High Definition Flux Sampler for USB

KryoFlux Disk Tool Console (DTC) and KryoFlux firmware engineered and written by István Fábián; additional product design and documentation by Christian Bartsch; Linux port by Adam Nielsen; macOS port by Alexander Coers; Amiga OS 4 port by Marcus Comstedt; Disk Tool Console UI by Kieron Wilkinson; hardware layout by Lars Reichel, Stefan Herzog and Olimex Ltd; Logo by Christian Krapp; project support by Tommi Lempinen, Aki Sivula and 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 (https://eab.abime.net) and Amiga & Phoenix Community (https://www.a1k.org) for morale, support and ideas.

Manual Revision 1.31, DTC version 3.50 (2024-05-03)

www.kryoflux.com | forum.kryoflux.com

For commercial use: kryoflux.com/licensing | licensing@kryoflux.com

Disclaimer:

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 that were stored on a 3", 3.5", 5.25" and 8" media. For details see the following pages.

Preface:

KryoFlux consists of two major components: hardware and software. The hardware is a USB Shugart interface which can read or write a corresponding raw stream of flux data (the information as stored on the disk itself). The software, DTC (Disk Tool Console) controls the board and deals with the data. It can ingest, convert and also write (restrictions may apply) data back to disk. For ease of use a GUI to control DTC is also provided.

Introduction:

While today's computers store data on huge hard disks, optical media or even now solid state drives, legacy computers utilised 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 disk 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 than 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 magnetic 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 state changes. This requires the disk to be spinning because without movement, no AC current is induced in the head. The actual data is normally encoded using a scheme like FM, MFM or GCR. This should not be confused with actual file systems, the logical formatting, which sits on top of the encoding scheme. 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 (EDC) happens at a higher logical level than flux reversals – EDC needs interpretation of the signal and knowledge about the scheme used for writing to determine if the readout is correct or not.

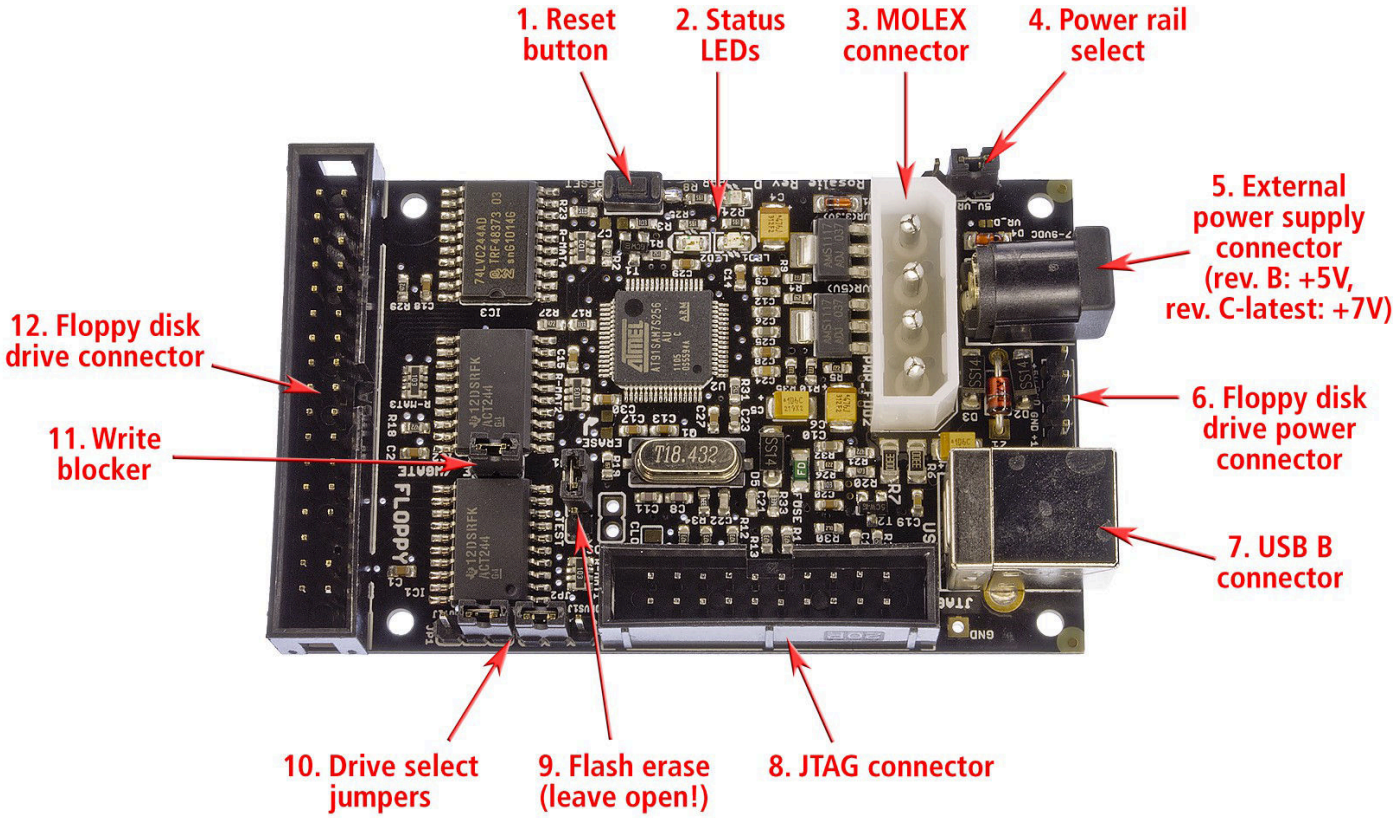
While optical media produced in a pressing plant can last for ages so to speak, magnetic media has a proven life span somewhere between five to 30 years, with the latter only applying to media which was stored under ideal conditions. The higher the capacity of the platter, the higher the risk of the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

System Requirements:

Computer with Windows, macOS, Linux or Amiga OS 4; a native USB 2.0 port; free hard disk space to store tools (~100MB) plus dump data. For best results, KryoFlux must be attached directly to the computer without any USB hubs or cable extensions. Due to the precise timing required and the high bandwidth used, results with hubs can be mixed, with the possibility of complete failure as well. For detailed system requirements please refer to the corresponding download page or included readme file.

You'll 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) drives; although the latter will require an additional adapter from 34p to 50p and TG43 signal. There is a broad range of disk drive 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.

KryoFlux Hardware overview



(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 as 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.**

(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.**

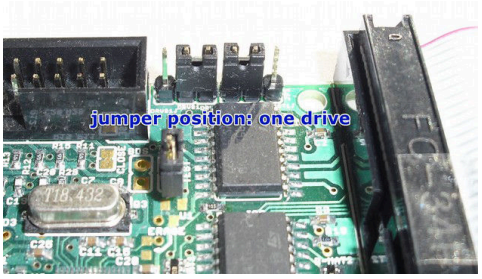
(6) Berg power connector to power a drive – or two with y-cable – via the board: You must use standard Berg connectors (also – incorrectly – referred to as Molex mini) to connect the drive to the board. In the picture above, +12V DC is to the bottom, so the yellow cable of the connector MUST face to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

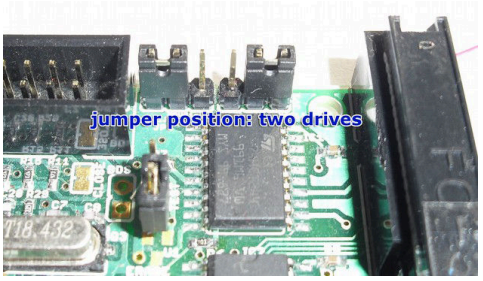
(7) USB B connector: KryoFlux connects to the computer with a USB A to USB B plug. The board (not the floppy drive) is solely powered through USB.

(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. 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.

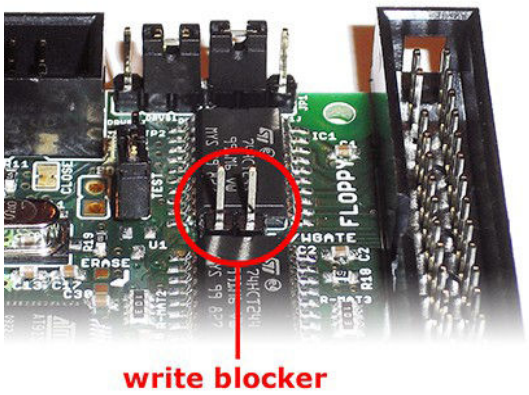
(10) Drive select jumpers: Floppy cables usually have two sections for connecting drives, each of them has two connectors (one for 3.5” drives, the other one for 5.25” drives). You must only connect one drive to one section at a time. The section where the cable is twisted over (at the very end of the 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 coloured 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 not to experiment with polarity. **More than +5V DC or wrong polarity will damage 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 the 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.

Using a so-called "flippy"-modified floppy disk drives

There currently exists two versions of this modification/drive:

Version #1 ("Panasonic") works by modifying the drive so that it can step into the negative domain and access a virtual track -8 on the upper head – which is the location of track 0 of a "flipped disk" written with a single headed drive. Note: During a dump you'll see a message that says "00.0 : Control Command Rejected by the Device" - When you see this, it means the read/write head has moved to the normal side (see below) of the track 0 sensor. This drive can also be used like a standard drive with a standard controller as well.

