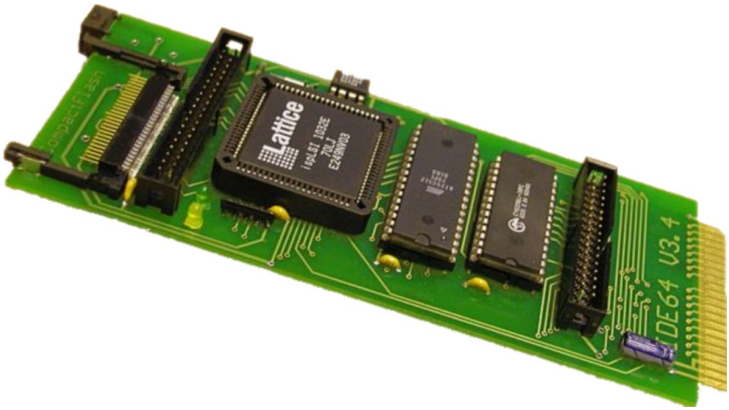
The IDE64 Project

IDE64
INTERFACE CARTRIDGE

user's guide

September 17, 2005 preview



for card versions V2.1, V3.1, V3.2, V3.4 and V4+
with **IDEDOS 0.9x (20050917)**!

THE ATA / ATAPI CONTROLLER CARD FOR COMMODORE 64 / 128 COMPUTERS
SUPPORTING HARD DISK, CDROM, DVD, ZIP DRIVE, LS-120 (A-DRIVE), COMPACTFLASH AND MORE

Document maintained by:

Kajtár Zsolt
Szigliget
Hóvirág u.15.
8264
Hungary
mail: soci at c64.rulez.org

Latest version of this document at:
http://singularcrew.hu/idedos/

Copyright © 2003-2005 Kajtár Zsolt (Soci/Singular).

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with no Invariant Sections, with the no Front-Cover Texts, and with no Back-Cover Texts. A copy of the license is included in the section entitled “20 GNU Free Documentation License”.

Foreword

This is the official user's guide for the IDE64 interface cartridge V2.1, V3.1, V3.2, V3.4 and V3.4+ with the upcoming **IDEDOS 0.9x**. Incomplete but planned parts are marked ~~this way~~.

This document always represents the actual state of development and the facts stated here may or may not apply to future or old versions of IDEDOS or the IDE64 cartridge. Please make sure you have the current version for your software and hardware!

It's recommended that you read all sections of this manual. For most of your questions the answers are somewhere in this text ;)

Disclaimer

All copyrights are held by their by their respective owners, unless specifically noted otherwise. Use of a term in this document should not be regarded as affecting the validity of any trademark or service mark. Naming of particular products or brands should not be seen as endorsements.

No liability for the contents of this document can be accepted. Use the concepts, examples and information at your own risk. There may be errors and inaccuracies that could be damaging to your system, although any damage is highly unlikely. Proceed with caution; the author(s) do not take any responsibility.

Contents

List of Tables

1 About the IDE64 interface cartridge

The IDE64 cartridge was created to provide the fastest I/O and biggest storage capacity available for the Commodore 64 and 128 computers, at cheap price. With this cartridge it's possible to connect and use ATA(PI) devices like hard disks, CDROMs, DVDs, CompactFlash cards, ZiP drives and LS-120 (A-Drive) just like ordinary disk drives. The IDE64 cartridge contains a 64 or 128 KiB PEROM (flash EPROM containing IDEDOS, Machine Code Monitor, File Manager and Setup), a 28 KiB RAM (used for internal buffers), a realtime clock chip (powered by a battery), two LEDs (to indicate the presence of cartridge and device activity), a Short-BUS for peripherals (like DUART Card, ETH64), a RESET button and an ispLSI chip.

The IDE64 cartridge is compatible with a wide variety of hardware including (but not limited to):

- Commodore serial bus drives, datassette
- CMD SuperCPU
- CMD FD2000 / FD4000 / HD
- JiffyDOS
- PAL / NTSC C64 or C128 in C64 mode
- REU
- Turbo232
- +60K
- 2nd SID

There are of course incompatible / unsupported hardwares (most notably cracking or fastload cartridges, RamLink, etc.), for more information read section 17 Compatibility at the end of the guide.

The architecture of the cartridge allows easy update of firmware, so it's possible to be up to date with the latest versions of IDEDOS.

IDEDOS can handle disks and partitions up to 137 GB (128 GiB). A disk can be divided into a maximal of 16 partitions. Files can be organized into a tree organized directory structure where each directory can contain 1023 files. Maximal file size is 4 GiB for regular files ~~and 16 MiB for relative files~~. Filenames can be 16 characters long plus a 3 character file type. Automatic file timestamping is supported.

IDEDOS 0.9x is free software, the source and the tools required to build the firmware are public and available for Linux and Win32. The code is licensed under the GPL-2:

> IDEDOS 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.
>
> IDEDOS 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

You are welcome to review the code and send suggestions, improvements or bugfixes if you want.

2 Set up and begin using the cartridge

2.1 Cabling, jumpers

Figure 1: IDE64 V3.4 with CompactFlash

Make sure that the computer is powered off. The cartridge must be plugged into the expansion port so that the chips are on top as seen on the picture. The ATA(PI) devices are connected with a 40 or 80 conductor IDE-cable to the port at the end of the cartridge. Make sure that the red line on the cable is at pin 1 on both ends. (pin 1 is usually near the RESET button at the right end of the cartridge, and near the power cord on the devices) Two devices can be connected on the cable, one is called master, the other slave. It's important that there's only one master and one slave on a cable, so check the jumper setting of your devices. Also select the IDE-DOS bank (on V3.4+ cartridge) while the computer is off.

2.2 Power supply

The external devices need external power too, so connect them to a pc power supply or something similar. AT-style power supplies will work without problems (some require a minimal load to start, so it may not work alone without any devices). ATX-style supplies require that the green wire is connected to a black

one, but do this at your own risk without connecting any devices to the supply to minimize possible damage. (if everything goes well, the cooler must spin up) First test your devices with the supply without connecting to the cartridge to see if everything is ok.

2.3 CompactFlash

CompactFlash card plugged into the V3.4 or V3.4+ version of the cartridge does not require any external power supply. Please note that the adapter is not hot plug capable, never change the CF card while the computer or the slave device is powered on! If you want to use an ATA(PI) device and CompactFlash card at the same time, then configure the device as slave, because the CF card is always master. When using the onboard CompactFlash connector and a slave device with a 80 conductor cable, make sure that the IDE64 cartridge and the slave device are connected to the device connectors, while the board connector with the longer part of the cable remains unconnected! (The required PDIAG signal is not connected to the board connector on 80 conductor cables, while on 40 conductor cables it is)

2.4 ZiP drive

When using an ATAPI ZiP drive it's important that it's jumpered as Master A or Slave A, otherwise it won't work. (old drives without A marking emulate a hard disk and do not need any special treatment) Note that not every ZiP-drive, cable and cartridge combinations work. It may be necessary to put the drive at the middle of the cable so that the remaining unconnected part of the cable is at least ca. 15cm long.

2.5 LS-120, A-Drive

The LS-120 (A-Drive) works fine with regular 720 KB, 1.2 MB and 1.44 MB floppy disks or with the 120 MiB superdisk. Disks of CBM 1581, CMD FD 2000 and CMD FD 4000 are not supported by the drive. Read the 3.4 Linear write max section before using it to avoid data corruption with some drive versions!

2.6 Peripherals

Peripherals (duart cartridge, ETH64, etc.) are connected to the other port (called Short Bus) of the cartridge with a 34 wire cable similar to the floppy cable in pcs, but without wire swapping. Peripherals do not require any external power. Never connect or disconnect peripheral devices while the computer is powered on!

2.7 Battery

There's a battery holder on the back of the cartridge. It's strongly recommended to put a battery into it (3 Volt) otherwise the setup settings can't be permanently stored and the file timestamping won't work. If using accu or super cap instead of battery, then enable recharging in the CMOS Setup utility.

2.8 Let's start

When everything is ready, you may turn on your equipment. It's recommended to first turn on the power supply, then the computer, however if you have devices that do not spin-up until the computer is

turned on and your AT-supply does not start without load (giving an annoying noise) you must do this the other way around. If your devices are not detected on power on, try giving HDINIT on the basic prompt. If still nothing, turn off your computer and power supply, then check the cables and jumper settings. Sometimes changing the master or slave configuration or the drives might help.

If everything is ok the boot screen should come up first (black screen with light-blue characters) with the information on the version of IDEDOS and the connected devices. Then the standard C64 reset screen should appear in less the 30 seconds, depending on the connected devices. Holding down CONTROL will hold the boot up screen, so it can be read. The boot screen only appears on power on, so if you want to auto detect your devices later to see everything is ok use HDINIT. If there's no boot screen just a standard C64 reset screen without the usual "`IDE-DOS xxxxx EXPANDED`" you may use the wrong version of IDEDOS. (eg. SuperCPU version on C64, C64 version on SuperCPU, config register version on old cartridge) If "`30719 BASIC BYTES FREE`" appears that means there's no valid IDEDOS burned into the perom of the cartridge. If the cartridge does no boot, and "`CORRUPT PEROM!`" message appears instead of regular boot, then consider rewriting your perom using the perom programmer utility. (read section 18 Updating IDEDOS)

To try the built in self test of IDEDOS hold down LEFTSHIFT while turning on the computer (or simply use SHIFTLOCK).

3 The CMOS Setup utility

The IDE64 cartridge has a battery backed up realtime clock with some memory (DS1302) used to store the configuration settings and current time. The cartridge of course works without a battery too, but then it's using the default settings and no clock. If you do not like these defaults, you can modify them in the CMOS Setup utility, and then save them.

The CMOS Setup utility is started by pressing ← + RESTORE while in interactive mode. (like STOP + RESTORE) Moving is done with cursor keys, RETURN selects or changes item, + and − also changes item. C= and STOP exits sub menu, while C= saves setting and STOP discards them in main menu.

Setup does not touch memory range $0800–$FFFF so your work won't lost when need to change some settings.

3.1 Standard setup

General settings. "CPULT" is the processor port leak time in 0.1 second resolution, depends on temperature and processor type.

Date, Time

Here's possible to set the built in clock. This has affect on the DATE command, the file timestamping and on TI$ (see 3.1). The calendar is built in and works in range 1980–2079. (Y2K compatible) The clock ticks while the computer is turned off. (but requires battery)

Figure 2: Setup screen

Start boot file

The file called "1//:BOOT,PRG" can be auto started from the boot drive at power up or always after reset. The file must be executable with RUN. Hold C= if you want to skip auto boot, or hold STOP to skip starting of program. The BASIC variable ST (at $90) contains 0 after power on, and 1 after reset.

Boot device

The default drive number (at $BA) after reset can be selected here. (useful for the dos wedge, monitor and auto booting) The file manager also seeks it's config file on this drive.

Figure 3: Standard setup

Disk fastloader

Use fastloader for the 1541, 1570, 1571 and 1581 floppy drives. (and also fast read and write in manager) The fastloader autodetects drive type, and is not used for unsupported drives. Also if the drive is capable of JiffyDOS fast protocol then the fastloader is automatically not used. Unfortunately some versions of 64HDD won't work unless it's turned manually off. (For more read section 17 Compatibility!)

Set basic clock

Sets the BASIC variable `TI$` to the time in the built in clock.

Keyboard repeat

Sets keyboard repeat after reset. (at $028A)

Lowercase chars

Select lower and uppercase chars or uppercase and graphics font after reset or STOP + RESTORE.

Use dos wedge

You may disable it, if you have JiffyDOS installed. For more read section 17 Compatibility!

C128 keyboard

If cartridge has cartconfig register (some V2.1, and any later version of the cartridge), support is compiled in, and computer is a C128, it makes use of extra keys on keyboard. For list of keys see Table 1!

Direct write

Enables or disables the use of "B=W". Cleared on reset. You may have to enable it when creating filesystem. Also if you have a corrupted disk, and cannot remove a directory with "RD:" because of a bad disk sector in the directory structure, then this option will make IDEDOS to forget about the error, and remove the directory, with everything in it. (dangerous, run fsck after this!)

Key	Function
ESC	`CHR$(27)`
TAB	`CHR$(9)`
ALT	-nothing-
HELP	Enter monitor
LINE FEED	`CHR$(10)`
NO SCROLL	-nothing-
CURSOR KEYS and NUM PAD	Their usual meaning

Table 1: C128 extra keys

Accu charging

Selects accu charging current. Leave it on `Battery` if not using accu or super cap. Charging a battery is useless, and some low quality products may even leak if charged.

CMD emulation

If enabled the returned ROM string for memory-read is different, and "/" is allowed in filenames which makes a difference in path handling. Also the partition listing is changed too.

Function keys

Enables function key support in direct mode. For list of default function key assignment see Table 2!

Key	Result	Function
F1	↑!∗,P + RETURN	LOAD"!∗,P" and RUN
F3	@$ + RETURN	List directory
F5	LI + RETURN	List program
F7	RU: + RETURN	Run program
F2	%:∗ + RETURN	LOAD":∗",DR,1
F4	@$∗=P	List programs in directory
F6	LL + RETURN	Detailed directory list
F8	MA + RETURN	Start manager
CONTROL + D	Device number	Change last used device

Table 2: Default function key assignment

Control stops

If enabled, scrolling will be stopped, instead of slowed down, when CONTROL is pressed. It may be easier to find things in a list, when it's not moving.

3.2 Color setup

The default colors of boot screen, manager and monitor can be changed here.

3.3 Device numbers

Device number mapping to drives. −− means device is disabled.

Figure 4: Device number setup

Serial device x

Device number mapping for serial devices. Selecting 9 for serial device 8, and 8 for serial device 9 will effectively swap them. (of course only if not using direct serial routines)

IDE x

Useful when programs think device number 8 is the only one.

Figure 5: Advanced setup

PC-LINK

Virtual drive. Needs serial or parallel cable and a server program on the PC. See section 14 PCLink!

3.4 Advanced setup

Master and Slave ATA(PI) device configuration for primary and secondary interfaces.

Power management

Drive spin down time from disabled to 2 hours in 15 steps. Default means do not touch drive's power management settings. (almost all devices support power management)

Retry on error

Tells the device to retry on error.

Write cache

Will speed up writes if supported by device.

Read look-ahead

Will speed up short sequential reads if supported by device. If you set "Linear read max" to it's maximum, this does not affect performance significantly, and it even slows down directory reads. (the look-ahead reads for random positioned directory sectors will flood the drive's cache with unneeded data, and take more time than the single sector reads)

Slow down CDROM

Selects $1\times$ speed on CDROMs and DVDs supporting this, reduces noise of faster than $24\times$ drives and gives faster access times. Has no effect on transfer rate on stock C64, because the cartridge transfers data much slower, however on SuperCPU this may limit reading speed. Also if you experience unreliable behaviour from the drive

(copying lot of files fail at different places due to spin up and downs) then disable this.

Linear read max

Maximal sequential read allowed by the device. (setting it lower than 128 will degrade the performance of CFS filesystem greatly!) You should decrease it from 128 if you get `?LOAD ERROR` when loading a big program. (this is very unlikely)

Linear write max

Same as above but for write operations. Some versions of LS-120 drives with 1.2 MB or 1.44 MB disk corrupt files if "Linear write max" is not set less than 12 (eg. 11)! It's dependent on the drive's firmware version and/or producer, so test your drive first! (With ZIP file extraction, 224 blocks program copy and load, etc.)

3.5 Restore defaults

Does restore the default settings.

3.6 Save & exit

Save changes and exit CMOS setup. (same as pressing C=)

3.7 Discard & exit

Forget changes made and exit setup. (same as pressing STOP) This does not apply to the realtime clock.

4 Preparing a blank disk

To use a new disk with IDE64 you must first create the filesystem on it. IDEDOS has it's own filesystem called CFS. It allows to use disks up to 128 GiB, files up to 4 GiB with holes and fast seeking, ~~relative files up to 16 MiB~~, 16 partitions, unlimited directories and files (only limited by the capacity of disk), customizable file types, and new file permissions.

To create the filesystem on a hard drive, CompactFlash, ZiP drive and LS-120 drive use the provided format utility. It allows to create partitions, change partition flags and create filesystems on them. Please note that you have to reformat these medias to IDEDOS's native filesystem in order to store data on them! Many disks come preformatted with some kind of DOS filesystem which is not suitable for CBM files (DOS is case insensitive, and can't store special characters or represent relative files).

First set the "Direct write" option in the setup utility to enabled, then start the format utility. Follow the instructions. Don't forget that formatting a partition will erase all data on it permanently!

CDROM device do not need any special treatment, just use ISO 9660 or Joliet format CDs to be able to read them. (Rock Ridge Extensions are not supported, these CDs will have short filename unless Joliet extension is also present) Multisession and mixed format disks are both supported. Everything after the last dot or after the last comma will be used as file type.

DVD drives work just like CDROM drives, DVDs written with the ISO9660/Joilet format will work, and ISO9660+UDF (most DVDs) will only have short filenames, as the UDF filesystem is not supported.

Floppy disks for LS-120 may require physical format before the creation of the filesystem. Use the format (N) command described in section 15 Command channel.

4.1 The format utility

The partitioner tool and the CFS filesystem creator is integrated into one executable called CFSfdisk. First it was a prototype utility running on GNU/Linux systems, then it was ported to C64 with the CC65 compiler.

Before partitioning or formatting go into setup and change `Direct write` to enabled. Then load and start CFSfdisk.

```
CFSfdisk version 9
Copyright (C) 2001-2004 Kajtar Zsolt
(Soci/Singular)

CFSfdisk comes with ABSOLUTELY NO
WARRANTY; for details see LICENSE.
This is free software, and you are
welcome to redistribute it
under certain conditions; see LICENSE
for details.

Drive number (4-30):
```

CFSfdisk is released under the GPL-2, the C source can be downloaded with the IDEDOS source. OK, first it needs to know what the drive's number is you want to format. It's usually 12 or 13, see your

setup settings on device number assignment for master and slave device. Remember that there will be no changes made on disk unless you exit with command “w”.

```
For LBA disks only the LBA sectors
matters, for CHS ignore LBA sectors.

Cylinders (1-65536, default 1244):
Heads (1-16, default 16):
Sectors (1-255, default 63):
LBA available (Y/N, default Y):
LBA (256-268435456, default 1253952):
```

Usually the auto detected defaults are ok, so you can just press enter for these questions. If not, enter the correct values.

```
Did not found any CFS partition entry in
PC BIOS partition table.
I assume you want to use the whole disk.
If you want to share the disk with other
operating systems, use the fdisk
utility, and make a partition entry
with type 0xCF.

Creating new disklabel.
```

It’s possible to share disk with other (non CBM) operating systems, and because there was no partition marked for IDEDOS usage, the disk will be repartitioned so that the whole disk will be used by IDEDOS. It you only want to use a part of the disk, then first create

a partition with type `0xCF`. (those familiar with the util-linux fdisk utility will know what I'm talking about for sure...) Anyway if you won't ever use this disk with anything else but a C64 then ignore the crap above ;-)

```
Creating new partition table.
```

A new empty C64 partition table was created. (inside the PC BIOS partition entry with type `0xCF`)

```
Command (m for help): m
Command action
   a   select boot partition
   b   change partition's name
   d   delete a partition
   g   set global disklabel
   h   toggle hidden flag
   l   toggle lbamode
   m   print this menu
   n   add new partition
   o   create an empty partition table
   p   print the partition table
   r   toggle writeable flag
   t   change partition's type
   q   quit without saving changes
   w   write table to disk and exit
```

Typing "m" list possibilities. There are no partitions yet, so let's create one.

```
Command (m for help): n
Partition number (1-16): 1
Start (4-1253952, default 4):
End or +x or +xG or +xM or +xK or +x%
(10-1253952, default 1253952): +50%
Partition's name: stuff
```

This creates partition 1 called "`stuff`" beginning on the start of the disk, and it will fill the half of the disk. (∼300 MiB) It's possible to give the exact start and end position or the size of the partition in sectors (eg. +2342) for power users. For everyday use +1048576K or +1024M or +1G creates an example partition with a size of 1 GiB. (metrics are powers of 1024, not 1000!) +25% means 1/4 of the maximal partition size possible in this extent.

