



## High Definition Flux Sampler for USB

© 2009-2014 István Fábíán and KryoFlux Products & Services Limited

KryoFlux DiskTool Console (DTC) and KryoFlux firmware engineered and written by István Fábíán; additional product design and documentation by Christian Bartsch; Linux port by Adam Nielsen; Mac OS X port by Alexander Coers; Amiga OS 4 port by Marcus Comstedt; DiskTool Console UI by Kieron Wilkinson; hardware layout by Lars Reichel, Stefan Herzog and Olimex Ltd; Logo by Christian Krapp; project support by the Softpres team

Original hardware design "Cyclone20" and proof of concept by Richard Aplin

Thanks to our beta testers, especially Dirk Verwiebe. Additional thanks to Jean-François del Nero, Toni Wilen, those we forgot to mention and the communities of English Amiga Board (<http://eab.abime.net>) and Amiga & Phoenix Community (<http://a1k.org>) for morale, support and ideas.

Manual Revision 1.11 (2014-10-10)

[www.kryoflux.com](http://www.kryoflux.com) | [forum.kryoflux.com](http://forum.kryoflux.com)

For commercial use: [kryoflux.com/licensing](http://kryoflux.com/licensing) | [licensing@kryoflux.com](mailto:licensing@kryoflux.com)

**Disclaimer:** This is commercial software, which is free for private, non-commercial use. Please see licence.txt for detailed information. The hardware design itself is free to explore and experiment with. Notwithstanding of the terms of the licence you are allowed to change, modify and improve the hardware design provided as long as you make your changes public and freely available as well. The design may not be exploited commercially (e.g. PCBs or assembled units sold where money is changing hands, regardless of profit) unless approved by us. Individuals interested in building and/or labelling the the hardware as "KryoFlux" and/or "designed by Softpres" must get prior written approval. This is to make sure an official hardware layout is used to prevent people from buying broken hardware associated with our name and trademarks. The Software Preservation Society, Softpres and KryoFlux are trademarks of The Software Preservation Society and KryoFlux Products & Services Ltd. The following information is provided as is. Any hardware made according to the enclosed schematics is built at the user's own risk. There are no guarantees that the information contained herein is complete or correct. All (registered) trademarks are the property of their respective owners and are used for informational purposes only.

**Usage:** Read, convert, store and write contents of various legacy disk formats, including but not limited to: Acorn Electron, Apple, Amstrad CPC, Archimedes, Atari 8-bit, Atari ST, BBC, Commodore 64, Commodore Amiga, MSX, IBM PC, PC-8801, Sam Coupe, Spectrum, E-MU Emulator II and many others.

**Preface:** KryoFlux by definition refers to the concept of the following project as well as the hardware design itself. While exact details on the hardware and the design are contained in the hardware section below, one should note that KryoFlux is based on the concept of having a small interface adapter using an ATMEL ARM CPU doing the actual sampling. The software side is divided into a dedicated driver for various flavours of Windows, the DiskTool Console (DTC), a flexible software for capturing flux transition data, and the firmware for the ARM board.

**Introduction:** While today's computers store data on huge hard disks, optical media or even now solid state drives, legacy computers utilized cassettes and floppy disks. Whereas data stored on compact cassettes can be easily sampled using a tape recorder and a sampling device, like a standard sound card found in any modern PC, floppy disks have several shapes and sizes and even more ways to actually store the data on them. Standard PC floppy controllers actually try to interpret and analyse the data before handing it over to the operating system. While some controllers can be tricked into delivering more – raw – data as they should, most of them simply can not be used to read anything but IBM PC compatible formatted media using MFM coding.

KryoFlux replaces any standard controller and makes data from an attached disk drive available as a flux data stream.

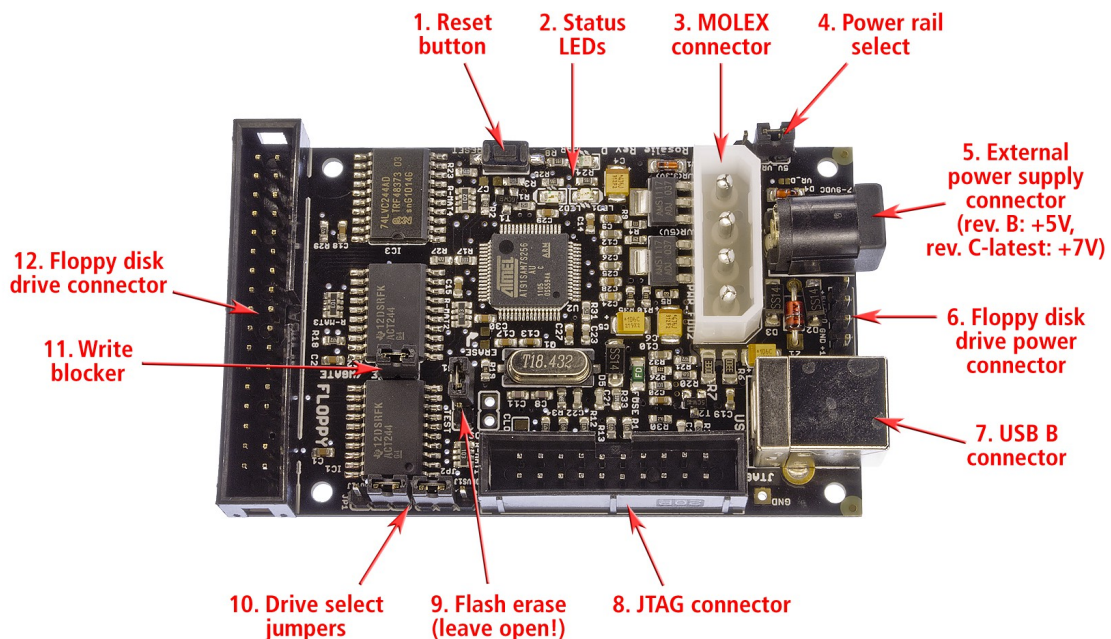
Every magnetic disk, regardless of type or size, stores data by changing the orientation of ferro oxide particles bound onto a durable and flexible plastic platter. The data itself is represented as "flux transitions" aka "flux reversals" which indicate a change of the polarity of the magnetic field. Because it is impossible to actually read the orientation of the particles on the disk surface using the head designs used, the only way to define data is by flux changes. This requires the disk to be spinning because without movement, no AC current is induced in the head. The actual data is normally coded using a scheme like FM, MFM or GCR. While MFM is the most popular scheme (in fact it just survived long enough) used on floppy disks, there are many other ways to encode and represent logical 0 and 1. Error detection and error correction is beyond what is stored in fluxes – both need interpretation of the signal and knowledge about the scheme used for writing to determine if the readout is correct or not.

While optical media produced in a pressing plant can last for ages so to speak, magnetic media has a proven life span somewhere between five to 30 years, with the latter only applying to media which was stored under ideal conditions. The higher the capacity of the platter, the higher the risk of the media failing early.

**System Requirements:** Computer with 32bit or 64bit flavour Windows (XP or later), Mac OS X (10.5 or later), Linux or Amiga OS 4; Dual Core, Atom or equivalent processor running at 1.6GHz or more; 1GB of RAM; a native USB 2.0 port; free hard disk space to store tools (~10MB) plus dump data. For best results, KryoFlux must be attached directly to the computer without any hubs or cable extensions inbetween. Due to the precise timing required, results with hubs can be mixed with the possibility of complete failure as well!

You also need a floppy disk drive with a standard 34 pin connector. Please note that KryoFlux was mainly developed for HD 3.5" ("1.44MB") and HD 5.25" ("1.2MB") drives. It also works well with selected 3" (e.g. Amstrad FDI-1) and 8" (e.g. Shugart 851; might require additional adapter) drives. There is a broad range of variants, with some "dinosaurs" not being very keen on standards. It is therefore possible that certain brands or models, especially old drives, may not work with the board. Solutions range from modifying software to modifying hardware.

This manual deals with the pre-built and fully assembled board distributed by KryoFlux Products & Services Limited. You can buy the unit directly from us via [kryoflux.com](http://kryoflux.com). Please note that only units sold by KryoFlux Products & Services Ltd come with support (as indicated).



(1) Reset button: If the board does not function or hangs after usage, press this button to reset the board.

(2) LEDs: There are three LEDs on the board. The LED on the upper right (red) should light up all the time when the unit is on. The LEDs to the lower left and right (yellow and green) are off as long the unit has not been used in a session. As soon as the firmware has loaded, the LEDs start to fade alternatingly. The green LED signals firmware activity, while the yellow one indicates an active USB connection.

(3) MOLEX power connector: KryoFlux is a fully bus powered device. Therefore no external power is needed. For special purposes the board allows to be powered externally. It is even possible to distribute power to an attached device (see 5.). Please note that the power rail for +5V is directly connected to the device's CPU. A bad (cheap, unreliable, broken) power supply can damage your board as well as external devices. The PSU must deliver a minimum of 1A per power rail (+5V/+12V). Check the orientation before attaching the plug. *Incorrect orientation of the cable will DESTROY your KryoFlux board and/or your drive. You will also void your warranty (prebuilt boards).*

(5) DC power connector: Standard power connector to supply +5V (rev. B board) or +7 to +9V (rev. C board and later; will internally be transformed to +5V) DC to the board (if desired). Useful when powering a 3.5" drive through the board, as these usually don't need +12V. The PSU must deliver a minimum of 1A, tip is hot, shield is ground. *Do not connect more than +5V DC to a rev. B board or more than +9V DC to a rev. C (or later) board! You will destroy the board and other equipment as well. You will also void your warranty.*

(6) Berg power connector to power a drive – or two with y-cable – via the board: You must use standard Berg connectors (also – incorrectly - referred to as Molex mini) to connect the drive to the board. On the picture above, +12V DC is to the bottom, so the yellow cable of the connector MUST face to the bottom as well, with the red cable facing up (+5V

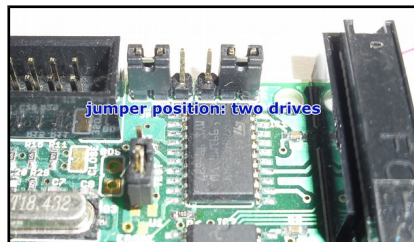
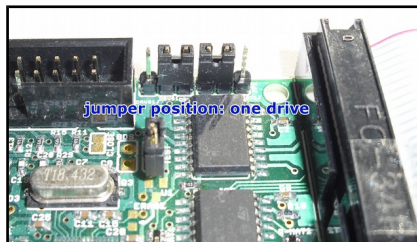
DC). *Incorrect orientation of the cable will DESTROY your KryoFlux board and/or your drive! You will also void your warranty (prebuilt boards).*

(7) USB B connector: KryoFlux connects to the computer with a USB A to USB B plug. The board (NOT an attached drive!) is solely powered through USB.

(8) JTAG connector (not used): This connector is for development purposes and advanced servicing only and can be ignored.

