selector (A15) (unused)
6    1 = resets FREEZE-mode (turns back to normal mode)
```

```
5    1 = enable RAM at ROML ($8000-$9FFF) &
         I/O-2 ($DF00-$DFFF = $9F00-$9FFF)
4    ROM bank selector high (A14)
3    ROM bank selector low  (A13)
2    1 = disable cartridge (turn off $DE00)
1    1 = /EXROM high
0    1 = /GAME low
```

Additionally the RAM or ROM can be available through a window in the I/O-2 range.

17.14.3.3 2 - KCS Power Cartridge

Size	16KiB (2 banks of 8KiB each)
EXROM	active (lo) (0)
GAME	active (lo) (0)
Load address	module #1 - $8000-9FFF module #2 - $A000-BFFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 20 01 00 00 02 00 00 00 00 00 00 00 00  ... ............
0020: 4B 43 53 20 50 6F 77 65 72 20 43 61 72 74 72 69  KCS Power Cartri
0030: 64 67 65 00 00 00 00 00 00 00 00 00 00 00 00 00  dge.............
0040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
0050: 09 80 5E FE C3 C2 CD 38 30 78 D8 A2 FF 9A A9 27  ..^....80x.....'
...
2050: 43 48 49 50 00 00 20 10 00 00 00 00 A0 00 20 00  CHIP.. ....... .
2060: 97 E3 16 A1 FF FF FF 20 13 A0 A5 01 09 01 85 01  ....... ........
```

17.14.3.4 3 - Final Cartridge III

Size	64KiB (4 banks of 16KiB each)
EXROM	inactive (hi) (1)
GAME	inactive (hi) (1)
Load address	$8000-BFFF (all modules)

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 03 00 00 00 00 00 00 00 00  ...@............
0020: 46 69 6E 61 6C 20 43 61 72 74 72 69 64 67 65 20  Final Cartridge
0030: 49 49 49 20 31 39 38 37 00 00 00 00 00 00 00 00  III 1987........
0040: 43 48 49 50 00 00 40 10 00 00 00 00 80 00 40 00  CHIP..@.......@.
0050: 09 80 5E FE C3 C2 CD 38 30 4C 4C 80 4C 55 95 4C  ..^....80LL.LU.L
...
4050: 43 48 49 50 00 00 40 10 00 00 00 01 80 00 40 00  CHIP..@.......@.
4060: 01 02 00 81 5D 81 61 81 99 81 D8 81 0B 82 33 82  ....].a.......3.
...
8060: 43 48 49 50 00 00 40 10 00 00 00 02 80 00 40 00  CHIP..@.......@.
8070: 20 43 80 20 52 80 A9 4E 20 05 DE 20 FD BF AD 39   C. R..N .. ...9
...
```

```
C070: 43 48 49 50 00 00 40 10 00 00 00 03 80 00 40 00  CHIP..@.......@.
C080: A2 06 BD DD 85 95 05 CA 10 F8 AE A0 02 E8 EC A2  ................
```

A total of 64 KiB of ROM memory is organized into four $4000 banks located at $8000-$BFFF.

The banks are arranged in the following way:

```
Bank 0:  BASIC, Monitor, Disk-Turbo
Bank 1:  Notepad, BASIC (Menu Bar)
Bank 2:  Desktop, Freezer/Print
Bank 3:  Freezer, Compression
```

The cartridges uses the entire I/O-1 and I/O-2 range. Bank switching is done by writing the bank number plus $40 into memory location $DFFF. For instance, to select bank 2, $DFFF is set to $42.

The CRT file contains four CHIP blocks, each block with a start address of $8000, length $4000 and the bank number in the bank field. In the cartridge header, both EXROM ($18) and GAME ($19) are set to 1 to enable the 16 KiB ROM configuration.

The registers are arranged in the following way:

One register at $DFFF:

```
bit  meaning
---  -------
7      Hide this register (1 = hidden)
6      NMI line  (0 = low = active) *1)
5      GAME line  (0 = low = active) *2)
4      EXROM line (0 = low = active)
2-3    unassigned (usually set to 0)
0-1    number of bank to show at $8000
```

1) if either the freezer button is pressed, or bit 6 is 0, then an NMI is generated

2) if the freezer button is pressed, GAME is also forced low

The rest of I/O-1/I/O-2 contain a mirror of the last 2 pages of the currently selected ROM bank (also at $dfff, contrary to what some other documents say)

17.14.3.5 4 - Simons' Basic

Size	16KiB (2 banks of 8KiB each)
EXROM	active (lo) (0)
GAME	inactive (hi) (1)
Load address	module #1 - $8000-9FFF module #2 - $A000-BFFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 04 00 00 00 00 00 00 00 00  ...@............
0020: 53 69 6D 6F 6E 27 73 20 42 61 73 69 63 00 00 00  Simon's Basic...
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
0050: 52 81 52 81 C3 C2 CD 38 30 41 4C 52 81 20 2C 81  R.R....80ALR. ,.
...
```

```
2050: 43 48 49 50 00 00 20 10 00 00 00 00 A0 00 20 00  CHIP.. ....... .
2060: 20 A4 A6 99 9E CB A0 05 A5 A8 91 20 A4 A6 99 A2   .......... ....
```

Simons' BASIC permanently uses 16 KiB ($4000) bytes of cartridge memory from $8000-$BFFF. However, through some custom bank-switching logic the upper area ($A000-$BFFF) may be disabled so Simons' BASIC may use it as additional RAM. Writing a value of $01 to address location $DE00 banks in ROM, $00 disables ROM and enables RAM.

The CRT file contains two CHIP blocks of length $2000 each, the first block having a start address of $8000, the second block $A000. In the cartridge header, EXROM ($18) is set to 0, GAME ($19) is set to 1 to indicate the RESET/power-up configuration of 8 KiB ROM.

17.14.3.6 5 - Ocean type 1

Size	32KiB, 128KiB, 256KiB or 512KiB sizes (4, 16, 32 or 64 banks of 8KiB)
EXROM	active (lo) (0)
GAME	active (lo) (0)
Load address	Banks 00-15 - $8000-9FFF Banks 16-31 - $A000-BFFF (except Terminator 2)

```
       00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
       -----------------------------------------------  ----------------
00000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
00010: 00 00 00 40 01 00 00 05 00 00 00 00 00 00 00 00  ...@............
00020: 53 48 41 44 4F 57 20 4F 46 20 54 48 45 20 42 45  SHADOW OF THE BE
00030: 41 53 54 00 00 00 00 00 00 00 00 00 00 00 00 00  AST.............
00040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
00050: 09 80 83 81 C3 C2 CD 38 30 4C 83 81 4C 76 82 80  .......80L..Lv..
 ...
02050: 43 48 49 50 00 00 20 10 00 00 00 01 80 00 20 00  CHIP.. ....... .
02060: 59 6D 00 56 AD 00 55 AE F0 00 01 A0 FE 00 01 F8  Ym.V..U.........
 ...
20140: 43 48 49 50 00 00 20 10 00 00 00 10 A0 00 20 00  CHIP.. ....... .
20150: 0A 9A 55 FF 9B 69 57 FE AA 65 96 FE 65 0F D6 D9  ..U..iW..e..e...
```

Here is a list of the known OCEAN cartridges:

```
    Batman The Movie    (128 KiB)
    Battle Command      (128 KiB)
    Double Dragon       (128 KiB)
    Navy Seals          (128 KiB)
    Pang                (128 KiB)
    Robocop 3           (128 KiB)
    Space Gun           (128 KiB)
    Toki                (128 KiB)
    Chase H.Q. II       (256 KiB)
    Robocop 2           (256 KiB)
    Shadow of the Beast (256 KiB)
    Terminator 2        (512 KiB)
```

The 32KiB type of cart has 4 banks of 8KiB ($2000), banked in at $8000-$9FFF.

The 128KiB type of cart has 16 banks of 8KiB ($2000), banked in at $8000-$9FFF.

The 256KiB type of cart has 32 banks of 8KiB ($2000), 16 banked in at $8000-$9FFF, and 16 banked in at $A000-$BFFF.

The 512KiB type of cart has 64 banks of 8KiB ($2000), banked in at $8000-$9FFF.

Bank switching is done by writing to $DE00. The lower six bits give the bank number (ranging from 0-63). Bit 7 in this selection word is always set.

17.14.3.7 6 - Expert Cartridge

Size	8KiB
EXROM	inactive (hi) (1)
GAME	active (lo) (0)
Load address	$8000-9FFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 06 01 00 00 00 00 00 00 00  ...@............
0020: 45 78 70 65 72 74 20 43 61 72 74 72 69 64 67 65  Expert Cartridge
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 40 10 00 02 00 00 80 00 20 00  CHIP..@....... .
0050: 00 00 00 0A F3 00 00 00 00 00 00 00 00 00 00 00  ................
```

17.14.3.8 7 - Fun Play, Power Play

Size	128KiB (16 banks of 8KiB modules)
EXROM	active (lo) (0)
GAME	inactive (hi) (1)
Load address	$8000-9FFF (all modules)

```
       00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
       -----------------------------------------------  ----------------
00000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
00010: 00 00 00 40 01 00 00 07 00 01 00 00 00 00 00 00  ...@............
00020: 46 55 4E 20 50 4C 41 59 00 00 00 00 00 00 00 00  FUN PLAY........
00030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
00050: 1E 80 86 EA C3 C2 CD 38 30 1B 00 81 0D 08 80 00  .......80.......
 ...
02050: 43 48 49 50 00 00 20 10 00 00 00 08 80 00 20 00  CHIP.. ....... .
02060: 78 A2 F0 86 01 BD 1D 08 9D F8 00 CA D0 F7 4C 00  x.............L.
 ...
04060: 43 48 49 50 00 00 20 10 00 00 00 10 80 00 20 00  CHIP.. ....... .
04070: 38 E5 68 85 03 B0 11 27 03 12 C0 18 69 27 42 90  8.h....'....i'B.
 ...
06070: 43 48 49 50 00 00 20 10 00 00 00 18 80 00 20 00  CHIP.. ....... .
06080: 44 D0 5E 06 02 C0 44 11 40 04 11 44 01 5F 1C 73  D.^...D.@..D._.s
 ...
1E130: 43 48 49 50 00 00 20 10 00 00 00 39 80 00 20 00  CHIP.. ....9.. .
```

```
1E140: 85 EB 41 EA 9E 08 03 00 C0 06 18 01 00 C0 08 03  ..A.............
```

The FUN PLAY Cartridge uses $DE00 for bank selection, and uses 8KiB banks ($2000) at $8000-$9FFF. There are 16 banks of ROM memory and are referenced by the following values:

```
$00 -> Bank 0
$08 -> Bank 1
$10 -> Bank 2
$18 -> Bank 3
$20 -> Bank 4
$28 -> Bank 5
$30 -> Bank 6
$38 -> Bank 7
$01 -> Bank 8
$09 -> Bank 9
$11 -> Bank 10
$19 -> Bank 11
$21 -> Bank 12
$29 -> Bank 13
$31 -> Bank 14
$39 -> Bank 15
```

The bank field in the chip headers is set according to the value written to $DE00. The following bits are used for bank decoding in $DE00 (0 being the LSB, 3 being the MSB).

```
Bit# 76543210
     xx210xx3
```

After copying memory from the ROM banks, the selection program writes a value of $86 to $DE00. This seems either to reset or disable the cartridge ROM.

17.14.3.9 8 - Super Games

Size	64KiB (4 banks of 16KiB each)
EXROM	active (lo) (0)
GAME	active (lo) (0)
Startup mode	16KiB game
Load address	$8000-BFFF (all modules)

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 08 00 00 00 00 00 00 00 00  ...@............
0020: 53 55 50 45 52 20 47 41 4D 45 53 00 00 00 00 00  SUPER GAMES.....
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 40 10 00 00 00 00 80 00 40 00  CHIP..@.......@.
0050: 0A 80 0A 80 C3 C2 CD 38 30 00 A9 80 A0 00 85 FB  .......80.......
...
4050: 43 48 49 50 00 00 40 10 00 00 00 01 80 00 40 00  CHIP..@.......@.
4060: 27 80 A8 80 C3 C2 CD 38 30 00 40 C0 40 C0 40 C0  '......80.@.@.@.
...
```

```
8060: 43 48 49 50 00 00 40 10 00 00 00 02 80 00 40 00  CHIP..@.......@.
8070: 00 00 00 49 4D C7 64 47 46 45 F3 48 DC 08 7E 0B  ...IM.dGFE.H..~.
...
C070: 43 48 49 50 00 00 40 10 00 00 00 03 80 00 40 00  CHIP..@.......@.
C080: D5 F9 F0 C1 D5 F7 F0 BD E8 B5 02 F0 FB C9 05 30  ..............0
```

The Super Games cartridge uses 4 16KiB banks ($8000-$BFFF) of ROM memory. Bank selecting is done by writing to $DF00.

$DF00 register is as follows:

```
bit   meaning
---   -------
 0    bank bit 0
 1    bank bit 1
 2    mode (0 = EXROM/GAME (bridged on the same wire - 16KiB config)
           1 = cartridge disabled)
 3    write-protect-latch  (1 = no more changes are possible until
                                the next hardware-reset )
4-7   unused
```

17.14.3.10 9 - Atomic Power

Size	32KiB (4 banks of 8KiB modules)
EXROM	active (lo) (0)
GAME	inactive (hi) (1)
Load address	$8000-9FFF (all modules)

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 09 00 01 00 00 00 00 00 00  ...@............
0020: 41 74 6F 6D 69 63 20 50 6F 77 65 72 00 00 00 00  Atomic Power....
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
0050: 09 80 0C 80 C3 C2 CD 38 30 4C 41 80 4C 1E 80 4C  .......80LA.L..L
...
2050: 43 48 49 50 00 00 20 10 00 00 00 01 80 00 20 00  CHIP.. ....... .
2060: 09 80 0C 80 C3 C2 CD 38 30 4C 3F 80 4C 91 80 4C  .......80L?.L..L
...
4060: 43 48 49 50 00 00 20 10 00 00 00 02 80 00 20 00  CHIP.. ....... .
4070: EF FC 09 80 C3 C2 CD 38 30 4C 27 80 4C DB 81 4C  .......80L'.L..L
...
6070: 43 48 49 50 00 00 20 10 00 00 00 03 80 00 20 00  CHIP.. ....... .
6080: 09 80 0C 80 C3 C2 CD 38 30 4C 73 86 4C 30 80 4C  .......80Ls.L0.L
```

This cart has 32KiB of ROM and 8KiB of RAM

Writing to I/O-1 will do the following:

```
bit  meaning
---  -------
7    extra ROM bank selector (A15) (unused)
```

```
6    1 = resets FREEZE-mode (turns back to normal mode)
5    1 = enable RAM at ROML ($8000-$9FFF) &
         I/O-2 ($DF00-$DFFF = $9F00-$9FFF)
4    ROM bank selector high (A14)
3    ROM bank selector low  (A13)
2    1 = disable cartridge (turn off $DE00)
1    1 = /EXROM high
0    1 = /GAME low
```

If bit 5 (RAM enable) is 1, bit 0,1 (exrom/game) is == 2 (cart off), bit 2,6,7 (cart disable, freeze clear) are 0, then cart ROM (Bank 0..3) is mapped at 8000-9FFF, and cart RAM (Bank 0) is mapped at A000-BFFF and cart RAM (Bank 0) is is enabled in the I/O-2 area using 16KiB game config.

The cart RAM or ROM is available through a window in the I/O-2 range.

17.14.3.11 10 - Epyx Fastload

Size	8KiB
EXROM	active (lo) (0)
GAME	inactive (hi) (1)
Load address	$8000-9FFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 0A 00 01 00 00 00 00 00 00  ...@............
0020: 45 50 59 58 20 46 41 53 54 4C 4F 41 44 00 00 00  EPYX FASTLOAD...
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
0050: 30 80 5E FE C3 C2 CD 38 30 20 04 90 4C 38 DF AB  0.^....80 ..L8..
```

The Epyx FastLoad cart uses a simple capacitor to toggle the ROM on and off:

the capacitor is discharged, and 8KiB game config enabled, by either reading ROML or reading I/O-1. If none of those accesses happen the capacitor will charge, and if it is charged (after 512 cycles) then the ROM will get disabled.

17.14.3.12 11 - Westermann Learning

Size	16KiB
EXROM	active (lo) (0)
GAME	active (lo) (0)
Load address	$8000-BFFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 0B 00 00 00 00 00 00 00 00  ...@............
0020: 57 45 53 54 45 52 4D 41 4E 4E 00 00 00 00 00 00  WESTERMANN......
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 40 00  CHIP.. .......@.
0050: 09 80 9C 80 C3 C2 CD 38 30 A2 00 8E 16 D0 20 84  .......80..... .
```

Any read from the I/O-2 range will switch the cart off.

17.14.3.13 12 - Rex Utility

Size	8KiB
EXROM	active (lo) (0)
GAME	inactive (hi) (1)
Load address	$8000-9FFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 0C 00 01 00 00 00 00 00 00  ...@............
0020: 52 45 58 00 00 00 00 00 00 00 00 00 00 00 00 00  REX.............
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
0050: 08 80 C1 FE C3 C2 CD 38 30 6C 95 E3 20 A3 FD 20  .......80l.. ..
```

Reading from $DF00-DFBF disables ROM, reading from $DFC0-DFFF enables ROM (8KiB game config).