After adding some more partitions, the partition table will look like this:

```
Command (m for help): p

Drive 12: 1253952 sectors (612 MiB)
Disklabel: "                "

Nr Boot Flags     Start       End
   Blocks   Id  System  Name
 1   *   --           4    626977
   313487    1  CFS     stuff
 2       --       626978    831777
   102400    1  CFS     work
 3       --       831778    864545
```

```
    16384    1  CFS     geos
 4       --      864546   1253952
   194703+   1  CFS     backup
```

Now let's change the global disklabel, and the default boot partition (I like to start at the work partition after boot). As you can see the listing above is not very readable as this program was designed for 80 column screen... The start and end of partition is displayed in sectors, while the partition size is displayed in KiB.

```
Command (m for help): g
New disklabel: soci's disk

Command (m for help): a
Partition number (1-16): 2

Command (m for help): p

Drive 12: 1253952 sectors (612 MiB)
Disklabel: "soci's disk     "

Nr Boot Flags     Start       End   Blocks   Id  System  Name
 1       --           4    626977   313487    1  CFS     stuff
 2   *   --      626978    831777   102400    1  CFS     work
 3       --      831778    864545    16384    1  CFS     geos
 4       --      864546   1253952   194703+   1  CFS     backup
```

Partition 3 will be used by GEOS (I hope the correct driver for GEOS will be available someday), so I'll change the partition type for it, and add the hidden flag, because it's not readable by IDEDOS anyway.

```
Command (m for help): t
```

```
Hex code (type L to list codes): l

 0  Empty
 1  CFS
 2  GEOS
Hex code (type L to list codes): 2

Command (m for help): h
Partition number (1-16): 3

Command (m for help): p

Drive 12: 1253952 sectors (612 MiB)
Disklabel: "soci's disk     "

Nr Boot Flags     Start       End   Blocks   Id  System  Name
 1       --           4    626977   313487    1  CFS     stuff
 2   *   --      626978    831777   102400    1  CFS     work
 3       H-      831778    864545    16384    2  GEOS    geos
 4       --      864546   1253952   194703+   1  CFS     backup
```

Now it's ready, let's format (or you can use "Q" to abort):

```
Command (m for help): w

Formatting partition 1...done.
Formatting partition 2...done.
Formatting partition 4...done.
Done.
```

The maximal formatting of a 137 GB partition takes ca. 28 mins. After filling partition 4 with lot of important stuff, let's mark it read only. Load the CFSfdisk utility and start it.

```
Command (m for help): r
Partition number (1-16): 4

Command (m for help): p

Drive 12: 1253952 sectors (612 MiB)
Disklabel: "soci's disk     "

Nr Boot Flags     Start       End   Blocks   Id  System  Name
 1       --           4    626977   313487    1  CFS     stuff
 2   *   --      626978    831777   102400    1  CFS     work
 3       H-      831778    864545    16384    2  GEOS    geos
 4       -R      864546   1253952   194703+   1  CFS     backup

Command (m for help): w

Done.
```

CFSfdisk auto detects CHS and LBA disks, so you don't have to worry about this (at least on C64). But if want to change it, use "L". The current setting is visible in the "`Drive xx:`" line, if there are numbers about cylinders, etc. then the disk is in CHS format.

If you only want to reformat a partition without deleting or re-adding it just change it's type to CFS even if was already in CFS format.

5 Using partitions

When using a new disk at least one partition must be created. Partitions provide the highest level of organizing data. Each partition has it's own filesystem, so possible disk or software errors can't destroy the whole data at once. It's also possible to select the default partition on boot, set the global disklabel, partition names and partition attributes (hidden and read only) with the cfsfdisk utility. The read only attribute is useful to prevent accidental changes to the partition.

Examples:

Listing available partitions. "∗" indicates default partition, "<" indicates read only partition. Hidden partitions are not listed.

```
@$=P
255 "SOCI'S DISK     " IDE64
1    "STUFF"           CFS
2    "WORK"           *CFS
4    "BACKUP"          CFS<
3 PARTITIONS.
READY.
```

Selecting partition 4 as working partition:

```
@CP4
02, PARTITION SELECTED,004,000,000,000
READY.
```

Selecting partition 1 as working partition the other way:

```
OPEN 15,12,15,"C"+CHR$(208)+CHR$(1):CLOSE15
READY.
```

Load a file from partition 2 from the directory "/GT". More about paths later in section 6.1 Paths.

```
LOAD"2//GT/:FILE"

SEARCHING FOR 2//GT/:FILE
LOADING $0801-$0932
READY.
```

6 Using directories

A directory is a list of files. To view the directory use LOAD"$" and LIST, DIR, or @$. Of course LOAD will overwrite the current program in memory, while the last 2 methods will preserve the computer's memory content.

The number before the first line is the partition number, it's followed by the directory label, and finally the ID string IDE64. The following lines provide informations about the size of each file in 256 byte blocks (so 4 blocks are exactly 1 KiB), the name of the file enclosed in quotation marks, and it's file type at the right side. The last line shows the total block count of all files in the list.

Example:

A simple directory list, using LOAD"$":

```
LIST

2 "TEST            " IDE64
0    "PLUGINS"          DIR
40   "BOOT"             PRG
2    "MAN"              USR
5    "TOD"              ASM
23   "VIEWER"           PRG<
70 BLOCKS USED.
READY.
```

It's possible to get a bit more detailed directory list, which is similar to the first one, but instead of the full filetype, only one letter is

Letter	Filetype
-	DEL entry
S	SEQ file
P	PRG file
U	USR file
R	Relative file
D	Directory
L	Link
?	Other

Table 3: Detailed directory filetypes

present. Then the timestamp of the file is displayed as month, day, hour, minute and the first letter of am/pm.

Example:

A detailed directory list, using LOAD"$=T":

```
LIST

2 "TEST            " IDE64
0    "PLUGINS"          D 10/14 08.16 P
40   "BOOT"             P 11/11 07.39 P
2    "CONFIG EXT"       U 11/15 09.48 A
5    "TOD"              ? 09/12 10.55 A
23   "VIEWER"           P 07/14 07.26 P
70 BLOCKS USED.
READY.
```

There's an even more verbose directory listing mode with dates,

which includes the full filetype, the protection flag, and the modification year too.

Example:

A more detailed directory list, using LOAD"$=T*=L":

```
LIST

2 "TEST            " IDE64
0    "PLUGINS"          DIR   10/14/04   08.16 PM
40   "BOOT"             PRG   11/11/04   07.39 PM
2    "CONFIG EXT"       USR   11/15/04   09.48 AM
5    "TOD"              ASM   09/12/04   10.55 AM
23   "VIEWER"           PRG<  07/14/04   07.26 PM
70 BLOCKS USED.
READY.
```

Putting a few thousand files in one long directory is not an optimal way of organizing data, so IDEDOS provides “subdirectories”. Subdirectories look like normal files with file type DIR in directory listings. Directories are organized into a tree like structure starting from the root directory. The root directory is the top level directory, it’s parent directory is itself. After boot this directory is selected as the working directory. The working directory is the directory which is used when no path is given in the filename just like in LOAD"$". Managing directories is done via channel #15 commands, but these examples will use the DOS Wedge to simplify things. (more about the channel #15 commands and the DOS Wedge later)

Examples:

Creating a subdirectory

```
@MD:DIRNAME
```

Changing the working directory

```
@CD:DIRNAME
```

```
@/DIRNAME
```

Changing the working directory to the parent directory

```
@CD←
@CD:..
```

Removing an empty subdirectory

```
@RD:DIRNAME
```

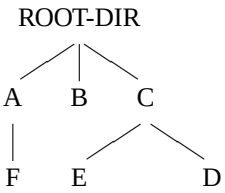
Changing the directory's label

```
@R-H:NEWHEADER
```

6.1 Paths

Each file in the tree structure can be reached with a "path". The path is composed from the name of the directories separated by a slash character.

Here's an example directory structure:



Let's say the working directory is "C" now. To load a file from directory "E" use the following: LOAD"/E/:FILE". This is a relative path. It's also possible to specify the location of a file from the root directory, it's called absolute path: LOAD"//C/E/:FILE". As it seems the path is enclosed between 2 slashes before a semicolon, after it is the name of file. In the last example the real path was "/C/E" where the slash before "C" indicates the it's an absolute path.

Now let's compose a relative path, but now our working directory is "F": LOAD"/../../C/E/:FILE". There's no directory named ". ." in the graph, as this is a special directory, and it means the parent directory. That's why "CD:.." means change to parent directory in some previous example. There's another special directory called "." which means the directory itself. These 2 special directories always exists.

To embed the partition number into the filename place it before the path, eg. "13/C/:E", and don't forget to include the colon!

Examples:

The directory structure above can be created by the following program:

```
10 OPEN 15,12,15
20 PRINT#15,"CD//"            :REM GOTO ROOT-DIR
30 PRINT#15,"MD:A"            :REM CREATE A
```

```
40 PRINT#15,"MD/A/:F"          :REM CREATE F
50 PRINT#15,"MD:B"             :REM CREATE B
60 PRINT#15,"CD:B"             :REM ENTER B JUST FOR FUN
70 PRINT#15,"MD//:C"           :REM CREATE C
80 PRINT#15,"MD//C/:E"         :REM CREATE E
90 PRINT#15,"MD/../C/./E/../:D":REM CREATE D ;-)
100 CLOSE 15
```

This small program lists all files in the working directory, and in it's subdirectories (recursive directory listing), by pharsing the formatted directory output (secondary address 0). The level of a directory in the tree is illustrated by the amount of tabulation before the file listing. Please note that this example does not work with serial devices or on PCLink, due to the illegal reuse of channel 0!

```
10 OPEN 15,12,15
20 A=A+1:OPEN A,12,0,"$":GET#A,A$,A$
30 GET#A,A$,A$,A$,B$:A$=A$+CHR$(0):B$=B$+CHR$(0):IF ST THEN 130
40 PRINT TAB(A*2-2);ASC(A$)+ASC(B$)*256;:B=1:D$="":N$=""
50 GET#A,A$:IF A$="" THEN 110
60 PRINT A$;:IF A$=CHR$(34) THEN B=B+1:GOTO 50
70 ON B GOTO 50,80,90
80 N$=N$+A$:GOTO 50
90 IF A$<>CHR$(32) THEN D$=D$+A$
100 GOTO 50
110 PRINT:IF D$="DIR" THEN PRINT:PRINT#15,"CD:"+N$:GOTO 20
120 GOTO 30
130 CLOSE A:A=A-1:IF A THEN PRINT#15,"CD"CHR$(95):PRINT:GOTO 30
140 CLOSE 15
```

6.2 Wildcards

It's possible to filter directory listings to show only a subset of files in a directory by using wildcards. There are 2 different wildcards:

"?" matches exactly one character, while "*" matches any number of characters. These wildcards can be used in path elements or filename too, in this case the first filename will be matched. To filter directory listing by file type, append "=TYP" to the pattern.

Wildcard filtering does only work on formatted directory lists (secondary address 0).

Examples:

List files starting with "A".

```
LOAD"$A*"

SEARCHING FOR $A*
LOADING $0801-$08DF
READY.
LIST

1 "P1CTURES        " IDE64
132  "ALIEN BAR BY WEC" FUN
29   "ALIEN-LIFE BY MB" SHI
132  "ALLEY BY MIKE"    FUN
132  "ANGEL BY FZ"      FUN
72   "ARNIE BY AMN"     DRL
497 BLOCKS USED.
READY.
```

List files ending with "DEPACKER".

```
@$*DEPACKER
```

```
1 "UT1LITIES       " IDE64
9    "DMC 5.0 DEPACKER" PRG
10   "DMC 5.1 DEPACKER" PRG
19 BLOCKS USED.
READY.
```

List files containing “PLAYER”.

```
@$*PLAYER*
```

```
1 "UTILIT1ES       " IDE64
0    "MUSIC PLAYER"     DIR
9    "CD PLAYER 1"      PRG
12   "CD PLAYER 3"      PRG
12   "CD PLAYER 4"      PRG
77   "CUBIC PLAYER"     PRG
110 BLOCKS USED.
READY.
```

List 4 character long filenames.

```
@$????
```

```
1 "UTIL1TIES       " IDE64
0    "DEMO"             DIR
0 BLOCKS USED.
READY.
```

List files starting with “K” and file type “D64”

```
@$K*=D64

1 "D3MOS           " IDE64
683  "KJU"              D64
683  "KRESTAGE 2"       D64
768  "KRESTOLOGY 1"     D64
768  "KRESTOLOGY 2"     D64
683  "KRESTOOLS"        D64
3585 BLOCKS USED.
READY.
```

6.3 Raw directory access

Ever wondered how to access creation date or attributes of files just like the BASIC command LL command does? It's done by using raw directory access!

Raw directory access gives lowlevel access to the filesystems directory structure, so it's format is not uniform across devices.

To open a raw directory channel, use secondary address 2–14. The first character for IDE64 devices is always "I" ($49). Then the 32 byte directory entries follow, in format of the CFS filesystem's directory entry layout. The end of list can be detected by the status variable. (6th bit set, end of file)

7 Using files

Files store programs and data in a filesystem. Traditionally files could only be accessed sequentially like on tape, but later relative files appeared with fixed record lengths on disk drives. Using IDEDOS with the CFS filesystem it's now possible to use randomly accessible files up to 4 GiB without the fixed record length limitation. ~~Relative files are also available for compatibility reasons, these are limited to 16 MiB.~~ There's also a special purpose file called link, which can be used to reference other files.

At maximum there can be 32 files open. (this includes all files, even on non-IDE64 drives) IDEDOS has 10 buffers for it's own files, these are shared between the IDE64 devices. IDEDOS supports opening of multiple files for writing too, but keep in mind that each file locks a 2 MiB part of the partition, which means that on partitions smaller than 2 MiB (like on a 1.44 MB disk) there can be only one file opened for write at the same time, and the next file open for write or create attempt will fail with a disk full error. This should be no problem on big partitions except when the partition is near full.

Opened files are locked, they cannot be removed until they are not closed. This applies to directories too, the current working directory cannot be removed even if it's empty.

In the format description everything between "*{*" and "*}*" is optional, and "*<name>*" means a parameter.

7.1 SAVE

The basic command SAVE (and also the similar kernal call) has the following format:

Format:

SAVE"*{{@}{<partition #>}{<path>}:}<filename>{,<type>}"{,<device #>}*

Wildcards ("?" and "*") and special characters like ":", "," and "=" are not allowed in filename. The file type has the same limitations plus it can't contain space, ">" and "<". The resulting file will be deletable, readable, writable, loadable, and not hidden. The directory, the partition and the disk must be writable where the file is created, and the file must be deletable if replaced.

Examples:

Simple save into current drive, current partition and into working directory, the result is "A" with file type "PRG".

```
SAVE"A"
```

This will save file "B" with file type "DAT" on drive 12 to partition 3 to directory "/A". If it's exists it will be first removed.

```
SAVE"@3//A/:B,DAT",12
```

This overwrite the first file in working directory beginning with "A". File type is not changed. Wildcards are only allowed if replace is used! ("@:")

```
SAVE"@:A*"
```

7.2 LOAD

The basic command LOAD has the following format:

Format:

LOAD"*{{<part #>}{<path>}:}<fname>{,<type>}"{,<device #>{,<mode #>}}*

The file must be loadable. (the executable flag must be set) For directory list load the directory must be readable.

Examples:

Simple load from current drive, current partition and from working directory. As file type is not specified the first will be loaded. The starting address will be ignored. (no mode specified means mode=0)

```
LOAD"A"
```

This will load file "B" with file type "PIC" from drive 13 from partition 4 from directory "/A". It will be loaded to the load address included in file (first 2 bytes), because mode≠0.

```
LOAD"4//A/:B,PIC",13,1
```

This will load the directory listing containing only "PRG" files as a basic program.

```
LOAD"$*=P",12
```

7.3 OPEN

Things are getting complicated here. The usual syntax of OPEN is as follows:

Format:

OPEN *<file #>{,<device #>{,<channel #>{,"<text>"}}}*

The file number identifies a file for IDEDOS, it must be in the range 1–127. If it's 128–255 then BASIC adds a CHR$(10) after each line. (it's an extra linefeed for certain printers)

The device number identifies the drive. There's a nice table at page 126 which lists valid device numbers.

The channel number identifies the communication channel within a device for a certain file, so it must be unique within one device. Channel numbers 0, 1 and 15 have special meaning, everything grater than 15 is illegal. Some commands use this channel number to identify the file. (like the position command, or direct access commands)

7.3.1 Opening a direct channel

The usage of direct channel is described in section 8 in detail. The format of open is:

Format:

OPEN *<file #>,<device #>,<2–14>,*"#"

Reading the last byte of the buffer will set the end of file bit in ST.

7.3.2 Opening a formatted directory list

Formatted directory list is a list of files in a directory formatted as a BASIC program.

Format:

OPEN *<file #>,<device #>*,0,"$*{<pattern>}*"

Reading the last byte of the directory list will set the end of file bit in ST.

Example:

This small program prints the directory list.

```
10 OPEN2,12,0,"$":GET#2,A$,A$
20 GET#2,A$,A$:IF ST THEN CLOSE2:END
30 GET#2,A$,B$:REM SIZE
40 PRINTASC(A$+CHR$(0))+ASC(B$+CHR$(0))*256;
50 GET#2,A$:PRINTA$;:IF A$ THEN 50
60 PRINT:GOTO20
```

7.3.3 Opening a raw directory list

Similar to the formatted directory, but gives all possible information about a file. It's not device independent.

Format:

OPEN *<file #>,<device #>,<2–14>*,"$"

Reading the last byte of the directory list will set the end of file bit in ST.

7.3.4 Opening a regular regular file for read

One way is to use secondary address 0, which means read only sequential access. If read only random access is required, then secondary address 2–14 must be used.

If read only random access is required, then the secondary address 2–14 must be used. The read operation is requested by the ",R" after the filetype, but it's optional.

Format:

OPEN *<file #>*,*<device #>*,0,"*{{<part #>}{<path>}:}<fname>{,<type>}*"
OPEN *<f #>*,*<d #>*,*<2–14>*,"*{{<part #>}{<path>}:}<fname>{,<type>{,*R*}}*"

The opened file have to be readable and must exists.

Example:

This small program prints the starting address of a file, and also has some nice error checking.

```
0 OPEN 15,12,15:CLOSE 15
5 IF ST THEN PRINT"DEVICE NOT PRESENT":END
7 OPEN 15,12,15
10 OPEN 2,12,0,"FILE,PRG":N$=CHR$(0)
20 GET#2,A$,B$:S=ST:REM SAVE STATUS
30 INPUT#15,A,A$,B,C,D,E:IF A=0 THEN 50
40 PRINT"DISK ERROR:"A;A$;B;C;D;E:GOTO 90
50 IF S=66 THEN PRINT"TOO SHORT!":GOTO 90
60 IF S=64 THEN PRINT"START ONLY?":GOTO 80
70 IF S THEN INPUT#15,A,A$,B,C,D,E:GOTO 40
80 PRINT"START:"ASC(A$+N$)+ASC(B$+N$)*256
90 CLOSE 2:CLOSE 15
```

7.3.5 Creating and replacing regular a file

If only one output file with write only sequential access is required, then the simplest is to use secondary address 1.