Version #2 ("Teac", "Newtronics") works by re-aligning track 0 to -8, meaning that it is permanently track-shifted, so data read off standard disks will need to be adjusted in software. Version #2 drive does therefore no longer work with a standard controller. Since the Teac and Newtronics do not bypass the track 0 sensor, neither are prone to getting stuck in the negative domain.

WARNING

Floppy disk drives tend to self-initialize at powerup, e.g. by doing a seek to track 0, which will activate the track 0 sensor ("/TRK00"). When using version #1 disk drives, stopping a dumping process while the drive is below track 0 (reading between -8 and -1) will leave the drive in an undefined position. If the drive is left undefined, the next access (with a seek for track 0) will move the head further back ("outwards") until it reaches a mechanical barrier. This will result in "banging", a loud rattling sound, which might misalign or further damage the drive. If you hear such a sound, remove power immediately. The drive can then be carefully repositioned by turning the motor spindle which moves the head. DO NOT PUSH the head carriage itself!

NOTE: Starting from DTC version 2.72, if the dumping process is interrupted by means of software (ie. ctrl + c), the head carriage will be repositioned back to track 0 regardless of its current position.

Version #1 drive must be the only device on the Shugart bus (=connected to KryoFlux) OR any other drive connected (only one modified drive per bus) must not have contact to pin 33 (usually ground, used here for TRK00 bypass). Also, regardless of what the manual might say, KryoFlux must be powered (=connected to USB) before the drive itself is powered. Otherwise track 0 might become "invisible" for the drive. In both scenarios the drive will bang its head against the mechanical barrier trying to find track 0.

Note: This drive is by definition to be used with the KryoFlux floppy disk controller only. Chances are that, depending on brand and model, the drive will work with other controllers. However, such an operation cannot be guaranteed.

The drive can be connected to the KryoFlux with the supplied (if bought with a KryoFlux) or a standard floppy disk data cable. Please note that the data connector on the PCB has a marking for data line #1 which must be aligned with the red (or otherwise coloured) stripe of the data cable. As outlined above, Version #1 must be the only drive on the bus or any other drive must not connect to data line 33 of the drive data cable, as this signal is used to control the negative track stepping feature.

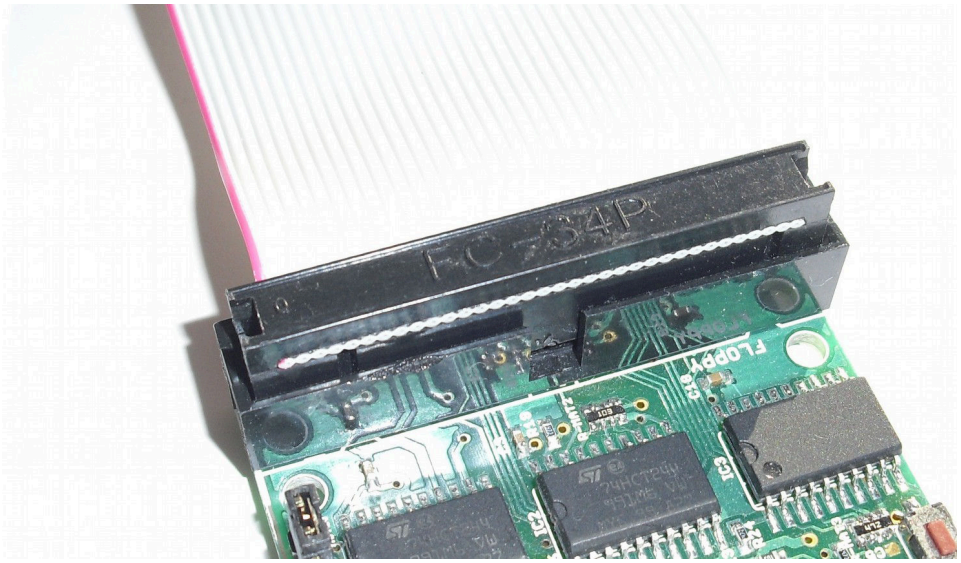
Damage caused by scenarios outlined above is exempt from warranty.

Setting up the hardware

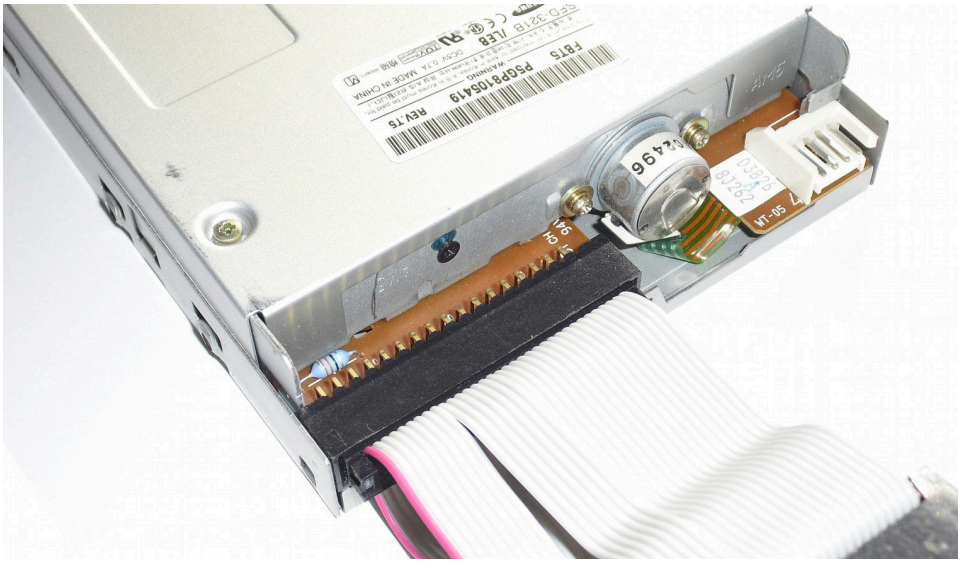
Place the KryoFlux board and the disk drive on a flat, non-conductive surface. Make sure you will not short circuit the device by placing it on a metal table or similar. Connect KryoFlux and drive with a floppy data cable.

Important! Always make the drive – board - connection first, then plug the power (first USB, then PSU). 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 the drive and board.

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 a 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 with a PSU. Powering the drive solely through the board without a PSU might permanently damage your computer's USB port or even mainboard. The reason is that especially older drives will draw a higher current than 500mA (limitation for standard USB ports) when spinning up the drive motor. This can get significantly worse if the motor gets stuck (e.g. resinified grease).**

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

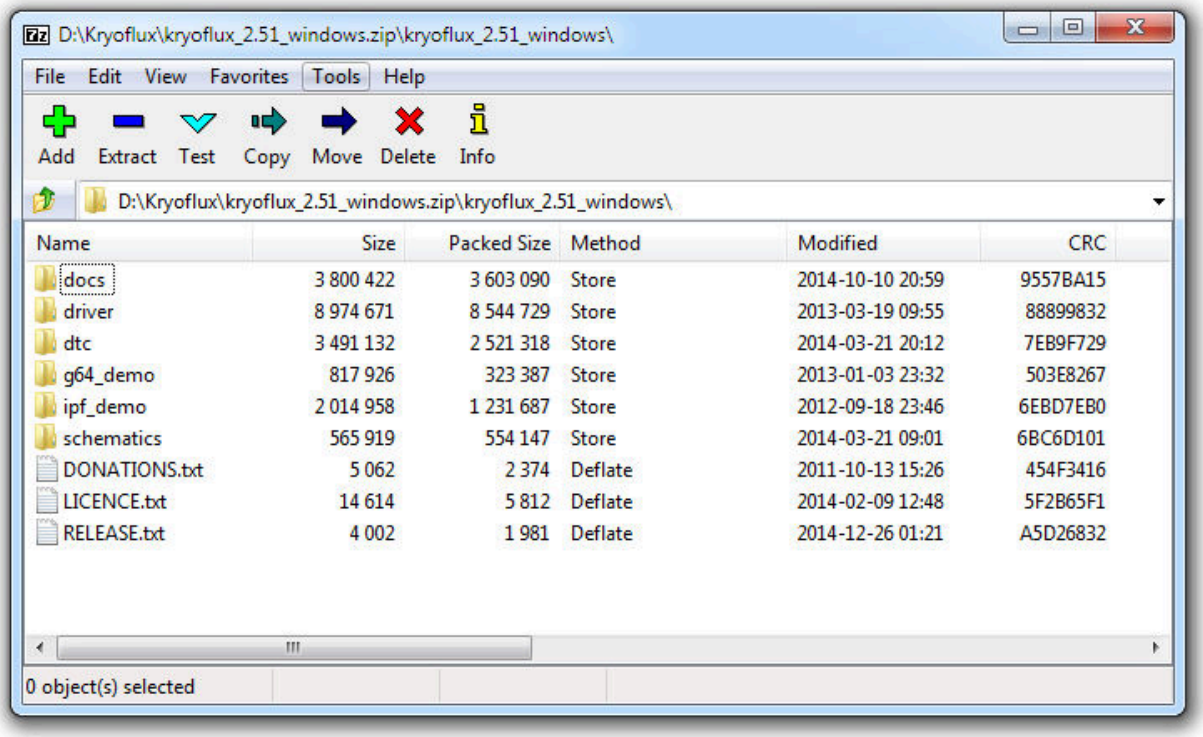
Unpack the software archive available from the KryoFlux web site (https://www.kryoflux.com/).