(9) Firmware erase jumper: KryoFlux uses an ATMEL CPU as the core of its system and can be booted from internal flash memory. KryoFlux does not flash firmware onto the device. Instead, it is downloaded at the beginning of each session (it's so fast, you won't even notice). If some other application accidentally writes something into the flash, unplug the device. Set the erase jumper to on. Connect the device, wait at least ten seconds. Now unplug the device and set the jumper to off again. KryoFlux is now back to normal.

(10) Drive select jumpers: Floppy cables usually have two sections for connecting drives, each of them has two connectors (one for 3.5" drives, the other one for 5.25" drives). You must only connect one drive to one section at a time. The section where the cable is twisted over (at the very end of the cable) is for drive 0 (which used to be drive A: in PCs). The other section is for drive 1 (which used to be drive B: in PCs).



(12) Floppy disk drive connector: This socket is for the other end of the drive cable. If it has a small nose, make sure its orientation matches the gap in the socket. If not, please check for pin 1, which is marked. Make sure line 1 (usually signalled by a colored cable) is pointing towards the drive select jumpers (no. 8).

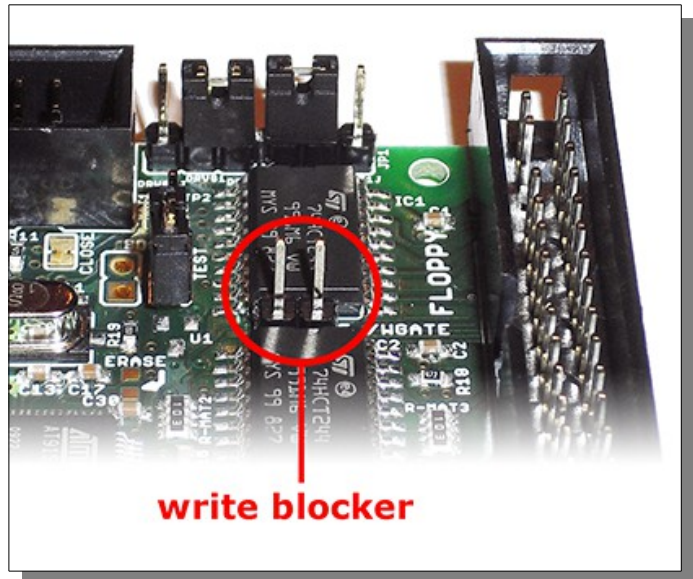


**Power rail select:** Revision C and later of the KryoFlux board come with a regulated external +5V power rail. All external power that is fed into the board via the MOLEX (3) and DC power (4) connector is regulated for a stable +5V DC power supply. This can come in handy for special usage scenarios and will also ensure that the bus driver ICs are operated at precisely +5V. While the DC power connector is always regulated (therefore rev. C boards or later need +7V to +9V DC present at the DC power connector), the routing from the MOLEX connector can be adjusted via a jumper. The setting on the left will transform +12V to precisely +5V. The opposite setting will route +5V or whatever is present at the +5V rail of the MOLEX connector. This rail has no protection

diode, so be sure to not experiment with polarity. *More than +5V DC or wrong polarity will fry your board!* We recommend keeping the jumper at the position shown on the picture at all times.

**(11) Write blocker:** With the introduction of writing to the KryoFlux host software (DTC) and firmware, protection of media on the hardware level became necessary. The commercial marketplace offers special devices that can be put between a floppy disk drive and a controller to prevent accidental writes. This functionality has been added to the KryoFlux hardware.

Revision D and later offer a built-in write blocker. The write block can be enabled by removing the jumper for WRITE GATE. After it has been removed KryoFlux can not write to disk, regardless of media and protection tab. Putting the jumper in place will enable writing again. The picture shows the board with the write block enabled. If you are using KryoFlux in a preservation environment at an archive, library or museum we strongly recommend setting this jumper as shown. This setting can not be circumvented in software.

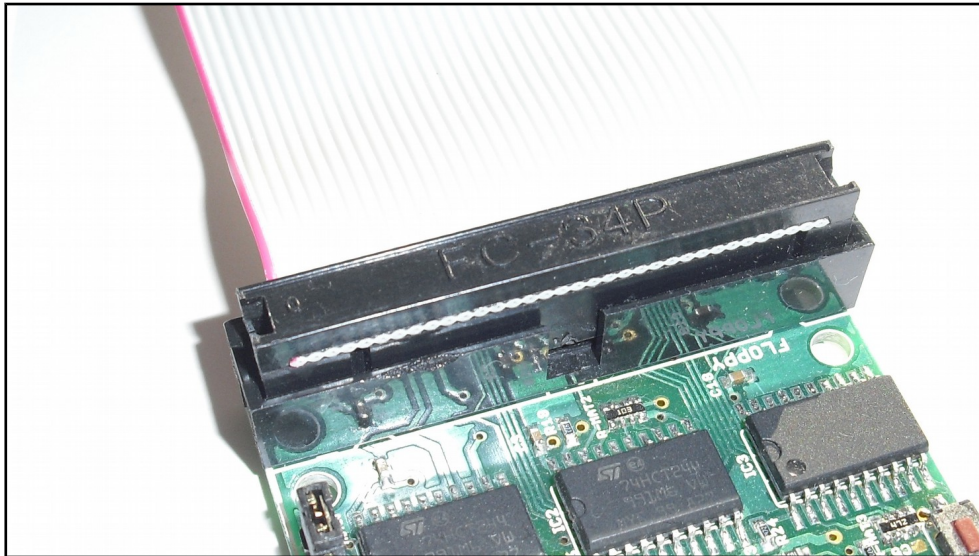


## Setting up the hardware

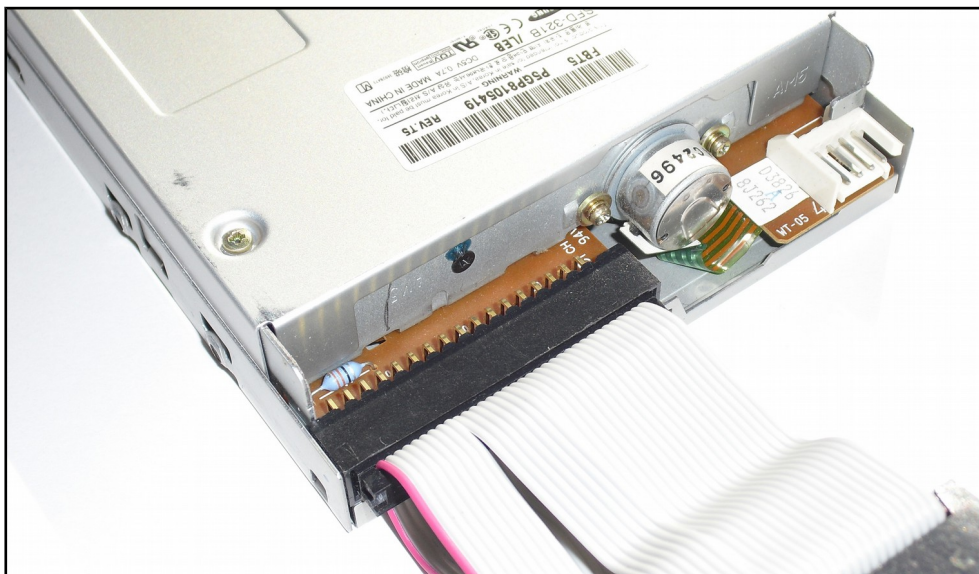
*Important! Always make the drive – board - connection first, then plug the power (PSU & USB). Power always comes last! You are connecting two systems with different grounds, so ground (connected via the floppy data cable) must always come first. Never ever connect or remove the floppy data cable while the drive and / or the board are still powered. Doing so will void your warranty and you risk damaging drive and board. You have been warned!*

Place the KryoFlux board and the disk drive on a flat, non-conductive surface. Make sure you will not short circuit the device by placing it on a metal table or similar. Connect KryoFlux and drive with floppy data cable.

*Always unplug and disconnect from mains when not in use! Do not leave unattended!*



Check for correct orientation, the marked wire (usually red or white) signals data line 1. With the board facing towards you and the floppy connector on the upper end, data line 1 is on the left.



Usually, pin 1 must face left when looking at the drive from above with the drive pointing away from you. Still, double check!

Depending on the package, Kryoflux comes with or without cables and a PSU. We recommend powering the drive *directly* with the external PSU. It is possible to route the power through the board, especially, if you happen to have two drives and only one PSU. You are doing this at your own risk. Please keep in mind that a malfunctioning PSU could destroy your board because of voltage spikes. Connect the drive to the PSU with the Molex plug (if you have a 3.5" drive, you need to attach the Berg adapter to the Molex plug), or the PSU to the board and the board to the drive with a Berg to Berg cable or Berg to Molex cable. *Again, we strongly recommend directly powering the drive.*

For normal operation, always connect the board to the computer first, then plug in an external power supply. Otherwise you might lock up the board. Simply unplug USB and power, and restart with USB.

*Do not power the PSU yet! Do not connect the USB plug to the computer yet!*

## Software Installation

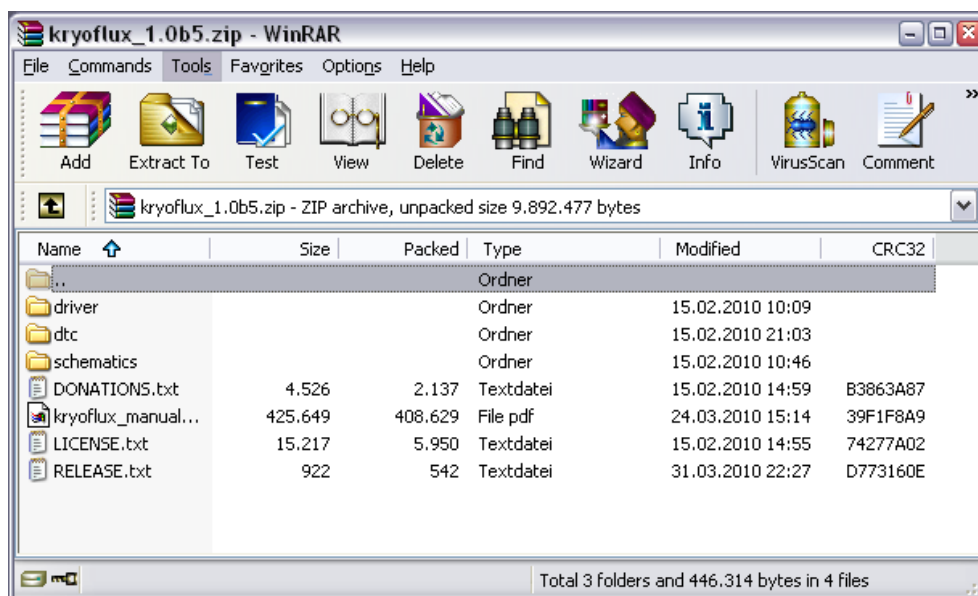
Unpack the software archive available from the KryoFlux web site (<http://www.kryoflux.com>).