If write only random access is required, then the secondary address 2–14 must be used. The write operation is requested by the ",W" after the filetype, it's mandatory.

Format:

OPEN *<file #>*,*<dev #>*,1,"*{{@}{<part #>}{<path>}:}<fname>{,<type>}*"
OPEN *<f #>*,*<d #>*,*<2–14>*,"*{{@}{<part #>}{<path>}:}<fname>*,*<type>*,W"

The specified file shouldn't exists, unless replace is used (@), then the original file will be removed before the creation of the new one. The file name and type can't contain any special characters, just like for SAVE.

Example:

This small program prints the starting address of a file.

```
10 OPEN2,12,1,"@:FILE,SEQ"
20 PRINT#2,"HELLO";
30 IF ST THEN PRINT"ERROR DURING WRITE"
40 CLOSE2
```

7.3.6 Opening a regular file for append

Appending to a file means write only random access starting from the end of the file. The append operation is requested by the ",A" after the filetype, it's mandatory.

Format:

OPEN *<file #>*,*<dev #>*,*<2–14>*,"*{{<part #>}{<path>}:}<fname>*,*<type>*,A"

The file must be read and writable, and must exists.

7.3.7 Opening a regular file for read and write

If a file have to be both read and writable with random access use this opening mode. This operation is requested by the ",M" after the filetype, it's mandatory.

Format:

OPEN *<fl #>*,*<dev #>*,*<2–14>*,"*{{<part #>}{<path>}:}<fname>*,*<type>*,M"

The file must be read and writable, and must exists.

7.3.8 Opening the command channel

The command channel is identified by channel number 15.

Format:

OPEN *<file #>*,*<device #>*,15*{*,"*<command>*"*}*

Reading the command channel returns the status message from the drive, writing to it will send commands.

7.4 CLOSE

An opened file must be closed after use. It's especially true when writing new data into a file, where data can be left in the buffer unwritten, so recent changes could be lost forever! For a newly created file this can mean the entire file, for appended ones the part beyond the previous end of file.

Close only requires a file number.

Format:

CLOSE *<file #>*

7.5 File operations

File operations are done through command channel commands, these are described in detail in section 17.

Remove a file:

```
@S:FILE=TYP
```

Rename or move a file:

```
@R:NEW,TXT=OLD,SEQ
```

Change flags:

```
@L:NAME,TYP
```

8 Direct access

By using direct access commands you can read and write any sector of disk. Common use of direct access commands is to access unknown filesystems, manage the partition table and to create the filesystem. It's also important in disk editor applications.

To use direct access, you need to open 2 channels, one for commands, and one for data. The command channel can be opened with the usual OPEN <LFN>,<DEVICE>,15. The data channel is opened similar to opening normal files, except that the file name must be a hash sign. (OPEN <LFN>,<DEVICE>,<CHANNEL>,"#") The channel number must be greater than 1. This "file" will be a circular data buffer holding 512 or 2048 bytes depending on the device used. The end of buffer can be found by checking the BASIC variable ST.

8.1 BLOCK-READ

The block-read command reads the specified sector into the buffer and sets the buffer pointer to the start of buffer. It can also be used to get the ATA(PI) device identity information to find out the geometry, model number, etc. as described in ATA(PI) standards.

8.1.1 Reading from a CHS-ATA device

Format:

"B=R" + CHR$(*channel #*) + CHR$(*head #*) + CHR$(*cylinder bits 8...15 #*) + CHR$(*cylinder bits 0...7 #*) + CHR$(*sector #*)

8.1.2 Reading from a LBA-ATA or ATAPI device

Format:

"B=R" + CHR$(*channel #*) + CHR$(64 + *LBA bits 24...27 #*) + CHR$(*LBA bits 16...23 #*) + CHR$(*LBA bits 8...15 #*) + CHR$(*LBA bits 0...7 #*)

8.1.3 Getting device identity

Format:

"B=R" + CHR$(*channel #*) + CHR$(0) + CHR$(0) + CHR$(0) + CHR$(0)

8.2 BLOCK-WRITE

The block-write command writes the buffer content to the specified sector and sets the buffer pointer to the start of buffer.

8.2.1 Writing to a CHS-ATA device

Format:

"B=W" + CHR$(*channel #*) + CHR$(*head #*) + CHR$(*cylinder bits 8...15 #*) + CHR$(*cylinder bits 0...7 #*) + CHR$(*sector #*)

8.2.2 Writing to a LBA-ATA or ATAPI device

Format:

"B=W" + CHR$(*channel #*) + CHR$(64 + *LBA bits 24...27 #*) + CHR$(*LBA bits 16...23 #*) + CHR$(*LBA bits 8...15 #*) + CHR$(*LBA bits 0...7 #*)

8.3 BUFFER-POINTER

The buffer pointer selects individual bytes in the buffer. Reading and writing will start on the buffer position and the buffer pointer will be incremented by the number of bytes read and written.

8.3.1 Selecting a byte

Format:

"B=P" + CHR$(*channel #*) + CHR$(*position bits 0...7 #*) + CHR$(*position bits 8...15 #*)

8.4 TOC-READ

The TOC (Table Of Contents) holds additional information about the disk. The various formats of TOC is not described here, it can be found in the SCSI-MMC standard.

8.4.1 Reading TOC from a CD

Format:

"B=T" + CHR$(*channel #*) + CHR$(*format #*) + CHR$(*starting track #*)

Examples:

Bit	Meaning
6...7	00 — Formatted TOC 01 — Multi-session info 10 — Raw 11 — Reserved
5	Reserved
2...4	000 — Bits 6...7 select format 001 — Multi-session info 010 — Raw 011 — PMA 100 — ATIP 101 — CD-TEXT 110 — Reserved 111 — Reserved
1	0 — LBA 1 — MSF (Minute, Second, Frame)
0	Unused

Table 4: Bits of TOC format

This example reads in the boot sector and prints out the CFS identification string from the first sector of disk.

```
10 LB=0:REM LB=64 FOR ATAPI OR LBA DEVICE!
20 OPEN15,12,15:OPEN4,12,4,"#"
30 PRINT#15,"B=R"CHR$(4)CHR$(LB)CHR$(0)CHR$(0)CHR$(1)
40 PRINT#15,"B=P"CHR$(4)CHR$(8)CHR$(0)
50 FORA=0TO15:GET#4,A$:PRINTA$;:NEXT
60 CLOSE4:CLOSE15
```

This example prints some information from the device using the device identity.

```
10 OPEN15,12,15:OPEN4,12,4,"#"
20 PRINT#15,"B=R"CHR$(4)CHR$(0)CHR$(0)CHR$(0)CHR$(0)
30 PRINT#15,"B=P"CHR$(4)CHR$(46)CHR$(0)
40 PRINT"FIRMWARE REVISION:":B=4:GOSUB90
50 PRINT"MODEL NUMBER:":B=20:GOSUB90
60 PRINT#15,"B=P"CHR$(4)CHR$(20)CHR$(0)
70 PRINT"SERIAL NUMBER:":B=10:GOSUB90
80 CLOSE4:CLOSE15:END
90 FORA=1TOB:GET#4,A$,B$:PRINTB$A$;:NEXT:PRINT:RETURN
```

8.5 SUB-CHANNEL-READ

This command can be used to read the current state of audio playing, the current position, media catalog number, and the ISCR. For more information read the SCSI-MMC standard.

Bit	Meaning
7	Unused
6	0 — Header only 1 — Sub channel information
2...5	Unused
1	0 — LBA 1 — MSF (Minute, Second, Frame)
0	Unused

Table 5: Bits of sub-channel read format

Value	Meaning
0	Reserved
1	CD current position
2	Media Catalog number (UPC/bar code)
3	International standard recording code (ISRC)
4...255	Reserved

Table 6: Sub-channel read modes

8.5.1 Reading sub-channel information

Format:

"B=S" + CHR$(*channel #*) + CHR$(*format #*) + CHR$(*starting track #*) + CHR$(*mode #*)

Byte	Meaning
0	Reserved
1	Audio status, see table 8
2	Sub-channel data length high
3	Sub-channel data length low

Table 7: Sub-channel read data header

Value	Meaning
$00	Audio status byte not supported or not valid
$11	Play operation in progress
$12	Play operation paused
$13	Play operation successfully completed
$14	Play operation stopped due to error
$15	No current audio status to return

Table 8: Sub-channel read audio status codes

Byte	Meaning
0...3	Sub-channel data header, see table 7
4	Sub-channel data format code ($01)
5	ADR and control, see table 10
6	Track number
7	Index in current track
8...11	Absolute CD-ROM address in LBA or MSF
12...15	Track relative CD-ROM address in LBA or MSF

Table 9: Sub-channel CD current position data format

Bit	Meaning
4...7	ADR, equals to $1
3	0 — Two-channel audio 1 — Four-channel
2	0 — Audio track 1 — Data track
1	0 — Digital copy prohibited 1 — Digital copy permitted
0	0 — Audio without pre-emphasis 1 — Audio with pre-emphasis

Table 10: Sub-channel control field of CD current position

9 The File Manager

The IDE File Manager is a software for copying, deleting, renaming, making directories, and starting programs. Working with this program is very easy. Start it from BASIC typing command MAN. Remember that this will destroy the program in memory!

The screen is divided into 2 parts called "panel"s, each containing a directory listing, with the maximal number of displayable files of 510/panel. The black line is the file selector, which can be moved by the CRSR and Fx keys. Selecting the active panel is done by the CONTROL key. The right side of each panel shows the sum of selected files block count, the number of selected files, the used or free blocks in directory or disk and the drive number with the current partition. The current directory path is displayed on the top of screen, while the drive type and disklabel is on top of each panel. The directories and files are displayed with different color for easier recognition.

To change the current drive of a panel press the C= key together with a number key, and it will select drives 10, 11, ..., 17, 8, and 9. Directory reload is done with the 1 key. Copying files is done with key 5, while deleting is done with 8. Both operations can work on multiple files and on subdirectories. File rename is done with the 6 key, this can also change the filetype on IDE64 devices. To send commands to ordinary disk drives (eg. for formatting or validating a disk) press 9. To jump quickly on a file press it's first letter. Directory creation is done with the key 7. You can also eject the disk from a CDROM, LS-120 or ZiP drive by pressing the key ↑. Reloading of the disk is automatically done when reloading the directory. To exit to BASIC press ←.

Tagging files for operations with multiple files is done with the

DEL key. The + key selects all files and the − deselects all except directories. The key ∗ inverts the current selection. These three keys can be combined with SHIFT to include directories too. If no files are selected then copying and deleting will operate on the file where the selector currently is.

Entering directories and loading files are done with the RETURN key. If a file is associated with a plugin then the plugin will be started instead. Going into the parent directory is done by pressing RETURN on the directory entry ". .". Or you can also use the key "." for entering the parent directory and key "/" to enter the root directory. Partition selection is similar like entering the parent directory, but you have to be in the root directory. (press key "/" and then "." to quickly reach it) The partition selection directory can be identified by the partition number 255 displayed on the side. (and of course there's no ". ." entry)

The manager remembers the last used directories, partitions and drive numbers across invocations unless the computer is turned off.

If you do not like the default colors then it's possible to change it in the CMOS setup utility. If you do not like the lowercase charset (eg. want to look at a nice directory art), use the well known C= + SHIFT combo.

Files are unsorted for 1541, 1570, 1571, 1581 while any other drive will have it's directory displayed alfabetically, and directories will be sorted to the top. Do not put more than 510 files in a directory or the rest will not be displayed.

In an input window (eg. when renaming a file) you can use the CRSR keys, HOME, CLR, INS and DEL keys. Aborting is done with the STOP key, while finishing the input is done with RETURN.

For question windows with multiple selections (eg. YES/NO/ALL)

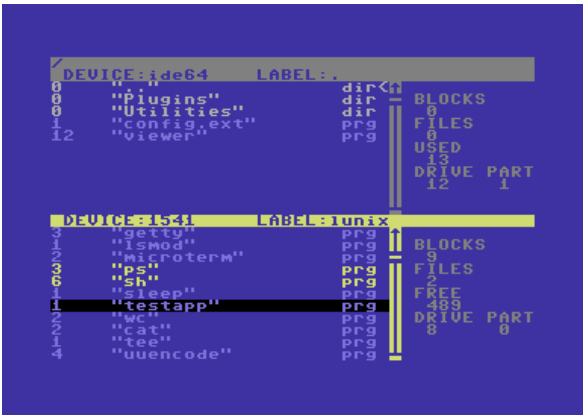


Figure 6: Manager screen

press the first letter of your choice. Simple message windows will accept any key. If you hold down the key long enough, you can see the window below the message. (eg. disk error while copying, but which file had this error?)

The recursive copy operation is not the “ultimate backup tool”, as it can’t copy relative files, links, preserve creation or modification date or file attributes.

Fast copying with a serial bus drive is only possible if it’s supported by the built in fastloader (if enabled) or it’s capable of the JiffyDOS or DolphinDOS protocol. Copying between serial drives will use the normal routines.

9.1 Plugins

External programs called by the file manager can extend it's functionality in a number of ways. Plugins can show text and various graphic formats, play sid files and animations, extract archives and disk images, fire up an assembler with the file, or just anything one can imagine!

These external programs are started by pressing RETURN or 3 on a file. For key 3 the default viewer is started. The action taken when RETURN is pressed is determined by the file called "1//:MAN,USR". The manager looks for these files on the drive specified as "Boot drive" in the CMOS Setup utility.

9.1.1 Plugin format

Plugins are machine code files compiled to $1000. When started at $1000 they should display some kind of prompt for getting the filename. The manager starts the plugins at $1003 with the accumulator loaded with the length of filename, the X register with the lower, the Y with the higher byte of the address to the filename. $BA contains the current device number the file is on. The filename always includes the file type separated by a comma on the end.

Example:

A sample plugin framework

```
*= $1000

jmp printc
jmp start
```

```
txt     .text 13,"FILENAME:",0

printc  lda #<txt
        ldy #>txt
        jsr $ab1e       ; print prompt
        ldy #255
oqu     iny
        jsr $ffcf       ; get input
        sta $200,y
        cmp #13
        bne oqu
        tya             ; setup fake manager start
        ldx #<$200
        ldy #>$200

start   jsr $ffbd       ; setup filename ("SD,MOV")
        lda #filenumber ; file number (1-255)
        ldx $ba         ; actual device number
        ldy #secaddy    ; secondary address
        jsr $ffba
        jsr $ffc0       ; open file

        ...             ; play movie

        lda #filenumber ; file number (1-255)
        jsr $ffc3 ; close file
        rts             ; done
```

The plugin must not destroy $0002–$0333 badly and must not modify $0800–$0FFF! Please leave the VIC, SID, CIA, etc. after exit of the plugin in a usable state...

9.2 Manager config file

The config file can be found in the root directory of the boot drive and it's called "MAN" with filetype "USR". The manager loads it on the first invocation, so if you want that your changes take affect immediately, then start the manager as "MAN!".

This file controls the plugin assignment based on powerful wildcard matching. Each line ends on CHR$(0). The first line contains the plugin directory, it's after a fake 2 bytes loading address. The path must contain the partition number and the full path from the root directory ending with ":". The second line contains the name of the plugin which is loaded when key 3 is pressed. Then pairs of lines follow, where the first is the plugins name, and the second the wildcard pattern. An empty line terminates the file.

For the easy creation of the config file here's an example program:

```
0 DR=PEEK(186):REM GET LAST USED DRIVE NUMBER
10 OPEN1,DR,1,"@1//:MAN,USR":PRINT#1,"." :N$=CHR$(0)
20 PRINT#1,"1//PLUGINS/:" N$; :REM PLUGIN DIRECTORY
30 PRINT#1,"VIEWER,PRG" N$; :REM VIEWER FOR KEY 3
40 PRINT#1,"SID,PRG" N$ "*,SID" N$; :REM SID
50 PRINT#1,"KLA,PRG" N$ "APIC ? *,PRG" N$; :REM KOALA
60 PRINT#1,"DRL,PRG" N$ "*.DRL" N$; :REM DRAZLACE
... more plugins
1000 PRINT#1,N$; :CLOSE1:REM CLOSE FILE
```

The program first opens the config file, and removes the old one if there was any. Also it writes the fake loading address. Line 20 defines "/PLUGINS/" as the plugin directory on partition 1. Line 30 associates key 3 with the plugin "1//PLUGINS/:VIEWER". Line 40 is an example of a filetype matching. All files with filetype "SID" will be played by the "1//PLUGINS/:SID" plugin. Line 50 associates all "PRG" type files beginning with "APIC" and a letter between spaces to the Koala painter viewer plugin. Line 60 associates all files ending on ".DRL" to the Drazlace viewer. In line 1000 the end of file marker is written, and then the file is closed.

The pattern matching is case sensitive, so it's not a bad idea to add both lowercase and uppercase variants to avoid problems with badly written CDs.

Save the config creator program, in case you want to modify or add some more plugins to you manager config file sometime later.

Key	Function
←	Exit
CONTROL	Change panel
UP , DOWN	Move cursor
F1 or LEFT	Page Up
F7 or RIGHT	Page Down
1	Refresh dir
3	Execute viewer plugin
5	Copy file(s)
6	Rename file or directory
7	Create directory
8	Delete file(s)
9	Send disk command
.	Go into parent dir
/	Go into root dir
DEL	(Un)select file
F2 or HOME	Go on top of listing
F8 or CLR	Go to end of listing
RETURN	Enter directory or LOAD"FILE",X and RUN or start viewer plugin
SHIFT + RETURN	LOAD"FILE",X,1
↑	Eject media
*	Invert file selection (SHIFT to include dirs)
+	Select all files (SHIFT to include dirs)
−	Deselect all files (SHIFT to include dirs)
C= + 0...9	Select drive
C= + SHIFT	Select charset
A...Z	Quick jump to filename

Table 11: Manager keys

10 Using the monitor

The monitor is designed to be fast and simple, but still useful for debugging and fixing. It's possible to edit every single byte in memory and I/O space without conflicting with internal variables of the monitor itself. Illegal opcodes, 65816 opcodes (emulation mode only) and wildcard searching is included.

The monitor is started by hitting C= + RESTORE at the same time, or by executing a BRK instruction, or pressing LEFTSHIFT + RESET, or alternatively by using the command SYS 0. If C128 keyboard is enabled it's possible to use HELP to start the monitor.

To recover from died monitor (not very likely) press LEFTSHIFT + RESET. Memory won't get distorted.

To rip from games, etc. press LEFTSHIFT + RESET while the program is running. (not a real freezer, but at least it's possible to rip IFLI pictures...)

Ripping on C64

A few bytes from stack ($1F3–$1FF), the contents of both CIAs, the memory configuration at $00, $01 and the processor registers (A, X, Y, SR, SP) will be lost, otherwise the memory is not touched. (ADDR will not have a valid restart address, instead it contains $FCE2)

Ripping on SuperCPU

A few bytes from stack ($1F3–$1FF), the contents of VIC registers $D020–$D02E, and zero page addresses $99, $9A will be lost, otherwise the memory is not touched. Due to end of stack corruption you may have problems with the restarted program. (ADDR will have a valid restart address!) Use SuperCPU reset button, stopping the program for a short time before freeze may be necessary.