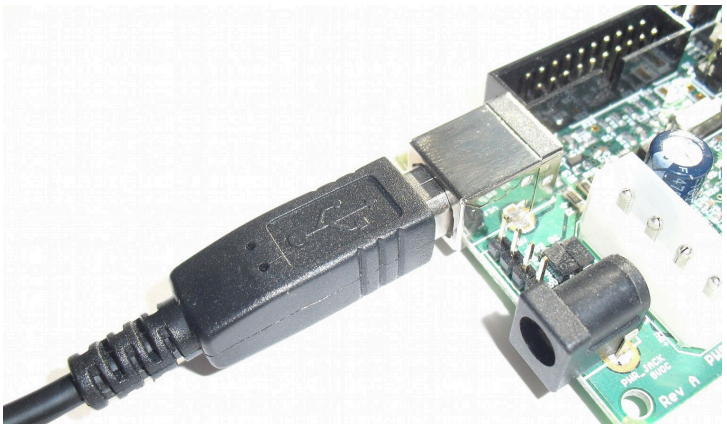
Windows: Copy the appropriate version (32 or 64 bits) of Disk Tool Console (DTC.exe, firmware.bin, CAPSImg.dll) to a location of your choice. Also take note of the location of the "driver" folder, as it will be needed to complete the following steps.

macOS: Just run the installer (KryoFlux.pkg). This will install DTC. Please connect the computer and the KryoFlux board with a USB cable (no USB hub!) and continue reading on page 10 ("All platforms again").

Linux: Copy the Disk Tool 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 10 ("All platforms again").

Amiga OS 4: Copy the Disk Tool 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 10 ("All platforms again"). Note that there is no Java VM for the Amiga (yet), hence you can not install the GUI.

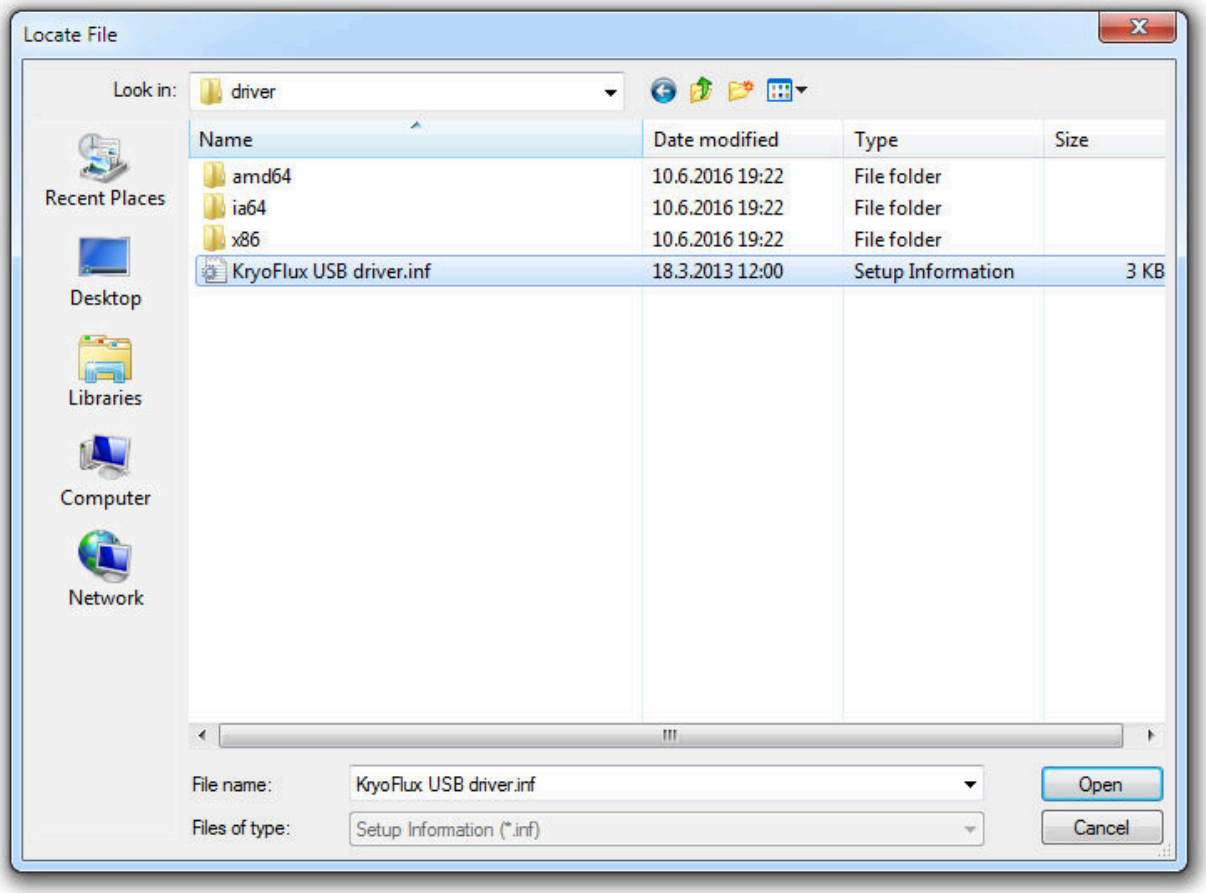




Attach the USB cable to KryoFlux and then attach it to your computer.

The following steps are for Windows only:

Open Device Manager and check under LPT & COM ports that a device called "Bossa" pops up. Older versions of Windows might just show an unknown device. Right click the "Bossa" (or unknown) device and choose to update the driver. Then choose to **pick the driver yourself** from a list. Click **"Have disk"**, then navigate to the drivers folder in the KryoFlux folder and pick the KryoFlux USB driver. Make sure to instruct Windows that you will pick the proper driver; Windows might insist on the Bossa driver being the right choice.



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

The device will re-enumerate and appear as a different device once the firmware has loaded, so Windows has to install another instance of the driver. **Please repeat the same procedure from the top of this page.** DTC will report an error, which is expected due to the driver being installed and can be ignored.

All platforms again:

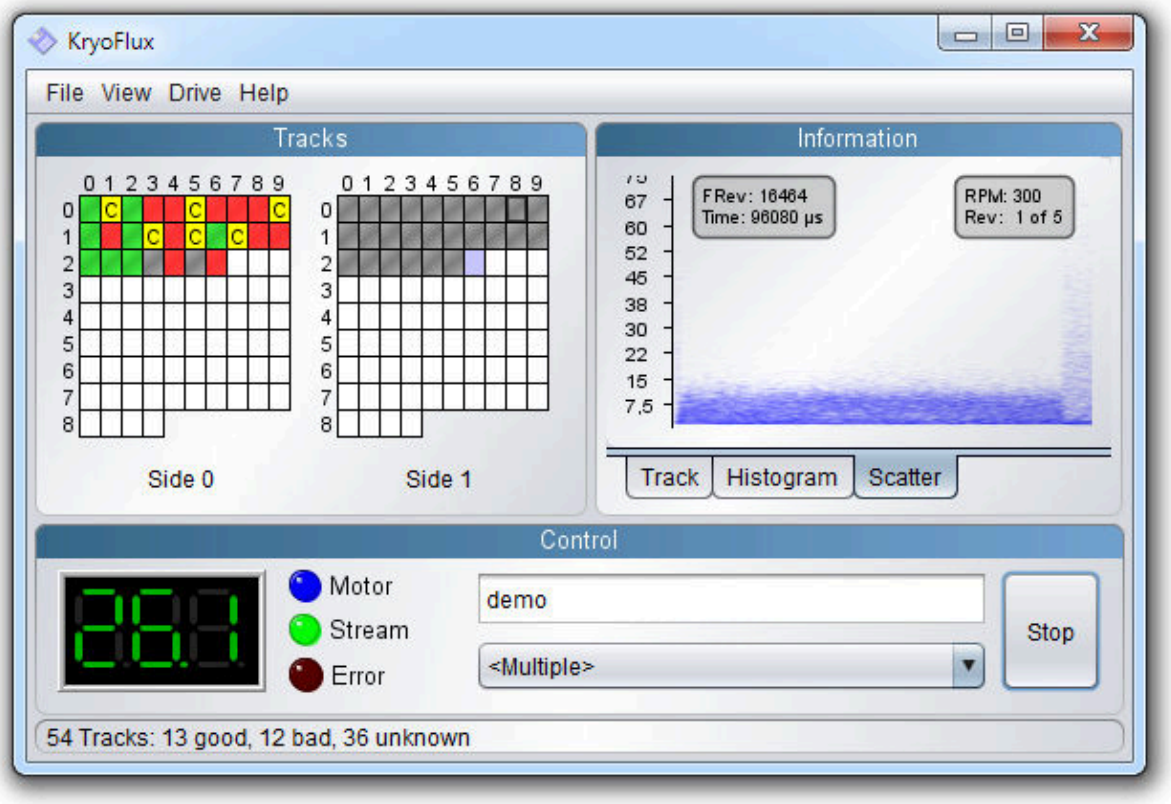
Plug the PSU into mains.