Windows: Copy the appropriate version (32 or 64bits) of DiskTool Console (DTC.exe, firmware.bin, CAPSImg.dll) to a location of your choice. Also take note of the location of the "driver" folder, as it will be needed to complete the following steps.

Mac OS X: Just run the installer (KryoFlux.pkg). This will install DTC as well as libusb. The installer includes a text file that contains a list of files and folders installed should you want to remove them later. Please connect the computer and the KryoFlux board with a USB cable (no USB hub!) and continue reading on page 11 ("All platforms again").

Linux: Copy the DiskTool Console (DTC32 or DTC64, firmware.bin) to a location of your choice. Please install libusb 1.0.8 (available separately, chances are it's already installed as this is a popular component). Please connect the computer and the KryoFlux board with a USB cable (no USB hub!) and continue reading on page 11 ("All platforms again").

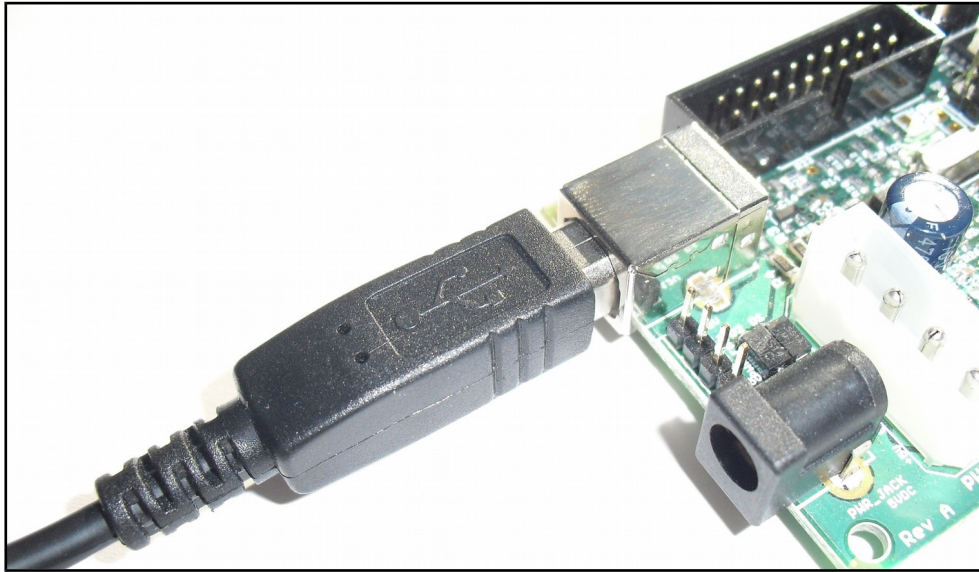
Amiga OS 4: Copy the DiskTool Console (DTC, firmware.bin) to a location of your choice. Copy capsimage.device to "DEVS:". Please connect the computer and the KryoFlux board with a USB cable (no USB hub!) and continue reading on page 11 ("All platforms again"). Note that there is no Java VM for the Amiga (yet), hence you can not install the GUI.



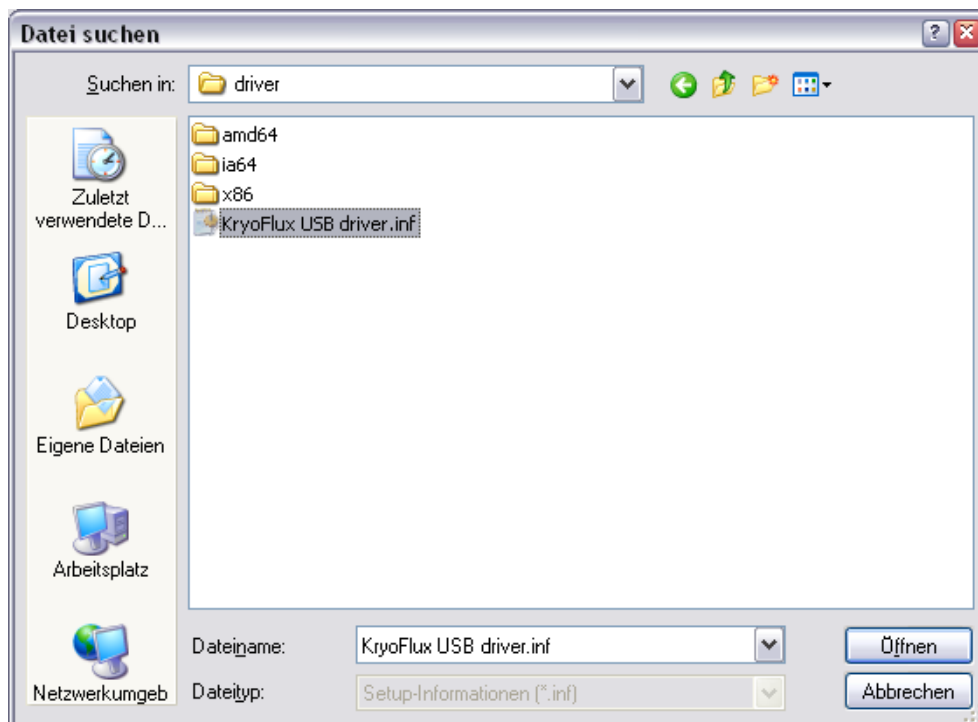
*Note for Windows users only:* To prevent Windows from searching for a driver for minutes or picking a wrong driver on its own, please disconnect from the Internet for now. The drivers used by KryoFlux are Microsoft's own "winusb".



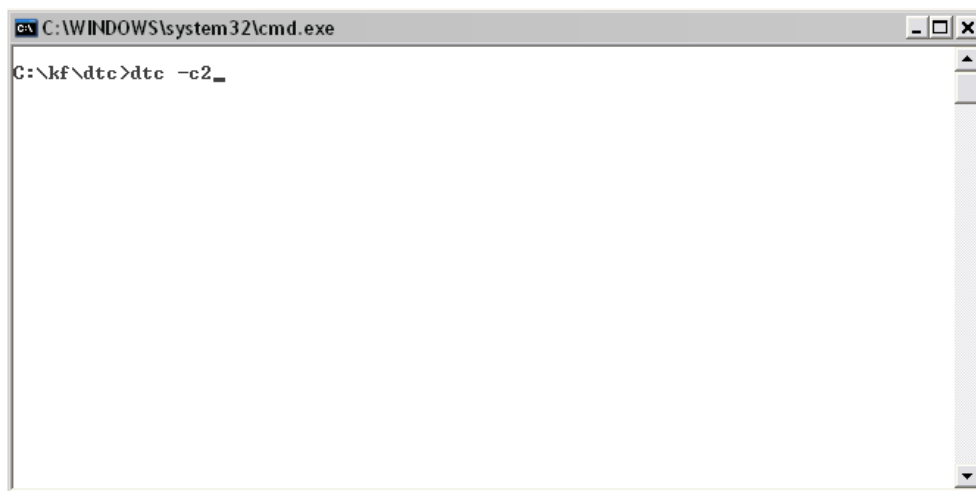
Attach the USB cable to KryoFlux and then attach it to your computer. *Do not use a USB hub!*



*The following steps are for Windows only:* Windows will ask for a driver, for fastest installation select that you want to pick the driver yourself (usually the last option in a series of requesters, depending on version of Windows used). Select "KryoFlux USB driver.inf". Wait for installation to finish until you continue.



Open a command line (Start Menu, "Run") and change to the folder where DTC resides. Enter "DTC -c2".



```
C:\WINDOWS\system32\cmd.exe
C:\kf\dte>dte -c2_
```


The device will re-enumerate, so Windows (versions 8 and before) has to install another instance of the driver. Please follow the same procedure mentioned above. DTC will report an error, which is expected due to the driver being installed.



```
C:\WINDOWS\system32\cmd.exe
C:\kf\dte>dte -c2
Timeout while waiting for device initialization
C:\kf\dte>
```

Plug the PSU into mains.

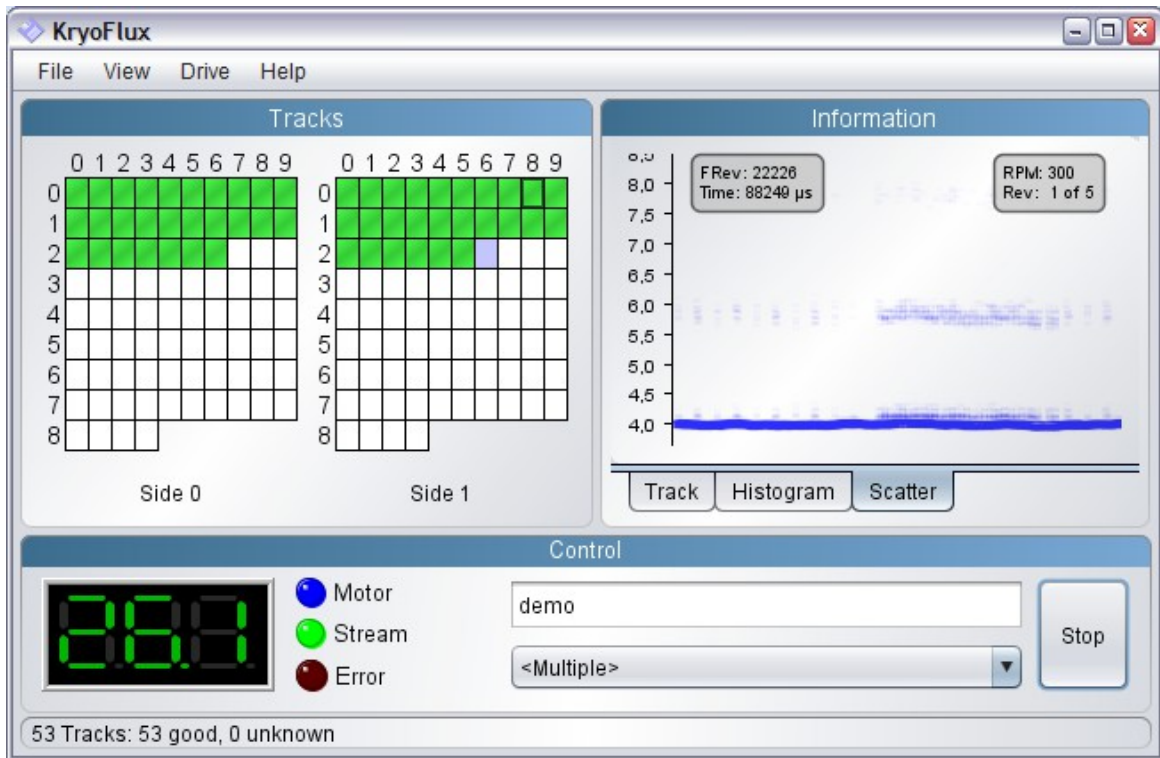
*All platforms again:* Enter "DTC -c2" (again). DTC will now check for the maximum track your drive can access. Depending on the drive type this seeking might fail; this usually does not interfere with standard operation.



```
C:\WINDOWS\system32\cmd.exe
C:\kf\dttc>dttc -c2
Timeout while waiting for device initialization
C:\kf\dttc>dttc -c2
CM: maxtrack=83
C:\kf\dttc>_
```

KryoFlux is set up. Congratulations! Windows users may now reconnect to the Internet.