Figure 7: Monitor screen

To use the monitor like a real freezer monitor you must make sure that the NMI vector (`$318`) won't get overwritten in RAM. If it's ok, press C= + RESTORE to freeze the program. The chip states, processor registers and memory is preserved.

There's support for two freeze points and a zeropoint. Unlike other monitor programs the freeze and zero point instructions are restored even if they were relocated since, but only the triggered one. In the C64 version it's possible to call the monitor with freeze points even from configurations like 5. (I/O area must be available)

Everything between "[" and "]" is an optional parameter, while "<" and ">" means a required paramenter.

10.1 Disk commands

SELECT DRIVE

Command: O
Purpose: Selects working drive for disk operations.
Syntax: O <DRIVE>

Drive numbers must be entered as hexadecimal. This drive number is used in all disk access commands. The default is the same as the boot drive selected in setup.

Example:

Select drive 12.

```
O C
```

DOS COMMAND

Command: @
Purpose: Read error channel and send commands.
Syntax: @<COMMAND>

Examples:

Print error channel:

```
@
00, OK,000,000,000,000
```

Send disk command:

```
@CD:SOURCE
00, OK,000,000,000,000
```

DIRECTORY

Command: @$
Purpose: Display directory of current working drive.
Syntax: @$[<PATTERN>]

Example:

Display directory:

```
@$T
2 "TEST           " IDE64
13105"TRANCEANDACID"
5    "TOD"              ASM
13110 BLOCKS USED.
```

SAVE

Command: S and SB
Purpose: Save program to disk and save a memory area to disk.
Syntax: S"NAME" <START ADDRESS> <END> [<START2>]
Syntax: SB"NAME" <START ADDRESS> <END>

Examples:

Save main file to disk, save sprites as 512 byte raw file without start address, and save 1024 bytes of data from $3000 with start address $2200.

```
S"GAME" 0801 21FF
OK

SB"SPRITES" 2000 21FF
```

```
OK

S"TO2000" 3000 33FF 2200
OK
```

LOAD

Command: L and LB
Purpose: Load a program from disk and load a memory area from disk.
Syntax: L"NAME" [<START ADDRESS>]
Syntax: LB"NAME" <START ADDRESS>

Examples:

Load the main program from disk, and link in sprite data, which is a 512 byte raw file without start address.

```
LB"SPRITES" 2200
2200-23FF OK
```

VERIFY

Command: V and VB
Purpose: Verify program from disk and verify a memory area from disk.
Syntax: V"NAME" [<START ADDRESS>]
Syntax: VB"NAME" <START ADDRESS>

Examples:

Verify the main program from disk, and the sprite data, which is a 512 byte raw file without start address.

```
V"SPRITES"
0801-21FF OK

VB"SPRITES" 2200
 2312 2313
2200-23FF OK
```

FREEZE

Command: S
Purpose: Save current machine state to disk.
Syntax: S"NAME"

Freeze (saves 64 KiB RAM + I/O) memory to disk. The actual bank must be 0–7.

Example:

Freeze memory from disk.

```
S"MYFREEZE"
OK
```

DEFREEZE

Command: K
Purpose: Restore machine state from disk.
Syntax: K"NAME"

Defreeze memory from disk (loads 64 KiB RAM + I/O). The actual bank must be 0–7.

Example:

Defreeze memory from disk.

```
K"MYFREEZE"
0000-FFFF OK
```

10.2 Display and modify memory

REGISTERS

Command: R
Purpose: Shows the processor registers.
Syntax: R

To modify the registers and flags, change the line beginning with ";", and press RETURN. ADDR is the address where the program continues (see X!), AC XR YR SP are the accumulator, x register, y register and stack pointer, BK is the currently edited bank selector (see B!), DR is the drive number used in disk access commands (see O!), NV-BDIZC are the processor flags.

Example:

Display registers.

```
R
 ADDR AC XR YR SP BK DR NV-BDIZC
;E5D1 00 00 0A F3 07 0C 00100010
```

IO CHIP REGISTERS

Command: IO
Purpose: Shows the I/O chip registers.
Syntax: IO

This command is a shortcut to quickly display VIC and CIA registers.

Example:

Display IO chip registers.

```
IO
-D000 00 00 00 00 00 00 00 00
-D008 00 00 00 00 00 00 00 00
-D010 00 1B 37 00 00 00 C8 00
-D018 15 71 F0 00 00 00 00 00
-D020 FE F6 F1 F2 F3 F4 F0 F1
-D028 F2 F3 F4 F5 F6 F7 FC FF

-DC00 7F 00 FF 00 25 40 FF FF
-DC08 00 00 00 01 00 81 01 08

-DD00 C7 00 3F 00 FF FF FF FF
-DD08 00 00 00 01 00 00 08 08
```

ASSEMBLE

Command: A
Purpose: Enter assembly code.
Syntax: A <ADDRESS> <OPCODE MNEMONIC> <OPERAND>

Illegal instructions are supported and emulation mode 65816 instructions too.

Examples:

Enter a few assembly instructions.

```
A1000 EE 20 D0 INC $D020
A1003 4C 00 10 JMP $1000
A1006 B3 30    LAX ($30),Y
A1008 NOP
```

These are legal instruction entering forms:

```
A1000 INC D020
A1003 LDA#$3
A1006 NOP:VSFV
```

DISASSEMBLE

Command: D
Purpose: Disassemble machine code into assembly.
Syntax: D [<ADDRESS 1>] [<ADDRESS 2>]

Pressing RETURN on modified hex bytes or on modified disassembly changes memory. (the cursor position selects what happens) To slow down listing hold CONTROL, to stop press STOP, to pause listing press anything else. If the whole screen is filled and you want to get an empty line then go into the last line and press SHIFT + RETURN. (or clear the screen)

Examples:

Disassembly from last address: D.

Disassembly a few lines: D 1234.
Disassembly continuously forward from: D1234-.
Disassembly continuously backwards from: D-1234.
Disassembly between addresses: D1234-5678.

Simply display memory at $1000.

```
D 1000
,1000 EE 20 D0 INC $D020
,1003 4C 00 10 JMP $1000
,1006 B3 30    LAX ($30),Y
```

Set freeze point at $1000. This will insert a JSR call to the monitor, and also remembers the old code. This code will be restored when control returns to monitor by the JSR freeze point, which may have been relocated since the original position. IO area must be available to be able to return, and only two different freeze points are available. Please note that freeze points won't get restored, unless executed.

```
D 1000
,1000 SF 20 D0 INC $D020
,1003 4C 00 10 JMP $1000
,1006 B3 30    LAX ($30),Y
```

Set zero point (BRK) at $1000. This will insert a BRK to call the monitor, and also remembers the old code. This code will be restored when control returns to monitor by a zero point. IO area must be available to return, and the BRK vector must be correct. Please note that a zero point won't get restored, unless executed.

```
D 1000
,1000 SZ 20 D0 INC $D020
,1003 4C 00 10 JMP $1000
,1006 B3 30    LAX ($30),Y
```

MEMORY IN HEX

Command: M
Purpose: Dump memory in hex and PETSCII.
Syntax: M [<ADDRESS 1>] [<ADDRESS2>]

Press RETURN to enter the modified hex values or PETSCII text into memory. Possible parameters are the same as for D.

Example:

Display memory from $E478.

```
M E478
:E478 20 2A 2A 2A 2A 20 43 4F  **** CO
:E480 4D 4D 4F 44 4F 52 45 20 MMODORE
:E488 36 34 20 42 41 53 49 43 64 BASIC
:E490 20 56 32 20 2A 2A 2A 2A  V2 ****
```

DISPLAY AS PETSCII

Command: I
Purpose: Display memory as PETSCII.
Syntax: I [<ADDRESS 1>] [<ADDRESS 2>]

Press RETURN to enter the modified text into memory. Some PETSCII values have the same screen code, so be careful when edit-

ing text mixed with code. Possible parameters are the same as for D.

Example:

Display memory from `$a0a0`.

```
i a0a0
'a0a0 DfoRnexTdatAinput inpuTdiMreaDle
'a0c0 TgotOruNiFrestorEgosuBreturNreMs
'a0e0 toPoNwaiTloaDsavEverifYdeFpokEpr
'a100 inT prinTconTlisTclRcmDsySopeNcl
```

DISPLAY AS SCREEN

Command: J
Purpose: Display memory as screen code.
Syntax: J [<ADDRESS 1>] [<ADDRESS 2>]

Press RETURN to enter the modified text into memory. (I∗ is available for MK7 addicts) Possible parameters are the same as for D.

Example:

Display memory from `$400`.

```
J 400
.0400     **** COMMODORE 64 BASIC V2 *
.0428 ***
.0450                  64K RAM SYSTEM
.0478  IDE-DOS BETA! EXPANDED
```

DISPLAY AS BINARY

Command: EC
Purpose: Display memory as binary.
Syntax: EC [<ADDRESS 1>] [<ADDRESS 2>]

Press RETURN to enter the modified bytes into memory. "." means 0, anything else (no space) 1. Possible parameters are the same as for D.

Example:

Display memory from the character rom at address $D008.

```
B 3
EC D008
[D008 ...##...
[D009 ..####..
[D00A .##..##.
[D00B .######.
[D00C .##..##.
[D00D .##..##.
[D00E .##..##.
[D00F ........
```

DISPLAY AS SPRITE

Command: ES
Purpose: Display memory as sprite.
Syntax: ES [<ADDRESS 1>] [<ADDRESS 2>]

Press RETURN to enter the modified bytes into memory. "." means 0, anything else (no space) 1. Possible parameters are the same as for D.

Example:

Display a sprite from $A00.

```
ES A00
]0A00 ........................
]0A03 ........................
]0A06 ........................
]0A09 ........................
]0A0C ........................
]0A0F ........................
]0A12 ........................
]0A15 ........................
]0A18 ...........#######......
]0A1B .........##########.....
]0A1E .......###########.....
]0A21 ......#######.....#.....
]0A24 .....########.....#.....
]0A27 ....########......#.....
]0A2A ....########..#...#.....
]0A2D ....#######...#..##.....
]0A30 .....##.......#..##.....
]0A33 ......#...........#.....
]0A36 ......#...........##....
]0A39 .......#....#......#....
]0A3C ........##...#....#.....
```

BACKTRACE

Command: BT
Purpose: Display call trace.
Syntax: BT

Example:

Do a backtrace. $1237, $1233 and $1230 are the caller JSR addresses, $30 is some pushed data.

```
A1230 20 33 12 JSR $1233
A1233 20 36 12 JSR $1236
A1236 08       PHP
A1237 20 3A 12 JSR $123A
A123A 00       BRK
A123B

G 1230

BRK EXCEPTION

 ADDR AC XR YR SP BK DR NV-BDIZC
;1230 10 00 0A C6 07 0D 00110000
BT
1237 30 1233 1230 7A E3 1009 050E 10
```

10.3 Execution control

GO

Command: G
Purpose: Execute a routine, and return to monitor.
Syntax: G <ADDRESS>

When program terminates it trys to return to monitor with a BRK.

Example:

Start program at $080D.

```
G 80D
```

EXIT

Command: X
Purpose: Exit monitor and continue execution.
Syntax: X

As the IDE64 monitor is always in freeze mode, exit will continue the interrupted program. If you want to get back to the BASIC prompt, use Q.

Example:

Continue program execution at ADDR.

```
IDE64 MONITOR

 ADDR AC XR YR SP BK DR NV-BDIZC
;E5D1 00 00 01 F2 07 08 00100010
X
```

QUIT

Command: Q
Purpose: Return to BASIC prompt from monitor.
Syntax: Q

Example:

Try to exit to basic prompt. Useful after hitting a BRK.

```
BRK EXCEPTION

 ADDR AC XR YR SP BK DR NV-BDIZC
;1230 10 00 0A C6 07 0D 00110000
Q
```

10.4 Memory area commands

TRANSFER

Command: T
Purpose: Copy a memory area to another address.
Syntax: T <START ADDRESS> <END> <DESTINATION>

Overlapping areas are supported.

Example:

Copy memory from $400–$7FF to $C00–$FFF.

```
T 400 7FF C00
```

COMPARE

Command: C
Purpose: Compare a memory area with another.
Syntax:C <START ADDRESS> <END> <START2>

Example:

Compare memory from $2000–$3FFF with $E000–$FFFF.

```
C 2000 3FFF E000
 3FC0 3FC1 3FC2
```

FILL

Command: F
Purpose: Fill a memory area with the specified byte.
Syntax: F <START ADDRESS> <END> <FILL BYTE>

Example:

Fill memory $2–$FFFF with $00.

```
B 4
F 2 FFFF 0
```

HUNT

Command: H
Purpose: Search memory for a specific pattern.
Syntax: H <START ADDRESS> <END> <PATTERN>

"?" is a 1 character wildcard matching everything.

Example:

Search memory for instructions accessing $277.

```
H E000 FFFF ? 77 2
 E5B4 E5BC EB3C
D E5B4
,E5B4 AC 77 02 LDY $0277
,E5B7 A2 00    LDX #$00
,E5B9 BD 78 02 LDA $0278,X
```

Search memory for PETSCII text.

```
H E000 FFFF "COMM"
 E47E
```

10.5 Miscellaneous

BANK

Command: B
Purpose: Select memory configuration and area.
Syntax: B <BANK>

Banks 0...7 have the same meaning like the first 3 bits of $01, banks 8...1E selects the drive's memory with that device number.

Example:

Select bank 4.

```
B 4
```

RAM/ROM

Command: *
Purpose: Change between banks 4 and 7.
Syntax: *

Example:

Quick switch between RAM and ROM (bank 4 and 7)

```
 ADDR AC XR YR SP BK DR NV-BDIZC
;EAB4 03 03 15 EA 07 08 00100101
*
RAM
R
 ADDR AC XR YR SP BK DR NV-BDIZC
;EAB4 03 03 15 EA 04 08 00100101
```

NUMBER CONVERT

Command: N
Purpose: Calculate an expression and display the result.
Syntax: N <EXPRESSION>

The result is displayed in decimal, octal, binary, screencode and PETSCII.

Example:

Calculate something.

```
N $0E00+23*SIN(1)
3603 0E13 0000111000010011 SN SN
```

VIEW

Command: W
Purpose: Look at screen.
Syntax: W

Example:

Useful for finding the video bank after LEFTSHIFT + RESET.

```
W
```

ADDRESS STACK

Command: ←
Purpose: Push addresses to the "address stack".
Syntax: ←

Search for something, then put a "←" before the address list, press RETURN (push), then list with "D↑", or with something else (pop). The address stack is 8 address deep only.

Example:

An example using the astack.

```
H E000 FFFF 20 BA FF
←E1DD E1F0 E22B E23F E24E
D↑
,E24E 20 BA FF JSR $FFBA
,E251 20 06 E2 JSR $E206
,E254 20 0E E2 JSR $E20E
D↑
```

```
,E23F 20 BA FF JSR $FFBA
,E242 20 06 E2 JSR $E206
,E245 20 00 E2 JSR $E200
D↑
,E22B 20 BA FF JSR $FFBA
,E22E 20 06 E2 JSR $E206
,E231 20 00 E2 JSR $E200
D↑
,E1F0 20 BA FF JSR $FFBA
,E1F3 20 06 E2 JSR $E206
,E1F6 20 00 E2 JSR $E200
D↑
,E1DD 20 BA FF JSR $FFBA
,E1E0 20 06 E2 JSR $E206
,E1E3 20 57 E2 JSR $E257
```

SCROLLING

List code upwards and downwards by using the keys F1, F3, F5 and F7! Real crackers use F2 and F8 for fast code search ;)

Example:

Type D FCE2 then hold F3 till the code before $FCE2 appears. (other keys work similar)

```
,FCDD D0 02    BNE $FCE1
,FCDF E6 AD    INC $AD
,FCE1 60       RTS
```

```
,FCE2 A2 FF    LDX #$FF
,FCE4 78       SEI
,FCE5 9A       TXS
,FCE6 D8       CLD
,FCE7 20 02 FD JSR $FD02
```

Command	Function
@COMMAND	Disk command and status
@$	Directory
A XXXX NOP	Assemble
B XX	Memory configuration
BT	Backtrace
C <START> <END> <START2>	Compare memory
D [<START>] [<END>]	Disassemble
,START XX XX XX	Write hex data to memory and disassemble
EC [<START>] [<END>]	Edit char (binary)
[START XXXXXXXX	Write binary data to memory
ES [<START>] [<END>]	Edit sprite (binary)
]START XXXXXXXX	Write binary sprite data to memory
F START END XX	Fill memory with byte
G XXXX	Execute at address
H START END XX ? "TXT"	Search hex / any / PETSCII
I [<START>] [<END>]	Dump memory in PETSCII
'START XXXXXXX	Write PETSCII data to memory
J [<START>] [<END>]	Dump memory in screen code
.START XXXXXXX	Write screen code data to memory
K "NAME"	Defreeze memory
L "NAME" [<START>]	Load program
LB "NAME" <START>	Load binary
M [<START>] [<END>]	Dump memory in hex and PETSCII
N <EXPRESSION>	Number conversion and calculator
O XX	Actual drive
:START XX XX XX XX	Write hex or PETSCII data to memory
-START XX XX XX XX	Write hex data to I/O memory
R	Show registers
;XXXX XX XX	Change registers
S "NAME"	Freeze memory
S "NAME" <START> <END> [<START2>]	Save program
SB "NAME" <START> <END>	Save binary
T <START> <END> <START2>	Copy memory
V "NAME" [<START>]	Verify program
VB "NAME" <START>	Verify binary
X	Continue program
Q	Exit to BASIC warm start
←	(left arrow) Push address(es) to astack.
↑	(up arrow) Pop address from astack.
*	Select RAM / ROM

Table 12: Monitor commands

11 DOS Wedge

These are one or two character long commands to speed up some frequently typed command at the basic prompt. Enable or disable this feature in the setup utility.

11.1 @ (at sign)

It sends the parameter (if any) to the last used device and then prints the command channel.

Examples:

Print error channel:

```
@
00, OK,000,000,000,000

READY.
```

Delete a file:

```
@S:MYFILE
01, FILES SCRATCHED,001,000,000,000

READY.
```

11.2 @# (at, number sign)

Selects the specified device as last used device.

Example:

Select drive 8:

```
@#8

READY.
```

11.3 @$ (at, dollar sign)

Display directory without affecting memory

Example:

List “PRG” files beginning with “S” ending with “.DRL”.

```
@$S*.DRL=P

1 "TEST           " IDE64
33   "SHUDDER.DRL"      PRG
33 BLOCKS USED.
READY.
```

11.4 / (slash)

Loads a basic file, like “LOAD"FILE" , < CURRENT DEVICE > ”.

Example:

To load a program from directory listing:

```
@$

1 "TEST           " IDE64
0    "PLUGINS"          DIR
132  "2 PLANETS BY"     FUN
```

```
45   "DRAZPAINT 2.0"    PRG
/1   "DRAZLACE V1.0"    PRG
64   "TASM"             PRG
SEARCHING FOR DRAZLACE V1.0
LOADING $0801-$3023
READY.
```

11.5 % (percent sign)

Loads an assembly file, like "LOAD"FILE" , < CURRENT DEVICE > , 1", but does not restart program if used in BASIC program.

Example:

To load a program from directory listing:

```
LIST

1 "TEST            " IDE64
0    "PLUGINS"          DIR
132  "2 PLANETS BY"     FUN
45   "DRAZPAINT 2.0"    PRG
41   "DRAZLACE V1.0"    PRG
%4   "TASM",8           PRG
33   "SHUDDER.DRL"      PRG
SEARCHING FOR TASM
LOADING $9000-$CF00
READY.
```

11.6 ´ (apostrophe)

Verifies an assembly file, like “VERIFY "FILE" , < CURRENT DEVICE > , 1”.