17.14.3.14 13 - Final Cartridge I

Size	16KiB
EXROM	active (lo) (0)
GAME	active (lo) (0)
Load address	$8000-BFFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 0D 00 00 00 00 00 00 00 00  ...@............
0020: 54 68 65 20 46 69 6E 61 6C 20 43 61 72 74 72 69  The Final Cartri
0030: 64 67 65 20 49 00 00 00 00 00 00 00 00 00 00 00  dge I...........
0040: 43 48 49 50 00 00 40 10 00 00 00 00 80 00 40 00  CHIP..@.......@.
0050: 80 BA 5E FE C3 C2 CD 38 30 00 A0 A0 20 2D FE 58  ..^....80... -.X
```

Any access to I/O-1 turns cartridge ROM off. Any access to I/O-2 turns cartridge ROM on.

The cart ROM is visible in I/O-1 and I/O-2.

17.14.3.15 14 - Magic Formel

Size	64KiB (8 banks of 8KiB)
EXROM	inactive (hi) (1)
GAME	active (lo) (0)
Load Address	$E000-FFFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 0E 01 00 00 00 00 00 00 00  ...@............
0020: 4D 61 67 69 63 20 46 6F 72 6D 65 6C 00 00 00 00  Magic Formel....
```

```
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 20 10 00 00 00 00 E0 00 20 00  CHIP.. ....... .
0050: 4D 46 30 8D 00 DF 60 8D 01 DF 60 8D 02 DF 60 8D  MF0...‘...‘...‘.
...
2050: 43 48 49 50 00 00 20 10 00 00 00 01 E0 00 20 00  CHIP.. ....... .
2060: 4C 5F E4 8D 00 DF 60 8D 01 DF 60 8D 02 DF 60 8D  L_....‘...‘...‘.
...
4060: 43 48 49 50 00 00 20 10 00 00 00 02 E0 00 20 00  CHIP.. ....... .
4070: 4D 46 32 8D 00 DF 60 8D 01 DF 60 8D 02 DF 60 8D  MF2...‘...‘...‘.
...
6070: 43 48 49 50 00 00 20 10 00 00 00 03 E0 00 20 00  CHIP.. ....... .
6080: 4D 46 33 8D 00 DF 60 8D 01 DF 60 8D 02 DF 60 8D  MF3...‘...‘...‘.
...
8080: 43 48 49 50 00 00 20 10 00 00 00 04 E0 00 20 00  CHIP.. ....... .
8090: 4D 46 34 8D 00 DF 60 8D 01 DF 60 8D 02 DF 60 8D  MF4...‘...‘...‘.
...
A090: 43 48 49 50 00 00 20 10 00 00 00 05 E0 00 20 00  CHIP.. ....... .
A0A0: 4D 46 35 8D 00 DF 60 8D 01 DF 60 8D 02 DF 60 8D  MF5...‘...‘...‘.
...
C0A0: 43 48 49 50 00 00 20 10 00 00 00 06 E0 00 20 00  CHIP.. ....... .
C0B0: 4D 46 36 8D 00 DF 60 8D 01 DF 60 8D 02 DF 60 8D  MF6...‘...‘...‘.
...
E0B0: 43 48 49 50 00 00 20 10 00 00 00 07 E0 00 20 00  CHIP.. ....... .
E0C0: 4D 46 37 8D 00 DF 60 8D 01 DF 60 8D 02 DF 60 8D  MF7...‘...‘...‘.
```

17.14.3.16 15 - C64 Game System, System 3

Size	512KiB (64 banks of 8KiB each)
EXROM	active (lo) (0)
GAME	inactive (hi) (1)
Load address	$8000-9FFF (all modules)

```
       00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
       -----------------------------------------------  ----------------
00000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
00010: 00 00 00 40 01 00 00 0F 00 01 00 00 00 00 00 00  ...@............
00020: 43 36 34 47 53 20 43 61 72 74 72 69 64 67 65 00  C64GS Cartridge.
00030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
00050: 6D 80 C5 80 C3 C2 CD 38 30 4C CB 80 4C 36 84 4C  m......80L..L6.L
...
02050: 43 48 49 50 00 00 20 10 00 00 00 01 80 00 20 00  CHIP.. ....... .
02060: 18 D0 A9 FF 8D 15 D0 8D 1D D0 8D 17 D0 A2 07 A9  ................
...
04060: 43 48 49 50 00 00 20 10 00 00 00 02 80 00 20 00  CHIP.. ....... .
04070: E0 08 19 21 77 84 52 98 9F 80 A5 21 31 01 31 89  ...!w.R....!1.1.
...
06070: 43 48 49 50 00 00 20 10 00 00 00 03 80 00 20 00  CHIP.. ....... .
```

```
06080: C0 08 1C 1D A0 92 03 03 D8 AA 04 C0 B8 01 40 EA  ..............@.
...
7E430: 43 48 49 50 00 00 20 10 00 00 00 3F 80 00 20 00  CHIP.. ....?.. .
7E440: 45 20 41 20 42 49 47 20 58 FE 4F 4E 20 54 48 49  E A BIG X.ON THI
```

Here is a list of the known cartridges:

```
    C64GS 4-in-1      (Commodore)  (512 KiB)
    Last Ninja Remix  (System 3)   (512 KiB)
    Myth              (System 3)   (512 KiB)
```

ROM memory is organized in 8KiB ($2000) banks located at $8000-$9FFF. Bank switching is done by writing to address $DE00+X, where X is the bank number (STA $DE00,X). For instance, to read from bank 3, address $DE03 is written to. Reading from anywhere in the I/O-1 range will disable the cart.

The CRT file contains a string of CHIP blocks, each block with a start address of $8000, length $2000 and the bank number in the bank field. In the cartridge header, EXROM ($18) is set to 0, GAME ($19) is set to 1 to enable the 8 KiB ROM configuration.

17.14.3.17 16 - Warp Speed

Size	16KiB
EXROM	active (lo) (0)
GAME	active (lo) (0)
Load address	$8000-BFFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 10 00 00 00 00 00 00 00 00  ...@............
0020: 57 61 72 70 73 70 65 65 64 00 00 00 00 00 00 00  Warpspeed.......
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 40 10 00 00 00 00 80 00 40 00  CHIP..@.......@.
0050: 4C 22 80 4C 22 80 FF 43 42 4D 20 53 E4 20 18 E5  L".L"..CBM S. ..
```

After RESET or POWER ON, 16KiB of cartridge ROM is visible at $8000-$BFFF. Additionally, ROM normally located at $9E00-$9FFF is mirrored into I/O-1 and I/O-2 at $DE00-$DFFF. ROM at $8000-$BFFF is disabled by writing into the I/O-2 area (typically $DF00) and may be re-enabled by writing into I/O-1 ($DE00). However, the $DE00-$DFFF (I/O-1/I/O-2) area itself always remains mapped to cartridge ROM.

17.14.3.18 17 - Dinamic

Size	128KiB (16 banks of 8KiB each)
EXROM	active (lo) (0)
GAME	inactive (hi) (1)
Load address	$8000-9FFF (all modules)

```
       00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
       -----------------------------------------------  ----------------
00000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
00010: 00 00 00 40 01 00 00 11 00 01 00 00 00 00 00 00  ...@............
00020: 4E 61 72 63 6F 20 50 6F 6C 69 63 65 00 00 00 00  Narco Police....
```

```
00030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
00050: 0B 80 0B 80 C3 C2 CD 38 30 00 00 78 A2 FF 9A D8  .......80..x....
..
02050: 43 48 49 50 00 00 20 10 00 00 00 01 80 00 20 00  CHIP.. ....... .
02060: 1C 8C 1B 8C 16 16 8F 16 16 88 1C 1C 86 1C 1C 89  ................
..
04060: 43 48 49 50 00 00 20 10 00 00 00 02 80 00 20 00  CHIP.. ....... .
04070: B6 02 07 08 07 07 00 0A 0A B6 00 05 0A 00 07 07  ................
..
1E130: 43 48 49 50 00 00 20 10 00 00 00 0F 80 00 20 00  CHIP.. ....... .
1E140: 00 D5 70 03 F5 70 0F 5F 70 0F F7 70 35 FD F0 37  ..p..p._p..p5..7
```

Here is a list of the known DINAMIC cartridges:

```
Narco Police  (128 KiB)
Satan         (128 KiB)
```

ROM memory is organized in 8KiB ($2000) banks located at $8000-$9FFF. Bank switching is done by reading from address $DE00+X, where X is the bank number (LDA $DE00,X). For instance, to read from bank 3, address $DE03 is accessed.

The CRT file contains a string of CHIP blocks, each block with a start address of $8000, length $2000 and the bank number in the bank field. In the cartridge header, EXROM ($18) is set to 0, GAME ($19) is set to 1 to enable the 8 KiB ROM configuration.

17.14.3.19 18 - Zaxxon, Super Zaxxon (SEGA)

Size	20KiB (3 banks of different sizes)
EXROM	active (lo) (0)
GAME	active (lo) (0)
Load address	$8000-8FFF (mirrored in $9000-9FFF, module 0, chip U1) $A000-BFFF (banked modules 1 and 2, chip U2)

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 12 00 00 00 00 00 00 00 00  ...@............
0020: 5A 61 78 78 6F 6E 00 00 00 00 00 00 00 00 00 00  Zaxxon..........
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 10 10 00 00 00 00 80 00 10 00  CHIP............
0050: 0D 80 29 80 C3 C2 CD 38 30 78 4C 09 80 78 A9 00  ..)....80xL..x..
..
1050: 43 48 49 50 00 00 20 10 00 00 00 00 A0 00 20 00  CHIP.. ....... .
1060: A2 0F BD 00 20 D0 04 CA 10 F8 60 BD 70 20 F0 0D  .... .....‘.p ..
..
3060: 43 48 49 50 00 00 20 10 00 00 00 01 A0 00 20 00  CHIP.. ....... .
3070: 65 A2 36 A3 E7 A3 CB A4 94 A5 86 A6 5E A7 35 A8  e.6.........^.5.
```

The (Super) Zaxxon carts use a 4KiB ($1000) ROM at $8000-$8FFF (mirrored in $9000-$9FFF) along with two 8KiB ($2000) cartridge banks located at $A000-$BFFF. One of the two banks is selected by doing a read access to either the $8000-$8FFF area (bank 0 is

selected) or to $9000-$9FFF area (bank 1 is selected). EXROM ($18 = $00) and GAME ($19 = $00) lines are always pulled to GND to select the 16 KiB ROM configuration.

The CRT file includes three CHIP blocks:

```
a) bank = 0, load address = $8000, size = $1000
b) bank = 0, load address = $A000, size = $2000
c) bank = 1, load address = $A000, size = $2000
```

17.14.3.20 19 - Magic Desk, Domark, HES Australia

Size	32KiB, 64KiB or 128KiB sizes (4 to 16 banks of 8KiB each)
EXROM	active (lo) (0)
GAME	inactive (hi) (1)
Startup mode	8KiB Game
Load address	(banks 00-15) - $8000-9FFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 13 00 01 00 00 00 00 00 00  ...@............
0020: 4D 61 67 69 63 20 44 65 73 6B 00 00 00 00 00 00  Magic Desk......
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
0050: 09 80 C6 CA C3 C2 CD 38 30 8E 16 D0 20 A3 FD 20  .......80... ..
..
2050: 43 48 49 50 00 00 20 10 00 00 00 01 80 00 20 00  CHIP.. ....... .
2060: 00 3F 0A 01 00 86 4E 24 28 31 30 29 3A 4A 4F 59  .?....N$(10):JOY
..
4060: 43 48 49 50 00 00 20 10 00 00 00 02 80 00 20 00  CHIP.. ....... .
4070: 00 8B C9 28 4E 24 2C 31 29 B3 B1 22 FF 22 A7 32  ...(N$,1)..".".2
..
6070: 43 48 49 50 00 00 20 10 00 00 00 03 80 00 20 00  CHIP.. ....... .
6080: AE 01 83 33 2C 37 2C 22 32 29 20 44 45 4C 20 4B  ...3,7,"2) DEL K
```

This cartridge type is very similar to the OCEAN cart type: ROM memory is organized in 8KiB ($2000) banks located at $8000-$9FFF. Bank switching is done by writing the bank number to $DE00. Deviant from the Ocean type, bit 7 is cleared for selecting one of the ROM banks. If bit 7 is set ($DE00 = $80), the GAME/EXROM lines are disabled, turning on RAM at $8000-$9FFF instead of ROM.

In the cartridge header, EXROM ($18) is set to 0, GAME ($19) is set to 1 to indicate the RESET/power-up configuration of 8 KiB ROM.

Here is a list of the known cartridges:

```
Ghosbusters        (HES Australia)   (32 KiB)
Magic Desk         (Commodore)       (32 KiB)
Badlands           (Domark)          (64 KiB)
Vindicators        (Domark)          (64 KiB)
Wonderboy          (HES Australia)   (64 KiB)
Cyberball          (Domark)         (128 KiB)
```

17.14.3.21 20 - Super Snapshot V5

Size	64KiB (4 banks of 16KiB each)
EXROM	active (lo) (0)
GAME	active (lo) (0)
Load address	$8000-BFFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 14 00 00 00 00 00 00 00 00  ...@............
0020: 53 75 70 65 72 20 53 6E 61 70 73 68 6F 74 20 35  Super Snapshot 5
0030: 20 4E 54 53 43 00 00 00 00 00 00 00 00 00 00 00   NTSC...........
0040: 43 48 49 50 00 00 40 10 00 00 00 00 80 00 40 00  CHIP..@.......@.
0050: 09 80 59 80 C3 C2 CD 38 30 20 03 9F 00 FA F4 20  ..Y....80 ..... 
...
4050: 43 48 49 50 00 00 40 10 00 00 00 01 80 00 40 00  CHIP..@.......@.
4060: 79 DE BC FE C3 C2 CD 38 30 A9 05 8D 20 D0 8D 21  y......80... ..!
...
8060: 43 48 49 50 00 00 40 10 00 00 00 02 80 00 40 00  CHIP..@.......@.
8070: 50 DE BC FE C3 C2 CD 38 30 A9 0A 85 6A A9 0D 85  P......80...j...
...
C070: 43 48 49 50 00 00 40 10 00 00 00 03 80 00 40 00  CHIP..@.......@.
C080: 50 DE BC FE C3 C2 CD 38 30 85 07 20 1A AD A5 76  P......80.. ...v
```

The first page of the currently selected ROM bank is mirrored in the I/O-1 range when reading.

The control Register is the I/O-1 range when writing:

```
bit  meaning
---  -------
7-5  unused
4    ROM/RAM bank bit 1
3    ROM enable
2    ROM/RAM bank bit 0
1    RAM enable, EXROM
0    release freeze, !GAME
```

17.14.3.22 21 - Comal-80

Size	64KiB (4 banks of 16KiB each)
EXROM	active (low) (0)
GAME	active (low) (0)
Load address	$8000-BFFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 15 00 00 00 00 00 00 00 00  ...@............
0020: 43 6F 6D 61 6C 20 38 30 00 00 00 00 00 00 00 00  Comal 80........
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

```
0040: 43 48 49 50 00 00 40 10 00 00 00 00 80 00 40 00  CHIP..@.......@.
0050: 87 87 70 CF C3 C2 CD 38 30 4C AA CF 4C 70 CF 4C  ..p....80L..Lp.L
...
4050: 43 48 49 50 00 00 40 10 00 00 00 01 80 00 40 00  CHIP..@.......@.
4060: AA CF 70 CF C3 C2 CD 38 30 01 29 01 28 01 2C 04  ..p....80.).(.,.
...
8060: 43 48 49 50 00 00 40 10 00 00 00 02 80 00 40 00  CHIP..@.......@.
8070: AA CF 70 CF C3 C2 CD 38 30 91 92 92 92 92 92 92  ..p....80.......
...
C070: 43 48 49 50 00 00 40 10 00 00 00 03 80 00 40 00  CHIP..@.......@.
C080: 7B C8 7E C8 C3 C2 CD 38 30 43 4F 4D 41 4C 80 93  {.~....80COMAL..
```

The Comal-80 Cartridge uses 16KiB banks ($4000) at $8000-$BFFF. There are 4 banks of ROM memory and the cart has 1 (write-only) bank control register which is located at $DE00 and mirrored throughout the $DE00-$DEFF range.

bit	meaning
7	exrom?
6	game?
5	unknown function (used by the software to disable the cartridge)
4	unused?
3	unused?
2	unknown function (used by the software however)
0-1	selects bank

17.14.3.23 22 - Structured Basic

Size	16KiB (2 banks of 8KiB each)
EXROM	inactive (hi) (1)
GAME	active (0)
Load address	$8000-9FFF

No sample data/file available.

IF YOU OWN THIS TYPE OF CARTRIDGE AND/OR CAN PROVIDE A ROM IMAGE, PLEASE GET IN TOUCH WITH US!

Any read/write access to $DE00 or $DE01 will switch in bank 0. Any read/write access to $DE02 will switch in bank 1. Any read/write access to $DE03 will switch off EXROM.

17.14.3.24 23 - Ross

Size	16KiB or 32KiB sizes (1 or 2 banks of 16KiB each)
EXROM	active (lo) (0)
GAME	active (lo) (0)
Load address	$8000-BFFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 17 00 00 00 00 00 00 00 00  ...@............
0020: 52 6F 73 73 20 31 34 00 00 00 00 00 00 00 00 00  Ross 14.........
```

```
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 40 10 00 00 00 00 80 00 40 00  CHIP..@.......@.
0050: 09 80 09 80 C3 C2 CD 38 30 A2 00 BD 20 80 4D 0E  .......80... .M.
...
4050: 43 48 49 50 00 00 40 10 00 00 00 01 80 00 40 00  CHIP..@.......@.
4060: 3F 5A 4D 4D 50 4D 8D 25 3F 1A 1F 77 3F CD E0 3F  ?ZMMPM.%?..w?..?
```

Any read access to $DE00 will switch in bank 1 (if cart is 32KiB). Any read access to $DF00 will switch off EXROM and GAME.

17.14.3.25 24 - Dela EP64

Size	8KiB to 72KiB sizes (1 to 9 banks of 8KiB each, or 1 bank of 8KiB and 1 or 2 banks of 32KiB each)
EXROM	active (lo) (0)
GAME	inactive (hi) (1)
Load address	$8000-9FFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 18 00 01 00 00 00 00 00 00  ...@............
0020: 44 45 4C 41 20 45 50 36 34 00 00 00 00 00 00 00  DELA EP64.......
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
0050: 00 85 5E FE C3 C2 CD 38 30 FF FF FF FF FF FF FF  ..^....80.......
...
2050: 43 48 49 50 00 00 80 10 00 00 00 01 80 00 80 00  CHIP............
2060: 54 45 53 54 0D 2A 0D 54 45 20 36 34 0D 00 00 00  TEST.*.TE 64....
```

This is an eprom cartridge. It has 1 2764 (8KiB) which holds the base eprom with the base menu, and 2 27256 eproms of which 8KiB parts are banked into the $8000-9FFF area.

The bank selecting is done by writing to $DE00. The following bits are used for bank decoding in $DE00 (0 being the LSB, 3 being the MSB).

```
Bit# 76543210
     xx10xx32
```

Any bank value below 4 or above 11 switches in the base bank (bank 0).

The bit values for each eprom bank are :

```
eprom bank 1 : xx00xx01
eprom bank 2 : xx01xx01
eprom bank 3 : xx10xx01
eprom bank 4 : xx11xx01
eprom bank 5 : xx00xx10
eprom bank 6 : xx01xx10
eprom bank 7 : xx10xx10
eprom bank 8 : xx11xx10
```

Setting bit 7 high will switch off EXROM.

17.14.3.26 25 - Dela EP7x8

Size	8KiB to 64KiB sizes (1 to 8 banks of 8KiB each)
EXROM	active (lo) (0)
GAME	inactive (hi) (1)
Load address	$8000-9FFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 19 00 01 00 00 00 00 00 00  ...@............
0020: 44 45 4C 41 20 45 50 37 78 38 00 00 00 00 00 00  DELA EP7x8......
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
0050: 09 80 5E FE C3 C2 CD 38 30 78 A2 FF 9A D8 8E 16  ..^....80x......
...
2050: 43 48 49 50 00 00 20 10 00 00 00 01 80 00 20 00  CHIP.. ....... .
2060: 94 83 A0 83 C3 C2 CD 38 30 02 BB 5A 30 5F EE 3D  .......80..Z0_.=
```

This is an eprom cartridge. It has 8 8KiB banks of which the first holds the base menu, the other eproms can be banked into the $8000-9FFF area.

The bank selecting is done by writing to $DE00. Each low bit is used to bank in the respective eprom. If all bits are high then the EXROM is switched off.

The bit values for each eprom bank is:

```
eprom bank 1 : 11111110 ($FE) (base eprom)
eprom bank 2 : 11111101 ($FD)
eprom bank 3 : 11111011 ($FB)
eprom bank 4 : 11110111 ($F7)
eprom bank 5 : 11101111 ($EF)
eprom bank 6 : 11011111 ($DF)
eprom bank 7 : 10111111 ($BF)
eprom bank 8 : 01111111 ($7F)

EXROM off    : 11111111 ($FF)
```

17.14.3.27 26 - Dela EP256

Size	8KiB to 262KiB sizes (1 to 33 banks of 8KiB each)
EXROM	active (lo) (0)
GAME	inactive (hi) (1)
Load address	$8000-9FFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 1A 00 01 00 00 00 00 00 00  ...@............
0020: 44 45 4C 41 20 45 50 32 35 36 00 00 00 00 00 00  DELA EP256......
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
0050: 00 85 5E FE C3 C2 CD 38 30 93 0D 2B 2B 2B 20 45  ..^....80..+++ E
```

```
...
2050: 43 48 49 50 00 00 20 10 00 00 00 01 80 00 20 00  CHIP.. ....... .
2060: 09 80 28 80 C3 C2 CD 38 30 78 A2 05 8E 16 D0 20  ..(....80x.....
...
4060: 43 48 49 50 00 00 20 10 00 00 00 02 80 00 20 00  CHIP.. ....... .
4070: 0B 80 BC FE C3 C2 CD 38 30 DC 10 8E 16 D0 20 87  .......80..... .
...
6070: 43 48 49 50 00 00 20 10 00 00 00 03 80 00 20 00  CHIP.. ....... .
6080: 09 80 F6 8E C3 C2 CD 38 30 A2 C8 8E 16 D0 20 ..  ?.?....80...?.?.
...
8080: 43 48 49 50 00 00 20 10 00 00 00 04 80 00 20 00  CHIP.. ....... .
8090: 94 83 A0 83 C3 C2 CD 38 30 02 BB 5A 30 5F EE 3D  .......80..Z0_.=
```

This is an eprom cartridge. It has 33 8KiB banks of which the first holds the base menu, the other eproms can be banked into the $8000-9FFF area.