**Using the GUI:** If you are not familiar with command line programs, we recommend you start with the GUI first to get a feel for how KryoFlux works. Technically speaking the GUI sits on top of DTC, the command line tool, which means whatever you can do with the GUI, can be done with DTC, too. Strictly separating functionality from the interface means all power of KryoFlux is also available via preservation frameworks and KryoFlux can work fully automated, e.g. via batch files or other external control mechanisms. The GUI is a multi-platform application written in Java. You might need to install the Java Virtual Machine on your computer if you haven't used Java applications before. We recommend you also read the chapter about DTC even if you don't plan using it right now, as it contains valuable information. Double click "kryoflux-ui.jar" to start the GUI.



The GUI is separated into three sub-windows. The upper left window contains the track grid. Each block of the grid represents a track on the disk's surface. The upper right window contains the track info block, with two more tabs called "Histogram" and "Scatter". The lower part of the GUI is the control section, where the current track, drive controls and the filename are displayed. Below the filename is the format selector, which itself is dependent on so called profiles. The complete last line of the window is the status line which displays additional information.

The track grid shows the maximum 84 possible track positions available on a disk, which means accessing 40 track disks will only use every second block. As a specialty, some 40 track designs, e.g. the floppy drive used for the Commodore 64, the 1541, actually make use of 80 track mechanisms which can be used by copy protection schemes. So don't be surprised if dumping with a certain format switches to 80 track mode. When you start dumping the complete grid gets filled with white. During dumping, blocks change their color according to the result of the process.

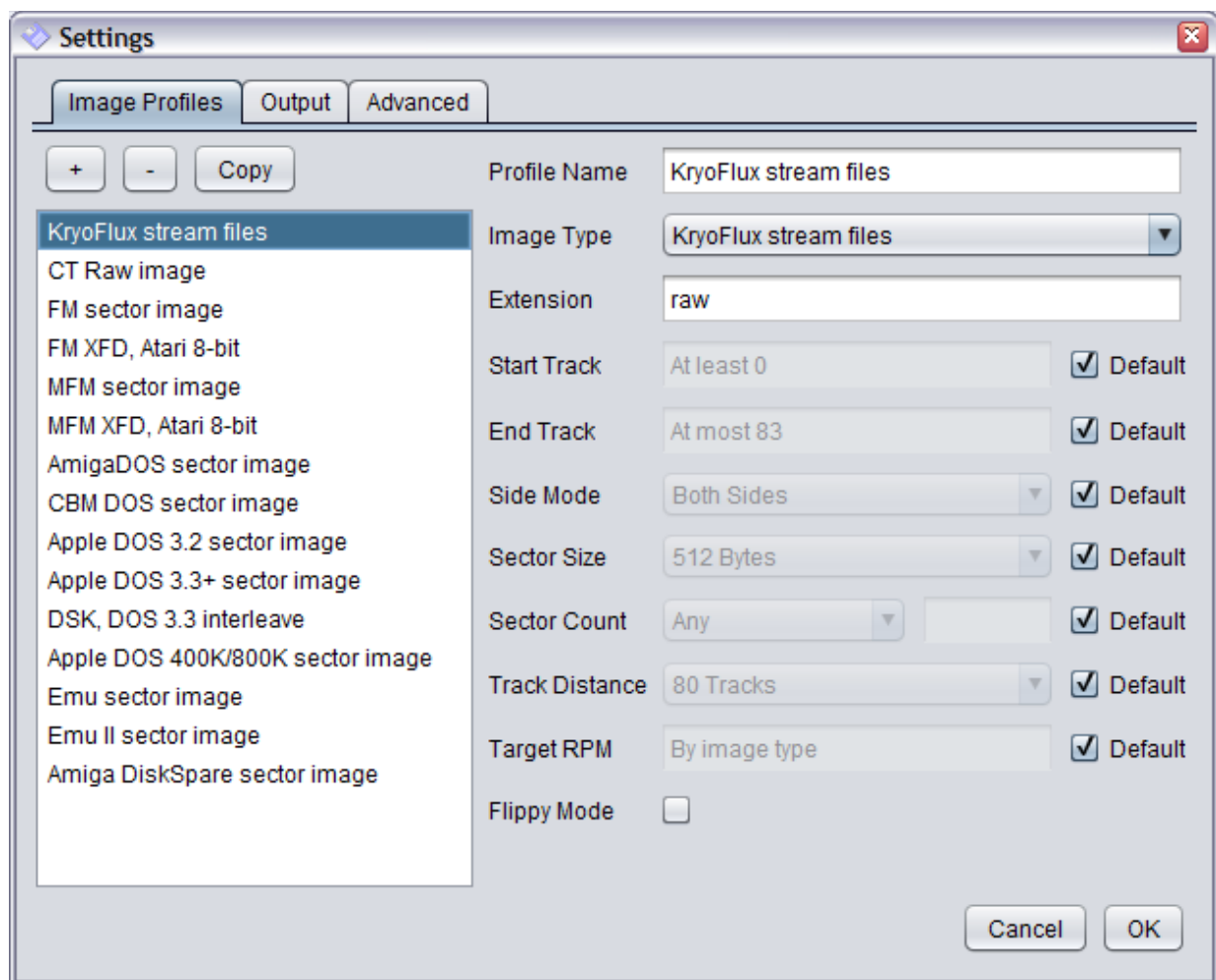
- green - track decoded, no errors found
- grey - noise (or unknown encoding scheme)
- red - track decoded, error(s) found, reading will be retried
- yellow - notifications and warnings, e.g. additional header data found
- glowing - track is being dumped

To get more information about the result of a certain track, move your mouse pointer over it. This will output the result of the operation in the status line.

The histogram and scatter views are only available if dumping stream files. While displaying the scatter data (starting at the index), pressing the function keys **F1** to **F5** will display the corresponding revolution (if present). Pressing "a" will automatically animate consecutive revolutions in the scatter view. "r" increases the RPM by 5 at which the track graph is being interpreted, "shift-r" decreases the RPM by 5. Pressing "i" will toggle the small info field placed in the scatter. Please note that real time decoding of data dumped needs resources which might make dumping troublesome on slow computers. If this turns out to be the case, just switch to the "Track" display.

The menu bar contains the menus "File", "View", "Drive" and "Help".

The file menu contains the settings. Among these settings are the so called profiles. A profile defines how a disk should be dumped. A profile is nothing else than a combination of command line parameters which are automatically set by the GUI according to the profile characteristics.

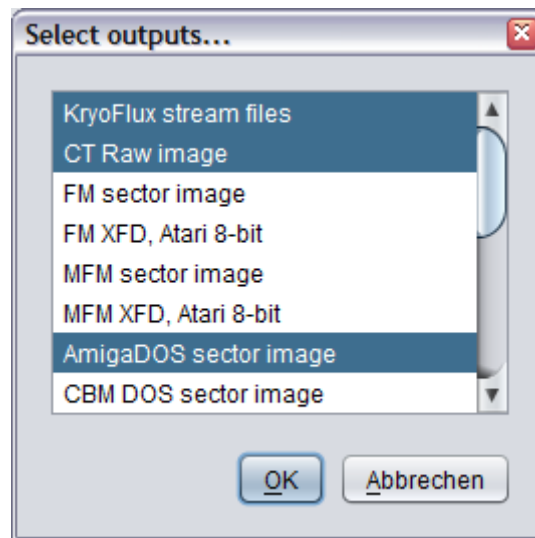


The good news is that Profiles can be cloned and edited which means that you can prepare specific settings for whatever task you have in mind. Please also note that several profiles can be used at the same time while dumping, meaning that a combination of stream files and e.g. AmigaDOS will create a perfect dump environment where the guide format (AmigaDOS) will make DTC retry if an error is found during decoding, delivering perfect stream files, even if these only contain raw data. This is actually the best of both worlds – verified raw and decoded image data.

To use multiple output formats at the same time, select "<multiple>" as shown in picture of the main window above. This will open a pop up window which will then let you select

the profiles required.

To create stream files and apply preservation parameters for an Amiga disk to be dumped, select "KryoFlux stream files", then add "AmigaDOS". "CT Raw image" can be omitted starting with release 2.20. It is only needed if you intend to export data for inspection with the Softpres Analyser (CTA; available separately) afterwards.



Before you start creating your first dump, please switch to the "Output" tab and select the destination directory. The GUI will create sub-folders for stream files, all other files will be named as per your filename selection with the corresponding extension added.

The "View" menu offers a separate scatter window which will be floating atop. This is handy if you have a large desktop and want to display track information or histogram data and keep an eye on the bands used for encoding.

The "Drive" menu will let you select the drive to be used for dumping. KryoFlux supports two drives, as used in most PCs back in the 1980s and 1990s. A drive needs to be calibrated before it can be used, which will be used to determine the maximum track accessible by the drive. There is no need to put a disk into the drive while calibrating.

→ In the unlikely event that calibration fails and you are using the fast firmware, you might want to try using the slower (standard) firmware which was specifically tuned for 8", 5.25" and 3" drives. Close the GUI, copy the file "firmware.bin" from the installation ZIP to the GUI directory, replacing the faster firmware file. Don't forget to reset your KryoFlux board and restart the GUI. It will now use the slow firmware.

If you have changed a drive or the calibration failed, you will need to recalibrate the drive, which is why it's listed as a separate option in this menu.

One key feature of KryoFlux is that decoding can be "replayed". You can therefore select "stream files" as a floppy drive in the menu. This will give you the option to create other images from a STREAM dump made earlier.

*Note:* The GUI currently does not support writing to disk, please use the command line tool DTC to write images back to disk.

**Using DTC:** DTC is a command line application with an optional graphical user interface (GUI) that runs on the Java Virtual Machine. The GUI is located in the DTC folder and can be used after necessary drivers have been installed. The GUI will take care of most tasks, but currently is not as versatile as the command line version.

DTC is the "DiskTool Console" and therefore controls all functions of the package. One special feature of DTC is to output several images at once. That means you can e.g. dump an Amiga game disk to stream files (raw files) while at the same time writing an .ADF of the sector data to see if the disk has a standard file system. You don't have to redump the same disk if you find the disk has some kind of protection which can not be represented in a standard sector dump file (e.g. ADF).

You'll find it even more convenient to know that DTC can generate all further disk images (image type 2 or higher) without a disk present. All you ever need to keep are STREAM or DRAFT files. This option is called "deviceless" mode and means it even works without the KryoFlux hardware present. Think of this as a converter mode, where DTC will operate on a virtual disk, based on a stream dump made earlier.