Example:

Verify Drazlace:

```
'DRAZLACE*

SEARCHING FOR DRAZLACE*
VERIFYING $0801-$3023
OK

READY.
```

11.7 ↑ (up arrow)

Loads a basic file, like “/”, and then starts it with “RUN”.

Example:

Load Drazlace and start it:

```
↑DRAZLACE*

SEARCHING FOR DRAZLACE*
LOADING $0801-$3023
READY.
Ru:
```

11.8 ← (left arrow)

Saves a basic file, like "SAVE"FILE" , < CURRENT DEVICE >".

Example:

To save a program:

```
←MYFILE

SAVING MYFILE $0801-$0932
READY.
```

11.9 £ (pound sign)

Loads an assembly file, like "%", and then starts it with JMP at it's load address.

Example:

Load and start TASM:

```
£TASM

SEARCHING FOR TASM
LOADING $9000-$CF00
```

11.10 . (period)

Changes directory, like "CD", but much better when used on directory list.

Example:

Enter directory "PLUGINS":

```
@$

1 "TEST            " IDE64
.    "PLUGINS"          DIR
132  "2 PLANETS BY"     FUN
READY.DRAZPAINT 2.0"    PRG
41   "DRAZLACE V1.0"    PRG
64   "TASM"             PRG
33   "SHUDDER.DRL"      PRG
315 BLOCKS USED.
READY.
```

11.11 # (hashmark)

Loads the machine language program called "1//:SH" from the boot device and executes it. The last used device number will be placed at $FF. The pointer $7A points to the rest of the non-tokenized command line.

Example:

This small assembly program defines "#T" to quickstart Turbo Assembler from "1//UTIL/:TASM" from the boot drive.

```
*=$334

jsr $79
cmp #"t"        ;#T?
bne not
ldx $ba         ;boot drive
ldy #1          ;,1
```

```
        jsr $ffba       ;setlfs
        lda #nameend-name
        ldx #<name
        ldy #>name
        jsr $ffbd       ;setnam
        lda #0
        jsr $ffd5       ;load it!
        bcs not         ;error?
        lda $ff
        sta $ba         ;restore last used device
        jmp $9000       ;start tasm

not     lda $ff
        sta $ba         ;restore last used device
        jmp $e37b       ;exit with ready.

name    .text "1//util/:tasm"
nameend
```

12 BASIC extensions

Most extension port cards have BASIC extensions, so IDE64 also has a few. The ?SYNTAX ERROR bug of LIST command was fixed, listing of BASIC programs protected by "REM L" is not a problem anymore.

Binary, octal, decimal and hexadecimal numbers are supported in expressions:

```
PRINT %11; &11; 11; $11
 3  9  11  17

READY.
```

On the next few pages the new and changed BASIC commands are described.

12.1 CD – change directory

Sends change directory to the last used or specified device. (Same as "@CD:PARAMETER") There's no short form of this command.

Format:

CD"*{{<part #>}{<path>}:}<directory name>"{,<device #>}*

Example:

```
CD"GAMES"

READY.
```

12.2 CDCLOSE – insert media

Sends close tray command to the last used or specified device. (Same as "@U0>E0") Not very useful, as the tray will be automatically closed on first media access anyway. The short form is "CDCLO".

Format:

CDCLOSE *{<device #>}*

12.3 CDOPEN – eject media

Sends eject media command to the last used or specified device. Beside a CDROM drive in works with LS-120 or ZiP drive too. (Same as "@U0>E1") The short form is "CDOP".

Format:

CDOPEN *{<device #>}*

12.4 CHANGE – change device number

Sends change device number to the last used or specified device. (Same as "@S-8" or "@S-D") The short form is "CHA".

Format:

CHANGE *{<device #>}*

Example:

Change device 12 to be device 8:

```
CHANGE12

READY.
```

Change device 8 back:

```
CHANGE8

READY.
```

12.5 DATE – display date

Prints the date and time of the last used or specified device. For date format see Reading time from RTC! (Same as "@T-RA") There's no short form of this command.

Format:

DATE *{<device #>}*

Example:

```
DATE
THUR 08/12/04 08:51:18 PM

READY.
```

12.6 DEF – redefine F-keys

If you do not like the default F1–F8 funcion key assignment, then this command can change it. The best practice to make this permanent to put it into the boot file, and select "Power on" or "Always" in the CMOS Setup for "Start boot file". Of course the "Function keys" setting must be enabled to make use of the function keys. All 8 string parameters are mandatory, if the string contains a CHR$(13) character then a RETURN keypress is simulated.

Format:

DEF"*<f1>*","*<f3>*","*<f5>*","*<f7>*","*<f2>*","*<f4>*","*<f6>*","*<f8>*"

Example:

This boot file redefines the F-keys, and cleans up the screen.

```
10 DEF"F1","F3","F5","F7","F2","F4","F6","F8"
20 PRINTCHR$(145)+"   "+CHR$(145)+CHR$(145);
30 NEW
```

12.7 DIR – list directory

Lists the directory of the last used or specified device. (Same as "@$:PATTERN") There's no short form of this command.

Format:

DIR *{"{{<part #>}{<path>}:}<pattern>"{,<device #>}}*

Example:

```
DIR"*=B

3 "ACE             " IDE64
0    "BIN"              DIR
0    "ETC"              DIR
0    "HOME"             DIR
0 BLOCKS USED.
READY.
```

12.8 HDINIT – redetect devices

Trys to auto detect devices connected to the cartridge. The short form is "HD".

Format:

HDINIT

Example:

```
HDINIT
MASTER:
ST9385AG

READY.
```

12.9 INIT – init memory

Fills memory with nulls or with the specified byte, and then it performs a reset. Useful before linking. The short form is "INI".

Format:

INIT *{<fill byte #>}*

Example:

Fill memory with `$55`

```
INIT$55
```

12.10 KILL – disable cartridge

Switches cartridge off. Useful if you suspect compatibility problems with a program. Will also shut down power managed devices, if typed as "KILL!". The short form is "KI".

Format:

KILL*{!}*

Example:

```
KILL
BYE!

READY.
```

12.11 LL – long directory list

Pretty verbose directory list for power users. First line is the directory label, same parameters as for command DIR. No short form.

Format:

LL *{"{{<part #>}{<path>}:}"{,<device #>}}*

Examples:

Simple listing:

```
LL

D--RWX-            1980-01-01 00:00:00  "WORK            " DIR 560/5/36
DC-R-XH          0 1980-01-01 00:00:00  "%DELETED  FILES%" DIR 560/5/35
-CDRWX-        363 2004-08-12 16:12:13  "LIST"             TXT 568/12/4
-CDRWX-     820528 2004-08-11 16:32:26  "UTILITIES"        ZIP 560/5/37
-CDRWX-    2234108 2004-08-11 18:46:01  "TXT"              ZIP 562/9/21
-CDRWX-      25513 2004-08-11 19:29:43  "CHANNEL15"        ASM 568/11/9
READY.
```

Part	Meaning
	`.` — DEL entry
	`-` — file
`-`	`R` — relative file
	`D` — directory
	`L` — link
`C`	`C` — Closed
`D`	`D` — Deletable
`R`	`R` — Readable
`W`	`W` — Writable
`X`	`X` — Loadable
`-`	`H` — Hidden
`363`	Size of file
`2004-08-12`	Date
`16:12:13`	Time
`"LIST"`	Filename
`TXT`	File type
`568/12/4`	Disk address (CHS or LBA)

Table 13: LL format

Redirect listing into file for later review.

```
OPEN 1,12,1,"LIST,S":CMD 1:LL:PRINT#1:CLOSE 1

READY.
```

Listing works on 1541, 1570, 1571, 1581, and CMD devices too:

```
LL"",8

"GEOS 128 V2.0AMă" CPă2A
P--              2 1988-08-22 13:00     "GEOS"             PRG 19/15
P--              6 1988-08-22 13:00     "GEOS BOOT"        PRG 19/17
U--            156 1988-08-22 13:00     "GEOS KERNAL"      USR 19/18
U-W            137 1988-10-10 13:06     "128 DESKTOP"      USR 8/20
U-W             79 1988-09-08 16:50     "128 CONFIGURE"    USR 15/6
BREAK
READY.
```

12.12 LOAD – load a program

Loads a program file into memory. Using no device number selects last used device. No filename means "∗", except for tape. The short form is "LO".

Format:

LOAD*{"{{<part #>}{<path>}:}<filename>"{,<device #>{,<mode #>}}}*

Example:

```
LOAD

SEARCHING FOR *
LOADING $0801-$099E
READY.
```

12.13 MAN – start manager

Starts the built in file manager. The short form is "MA".

Format:

MAN*{!}*

12.14 MKDIR – create directory

Sends make directory to the last used or specified device. (Same as "@MD:NEWDIR") The short form is "MK".

Format:

MKDIR"*{{<part #>}{<path>}:}<directory name>"{,<device #>}*

Example:

```
MKDIR"PICS",12

READY.
```

12.15 RM – remove file

Sends scratch file to the last used or specified device. (Same as "@S:FILENAME") There's no short form of this command.

Format:

RM"*{{<part #>}{<path>}:}<file name>"{,<device #>}*

Example:

```
RM"OLDSTUFF",8

READY.
```

12.16 RMDIR – remove directory

Sends remove directory to the last used or specified device. (Same as "@RD:DIRNAME") There's no short form of this command.

Format:

RMDIR"*{{<part #>}{<path>}:}<directory name>"{,<device #>}*

Example:

```
RMDIR"OLDDIR"

READY.
```

12.17 SAVE – save a program

Saves program to disk. Using no device number selects last used device. You'll get a "?FILE DATA ERROR" if an error happens during save. The short form is "SA".

Format:

SAVE"*{{@}{<part #>}{<path>}:}<filename>"{,<device #>{,<mode #>}}*

Example:

```
SAVE"TEST"

SAVING TEST $0801-$1A43
READY.
```

12.18 SYS – start ML program

This keyword can be used to call machine code subroutines for BASIC. Optionally the accumulator, the x and y registers and the status can be specified sepearated by semicolons.

Format:

SYS *<address #>{;<a #>{;<x #>{;<y #>{;<sr #>}}}}*

Examples:

Print a star to the center of screen:

```
SYS $fff0;;12;20;0:PRINT"*"
```

Start the built in monitor:

```
SYS.
```

12.19 VERIFY – verify program

Verifies that the program in memory matches the on-disk version. Using no device number selects last used device. No filename means "∗". The short form is "VE".

Format:

VERIFY*{"{{<part #>}{<path>}:}<filename>"{,<device #>{,<mode #>}}}*

Example:

```
VERIFY"TEST"

SEARCHING FOR TEST
```

```
VERIFYING $0801-$1A43
OK

READY.
```

13 Programming in assembly

This part is not finished...

13.1 Standard kernal routines

These routines work for all devices, you should use them in your programs for compatibility with IDE64, RamLink and other non-serial bus devices.

13.1.1 $FFB7 – READST

Implementation:	Standard KERNAL
Communication registers:	None
Preparatory routines:	None
Error returns:	None
Status:	None
Registers affected:	A

Returns the status in accumulator. It's used to detect errors, end of file, etc.

Example:

Check end of file while reading

```
jsr chrin       ; read data from file
sta data,y
jsr readst      ; test status
and #$40        ; end of file flag
bne endoffile
```

Bit	Meaning
7	Device not present
6	End of File
5	(Tape CRC error)
4	Verify / read error (Tape read error)
3	(Tape long block)
2	(Tape short block)
1	Timeout on receive
0	Timeout on send

Table 14: Device status (`$90`)

13.1.2 $FF90 – SETMSG

Implementation: Standard KERNAL
Communication registers: A
Preparatory routines: None
Error returns: None
Status: None
Registers affected: A

Controls IDEDOS and KERNAL message printing. Messages are things like "`SEARCHING FOR XXX`" and "`I/O ERROR#05`". Sometimes it's useful to suppress such messages to not destroy the screen.

Example:

Turn off messages to prevent screen distortion

```
lda #$00
jsr setmsg      ; turn off messages
```

Bit	Meaning
7	Full error messages (LOADING, etc.)
6	KERNAL error messages (I/O ERROR#x)
5...0	Undefined

Table 15: Messages (`$9D`)

13.1.3 $FFE1 – STOP

Implementation: Standard KERNAL
Communication registers: A
Preparatory routines: None
Error returns: None
Status: None
Registers affected: A, X

Checks if STOP was pressed.

Example:

Check STOP key while reading a file (STOP calls CLRCHN if STOP was pressed)

```
        jsr chrin
        sta ($fb),y
        jsr stop
        beq stopped     ; stop was pressed

        ...

stopped lda #filenum
```

```
jmp close       ; close file and exit
```

13.1.4 $FFBA – SETLFS

Implementation:	Standard KERNAL
Communication registers:	A, X, Y
Preparatory routines:	None
Error returns:	None
Status:	None
Registers affected:	None

Sets logical file number, device number and secondary address. These parameters are the same as for the OPEN BASIC command. It's used before OPEN, LOAD and SAVE. File number is ignored for LOAD and SAVE, and secondary address too for SAVE.

Example:

Set the first three parameters of OPEN.

```
lda #filenumber; file number (1-255)
ldx $ba        ; actual device number
ldy #secaddy   ; secondary address
jsr setlfs

...

jsr open
```

File number	Meaning
0	Illegal
1...127	Nothing special
128...255	BASIC adds `CHR$(10)` after each line

Table 16: File numbers (`$B8`)

13.1.5 $FFBD – SETNAM

Implementation:	Standard KERNAL
Communication registers:	A, X, Y
Preparatory routines:	None
Error returns:	None
Status:	None
Registers affected:	None

Sets filename and name length for OPEN, LOAD, and SAVE routines. The filename must be located below `$D000` in memory. Don't forget to set `$01` if it's under the BASIC ROM.

Example:

Set filename for OPEN

```
lda #8          ; filename length
ldx #<name      ; pointer to filename, low byte
ldy #>name      ; high byte
jsr setnam

...
```

Device number	Device
0	Keyboard
1	Datassette
2	RS-232C device
3	CRT display
4…5	Serial bus printer
6…7	Serial bus plotter
8…11	CBM serial bus disk drive
12	Primary IDE controller, Master (default)
13	Primary IDE controller, Slave (default)
14	PC-link serial / parallel (default)
15…30	Other
31…255	Illegal

Table 17: Device numbers (`$BA`)

```
        jsr open

name    .text "filename"
```

13.1.6 $FFC0 – OPEN

Implementation:	IDEDOS Extended
Communication registers:	None
Preparatory routines:	SETLFS, SETNAM
Error returns:	1, 2, 4, 5, 6, ~~7~~, 240 (see error codes)
Status:	$00, $80
Registers affected:	A, X, Y

Secondary address	Open mode
0	Read access (load)
1	Write access (save)
2...14	Bidirectional data channel
15	Status and command channel
16...127	Illegal
128...255	Illegal (serial bus raw mode)

Table 18: Secondary addresses (`$B9`)

Opens a file and associates it with a logical file number.

Example:

Open a file

```
        lda #filenumber; file number (1-255)
        ldx $ba        ; actual device number
        ldy #secaddy   ; secondary address
        jsr setlfs
        lda #8         ; filename length
        ldx #<name     ; pointer to filename, low byte
        ldy #>name     ; high byte
        jsr setnam
        jsr open

name    .text "filename"
```

13.1.7 $FFC3 – CLOSE

Implementation:	IDEDOS Extended
Communication registers:	A
Preparatory routines:	None
Error returns:	0, 240 (see error codes)
Status:	$00, $03, $80
Registers affected:	A, X, Y

Closes the file associated by the logical file number

Example:

Close a file

```
lda #filenumber; opened file number
jsr close
```

13.1.8 $FFC6 – CHKIN

Implementation:	IDEDOS Extended
Communication registers:	X
Preparatory routines:	valid OPEN
Error returns:	0, 3, 5, ~~6~~ (see error codes)
Status:	$00, $03, $80
Registers affected:	A, X

Set standard input to the logical file number. This means now you can use CHRIN, GETIN and READ on the file.

Example:

Start to read from a file

```
...
jsr open        ; open file

ldx #filenumber ; opened file number
jsr chkin       ; set input

ldx #0
jsr chrin       ; get bytes
sta $400,x
...
```

13.1.9 $FFC9 – CHKOUT

Implementation:	IDEDOS Extended
Communication registers:	X
Preparatory routines:	valid OPEN
Error returns:	0, 3, 5, ~~7~~ (see error codes)
Status:	$00, $03, $80
Registers affected:	A, X

Set standard output to the logical file number. This means now you can use CHROUT and WRITE on the file.

Example:

Starts to write to a file

```
...
jsr open        ; open file

ldx #filenumber ; opened file number
```

```
jsr chkout      ; set output

ldx #0
lda $400,x
jsr chrout      ; write out data
...
```

13.1.10 $FFCF – CHRIN

Implementation:	IDEDOS Extended
Communication registers:	None
Preparatory routines:	valid OPEN, CHKIN
Error returns:	6 (see error codes)
Status:	$00, $40, $42, $52, $80
Registers affected:	A (Y but not for file I/O)

Get a character from standard input. If it's the screen the cursor will appear and you can type in characters until RETURN.

Example:

Get in a byte from standard input

```
...
jsr chkin

ldy #0
jsr chrin
sta data,y
iny
...
```

13.1.11 $FFE4 – GETIN

Implementation:	IDEDOS Extended
Communication registers:	None
Preparatory routines:	valid OPEN, CHKIN
Error returns:	6 (see error codes)
Status:	$00, $40, $42, $52, $80
Registers affected:	A (X, Y but not for file I/O)

Get a character from standard input. If it's the screen the last pressed keys from the keyboard buffer will be be returned. If there is none, then `$00` will be returned.

Example:

Get in a byte from standard input

```
...
jsr chkin

ldy #0
jsr getin
sta data,y
iny
...
```

Wait until a space is pressed

```
...
jsr clrchn      ; keyboard/screen
...
```

```
wait    jsr getin       ; get key
        beq wait        ; nothing pressed?

        cmp #32         ; space pressed?
```

13.1.12 $FFD2 – CHROUT

Implementation:	IDEDOS Extended
Communication registers:	A
Preparatory routines:	valid OPEN, CHKOUT
Error returns:	~~7~~ (see error codes)
Status:	$00, $03, $80
Registers affected:	None

Output a character to standard output.

Example:

Put in a byte to output

```
        ...
        jsr chkout

        lda #$00
        jsr chrout      ; write 0
        ...
```

13.1.13 $FFE7 – CLALL

Implementation:	IDEDOS Extended
Communication registers:	None
Preparatory routines:	None
Error returns:	~~5~~ (see error codes)
Status:	$00, $03, $80
Registers affected:	A, X

Forget about all files and set standard input and output to keyboard and screen. This will NOT close the files properly!

Example:

Make sure all files are closed before starting.

```
jsr clall       ; close all files, default i/o
jmp run         ; run program
```

13.1.14 $FFCC – CLRCHN

Implementation:	IDEDOS Extended
Communication registers:	None
Preparatory routines:	None
Error returns:	~~5~~ (see error codes)
Status:	$00, $03, $80
Registers affected:	A, X

Set standard input and output to keyboard and screen

Example:

Set standard keyboard and screen in and output.

```
...
jsr chkin
...

lda #1
jsr close
jsr clrchn      ; set default i/o
```

13.1.15 $FFD5 – LOAD

Implementation:	IDEDOS Extended
Communication registers:	A, X, Y
Preparatory routines:	SETLFS, SETNAM
Error returns:	0, 4, 5, 8, 9, 16 (see error codes)
Status:	$00, $10, $40, $42, $50, $52, $80
Registers affected:	A, X, Y

Loads or verifies a program file. Program files start with a 2 byte little endian memory start address.