The bank selecting is done by writing to $DE00.

The values for the (extra) eprom banks are:

```
eprom banks  1- 8 : $38-3F
eprom banks  9-16 : $28-2F
eprom banks 17-24 : $18-1F
eprom banks 25-32 : $08-0F
```

Setting bit 7 high will switch off EXROM.

17.14.3.28 27 - Rex EP256

Size	8KiB to 262KiB sizes (1 bank of 8KiB and 1 to 8 banks of either 8KiB, 16KiB or 32KiB)
EXROM	active (lo) (0)
GAME	inactive (hi) (1)
Load address	$8000-9FFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 1B 00 01 00 00 00 00 00 00  ...@............
0020: 52 45 58 20 45 50 32 35 36 00 00 00 00 00 00 00  REX EP256.......
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
0050: 09 80 C1 FE C3 C2 CD 38 30 20 A3 FD 20 50 FD 20  .......80 .. P.
...
2050: 43 48 49 50 00 00 20 10 00 00 00 01 80 00 20 00  CHIP.. ....... .
2060: 09 80 F2 8F C3 C2 CD 38 30 A2 C8 8E 16 D0 20 A3  .......80..... .
...
4060: 43 48 49 50 00 00 40 10 00 00 00 02 80 00 40 00  CHIP..@.......@.
4070: 09 80 09 80 C3 C2 CD 38 30 58 D8 20 84 FF 20 8A  .......80X. .. .
```

This is an eprom cartridge. It has 9 eprom sockets, of which the first holds the base eprom with the base menu which is an 8KiB eprom, the other eprom sockets can handle 8KiB, 16KiB or 32KiB eproms, of which 8KiB can be banked into the $8000-9FFF area.

The bank selecting is done by writing to $DFA0. Bits 2, 1 and 0 determine which socket is used and bits 5 and 4 are used to select an 8KiB piece of the eprom.

The possible values for bits 5 and 4 for the (extra) eprom banks are:

```
 8KiB        : 3, 2, 1, 0

16KiB bank 0 : 2, 0
16KiB bank 1 : 3, 1

32KiB bank 0 : 0
32KiB bank 1 : 1
32KiB bank 2 : 2
32KiB bank 3 : 3
```

Reading from $DFC0 switches off the EXROM. Reading from $DFE0 switches on the EXROM.

17.14.3.29 28 - Mikro Assembler

Size	8KiB (1 bank of 8KiB)
EXROM	active (lo) (0)
GAME	inactive (hi) (1)
Load address	$8000-9FFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 1C 00 01 00 00 00 00 00 00  ...@............
0020: 56 49 43 45 20 43 41 52 54 00 00 00 00 00 00 00  VICE CART.......
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
0050: 60 80 FE 80 C3 C2 CD 38 30 4C 07 87 4C CA 82 41  `......80L..L..A
```

The $9E00-$9EFF range is mirrored at $DE00-$DEFF. The $9F00-$9FFF range is mirrored at $DF00-$DFFF.

17.14.3.30 29 - Final Cartridge Plus

Size	32KiB (1 bank of 32KiB)
EXROM	inactive (hi) (1)
GAME	active (lo) (0)
Load address	$0000-$7FFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 1D 01 00 00 00 00 00 00 00  ...@............
0020: 56 49 43 45 20 43 41 52 54 00 00 00 00 00 00 00  VICE CART.......
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 80 10 00 00 00 00 00 00 80 00  CHIP............
0050: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
```

This cart has 32KiB of ROM, bank 0 is in the cart image but is unused. The first 8KiB of the cart image is unused, the second 8KiB of the cart image is mapped to $E000-$FFFF, the third 8KiB of the cart image is mapped to $8000-$9FFF and the fourth 8KiB of the cart image is mapped to $A000-$BFFF. An NMI can be triggered by the cart, if address $0001 is written to and the cartridge is enabled. The cart can be disabled by software, by clearing bit 4 when writing to $DF00-$DFFF. Cart ROM at $E000-$FFFF can be disabled by setting bit 5 to 0 when writing to $DF00-$DFFF. Cart ROM at $8000-$BFFF can be disabled by setting bit 6 to 1 when writing to $DF00-$DFFF. Bit 7 of a byte written to $DF00-$DFFF can be read back from the cartridge if enabled (like a memory cell).

17.14.3.31 30 - Action Replay 4

Size	32KiB (4 banks of 8KiB)
EXROM	active (lo) (0)
GAME	inactive (hi) (1)
Load address	$8000-$9FFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 1E 00 01 00 00 00 00 00 00  ...@............
0020: 56 49 43 45 20 43 41 52 54 00 00 00 00 00 00 00  VICE CART.......
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
0050: EA 78 48 A9 7F 8D 0D DD D0 0E 48 AD 0D DD 10 04  .xH.......H.....
...
2050: 43 48 49 50 00 00 20 10 00 00 00 01 80 00 20 00  CHIP.. ....... .
2060: 09 80 0C 80 C3 C2 CD 38 30 4C E9 80 4C 81 81 4C  .......80L..L..L
...
4060: 43 48 49 50 00 00 20 10 00 00 00 02 80 00 20 00  CHIP.. ....... .
4070: 09 80 0E 80 C3 C2 CD 38 30 A2 00 4C EF FC 20 BC  .......80..L.. .
...
6070: 43 48 49 50 00 00 20 10 00 00 00 03 80 00 20 00  CHIP.. ....... .
6080: 09 80 0C 80 C3 C2 CD 38 30 4C 70 88 4C 3F 80 4C  .......80Lp.L?.L
```

The control register is the I/O-1 range:

```
bit  meaning
---  -------
0    Eprom banking bit 0 (bank address 13)
1    Controls the GAME line (0 sets GAME low, 1 sets GAME high)
2    Freeze-end bit (disables the register and hides any rom bank)
3    Controls the Exrom line (1 sets EXROM low, 0 sets EXROM high)
4    Eprom banking bit 1 (bank address 14)
5-7  Unused
```

The first page of the currently banked ROM block can be read in the I/O-2 range.

17.14.3.32 31 - Stardos

Size	16KiB (2 banks of 8KiB)
EXROM	inactive (hi) (1)

GAME active (lo) (0)
Load address $8000-$9FFF (bank 0), $E000-$FFFF (bank 1)

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 1F 01 00 00 00 00 00 00 00  ...@............
0020: 56 49 43 45 20 43 41 52 54 00 00 00 00 00 00 00  VICE CART.......
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
0050: F9 80 B6 80 C3 C2 CD 38 30 FD 80 89 80 4C 0C 88  .......80....L..
...
2050: 43 48 49 50 00 00 20 10 00 00 00 00 E0 00 20 00  CHIP.. ....... .
2060: 85 56 20 0F BC A5 61 C9 88 90 03 20 D4 BA 20 CC  .V ...a.... .. .
```

This cart has 16KiB of ROM, of which the first 8KiB is mapped in at $8000-$9FFF and the second 8KiB is used as a kernel replacement. The kernel replacement is achieved by a clip that needs to be installed inside the C64.

Reading from I/O-1 causes a capacitor to get charged with every read, once the capacitor is charged enough it switches the cart on.

Reading from I/O-2 causes a different capacitor to get charged with every read, once the capacitor is charged enough it switched the cart off.

17.14.3.33 32 - EasyFlash

Size 1024KiB (64 banks of 2 * 8KiB)
EXROM inactive (hi) (1)
GAME active (lo) (0)
Load address $8000-$9FFF (ROML), $A000-$BFFF or $E000-$FFFF (ROMH)

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 20 01 00 00 00 00 00 00 00  ...@... ........
0020: 45 61 73 79 46 6C 61 73 68 20 43 61 72 74 72 69  EasyFlash Cartri
0030: 64 67 65 00 00 00 00 00 00 00 00 00 00 00 00 00  dge.............
0040: 43 48 49 50 00 00 20 10 00 00 00 01 80 00 20 00  CHIP.. ....... .
0050: 00 85 5E FE C3 C2 CD 38 30 93 0D 2B 2B 2B 20 45  ..^....80..+++ E
```

EasyFlash is a 1 MiB Flash EPROM card with multiple configurations and banks possible, it also has 256 bytes of RAM which is mapped into the I/O-2 range.

There are two control registers, one at $DE00 and one at $DE02.

The register at $DE00 does the following:

```
bit  meaning
---  -------
7    LED control
6-3  Unused
2    Mode (0/1)
1    Exrom line control
```

```
0    Game line control
```

The register at $DE02 controls which bank is mapped into ROMH and ROML.

17.14.3.34 33 - EasyFlash Xbank

Size	-
EXROM	-
GAME	-
Load address	-

This CRT type is not actually related to a seperate hardware, it is used by some EasyFlash related tools as a container format. Consequently VICE does (can) not load files of this type.

17.14.3.35 34 - Capture

Size	8KiB (1 bank of 8KiB)
EXROM	inactive (hi) (1)
GAME	inactive (hi) (1)
Load address	$E000-$FFFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 22 00 00 00 00 00 00 00 00  ...@..."........
0020: 4D 61 67 69 63 20 46 6F 72 6D 65 6C 00 00 00 00  Magic Formel....
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 20 10 00 00 00 01 E0 00 20 00  CHIP.. ....... .
0050: 00 0A 0D 8A B4 A1 20 80 00 0A 82 8A 8D 20 9E 20  ...... ...... .
```

This cart has 8KiB of ROM which is mapped to $E000, and 8KiB of RAM which is mapped to $6000. The cartridge is disabled after a reset.

When the freeze button is pressed the following happens:

- an NMI is generated
- as soon as the current adress is in bank 0xfe the cart switches to ultimax mode. The cart ROM then contains one page full of "jmp $eaea", which ultimately calls the freezer code.
- the $FFF7/$FFF8 "register" logic is enabled and any access (read or write) to $FFF7 will turn the cart_enabled off (leave ultimax mode), and an access to $FFF8 will turn the cart back on (enter ultimax mode). the "register logic" that causes this can only be disabled again by a hardware reset.

17.14.3.36 35 - Action Replay 3

Size	16KiB (2 banks of 8KiB)
EXROM	active (lo) (0)
GAME	inactive (hi) (1)
Load address	$8000-$9FFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
```

```
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 23 00 01 00 00 00 00 00 00  ...@...#........
0020: 56 49 43 45 20 43 41 52 54 00 00 00 00 00 00 00  VICE CART.......
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
0050: EA A9 E3 48 A9 7B 48 08 4C 1A 80 EA EA EA 48 AD  ...H.{H.L.....H.
...
2050: 43 48 49 50 00 00 20 10 00 00 00 01 80 00 20 00  CHIP.. ....... .
2060: 09 80 5E FE C3 C2 CD 38 30 78 A2 FB D8 9A A9 27  ..^....80x.....'
```

This cart has 16KiB of ROM of which 8KiB is mapped in at both ROML and ROMH. Bank switching and control register is done through the I/O-1 range:

```
bit  meaning
---  -------
7-4  unused
3    Exrom line control
2    Disable cart
1    Unused
0    Bank
```

17.14.3.37 36 - Retro Replay

Size	32KiB, 64KiB or 128KiB (4, 8 or 16 banks of 8KiB)
EXROM	active (lo) (0)
GAME	inactive (hi) (1)
Load address	$8000-$9FFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 24 00 01 00 00 00 00 00 00  ...@...$........
0020: 56 49 43 45 20 43 41 52 54 00 00 00 00 00 00 00  VICE CART.......
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
0050: 09 80 0C 80 C3 C2 CD 38 30 4C 7F 81 4C 87 81 4C  .......80L..L..L
```

Note: When the Cartridge Hardware Subtype/Revision (Offset 0x1A in the CRT Header) is 1, then the image utilizes features only present in the "Nordic Replay" (see description of Nordic/Atomic Power)

The Retro Replay has three registers: Two write-only ($DE00 & $DE01) and one read-only register ($DE00 & $DE01 giving the same results).

The register at $DE00 is reset to $00 on a hard reset if not in flash mode. If in flash mode, it is set to $02 in order to prevent the computer from starting the normal cartridge. Flash mode is selected with a jumper.

Register at $DE00:

```
bit  meaning
---  -------
0    Controls the GAME line: A 1 asserts the line, a 0  will  deassert
     it.
```

```
1    Controls the EXROM line: A 0 will assert it, a  1  will  deassert
     it.
2    Writing a 1 will disable further write accesses to all  registers
     of the Retro Replay, and  set  the  memory  map  of  the  C64  to
     standard, as if there is no cartridge installed at all.
3    Controls bank-address 13 for ROM and RAM banking.
4    Controls bank-address 14 for ROM and RAM banking.
5    Switches between ROM and RAM: 0=ROM, 1=RAM
6    Must be written once to "1" after a successful freeze in order to
     set the correct memory map and  enable  bits  0  and  1  of  this
     register. Otherwise no effect.
7    Controls bank-address 15 for ROM banking.
```

The register at $DE01 is the extended control register. If not in Flash mode, bits 1, 2 and 6 can only be written once. If in Flash mode, the REUcomp bit cannot be set, but the register will not be disabled by the first write. Bit 5 is always set to 0 if not in Flash mode.

Register at $DE01:

```
bit  meaning
---  -------
0    Enable clockport connector.
1    AllowBank (1 allows banking of RAM in $DF00/$DE02 area)
2    NoFreeze (1 disables Freeze function)
3    Bank-address 13 for RAM and ROM (mirror of $DE00)
4    Bank-address 14 for RAM and ROM (mirror of $DE00)
5    Bank-address 16 for ROM (only in flash mode)
6    REU compatibility bit. 0=standard memory map, 1 = REU  compatible
     memory map
7    Bank-address 15 for ROM (mirror of $DE00)
```

Reading from the registers at either $DE00 or $DE01 will return the content of the status register.

Status register:

```
bit  meaning
---  -------
0    1=Flashmode active (jumper set)
1    feedback of AllowBank bit
2    1=Freeze button pressed
3    feedback of banking bit 13
4    feedback of banking bit 14
5    feedback of banking bit 16
6    1=REU compatible memory map active
7    feedback of banking bit 15
```

The following memory maps are available:

- standard - $DE00 and $DE01 registers are active, $DF00-$DFFF contain the last page of the selected 8KiB bank of either ROM or RAM, whatever is selected. RAM can only be accessed in $8000-$9FFF. ROM can be mapped to $8000, $A000 or $E000 with the corresponding status on GAME and EXROM.

Note: If the AllowBank bit is not set, the $DF00-$DFFF area will always access bank 0 of the RAM, so the older cartridge images will work. The AllowBank bit does not have any effect on the ROM mirror in that area.

- Freeze - ROM is mapped to $E000-$FFFF, bank 0 is active directly after Freeze. Writing to bits 0 and 1 of the $DE00 register will have no effect on GAME and EXROM. RAM can be selected and used in $DF00 or $DE02, respectively, but not in $8000. Banking bits work, so you have full read access to the ROM, and access to up to four RAM pages with the AllowBank bit set (minus 2 bytes if REU compatible bit is set). You should leave this memory map ASAP by setting bit 6 of $DE00, because C64 RAM in the $E000 area is not available, and you don't have control of the GAME and EXROM lines.
- REU compatible - $DE00 and $DE01 registers are active, $DE02-$DEFF contain a mirror of $9E02-$9EFF of the selected 8KiB-bank of either ROM or RAM, whatever is selected. RAM can only be accessed in $8000-$9FFF. ROM can be mapped to $8000, $A000 or $E000 with the corresponding status on GAME and EXROM. The $DF00 stays free for use with the 1764 Ram Expansion Unit (REU).

  Note: If the AllowBank bit is not set, the $DE02-$DEFF area will always access bank 0 of the RAM, so the older cartridge images will work. The AllowBank bit does not have any effect on the ROM mirror in that area.

17.14.3.38 37 - MMC64

Size	8KiB (1 bank of 8KiB)
EXROM	active (lo) (0)
GAME	inactive (hi) (1)
Load address	$8000-$9FFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 25 00 01 00 00 00 00 00 00  ...@...%........
0020: 56 49 43 45 20 43 41 52 54 00 00 00 00 00 00 00  VICE CART.......
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
0050: 09 80 64 97 C3 C2 CD 38 30 78 D8 A2 FF 9A 20 D4  ..d....80x.... .
```

The clockport registers of this cart can be switched to either $DE01-$DE0F or $DF21-$DF2F. The control registers are available at $DF10-$DF13.

The register at $DE01 / $DF21 is write only:

```
bit  meaning
---  -------
7-1  Unused
0    0 = disable clock port, 1 = enable clockport
```

The registers at $DE02-$DE0F / $DF22-$DF2F are for the clock port and are read/write.

The register at $DF10 is the MMC64 SPI transfer register, a byte written to this registers is sent to the card & response from the card is read here.

The register at $DF11 is the MMC64 control register:

```
bit  meaning
```

```
bit  meaning
---  -------
0    0 = MMC64 BIOS active, 1 = external ROM active
1    0 = card selected, 1 = card not selected
2    0 = 250khz transfer, 1 = 8mhz transfer
3    0 = clock port @ $DE00, 1 = clock port @ $DF20
4    0 = normal Operation, 1 = flash mode (*)
5    0 = allow external rom when BIOS is disabled,
     1 = disable external ROM
6    0 = SPI write trigger mode, 1 = SPI read trigger mode
7    0 = MMC64 is active, 1 = MMC64 is completely disabled (**)
```

(*) bit can only be programmed when flash jumper is set (**) bit can only be modified after unlocking

The register at $DF12 is the MMC64 status register, which is read-only:

```
bit  meaning
---  -------
0    0 = SPI ready, 1 = SPI busy
1    external GAME line
2    external EXROM line
3    0 = card inserted, 1 = no card inserted
4    0 = card write enabled, 1 = card write disabled
5    0 = flash jumper not set, 1 = flash jumper set
6-7  unused
```

The register at $DF13 is the MMC64 identification register, which when reading from it can have the following values:

$64 when bit 1 of $DF11 is 0. $01 when bit 1 of $DF11 is 1 and REV A hardware is used. $02 when bit 1 of $DF11 is 1 and REV B hardware is used.

when writing to it it can be used to unlock bit 7 of $DF11 or to re-enable the cart:

Write $55 & $AA into this register to unlock bit 7 of $DF11. Write $0A & $1C into this register to re-enable MMC64 hardware.