The only difference between KryoFlux stream files (image file type 0) and KryoFlux DRAFT files (image file type 1 – to be implemented later) is that the latter is a convenient one-file only device-independent image that is handy for transportation, while the former is a group of files representing uncompressed raw data, one for each track and side. This can be more comfortable for development of converters and similar. Please note that generation and usage of stream files should be preferred.

*DTC offers the following command line options:*

```
-f<name>: set filename
-i<type>: set image type
-m<id>  : set device mode
          1=image file, 2=KryoFlux (Model: Rosalie) (default 2)
-d<id>  : select drive (default 0)
-dd<val>: set drive density line (default 0)
          0=L, 1=H
-l<mask>: set output level, add values to define mask (default 62)
          1=device, 2=read, 4=cell, 8=format, 16=write, 32=verify
-r<rev>  : set number of revolutions to sample (default by image type)
-t<try>  : set number of retries per track, min 1 (default 5)
-tc<try>: set number of retry cycles per track, min 1 (default 2)
-a<trk>  : set side 0/a track0 physical position (default 0)
-b<trk>  : set side 1/b track0 physical position (default 0)
-s<trk>  : set start track (default at least 0)
-e<trk>  : set end track (default at most 83)
-g<side>: set single sided mode
          0=side 0, 1=side 1, 2=both sides
-z<size>: set sector size
          0=128, 1=256, 2=512, 3=1024 (default 2)
-n<scnt>: set sector count
          0=any, +Z=exactly Z, -Z=at most |Z| (default 0, by image type)
-k<step>: set track distance
          1=80 tracks, 2=40 tracks (default 1)
-ks      : use only selected tracks during analysis (default auto)
-v<rpm>  : set target system's drive speed, RPM (default by image type)
-x<mode>: set extended cell band search (default by image type)
          0=image only, 1=all, 2=reference only
-y       : set flippy disk mode
-oo<ord>: output image track order, add values to define ord (default by
image)
          1=side 0 descending (side 0 ascending if 0)
          2=side 1 descending (side 1 ascending if 0)
          4=side 1 then side 0 (side 0 then side 1 if 0)
          8=side oriented (track oriented if 0)
```

```

-os<trk>: output image start track (default by image)
-oe<trk>: output image end track (default by image)
-ot<pct>: data band threshold (default 30)
-p      : create path
-c<mode>: read calibration mode
        1=track read, 2=maximum track, 3=RPM

-w      : write image to disk
-wi<type>: write: set source image type (default 0)
-wp<par> : write: set platform specific parameter (default 0)
-wm<mode>: write: set mode, add values to define mode (default 0)
-wy      : write: write side 1 to side 0, side 1 becomes unformatted
-y      : write: -wy for floppy disks imaged in a single pass
-g<side> : write: select sides to write (default auto)
        0=side 0, 1=side 1, 2=both sides
-k<step> : write: set track distance preference in source image
        1=80/all tracks, 2=40/even tracks (default 2)
-ks<step>: write: enforce track distance; disables crosstalk filter
-wg<side>: write: enable unformatted side filter (default 3)
        0=disable, 1=side 0, 2=side 1, 3=both sides
-wk<side>: write: enable track crosstalk filter (default 3)
        0=disable, 1=side 0, 2=side 1, 3=both sides
-wv<mode>: write: verify (default 1)
        0=off, 1=verify
-ww<ns>  : write: precompensation window in ns, max 10000 (default auto)
-wt<ns>  : write: precompensation time in ns, max 1000 (default auto)
-wb<bias>: write bias (default by image)
        0=neutral, 1=bias out, 2=bias in
-we<mode>: write: erase mode (default by bias)
        0=normal, 1=used only, 2=wipe

```

#### Image types supported:

```

0 : Kryoflux stream files, preservation
0a: Kryoflux stream files, format guided
2 : CT Raw image, 84 tracks, DS, DD, 300, MFM
3 : FM sector image, 40/80+ tracks, SS/DS, SD/DD, 300, FM
3a: FM XFD, Atari 8-bit
4 : MFM sector image, 40/80+ tracks, SS/DS, DD/HD, 300, MFM
4a: MFM XFD, Atari 8-bit
5 : AmigaDOS sector image, 80+ tracks, DS, DD/HD, 300, MFM
6 : CBM DOS sector image, 35+ tracks, SS, DD, 300, GCR
6a: CBM DOS sector image with error map
7 : Apple DOS 3.2 sector image, 35+ tracks, SS, DD, 300, GCR
8 : Apple DOS 3.3+ sector image, 35+ tracks, SS, DD, 300, GCR
8a: DSK, DOS 3.3 interleave
9 : Apple DOS 400K/800K sector image, 80+ tracks, SS/DS, DD, CLV, GCR
10 : Emu sector image, 35+ tracks, SS, DD, 300, FM
11 : Emu II sector image, 80+ tracks, DS, DD, 300, FM
12 : Amiga DiskSpare sector image, 80+ tracks, DS, DD/HD, 300, MFM
13 : DEC RX01 sector image, 77+ tracks, SS, SD, 360, FM
14 : DEC RX02 sector image, 77+ tracks, SS, SD/DD, 360, FM/DMMFM
15 : CBM MicroProse sector image, 35+ tracks, SS, DD, 300, GCR
16 : CBM RapidLok sector image, 35+ tracks, SS, DD, 300, GCR
17 : CBM Datasoft sector image, 35+ tracks, SS, DD, 300, GCR
18 : CBM Vorpal sector image, 35+ tracks, SS, DD, 300, GCR
19 : CBM V-MAX! sector image, 35+ tracks, SS, DD, 300, GCR
20 : CBM Teque sector image, 35+ tracks, SS, DD, 300, GCR
21 : CBM TDP sector image, 35+ tracks, SS, DD, 300, GCR
22 : CBM GCR image, SS, DD, 300, GCR
22a: CBM GCR image with mastering info, SS, DD, 300, GCR
23 : CBM Big Five sector image, 35+ tracks, SS, DD, 300, GCR
24 : CBM DOS extended sector image, 35+ tracks, SS, DD, 300, GCR
25 : CBM OziSoft sector image, 35+ tracks, SS, DD, 300, GCR

```



### *Image types supported for writing:*

0: auto-detect  
1: IPF image  
2: Amiga ADF sector image  
3: CBM G64 image  
4: KryoFlux stream files

Combining of export formats enables stream file dumping with error detection. Therefore, just add the format you want the raw stream verified against as a second export filter. You can add several export filters ("guide formats") to the same command; e.g. disks using multiple-formats (such as Atari ST and Amiga, CBM and Atari 8-bit and any other combination) can be verified in a single step. If you don't want to keep the images generated by the guide formats and just want to make sure that the raw stream is verified, just omit the file names that would normally be given as guide format parameters. Combined with a large number of retries this sometimes helps rescuing data from worn disks without further recovery work.

Examples:

*Remember to set your drive number with option "-d0" or "-d1"!*

3.5" AmigaDOS formatted, generate stream files for preservation & .ADF file for emulation (e.g. WinUAE):

```
dtc -f<dumpdir/dumpfile> -i0 -f<dumpdir/dumpfile>.adf -i5
```

3.5" AmigaDOS formatted, generate stream files for preservation only, do format checks, 500 retries:

```
dtc -f<dumpdir/dumpfile> -i0 -i5 -t500
```

3.5" AmigaDOS formatted, only generate .ADF file for emulation (fast!):

```
dtc -f<dumpdir/dumpfile>.adf -i5
```

3.5" 720kb DOS formatted, generate stream files & .img file for emulation ONLY (e.g. DOSBox):

```
dtc -f<dumpdir/dumpfile> -i0a -f<dumpdir/dumpfile>.img -i4
```

3.5" 1,44MB DOS formatted, generate stream files & .img file for emulation ONLY:

```
dtc -f<dumpdir/dumpfile> -i0a -f<dumpdir/dumpfile>.img -i4
```

5.25" 700kb DOS formatted, generate stream files for preservation & .img file for emulation (e.g. DOSBox):

```
dtc -f<dumpdir/dumpfile> -i0 -f<dumpdir/dumpfile>.img -i4
```

5.25" 1,2MB DOS formatted, generate stream files for preservation & .img file:

```
dtc -f<dumpdir/dumpfile> -i0 -f<dumpdir/dumpfile>.img -v360 -i4
```

5.25" CBM DOS formatted, generate stream files for preservation & d64 file for emulation (e.g. VICE):

```
dtc -f<dumpdir/dumpfile> -i0 -f<dumpdir/dumpfile>.d64 -i6
```

5.25" CBM DOS formatted, generate stream files for preservation , one-pass flippy mode (modified drive required):

```
dtc -f<dumpdir/dumpfile> -i0 -g2 -y -i6 -b-8 -l8 -t10
```

When writing a sector dump, it is recommended to use option "-l8" to restrict output to decode errors only.

**IMPORTANT NOTE on command line parameters order:** The following settings are "image local" and therefore must appear **before** the image type and would affect only the first image type specified after the parameter. Their values automatically revert to the default after an image type setting (ie once they get used).

**Correct:** DTC.exe -ffilename.ext -v360 -z3 -i4

**Wrong:** DTC.exe -ffilename.ext -i4 -v360 -z3

file name  
flippy disk mode  
image type  
start track  
end track  
output image start track  
output image end track  
output image track order  
single sided mode  
sector size  
sector count  
track distance  
rpm  
data band threshold  
extended cell band search

Other settings are global and can be anywhere in the command line, they'd still affect every operation:

device mode  
output level  
revolutions  
retries  
track 0 positions  
create path \*  
calibration mode

\* create path is special as it is only active from the point it's been defined on the command line, ie you can limit which images create their path if needed.

Special, local and global settings:

start track  
end track

If these appear before an image type they affect the image type, then reset to their defaults. If they appear without a further image type setting (that is, anything defined after the last image type command) they'd affect ALL images.

The tracks defined here are primary restrictions; no matter what, DTC will only operate within the global limits. However, when processing each image type, there is a further check if the image has any further constraints and if yes, those can add further limitations to the track range. Once a global setting excluded a track there is no way of adding that back by an image local definition.

By default images contain all the sides specified by the disk geometry of the image type. If single sided mode is enabled, using an image type that is double sided, but allows single sided image will be forced to contain only the selected side (side 0, side 1 or both). If both sides are selected two images will be created one for each side (naming is automatic with side added) If single sided mode is enabled, using an image type that is single sided, it is possible to select which side will be imaged and disk geometry will be forced to use the selected side(s). Disk geometries for single sided image types only contain side 0,

therefore selecting side 1 will transpose the geometry for side 0 to use side 1 instead. In the case of flippy disks (those that were meant to be read with the disk flipped over for side 1) normally a track0 physical position should be defined as well for side 1 (-8) if the disk is to be read in a single pass without flipping the disk. For dumping disks that may or may not have side 1 formatted, but the target system's drive is capable of reading side 1 without flipping the disk should not have a track 0 position defined. If both sides are selected two images will be created one for each side (naming is automatic with side added).