Example:

Load a file. It's possible to load from $0400 to $FFFF (IDEDOS switches $01 memory configuration register automatically if needed).

```
lda #1          ; filename length
ldx #<dirnam
ldy #>dirnam    ; filename pointer
jsr setnam

lda #1          ; file number
```

```
        ldx $ba         ; actual device number
        ldy #0          ; sec.address 0=specified,
        jsr setlfs      ; else original location

        lda #$00        ; load flag (1=verify)
        ldx #<dirbuff
        ldy #>dirbuff   ; new start address
        jsr load
        bcc loadok
        ...
        rts

loadok  stx $ae         ; new end after load/verify
        sty $af
        ...
        rts

dirnam  .text "$"
```

13.1.16 $FFD8 – SAVE

Implementation:	IDEDOS Extended
Communication registers:	A, X, Y
Preparatory routines:	SETLFS, SETNAM
Error returns:	0, 5, 8, 9, 24 (see error codes)
Status:	$00, $03, $80
Registers affected:	A, X, Y

Save program to file. A 2 byte start address is inserted in front of

the data.

Example:

Save a file. It's possible to save RAM from `$0200` to `$CFFF`. For save RAM under the BASIC ROM don't forget to set `$01`!

```
databegin = $fb

        lda #1          ; file number
        ldx $ba         ; actual device number
        ldy #0          ; sec.address
        jsr setlfs

        lda #8          ; filename length
        ldx #<name
        ldy #>name      ; filename pointer
        jsr setnam

        lda #<$1000
        sta databegin   ; begin
        lda #>$1000
        sta databegin+1
        lda #databegin
        ldx #<$8000
        ldy #>$8000     ; end
        jsr save

name    .text "filename"
```

Accu	Meaning
0	Routine terminated by the STOP key
1	Too many open files
2	File already open
3	File not open
4	File not found
5	Device not present
6	File is not an input file
7	File is not an output file
8	File name is missing
9	Illegal device number
16	Out of memory (load)
24	File data error (save)
240	Top-of-memory change RS-232 buffer (de)allocation

Table 19: Error codes returned by IDEDOS and KERNAL

13.2 IDE64 specific routines

13.2.1 IDE64 card detection

If you want that your application using READ or WRITE calls remain still runnable on a not IDE64 equipped machine, check for IDE64 presence before calling these two calls, and use standard routines instead. (Imagine what happens at `JSR $DEF1` if there's open I/O space at `$DE00–$DEFF...`)

Example:

Detect IDE64 before write and workaround if not present.

```
        lda $de60       ; check IDE64
        cmp #$49
        bne old
        lda $de61
        cmp #$44
        bne old
        lda $de62
        cmp #$45
        bne old

        lda #zp
        jsr $def1
        bcs old2        ; write not available?
        rts

old2    ldx #channel
        jsr chkout ; or chkin
old     ...             ; old byte by byte routine
        rts
```

That works nice, unless someone has an old ActionReplay installed instead, which will crash. Check for AR first:

```
        lda $df09
        cmp $df09
        bne notaction   ;no action replay there
        cmp #$78 ;maybe it's and AR
        beq old         ;do not check for IDE64
```

```
notaction
        ...             ; regular detection, as above

old     ...             ; old byte by byte routine
        rts
```

13.2.2 $DEF1 – WRITE

Implementation:	IDEDOS only
Communication registers:	A, X, Y
Preparatory routines:	valid OPEN, CHKOUT
Error returns:	5, 7, 9 (see error codes)
Status:	$00, $80
Registers affected:	A, X, Y

Save memory to a IDE64 device. It's much faster than calling CHROUT a lot of times.

Note: It's not possible to save under I/O. (eg. saving from $D800 will save color RAM) To access RAM under the BASIC and KERNAL ROM, set $01 correctly. Saving RAM under the KERNAL ROM is not supported on SuperCPU.

13.2.3 $DEF1 – WRITE replacement

Here's an example replacement byte-by-byte routine for non-IDE64 devices, can be shortened of course:

```
        stx wnum
        sty wnum+1      ;byte to be written
```

```
        stx tmp
        sty tmp+1
        tax
        lda $00,x
        sta pointer
        lda $01,x
        sta pointer+1   ;copy pointer

        ldx #channel
        jsr chkout      ;do select output
        bcs end         ;error happened

loop    lda wnum        ;write loop
        ora wnum+1
        beq end

        ldy #0
        lda (pointer),y
        jsr chrout
        lda $90         ;status
        bne end2        ;error during write

        lda wnum
        bne at2
        dec wnum+1
at2     dec wnum

        inc pointer
        bne loop
```

```
        inc pointer+1
        jmp loop

end2    jsr clrchn
        lda #5          ;device not present
        sec
end     php
        pha
        lda tmp
        sec
        sbc wnum
        tax
        lda tmp+1
        sbc wnum+1      ;calculate
        tay             ;bytes written
        pla
        plp
        rts
```

13.2.4 $DEF4 – READ

Implementation:	IDEDOS only
Communication registers:	A, X, Y
Preparatory routines:	valid OPEN, CHKIN
Error returns:	5, 6, 9 (see error codes)
Status:	$00, $40, $42, $52, $80
Registers affected:	A, X, Y

Load data block from a IDE64 device. Much faster than a lot of

CHRIN or GETIN.

Note: This routine does not load under I/O. (e.g. reading to $D800 will overwrite color RAM)

Example:

Copy a file with READ and WRITE

```
lda #1          ; source file number
ldx $ba         ; actual device number
ldy #0          ; secondary address for read
jsr setlfs
lda #outputname-inputname
ldx #<inputname
ldy #>inputname
jsr setnam
jsr open        ; open input file

lda #2          ; destination filenumber
ldx $ba         ; actual device number
ldy #1          ; secondary address for write
jsr setlfs
lda #status-outputname
ldx #<outputname
ldy #>outputname
jsr setnam
jsr open        ; open output file

lda #<startadd
sta $fb
lda #>startadd  ; buffer start address
```

```
        sta $fc

        ldx #1          ; set input to source file
        jsr chkin

        ldx #2
        jsr chkout      ; set output to dest. file

loop    lda #$fb        ; start address is here
        ldx #<blocksize
        ldy #>blocksize ; block size
        jsr read        ; read
        bit $90         ; readst
        php             ; status to stack

        lda #$fb
        jsr write       ; write
        plp             ; status
        bvc loop        ; test end of file

        lda #2
        jsr close       ; close output file
        lda #1
        jsr close       ; close input file
        jsr clall       ; set default i/o device
        rts

inputname .text "/bin/input-file"
outputname .text "/tmp/output-file"
```

```
status  .byte 0
```

13.2.5 $DEF4 – READ replacement

Here's an example replacement byte-by-byte routine for non-IDE64 devices:

```
        stx rnum
        sty rnum+1      ;byte to be written
        stx tmp
        sty tmp+1
        tax
        lda $00,x
        sta pointer
        lda $01,x
        sta pointer+1   ;copy pointer

        ldx #channel
        jsr chkin       ;do select output
        bcs end         ;error happened

loop    lda rnum        ;read loop
        ora rnum+1
        beq end

        jsr chrin
        ldx $90         ;status
        beq ok
        cpx #$40
```

```
        bne end2        ;error happened
ok      ldy #0
        sta (pointer),y

        lda rnum
        bne at2
        dec rnum+1
at2     dec rnum

        clc
        txa
        bne end         ;end of file reached

        inc pointer
        bne loop
        inc pointer+1
        jmp loop

end2    jsr clrchn
        lda #5          ;device not present
        sec
end     php
        pha
        lda tmp
        sec
        sbc rnum
        tax
        lda tmp+1
        sbc rnum+1      ;calculate
```

```
tay ;bytes read
pla
plp
rts
```

13.3 Common programming problems

13.3.1 Serial bus specific code

Serial bus specific code won't work with IDE64 devices, it have to be rewritten to use standard kernal calls.

$F3D5 – OPEN, $F642 – CLOSE

Serial bus file open and close. Can be replaced by OPEN and CLOSE.

$FFB1 ($ED0C) – LISTEN, then $FF93 ($EDB9) – SECOND

Used to prepare the device to send data to a channel, can be replaced by CHKOUT.

$FFB4 ($ED09) – TALK, then $FF96 ($EDC7) – TKSA

Used to prepare the device to read data from a channel, can be replaced by CHKIN.

$FFA5 ($EE13) – ACPTR, $FFA8 ($EDDD) – CIOUT

Read and write a byte from and to the serial bus, can be replaced by CHRIN and CHROUT.

$FFAB ($EDEF) – UNTLK, $FFAE ($EDFE) – UNLSN

Send untalk and unlisten, can be replaced by CLRCHN.

An example replacement can be seen in Table 21.

13.3.2 Direct kernal calls

As IDEDOS uses the vector table at `$03xx`, directly calling kernal is not a wise idea.

Old	Replace with	Name
`$f34a`	`$ffc0`	OPEN
`$f291`	`$ffc3`	CLOSE
`$f20e`	`$ffc6`	CHKIN
`$f250`	`$ffc9`	CHKOUT
`$f157`	`$ffcf`	CHRIN
`$f13e`	`$ffe4`	GETIN
`$f1ca`	`$ffd2`	CHROUT
`$f4a5`	`$ffd5`	LOAD
`$f5ed`	`$ffd8`	SAVE
`$f32f`	`$ffe7`	CLALL
`$f333`	`$ffcc`	CLRCHN

Table 20: Direct kernal call replacement table

lda #0 sta $90 lda #8 ;drive 8 sta $ba lda #$6f ;channel sta $b9 lda #0 jsr $ffbd ;setname	lda #2 ;filenumber ldx $ba ;last drive ldy #15 ;channel jsr $ffba ;setparam lda #0 jsr $ffbd ;setname
jsr $f3d5 ;open	jsr $ffc0 ;open
lda $ba jsr $ed0c ;listen lda $b9 jsr $edb9 ;second	ldx #2 ;filenumber jsr $ffc9 ;chkout
lda #$49 jsr $eddd ;send	lda #$49 jsr $ffd2 ;chrout
jsr $edfe ;unlisten	jsr $ffcc ;clrchn
lda $ba jsr $ed09 ;talk lda $b9 jsr $edc7 ;tksa	ldx #2 ;filenumber jsr $ffc6 ;chkin
jsr $ee13 ;read jsr $ffd2 ;print bit $90 bvc *-8	jsr $ffcf ;read jsr $ffd2 ;print bit $90 bvc *-8
jsr $edef ;untalk	jsr $ffcc ;clrchn
jsr $f642 ;close	lda #2 jsr $ffc3 ;close

Table 21: Disk init and message display example

14 PCLink

IDEDOS also supports a virtual drive called as “PCLink”. This drive is a network drive, which makes file copying possible across computers. The server software is called “ideserv”.

The “PCLink” name can be misleading, because any computer capable of running win32 or Linux can be a host for file transfers, if it has a suitable printer port.

Programs can also directly run of the virtual PCLink drive. There are several variations for the connection:

14.1 PCLink on XPClink cable

The slowest method but the easiest, if you already have a X1541, XE1541, XM1541 or XA1541 cable. The maximal transfer speed is around ∼4500 bytes/sec.

Pin	C64	Pin	Host
2	GND	18...25	GND
3	ATN	1	STROBE
4	CLK	14	AUTOFEED
5	DATA	17	SELECT IN

Table 22: X1541 PCLink cable

This cable will not work with all printer ports. The other cable types are described in the Star Commander’s documentation, written by Joe Forster/STA.

14.2 PCLink on PC64 cable

This connection is made between the parallel port of the host computer and the user port of a C64 using a cable similar to Laplink. The layout is the same as for the PC64 cable. Please note that there are no optional wires here, so if you have an old pclink cable, then check if B–FLAG is connected! The maximal transfer speed is around ∼9 Kbytes/sec.

Pin	C64 user port	Pin	Host printer port
A, 1	GND	18...25	GND
B	FLAG in	9	D7 out
C	PB0 out	15	ERROR in
D	PB1 out	13	SELECT in
E	PB2 out	12	PAPER in
F	PB3 out	10	ACK IN in
H	PB4 in	2	D0 out
J	PB5 in	3	D1 out
K	PB6 in	4	D2 out
L	PB7 in	5	D3 out
M	PA2 out	11	BUSY in
N, 12	GND	18...25	GND

Table 23: Parallel PCLink cable

Incorrectly built cable can damage the printer and / or userport! Same for plugging the cable while the computers are turned on...

14.3 PCLink on null modem cable

It's using a null modem cable with handshake between the host computer's serial port and one port of the duart card. Communication is done at 115200 bit/s resulting in an effective transfer speed of around ∼11 Kbytes/sec.

9 pin	C64 duart port	9 pin	25 pin	Host serial port
2	RxD	3	2	TxD
3	TxD	2	3	RxD
7	RTS	8	5	CTS
8	CTS	7	4	RTS
5	GND	5	7	GND

Table 24: Serial PCLink cable

15 Command channel

This section is about the DOS commands known by IDEDOS. Some examples use the dos wedge like "@I", of course the "@" at the beginning of line is not part of the command.

15.1 File management commands

For more see section 7 Using files.

Position

Seek to file position.

Format:

"P" + CHR$(*channel #*) + CHR$(*position bits 0...7 #*) + CHR$(*position bits 8...15 #*) + CHR$(*position bits 16...23 #*) + CHR$(*position bits 24...31 #*)

Example:

Seek in file to the 100000th byte and read it. (counting begins at byte 0)

```
10 OPEN 15,12,15:OPEN 4,12,4,"FILE"
20 PRINT#15,"P"; CHR$(4); CHR$(159); CHR$(134); CHR$(1); CHR$(0)
30 GET#4,A$:CLOSE 4:CLOSE 15
```

15.2 Filesystem management commands

Initialize

Initialize the filesystem. In case of disk change it redetects the filesystem.

Format:

"I{*partition #*}"

Example:

```
@I
00, OK,000,000,000,000
```

Scratch

Delete a file, or more files. Multiple files are specified by wildcards, or by a colon, which marks the beginning of a new pattern. The exact filetype can be specified by "=". For empty directories use "RD"! File must have the DELETABLE flag set, and must be on a writable partition in a writeable directory. If the filetype is not given, it means any, so watch out!

Format:

"S{*partition #*}:*file name*{=*file type*}{,*file name*{=*file type*}}"
"S{*partition #*}{/*path*/}:*file name*{=*file type*}
{,{*partition #*}{/*path*/}:*file name*{=*file type*}}"

Examples:

Delete all files in the current working directory:

```
@S:*
01, FILES SCRATCHED,028,000,000,000
```

Delete files called “FILE” with any type:

```
@S:FILE
01, FILES SCRATCHED,003,000,000,000
```

Delete all files with type “BAK”:

```
@S:*=BAK
01, FILES SCRATCHED,009,000,000,000
```

Delete the file called “FILE,PRG”

```
@S:FILE=PRG
01, FILES SCRATCHED,001,000,000,000
```

Delete the file called “FILE,PRG” in partition 3.

```
@S3:FILE=PRG
01, FILES SCRATCHED,001,000,000,000
```

Delete all files with type “OLD” and “BAK” in directory called “STUFF”.

```
@S/STUFF/:*=OLD,*=BAK
01, FILES SCRATCHED,015,000,000,000
```

Delete all files with type “OLD” and “BAK” in directory called “STUFF”, and everything from “STUFF/BAK”

```
@S/STUFF/:*=OLD,*=BAK,/STUFF/BAK/:*
01, FILES SCRATCHED,043,000,000,000
```

Rename and move

Rename or move a file or directory. The source and destination file must be on the same partition, on a writable partition and in a writeable directory. To rename the directory header, use "R-H", it's described in section 15.6!

Moving files is not an atomic operation, that means you can lose data if you turn off the computer at the wrong time! Because the filesystem must be consistent at all time, the moving operation takes place by first writing out a non-closed version in the destination directory. If this succeeds then the original file gets removed, and finally the destination will be closed. When moving a directory, the first test non-closed entry will be a deleted one.

Format:

"R{*partition #*}{*/path /*}:*new file name*{,*file type*}
={*/path /*:}*old file name*{,*file type*}"

Examples:

Rename the file called "OLD,PRG" to "NEW,SEQ"

```
@R:NEW,SEQ=OLD,PRG
00, OK,000,000,000,000
```

Move "OLD,PRG" from directory "SOURCE" into directory "DEST" as "NEW,SEQ"

```
@R/DEST/:NEW,SEQ=/SOURCE/:OLD,PRG
00, OK,000,000,000,000
```

Lock

Change the protection flags of a file or directory. It must be on a writable partition and in a writeable directory. If the filetype is not given, it means any. It operates only on one file at a time.

Format:

"L{*partition #*}:*file name*{,*file type*}"
"L{*partition #*}{/*path* /}:*file name*{,*file type*}"

Example:

Toggles WRITABLE and DELETABLE flags on "FILE,PRG"

```
@L:FILE,PRG
00, OK,000,000,000,000
```

15.3 Partition management commands

If you want to get the list of partitions, then see section 5 Using partitions!

Change partition

Partition changing.

Format:

"CP*partition #*"
"CP" + CHR$ (*partition #*)

Examples:

Select partition 2

```
@CP2
02, PARTITION SELECTED,002,000,000,000
```

Select partition 3 the other way

```
OPEN 15,12,15,"C"+CHR$(208)+CHR$(3):CLOSE 15
READY.
```

Get partition info

Get information about a partition. The returned data format is described in Table (25).

Format:

"G-P" {+ CHR$ (*partition #*)}

Byte	Meaning
0	Partition type: 0=not available, 1=CFS
1	Unused
2	Partition number
3-18	~~Partition name~~
19-26	Unused
27-28	Size (65535)

Table 25: G-P data format

Examples:

What's the current partition?

```
10 OPEN 15,12,15,"G-P"
```

```
20 GET#15,A$,B$,C$:CLOSE 15
30 PRINT"CURRENT PARTITION IS:"ASC($C$+CHR$(0))
```

Is partition 2 there?

```
10 OPEN 15,12,15,"G-P"+CHR$(2)
20 GET#15,A$,B$,C$:CLOSE 15
30 IF A$="" THEN PRINT"NO SUCH PARTITION!"
```

15.4 Device management commands

Drive number change

If you want to temporary change the device number (till next reset) send “"U0>"+CHR$(NEW)” to the device.

Format:

"U0>" + CHR$ (*new drive #*)
"S-8"
"S-9"
"S-D"

Example:

Use device 12 as 8 from now on

```
OPEN 15,12,15,"U0>"+CHR$(8):CLOSE15
```

An easier typed variant is the “S-8”, “S-9” and “S-D”. The last one restores the default device number.

Examples:

Use device 12 as 8 from now on

```
@S-8
00, OK,000,000,000,000
READY.
```

Revert to default device number

```
@S-D
00, OK,000,000,000,000
READY.
```

Identify device

Sending "UI" or "U9" on command channel returns the dos version of the device.

Format:

"UI"
"U9"

Example:

```
@UI
73, IDE DOS BETA! IDE64,000,000,000,000
```

Reset device

Sending "UJ" or "U:" on command channel reconfigures the device.

Format:

"UJ"
"U:"

Example:

```
@UJ
73, IDE DOS BETA! IDE64,000,000,000,000
```

Power management

It's possible to enter or exit power saving mode if the device supports it.