17.14.3.39 38 - MMC Replay

Size	64KiB or 512KiB (8 or 64 banks of 8KiB)
EXROM	active (lo) (0)
GAME	active (lo) (0)
Load address	$8000-$9FFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 26 00 00 00 00 00 00 00 00  ...@...&........
0020: 56 49 43 45 20 43 41 52 54 00 00 00 00 00 00 00  VICE CART.......
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
0050: 1A 80 6E 9E C3 C2 CD 38 30 4D 4D 43 52 45 50 4C  ..n....80MMCREPL
```

The cart uses the following registers:

$DE00 - RR control register write

```
bit  meaning
---  -------
0    GAME line
1    EXROM line
2    1 = disable RR, bit can be reset by setting bit 6 of $DF12
3    bank address 13
4    bank address 14
5    0 = rom enable, 1 = ram enable
6    1 = exit freeze mode
7    bank address 15
```

$DE01 - extended RR control register write

```
bit  meaning
---  -------
0    0 = disable clockport, 1 = enable clockport
1    0 = disable I/O RAM banking, 1 = enable I/O RAM banking
2    0 = enable freeze, 1 = disable freeze
3    bank address 13 (mirror of $DE00)
4    bank address 14 (mirror of $DE00)
5    0 = enable MMC registers, 1 = disable MMC registers. Can only  be
     written when bit 6 of $DF12 is 1. Register becomes effective when
     bit 0 of $DF11 is 1.
6    0 = RAM/ROM at $DFxx, 1 = RAM/ROM at $DExx
7    bank address 15 (mirror of $DE00)
```

$DE02-$DE0F - Clockport memory area (when enabled)

$DF10 - MMC SPI transfer register, a byte written is sent to the card & response from the card is read here.

$DF11 - MMC control register

```
bit  meaning
---  -------
0    0 = MMC BIOS enabled, 1 = MMC BIOS disabled.  Enabling  MMC  BIOS
     sets ROM banking to the last 64KiB bank.
1    0 = card selected, 1 = card not selected. This bit also  controls
     the green activity LED.
2    0 = 250khz transfer, 1 = 8mhz transfer
3    ALWAYS 0
4    ALWAYS 0
5    (in RR-Mode:)
      0 = allow RR rom when MMC BIOS disabled , 1 = disable RR ROM
      (in mmcreplay bios mode:)
      RAM banking (0 = $E000 - $FFFF, 1 = $8000 - $9FFF)
      (in 16KiB mode:)
      enable RAM at $A000 - $BFFF
6    0 = SPI write trigger mode, 1 = SPI read trigger mode
7    ALWAYS 0
```

$DF12 - MMC status register

```
bit  meaning
---  -------
0    0 = SPI ready, 1 = SPI busy (read)
                    1 = forbid ROM write accesses (write). Setting
                    this bit will disable writes to ROM until next
                    reset
1  feedback of $DE00 bit 0 (GAME)
2  feedback of $DE00 bit 1 (EXROM)
3  0 = card inserted, 1 = no card inserted
4  0 = card write enabled, 1 = card write disabled
5  EEPROM DATA line / DDR register. Setting DATA to "1" enables
   reading data bit from EEPROM at this position.
6  0 = RR compatibility mode, 1 = Extended mode
   Selecting RR compatibility mode limits RAM to 32KiB and disables
   writes to extended banking register. Selecting Extended mode
   enables full RAM banking and enables Nordic Power mode in RR mode.
7  EEPROM CLK line
```

$DF13 - Extended banking register Can only be read/written to when bit 6 of $DF12 is 1

```
bit  meaning
---  -------
0    bank address 16
1    bank address 17
2    bank address 18
3    ALWAYS 0
4    ALWAYS 0
5    16KiB rom mapping
6    1 = enable RR register. Disabling RR register disables ALL
     ROM/RAM banking too.
7    ALWAYS 0
```

NOTE THAT THE MMC REPLAY EMULATION IS CURRENTLY INCOMPLETE AND CONSIDERED BROKEN.

17.14.3.40 39 - IDE64

Size	64KiB or 128KiB (4 or 8 banks of 16KiB)
EXROM	active (lo) (0)
GAME	inactive (hi) (1)
Load address	$8000-$BFFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 27 00 01 00 00 00 00 00 00  ...@...'........
0020: 49 44 45 36 34 20 43 41 52 54 52 49 44 47 45 20  IDE64 CARTRIDGE
0030: 49 44 45 44 4f 53 20 32 30 31 33 31 32 31 32 00  IDEDOS 20131212?
0040: 43 48 49 50 00 00 40 10 00 02 00 00 80 00 40 00  CHIP..@.......@.
0050: 63 80 5E FE C3 C2 CD 38 30 20 49 44 45 36 34 20  c.^....80 IDE64
```

The IDE64 cart uses the following registers:

```
$DE20 - $DE2F IDE BUS Registers
$DE30 - Low Data HDD register
$DE31 - High Data HDD register
```

$DE32 register:

```
bit  meaning
---  -------
7    unused (0)
6    unused (0)
5    unused (0)
4    version number (1)
3    romaddr15
2    romaddr14
1    game
0    exrom
```

```
$DE32 - $DE35 = IDE64 ROM bank select registers
$DE5F         = RTC access (bit 0 only to serial accessed RTC)
$DE60 - $DEFF = ROM used by software
$DEFB         = IDE64 clock reset, kill the cartridge
$DEFC - $DEFF = IDE64 memory configuration registers
```

17.14.3.41 40 - Super Snapshot V4

Size	32KiB (4 banks of 8KiB)
EXROM	active (lo) (0)
GAME	active (lo) (0)
Load address	$8000-$9FFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 28 00 00 00 00 00 00 00 00  ...@...(........
0020: 56 49 43 45 20 43 41 52 54 00 00 00 00 00 00 00  VICE CART.......
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
0050: 80 AD B5 80 C3 C2 CD 38 30 08 48 A9 06 8D 00 DF  .......80.H.....
...
2050: 43 48 49 50 00 00 20 10 00 00 00 00 A0 00 20 00  CHIP.. ....... .
2060: 4C FA A0 A9 07 8D 00 DD 2C 00 DD 50 FB 2C 00 DD  L.......,..P.,..
...
4060: 43 48 49 50 00 00 20 10 00 00 00 01 80 00 20 00  CHIP.. ....... .
4070: 13 80 BC FE C3 C2 CD 38 30 08 48 A9 02 8D 00 DF  .......80.H.....
...
6070: 43 48 49 50 00 00 20 10 00 00 00 01 A0 00 20 00  CHIP.. ....... .
6080: F0 8A 48 A9 00 85 22 85 23 8D 53 0F 20 0C A1 B0  ..H...".#.S. ...
```

This cart has 32KiB of ROM and 8KiB of RAM, it uses I/O-1 as a mirror of the last page of cart RAM. It has the following registers in the I/O-2 range:

ROM config register at $DF00 (can only be written to):

```
bit  meaning
---  -------
0    ?
1    ? (write 1 to release freeze mode)
2    ROM bank select
3    write 1 to disable cartridge
4-6  unused
7    ?
```

Note: if bit0, bit1, bit7 are all 0, then ultimax mapping is selected and RAM is enabled at ROML, otherwise if bit 0 is 0, then 16KiB mapping is enabled, or if bit 0 is 1, then 8KiB mapping is enabled.

RAM config register at $DF01 (read/write):

If written value == last value - 1, then ultimax mapping is selected and RAM is enabled at ROML, if written value == last value + 1, then ROM is enabled at ROML and exrom is deasserted (switch to either 8KiB or 16KiB mapping)

$DF02-$DFFF holds the last page of the first 8KiB of the current bank.

17.14.3.42 41 - IEEE-488

Size	4KiB (1 bank of 4KiB)
EXROM	active (lo) (0)
GAME	inactive (hi) (1)
Load address	$8000-$8FFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 29 00 01 00 00 00 00 00 00  ...@...).......
0020: 56 49 43 45 20 43 41 52 54 00 00 00 00 00 00 00  VICE CART.......
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 10 10 00 00 00 00 80 00 10 00  CHIP............
0050: 09 80 7A 80 C3 C2 CD 38 30 8E 16 D0 20 84 FF 20  ..z....80... ..
```

The cart uses a TPI for the IEEE488 interface/communication in the I/O-2 range:

```
$DF00 - Port A Data
$DF01 - Port B Data
$DF02 - Port C Data
$DF03 - Port A Direction
$DF04 - Port B Direction
$DF05 - Port C Direction
$DF06 - Control register
$DF07 - Active Interrupt register
```

17.14.3.43 42 - Game Killer

Size	8KiB (1 bank of 8KiB)
EXROM	inactive (hi) (1)
GAME	active (lo) (0)

Load address $E000-$FFFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 2A 01 00 00 00 00 00 00 00  ...@...*........
0020: 56 49 43 45 20 43 41 52 54 00 00 00 00 00 00 00  VICE CART.......
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 20 10 00 00 00 00 E0 00 20 00  CHIP.. ....... .
0050: 00 25 08 CF 07 9E 32 30 38 38 20 4D 43 2E 43 52  .%....2088 MC.CR
```

When the cartridge is active, ultimax is enabled when the address being accessed is is the $E000-$FFFF range, so the ROM is visible at $E000, below is normal C64 RAM. The cart can be disabled by writing to either I/O-1 or I/O-2 range. When the freezer button is pressed, the cartridge will be enabled and an NMI will be triggered.

17.14.3.44 43 - Prophet64

Size 256KiB (32 banks of 8KiB)
EXROM active (lo) (0)
GAME inactive (hi) (1)
Load address $8000-$9FFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 2B 00 01 00 00 00 00 00 00  ...@...+........
0020: 56 49 43 45 20 43 41 52 54 00 00 00 00 00 00 00  VICE CART.......
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
0050: 09 80 09 80 C3 C2 CD 38 30 78 A0 00 84 F8 84 FA  .......80x......
```

The control register is the I/O-2 range:

```
bit  meaning
---  -------
7-6  unused
5    disable cart
4-0  bank select
```

17.14.3.45 44 - EXOS

Size 8KiB (1 bank of 8KiB)
EXROM inactive (hi) (1)
GAME active (lo) (0)
Load address $E000-$FFFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 2C 01 00 00 00 00 00 00 00  ...@...,........
0020: 56 49 43 45 20 43 41 52 54 00 00 00 00 00 00 00  VICE CART.......
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

```
0040: 43 48 49 50 00 00 20 10 00 00 00 00 E0 00 20 00  CHIP.. ....... .
0050: 85 56 20 0F BC A5 61 C9 88 90 03 20 D4 BA 20 CC  .V ...a.... .. .
```

This cart has 8KiB of ROM, mapped in at $E000-$FFFF only when hirom is selected. The cart uses a clip that needs to be installed inside the C64.

17.14.3.46 45 - Freeze Frame

Size	8KiB (1 bank of 8KiB)
EXROM	active (lo) (0)
GAME	inactive (hi) (1)
Load address	$8000-$9FFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 2D 00 01 00 00 00 00 00 00  ...@...-........
0020: 56 49 43 45 20 43 41 52 54 00 00 00 00 00 00 00  VICE CART.......
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
0050: 10 80 10 80 C3 C2 CD 38 30 20 00 00 00 00 00 00  .......80 ......
```

When reading from the I/O-1 range the cart is enabled, when reading from the I/O-2 range the cart is disabled. When the freeze button is pressed the ROM is mapped to both $8000-$9FFF and $E000-$FFFF.

17.14.3.47 46 - Freeze Machine

Size	16KiB or 32KiB (2 or 4 banks of 8KiB)
EXROM	active (lo) (0)
GAME	inactive (hi) (1)
Load address	$8000-$9FFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 2E 00 01 00 00 00 00 00 00  ...@............
0020: 56 49 43 45 20 43 41 52 54 00 00 00 00 00 00 00  VICE CART.......
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
0050: 3A 83 60 80 C3 C2 CD 38 30 20 00 00 40 00 00 00  :.‘....80 ..@...
...
2050: 43 48 49 50 00 00 20 10 00 00 00 00 A0 00 20 00  CHIP.. ....... .
2060: 78 A9 34 85 01 A0 00 B1 F8 91 F6 E6 F8 D0 02 E6  x.4.............
...
4060: 43 48 49 50 00 00 20 10 00 00 00 01 80 00 20 00  CHIP.. ....... .
4070: 3A 83 60 80 C3 C2 CD 38 30 20 00 00 40 00 00 00  :.‘....80 ..@...
...
6070: 43 48 49 50 00 00 20 10 00 00 00 01 A0 00 20 00  CHIP.. ....... .
6080: 78 A9 34 85 01 A0 00 B1 F8 91 F6 E6 F8 D0 02 E6  x.4.............
```

Warning, the following information is based on guess-work and might be incorrect, any further information and/or corrections are appreciated.

When reading from the I/O-1 range ROM bank 0(/2) is mapped to $8000-$9FFF and ROM bank 1(/3) is mapped to $A000-$BFFF. When reading from the I/O-2 range the cart is disabled. When a reset happens the ROM banks get switched and ROM bank 0(/2) is mapped to $8000-$9FFF. When a freeze happens ROM bank 0(/2) is mapped to both $8000-$9FFF and $E000-$FFFF.

17.14.3.48 47 - Snapshot 64

Size	4KiB (1 bank of 4KiB)
EXROM	inactive (hi) (1)
GAME	inactive (hi) (1)
Load address	$E000-$EFFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 2F 00 00 00 00 00 00 00 00  ...@.../........
0020: 56 49 43 45 20 43 41 52 54 00 00 00 00 00 00 00  VICE CART.......
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 10 10 00 00 00 00 E0 00 10 00  CHIP............
0050: 78 D8 48 8A 48 98 48 AC 0D DD 10 03 4C EE F2 AD  x.H.H.H.....L...
```

Warning, the following information is based on guess-work and might be incorrect, any further information and/or corrections are appreciated.

The cart has a control bit (bit 0) in the I/O-2 range which is used to disable or enable the cart.

17.14.3.49 48 - Super Explode V5.0

Size	16KiB (2 banks of 8KiB)
EXROM	active (lo) (0)
GAME	inactive (hi) (1)
Load address	$8000-$9FFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 30 00 01 00 00 00 00 00 00  ...@...0........
0020: 56 49 43 45 20 43 41 52 54 00 00 00 00 00 00 00  VICE CART.......
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
0050: D7 86 5E FE C3 C2 CD 38 30 A9 00 2C A9 FF 85 FE  ..^....80..,....
...
2050: 43 48 49 50 00 00 20 10 00 00 00 01 80 00 20 00  CHIP.. ....... .
2060: E8 96 5E FE C3 C2 CD 38 30 20 6C 81 A9 09 8D 99  ..^....80 l.....
```

Warning, the following information is based on guess-work and might be incorrect, any further information and/or corrections are appreciated.

The cart has 16KiB of ROM which are used as two banks of 8KiB, they are mapped into $8000-$9FFF and the last page of the current ROM bank is mirrored in $DF00-$DFFF. The cart has a control bit (bit 7) at $DF00, which is used to select what ROM bank is used.

17.14.3.50 49 - Magic Voice

Size	16KiB (2 banks of 8KiB)
EXROM	inactive (hi) (1)
GAME	active (lo) (0)
Load address	$8000-$9FFF (bank 1), $A000-$BFFF (bank 2)

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 31 01 00 00 00 00 00 00 00  ...@...1........
0020: 56 49 43 45 20 43 41 52 54 00 00 00 00 00 00 00  VICE CART.......
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
0050: EA 2C 80 DF 50 FB A0 00 8C 80 DF B9 E3 A3 29 0F  .,..P.........).
...
2050: 43 48 49 50 00 00 20 10 00 00 00 00 A0 00 20 00  CHIP.. ....... .
2060: 4A EB C0 49 6A EA BB FB 4E CA 43 1E 75 63 15 97  J..Ij...N.C.uc..
```

This cart has 16KiB of ROM, mapped in at reset at $8000-$BFFF. The cart is controled through a TPI at $DF80-$DF87:

```
$DF80 - Port A Data
$DF81 - Port B Data
$DF82 - Port C Data
$DF83 - Port A Direction
$DF84 - Port B Direction
$DF85 - Port C Direction
$DF86 - Control register
$DF87 - Active Interrupt register
```

The cart has a pass-through port and does the following at start-up:

- Program starts after reset at $FFD3, and copies code from $FF36-$FFD2 to $0200-$029C (157 bytes)
- Program continues at $021A, copies $A000-$BFFF from EPROM to RAM at $A000-$BFFF (8KiB), copies $E000-$FFFF from EPROM to RAM at $E000-$FFFF (8KiB), copies $AE62-$B461 from RAM to RAM at $C000-$C5FF (Magic Voice Code)
- Jump to beginning of Magic Voice code at $C000

17.14.3.51 50 - Action Replay 2

Size	16KiB (2 banks of 8KiB)
EXROM	active (lo) (0)
GAME	inactive (hi) (1)
Startup mode	8KiB Game
Load address	$8000-$9FFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 32 00 01 00 00 00 00 00 00  ...@...2........
0020: 56 49 43 45 20 43 41 52 54 00 00 00 00 00 00 00  VICE CART.......
```

```
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
0050: EA EA 68 AA 68 85 94 68 85 95 68 85 96 68 85 97  ..h.h..h..h..h..
...
2050: 43 48 49 50 00 00 20 10 00 00 00 01 80 00 20 00  CHIP.. ....... .
2060: 30 80 5E FE C3 C2 CD 38 30 20 04 90 4C 38 DF 1A  0.^....80 ..L8..
```

Warning, the following information is based on guess-work and might be incorrect, any further information and/or corrections are appreciated.

I/O-1 is somehow used to enable the cart, the exact way in which this is done is unknown. Reading from the I/O-2 range will give you the last page of the current ROM bank, and writing to it will disable the cart.

17.14.3.52 51 - MACH 5

Size	4KiB or 8KiB (1 bank of 4KiB or 8KiB)
EXROM	active (lo) (0)
GAME	inactive (hi) (1)
Startup mode	8KiB Game
Load address	$8000-$8FFF (4KiB), $8000-$9FFF (8KiB)

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 33 00 01 00 00 00 00 00 00  ...@...3........
0020: 56 49 43 45 20 43 41 52 54 00 00 00 00 00 00 00  VICE CART.......
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
0050: AF 83 5E FE C3 C2 CD 38 30 4D 41 43 48 35 A5 93  ..^....80MACH5..
```

This cart has 8KiB ROM mapped at $8000-$9FFF, the $9E00-$9EFF range is mirrored at $DE00-$DEFF and the $9F00-$9FFF range is mirrored at $DF00-$DFFF.

17.14.3.53 52 - Diashow maker

Size	8KiB (1 bank of 8KiB)
EXROM	active (lo) (0)
GAME	inactive (hi) (1)
Startup mode	8KiB Game
Load address	$8000-$9FFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 34 00 01 00 00 00 00 00 00  ...@...4........
0020: 56 49 43 45 20 43 41 52 54 00 00 00 00 00 00 00  VICE CART.......
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
0050: 09 80 09 80 C3 C2 CD 38 30 AD 11 D0 29 10 D0 62  .......80...)..b
```

Accessing I/O-1 (the software uses $DE00 only it seems) disables cartridge ROM. A reset enables 8KiB game mode and the ROM bank is mapped to $8000. A freeze causes ROM to be mapped to $8000.