Flippy disk mode (option -y) reverses the bitstream on the flipside. Note that the position of the index in the bitstream is probably correct only for disks duplicated as "single pass flippy" since those disks used the same index hole for both sides, with modified drives. Disks that were duplicated with earlier drives were actually flipped over, and hence the index is likely to be at a different position. Note that the flipside needs dumping -8 tracks relative to the other side, therefore the drive needs to be able to step to track -8. There is a tutorial video<sup>1</sup> available with more information.

**Wildcards:** Wildcards accepted are "\*" (without quotation marks) and "?" in any combination. "\*" is 0 or more characters with any value, "?" is exactly 1 character with any value. Wildcards are only accepted in the last part of the filename, ie the filename itself, not any part of the path. This is by design, as allowing wildcards for searching through directories would result in lots of ambiguity.

If the filename specified for stream files contains an illegal wildcard pattern, e.g. "-fte? *stdir/test\**" DTC will stop with an error and display the problem path, in this example "*te?stdir/*" - there is a "?" in the directory name. DTC will pick the first stream file that matches your wildcard pattern and the pattern used for naming stream files. It will not choose other type of files, e.g. it is safe to use "-f\* -i0" in a directory with mixed stream files, scans etc. DTC will correctly select whatever stream file is there. You can also use the full name of any file (as e.g. the windows shell would add it when pressing TAB) and the stream will be guessed, e.g. "*test\_23.1.raw*" will be stream "*test\_*" - no need to truncate the name. Of course you use "*test\_*" as before, as well.

DTC now always displays the stream name selected finally, so you can make sure that the correct stream files were being used. DTC also checks whether ANY stream files exists before commencing with read operations from streams, and stops immediately if no stream file exists at all that matches the name requested. It is possible to have just at least 1 track present as input, and you will get the usual errors for the missing files, but if no track stream is present at all with the given name, DTC stops immediately.

**Automatic Image Size:** DTC automatically creates the output image to be in the minimum size required to represent all sector data without losing content. If a side does not contain valid data according to the image type selected and the format allows it, the sector dump image will be automatically single sided. If tracks do not contain valid data beyond the platform specific minimum number of tracks that should be present in an image, additional tracks won't be added to sector dumps. The minimum number of sectors required to represent all tracks on a disk uniformly (for formats that have this requirement) will be automatically selected.

An image that does not contain any valid data at all in the format selected by the image type will generate a 0 sized file - this is by design. If the automatic image sizing is not desirable (for example an application can only work with a certain number of tracks or sectors) it is possible to change the automatic behaviour by changing the various command parameters - remember those are image local settings and must precede the image format parameter. As an example, some older IBM PC disk formats should only contain 77 tracks. You can limit the image generated to contain only 77 tracks by using

---

<sup>1</sup>[https://www.youtube.com/playlist?list=PLecGtGq1QOG\\_g9TFvhmFRME4FsqFiz2ir&feature=view\\_all](https://www.youtube.com/playlist?list=PLecGtGq1QOG_g9TFvhmFRME4FsqFiz2ir&feature=view_all)

the -oe76 parameter (tracks 0...76).

Non-sector dump formats representing low-level disk data (such as stream files and DRAFT) are not affected by automatic image sizing - they always contain all track data dumped.

**Sector dump track ordering:** Generally, two methods are being used to represent track data in most disk image formats.

*Option A (track oriented) :*

track 0, side 0  
track 0, side 1  
track 1, side 0  
track 1, side 1  
[...]

*Option B (side oriented):*

track 0, side 0  
track 1, side 0  
track 2, side 0  
[...]  
track 0, side 1  
track 1, side 1  
track 2, side 1

DTC uses track oriented ordering (Option A) as default for e.g. .ADF and .ST sector image files.

Sectors are always ordered by their physical sector number as stored in the media. The smallest numbered sectors starts at the <track offset>+0 position, the next sector at <track offset>+<sector size> and so on. It does NOT matter whether the system numbers its sectors from 0, 1, 0x41, 0x81, 0xc1 or any other arbitrary value; the lowest sector number found is used as a base for offset 0.

It is possible to define track order using the -oo command. Otherwise the preset will be used.

1=side 0 descending (side 0 ascending if 0)  
2=side 1 descending (side 1 ascending if 0)  
4=side 1 then side 0 (side 0 then side 1 if 0)  
8=side oriented (track oriented if 0)

To set side 0 and side 1 to descending order, just add up both definitions for a total of 3 (1+2=3). To save side 1 before side 0, with track order descending use 7 (1+2+4).

**Dump information:** KryoFlux has a very sophisticated cell detection algorithm. Cell analysis is used to identify bits written to disk.

base: 2.00 us, band: 4.00 us?, 6.00 us?, 8.00 us?

The first value on that line the "base" is the reference clock derived from the type of encoding expected. The following values represent the different bit combinations possible using the encoding scheme (two bands usually used for FM, three bands used for GCR and MFM). This interpretation happens depending on the format specified for a sector dump, so trying to dump to two different formats (e.g. .ADF and .D64) would give two lines of results. A question mark indicates that DTC's detection is an estimation only, but in many cases it's still very accurate.

The "-x" parameter affects which bands will be used for analysis. There are formats that do not fill an entire track, so the rest of the track might contain garbage left over from an earlier formatting. "-x0" will make sure only bands matching the following format decoder ("-i?") will be used. "-x2" restricts the bands detected to only use the theoretical reference value associated with the encoding of the specific track and format being processed. This does not work for format independent settings ("-i0", "-i1").

**Exceptions:** During operation, DTC might encounter exceptions that will trigger warnings or errors. While warnings are for informational purposes only, errors will have a direct effect on the operation.

b: Sector number is not within the allowed range; the sector was NOT included in the image; error.

c: Data checksum could not be verified (might be part of a protection, e.g. calculation based on some seed only accessible by the original loader); warning only.

e: non-standard slip marks (sector end) found; warning only.

h: Header extra data was found. Data is hidden in unused parts of the block header. Sector images can't hold such data; warning only.

i: format type/block ID is non-standard; warning only.

l: sector length is non-standard. If considering it as a protection measure it is possible to decode the sector and saving it in the image; if not, you'll get another flag saying so; warning only.

n: Sector ignored, sector was found but sector image was not created. Reason could be sector having a different size set compared to what the image uses; error.

p: Special protection detected which will malform the sector on purpose, retry will be suppressed; warning only.

s: Side number found is different from what it should be; warning only.

t: Track number found is different from what it should be; warning only.

x: Sector truncated. Sector data is incomplete, decoding stopped. Reason is another sync/mark was found in the data block. Almost certainly protection that a sector image can't deal with; sector is not included in the image; error.

z: Sector offset found is illegal. Sector is still decoded; warning only.

+<n>: Found <n> modified sectors in the track dumped. It means that <n> number of sectors have been user-written since formatting a disk or duplication

**Reading 5.25" floppy disks (e.g. C64):** For the record - we are using the correct term "cylinder" here, which means the physical location of data on the platter. The word "track" is very often used instead of "cylinder", but usually means the lower or the upper side of a cylinder. Cylinder 0 has two sides, 0 and 1. Speaking of tracks these would be track 0 and 1. Cylinder 1 has two sides, 0 and 1, with the tracks 2 and 3. Therefore a disk with 80 cylinders (0-79) would have a total of 160 tracks (0-159).

Flippy disks read fine on a single headed drive by flipping them as usual. Newer drives, which refers to all standard PC drives made after 1985, are dual headed. To avoid mechanical problems (too much pressure applied to the platter from two sides), there is an offset between these two heads. The relative distance is 8 cylinders. This distance is irrelevant when using such a drive with a disk written with or for such a drive. When accessing a cylinder, e.g. cylinder 10, side 0 for this cylinder will appear at head 0, side 1 for this cylinder will appear at head 1.

Now let's try this with a flippy disk. Let's try and read cylinder 10 again. Side 0 will read back correctly and will read data meant for cylinder 10. Side 1 will have a problem. The disk was written in a drive with one head only. Therefore track 10 is on the exact same position on both sides of the platter. Because of the offset (which is -8 for head 1 to be precise), head 1 will read data meant for track 18, not 10.

This problem could be taken care of in software. If you know the offset, data can be shifted to appear at the correct position. But let's try to read cylinder 0. Side 0 will read back correctly again, side 1 will return data meant for cylinder 8.

We would have to step back another 8 cylinders to access data for cylinder 0... but we can't! The drive will stop stepping when reaching cylinder 0.

There is a simple solution to this problem. Drives with one head only had their disks flipped to read or write the second side. Just flip the disk, and side 1 will appear at head 0 at the correct position.

**BUT:** You might notice the disk is not spinning. Chances are high it really does not. It might, but only if the disk has two index holes punched into the jacket. The reason is that modern drives use the index hole to detect drive speed and if the disk is spinning at all. Using a jacket with one index hole only will make this hole appear on the wrong side which the drive can not see. As long as no index is detected, "modern" drives will reject all further commands to read a disk.

To read a flippy disk with a modern drive, there are three major options:

1. Cut the jacket open and place the platter into a jacket that does have two index holes. You can punch in another hole into the original jacket as well. Now tell this to a game collector and make sure you can run fast enough...
2. Add a fake index to your floppy drive. Place a small magnet on the underside of the motor spindle and attach a small hall effect sensor (it will measure the magnet going by) to the drive's electronics. The drawback is that the index you generate is not synced with the original one. So while this does work, it's not useful for preservation.
3. Modify a drive to make it step to track -8. This kind of modification requires an additional TRK00 bypass circuit.

The latter option is recommended for preservation environments.

Alternatively, if you don't plan to use the drive for other purposes than reading flippy disks, you can change the track 0 sensor position - this kind of modification does not require the TRK00 bypass circuit. Users of the first kind of modification (with bypass circuit) should normally specify -b-8, users of the second type (track 0 sensor repositioning) should use -a8 as an additional parameter for imaging disks.

**One-pass floppy mode:** DTC supports dumping of floppy disks in one pass. For this to work your drive must be modded to access cylinder -8 and must have a TRK00 detection bypass circuit installed, or the track 0 sensor repositioned. Detailed information is available via the tutorial video<sup>2</sup> or on our forums.

To dump a C64 floppy disk, use the following command line:

```
dtc -p -b-8 -f<dumpdir/dumpfile> -i0 -y -g2 -i6 -l8 -t10
```

In case you are curious what the parameters do, here's a quick list of the features used.

- p : force directory creation
- fdumpdir/dumpfile: create a directory called <dumpdir> and name all stream files starting with <dumpfile>.
- i0 : create stream files (preservation quality)
- g2 : both sides of the disk
- y : floppy mode on side b
- i6 : CBM DOS format
- b-8 : track offset for side 1 with bypass circuit type modification
- l8 : limited output verbosity
- t10: 10 retries on errors

Please consider choosing the correct guide format decoder as well (e.g. -i15 to -i25) as using the correct format will make it possible to verify the dump in real time against the custom format or protection specifications, and makes it possible for DTC to retry badly read tracks.