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.

KryoFlux is set up. Congratulations! Versions of Windows 10 and above tend to automatically install the “Bossa” device again. In this case, right click and choose to replace the driver manually as instructed earlier. This might happen especially after Windows updates.

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 as well. Strictly separating functionality from the interface means all power of KryoFlux is also available via preservation frameworks and KryoFlux can work in a fully automated environment, 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 a Java Virtual Machine (we recommend OpenJDK or Eclipse Temurin) 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 colour according to the result of the process.

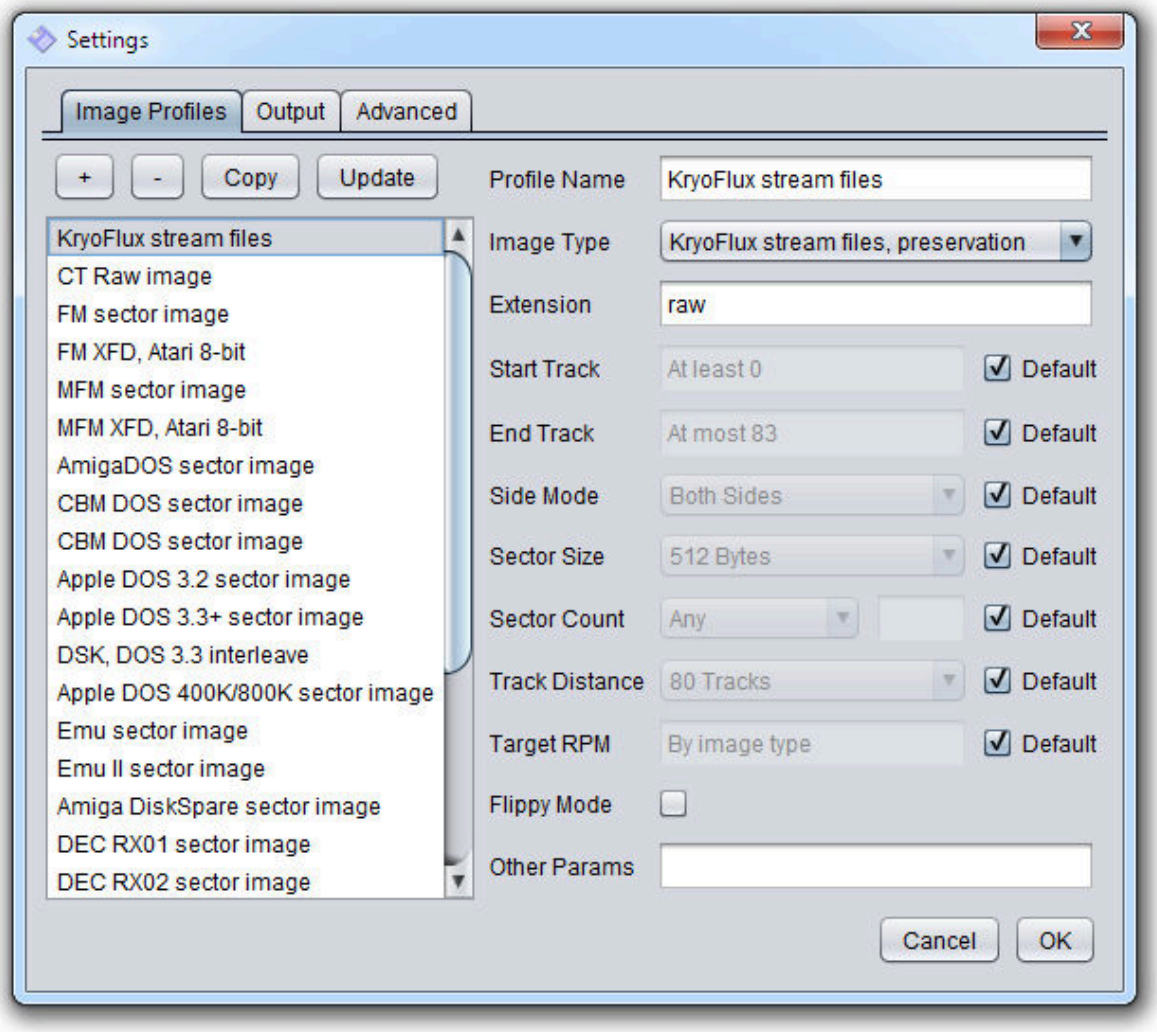
green - track decoded, no errors found
grey - noise (or unknown encoding scheme)
red - track decoded, error(s) found, reading will be retried
yellow - notifications and warnings, e.g. additional header data found
glowing - track is being dumped

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 to 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".

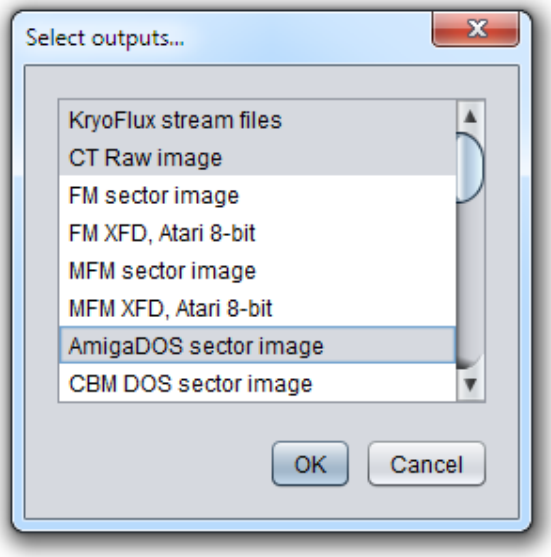
The file menu contains the settings. Among these settings are the so-called profiles. A profile defines how a disk should be dumped. A profile is just a combination of command line parameters which are automatically set by the GUI according to the profile characteristics.



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 the 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. ('Fast' firmware versions are no longer supported starting with the 3.00 release of the 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 flux data 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 does not support writing to disk, please use the command line 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. The following examples will use the Windows version called DTC.exe. On other platforms, there might not be a file suffix. On Linux, DTC is called “kf_dtc” to avoid collision with another Linux package.

DTC is the **“Disk Tool 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). Combining 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.

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’ll ever need to keep are the STREAM 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.

DTC offers the following command line options:

```
Commands:

-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, 64=TI
-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)
-tm<try> : treat missing sectors as bad sectors
           0=off, 1=on (default on)
-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

-ec      : EDOS: add catalogue/database filename for key recovery
-ep      : EDOS: add image file path
-ek      : EDOS: set key filename (default: edos_key.txt)
-er<cnt> : recover EDOS encryption keys (default first key only, -1=all)
-ed      : decrypt EDOS image header(s) to TXT format
-edc     : decrypt EDOS image header(s) to CSV format

-pg<type>: plot: graph type (default 0)
           0=no graphs, 1=sample, 2=consistency, 3=path
-pf<mode>: plot: flip mode (default 0)
           0=off, 1=side 1, 2=both sides
-pw<size>: plot: graph width (default 800)
-ph<size>: plot: graph height (default 600)
-px<fval>: plot: graph x origin (default 0.0)
-py<fval>: plot: graph y origin (default 0.0)
-pd<fval>: plot: graph domain (default entire track by RPM)
-pr<fval>: plot: graph range (default 0.000020)
-pv<rpm> : plot: set target system's drive speed, RPM (default 300)

-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 flippy 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
-wo<mode>: write: save output stream to file (default 0)
           0=off, 1=on
```

Image types supported for reading from a disk:

```
 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 to a disk:

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

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 (i.e. once they get used).

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

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

```
file name                 (-f)
flippy disk mode          (-y)
image type                (-i)
start track               (-s)
end track                 (-e)
output image start track  (-os)
output image end track    (-oe)
output image track order  (-oo)
single sided mode         (-g)
sector size               (-z)
sector count              (-n)
track distance            (-k)
rpm                       (-v)
data band threshold       (-ot)
extended cell band search (-x)
```

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

```
device mode           (-m)
output level          (-l)
Revolutions           (-r)
Retries               (-t)
track 0 positions     (-a/-b)
create path *)        (-p)
calibration mode      (-c)
```

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

Special, **local and global** settings:

```
start track           (-s)
end track             (-e)
```

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 excludes 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 a single side, the 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 available with more information: https://www.youtube.com/playlist?list=PLecGtGq1QOG_g9TFvhmFRME4FsqFiz2ir&feature=view_all

Filename 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, i.e. 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?stidr/**" - 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 can 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 are being used. DTC also checks whether ANY stream files exist 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 sizing:

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 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 sector 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 analysation. 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.

Hard-sectored disks:

Starting with the 3.00 release of the host software and the KryoFlux firmware, imaging hard-sectored disks is supported in addition to normal, soft-sectored disks. (Note, that earlier releases of the software and the firmware cannot image hard-sectored disks.)

Soft-sectored formats written on hard-sectored disks (as it was once popular for systems not having index sensors at all in the drives) can be imaged and decoded, just like as if they were written to a normal disk, e.g. it is possible now to image and

decode C64 or Apple 2 content written to a hard-sectored disk with any number of hard-sectors.

Additionally, real hard-sectored disk content can be imaged now. We'll add support for decoding genuine hard-sectored disk formats in later software updates.

There is no need to tell host software the type of disk to be used, e.g. normal, soft-sectored or hard-sectored with 10 sectors, or 16 sectors, or N sectors etc. The disk type inserted is automatically detected before the execution of a command and no user intervention is required.

Any disk operation is guaranteed to use the user-defined or preset number of disk revolutions, regardless of whether the disk is soft or hard-sectored. This is fully automatic; the user does not have to specify the disk type being used.

The STREAM files now contain the number of hard sectors detected to simplify later processing; in the form of hc=<sector count>. If the sector count is 0, it's a normal, soft-sectored disk; any other value is the number of hard-sectors. If this value is not present, it simply means a soft-sectored disk imaged with a pre-3.00 DTC; it can't be a hard-sectored disk since imaging that was not possible.

Any stream file generated by 3.00+ firmware is marked to support hard-sectored streaming, and has hs=1 in the info area of the stream. Write operations are currently not allowed to hard-sectored disks, since they'd only damage the existing data.

When a hard-sectored disk is detected, hs: <hard_sector_count> is added to the read level output.

Note, that not all media/drive combinations are necessarily compatible when imaging hard-sectored disks. There might be drives that refuse disks with too many index holes, and at least one "modern" NEC FD1165A 8" drive requires modification to the index sensors if a disk with more than 26 hard-sector indexes is to be read.

Reading 5.25" "flippy" 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. 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 track 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 flippy mode (-y):

DTC supports dumping of flippy disks in one pass. For this to work your drive hardware must be modified to access cylinder -8 and must have a TRK00 detection bypass circuit installed (Panasonic), or the track 0 sensor repositioned (Newtronics/Teac).

Detailed information is available via the tutorial video or on our forums:

https://www.youtube.com/playlist?list=PLecGtGq1QOG_g9TFvhmFRME4FsqFiz2ir&feature=view_all

To dump a C64 flippy disk with modified Panasonic drive, use the following command line:

```
dtc -p -b-8 -f<dir/file> -i0 -y -g2 -i6
-l8 -t10 -tm5
```

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

-p	force directory creation
-b-8	physical track offset for side 1 with bypass circuit type modification
-fdir/file	create a directory called <dir> and name all stream files starting with <file>.
-i0	create stream files (preservation quality)
-y	flippy mode on side b
-g2	both sides of the disk
-i6	CBM DOS format
-l8	limited output verbosity
-t10	10 retries on errors
-tm5	5 retries if a track reports a missing sector

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.

Protections:

Certain disks require custom formats for dumping. Typical examples for Commodore 64 are Rapidlok or Vorpal protected disks. In such a case it's best to change the "-i6" in the command line to reflect the specific custom format.

Pete Rittwage's Commodore 64 preservation page database can often give guidance for possible protections:

https://c64preservation.com/

However many games have different revisions and releases and they might contain completely different protections.

Image type setting (-i):

-i or (-i0 in its full form): Read every track with full preservation settings, regardless of whether it's part of the selected guide format(s) or not - unless restricted by other (not the format) parameters.

-i0a: Read only the tracks that are part of the specifications of the guide format(s) selected. This makes it quick to read minimal necessary information from i.e. home made disks as stream files for possible further processing. **NOT TO BE USED FOR PRESERVATION.**

Effect of drive density select (-dd):

Most drives can switch their filtering/AGC method for reading and writing. If there are errors during reading and/or writing, once -dd1 (DD mode instead of HD mode) is added to the parameters, the errors might disappear or become less. Some tracks might still show up as unformatted if they are noisy. Typically -dd1 sets the drive density line to 1, but it actually depends on the model, jumper etc. settings. Hence it's not really guaranteed to switch DD on, it might as well be –dd0. It's easy to test: dump a disk with switch -dd0 and then with -dd1. In whichever -dd mode the stream files become significantly smaller is the DD mode.

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 flippy 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 sees them - the tracks we want to dump at the moment. If the numbers are identical there is no number in the 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 and 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 - 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 look at the same track if read correctly vs. 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 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. We have seen a few cases 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. The reason this was possible to do by a duplicator is that for example the Commodore 1541 disk drive did not have a track 0 sensor - the soft track number found in the sector header was used to correct the head position once a disk was read and in cases of seeking errors the head was "bumped" until it should have reached track 0 (track 1 as used by CBM DOS). The next read 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 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; disks 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 Vorpal 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.

Some command line examples: (Note - All of these command line parameters can also be added to the GUI as presets.)

Remember: In case of getting errors from a 5.25" disk in a seemingly good condition, you may want to experiment with the drive density line setting "-dd0" or "-dd1" added to the command line parameters.

Open a command prompt and navigate to where DTC resides. Now use the following commands depending on which drive you have.

3.5" AmigaDOS formatted, generate stream files for preservation & .ADF file for emulation (e.g. WinUAE):
```
dtc -p -f<dumpdir/dumpfile> -i0 -f<dumpdir/dumpfile>.adf -i5 -l8
```

3.5" AmigaDOS formatted, generate stream files for preservation only, do format checks, 500 retries:
```
dtc -p -f<dumpdir/dumpfile> -i0 -i5 -t500 -l8
```

3.5" AmigaDOS formatted, only generate .ADF file for emulation (fast!):
```
dtc -p -f<dumpdir/dumpfile>.adf -i5 -l8
```

Convert Amiga stream files to .ADF files:
```
dtc -p -m1 -f<dumpdir/dumpfile> -i0 -f<dumpdir/dumpfile>.adf -i5 -l8
```

3.5" 720kb or 1,44MB Atari ST or PC DOS formatted, generate stream files & .img file for emulation:
```
dtc -p -f<dumpdir/dumpfile> -i0 -f<dumpdir/dumpfile>.img -i4 -l8
```

Convert IBM PC stream files to .img files:
```
dtc -p -m1 -f<dumpdir/dumpfile> -i0 -f<dumpdir/dumpfile>.img -i4 -l8
```

3.5" Apple DOS 400k or 800k formatted, generate stream files & .img file for emulation:
```
dtc -p -f<dumpdir/dumpfile> -i0 -f<dumpdir/dumpfile>.img -i9 -l8
```

Standard (non-modified) drive or a flippy-modified Panasonic:

5.25" 700kb DOS formatted, generate stream files for preservation & .img file for emulation (e.g. DOSBox):
```
dtc -p -f<dumpdir/dumpfile> -i0 -f<dumpdir/dumpfile>.img -i4 -l8
```

5.25“ 1,2MB DOS formatted, generate stream files for preservation & .img file:
```
dtc -p -f<dumpdir/dumpfile> -i0 -f<dumpdir/dumpfile>.img -v360 -i4 -l8
```

5.25“ CBM DOS formatted, generate stream files for preservation & d64 file for emulation (e.g. VICE), (non-modified drive):
```
dtc -p -f<dumpdir/dumpfile> -i0 -f<dumpdir/dumpfile>.d64 -i6 -l8
```

Flippy-modified Newtronics or Teac:

5.25“ 700kb DOS formatted, generate stream files for preservation & .img file for emulation (e.g. DOSBox):
```
dtc -p -a8 -b8 -f<dumpdir/dumpfile> -i0 -f<dumpdir/dumpfile>.img -i4 -l8
```

5.25“ 1,2MB DOS formatted, generate stream files for preservation & .img file:
```
dtc -p -a8 -b8 -f<dumpdir/dumpfile> -i0 -f<dumpdir/dumpfile>.img -v360 -i4 -l8
```

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

Panasonic, dual sided dump:
```
dtc -p -b-8 -f<dumpdir/dumpfile> -i0 -y -g2 -i6 -l8
```

Panasonic, single sided dump (g0 = side 0, g1 = side 1):
```
dtc -p -b-8 -f<dumpdir/dumpfile> -i0 -g0 -i6 -l8
```

Panasonic, generate D64 files quickly from a disk without streamfiles:
```
dtc -p -b-8 -f<dumpdir/dumpfile>.d64 -y -g2 -k2 -i6 -l8
```

Panasonic, Commodore 64 (side 0) and Atari 8-bit (side 1):
```
dtc -p -b-8 -f<dumpdir/dumpfile> -i0 -g0 -i6 -y -g1 -i3a -l8
```

Newtronics and Teac, dual sided dump:
```
dtc -p -a8 -f<dumpdir/dumpfile> -i0 -y -g2 -i6 -l8
```

Newtronics and Teac, single sided dump (g0 = side 0, g1 = side 1):
```
dtc -p -a8 -f<dumpdir/dumpfile> -i0 -g0 -i6 -l8
```

Convert C64 stream files to G64 file:
```
dtc -p -m1 -f<dumpdir/dumpfile> -i0 -f<dumpdir/dumpfile>.g64 -y -g2 -k2 -i22a -l8
```

Convert C64 stream files to D64 file:
```
dtc -p -m1 -f<dumpdir/dumpfile> -i0 -f<dumpfile/dumpfile>.d64 -y -g2 -k2 -i6 -l8
```

5.25“ CBM DOS formatted stream files, generate D64 with added error map information for emulation (e.g. VICE):
```
dtc -p -f<dumpdir/dumpfile>.d64 -y -g2 -i6a -l8
```

5.25“ Apple II DOS 3.3+ formatted, professionally duplicated disk:

Newtronics and Teac, generate stream files for preservation & .dsk file for emulation (e.g. OpenEmulator):
```
dtc -p -b-8 -f<dumpdir/dumpfile> -i0 -f<dumpdir/dumpfile>.dsk -y -g2 -i8 -l8
```

Convert Apple II stream files to .dsk files:
```
dtc -p -m1 -f<dumpdir/dumpfile> -i0 -f<dumpdir/dumpfile>.dsk -y -g2 -i8 -l8
```

3” Amstrad CPC, stream files for preservation:

```
dtc -p -f<dumpdir/<dumpfile> -i0 -g0 -e41 -l8
```

8” FM 128 byte sector single sided floppy, stream files for preservation & .img file:

```
dtc -p -f<dumpdir/<dumpfile> -i0 -f<dumpdir/dumpfile>.img -v360 -z0 -g0 -i3 -e76 -l8
```

When creating a sector dump, it is recommended to use option “-l8” to restrict output to decode errors only. If a more detailed analysis report is required, for example to track hardware issues, the option should not be used.

Polymorphic export formats (e.g. G64)

Every format that does have several physical subformats is a polymorphic format. Polymorphic means that the 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 is from an original disk and they tend to be fragile. Because of this, it is 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 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 -g2
-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 flippy 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 minimise 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 (i.e. exact multiples of 8 bitcells), while in reality this is hardly ever the case with real data. If g64 supported an 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, i.e. 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 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, i.e. 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 the -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 work as well, i.e. a sample command would be to read side 0:

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

As a further explanation, there are following two filtering options for analysis you normally don't want to change, however, for completeness, they can be altered:

Parameter -x changes what and how timing (base) bands for flux reversals are considered as possibly to be valid during band analysis.
Band analysis happens before bit cell tracking - but when a blind dump is being made only band analysis is performed, which would still give a clue about the media/encoding type being imaged.

–x0: Restricts band identification to ranges allowed for the encoding of a specific format (e.g. about 2us and harmonics for a format using MFM DD)

–x1: Considers any kind of band frequency as possibly to be valid

–x2: Is like -x0, but the base band found is enforced to be one of the pre-defined, exact fundamental frequencies defined by the format description, instead of the value actually derived from the analysis. e.g. if the format describes a 2us band, but the analysis finds 2.2us instead the result will be changed to 2us.

The fundamental frequency derived is used as a base for tracking the bit cells.

Most of the formats are actually defined to use (1), but some more exotic disk formats, that contain less data than what would fill a whole track, can give confusing analysis results which are technically correct, but would render a track unreadable due to an invalid base being used for bit cell tracking later on.

Imagine, that you first format a track as MFM DD on a PC, which would write the track index-to-index, with a fundamental frequency of 2us.

Now, the same track would be re-formatted in E-mu Emulator synth, which happens to use a 3.2us base frequency.

The band analysis would find that the track has two very common bands; 2us and 3.2us (and their respective harmonics) and would try to find a fundamental frequency, that would describe both (and their harmonics) which is technically correct, but would trip up the bit cell tracking - the conversion of flux reversals to bitcells.

This can never happen on a real FDC and its PLL or similar tracking system for the sole reason that they use method (0), i.e. hardware FDCs are always band restricted - the alien format encoding won't register as legible data usually.

The host software of KryoFlux is a universal "FDC", that by default is not band restricted so it is able to detect any kind of encodings, including any kind of timing used for that encoding. In most cases this behaviour is very desirable, however as per the example above, it can cause problems that are very specific to reading tracks that were re-written with encodings using different base frequencies and are only partially filled with the encoded data.

–ot(0–100): Affects output of (raw) track data when the content is unknown, e.g. for g64 files where the track format was not understood, like a one-off protection type.

If the confidence level in the data band range measured from flux reversals is lower than this value, the output image will not contain the track data, as it is very likely to be not legible or to use any kind of encoding.
This is just a precondition to further validity checks, to speed up the track analysis.

Writing back to a real disk

DTC can write images onto a floppy disk as well, even mixing different physical formats such as writing data originally from a 8" disk to a 3.5" disk. But 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 data 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.

Notice how some of the parameters do change their meaning when used for writing.

For *any* kind of write operation from any kind of image source DTC decides the best parameters to use - except for precompensation settings, since those change during the write process.

There are few caveats when writing STREAM files to a disk:

Any side to be written must have been originally synced to side 0 index; there is no write splice/track gap detection currently implemented. NFA (non flux area) does not work. NFA must be detected and generated with a high frequency. Weak bits may or may not work. Weak bits must be properly detected unless you want to rely on the goodwill of your drive electronics.

Write operations are not allowed to hard-sectored disks, since they'd only damage the existing data. DTC will stop if writing to a hard-sectored disk is attempted.

YOUR KRYOFLUX DRIVE AS WELL AS YOUR TARGET SYSTEMS DISK DRIVE NEED TO BE PERFECTLY ALIGNED. DEGAUSSING IS RECOMMENDED FOR 5.25" DISKS PREVIOUSLY WRITTEN TO.

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 than a 80 track head; this 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.

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 precompensation and generate necessary duplicator information ("watermark", usually on the last unused track > 79) 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 "CAPSImg.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. 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 e.g. ZIP (by Iomega) had already rendered floppies obsolete. The occasional install of a driver was the reason floppy drives were included in PCs for so long.

Using the write parameters:

It is possible to override most of the settings made by DTC – making any user setting prevents the automatic one. Incorrectly setting the flippy mode (reversed or not); DTC knows that only sampled data can be reversed, so -y gets demoted to -wy, see below. Some other parameters may also be demoted or forced to change if it is absolutely necessary.

Delay before writing:

There will be a delay (at least on a fast PC and a 64 bit system) after displaying the filename. This is when loading the entire image happens. After that, basic image information that could be retrieved directly from the image is displayed, such as image type and geometry. After that there is a longer delay, if the image loaded is sampled data, e.g. stream. How long this delay is, very much depends on how fast your PC is, what does the image contain, and whether you use the 64 bit version or the 32 bit one. The 64 bit version can be several times faster on the same image, so if you can run that, make sure to use it. Which version is running is always displayed with dtc command (–h or no command line) Win64 or Win32. You normally want the Win64 one.

During the delay all tracks in the image go through entropy analysis. Based on the results of that analysis, various filters, user settings (or the lack of them!) a quite complex logic, heuristics decides on each write setting to be used to at least have a chance to succeed. DTC will set all of the write parameters (except for precomp) based on the outcome of the various stages. Once the analysis is complete DTC may issue warnings (operation continues) or errors (write does not commence). The final parameter settings displays the symbolic names, their command equivalent and the value set. If the specific parameter has already been set by the user and the user setting has been overridden by DTC the user value is in brackets. E.g. 0[3] means the user set the parameter to 3, but DTC forced it to 0. When this happens it cannot be overridden at all; there is a very good reason for it, such as physical limits found.

The summary of the side statistics also get displayed. Each side is treated in isolation from the other one, so parameters found for one do not apply to the other. Note, that some of the settings cannot be individually changed by the user (intentionally), but can be by DTC.

(td) Track distance found: If this is 1, but you want to make sure it's 2, just use -k2, see below. If DTC finds that td is 2 for any side, that side can be safely written with 80 or 40 track drives without loss of data. If td is 1, there are tracks that can only be written with a 80 track drive. (40 as in 40+, 80 as in 70+, ie 48 tpi, 96/135 tpi, etc)

Note, that it is perfectly legit to have td: 1 for e.g. a C64 disk. Apart from the cases when filtering cannot safely eliminate all offending tracks, this happens as well when you analyse a XEMAG or other so-called "Fat „Track" -protected disk.

Data: The number of tracks containing data. These tracks may be anywhere on the side, 40 tracks is not necessarily the first 40 tracks of the image, but usually is.

Unformatted: Unformatted tracks, as per entropy and filtering. The original number of tracks is the number of tracks as displayed for the image. The number here is the formatted tracks after entropy analysis and filtering. If the filtering (or user by using -k2) did not remove any tracks, you will see a single value, like 35 which is always after entropy analysis. If filtering did remove any track the number of tracks after entropy check is in brackets, e.g. 35[68] means that entropy analysis found 68 tracks with any kind of possibly legit content, and later filters reduced this number to 35.

NFR: Tracks that contain no flux reversal at all (see Data). These tracks get demoted to unformatted under specific circumstances; if the side does not contain any legit data track at all. This makes it possible to completely ignore unformatted sides if it is a viable option.

Write parameters (notice, some of them have different meaning when used for reading):

–wy: Write side 1 to side 0, side 1 becomes unformatted. The unformatted side is by design specifically to prevent creating fake combinations of disk sides. This feature can be used to write any kind of flippy disk content on a single sided drive when the direction of the data does not change. Typically, preprocessed images, like say a Spectrum+3 IPF. Those 3" drives are single sided, and the images are made by manually flipping the disk. Writing is obviously the same process, since there is physically no side 1 head. Write side 0, limit writing to side 0 (-g0), flip the disk in the drive, then add -wy (this automatically uses -g0 unless user selected another value)

–y: Same as -wy, but side 1 is transposed to side 0, i.e. the written data is reversed. This is useful for data that is NOT pre-processed, such as stream files, where the data was sampled in a single pass for flippy disks (CBM, A8, BBC? Etc). This automatically enforces -wy.

Trying to flip a disk in any way when the resulting image would be unformatted (ie side 1 was found to be unformatted during analysis) is not possible.

–g: Enforce writing of a side. DTC selects the sides to be written automatically; if only one side has content, that side, if both then both. While this can be a lot faster, and is what is absolutely necessary to have the same disk as the original for disks where both sides were generally used (ST, Amiga...) you may want to enforce -g2 should the program ever decides to check the side that should be unformatted. Usually they don't though, since many of them ended up on recycled disks.

–k: The preferred image type of the analysis result - you prefer to have a 40/80 track image if at all possible. –k1 allows crosstalk filtering, except for the crosstalk elimination phase as specified at the description of command –k2. –k2 runs the crosstalk detector and if it finds crosstalk on a side at all, it wipes all potential crosstalk tracks, unless the SNR of the data content of the track to be wiped is the same as the neighbouring tracks. In that case the track will remain untouched.

–ks: Enforce the image type. –ks1 will keep all tracks of the image, regardless of analysis saying otherwise. –ks2 will wipe every odd track blindly. Any –ks command disables crosstalk filtering. Entropy analysis and other filters are not affected, so what those find may still affect the image to be written.

You do not and should not use e.g. –k2 and –ks2 at the same time. Use only the one you need.

–wg: Enable side noise filter for any of the sides. The side is actually a bitmask, b0 is side 0, b1 is side1, ie 0: no filter, 1: filter side 0, 2: filter side 1, 3: filter both sides. It is very unlikely that a side contains various short bursts of random small amounts of data by design; this filter eliminates a side if only such data content is found for all tracks.

–wk: Enable crosstalk filter (see -wg for defining the sides affected). Crosstalk happens when 80 track drives are used to sample disks intended for 40 track systems; there is often crosstalk on every odd track (1, 3, 5...). DTC tries to remove these tracks from the image, but the results depend on the drive used for sampling and whether legit data can be found on a track that is potentially just crosstalk. The approach is to not remove anything unless it is 100% certain that a track only contains crosstalk, e.g it won't ruin your Amiga disk image, even though it has no idea of what kind of image you are using at all. The crosstalk filter is automatically activated based on various properties of the entire disk side for each side separately. If it was activated at all (on either side), changing the write bias to any other value than 0 is the clue, e.g. "bias -wb: 1" displayed as a derived parameter. When write bias changes, so does write erasure as well; using a bias only makes sense if the entire side is wiped first. DTC will just use normal writing if no crosstalk was detected at all.

There is a trick to see the result of the image analysis without writing or having a KF board attached:

```
dtc -w -m1 -fimage/* -i -fimage/<your
real image> <your parameters>
```

This would try to write into the stream image in the first parameter, which will only fail once the actual writing is starting. "*" should be any stream file, usually * is enough, but if you try what happens with ADF, G64, IPF make sure there is a stream file accessible as well, otherwise DTC finds out that the device is invalid for writing before the analysis. This is a diagnostic feature.

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. 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 disk drive 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 an 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.

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 how it was written.

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.

G64 writing prefers 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 Commodore C64 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.

Writing with 5.25" 40 track or 8" 77 track drive:

Since KryoFlux was created to be used with standard 5.25" and 3.5" 80 track drives, using other types of drives to write a disk most likely does not work out-of-the-box without at least some user involvement in the process.

EDOS (Electronic Distribution Of Software) support:

EDOS is an acronym for **Electronic Distribution Of Software**.

EDOS used to be a hardware and software-based system where commercial software was being duplicated in the store on demand. It was especially popular in the UK, but it seems EDOS machines were in circulation at other markets as well.

EDOS used original mastering data (or data generated from masters/commercially duplicated media) so it is an important aspect of software preservation.

So far we only have 2 early CD images, but there were new collections available once every few months (quarterly?) during the lifetime of the EDOS system, so if you happen to have more CDs (or CD images), HDDs or any of the update files available, please do get in touch.

EDOS distribution CDs and updates contain image files and catalogue database files referring to each disk or tape available for duplication in the specific update release.

EDOS image files have all of their metadata encrypted and stored in the file headers, however the stream data itself does not seem to be encrypted after the header.

DTC is capable of recovering any key used for encrypting the metadata part of EDOS files.

Due to the limited material currently available it is not yet known whether the encryption key was ever changed - DTC however is capable of finding it automatically as long as a large amount of EDOS files (e.g. full CD) and any catalogue file referencing them are both available. The catalogue file contents and the image files don't have to match perfectly - DTC will match them as long as the possible sources are listed as parameters for the key recovery process.

The catalogue files are located at the root of the distribution folders, compressed using PKZip format. They may be password protected, but a simple dictionary based zip recovery tool can easily give the password if needed. The files are named like their content with a Z added to the extension, e.g. 00070001.00Z is the PKZip archive that contains the 00070001.00 catalogue file.

DTC does not decompress the archive; it must be decompressed first to make it usable for DTC.

Please do not change the original directory structure of the distribution data, as DTC expects the original layout.

DTC can decrypt the EDOS image file headers using the previously recovered key and save them as TXT or CSV files.

The following command uses the (previously decompressed) catalogue file `00070001.00` and any EDOS file that can be found in the `d:/edos/cd1` folder to recover the key:

```
dtc -f"d:/edos/00070001.00" -ec
-f"d:/edos/cd1" -ep -er
```

In the command the catalogue file(s) have to be specified, alongside the folder(s) where EDOS files may reside, and finally `-er` activates the key solver.

By default, the result(s) are saved into the file called `edos_key.txt`, but this can be changed using the `-ek` option, following the parameter as usual, e.g. `-fmy_key.txt -ek` will save the results to the `my_key.txt` file.

Due to the way the encryption works, there are multiple valid solutions possible - at least this is true for all the images currently available.
The solver can save the first valid solution, or all of them if `-er-1` is specified.

Depending on the speed of your computer and the amount of files used, the recovery process can take just a few seconds or several minutes. As usual, we recommend using the 64 bit version of DTC for this function.

You can define any number of catalogue files and image locations in a single command.
In the following example we use 2 catalogue files and 2 image folders:

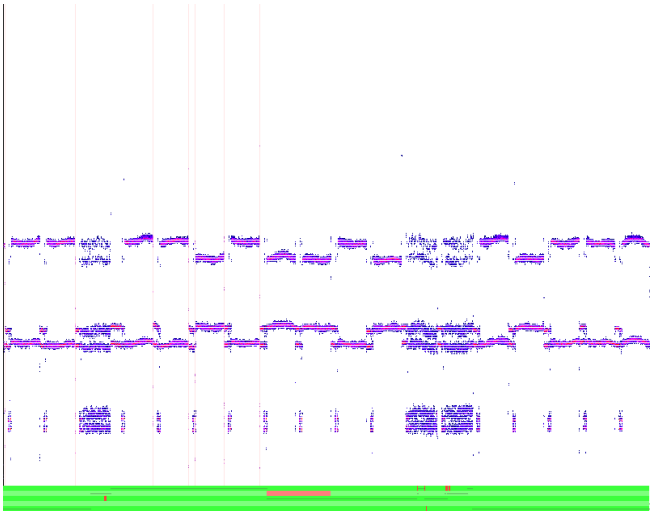
```
dtc -f"d:/edos/00070001.00" -ec
-f"d:/edos/00080001.00" -ec
-f"d:/edos/cd1" -ep -f"d:/edos/cd2" -ep
-er
```

Once you have the encryption key recovered, you can collate all the metadata from your available EDOS files to either simple text files (`-ed`) or CSV files (`-edc`). In both cases, the file will contain the name of the image file as the first field for each new data entry, followed by all the decrypted metadata fields in the same order as they are present in the EDOS image files.

This example will collate all the data into a CSV file from the folders used in the previous example.