17.14.3.54 53 - Pagefox

Size	64KiB (4 banks of 16KiB)
EXROM	active (lo) (0)
GAME	active (lo) (0)
Startup mode	16KiB Game
Load address	$8000-$BFFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 35 00 00 00 00 00 00 00 00  ...@...5........
0020: 56 49 43 45 20 43 41 52 54 00 00 00 00 00 00 00  VICE CART.......
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 40 10 00 00 00 00 80 00 40 00  CHIP..@.......@.
0050: 31 80 BB 0E C3 C2 CD 38 30 50 46 20 56 31 2E 30  1......80PF V1.0
...
4050: 43 48 49 50 00 00 40 10 00 00 00 01 80 00 40 00  CHIP..@.......@.
4060: A2 FE 9A 20 EC AE 20 82 80 20 74 86 20 A5 8B 4C  ... .. .. t. ..L
...
8060: 43 48 49 50 00 00 40 10 00 00 00 02 80 00 40 00  CHIP..@.......@.
8070: 5A 01 02 03 04 06 0A 0B 10 14 1E 28 3C 00 00 00  Z..........(<...
...
C070: 43 48 49 50 00 00 40 10 00 00 00 03 80 00 40 00  CHIP..@.......@.
C080: 1E 03 14 82 09 05 09 0F 0C 0D 0F 05 09 09 0B 0A  ................
```

This cart has 64KiB ROM (2 32KiB Eproms, mapped to $8000 and $A000 in 16KiB Game Mode), and 32KiB RAM (mapped to $8000 and $A000 in 16KiB Game Mode). The cart has 1 (write-only) bank control register which is located at $DE80 and mirrored throughout the $DE80-$DEFF range:

```
Bit 0: unused/don't care
Bit 1: Bank select: 0=upper, 1=lower (not correct ?!)
Bit 2: chip select 0
Bit 3: chip select 1
Bit 4: cartridge enable/disable: 0=enable, 1=disable
Bits 5-7: unused/don't care
```

Chip select combinations of 0/1 are:

```
00: Eprom "79"
01: Eprom "ZS3"
10: Ram
11: empty space (reading returns VIC-II data)
```

note: on the original hardware "disabling" the cartridge by setting bit 4 of the control register does NOT prevent write accesses to the cartridge RAM!. So to actually disable the RAM, it is suggested to write $FF to the register.

17.14.3.55 54 - Kingsoft

Size	24KiB (3 banks of 8KiB)
EXROM	active (lo) (0)

GAME	active (lo) (0)
Startup mode	16KiB Game
Load address	$8000-$9FFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 36 00 00 00 00 00 00 00 00  ...@...6........
0020: 56 49 43 45 20 43 41 52 54 00 00 00 00 00 00 00  VICE CART.......
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
0050: 09 80 A2 89 C3 C2 CD 38 30 20 D3 83 78 8D 00 DE  .......80 ..x...
...
2050: 43 48 49 50 00 00 20 10 00 00 00 01 80 00 20 00  CHIP.. ....... .
2060: 72 A2 05 A5 45 10 01 CA A5 46 10 02 CA CA 86 28  r...E....F.....(
...
4060: 43 48 49 50 00 00 20 10 00 00 00 02 80 00 20 00  CHIP.. ....... .
4070: 78 A2 FF 9A D8 A9 08 8D 16 D0 A0 00 98 99 02 00  x...............
```

This cart has 24KiB ROM (3 8KiB Eproms)

```
reading io1:
- switches to 16KiB game mode
- first eprom mapped to 8000 (ROML)
- second eprom mapped to A000 (ROMH)

writing io1:
- switches to ultimax mode _only_:
  - if 0xc000 > address >= 0x8000
  - if address >= 0xe000
  (meaning 0x0000-0x7fff and 0xc000-0xdfff gives normal c64 ram/io)
- first eprom mapped to 8000 (ROML)
- third eprom mapped to e000 (ROMH)
```

17.14.3.56 55 - Silverrock 128

Size	128KiB (16 banks of 8KiB)
EXROM	active (lo) (0)
GAME	inactive (hi) (1)
Startup mode	8KiB Game
Load address	$8000-$9FFF

```
       00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
       -----------------------------------------------  ----------------
00000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
00010: 00 00 00 40 01 00 00 37 00 01 00 00 00 00 00 00  ...@...7........
00020: 56 49 43 45 20 43 41 52 54 00 00 00 00 00 00 00  VICE CART.......
00030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
00050: 09 80 09 80 C3 C2 CD 38 30 78 A2 FF 9A D8 A9 00  .......80x......
...
```

```
02050: 43 48 49 50 00 00 20 10 00 00 00 01 80 00 20 00  CHIP.. ....... .
02060: 00 21 1D A9 90 1B 67 70 FD B0 04 C3 19 B9 11 2D  .!....gp.......-
...
04060: 43 48 49 50 00 00 20 10 00 00 00 02 80 00 20 00  CHIP.. ....... .
04070: 00 41 62 92 AD 71 32 87 08 20 BE 90 4C 36 8F 20  .Ab..q2.. ..L6.
...
06070: 43 48 49 50 00 00 20 10 00 00 00 03 80 00 20 00  CHIP.. ....... .
06080: 00 61 00 02 02 03 0D 40 82 15 D0 A1 08 40 84 00  .a.....@.....@..
...
08080: 43 48 49 50 00 00 20 10 00 00 00 04 80 00 20 00  CHIP.. ....... .
08090: 00 81 8A 85 0D F0 8A 28 83 F8 8A 00 8B 58 83 60  .......(.....X.‘
...
0A090: 43 48 49 50 00 00 20 10 00 00 00 05 80 00 20 00  CHIP.. ....... .
0A0A0: 00 A1 C2 E3 C4 86 32 14 00 C5 40 EA 13 CA CB CC  ......2...@.....
...
0C0A0: 43 48 49 50 00 00 20 10 00 00 00 06 80 00 20 00  CHIP.. ....... .
0C0B0: 00 C1 81 59 60 00 81 5D D9 58 5E EE 58 6E 3C 28  ...Y‘..].X^.Xn<(
...
0E0B0: 43 48 49 50 00 00 20 10 00 00 00 07 80 00 20 00  CHIP.. ....... .
0E0C0: 00 E1 0F BF 3D 56 00 7E 52 FD 50 03 AA 00 0D 40  ....=V.~R.P....@
...
100C0: 43 48 49 50 00 00 20 10 00 00 00 08 80 00 20 00  CHIP.. ....... .
100D0: 01 01 B9 D1 0D 15 89 55 65 45 41 C5 01 45 45 A8  .......UeEA..EE.
...
120D0: 43 48 49 50 00 00 20 10 00 00 00 09 80 00 20 00  CHIP.. ....... .
120E0: 01 21 0C C7 29 54 41 29 4D C5 06 24 C7 24 8F 81  .!..)TA)M..$.$..
...
140E0: 43 48 49 50 00 00 20 10 00 00 00 0A 80 00 20 00  CHIP.. ....... .
140F0: 01 41 EA 87 7A AF 95 67 BD F7 00 7D 6E C4 5D A6  .A..z..g...}n.].
...
160F0: 43 48 49 50 00 00 20 10 00 00 00 0B 80 00 20 00  CHIP.. ....... .
16100: 01 61 6A 92 6B 93 6B 94 6C 95 6D 96 6E 97 6E 98  .aj.k.k.l.m.n.n.
...
18100: 43 48 49 50 00 00 20 10 00 00 00 0C 80 00 20 00  CHIP.. ....... .
18110: 01 81 0E 83 0D 0B 0F 81 FF 28 28 1E E3 0A 14 62  .........((....b
...
1A110: 43 48 49 50 00 00 20 10 00 00 00 0D 80 00 20 00  CHIP.. ....... .
1A120: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
...
1C120: 43 48 49 50 00 00 20 10 00 00 00 0E 80 00 20 00  CHIP.. ....... .
1C130: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
...
1E130: 43 48 49 50 00 00 20 10 00 00 00 0F 80 00 20 00  CHIP.. ....... .
1E140: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Cartridge hardware is designed around a 128KiB ROM (PROM or Mask ROM) which is divided into 16 banks each of 8KiB. When adressing the onboard bank-switching logic the requested ROM bank is mapped at address range $8000-$9fff (8KiB)

The cartridge PCB layout was cost-optimized for mass-production purposes.

The address-/data-lines and bank-switching logic uses the closest address-/data-lines with shortest and most direct path/distance in order to avoid too much PCB re-routing and keep production costs as low as possible.

This means that the respective address-/data-lines from the cartridge port may not necessarily connect with the corresponding address-/data-line of the ROM according to the official specs of the ROM.

This has over the years given some headaches and invalid dumps when cartridge dumpers that wanted to dump the cartridge contents by de-soldering the ROM and dumping it using an EPROM programmer/reader.

The image extracted from such an operation would need transformation according to the actual cartridge PCB re-routing of address-/data-lines. Alternatively one can make an adapter that implements this applied re-routing of the address-/data-lines.

This mass-production cost optimization also results in an obscured bank-switching address pattern/values more info on that later.

[HWrev1]: The original "Hugo" PCB [HWrev1] is labeled "HUGO Copyright 1990" on both sides of the PCB. PCB production date is stamped on the back og PCB in the format: "YYMM" 1Mib 27C010 EPROM or PROM (typical Atmel) "Hugo" PCB [HWrev1] contains the following discrete logic IC components: 74LS00N (DIP) and 74LS237N (DIP) DIP pitch (pin spacing) 2.54mm (0.1 inch) Assembly/production facilities: Philips, Strandlodsvej (Amager), Copenhagen, Denmark

[HWrev1] Bank-switching pattern: In order for the rom contents to appear as a continuous memory layout the following ROM bank-switching pattern must be applied by writing to adress 0xDE00: Cartridge bank-switching values: 00 80 10 90 20 a0 30 b0 40 c0 50 d0 60 e0 70 f0 Writing the bank-switch value to any address in address-range 0xDE00-0xDEFF will work.

[HWrev2]: Revised PCB [HWrev2] for SMD mount is labeled "SO-A4-1" on both sides of PCB. 1Mib Mask ROM (SOIC-32) PCB contains the following discrete logic SMD IC components: 74HCT02T (SOIC-14) and 74HCT174T (SOIC-16) SOIC (pitch) pin spacing 1.27mm (SMD mounted) Assembly/production facilities: Sono Press, Germany.

[HWrev2] Uses alternative bank-switching pattern: Writing *ANY* value to address 0xDE0y (offset y) will select ROM bank y Cartridge still has 16 banks and the lower four bits in the address selects the right bank. Hence the valid address range is 0xDE00-0xDE0F

17.14.3.57 56 - Formel 64

Size	32KiB (4 banks of 8KiB)
EXROM	inactive (hi) (1)
GAME	active (lo) (0)
Startup mode	Ultimax
Load address	$E000-$FFFF

```
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
```

```
      -------------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 38 00 00 00 00 00 00 00 00  ...@...8........
0020: 56 49 43 45 20 43 41 52 54 00 00 00 00 00 00 00  VICE CART.......
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 20 10 00 00 00 00 E0 00 20 00  CHIP.. ....... .
0050: 48 A9 FC 0C A9 FE 8D C2 DF 68 48 C9 8C F0 1C C9  H........hH.....
...
2050: 43 48 49 50 00 00 20 10 00 00 00 01 E0 00 20 00  CHIP.. ....... .
2060: EA EA 58 48 A9 FC 8D C2 DF 4C 6E E2 EA EA A9 FA  ..XH.....Ln.....
...
4060: 43 48 49 50 00 00 20 10 00 00 00 02 E0 00 20 00  CHIP.. ....... .
4070: 4C 86 E4 A9 FE 78 8D C2 DF 60 78 8C C3 DF A9 FA  L....x...‘x.....
...
6070: 43 48 49 50 00 00 20 10 00 00 00 03 E0 00 20 00  CHIP.. ....... .
6080: EA EA 48 58 A9 FC 8D C2 DF 68 48 C9 46 D0 7E 68  ..HX.....hH.F.˜h
```

the following is a quick overview of how the cartridge works, as its a bit unusual and different from most other cartridges:

```
- 27256 EPROM (32KiB)
- 7430 TTL
- MC6821
- 1 button (reset)

rom bank 0x00 - 0x03 (0x04) 8192* 4 32KiB mapped to e000
MC6821 registers mapped to io2 at dfc0..dfc4

- press reset and hold delete to get the main menu
- press reset and hold control to skip cbm80 check
  (dont start additional cartridge)

- press RESTORE, then...
  - left arrow, return  show disk directory
  - delete, 1           load"*" from disk, run
  - f1/f2               ? (disk stuff?)
  - f3/f4               ? (disk stuff?)
  - f5/f6, q            enter monitor
  - f7/f8, 2            show drive error channel
  - control, cursor     back to basic

  - type "help" in basic to get a list of available commands

*** MC6821 Port usage

Port A (parallel cable to floppy drive):
dfc0 port a ddr
dfc1 port a (in/out)
```

```
Port B (controls banking)
dfc2 port b ddr ($7f)
dfc3 port b (out)

bit3    1 = rom at $e000 enabled, 0 = cartridge disabled
bit1-2  rom bank number
bit0    ?
```

17.14.3.58 57 - RGCD

Size	64KiB (8 banks of 8KiB)
EXROM	active (lo) (0)
GAME	inactive (hi) (1)
Startup mode	8KiB Game
Load address	$8000-$9FFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 39 00 01 00 00 00 00 00 00  ...@...9........
0020: 56 49 43 45 20 43 41 52 54 00 00 00 00 00 00 00  VICE CART.......
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
0050: 0C 80 0C 80 C3 C2 CD 38 30 31 30 34 8E 16 D0 20  .......80104...
...
2050: 43 48 49 50 00 00 20 10 00 00 00 01 80 00 20 00  CHIP.. ....... .
2060: B9 0A 72 AB 0B F0 08 C9 2F 0C 10 27 E8 EF 5A C5  ..r...../..'..Z.
...
4060: 43 48 49 50 00 00 20 10 00 00 00 02 80 00 20 00  CHIP.. ....... .
4070: C0 B4 6C A6 6A 0A 14 5E 65 AD 94 02 86 C8 30 1F  ..l.j..^e.....0.
...
6070: 43 48 49 50 00 00 20 10 00 00 00 03 80 00 20 00  CHIP.. ....... .
6080: A4 89 C6 3A C4 60 F0 10 4D F5 89 F0 13 6C E0 78  ...:.`..M....l.x
...
8080: 43 48 49 50 00 00 20 10 00 00 00 04 80 00 20 00  CHIP.. ....... .
8090: 00 0B 08 14 00 9E 32 30 36 31 00 00 00 A0 35 BA  ......2061....5.
...
A090: 43 48 49 50 00 00 20 10 00 00 00 05 80 00 20 00  CHIP.. ....... .
A0A0: 1B 4A 00 53 90 77 8E 1A 1D 94 00 A6 E2 50 29 11  .J.S.w.......P).
...
C0A0: 43 48 49 50 00 00 20 10 00 00 00 06 80 00 20 00  CHIP.. ....... .
C0B0: 14 F7 2E 1A 34 B4 60 53 07 88 C4 F0 21 F6 88 20  ....4.`S....!..
...
E0B0: 43 48 49 50 00 00 20 10 00 00 00 07 80 00 20 00  CHIP.. ....... .
E0C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

This cartridge type is very similar to the Magic Desk cart type: ROM memory is organized in 8KiB ($2000) banks located at $8000-$9FFF. Bank switching is done by writing the bank number to $DE00.

```
bit 0-2   bank number
bit 3     exrom latch (1 = cart disabled until reset or powercycle)
```

17.14.3.59 58 - RR-Net MK3

Size	8KiB
EXROM	active (lo) (0)
GAME	inactive (hi) (1)
Startup mode	8KiB Game
Load address	$8000-$9FFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 3A 00 01 00 00 00 00 00 00  ...@...:........
0020: 56 49 43 45 20 43 41 52 54 00 00 00 00 00 00 00  VICE CART.......
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
0050: 09 80 09 80 C3 C2 CD 38 30 78 A2 FF 9A D8 8E 16  .......80x......
```

This cartridge has a CS8900a based RR-Net compatible network interface, and on 8KiB ($2000) bank ROM located at $8000-$9FFF. The ROM can be switched on/off by writing to the IO1 space:

```
a write to $de80 enables the ROM
a write to $de88 disables the ROM
```

17.14.3.60 59 - EasyCalc

Size	24KiB (3 banks of 8KiB)
EXROM	active (lo) (0)
GAME	active (lo) (0)
Startup mode	16KiB Game
Load address	$8000-$9FFF (bank 1), $A000-$BFFF (banks 2/3)

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 3B 00 00 00 00 00 00 00 00  ...@...;........
0020: 56 49 43 45 20 43 41 52 54 00 00 00 00 00 00 00  VICE CART.......
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
0050: 59 89 BC FE C3 C2 CD 38 30 0B 80 B1 84 48 E6 84  Y......80....H..
...
2050: 43 48 49 50 00 00 20 10 00 00 00 00 A0 00 20 00  CHIP.. ....... .
2060: 04 0B 20 52 45 50 4C 49 43 41 54 45 20 FB 89 0F  .. REPLICATE ...
...
4060: 43 48 49 50 00 00 20 10 00 00 00 01 A0 00 20 00  CHIP.. ....... .
```

```
4070: 02 10 B5 00 C9 E3 D0 01 C8 C9 E2 D0 01 C8 C9 E0  ................
```

This cart uses 8KiB mapped in at $8000-$9FFF and 2 banks of 8KiB mapped in at $A000-$BFFF.

The bank at $A000-$BFFF is selected by bit 0 of the address of a write access to I/O-1.

```
A0 | bank
---------
 0 |  0
 1 |  1
```

17.14.3.61 60 - GMod2

Size	512KiB (64 banks of 8KiB)
EXROM	active (lo) (0)
GAME	inactive (hi) (1)
Startup mode	8KiB Game
Load address	$8000-$9FFF

```
       00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
       -----------------------------------------------  ----------------
00000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
00010: 00 00 00 40 01 00 00 3C 00 01 00 00 00 00 00 00  ...@...<........
00020: 56 49 43 45 20 43 41 52 54 00 00 00 00 00 00 00  VICE CART.......
00030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
00050: 09 80 09 80 C3 C2 CD 38 30 78 8E 16 D0 20 A3 FD  .......80x... ..
...
02050: 43 48 49 50 00 00 20 10 00 00 00 01 80 00 20 00  CHIP.. ....... .
02060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
...
7E430: 43 48 49 50 00 00 20 10 00 00 00 3F 80 00 20 00  CHIP.. ....?.. .
7E440: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

GMod2 (Individual Computers)

This cart uses 512KiB Flash ROM (29F040) in 64 banks, mapped in at $8000-$9fff and has a 2KiB serial EEPROM (m93C86).

```
io1
    - register at de00

    bit7   (rw)  write enable (write 1), EEPROM data output (read)
    bit6   (ro)  EXROM (0=active) and EEPROM chip select (1=selected)
    bit5-0 (ro)  rom bank  bit5 EEPROM clock bit4 EEPROM data input

see http://wiki.icomp.de/wiki/GMod2
```

17.14.3.62 61 - MAX Basic

Size	16KiB (ULTIMAX mode)
EXROM	inactive (hi) (1)
GAME	active (lo) (0)

Load address $8000-$9FFF (bank 1), $E000-$FFFF (bank 2)

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 3D 01 00 00 00 00 00 00 00  ...@...=........
0020: 4D 41 58 20 42 41 53 49 43 00 00 00 00 00 00 00  MAX BASIC.......
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
0050: 94 E3 7B E3 43 42 4D 42 41 53 49 43 30 88 41 87  ....CBMBASIC0.A.
```

This cartridge uses two 8KiB ROMs that contain BASIC and KERNAL for the MAX Machine. It also provides additional 2KiB RAM which will be mapped to $0800.

17.14.3.63 62 - GMod3

Size	2MiB/4MiB/8MiB/16MiB (256/512/1024/2048 banks of 8KiB)
EXROM	active (lo) (0)
GAME	inactive (hi) (1)
Startup mode	8KiB Game
Load address	$8000-$9FFF

```
       00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
       -----------------------------------------------  ----------------
00000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
00010: 00 00 00 40 01 00 00 3E 00 01 00 00 00 00 00 00  ...@...<........
00020: 56 49 43 45 20 43 41 52 54 00 00 00 00 00 00 00  VICE CART.......
00030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
00050: 09 80 09 80 C3 C2 CD 38 30 78 8E 16 D0 20 A3 FD  .......80x... ..
...
02050: 43 48 49 50 00 00 20 10 00 00 00 01 80 00 20 00  CHIP.. ....... .
02060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
...
001FF030: 43 48 49 50 00 00 20 10 00 00 00 FF 80 00 20 00 CHIP.. ....?.. .
001FF040: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
```

GMod3 (Individual Computers)

This cart uses 2MiB/4MiB/8MiB/16MiB Flash ROM (EN25QH128A(2T)) in 256/512/1024/2048 banks, mapped in at $8000-$9fff.