Format:

"U0>P0"
"U0>P1"
"U0>P"

Examples:

Spin down drive

```
@U0>P0
00, OK,000,000,000,000
```

Spin up drive

```
@U0>P1
00, OK,000,000,000,000
```

Get power management state. Returned is 1 byte followed by CHR$(13).

```
10 OPEN 15,12,15,"U0>"+CHR$(208)
20 GET#15,A$,B$:CLOSE 15:A=ASC(A$+CHR$(0))
30 IF A=0 THEN PRINT"STANDBY"
40 IF A=128 THEN PRINT"IDLE"
50 IF A=255 THEN PRINT"ACTIVE"
```

Eject or load media

More useful on CDROM, DVD, ZiP drive and LS-120 than on hard disk. Don't forget to unlock the media before eject! (See Lock / Unlock media below)

Format:

```
"U0>E0"
"U0>E1"
```

Examples:

Load media (CDROM and DVD only)

```
@U0>E0
00, OK,000,000,000,000
```

Eject media

```
@U0>E1
00, OK,000,000,000,000
```

Lock or unlock media

It's possible to prevent media removal on CDROM, DVD, ZiP and LS-120 drives.

Format:

```
"U0>L0"
"U0>L1"
```

Examples:

Unlock media

```
@U0>L0
00, OK,000,000,000,000
```

Lock media

```
@U0>L1
00, OK,000,000,000,000
```

Reading time from RTC

Sending "T-RA" will read the current time in ASCII, "T-RB" in BCD, while "T-RD" in decimal.

Format:

"T-RA"
"T-RB"
"T-RD"

Byte	Meaning
0	Day of week: 0=Sunday, 1=Monday…
1	Year 00-99, 00=2000, 79=2079, 80=1980, etc.
2	Month 1-12, 1=January
3	Date 1-31
4	Hour 1-12
5	Minute 0-59
6	Second 0-59
7	AM / PM: 0=AM, else PM

Table 26: T-RB and T-RD data format

Example:

Get the current time in human readable form (day of week, month, date, year, hour, min, sec, am / pm)

```
@T-RA
SAT. 08/07/04 07:42:48 PM
```

Format disk

Formatting of floppy disks is sometimes necessary to eliminate bad sectors, or just bring the media into a usable format. Formatting changes the physical format, it does not create a filesystem on disk. Use CFSfdisk to create a filesystem!

Note that while formatting the drive is not accessible (this can take more than 30mins), fortunately it's possible to use other devices. And of course HDINIT will break formatting, so try to avoid it.

Format:

"N=*format code*"

Code	Meaning
720K	Double density disk with 720 KB capacity (LS-120)
1.2M	High density disk with 1200 KB capacity (LS-120)
1.44M	High density disk with 1440 KB capacity (LS-120)
120M	SuperDisk with 120 MiB capacity (LS-120)

Table 27: Disk format codes

Example:

Format a 1.44 MB disk

```
@N=1.44M
00, OK,000,000,000,000
```

Write protect

Software write protect switch for the entire drive. This is not permanent.

Format:

"W-0"
"W-1"

Example:

Enable write protection.

```
@W-1
00, OK,000,000,000,000
```

15.5 Direct access commands

For more see section 8 Direct access.

Identify

Format:

"B=R" + CHR$(*channel #*) + CHR$(0) + CHR$(0) + CHR$(0) + CHR$(0)

Buffer read

Format:

"B=R" + CHR$(*channel #*) + CHR$(*head #*) + CHR$(*cylinder bits 8...15 #*) + CHR$(*cylinder bits 0...7 #*) + CHR$(*sector #*)
"B=R" + CHR$(*channel #*) + CHR$(64 + *LBA bits 24...27 #*) + CHR$(*LBA bits 16...23 #*) + CHR$(*LBA bits 8...15 #*) + CHR$(*LBA bits 0...7 #*)

Buffer position

Format:

"B=P" + CHR$(*channel #*) + CHR$(*position bits 0...7 #*) + CHR$(*position bits 8...15 #*)

Buffer write

Format:

"B=W" + CHR$(*channel #*) + CHR$(*head #*) + CHR$(*cylinder bits 8...15 #*) + CHR$(*cylinder bits 0...7 #*) + CHR$(*sector #*)
"B=W" + CHR$(*channel #*) + CHR$(64 + *LBA bits 24...27 #*) + CHR$(*LBA bits 16...23 #*) + CHR$(*LBA bits 8...15 #*) + CHR$(*LBA bits 0...7 #*)

15.6 Directory handling commands

For more see section 6 Using directories.

Change working directory

All directories of the path must be executeable.

Format:

"CD{*partition #*}:*path*"
"CD←"
"CD/*path*"

Example:

```
@CD:NEWDIR
00, OK,000,000,000,000
```

Make directory

Create a directory. The new directory must be on a writable partition and in a writable directory.

Format:

"MD{*partition #*}:*directory name*"
"MD{*partition #*}{/*path* /}:*directory name*"

Example:

Create the directory called "NEWDIR"

```
@MD:NEWDIR
00, OK,000,000,000,000
```

Remove directory

Remove a directory. The directory must be on a writable partition and in a writable directory.

Format:

"RD{*partition #*}:*directory name*"
"RD{*partition #*}{/*path /*}:*directory name*"

Example:

Remove the directory called "OLDDIR"

```
@RD:OLDDIR
01, FILES SCRATCHED,001,000,000,000
```

Rename directory header

Rename the directory header. The directory must be writable and on a writable partition.

Format:

"R-H{*partition #*}:*header*"
"R-H{*partition #*}{/*path /*}:*header*"

Example:

Change the header of current directory to "NEW LABEL"

```
@R-H:NEW LABEL
00, OK,000,000,000,000
```

15.7 CDROM related commands

Read TOC

For more see section 8 Direct access.

Format:

"B=T" + CHR$(*channel #*) + CHR$(*format #*) + CHR$(*starting track #*)

Read sub-channel information

For more see section 8 Direct access.

Format:

"B=S" + CHR$(*channel #*) + CHR$(*format #*) + CHR$(*starting track #*) + CHR$(*mode #*)

Audio playback

This command will play a part of audio CD. Start and end position is specified in MSF (Minute (0-99), Second (0-59), Frame (0-74)). Note that audio CDs start after 2 seconds.

Format:

"U0>CA" + CHR$(*end frame #*) + CHR$(*end second #*) + CHR$(*end minute #*) + CHR$(*start frame #*) + CHR$(*start second #*) + CHR$(*start minute #*)

Fast forward and reverse

This command starts fast forward and reverse from the specified position until the end of disc. The direction and the position format is

specified by the mode byte, as described in Table (28).

LBA position It's specified in sectors from the beginning of the disc, first byte is the least significant, and the 4th the most.

MSF position It's the elapsed time from the beginning of the disc, first byte is the frame, then second and minute, while the 4th is reserved (0).

Track position It's specified by the stating track number, which is the 1st byte, the other 3 are reserved.

Format:

"U0>CF" + CHR$(*pos1 #*) + CHR$(*pos2 #*) + CHR$(*pos3 #*) + CHR$(*pos4 #*) + CHR$(*mode #*)

Bit	Meaning
0...3	Unused
4	0 — Forward 1 — Reverse
5	Unused
6...7	00 — LBA position 01 — MSF position 10 — Track position 11 — Reserved)

Table 28: Bits of fast forward and reverse mode byte

Example:

Fast forward from track 2:

```
PRINT#15,"U0>CF"+CHR$(2)+CHR$(0)+CHR$(0)+CHR$(0)+CHR$(128)
```

Pause, resume, stop audio playback

Format:

"U0>CP0"
"U0>CP1"
"U0>CS"

Examples:

Pause playback

```
@U0>CP0
00, OK,000,000,000,000
```

Continue playback

```
@U0>CP1
00, OK,000,000,000,000
```

Stop playback

```
@U0>CS
00, OK,000,000,000,000
```

Volume control

This command lets control the output volume of CDROM drive. Output / channel 0 is the left side, while output / channel 1 is the right.

Format:

"U0>CV" + CHR$(#) + CHR$(#) + CHR$(#) + CHR$(#) + CHR$(#) + CHR$(#) + CHR$(#) + CHR$(#)

Byte	Meaning
0	Output port 0 channel selection (0-15)
1	Output port 0 volume (0-255)
2	Output port 1 channel selection (0-15)
3	Output port 1 volume (0-255)
4	Output port 2 channel selection (0-15)
5	Output port 2 volume (0-255)
6	Output port 3 channel selection (0-15)
7	Output port 3 volume (0-255)

Table 29: Volume control format

Examples:

Reverse left and right speakers, and -6dB amplification

```
10 OPEN15,12,15
20 V$=CHR$(2)+CHR$(128)+CHR$(1)+CHR$(128)
30 V$=V$+CHR$(0)+CHR$(0)+CHR$(0)+CHR$(0)
40 PRINT#15,"U0>CV"V$
50 CLOSE15
```

Mono output on left speaker only, full volume

```
10 OPEN15,12,15
20 V$=CHR$(3)+CHR$(255)+CHR$(0)+CHR$(0)
30 V$=V$+CHR$(0)+CHR$(0)+CHR$(0)+CHR$(0)
40 PRINT#15,"U0>CV"V$
50 CLOSE15
```

Value	Meaning
0	Output port muted
1	Audio channel 0
2	Audio channel 1
3	Audio channel 0 and 1 mixed
4	Audio channel 2
8	Audio channel 3
15	All mixed

Table 30: Output channel selection

Volume settings query

This command returns the current settings from the CDROM drive in a structure described in Table (29), plus a `CHR$(13)` at the end.

Format:

"U0>CV"

Example:

Print current volume setting

```
0 PRINT"DRIVE":INPUT DR
10 OPEN15,DR,15,"U0>C"+CHR$(214)
20 GET#15,A$
30 IF A$<CHR$(16) THEN 60
40 INPUT#15,B$,C$,D$,E$:REM ERROR
50 PRINTA$B$,C$,D$,E$:GOTO 90
60 GET#15,B$,C$,D$,E$,F$,G$,H$,I$
70 PRINT"LEFT  CHANNEL:"ASC(B$+CHR$(0))
80 PRINT"RIGHT CHANNEL:"ASC(D$+CHR$(0))
90 CLOSE 15
```

Medium type

This command returns the CDROM media type in one byte, plus a `CHR$(13)` at the end.

Format:

"U0>CM"

Example:

This small program will display the media type in CDROM drive

```
0 PRINT"DRIVE":INPUT DR
10 OPEN15,DR,15,"U0>CM":GET#15,A$,B$:CLOSE15
15 IF ST<>64 THEN PRINT"NOT SUPPORTED":END
20 T=ASC(A$+CHR$(0)):A=T AND 15:B=INT(T/16)
30 IF B=7 AND A=0 THEN PRINT"NO CD IN DRIVE":END
40 IF B=7 AND A=1 THEN PRINT"TRAY OPEN":END
50 IF B=7 AND A=2 THEN PRINT"FORMAT ERROR":END
60 IF B=1 AND A<9 THEN PRINT"CD-R ";
70 IF B=2 AND A<9 THEN PRINT"CD-E ";
80 IF B=3 AND A=0 THEN PRINT"HD UNKNOWN":END
90 IF B=3 AND A=1 THEN PRINT"HD 120MM":END
100 IF B=3 AND A=5 THEN PRINT"HD 80MM":END
110 IF B>2 OR A>8 THEN PRINT"?":END
120 IF A>0 AND A<5 THEN PRINT"120MM "
130 IF A>4 AND A<9 THEN PRINT"80MM "
140 IF A=0 THEN PRINT"UNKNOWN"
150 IF A=1 OR A=5 THEN PRINT"DATA"
160 IF A=2 OR A=6 THEN PRINT"AUDIO"
170 IF A=3 OR A=7 THEN PRINT"DATA AND AUDIO"
180 IF A=4 OR A=8 THEN PRINT"MULTISESSION"
```

Drive capabilities

This command returns the drive capabilities in 14 bytes plus a `CHR$(13)` at the end.

Format:

"U0>CC"

Example:

This little longer program will display all informations this command can return.

```
0 PRINT"DRIVE":INPUT DR
10 OPEN15,DR,15,"U0>CC":DIM C(14)
20 FOR A=0 TO 14:GET#15,A$:C(A)=ASC(A$+CHR$(0)):NEXT
30 CLOSE 15:IF ST<>64 THEN PRINT"NOT SUPPORTED":END
35 PRINT"DRIVE CAPABILITIES:":PRINT
40 IF C(0) AND 1 THEN PRINT"READS CD-R"
50 IF C(0) AND 2 THEN PRINT"READS CD-RW"
60 IF C(0) AND 4 THEN PRINT"READS CD-R METHOD2"
70 IF C(0) AND 8 THEN PRINT"READS DVD-ROM"
80 IF C(0) AND 16 THEN PRINT"READS DVD-R"
90 IF C(0) AND 32 THEN PRINT"READS DVD-RAM"
100 IF C(1) AND 1 THEN PRINT"WRITES CD-R"
110 IF C(1) AND 2 THEN PRINT"WRITES CD-RW"
120 IF C(1) AND 4 THEN PRINT"WRITES SIMULATION"
130 IF C(1) AND 16 THEN PRINT"WRITES DVD-R"
140 IF C(1) AND 32 THEN PRINT"WRITES DVD-RAM"
150 IF C(2) AND 1 THEN PRINT"PLAYS AUDIO"
160 IF C(2) AND 2 THEN PRINT"COMPOSITE AUDIO/VIDEO STREAM"
170 IF C(2) AND 4 THEN PRINT"DIGITAL OUT ON PORT1"
180 IF C(2) AND 8 THEN PRINT"DIGITAL OUT ON PORT2"
190 IF C(2) AND 16 THEN PRINT"READS MODE2 FORM1"
```

```
200 IF C(2) AND 32 THEN PRINT"READS MODE2 FORM2"
210 IF C(2) AND 64 THEN PRINT"READS MULTISESSION"
220 IF C(3) AND 1 THEN PRINT"READS CDDA"
230 IF C(3) AND 2 THEN PRINT"READS CDDA WITH CONTINUE"
240 IF C(3) AND 4 THEN PRINT"RW SUPPORTED"
250 IF C(3) AND 8 THEN PRINT"RW CORRECTION"
260 IF C(3) AND 16 THEN PRINT"C2 POINTER SUPPORT"
270 IF C(3) AND 32 THEN PRINT"ISRC CODE SUPPORT"
280 IF C(3) AND 64 THEN PRINT"UPC CODE SUPPORT"
290 IF C(4) AND 1 THEN PRINT"MEDIA LOCKABLE"
300 PRINT"MEDIA CURRENTLY ";
302 IF (C(4) AND 2)=0 THEN PRINT"UN";
305 PRINT"LOCKED"
310 PRINT"MEDIA LOCK JUMPER ";
312 IF (C(4) AND 4)=0 THEN PRINT"NOT ";
315 PRINT"SET"
320 IF C(4) AND 8 THEN PRINT"MEDIA EJECTABLE"
330 PRINT"LOADING MECHANISM: ";:A=INT(C(4)/32)
340 IF A=0 THEN PRINT"CADDY"
350 IF A=1 THEN PRINT"TRAY"
360 IF A=2 THEN PRINT"POP-UP"
370 IF A=3 OR A>5 THEN PRINT"UNKNOWN"
380 IF A=4 THEN PRINT"CHANGER WITH DISCS"
390 IF A=5 THEN PRINT"CHANGER USING CARTRIDGE"
400 IF (C(5) AND 1)=0 THEN PRINT"NOă";
405 PRINT"SEPARATE VOLUME CONTROL"
410 IF (C(5) AND 2)=0 THEN PRINT"NO ";
415 PRINT"SEPARATE VOLUME MUTING"
420 IF C(5) AND 4 THEN PRINT"CHANGER REPORTS SLOT CONTENT"
430 IF C(5) AND 8 THEN PRINT"CHANGER CAN SELECT SLOT"
460 PRINT"MAXIMUM SPEED:"INT((C(6)*256+C(7)+88)/176)"X"
```

```
470 PRINT"VOLUME LEVELS:"C(8)*256+C(9)
480 PRINT"BUFFER SIZE:"C(10)*256+C(11)"KB"
490 PRINT"CURRENT SPEED:"INT((C(12)*256+C(13)+88)/176)"X"
```

15.8 Misc commands

Memory read

It's included for compatibility. It reads from a fake ROM filled with the message "`IDE64 CARTRIDGE `" or depending on the "CMD emulation" setting in the CMOS Setup utility it can also be the text "`CMD HD EMULATED IDE64 CARTRIDGE `". (see section 3.1)

Format:

"M-R" + CHR$(*address bits 0...7 #*) + CHR$(*address bits 8...15 #*) + CHR$(*number of bytes #*)

Memory write

It's included for compatibility, write to `$77`–`$78` changes drive number, otherwise it has no effect.

Format:

"M-W" + CHR$(*address bits 0...7 #*) + CHR$(*address bits 8...15 #*) + CHR$(*number of bytes #*) + CHR$(*first byte #*) + ...

Validate

It's included for compatibility, has no effect. Use the CFSfsck utility to check filesystem integrity.

Format:

"V{*partition #*}"

16 Command channel error messages

These are the possible returned error messages on channel #15 with their short descriptions.

`00, OK,000,000,000,000`
Last action was successful. No errors since last read.

`FILES SCRATCHED,xxx,yyy,000,000`
xxx+yyy*256 files where deleted.

`02, PARTITION SELECTED,xxx,000,000,000`
Partition number #xxx was selected.

`20, READ ERROR,dp1,dp2,dp3,dp4`
Unrecoverable error. `dp1`, `dp2`, `dp3`,`dp4` is drive specific address of sector.

`21, READ ERROR,dp1,dp2,dp3,dp4`
Timeout while reading.

`22, READ ERROR,dp1,dp2,dp3,dp4`
Unformatted media.

`23, READ ERROR,dp1,dp2,dp3,dp4`
Medium error. Bad block. Read aborted.

`25, WRITE ERROR,dp1,dp2,dp3,dp4`
Medium error. Verify error. Bad block. Write aborted.

`26, WRITE PROTECT ON,000,000,000,000`
Write protected file, directory, partition. Data protect error during write, write protected media.

`27, ACCESS DENIED,dp1,dp2,dp3,dp4`
Cannot read the read the content of a non-readable directory. Data protect error during read.

`28, WRITE ERROR,dp1,dp2,dp3,dp4`
Timeout while writing.

`29, DISK CHANGED,000,000,000,000`
Disk change detected.

`30, SYNTAX ERROR,000,000,000,000`
Syntax error in command, something is missing.

`31, UNKNOWN COMMAND,000,000,000,000`
Unknown or unimplemented command. Command not supported by drive.

`32, SYNTAX ERROR,000,000,000,000`
Command buffer overflow.

`33, SYNTAX ERROR,000,000,000,000`
Illegal character in filename.

`34, SYNTAX ERROR,000,000,000,000`
Missing filename for command.

`39, PATH NOT FOUND,000,000,000,000`
Path not found. Tried to move file to different directory using rename. Possible link loop found. Overflow during link expansion.

`60, FILL OPEN,000,000,000,000`
Better close opened files, before remove or move. Also do not try to remove the working directory.

`62, FILE NOT FOUND,000,000,000,000`
File not found. Loading or opening a file not supported on this filesystem.