It is possible to create sector images from tracks using custom formats. Although these sector images are not useful for emulation, it is possible to examine or compare the content of custom format protected disks using the decoded sector images. Please note, that some custom formats have disk specific information encoded; for example the last 34 bytes of RapidLok sector images are unique for each disk.

**Physical vs. logical track addressing:** DTC has the idea of physical vs logical track numbers. This is necessary to dump formats that have a logical addressing which is different from the physical layout. So called floppy disks, e.g. disks which had side one written on a 1541 (single headed C64 floppy drive) by turning the disk over, have a different physical layout than disks written on dual headed drives (e.g. standard PC HD).

Here is an example log:

```
-8.1[00]: CBM DOS: OK, trk: 001, sec: 21
-6.1[02]: CBM DOS: OK, trk: 002, sec: 21
-4.1[04]: CBM DOS: OK, trk: 003, sec: 21
-2.1[06]: CBM DOS: OK, trk: 004, sec: 21
00.0    : CBM DOS: OK, trk: 001, sec: 21
00.1[08]: CBM DOS: OK, trk: 005, sec: 21
02.0    : CBM DOS: OK, trk: 002, sec: 21
02.1[10]: CBM DOS: OK, trk: 006, sec: 21
04.0    : CBM DOS: OK, trk: 003, sec: 21
04.1[12]: CBM DOS: OK, trk: 007, sec: 21
06.0    : CBM DOS: OK, trk: 004, sec: 21
06.1[14]: CBM DOS: OK, trk: 008, sec: 21
08.0    : CBM DOS: OK, trk: 005, sec: 21
```

[ ] brackets indicate that the real value is different from the theoretical value. The first number(s) before the colon ':' are the physical track numbers being processed as the *drive*

[https://www.youtube.com/playlist?list=PLecGtGq1QOG\\_g9TFvhmFRME4FsqFiz2ir&feature=view\\_all](https://www.youtube.com/playlist?list=PLecGtGq1QOG_g9TFvhmFRME4FsqFiz2ir&feature=view_all)



sees them - the tracks we want to dump at the moment. If the numbers are identical there is no number in bracket.

On most modified drives (drives modified to be able to step to -8), using a -8 offset for side 1 is what we need to get correct data. That is what is set with -b-8 in the command line.

So as long as you are using these settings, you will always see a physical offset for side 1, never for side 0. For example, from the below log:

```
00.0      : CBM DOS: OK, trk: 001, sec: 21
```

Track 0, side 0. It's side 0 no physical offset, so you get only one track number.

```
00.1[08]: CBM DOS: OK, trk: 005, sec: 21
```

Track 0, side 1. It's side 1, we use a -8 physical offset due to how the hardware works. So by the time we dump something that is on track 0 on side 0, it is on track 8 on side 1.

You can see here why:

```
-8.1[00]: CBM DOS: OK, trk: 001, sec: 21
```

We start on track -8 (physically -8, logically we do this because we want track 0):

```
-6.1[02]: CBM DOS: OK, trk: 002, sec: 21
```

```
-6
```

```
-4.1[04]: CBM DOS: OK, trk: 003, sec: 21
```

```
-4
```

```
-2.1[06]: CBM DOS: OK, trk: 004, sec: 21
```

```
-2
```

```
00.1[08]: CBM DOS: OK, trk: 005, sec: 21
```

So by the time we reach logical track 0, physical track on side 1 is 8.

These are for your information only - it can't possibly change (unless changed by a command!), since it works as per the parameters given, ie -b-8

Most disk operating systems store a track number on the track, to verify there was no hardware failure during seeking - it can and does happen. Luckily, CBM DOS has such numbers so DTC can verify where the head is - compared to where it should be. Various DOSes use different numbering systems for tracks, CBM DOS numbers tracks from 1 to 42, most other DOSes would number the same tracks from 0 to 41.

Let's take a look at the same track if read correctly vs. if read incorrectly - we want to see the DOS numbers, so just ignore anything before the ':' :

```
04.0      : CBM DOS: OK, trk: 003, sec: 21
```

You got no \* warnings whatsoever, as the DOS track number found matches the expected number. For this reason, you see no brackets at all; the numbers match.

Now what happens if the head is in an incorrect position:

```
04.0      : CBM DOS: OK*, trk: 003[001], sec: 21, *T
```

What happens here? We *should have* read track CBM DOS track number 3 on this track - indeed in the previous example we see trk: 003. Here we get trk: 003[001]. It means we should have found CBM DOS track 3, but what we found was CBM DOS track number 1.

Normally this means the head got lost. Out of about 1000 dumps we have seen 3 where it means the disk was mastered with a different track offset; the duplicator did not have a drive modified that could go to track -8, so they used a different value, usually -4 in those cases you should use -a4 -b-4, but always make a note if that happens.

If the head is lost:

1. dump any disk, feel free to stop the dumping when track 1 has been reached on side 0, ie output starts with at least 00.0

2. stop dumping with ctrl-c, it's ok now
3. reset the KF board with the reset switch on it; the led pulsing should stop

Next dump attempt should show no DOS track offset - again ignore anything before the colon.

So instead of

```
04.0      : CBM DOS: OK*, trk: 003[001], sec: 21, *T
```

you should see:

```
04.0      : CBM DOS: OK, trk: 003, sec: 21
```

\*T is just helps you spotting this, so you don't have to watch the number, all you have to do is to see if you get any \* warnings or not.

T: is track number mismatch; there are other warnings as well, like modified sector found, data found in gap etc.

The only warning that normally affects dumping quality is \*T. Another one is \*H, which is usual for modified data or protection, but sometimes (very rarely) it happens when the data is very hard to read, so the bitcells get delayed. You will always see \*H on XROM protected disks and always see \*H on modified tracks.

This is only about most disks, disk containing "fat tracks" (no such thing by the way, but that's the popular name for a repeated/index synced track protection) e.g. many EA and Activision titles would show CBM track numbers with offsets. The later Vorpak format also has two tracks with incorrect track numbers mastered by design. That is normal. If you see any DOS track offset at the beginning of dumping, that's not normal.

## Polymorphic export formats (e.g. G64)

Every format that does have several physical subformats is a polymorphic format. Polymorphic means that format selected for export is ambiguous and DTC has to pick the correct physical format on its own. G64 files (CBM GCR, format 22 in DTC, or format 22a for GCR plus mastering data needed for rewriting), representing C64 floppy data, are polymorphic.

DTC enforces the use of stream files for this operation to avoid unnecessarily reading a disk several times. These disks are old. If you want g64 output, chances are it's from an original disk and they tend to be fragile. Because of this is, it's necessary to first do a regular STREAM dump with e.g. CBM DOS (D64, format 6 in DTC) as a guide format, and then re-process the data dumped.

To create your dump data in the first place, this command line will come handy:

```
DTC -f<stream> -i0 -i6
```

The following command line should to convert your dump data to G64 for most scenarios. It will work on the stream files you made in the step before. You will not need the Kryoflux hardware attached to your computer for this process:

```
DTC -m1 -f<stream> -i0 -f<g64name> -y -k2 -i22 -l8
```

DTC is capable of generating output for either or both sides of a disk at the same time as usual. When generating images from both sides separate g64 files named <name>\_s0 and <name>\_s1 will be created.

-y is required for dumps made using floppy drives for side 1.

To dump a specific side (which can save time) only, use -g0 or -g1 instead of -g2

DTC uses all tracks by default; in c64 terms half-tracks as well.  
By using -k2 this can be overridden.

Polymorphic analysis explained in depth: DTC tries to match a stream against various formats and picks the most likely one for each track based on various criteria. Therefore it does not matter whether the disk changes a format for some of its tracks or not.

The format picked may not be the real one (although it usually is) it is the format that would be the best fit for various reasons; essentially DTC tries to get as much of the data from a stream mapped to known and fully understood data as is possible, to minimize the risk of missing anything.

There are several processing passes to achieve this. Data that is not mapped is considered to be gap and is replicated as faithfully as is possible by trying to select the "best" gap area to complement any specific mapped area. DTC chooses a gap area as the track gap (where the track write splice occurs) during processing.

Unfortunately there is no 1:1 mapping between real data read from a disk and a g64 file. G64 is limited to byte sized data (ie exact multiples of 8 bitcells), while in reality this is hardly ever the case with real data. If g64 supported arbitrary number of bitcells per track we could create an exact replica of everything found on a track. Sadly this is impossible most of the time, so usually we have to inject 1...7 bitcells somewhere to make the stream match the byte size constraints of g64. The only place where this can be done without causing side effects is the track gap/write splice area, therefore it is very important to find the correct position for that. The injected bits are "unformatted" by default, so the net effect a game would see is what would happen on a real disk.

One exception to this is RapidLok's key track: the original data is clearly written by outputting data much longer than a track size, ie relying on the fact that data output later will overwrite previously written data. So a RapidLok key track looks like it has no write splice at all - while in reality it is the net result of how it was written. The entire track is filled with sync and the key data is in it somewhere. This can be achieved by outputting tons of syncs when writing then your key data. The key data will overwrite the syncs, but only to the exact extent it needs to be. So essentially the rest of the track will be a continuous sync. Obviously, injecting unformatted data anywhere into the sync area would break the continuity. Injecting data into the key data is not a good idea either. RapidLok does check both. Anything that is not sync must be key data or it fails (can't break its continuity by injecting unformatted bitcells). DTC does the only thing possible to compensate for g64 limitations: it injects syncs into the sync area for a RapidLok key track - the result is an exact replica, but with 0 to 7 sync bits more than the original.

DTC sees G64 as a polymorphic format and tries to understand the track content as much as is possible. After having failed to map track contents to well defined formats, DTC tries to check if the track contains any legible data as a last resort, ie would the track bitcells at least partially conform to any sort of flux encoding. If yes, DTC replicates track content from index to index which may or may not be what should be done. When this happens the format identified will be "CBM GCR DATA" and marked to contain a single sector. It means blind copying of unknown data.

Due to crosstalk (and possibly recycled disks etc) on many disks half-tracks will be read as CBM GCR DATA data. Normally this is unnecessary for most software, which is why -k2 option should be used to ignore half-tracks and reduce image size. You can try fine tuning the image generated by setting the data band threshold as well as using -k2 option vs -ks2 option.

-k2 might find data in the odd (half) tracks on some disks, while -ks2 enforces always

ignoring odd tracks. Normally just use `-k2`, and only use `-ks2` if you are certain that the data found in odd tracks is just junk - crosstalk detected on disks only ever formatted for 40 track usage.

Whenever possible use format 22a, which also stores mastering data (to the extent possible) in the G64 file. This information is needed to write the image back to disk.

Please note that at the time of writing several emulators have issues with "true" G64 files. You will need to use VICE 2.3.20 SPS or later (make sure it has the SPS changes applied). If an image does not load in CCS64 or Hoxs, make sure to try this image in VICE before deeming it broken.