```
    io1
        - register at de00-de07 (write)

        To select the ROM bank, write to de00-7. The lower 3 bits of the address
        are the upper three bits of the bank, as in:

        ldx #bankhi  ; upper 3 bits of the bank number
        lda #banklo  ; lower 8 bits of the bank number
```

```
        sta $de00,x

        - register at de00/de08 (read)

        The current ROM bank can be read from de00/de08, as in:

        ldx $de08    ; upper 3 bits of the bank number
        lda $de00    ; lower 8 bits of the bank number

        - register at de08 (write)

        bit7   (w)  bitbang mode enabled
        bit6   (w)  EXROM (0=active)
        bit5   (w)  enable cartridge irq vectors

        - register at de00 (bitbang mode)

        bit7   (r)  EEPROM data output
        bit6   (w)  EEPROM chip select (0=selected)
        bit5   (w)  EEPROM clock
        bit4   (w)  EEPROM data input

    see http://wiki.icomp.de/wiki/GMod3
```

17.14.3.64 63 - ZIPP-CODE 48

Size	8KiB
EXROM	active (lo) (0)
GAME	inactive (hi) (1)
Load address	$8000-9FFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 3f 00 01 00 00 00 00 00 00  ...@...?........
0020: 5a 49 50 50 2d 43 4f 44 45 20 34 38 00 00 00 00  ZIPP-CODE 48....
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
0050: 11 de 7b e3 c3 c2 cd 38 30 41 80 00 00 03 81 31  ..?....80A.....1
```

```
    reading IO1 enables the cartridge ROM
    reading IO2 disables the cartridge ROM
    the second last page of the ROM is mirrored in IO1
```

17.14.3.65 64 - Blackbox V8

Size	32KiB or 64KiB
EXROM	active (lo) (0)
GAME	active (lo) (0)

Load address $8000-BFFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 40 00 00 00 00 00 00 00 00  ...@...@........
0020: 42 4c 41 43 4b 42 4f 58 20 56 38 00 00 00 00 00  BLACKBOX V8.....
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 40 10 00 00 00 00 80 00 40 00  CHIP..@.......@.
0050: 09 80 22 80 c3 c2 cd 38 30 8e 16 d0 20 a3 fd 20  ..?....80... ..
```

```
writing to IO2 sets the cartridge config:
A0 - EXROM
A1 - GAME
A2 - bank lsb
A3 - bank msb
```

17.14.3.66 65 - Blackbox V3

Size 8KiB
EXROM active (lo) (0)
GAME inactive (hi) (1)
Load address $8000-9FFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 41 00 01 00 00 00 00 00 00  ...@...A........
0020: 42 4c 41 43 4b 42 4f 58 20 56 33 00 00 00 00 00  BLACKBOX V3.....
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
0050: a7 85 2b 80 c3 c2 cd 38 30 8e 16 d0 20 a3 fd 20  ..+....80... ..
```

```
writing IO1 disables the cartridge ROM
writing IO2 enables the cartridge ROM
```

17.14.3.67 66 - Blackbox V4

Size 16KiB
EXROM active (lo) (0)
GAME active (lo) (0)
Load address $8000-BFFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 42 00 00 00 00 00 00 00 00  ...@...B........
0020: 42 4c 41 43 4b 42 4f 58 20 56 34 00 00 00 00 00  BLACKBOX V4.....
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 40 10 00 00 00 00 80 00 40 00  CHIP..@.......@.
```

```
0050: a7 85 2b 80 c3 c2 cd 38 30 8e 16 d0 20 a3 fd 20  ..+....80... ..
```

```
    reading IO1 enables the cartridge ROM
    reading IO2 disables the cartridge ROM
```

17.14.3.68 67 - REX RAM-Floppy

Size	8KiB
EXROM	active (lo) (0)
GAME	inactive (hi) (1)
Load address	$8000-9FFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 43 00 01 00 00 00 00 00 00  ...@...C........
0020: 52 45 58 20 52 41 4d 2d 46 4c 4f 50 50 59 00 00  REX RAM-FLOPPY..
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
0050: 09 80 09 fe c3 c2 cd 38 30 20 a3 fd 20 50 fd 20  .......80 .. P.
```

This cartridge contains an 8KiB ROM which contains the software, plus up to 256KiB RAM which can then be used to store up to 100 programs. The RAM is battery buffered, and the cartridge can be disabled with a switch on the board.

```
    dfa0    (write) selects RAM bank
    df50    (read) toggles RAM writeable
    dfc0    (read) toggles cartridge enable
    dfe0    (read) toggles RAM enable
```

17.14.3.69 68 - BIS-Plus Cartridge

Size	2, 4 or 8KiB
EXROM	active (lo) (0)
GAME	inactive (hi) (1)
Load address	$8000-9FFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 44 00 01 00 00 00 00 00 00  ...@...D........
0020: 42 49 53 2d 50 4c 55 53 00 00 00 00 00 00 00 00  BIS-PLUS........
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
0050: 09 80 5e fe c3 c2 cd 38 30 8e 16 d0 20 a3 fd a0  ..^....80... ...
```

This cartridge contains a 2, 4 or 8KiB ROM. The software is copied to RAM and then the cartridge disables itself by writing to IO1 (de00).

17.14.3.70 69 - SD-BOX

Size	128KiB

EXROM active (lo) (0)
GAME active (lo) (0)
Load address $8000-BFFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 45 00 00 00 00 00 00 00 00  ...@...E........
0020: 53 44 2d 42 4f 58 00 00 00 00 00 00 00 00 00 00  SD-BOX..........
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 40 10 00 00 00 00 80 00 40 00  CHIP..@.......@.
0050: 09 80 81 ea c3 c2 cd 38 30 a9 00 8d 00 de a9 40  .......80......@
```

This cartridge contains a 128KiB banked ROM with a filebrowser and some extra software on it. It also contains a sd2iec clone, which seems to be seperate and runs the original sd2iec firmware.

```
there is one register at de00:

bit 0-3     ROM bank
bit 4       reset SD
bit 5       CE ROM
bit 6       EXROM
bit 7       register enable

additionally there are 3 "ram cells" at de01-de03
```

17.14.3.71 70 - MultiMAX

Size 1MiB, 64*16KiB (ULTIMAX mode)
EXROM inactive (hi) (1)
GAME active (lo) (0)
Load address $8000-$9FFF (odd banks), $E000-$FFFF (even banks)

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 46 01 00 00 00 00 00 00 00  ...@...F........
0020: 4d 55 4c 54 49 4d 41 58 00 00 00 00 00 00 00 00  MULTIMAX........
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 40 10 00 00 00 00 80 00 40 00  CHIP..@.......@.
0050: 48 80 48 83 48 83 48 83 64 85 64 85 f5 87 aa 88  H.H.H.H.d.d.....
```

This cartridge contains 1MiB ROM organised in 64 16KiB banks. It operates in ultimax mode and contains all known ultimax releases. It also provides additional 2KiB RAM for MAX-Basic, which will be mapped to $0800.

```
there is one register mirrored in the entire IO1 space:

bit 7       when set, the register is disabled and can only be reenabled by reset
bit 0-5     select ROM bank 0-63
```

17.14.3.72 71 - Blackbox V9

Size	32KiB, 2*16KiB
EXROM	inactive (hi) (1)
GAME	active (lo) (0)
Load address	$8000-$9FFF $E000-$FFFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 47 00 00 00 00 00 00 00 00  ...@...G........
0020: 42 4c 41 43 4b 42 4f 58 20 56 39 00 00 00 00 00  BLACKBOX V9.....
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 40 10 00 00 00 00 80 00 40 00  CHIP..@.......@.
0050: 00 9a b6 fe c3 c2 cd 38 30 c3 ff 00 ff 00 ff 00  .......80.......
```

The cartridge starts in ultimax mode in the last bank. The IO1 area also contains a mirror of the second last ROM page.

```
    a register is mapped to IO1, which is changed by accessing an address in
    the IO space, the address bits are mapped like this:

    bit 7   -   bank    (inverted on write)
    bit 6   -   exrom
    bit 0   -   game
```

17.14.3.73 72 - Lt. Kernal Host Adaptor

Size	8KiB ROM, 16KiB RAM
EXROM	active (lo) (0) Initially
GAME	inactive (hi) (1) Initially
Load address	$8000-$9FFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 48 00 01 00 00 00 00 00 00  ...@...H........
0020: 4c 54 2e 20 4b 45 52 4e 41 4c 20 48 4f 53 54 20  LT. KERNAL HOST
0030: 41 44 41 50 54 4f 52 20 36 2e 32 20 52 4f 4d 00  ADAPTOR 6.2 ROM.
0040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP............
0050: 14 80 14 80 c3 c2 cd 38 30 00 38 37 30 30 30 30  .......80.......
```

The Lt. Kernal Host Adaptor (SCSI ID 7) boot rom looks on sector 0 of drive 0 (ID 0) to find the partition (LU or logical unit) information and DOS. If the drive is not found, it will start up in stock mode. See src/c64/cart/ltkernal.c for more details.

The Lt. Kernal Host Adaptor uses the following registers:

```
    $DF00    = MC6821 Port A: Data and DDR
    $DF01    = MC6821 Port A: Control
    $DF02    = MC6821 Port B: Data and DDR
    $DF03    = MC6821 Port B: Control
    $DF04..7 = LTK Port and Freeze state
```

MC6821 Port A connects to the SCSI data bus (inverted). Data is saved to hard disk inverted to save data processing time.

```
signal  meaning
------  -------
PA 7-0  /Data bus on SCSI (input/output)
CA2     Pulses SCSI ACK (low to high)
```

MC6821 Port B connects to mostly the SCSI control bus. PB2.6 controls write access to 16KiB SRAM. When low (0), no writes are permitted, when high (1) writes are permitted to $8000-$9FFF AND $E000-$FFFF. Initially, this signal is low, which keeps the boot ROM mapped to $8000-$9FFF (lower 4KiB for the C64, upper 4KiB for the C128). Once a high (1) is written, the boot ROM is mapped out permanently until a reset. CB2 controls which KERNAL is in place. A high (1) uses the stock KERNAL while a low (0) maps in one of the 8K RAMs as the KERNAL. The other RAM at $8000-$9FFF memory is ONLY mapped in when PB2.6 is high (1). The HIRAM line from the PLA goes to the adaptor so it can determine the proper write action to the main RAM or adaptor RAM. Writing a high (1) to CB2 while PB2.6 is low (0) causes a system reset. The stock KERNAL is copied and then patched by the LTK DOS on startup.

```
signal  meaning
------  -------
PB 7    /REQ on SCSI (input)
PB 6    SRAM control
PB 5    RST on SCSI (output)
PB 4    SEL on SCSI (output)
PB 3    /BUSY on SCSI (input)
PB 2    /CD on SCSI (input)
PB 1    /MSG on SCSI (input)
PB 0    /IO on SCSI (input)
CB2     KERNAL control
```

$DF04 - $DF07:

The LTK Port can be between 0 and 15. Only adaptors set to port 0 are allowed to change the host configuration. Any writes to this register while PB2.6 and CA2 are high (1), will result in the Lt. Kernal Host Adaptor being removed from the bus until a reset, essentially reconfiguring to a stock system.

```
bit  meaning
---  -------
3-0  LTK port number (input)
4    Freeze state (0: active, 1: inactive)
```

17.14.3.74 73 - RAMLink

Size	64KiB ROM, 8KiB RAM
EXROM	active (lo) (0) Initially
GAME	inactive (hi) (1) Initially
Load address	$8000-$BFFF and $E000-$FFFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
```

```
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 49 00 01 00 00 00 00 00 00  ................
0020: 43 4d 44 20 52 41 4d 4c 49 4e 4b 20 32 2e 30 31  CMD RAMLink 2.01
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 40 10 00 00 00 00 80 00 20 00  CHIP..........
0050: 00 00 00 00 00 00 00 00 ee 42 df 20 1c b9 ee 42  ................
...
2050: 43 48 49 50 00 00 20 10 00 00 00 01 80 00 20 00  CHIP..........
2060: 03 4c 75 9c 88 f0 01 88 8c 80 de a9 8d 20 38 8f  ................
...
4060: 43 48 49 50 00 00 20 10 00 00 00 02 80 00 20 00  CHIP..........
4070: 0a 17 d8 8d 64 8c be 00 ce 42 df 20 48 88 ce 42  ................
...
6070: 43 48 49 50 00 00 20 10 00 00 00 03 80 00 20 00  CHIP..........
6080: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
...
8080: 43 48 49 50 00 00 20 10 00 00 00 04 80 00 20 00  CHIP..........
8090: 40 48 20 c9 ff aa 68 90 01 8a 60 20 bd ff 20 64  ................
...
a090: 43 48 49 50 00 00 20 10 00 00 00 05 80 00 20 00  CHIP..........
a0a0: 85 56 20 0f bc a5 61 c9 88 90 03 20 d4 ba 20 cc  ................
...
c0a0: 43 48 49 50 00 00 20 10 00 00 00 06 80 00 20 00  CHIP..........
c0b0: 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
...
e0b0: 43 48 49 50 00 00 20 10 00 00 00 07 80 00 20 00  CHIP..........
e0c0: a2 ff 78 9a d8 a9 00 8d 00 ff a2 0a bd 4b e0 9d  ................
```

or

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 49 00 01 00 00 00 00 00 00  ................
0020: 43 4d 44 20 52 41 4d 4c 49 4e 4b 20 31 2e 34 30  CMD RAMLink 1.40
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 40 10 00 00 00 00 80 00 20 00  CHIP..........
0050: 52 41 4d 4c 49 4e 4b 20 44 4f 53 20 28 43 29 31  ................
...
2050: 43 48 49 50 00 00 20 10 00 00 00 01 80 00 20 00  CHIP..........
2060: 4e 54 41 58 20 45 52 52 4f 52 00 57 52 49 54 45  ................
...
4060: 43 48 49 50 00 00 20 10 00 00 00 02 80 00 20 00  CHIP..........
4070: aa 09 f9 10 ec b9 9e 1b ec ce 3f 70 8d 84 df a9  ................
...
6070: 43 48 49 50 00 00 20 10 00 00 00 03 80 00 20 00  CHIP..........
6080: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
```

```
...
8080: 43 48 49 50 00 00 20 10 00 00 00 04 80 00 20 00  CHIP............
8090: 40 48 20 c9 ff aa 68 90 01 8a 60 20 bd ff 20 64  ................
...
a090: 43 48 49 50 00 00 20 10 00 00 00 05 80 00 20 00  CHIP............
a0a0: 85 56 20 0f bc a5 61 c9 88 90 03 20 d4 ba 20 cc  ................
...
c0a0: 43 48 49 50 00 00 20 10 00 00 00 06 80 00 20 00  CHIP............
c0b0: 80 8d 3f de a9 1b 8d 32 de 8d 34 de a9 ee 8d 33  ................
...
e0b0: 43 48 49 50 00 00 20 10 00 00 00 07 80 00 20 00  CHIP............
e0c0: a2 ff 78 9a d8 a9 00 8d 00 ff a2 0a bd 4b e0 9d  ................
```

The RAMLink CRT file is 64 KiB and consists of eight 8 KiB banks. The first two are for the RAMLink DOS which is banked in from $8000-$BFFF (16 KiB). The third and forth are the configuration portion of the DOS which also banks in at $8000-$BFFF, however only the first 8 KiB is officially used. The banks are the 64 shadow kernal, 64 trap kernal, 128 shadow kernal, and 128 trap kernal respectively. The first bytes of the configuration DOS bank contain 16-bit checksums of various parts of the ROM. The precise mapping is specific to the particular version of the ROM as the code resides there. The ROMs verify these checksums on boot up. Should this fail, the firmware turns on the error LED and goes into an infinite loop. Since vice doesn't provide an error or activity LED for RAMLink, the system would just appear to hang.

The actual RAMLink appears to have the ability to upgrade to a 128 KiB ROM, but no firmware which utilizes it is known to exist.

The RAMLink uses many locations in IO2 primarily so that it can co-exist with other cartridges. Most of these registers are write only, that is to say, all that is required to activate them is a write; the contents of the write are not important. This technique minimizes the coding required to interact with the device. Although RAMLink has 2 pass-thru ports, they are not complete shorts to the cartridge bus, some of the control lines are altered based on the state of the device. Initially, RAMLink is disabled, the trap kernal is active and only one of its registers is visisble:

```
$DF7E    = Enable RAMLink registers and switch to shadow kernal
```

Once a write is done to $DF7E, all the other registers become active. The shadow kernal also switches in.