`63, FILE EXISTS,000,000,000,000`
Cannot create file or directory, it already exists. Cannot rename file, it already exists.

```
64, FILE TYPE MISMATCH,000,000,000,000
```
Cannot change this file type to the specified with rename. Cannot create file with this type. This type of access on file is not implemented.

```
66, ILLEGAL REQUEST,dp1,dp2,dp3,dp4
67, ILLEGAL REQUEST,dp1,dp2,dp3,dp4
```
Illegal request. Tried to access beyond end of disk. Track 0 not found. No such sector.

```
70, NO CHANNEL,000,000,000,000
```
Channel number incorrect for direct access commands. Channel number incorrect for the seek command. Tried to reopen an already open channel. No more free buffers left. Invalid direct access channel. Invalid directory read channel.

```
71, DIR ERROR,dp1,dp2,dp3,dp4
```
Directory header not found. There's no partition directory for this filesystem.

```
72, PARTITION FULL,000,000,000,000
```
There's no more space left on partition. Tried to create more than 1023 entries in a directory.

```
73, IDE DOS Vx.xx IDE64,000,000,000,000
73, IDE DOS Vx.xx CDROM,000,000,000,000
73, IDE DOS Vx.xx FDD64,000,000,000,000
73, IDESERV x.xx PCLINK,000,000,000,000
```
Dos version, and drive type.

```
74, DRIVE NOT READY,000,000,000,000
```
Drive not ready. No disk in drive. Command aborted by drive. Unexpected response from drive.

```
75, FORMAT ERROR,000,000,000,000
```
Unknown filesystem on disk.

`76, HARDWARE ERROR,000,000,000,000`
Drive reports hardware error. Cabling problem, unreliable communication.

`77, SELECTED PARTITION ILLEGAL,xxx,000,000,000`
Partition number #xxx not found or illegal.

17 Compatibility

- **Commodore serial devices, datassette**
  Of course they work. There's also fastload support for 1541, 1570, 1571 and 1581. Datassette is supported, if kernal supports it.
- **SuperCPU**
  The SuperCPU IDEDOS expects emulation mode with direct page starting at the 0th byte of bank 0, like normal. The high byte of register A is not preserved during operation. Tested on SCPU64V2 with SuperCPU DOS 2.04.
- **RetroReplay, Action Replay, Final cartridge**
  These hardwares use the same I/O space as the IDE64 cartridge, so they won't work.
- **RRnet, SilverSurfer**
  Works, but only if RetroReplay is jumpered to be flashed. (this way there's no I/O space conflict) This may require a cartridge port expander.
- **MMC64**
  No conflict, but needs support in IDEDOS.
- **RamLink**
  Probably does not work on C64, but needs more testing. RL-DOS won't work with IDEDOS if using SuperCPU.
- **CMD HD / FD / etc. (serial)**
  As these are serial devices they should work. They are supported by the manager too.
- **64HDD**
  With some versions of the 64HDD virtual serial bus drive emulation program you have to turn off the "Disk fastloader" option in the CMOS Setup. (The problem is that 64HDD simply can't deal with the multiple channels open at the same time on a drive case...)

- **C128**
  Works in C64 mode. Initializes VDC, clears $D02F–$D030. The cartridge is not designed for 2 MHz operation, so do not call IDE64 routines in 2 MHz mode! C128 keyboard is available if cartridge has cartconfig register and support is compiled in. (Numeric keypad, cursor keys, esc, tab, linefeed are available, help calls monitor, alt and noscroll unused)

- **JiffyDOS**
  It's detected. (tested with v6.01) The built-in DOS Wedge has higher priority, if you want to use the JiffyDOS one then disable it in the CMOS Setup utility!

  Loading, saving and file reading in manager is accelerated if the drive has JiffyDOS, even if there's no JiffyDOS ROM installed in the computer (selectable at compile time). If you get sometimes a "?LOAD ERROR" during the load of directory from a JiffyDOS drive, then that's not a bug in IDEDOS. The original JiffyDOS load routine trys to workaround this by retrying IECIN, and this causes an endless loop until STOP is pressed. I prefer getting an error over waiting forever...

- **DolphinDOS**
  Loading a file and copying in manager is accelerated if the drive has DolphinDOS and parallel cable, even if there's no DolphinDOS ROM installed in the computer (selectable at compile time). Saving is slower then usual, around 2 KB/s.

- **NTSC, PAL**
  Both works.

- **REU**
  Works, not touched. RamDOS does not work, but if someone is interested it's possible to support it.

- **Second SID**

Supported if installed at $D420. (volume 0 is set on STOP + RESTORE) The address is changeable at compile time.

- **+60K**
  Works, not touched.
- **IEEE-488, SwiftLink**
  IEEE-488 does not work, because the IDE64 cartridge cannot manipulate the EXROM line. Also SwiftLink is mirrored in the whole IO area, but can be modified.
- **Turbo 232**
  Works. (?)
- **Software**
  IDEDOS does not support auto starting programs, so use the KILL command before loading. Also programs using serial bus specific routines, custom loaders (IRQ or fast loaders), and direct disk access won't work.
- **CHS or LBA hard disk**
  Use LBA formatted hard disks if possible, it's much faster to scratch files, check filesystem, etc. because there's no CHS → LBA → CHS translation, which is slow due to multiplications and divisions required for the conversion.

18 Updating IDEDOS

The IDE64 interface cartridge is equipped with 64 KiB or 128 KiB flash memory (AT29C512 or AT29C010) for the firmware. The content of flash memory is non-volatile and can be updated, so you can always run the latest firmware with the new features and bug fixes.

Updating IDEDOS is very easy and does not require any special skills. The cartridge was designed so that it's always possible to update the firmware even if it was messed up, so you can't render your cartridge unusable by an interrupted update or wrong firmware.

With a SuperCPU equiped machine it's necessary to change between the two versions of IDEDOS depending on the presence of SuperCPU. Updating firmware every time you want to use IDE64 with or without the SuperCPU can be frustrating. By using a 128 KiB flash memory (AT29C010) it's possible to change the firmware fast and avoid time wasting updates. (included in version V3.4+ of the cartridge, upgrade of older cartridges is possible but requires soldering and electronic skills!)

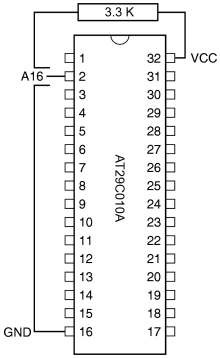


Figure 8: The 128 KiB perom upgrade for older cartridges.

Here's a short guide to updating IDEDOS:

1. Get the latest Perom programmer and the new firmware and copy both to a serial bus device. (eg. floppy drive, CMD HD, etc.) If having a XPCLink cable it's also possible to use it for update. (and it's much faster)

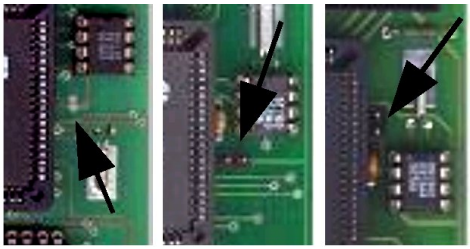


Figure 9: Location of the PGM pin on different cards

2. Plug the cartridge into the expansion port first, while the computer is switched off. Removing of SuperCPU is not necessary, it not disturb the process, and it also speeds up the update (JiffyDOS and 20 MHz), however if you are having trouble you may remove it.
3. There are two squares near the big Lattice chip and the 8 pin DS1302 called PGM. By connecting these 2 squares on board together you can enable flash update. (use screwdriver) Newer version of the IDE64 cartridge have pins instead of squares, and you have to use a jumper to connect them. (see picture)
4. If using a greater than 64 KiB flash memory, then select the bank you want to program by the switch! Never change bank or set the programming pins while IDEDOS is running! Or you will trash your disks seriously.
5. Now you can turn on your computer or press reset if it's already on. The green led should blink now, except if you have your SuperCPU enabled, then it's off. If the led is steady on it means the flash write

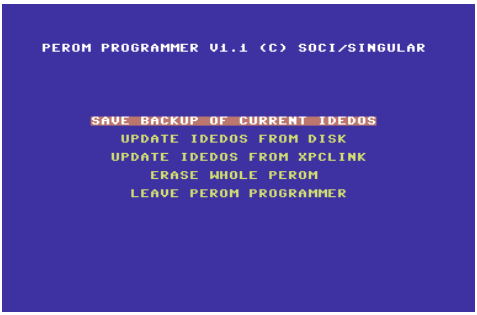


Figure 10: The perom programmer utility

protect is still enabled, this case retry from the beginning. (it's not easy to connect the 2 squares on older cartridges for the first try...)

6. The normal C64 screen should appear, the only difference is that there are now only "`30719 BYTES FREE`". Now you can remove the screwdriver or jumper.
7. Load the Perom programmer and start it. It should start with the screen shown in the picture. (if using v1.1) Now you may backup your current IDEDOS if you want, and update to the latest from XPCLink or disk. Newer IDEDOS versions may come compressed, if the file is less than 259 blocks then you must use version 1.2 of the programmer. Now select the action with the CRSR keys and hit RETURN. Erasing of perom chip is not required before programming, so do not select it unless you have reasons to do so! (it cleans out both 64 KiB banks!)
8. After selecting one of the update menu items the program will ask for device number and filename. Leaving the filename blank cancels the update. If the program says it cannot find the perom then make sure

you removed the jumper! (also it's not recommended using pencil for connecting the squares because the remaining carbon may permanently connect them unless cleaned)

9. If the filename was correct and the PCLink server was running then the programming should start. A box filled with characters will appear on screen each character representing a 128 byte sector of flash memory. During the rewrite this area will be filled. If finished you get an "Update successful" message. If hexadecimal numbers appear in the right corner of the screen and you get "Update failed" then it's meaning that you exceeded the typical 10000 rewrites for some sectors, which is very unlikely. (or the flash memory is just failing for some other unknown reason) Also disk errors may interrupt the update, this case get a new disk and retry the update.
10. Now if ready ("Update successful") you may leave the program and press reset to start the new firmware. If using older versions of the firmware (before 0.9x) then power cycle the computer!

19 Frequently Asked Questions

I copyed some programs from floppy to IDE64, but some of them got shorter by a few blocks. Is this a bug?

No, it's just a difference of block size. Traditional CBM / CMD equiment has a blocksize of 254 bytes, while IDE64 devices have a virtual 256 byte blocksize (in reality it's 512 or 2048 bytes depending on the media used). A program which is 49920 bytes long will be 197 block long on floppy and 195 on IDE64.

Is it true that IDE64 can only be used to store one filer games?

Unfortunately most multipart games are written too 1541 or serial bus specific. If you want your favourite game fixed for IDE64, then ask someone who is able to do this. Looking at the IDE64 warezsite will give you some hints about these persons or groups ;-) And no, you can use IDE64 for much more!

Will IDE64 read/write my DOS formatted floppy with LS-120, or my CompactFlash card from my camera?

Yes, IDEDOS can read FAT12/16/32 filesystems up to 128 GiB with or without partition table up to 8 partitions per drive. Cluster sizes of power of two from 0.5 to 64 KiB will work. Only short filenames are supported. Do to memory limitations there won't be direct write support included in IDEDOS, this must be coded as an external application. And no, NTFS won't be ever supported. (unless you code it)

How comes all the stuff to a IDE64 HDD, LS-120, ZIP disk, CompactFlash card?

The easiest way is to get a CDROM and burn all your stuff to CD, and then use the builtin filemanager to copy files. Also you can use Star Commander or similar utility with a floppy drive. You can use a virtual serial bus drive emulation program like 64HDD too. Or build

/ buy a PCLink cable and use that for the transfer. Also it's possible to use Contiki or Wings with an ethernet cartridge and download the stuff from the Internet. The fastest method is to use Linux or MAC to create the filesystem including all the files with CFSutils. Also you can use VICE on Linux to transfer to files when the emulated disk is the blockdevice with the correct geometry.

Ok, now I want to backup my IDE64 device. What are the possibilities?

As the filemanager supports recursive copying it's only a matter of selecting all the directories you want, and then copy it to another device. (like another HDD, ZIP disk, LS-120 disk, CompactFlash card, CMD drive, floppy, 64HDD, etc.) Beware of limited filename / directory support of non-IDE64 devices! Also CFSutils can extract all your files from an IDE64 device on Linux, or PCLink. Of course you can always use 'dd' to create an image on Linux.

20 GNU Free Documentation License

GNU Free Documentation License

Version 1.1, March 2000

Copyright (C) 2000 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

C.1. PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

C.2. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

C.3. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

C.4. COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a

publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

C.5. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the

Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

C.6. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the

various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

C.7. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

C.8. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

C.9. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant

Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

C.10. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document 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.

C.11. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation 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. See [http://www.gnu.org/copyleft/] http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

C.12. How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright (c) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation

> License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front-Cover Texts, write "no Front-Cover Texts" instead of "Front-Cover Texts being LIST"; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

21 Appendix A – The Short Bus

This section is about the Short Bus connector of the IDE64 cartridge. It's a 34 pin connector containing a selection of processor and some extra decoded signals. It was designed for connecting extra hardware to your IDE64 cartridge. Although it's looking similar to a pc floppy connector but it is not a floppy controller interface, so never connect a floppy drive or other non Short Bus hardware, or you'll damage your C64 or IDE64 card! Here's a short description of Short Bus devices I'm aware of.

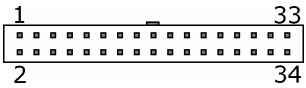


Figure 11: Short Bus female cable connector

ETH64 – Ethernet card

A LAN91C96 chip based ethernet card. This chip features: Full duplex switched ethernet; Supports enhanced transmit queue management; 6 K bytes of on-chip RAM; Supports IEEE 802.3 (ANSI 8802-3) ethernet standards; Automatic detection of TX / RX polarity reversal; Enhanced power management features; SimulTasking early transmit and early receive functions; Enhanced early transmit function; Receive counter for enhanced early receive; Hardware memory management unit; Automatic retransmission, bad packet rejection, and transmit padding; External and internal loopback modes; Four direct driven LED outputs for status and diagnostics.

There are 2 jumpers on board: JP1 for selecting address space `$DE00`–`$DE0F` (1-2) and `$DE10`–`$DE1F` (2-3), and JP2 for enabling IRQ generation. It's supported by Contiki, Wings and some other software.

For more information visit http://www.ide64.org/
Chip datasheet can be found at http://www.smsc.com/

DUART – dual port RS-232 interface

A XR68C681 based dual RS-232 card featuring: Two full duplex, independent channels; Asynchronous receiver and transmitter; Dual buffered transmitter, quadruple-buffered receiver; Programmable stop bits in 1/16 Bit increments; Internal baud rate generators with 23 different baud rates from 50 to 115200; Independent baud rate selection for each transmitter and receiver; Normal, autoecho, local loopback and remote loopback modes; Multi-function 16 bit counter or timer; Interrupt output with eight maskable interrupt conditions; Interrupt vector output on acknowledge; 8 general purpose outputs; 6 general purpose inputs with change of states detectors on inputs; Standby mode to reduce operating power.

There are 2 jumpers on board: JP1 for selecting address space `$DE00`–`$DE0F` (1-2) and `$DE10`–`$DE1F` (2-3), and JP2 for enabling NMI generation. It's supported by Contiki, Wings, Novaterm9.6 and some other software.

For more information visit http://www.ide64.org/
Chip datasheet can be found at http://www.exar.com/

DigiMAX – 4 channel 8bit DAC

A MAX506 based 4 channel 8 bit digital to analog converter card. Simple programming interface, 4 registers represent the four outputs, the written byte will appear as a voltage level between 0V and 5V. The output comes out on 2 jack plugs. This card is supported by Modplay, Wings and maybe some other programs. The base address is selectable by jumper: IO1 `$DE40`–`$DE47` or IO2 `$DE48`–`$DE57`.

For more information visit
http://www.jbrain.com/vicug/gallery/digimax/
Chip datasheet can be found at http://www.maxim-ic.com/

ETFE – Ethernet card

CS8900 based ethernet card, featuring: Full duplex operation; 4 Kbyte RAM buffer for transmit and receive frames; Automatic polarity detection and correction; Automatic re-transmission on collision; Automatic padding and CRC generation; Automatic rejection of erroneous packets; Boundary scan and loopback test; LED drivers for link status and LAN activity; Standby and suspend sleep modes.

The ethernet card has one jumper only, which enables chip reset. The jumpers on the Short Bus interface: JP7 selects address space `$DExx` (1-2), `$DFxx` (2-3). If JP7 is set to 1-2, then JP5 selects `$DE00`–`$DE0F` (1-2), `$DE10`–`$DE1F` (2-3). If the address space is set to `$DFxx`, then JP8 selects the exact memory location. The card works fine with Contiki and software supporting the original TFE card.

For more information visit http://c64.rulez.org/etfe/
Chip datasheet can be found at http://www.cirrus.com/

Pin	Name	Description
1	GND	Ground
2	Vcc	+5V
3	/Reset	Reset signal, active low
4	CSEL4	Chip select signal, active high $DE58–$DE59
5	R/W	Read/Write signal from processor
6	/CSEL3	Chip select signal, active low $DE48–$DE57
7	FI2	Fi2 clock signal from processor
8	/CSEL2	Chip select signal, active low $DE38–$DE47
9	BA	Bus Available, control signal from VIC
10	/CSEL1	Chip select signal, active low $DE10–$DE1F
11	DOT clock	Clock signal from VIC
12	/CSEL0	Chip select signal, active low $DE00–$DE0F
13	/NMI	Non maskable interrupt, active low
14	/IO2	Chip select signal, active low
15	/IRQ	IRQ, active low
16	/IO1	Chip select signal, active low
17	D7	Data bus bit 7
18	A7	Adress bus bit 7
19	D6	Data bus bit 6
20	A6	Adress bus bit 6
21	D5	Data bus bit 5
22	A5	Adress bus bit 5
23	D4	Data bus bit 4
24	A4	Adress bus bit 4
25	D3	Data bus bit 3
26	A3	Adress bus bit 3
27	D2	Data bus bit 2
28	A2	Adress bus bit 2
29	D1	Data bus bit 1
30	A1	Adress bus bit 1
31	D0	Data bus bit 0
32	A0	Adress bus bit 0
33	Vcc	+5V
34	GND	Ground

Table 31: Short Bus pinout

22 Appendix B – More information

Online resources about the IDE64 cartridge and related material. It's just a short collection, so you may also use your searching skills to get more ;-)

22.1 Related Internet sites

The IDE64 project homepage http://ide64.org/
The IDE64 project's homepage with information about the cartridge, peripherals, and lot more.

The IDE64 Information Portal http://news.ide64.org/
The latest news concerning IDE64.

The IDE64 warez site http://singularcrew.hu/ide64warez/
Lots of stuff to fill your empty disks.

The IDEDOS 0.9x page http://singularcrew.hu/idedos/
The latest version of IDEDOS can be found here.

The IDE64 list http://ide64.wwww.dk/
Subscribe to the list and get your questions answered.

The IDE64 archive http://singularcrew.hu/ide64archive/
The archive of the IDE64 list.

22.2 Distributors

Worldwide distributors of the IDE64 cartridge and related hardware (in alphabetical order)

C64Hardware (USA) http://c64hardware.com/

Eric Warthan
1821 Jackie Way, Chino Valley, AZ 86323, United States of America
phone: (928) 710-6659
E-mail: eric.warthan@c64hardware.com

GO64! (Germany) http://www.go64.de/

CSW-Verlag
WeidenstraSSe 13/1, 71364 Winnenden, Deutschland
phone: +49 (0) 7195/61120
E-Mail: info@go64.de

Modernity Inc. (Canada) http://wings.webhop.org/

Greg Nacu
phone: 613.531.0383
E-mail: gregnacu@kingston.net

Protovision (Germany) http://www.protovision-online.de/

Oliver Foerster
Koenigsberger Str. 12, 35576 Wetzlar, Germany
E-mail: poison@protovision-online.de

Index