**Dumping Apple DOS (3.3) disks:** These are difficult to read with the Panasonic 5.25 drive we normally use for archival - as they are difficult to read with indeed any drive, unless it's a professionally duplicated disk.

These drives can be switched to change their filtering/AGC method not only for writing, but for reading as well. Once `-dd1` (DD mode instead of HD mode) is added to the parameters the read errors disappear. Some tracks might still show up as unformatted if they are noisy. Adding the `-x0` parameter before the image type makes these tracks to work as well, i.e. a sample command would be to read side 0:

```
dtc -f<streamname> -i0 -f<imagename> -x0 -i8 -l8 -dd1
```

**Writing:** DTC can write images onto a floppy disk as well. Due to the way encoding schemes work, only well known data can be written back to disk reliably. Therefore, DTC can write sector formats and IPF files as listed above, regardless of format or content. This does not apply to data present in raw images (e.g. STREAM, DRAFT, extended ADF and similar). Although DTC will try to write any image type supported, results for everything that contains raw will be mixed. If you rely on something, using a format that can not be verified should be avoided.

It does not matter if tracks are formatted, DTC will unformat tracks as needed.

The basic parameter to enable writing is `-w`. Parameter `-wi` sets the source image type, which defaults to 0 (auto-detect). Parameter `-wp` sets platform specific recording parameters (0 being default). Platform specific parameters are not available for all image types and usually only vary nuances that typically do not affect the overall usability of the disk written.

ADF: set target system video clock; 0=PAL, 1=NTSC (affects bitcell size)

To write an IPF, please use the following command line:

```
dtc -f<imagefile.ipf> -w
```

To write STREAM files, please use the following command line (point to first raw file):

```
dtc -f<pathtostreamfiles/firstfile> -w
```

To write an ADF (like written on an NTSC Amiga), please use the following command line:

```
dtc -f<imagefile.adf> -w -wp1
```

To write a G64, please use the following command line:

```
dtc -f<imagefile.g64> -w
```

*Disks used with DTC for writing should be considered empty.* Although you can set start and end cylinders with `-s` and `-e` parameters, DTC will erase tracks as needed. DTC will automatically adjust pre-compensation and generate necessary duplicator information on the fly. Do not write to a disk with DTC if you don't consider it empty!

**Important:** Write support for IPF files requires the latest IPF decoder library (e.g. Windows "CAPSIImg.dll") which is included in the installation archive. If you have installed an older version of it (on e.g. Windows usually placed in "C:\Windows\System32"), overwrite it with the one supplied. It is recommended to store the library in one location only.

Use genuine DD disks if you are writing DD data. Do not use HD media with the detection hole covered unless you know what you are doing. It is also important that the drive used for writing does support writing of DD data if needed. There are 3.5" HD drives available that are missing DD functionality. In fact many drives sold after 2000 don't write well at all. They were never meant to be used for this as CD ROM and ZIP had already rendered floppies obsolete. The occasional install of a driver was the reason floppy drives were included in PCs for so long.

*Writing disks from STREAM files:* STREAM files are pure, unanalysed raw data. Be warned that writing of raw data can be neither verified, nor can DTC give any feedback if the data read was good in the first place. In other words: garbage in, garbage out. Since data on floppy disks is digital data stored on a basically analogue carrier, creating 1:1 perfect clones for preservation and archival is not a trivial task. Unlike controlled clones (using a preservation container that has been verified for authenticity and integrity as the result of an analysis based on a well-known format – like IPF), KryoFlux will create a clone based on entropy analysis. A major challenge in the replication process is that the data can be read from the disk differently than it was written.

Please note that version 2.5 only features a *technology preview*. It is only suited for 80 track disks (=no C64 support, but you can use G64 to create your clone), and the disks

you want to replicate must be index aligned. This means that user created disks or disks duplicated without index alignment will not work. This will be addressed in later releases.

*Writing C64 disks from G64 files:* The biggest obstacle in writing disks for the C64 is finding a drive that can WRITE double density (DD) data. Most "modern" 5.25" drives can not properly write DD data, and are always locked to HD. If you want to write DD disks in such a drive, you have to switch the drive to DD mode. This can usually be done via a jumper and / or by setting the correct control line of the Shugart interface. Usually pulling line 2 high (logically low) with *-dd1* will switch the drive to DD (not HD) mode – but better be safe than sorry. YOUR KRYOFLUX DRIVE AS WELL AS YOUR 1541 NEED TO BE PERFECTLY ALIGNED. DISKS WRITTEN IN A 1541 NEED TO BE DEGAUSSED BEFORE USAGE.

Degaussing of used disks can be achieved by using a very strong magnet that is moved over the disk jacket in a continuous motion for both sides. This will make sure to erase all leftovers from a 40 track head, which is wider, which is why a 80 track head can not erase all of the original data. A 40 track head would, as a result of this, pick up combined signals, resulting in garbage.

To verify your 5.25" drive does write DD data correctly, download a standard .ADF from the internet and write it back to disk. If the disk writes back fine and does not give any verification errors, all should be fine. If you encounter verification errors, try changing the DD switch to see if this remedies the problem.

You should use good quality disks, with the correct DD parameter and do not change any precompensation values.

Currently, writing of G64 files is single sided only. If you want to write a double sided disk, you need one with two index holes so it can be flipped.

Because the head in a 80 track drive is narrower than in a 40 track drive (or you could say the 40 track head is wider), it is very important that your 1541 is properly aligned. This is extremely important for half-track or fat track protections. If anything with such protection does not work, check alignment of both your 1541 as well as your KryoFlux drive used for writing. You will notice that DTC will erase a disk in case it encounters such issue, which will make it process every track twice. Please note that this might lead to issues when writing with a drive that can access less than 84 tracks, so usage of the end track parameter (*-e<xx>*; e.g. *-e81*) is highly recommended.

G64 writing requires G64 files with mastering information, which can be produced by DTC with image format 22a (*<-i22a>*). These G64 files should work with all emulators. This functionality will later be replaced by IPF files with full mastering information as required. If a G64 is missing mastering information, you will receive a warning. It does not mean that the image won't work, but it's likely that any advanced protection that relies on proper bitcell timing etc. will fail.

It is highly recommended to create clean master images for writing, using *-k2* and *-ot* parameters where applicable. It is important that an image only contains odd ("half") tracks, where they are really required, such as XEMAG protected ("fat track") disks. Writing unused half-tracks will cause readability issues on a real 1541 drive due to the differences in head size (48 TPI), despite the fact that verification using a 80 track, 96 TPI drive will show no errors during the writing process.

If you write back a disk and it does not work, it's very likely a compatibility issue: old vs new CIA chips (they do matter for some games, most notably Melbourne House games and Dragon's Lair), parallel ports/cables, incorrect/modified rom, cartridges, disk write protected/enabled, ram board, 1541 vs 1541 II, sometimes SID used as well.

## About the Software Preservation Society (SPS)

SPS is a privately funded association of art collectors and computer enthusiasts striving for the preservation of computer art, namely computer games.

Art is an important cultural asset. Thousands of museums and archives all over the world preserve and restore pictures, books, movies and audio recordings and information in general for generations to come. To accomplish their assignment, national libraries are backed by law which, varying from country to country, forces production companies to deliver copies of publications, books, audio recordings and movies to the archives for long term preservation. It seems that as of today, nobody has ever thought or actively cared about the true, unmodified and verified preservation of computer games. Without any action taken, time will run out, very quickly.

Unlike games from the 1970s (delivered on solid state ROM-modules) and games from and after the mid-1990s (delivered on optical media like CD-ROMs and DVDs which are supposed to last for decades), computer games from the 1980s and early 1990s were delivered on magnetic media like tapes or floppy disks and are now at the brink of extinction.

From a preservation point of view, tapes and floppy disks are a nightmare for several reasons:

1. Tapes and floppy disks constantly degrade, in two ways. First is the physical degradation of the orientation of the metal particles which form the magnetic field and store the data. This process is slow, and given the fact that the data is encoded digitally, it may be too late to do anything when reading errors occur. Reading errors happen when it has become difficult to decide if a particular bit is 0 or 1. Preservation should occur before it becomes a gamble to get a good read.

2. Second is the chemical degradation. The metal particles bound to the plastic platter of a floppy disk or the surface of a tape can come off the surface. In fact, in most cases the bonding will simply fall apart after years of temperature changes, moisture and other issues of improper storage. Record companies struggle with this problem when remastering old recordings and have developed a process called baking where the original master tape is actually put in an oven to rebind the coating to the transport material. After baking, playback is a one try only process because the media will fall apart after passing the playback head of the machine. While similar to the original is sufficient for analogue material, even a single misinterpreted bit in the digital world means instant failure.

3. While no user can actually press industry standard vinyl recordings, CDs or DVDs at home (recordable media can be spotted by simply looking at it), tapes and floppies can actually be written and modified with consumer-grade equipment. It takes a lot of expertise to distinguish a professionally replicated medium from a home made copy. Even if a disk was produced by a commercial replicator, it does not necessarily mean that disk is still authentic and appropriate for preservation. Apart from a game possibly being copied over the original (as we have seen many times to "fix" a broken disk), many games themselves persist some kind of save state or high score, thus changing or erasing data that was available on the disk in the first place. As soon as the disk has been modified in any way, the authenticity of that copy is put into serious doubt.

SPS has successfully mastered these challenges and developed software and hardware technology to deal with the problems arising during the preservation process. Founded by computer expert and preservation pioneer István Fábrián in 2001 as CAPS (the Classic Amiga Preservation Society), our highly specialized team has more than a decade of field experience. SPS members have not only been involved in playing games on the machines which are regarded retro today, but were programmers and designers also responsible for some of the games and programs available on these platforms.

While our original disk imaging tools (working on e.g. a standard Amiga 1200 with a compact flash adapter) are still good and easy to use, we are currently moving on to a completely self-contained floppy controller "KryoFlux" developed by SPS that works with any modern PC via an USB connection. This does not only speed up imaging of disks, but also enables physical media restoration of any title preserved so far.

Preservation at SPS usually is a two step process. Contributors from all over the world can help imaging disks with our unique technology. At SPS, our experts then use the Softpres Analyser to investigate the disk structure and create an IPF (Interchangeable Preservation Format) file. Scripting allows a flexible, even game-specific, way of representing data when read by a tool, or when rewritten to disk. Often rather different methods are required to represent various disk formats or copy protection methods when intended to be read by e.g. an emulator or to be written back when restoring an original disk. Due to the high quality of the preservation technology, IPFs have become the de facto standard demanded by Amiga users when looking for unmodified images true to the original.

While disks themselves are the problem that needs to be addressed quickly while they are still readable, SPS is also striving for complete archival of manuals and boxes in the form of physical products as well as digital scans. As of today, SPS has digitally archived about 5.000 games produced for the Commodore Amiga, C64, Atari ST and others. This is a race against time to protect gems of yesterday from fading into oblivion.

For more information: [softpres.org](http://softpres.org)

Contact: [softpres.org/contact](http://softpres.org/contact)