```
$DF20     = NMI Trap Enable? (not certain as to its function)
$DF21     = Mirror to $DF01 (for REU) when NMI trap is on
$DF22     = NMI Trap Disable? (not certain as to its function)
$DF40     = I8255A Port A (Main board and parallel interface)
$DF41     = I8255A Port B
$DF42     = I8255A Port C
$DF43     = I8255A Control
$DF60     = Enable DOS mapping (maps in $8000-$BFFF)
$DF70     = Disable DOS mapping
$DF7F     = Disable RAMLink registers and switch to trap kernal
            (also disables DOS mapping)
```

```
$DF80..9F = Switch SRAM page ($DF80 selects page 0,
            $DF81 selects page 1, $DF82 selects page 2, etc.)
$DFA0     = I8255A Port A (RAMCard 1; RAMCard 2 uses custom circuit
            but same interface)
$DFA1     = I8255A Port B
$DFA2     = I8255A Port C
$DFA3     = I8255A Control
$DFB0..BF = 72421 RTC interface (RAMCard 2 only)
$DFC0     = Maps in SRAM to $DE00-$DEFF
$DFC1     = Maps in RAMCard to $DE00-$DEFF
$DFC2     = Maps in RAM Port $DE00-$DEFF (GEORAM, RAMDrive)
$DFC3     = Maps in Pass-Thru to $DE00-$DEFF
```

Note that only address $DF40..42, $DFB0..BF present valid information when read. Addresses $DFA0..$DFA3 also require valid data when being written to. See src/c64/cart/ramlink.c for more details.

When kernal swiching, the ROMs change inbetween instructions. This is possible as the CPU does not cache any incoming instructions. The following is an example of a JSR to the trap kernal at $E000:

```
PC     TRAP         SHADOW       DETAILS
          KERNAL       KERNAL
-----  ---------    ---------    ----------------------------------------
$C000  JSR $E000    JSR $E000    Same code
$E000  STA $DF7E    STA $DF7F    In trap kernel, next instruction in shadow
$E003  INC $D021    INC $D020    Increment $D020
$E006  STA $DF7E    STA $DF7F    Switch back to trap on next instruction
$E009  RTS          NOP          Return to $C003
```

If the shadow kernal was active, a JSR to $E000 would result in $D021 being incremented and continuing on back to the NOP in the shadow kernal. The use of write only registers allows the RAMLink Kernal and DOS to perform ROM switching with minimal instructions and time.

The RAMLink includes two switches: "Enable/Disable" and "Normal/Direct". "Enable/Disable" controls whether or not RAMLink is active. In the "Disabled" state, control will be given back to the stock ROMs. "Normal/Direct" controls how the IO cartidge lines to the RAM Port (REU, GEORAM, RAMDrive) are passed. In "Normal" mode, these lines are connected only when the RAMLink registers are active. This prevents any unauthorized access to these devices while RAMLink code isn't running. "Direct" connects the IO lines all the time.

17.14.3.75 74 - H.E.R.O.

Size	32KiB (4 banks of 8KiB each)
EXROM	active (lo) (0)
GAME	inactive (hi) (1)
Startup mode	8KiB Game
Load address	$8000-9FFF

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
```

```
      ------------------------------------------------  ----------------
0000: 43 36 34 20 43 41 52 54 52 49 44 47 45 20 20 20  C64 CARTRIDGE
0010: 00 00 00 40 01 00 00 4a 00 01 00 00 00 00 00 00  ......J........
0020: 44 52 45 41 4e 20 48 45 52 4f 00 00 00 00 00 00  DREAN HERO......
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040: 43 48 49 50 00 00 20 10 00 00 00 00 80 00 20 00  CHIP.. ....... .
0050: 0b 80 bc fe c3 c2 cd 38 30 03 49 8e 16 d0 bd 00  .......80.I.....
..
2050: 43 48 49 50 00 00 20 10 00 00 00 01 80 00 20 00  CHIP.. ....... .
2060: 32 aa 5a aa 00 00 ff 00 90 d9 68 d9 00 00 90 05  2.Z.......h.....
..
4060: 43 48 49 50 00 00 20 10 00 00 00 02 80 00 20 00  CHIP.. ....... .
4070: ff 7f 7f ff ff 3f 00 00 00 00 00 c0 00 c0 ff 3f  .....?.........?
..
6070: 43 48 49 50 00 00 20 10 00 00 00 03 80 00 20 00  CHIP.. ....... .
6080: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ................
```

This cartridge type was found in an argentinian H.E.R.O. clone cartridge, ROM memory is organized in 8KiB ($2000) banks located at $8000-$9FFF. There appears to be one register mirrored over IO2 (the software uses $DFFF). Banks are selected by writing to the lower 2 bits of the register, bit 5 disables the cartridge.

17.15 The PSID image format for ripped SID tunes

This section describes the SID file format used for SID tunes in the HVSC (High Voltage SID Collection - http://hvsc.de). It is based mostly on Michael Schwendt's document that describes the file format and the PSID v2NG extensions described by Simon White and Dag Lem and was further extended by Wilfred Bos (PSID v3/v4, RSID v3/v4) and LaLa (assembled most of the following text) and last not least tweaked a bit by Groepaz to fit into the VICE documentation.

The original documentation maintained by the HVSC team can be found here: `http://www.hvsc.de/download/C64Music/DOCUMENTS/SID_file_format.txt`

SID files use the .sid file extension.

Since PSID v2 is simply an extension of PSID v1, PSID v2NG is an extension of PSID v2, RSID is a restricted version of PSID v2NG, PSID v3/v4 and RSID v3/v4 are extentions of PSID v2NG and RSID v2, all of the formats are discussed together below. RSID in specific is discussed in detail under the 'magicID' field description.

The information presented here targets programmers or other people with reasonable background. It is not suitable for newbies who have never before used a machine code monitor, a disassembler, or a hexadecimal editor.

17.15.1 The SID file header v1

The detailed structure of the SID header looks like the following. Header offsets are in hexadecimal notation. Other integer values are decimal unless explicitly marked otherwise. Any stored integer values are in big-endian format:

```
+00    magicID: ‘‘PSID’’ or ‘‘RSID’’
```

This is a four byte long ASCII character string containing the value 0x50534944 or 0x52534944. 'RSID' (Real SID) denotes that the file strictly requires a true Commodore-64 environment to run properly. 'PSID' files will generally run trouble-free on older PlaySID and libsidplay1 based emulators, too.

Some words about the Real C64 SID file format (RSID):

The RSID format was designed to contain tunes that are not PlaySID compatible, but strictly require a real C64 environment to run. Tunes that are multi-speed and/or contain samples and/or use additional interrupt sources or do busy looping will cause older SID emulators to lock up or play very wrongly (if at all).

By using the name RSID for such rips all existing SID emulators will reject these tunes safely until they can be upgraded to cope with the additional requirements.

Due to the nature of these tunes, every effort must be made to make sure they are directly runnable on an actual C64 computer. As such the tunes will only be presented with the default C64 power-on environment and expected to configure and use all hardware appropriately.

RSID is based on PSIDv2NG with the following modifications:

- magicID = RSID
- version = 2, 3 and 4 only
- loadAddress = 0 (reserved)
- playAddress = 0 (reserved)
- speed = 0 (reserved)
- psidSpecific flag is called C64BASIC flag

The above fields MUST be checked and if any differ from the above then the tune MUST be rejected. The definitions above will force tunes to contain proper hardware configuration code and install valid interrupt handlers.

See section "The SID file environment" below for the default C64 power-on environment for each SID file format.

`+04 WORD version`

Available version number can be 0001, 0002, 0003 or 0004. Headers of version 2, 3 and 4 provide additional fields. RSID and PSID v2NG files must have 0002, 0003 or 0004 here.

`+06 WORD dataOffset`

This is the offset from the start of the file to the C64 binary data area. Because of the fixed size of the header, this is either 0x0076 for version 1 and 0x007C for version 2, 3 and 4.

`+08 WORD loadAddress`

The C64 memory location where to put the C64 data. 0 means the data are in original C64 binary file format, i.e. the first two bytes of the data contain the little-endian load address (low byte, high byte). This must always be true for RSID files. Furthermore, the actual load address must NOT be less than $07E8 in RSID files.

You must be absolutely sure what to enter here. There is no way to detect automatically whether the first two bytes in a C64 data file are meant to be a load address or some arbitrary bytes of code or data. Unless your C64 file is not a normal binary file and thus

has no load address in front, you need not enter anything else than 0 here. The SID tune will not play if you specify a load address which is present in the C64 file already.

Normal C64 binary data files have a load address in their first two bytes, so they can be loaded to a pre-defined destination address by executing LOAD"FILE",8,1, for instance. If a load address is explicitly specified in the sidtune info file, some sidtune converters and utilities conjecture that the C64 data don't have a load address in their first two bytes. Hence, the explicit load address from the info file is moved in front of the C64 data to create a valid C64 binary file which can be easily loaded on a C64, too. If that C64 file were to be saved, it would contain two superfluous data bytes at offset 2 if an original load address had been in the first two bytes of the old file. This process of adding a duplicate load address can be repeated. The file loader strips off the first two bytes (the used load address) and puts the rest of the file contents (including the now obsolete load address at file offset 2) into memory. If the new load address is the same than the old one the two added bytes cause the whole data to be displaced by two bytes, which most likely results in malfunctioning code. Also, superfluous bytes in memory then can confuse disassemblers which start at the beginning of the file or memory buffer.

`+0A WORD initAddress`

The start address of the machine code subroutine that initializes a song, accepting the contents of the 8-bit 6510 Accumulator as the song number parameter. 0 means the address is equal to the effective load address.

In RSID files initAddress must never point to a ROM area ($A000-$BFFF or $D000-$FFFF) or be lower than $07E8. Also, if the C64 BASIC flag is set, initAddress must be 0.

`+0C WORD playAddress`

The start address of the machine code subroutine that can be called frequently to produce a continuous sound. 0 means the initialization subroutine is expected to install an interrupt handler, which then calls the music player at some place. This must always be true for RSID files.

`+0E WORD songs`

The number of songs (or sound effects) that can be initialized by calling the init address. The minimum is 1. The maximum is 256.

`+10 WORD startSong`

The song number to be played by default. This value is optional. It often specifies the first song you would hear upon starting the program is has been taken from. It has a default of 1.

`+12 LONGWORD speed`

This is a 32 bit big endian number.

For version 1 and 2 and for version 2NG, 3 and 4 with PlaySID specific flag (+76) set, the 'speed' should be handled as follows:

Each bit in 'speed' specifies the speed for the corresponding tune number, i.e. bit 0 specifies the speed for tune 1. If there are more than 32 tunes, the speed specified for tune 32 is the same as tune 1, for tune 33 it is the same as tune 2, etc.

For version 2NG, 3 and 4 with PlaySID specific flag (+76) cleared, the 'speed' should be handled as follows:

Each bit in 'speed' specifies the speed for the corresponding tune number, i.e. bit 0 specifies the speed for tune 1. If there are more than 32 tunes, the speed specified for tune 32 is also used for all higher numbered tunes.

For all version counts:

- A 0 bit specifies vertical blank interrupt (50Hz PAL, 60Hz NTSC), and a 1 bit specifies CIA 1 timer interrupt (default 60Hz).
- Surplus bits in 'speed' should be set to 0.
- For RSID files 'speed' must always be set to 0.

Note that if 'play' = 0, the bits in 'speed' should still be set for backwards compatibility with older SID players. New SID players running in a C64 environment will ignore the speed bits in this case.

WARNING: This field does not work in PlaySID for Amiga like it was intended, therefore the above is a redefinition of the original 'speed' field in SID v2NG! See also the 'clock' (video standard) field described below for 'flags'.

```
+16    ''<name>''
+36    ''<author>''
+56    ''<released>'' (once known as ''<copyright>'')
```

These are 32 byte long ASCII character strings. Upon evaluating the header, these fields may hold a character string of 32 bytes which is not zero terminated. For less than 32 characters the string should be zero terminated. The maximum number of available free characters is 32.

```
+76    <data>
```

Version 1 of the SID header is complete at this point. The binary C64 data starts here.

17.15.2 The SID file header v2, v3 and v4

Version 2, 3 and 4 of the header incorporates the v1 header fields and provides additional fields. Some of these are actually v2NG, v3 or v4 specific - those are noted below.

```
+76    WORD flags
```

This is a 16 bit big endian number containing the following bitfields:

- Bit 0 specifies format of the binary data (musPlayer): 0 = built-in music player, 1 = Compute!'s Sidplayer MUS data, music player must be merged.

If this bit is set, the appended binary data are in Compute!'s Sidplayer MUS format, and does not contain a built-in music player. An external player machine code must be merged to replay such a sidtune.

- Bit 1 specifies whether the tune is PlaySID specific, e.g. uses PlaySID samples (psidSpecific): 0 = C64 compatible, 1 = PlaySID specific (PSID v2NG, v3, v4) 1 = C64 BASIC flag (RSID)

This is a v2NG and RSID specific field.

PlaySID samples were invented to facilitate playback of C64 volume register samples with the original Amiga PlaySID software. PlaySID samples made samples a reality on slow Amiga hardware with a player that was updated only once a frame.

Unfortunately, converting C64 volume samples to PlaySID samples means that they can no longer be played on a C64, and furthermore the conversion might potentially break the non-sample part of a tune if the timing between writes to the SID registers is at all altered. This follows from the ADSR bugs in the SID chip.

Today, the speed of common hardware and the sophistication of the SID players is such that there is little need for PlaySID samples. However, with all the PlaySID sample PSIDs in existence there's a need to differentiate between SID files containing only original C64 code and PSID files containing PlaySID samples or having other PlaySID specific issues. As stated above, bit 1 in 'flags' is reserved for this purpose.

Since RSID files do not have the need for PlaySID samples, this flag is used for a different purpose: tunes that include a BASIC executable portion will be played (with the BASIC portion executed) if the C64 BASIC flag is set. At the same time, initAddress must be 0.

- Bits 2-3 specify the video standard (clock): 00 = Unknown, 01 = PAL, 10 = NTSC, 11 = PAL and NTSC.

This is a v2NG specific field.

As can be seen from the 'speed' field, it is not possible to specify NTSC C64 playback. This is unfortunate, since the different clock speeds means that a tune written for the NTSC C64 will be slightly detuned if played back on a PAL C64. Furthermore, NTSC C64 tunes driven by a vertical blank interrupt have to be converted to use the CIA 1 timer to fit into this scheme. This can cause severe problems, as the NTSC refresh rate is once every 17045 cycles, while the CIA 1 timer A is latched with 17095 cycles. Apart from the difference in timing itself, the SID ADSR bugs can actually break the tune.

The 'clock' (video standard) field was introduced to circumvent this problem.

- Bits 4-5 specify the SID version (sidModel): 00 = Unknown, 01 = MOS6581, 10 = MOS8580, 11 = MOS6581 and MOS8580.

This is a v2NG specific field.

- Bits 6-7 specify the SID version (sidModel) of the second SID: 00 = Unknown, 01 = MOS6581, 10 = MOS8580, 11 = MOS6581 and MOS8580.

If bits 6-7 are set to Unknown then the second SID will be set to the same SID model as the first SID.

This is a v3 specific field.

- Bits 8-9 specify the SID version (sidModel) of the third SID: 00 = Unknown, 01 = MOS6581, 10 = MOS8580, 11 = MOS6581 and MOS8580.

If bits 8-9 are set to Unknown then the third SID will be set to the same SID model as the first SID.

This is a v4 specific field.

The MOS6581 and the MOS8580 have three notable differences. First, combined waveforms are generally louder on a MOS8580, to the extent that some combinations that are clearly audible on a MOS8580 are completely silent on a MOS6581. Second, the internal DC levels in the MOS8580 are so small that software or hardware tricks must be used to play volume samples. Third, the MOS8580 analog filter has totally different characteristics from the MOS6581 analog filter.

To ensure that music specifically written for one of the two SID versions can be played back correctly, bits 4-9 in 'flags' are used as stated above.

- Bits 10-15 are reserved and should be set to 0.

`+78 BYTE startPage (relocStartPage)`

This is a v2NG specific field.

This is an 8 bit number. If 'startPage' is 0, the SID file is clean, i.e. it does not write outside its data range within the driver ranges. In this case the largest free memory range can be determined from the start address and the data length of the SID binary data. If 'startPage' is 0xFF, there is not even a single free page, and driver relocation is impossible. Otherwise, 'startPage' specifies the start page of the single largest free memory range within the driver ranges. For example, if 'startPage' is 0x1E, this free memory range starts at $1E00.

`+79 BYTE pageLength (relocPages)`

This is a v2NG specific field.

This is an 8 bit number indicating the number of free pages after 'startPage'. If 'startPage' is not 0 or 0xFF, 'pageLength' is set to the number of free pages starting at 'startPage'. If 'startPage' is 0 or 0xFF, 'pageLength' must be set to 0.

The relocation range indicated by 'startPage' and 'pageLength' should never overlap or encompass the load range of the C64 data. For RSID files, the relocation range should also not overlap or encompass any of the ROM areas ($A000-$BFFF and $D000-$FFFF) or the reserved memory area ($0000-$03FF).

`+7A BYTE secondSIDAddress`

This is a v3 specific field. For v2NG, it should be set to 0.

This is an 8 bit number indicating the address of the second SID. It specifies the middle part of the address, $Dxx0, starting from value $42 for $D420 to $FE for $DFE0). Only even values are valid. Ranges $00-$41 ($D000-$D410) and $80-$DF ($D800-$DDF0) are invalid. Any invalid value means that no second SID is used, like $00.

`+7B BYTE thirdSIDAddress`

This is a v4 specific field. For v2NG and v3, it should be set to 0.

This is an 8 bit number indicating the address of the third SID. It specifies the middle part of the address, $Dxx0, starting from value $42 for $D420 to $FE for $DFE0). Only even values are valid. Ranges $00-$41 ($D000-$D410) and $80-$DF ($D800-$DDF0) are invalid. Any invalid value means that no third SID is used, like $00. The address of the third SID cannot be the same as the second SID.

`+7C <data>`

Version 2, 3 and 4 of the SID header ends here. This offset is the start of the binary C64 data. See also 'loadAddress' for what the first 2 bytes of 'data' might indicate.

17.15.3 The SID file environment

Before the data of a SID file is loaded in memory of a C64, certain addresses and chips must be initialized in order to play the SID tune correctly.

For RSID and PSID files the following address must be set:

$02A6	depending on the PAL/NTSC flag in the SID file header, it is set to 0x01 for PAL and set to 0x00 for NTSC.

The default C64 environment for PSID files is as follows:

VIC	IRQ set to any raster value less than 0x100. Enabled when speed flag is 0, otherwise disabled.
CIA 1 timer A	set to 60Hz (0x4025 for PAL and 0x4295 for NTSC) with the counter running. IRQs active when speed flag is 1, otherwise IRQs are disabled.
Other timers	disabled and loaded with 0xFFFF.

When the init and play addresses are called the bank register value must be written for every call and the value is calculated as follows:

```
if   address < $A000  -> 0x37 // I/O, Kernal-ROM, Basic-ROM
else address < $D000  -> 0x36 // I/O, Kernal-ROM
else address >= $E000 -> 0x35 // I/O only
else                  -> 0x34 // RAM only
```

The default C64 environment for RSID files is as follows:

VIC	IRQ set to raster 0x137, but not enabled.
CIA 1 timer A	set to 60Hz (0x4025 for PAL and 0x4295 for NTSC) with the counter running and IRQs active.
Other timers	disabled and loaded with 0xFFFF.
Bank register	0x37

- A side effect of the bank register is that init MUST NOT be located under a ROM/IO memory area (addresses $A000-$BFFF and $D000-$FFFF) or outside the load image.
- Since every effort needs to be made to run the tune on a real C64 the load address of the image MUST NOT be set lower than $07E8.
- If the C64 BASIC flag is set, the value at $030C must be set with the song number to be played (0x00 for song 1).

18 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:

- **Martin Pottendorfer** Implemented the Gnome Port based on Oliver Schaertels GTK+ port. Added support code for internationalization based on gettext. Improved the *nix fullscreen support. Added multi- threaded GUI display for *nix. Translated the UI to German. Implemented the fliplists + UI (*nix).
- **Marco van den Heuvel** Translated the UI to Dutch. Made the internationalization support for the Win32 and Amiga ports. Wrote the GEO-RAM 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 DOS port. Maintains the QNX 4.x, QNX 6.x, Solaris, Openserver, Unixware, Minix 3.x, Amiga, Syllable and OS/2 binary ports. Maintains the Win64 and Open Watcom project files. Maintains the SDL port(s). Added new .crt support. Added new screenshot formats. Added new sound recording support. Added SIDcart support for xpet, xplus4 and xvic. 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 Sound Sampler and SFX Sound Expander support. Added PCI support to the Amiga and DOS ports. Rewrote the sound system into a modular one, added always mono and always stereo support for the sound output. Added the RTC system. Added digiblaster support. Added 3rd SID support. Added the 6309 CPU emulation. Added the 65(S)C02 emulation. Added the 65816 emulation. Added native screenshot (koala/doodle) support. Added 6502/6510/8500/8502 cpu port unused bit fading. Added vice.chm (windows), vice.guide (amiga), vice.hlp (windows), vice.inf (os/2), vice.pdf (generic) and vice.txt documentation generation. Added Android port based on 'AnVICE 1.0.5'. Improved/fixed the 'in-source' FFMPEG support and fixed it for msvc7.0 and up. Generalized the 'RS232 net' support to be able to be used on any network supporting arch. Started the SDL2 port. Added generic sampler input support. Added new joyport system and converted the joystick, mouse and lightpen code to use the new system. Added joyport attached rtc (bbrtc). Added joyport attached cardkey keypad. Added joyport attached coplin keypad. Added joyport attached atari cx21 keypad. Added joyport attached atari cx85 keypad. Added joyport attached paperclip 64 dongle. Added joyport attached rushware keypad. Added joyport attached 2/4 bit samplers. Added vic20 I/O-2 and I/O-3 RAM support. Added the c64 cp/m (z80) cartridge. Added IDE64 digimax short bus device emulation. Added tapeport system and the tapelog, cp-clockf83, tape-sense dongle, and (not yet working) dtl basic dongle devices. Added easy calc result cartridge emulation. Added hardware SID I/O access system which allows CW3, HardSID, ParSID and SSI2001 to work on AmigaOS, BeOS, DOS, SDL, *nix and windows with and without device drivers. Added clockport system for mmc64, mmcreplay, retroreplay and ide64. Added rrnet clockport device. Added mp3@64 clockport device. And lots of other fixes and improvements.

- **Fabrizio Gennari** Added some improvements to the DOS and GTK+ ports. Changed the Windows video to use GDI as fallback, making it compile without DX if needed. Fixed the t64 support. Added monitor window support using VTE to the GTK+ GUI. Made some monitor fixes. Fixed some tape code issues.
- **Groepaz** Added new more precise CRT emulation. Added support for the new cartridge system and many new cartridges. Fixed up parts of cartconv, c1541 and petcat. Added video to audio leak sound support. Improved the GTK3 GUI. Added x64(sc), x128, x64dtv, xplus4, xvic model selection system. Added KoalaPad emulation. Added keyrah keymaps. Added joystick keys mapping to the keymap system. CIA emulation improvements. Added basic support for compute gazette sidplayer files (mus/str) to vsid. Added new palette files. Added 'available features' code. Fixed/added some items in the OSX GUI. Added random tape wobble emulation. Added 64KiB RGCD cartridge emulation. Added psid v4 3sid support to vsid. Added single frame advance. Added drive RPM and wobble support. Added rrnet mk3 emulation. Added GMod2 and GMod3 cartridge emulation. Added xvic BehrBonz cartridge emulation. Added VICII VSP-bug emulation. Wrote alot of test programs. Updated this document after a long period of outdated mess. And various fixes and improvements.
- **Olaf Seibert** Contributed some PET, including PET DWW hires, Xaw, lightpen, hardware scaling, and disk drive patches. Added proper SuperPET support, including 6809/6309 CPU emulation. Maintains the Xaw UI. Added PET HRE (High Res Emulator) board emulation. Added the 2-chip colour board for the Universal PET mainboard. Added support for 'printer/plotter' Commodore 1520.
- **Marcus Sutton** Made some console, dialog and joystick fixes for the BeOS port. Maintains the BeOS port. Added some GTK2 fixes. Fixed some PET model selection issues. Revived the Windows NT Unicode port.
- **Kajtar Zsolt** Wrote the IDE64 interface emulation, FD2000/4000 drive emulation, SCPU64 emulation and alot of fixes. Improved the mouse support. Added drive burst modification support. Added 1541 drive sounds emulation. Improved c64 cart emulation. Added DAC high pass filtering. Added the xscpu64 emulator. Added scroll wheel support for the Micromys mouse emulation. Added Swiss ROM support to x128. Added IDE64 USB server emulation. Added 1540 drive emulation. Improved monitor support. Improved vdrive compatibility. Added godot screenshot support. And fixed various issues.
- **AreaScout** Fixed the SDL2 port. Revived the Android port. Maintains the SDL1, SDL2 and Android ports.
- **Bas Wassink** Fixed some gtk2/3 issues. Fixed t64 file handling. Fixed memory leaks. Added more doxygen documentation. Fixed c1541 issues. Updated the Linux-Native-Howto.txt file. Added autoconf, automake and yasm version checking. Fixed runtime linker path issues with *BSD. Fixed out-of-tree building. One of the driving forces behind the 'native' GTK3 port.
- **Michael C. Martin** One of the driving forces behind the 'native' GTK3 port.

Former/inactive team members:

- **BSzili** Provided many amigaos4 fixes. Maintained the amigaos based and derived ports.
- **Errol Smith** Improved VDC emulation. Improved MPS803 printer 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. Wrote large parts of the new VIC-II emulation used in x64sc, especially the dot clock domain emulation. Wrote many test programs for hardware analysis.
- **Andreas Matthies** Improved the datasette support, the VIC20 video emulation and some UI stuff in the Win32 and DOS ports. He also wrote the BeOS port and implemented video/audio capture support. Improved history recording/playback and implemented support for video recording and the netlink feature. Made the Win32 user changable keyboard shortcut system. Improved CIA and VIA emulation. Worked on x64sc, especially interrupt timing. Improved the FFMPEG support and started the 'in-source' FFMPEG support/merge. Wrote test programs. Various bug(fixe)s. ;-)
- **Ulrich Schulz** Maintains the Dingoo port(s).
- **Stefan Haubenthal** Added some Amiga fixes.
- **Thomas Giesel** Added new monitor commands, features and improvements.
- **Ingo Korb** Corrected block allocation and interleave for c1541/vdrive, added rudimentary xplus4 tape recording support, fixed some GTK2 issues, 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.
- **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). Improved 1541 drive rotation emulation. Worked on x64sc. Added RSID BASIC tunes support to vsid. Several ReSID fixes and improvements.
- **Christian Vogelgsang** Maintained 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.
- **Dag Lem** Implemented the reSID SID emulation engine and video hardware scaling.
- **Spiro Trikaliotis** Copyright © 2000-2011 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.
- **Hannu Nuotio** Copyright © 2007-2011 Implemented DTV flash emulation, DTV support in the monitor, large parts of the DTV VIC, 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. Implemented C64 cartridge snapshot support. Initiated and worked on all parts of implementing x64sc. Wrote test programs.
- **Andreas Boose** Copyright © 1998-2010 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 DOS, Win32 and *nix 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.

- **Tibor Biczo** Copyright © 1998-2010 Improved the Win32 port and plus4 emulation.
- **M. Kiesel** Copyright © 2007-2010 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.
- **Andreas Dehmel** Copyright © 1999-2007 Wrote the Acorn RISC OS port.
- **David Hansel** Copyright © 2003-2005 Wrote the Star NL10 printer driver, implemented IEC devices and improved the tape emulation.
- **Markus Brenner** Copyright © 2000-2004 Added VDC emulation to x128 and added support for some more cartridges.
- **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.
- **André 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 DOS port and the initial Win32 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:

- **Mikkel Holm Olsen** Copyright © 2009-2017 Provided the Danish user interface translations and fixed a few monitor bugs.

  Martin Pottendorfer Copyright © 2000-2017 Provided the German user interface translations.

  Manuel Antonio Rodriguez Bas Copyright © 2011-2017 Provided the Spanish user interface translations.

  Paul Dubé Copyright © 2004-2017 Provided the French user interface translations.

  Czirkos Zoltan Copyright © 2006-2017 Provided the Hungarian user interface translations.

Karai Csaba Copyright © 2006-2017 Provided the Hungarian user interface translations.

Andrea Musuruane Copyright © 2001-2017 Provided the Italian user interface translations.

Jesse Lee Copyright © 2011-2016 Provided the Korean user interface translations.

Marco van den Heuvel Copyright © 2005-2017 Provided the Dutch user interface translations.

Jarek Sobolewski Copyright © 2011-2017 Provided the Polish user interface translations.

Michael Litvinov Copyright © 2010-2017 Provided the Russian user interface translations.

Peter Krefting Copyright © 2000-2017 Provided the Swedish user interface translations.

Emir Akaydin Copyright © 2008-2017 Provided the Turkish user interface translations.

Documentation Team:

- **Daniel Kahlin**

  Groepaz

  Kajtar Zsolt

  Marco van den Heuvel

  Olaf Seibert

External contributors:

- **Alexis Ballier** Provided some ffmpeg fixes.
- **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 not have been possible.
- **Anthony J. Bentley** Provided some man page fixes.
- **Eliseo Bianchi** Provided the italian Amiga translations.
- **Enrico Bruttomesso** Improved some italian translations.
- **Frank Buss** Provide some midi fixes.
- **ck!** Provided a win32 cbm character font.
- **iAN CooG** Added win32 vsid GUI and contributed various patches.
- **Mike Dawson** Provided the GP2X port.
- **Hans Deragon** Added support for dead keys in X11.
- **Paul David Doherty** Wrote `zip2disk`, on which the Zipcode support in `c1541` is based.
- **Sven A. Droll** Added Supergrafik support to petcat.
- **Peter Edwards** Implemented the SDL UI slider control and fixed some GP2X/Dingoo SDL UI issues.
- **István Fábián** Contributed a initial patch with the more correct 1541 bus timing code and which gave us hints for to improving the 1541 emulation.
- **Daniel Fandrich** Contributed some disk drive patches.

- **Dirk Farin** Rewrote the MITSHM code.
- **Georg Feil** Added support for toggling CB2 sound output line in the PET emulator.
- **Peter Andrew Felvegi aka Petschy** Fixed a couple of bugs in the fast serial emulation.
- **Ricardo Ferreira** Contributed the `unlynx` and `system` commands in `c1541` and added aRts sound support.
- **Flooder** Provided parts of the Polish user interface translations.
- **Robert H. Forsman Jr.** Provided parts of the widget set for implementing the Xaw GUI.
- **Ian Gledhill** Added support for the catweasel.device driver.
- **Peter Gordon** Provided support for native AmigaOS4 compiling.
- **Richard Hable** Contributed the initial version of the REU emulation.
- **Shawn Hargreaves** Wrote Allegro, the graphics and audio library used in the DOS version.
- **Ville-Matias Heikkila** Rewrote the vic20 sound code.
- **David Holz** Provided a label file which gives the built-in monitor the labels for the C64.
- **Nathan Huizinga** Added support for Expert and Super Snapshot carts.
- **Derrick Inksley** Fixed loading of zip files with brackets ([]) in the filename for the windows port. Added drive selection functionality to the window sdl port. Some drive sound fixes. Fixed some windows drag'n'drop issues. Fixed some monitor code issues.
- **Craig Jackson** Contributed miscellaneous patches in the old X64 times.
- **Dirk Jagdmann** Wrote the Catweasel sound driver.
- **Uffe Jakobsen** Wrote the Silverrock cartridge emulation and fixed the ocean cartridge bank wrap. Fixed some network code issues. Fixed some monitor issues. Provided some SDL port fixes.
- **Lasse Jyrkinen** Contributed miscellaneous patches in the old X64 times.
- **Peter Karlsson** Provided the swedish UI translations in the past.
- **Greg King** Added a working RTC to the emulation of the IDE64 cartridge. Provided some vdrive fixes. Added Inkwell Systems lightpen support. Fixed some monitor code issues. Provided some windows port fixes. Provided some SDL port fixes.
- **Michael Klein** Fixed some MacOSX code issues. Contributed the ESD sound driver, basic support for the OPENCBM library and some other patches.
- **Frank König** Contributed the Win32 joystick autofire feature.
- **Bernd Kortz** Provided some fixes for ZETA and the ZETA binary package.
- **Bernhard Kuhn** Made some joystick improvements for Linux.
- **Alexander Lehmann** Added complete support for all the VIC20 memory configurations for the old VICE 0.12.
- **Ilkka "itix" Lehtoranta** Provided the routines for the cybergraphics support for the Amiga ports.
- **Magnus Lind** Atari ST mouse and Atari CX-22 trackball emulation and pixel aspect fixes. Improved the Amiga mouse emulation. Improved the vic20 sound output. Added windows POV hat support. Improved sound fragment size handling. Provided some windows fixes. Provided some gtk port fixes. Provided some SDL port fixes.

- **Lioncash** Provided some c1541 fixes. Fixed some 'geninfocontrib' helper tool issues. Fixed some 'embedded' build issues. Fixed some memory leaks.
- **Locnet** Made the initial android port of x64.
- **Wolfgang Lorenz** Wrote an excellent 6510 test suite that helped us to debug the CPU emulation.
- **lvd** Provided some monitor fixes.
- **Marko Mäkelä** Wrote lots of CPU documentation. Wrote the VIC Flash Plugin cartridge emulation in xvic. Wrote the Ultimem cartridge emulation in xvic.
- **mar77i** Fixed some resource handling issues.
- **Robert McIntyre** Bugged people enough to get the improved g64 support rolling, updated g64 support to allow variable-length tracks, and performed initial development of half-track support.
- **Robert W. McMullen** Provided parts of the widget set for implementing the Xaw GUI.
- **Jennifer Medkief** Was in charge of checking up on the GUIs for elements that are wrong, unaccessable, and missing.
- **Dan Miner** Contributed some patches to the fast disk drive emulation.
- **mjhn** Provided some *nix 'readme' fixes.
- **Moiree** Provided some command line fixes. Improved the build system. Fixed some vsid bugs. Fixed some SDL GUI issues. Fixed some documentation items.
- **Luca Montecchiani** Contributed a new Unix joystick driver.
- **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.
- **Roberto Muscedere** Improved support for REL files in vdrive. Added Lt. Kernal Host Adaptor, CMD HD, CMD RAMLINK, and CBM D9090/60 emulation. Added D90 support to vdrive.
- **Leandro Nini** Improved ReSID emulation. And fixed some c64dtv cpu opcode issues.
- **Tomi Ollila** Donated `findpath.c`.
- **Per Olofsson** Digitalized the C64 colors used in the (old) default palette.
- **Lasse Öörni** Contributed the Windows Multimedia sound driver
- **Stein Pedersen** Fixed windows midi driver for 64bit windows.
- **Helfried Peyrl** Supplied a patch that fixes REL file records larger 256 bytes when using vdrive.

  Christopher Phillips Fixed and improved some Mac OS X gui items.
- **Frank Prindle** Contributed some patches.
- **Giuliano Procida** Used to maintain the VICE `deb` package for the Debian distribution, and also helped proofreading the documentation.
- **Vesa-Matti Puro** Wrote the very first 6502 CPU emulator in x64 0.1.0. That was the beginning of the story. . .
- **Rami Rasanen** Rewrote the VIC20 sound code.
- **Peter Rittwage** Made 1541 GCR hardware tests.

- **David Roden** Fixed various issues related to ffmpeg settings.
- **Pablo Roldán** Contributed initial patch for VIC-II PAL-N model selection.
- **Mathias Roslund** Provided the AmigaOS4 port.
- **Gunnar Ruthenberg** Provided some VIC-II enhancements and improved the Win32 port.
- **Johan Samuelsson** Provided the Swedish Amiga translations.
- **Oliver Schaertel** Wrote the X11 full screen, parts of custom ROM set support and 1351 mouse emulation for unix.
- **Peter Schepers** Contributed a document describing the G64 image format.
- **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.
- **Heiko Selber** Contributed some VIC20 I/O patches.
- **John Selck** Improved the video rendering and added the fast PAL emulation. Implemented new color generation based on P. Timmermanns knowledge.
- **Chris Sharp** Wrote the AIX sound driver.
- **Andr351 "JoBBo" Siegel** Provided the native MorphOS icons.
- **Harry "Piru" Sintonen** Provided lots of fixes and improvements for the Amiga ports.
- **Manfred Spraul** Wrote the Win32 text lister.
- **Markus Stehr** Provided the MMC64 emulation.
- **Michael Steil** Provided some c1541 fixes.
- **Dominique Strigl** Contributed miscellaneous patches in the old X64 times.
- **Samuli Suominen** Fixed XShm includes for newer xextproto versions and updated libpng check for newer versions.
- **Steven Tieu** Added initial support for 16/24 bpp X11 displays.
- **Philip Timmermann** Did a lot of research about the VIC-II colors.
- **TMLPiper** Provided fixes for the osx port.
- **Brian Totty** Provided parts of the widget set for implementing the Xaw GUI.
- **Mustafa "GnoStiC" Tufan** Made improvements to the GP2x port.
- **Lionel Ulmer** Implemented joystick support for Linux and a first try of a SID emulation for SGI machines.
- **vonred** Added multiple monitor support to the windows port.
- **Krister Walfridsson** Implemented joystick and sound support for NetBSD.
- **webulator** Provided Win32 drag & drop support
- **Robert Willie** Added some additional commands to the fsdevice emulation.
- **Peter Weighill** Gave many ideas and contributed the ROM patcher.
- **Gerhard Wesp** Contributed the `extract` command in `c1541`.
- **Maciej Witkowiak** Did some IDE64 and C1541 fixes.
- **David Wood** Provided some monitor fixes.
- **Count Zero** Fixed some monitor issues.

- **Bjoern Odendahl** Created the new VICE logo and volunteered to create new icons.

(We hope we have not forgotten anybody; if you think we have, please tell us.)

The people around the world providing results from running our test programs on various machines deserve a special mention:

- **hedning** (Drean C64 PAL-N, various C64 PAL boxes)
- **Jason Compton** (Various C64 and C128 NTSC boxes)
- **The Woz** (Drean C64 PAL-N)
- **Thierry** (Drean C64 PAL-N)
- **MOS6569** (C64C PAL)
- **Mike** (VIC-20 PAL)
- **Wilson** (VIC-20 NTSC)
- **Vicassembly** (VIC-20 NTSC)
- **David "jbevren" Wood** (C64 NTSC-OLD)

Thanks also to everyone else for sending suggestions, ideas, bug reports, questions and requests. In particular, a warm thanks goes to the following people:

- **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**
- **David "jbevren" Wood**
- **Gary Glenn**

Last but not least, a very special thank to Andreas Arens, Lutz Sammer, Edgar Tornig, Christian Bauer, Wolfgang Lorenz, Miha Peternel, Per Håkan Sundell, David Horrocks, Benjamin Rosseaux and William McCabe for writing cool emulators to compete with. `:-)`

19 Copyright

- Copyright © 1993-1996 Jouko Valta

20 Contact information

20.1 VICE home page

You can find the latest news about VICE at the official VICE home page:

`http://vice-emu.sourceforge.net`

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.

20.2 How to send feedback

Before contacting us, have a look at this manual and see 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

- `https://sourceforge.net/p/vice-emu/bugs/`.

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-emu-mail@lists.sourceforge.net`) 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!). Be patient when your mail does not show up on the mailing list - it is moderated manually. Once your mail shows up you should be approved and be able to send mail without further delay.
- 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://vice-emu.sourceforge.net`);

- 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. It should be very easy to find them using your favourite search engine.
- 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!

20.3 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. Get it with the command:

  `'svn checkout svn://svn.code.sf.net/p/vice-emu/code/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;

Preferably post your patch on the tracker: `https://sourceforge.net/p/vice-emu/patches/`.

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.

Concept Index

Index of Resources

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

Index of Command-line options

Table of Contents