```
dtc -f"d:/edos/cd1" -ep -f"d:/edos/cd2"
-ep -f"edos.csv" -edc
```

Graphs may be optionally generated whenever DTC reads a track from either stream files or a KryoFlux board:



The default is to not create any graphs. Tracks that are not accessed during the read command do not generate graphs, but for example reading an entire disk image uses all the tracks.

Graphs do not rely on guide formats or any other information, other than the sample data taken from a specific track. A new graph is rendered each time a track has been read in track read calibration mode - this is by design.

Graphs will be rendered per track as .bmp image files to the same path along with stream files.

-pg: Graph type

-pg1: Heat map is generated from the flux reversal samples
-pg2: Heat map is generated with consistency bars
-pg3: Heat map is generated with consistency and Path Solver

Note: consistency graphs require significantly more processing, than the sample heat map alone, hence generating these graphs takes more time.

The consistency bars represent how well aligned are the samples on each revolution sampled from the same track. Each horizontal bar at the bottom of the graph represents one revolution sampled from the same track. There is a limit on the number of revolutions added to graphs, so a track sampled for example a 100 times won't have 100 bars added to the graph.

The green coloured areas in a bar represent areas that match samples taken at a different revolution from the same track area. Red areas are inconsistencies among the various samples taken from the same track area.

A track image that has all green consistency bars has consistent samples for all of its revolutions sampled.

Consistent samples are not necessarily good samples; e.g. a disk defect might produce consistently bad readings for each revolution, however consistent data is unlikely to change due to redumping a track, unless e.g. the disk surface gets cleaned between imaging sessions.

Disk defects, as well as certain copy-protections, always generate other artefacts that are automatically highlighted in the heat map, such as areas with "weak" bits, or without any flux reversals. A weak bit area is expected to produce inconsistent reading on each attempt.

Note that due to various reasons, consistent data does not mean identical sample values.

Path Solver:

With the path solver enabled for the consistency graphs, one or more thin lines visible on the consistency bars represent the parts used from various sampled revolutions of the same track area that can be used to reconstruct a single, good continuous data stream at the flux transition level.

The Path Solver is generating a "best possible" data stream path from all the flux reversal samples present in the stream files, at the flux level.

The Path Solver is using the consistency data generated by the Revolution Analyser.

Paths represent the solution(s) from the path solver telling us whether or how it is possible to construct a single contiguous stream with all consistent flux reversals across multiple samples taken, effectively generating a single track sampled without any defects that are common reading magnetic media, especially old magnetic media.

The Path Solver currently sees weak bits as blocked paths (since they are always inconsistent) and reports them as having no solution. This behaviour is by design, but we have some enhancements planned for more selective reporting of such data, so that it works as intended on e.g. weakbit protected tracks.

Path Maps:

Path maps are built from the data generated by the Path Solver; they represent the solution(s) from the path solver telling us whether or how it is possible to construct a single contiguous stream with all correct flux reversals.

Since path maps are derived from consistency data, they show both consistency bars as well as the path(s) resolved superimposed over the consistency bar.

DTC checks the image consistency for us when a path map is being generated.

If there is no error reported during the generation of a path map, the current track image is as good as it gets; it is possible to create a "best" track without any discontinuity or inconsistency from all the available track samples.

If there is an error it may be due to the track containing weak bits (e.g. fully or partially unformatted) or because there is no way to get a consistent reading from the currently sampled track. In the latter case, the disk should be cleaned and re-imaged. In the case of unformatted or partially unformatted tracks or weakbit based protections, no action is necessary.

Having a full path resolved in a track image does not necessarily mean that the data is good; but it is checked for us that the sampled data is consistent enough. As an example, a physical damage on the disk surface may generate consistently bad flux reversals. Generally speaking, having track data with a path fully resolved can be considered a good quality track sample and using this feature is more convenient than manually checking the consistency bars.

Having poor quality, inconsistent samples read from old magnetic media is very common.

The path effectively represents the quality of the samples, not the quality of the data - the quality of the data content can only be verified once the data is decoded, but having good quality samples for that is essential.

Since the path solver can take some time to run depending on the number of inconsistencies found in the samples, it is only used when a path map graph is requested.

-pf: Graph flip mode

0: Disabled
1: Side 1 is from a flippy disk, read in backwards, so graph will be mirrored horizontally (similar effect to -y for decoding a format)
2: Flip sample data for both sides

-pw: Graph width for rendering in pixels (default is 800)

-ph: Graph height for rendering in pixels (default is 600)

-px: Origin of the graph rendering on the x axis (default is 0.0)

-py: Origin of the graph rendering on the y axis (default is 0.0)

-pd: Graph domain; the graph is rendered for the time period between [px, pd] (default is calculated from the RPM; e.g. 0.2s for 300RPM)

-pr: Graph range; the samples are rendered for each sample in the range [py, pr] (default is 0.00002s, ie 20us)

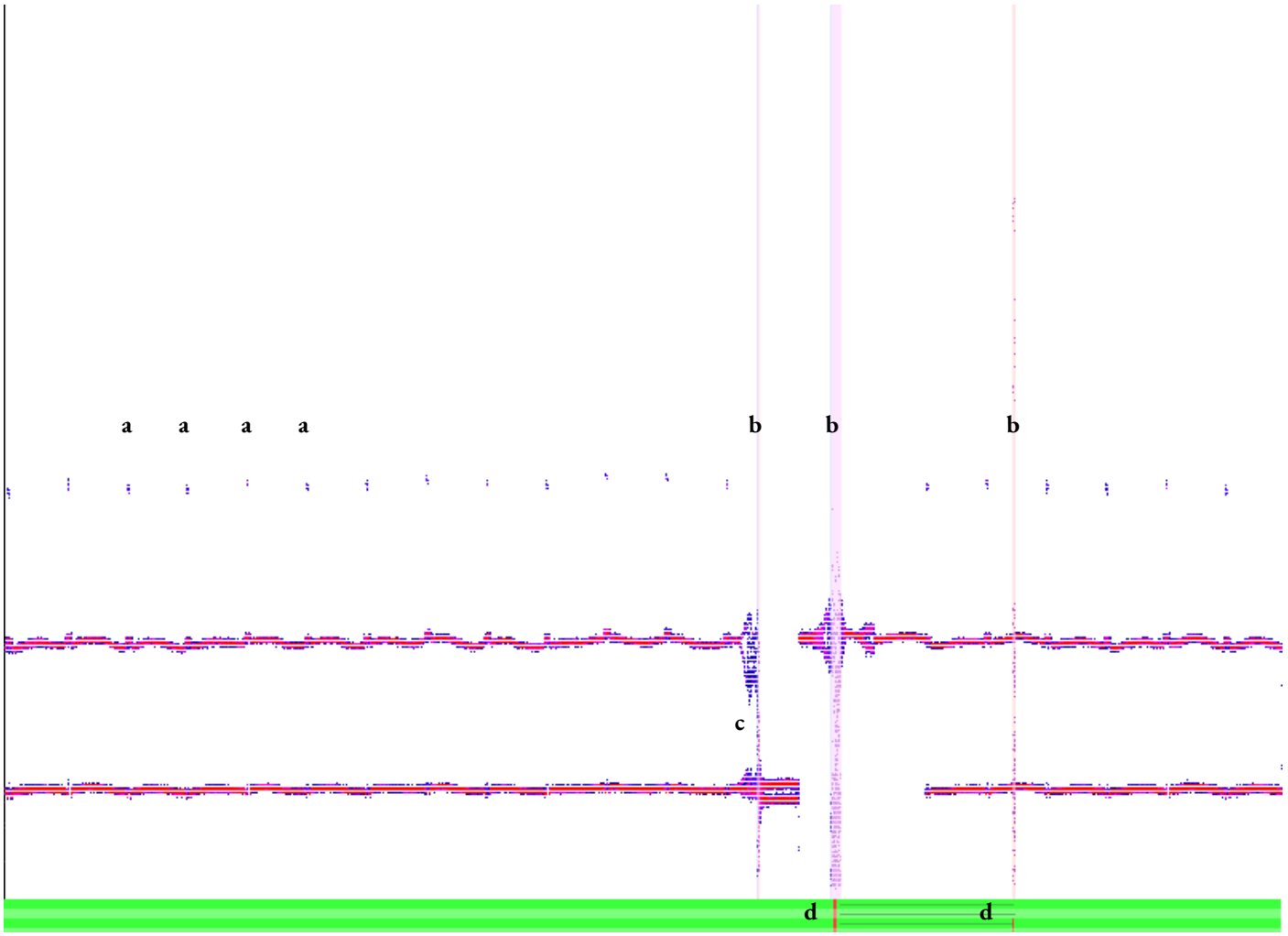
Graphs are rendered as if the track was imaged at a stable, exact 300 RPM. Therefore the default domain setting of [0.0, 0.2] represents an entire revolution between two index signals, which is exactly 0.2s at 300 RPM.

The default range setting [0.0, 2e-5] represents flux reversals that are at most 20us long.

Note, that all samples get processed while generating a graph, but depending on the settings the visible domain and/or range can be different.

All domain and range parameters can be negative, i.e. you can shift the view to a “proper” coordinate system. For these parameters scientific notation is accepted as well for easier input, e.g. 2e-5 is 0.00002.

Understanding the graphs



On standard disk formats the heatmap graph often consists of two (FM) or three (MFM and GCR) horizontal bands that represent the encoding on the disk at flux level, over a full revolution. The consistency graph at the bottom sums up how all the sampled revolutions match.

X-axis: An entire track revolution from index-to-index, representing the time elapsed since the index signal, i.e. the distance from the index hole. The x origin of the graph is the index signal and the graph ends just before the next index signal (meaning a complete revolution, since the disk rotates).

Y-axis: The length of flux reversals (aka density) at a specific point in time on the track. Given that various encodings normally use 2 or 3 different bitcell densities for encoding data (or more precisely 1 band for fundamental frequency and 1 or 2 of its harmonics) naturally you can see 2 or 3 bands on the graphs horizontally depending on the encoding used.

a) Start of a sector often can be identified by looking for this kind of evenly spaced repetitive marks on the heatmap.

b) The vertical outliers from top to bottom are a sign that this part of track contains noise, ie. no well defined flux timings are present.

c) The horizontal bands, when they start to bleed as seen here, are a sign of a dirty or damaged (surface or magnetic damage) disk or drive heads. If the bands become mixed, the correct flux timing and the data is unrecoverable.

d) In the green consistency / path solver graph (displaying all the sampled revolutions of a track stacked vertically), even if there are inconsistencies, displayed in red, for example from a worn out disk, the full track is still recoverable as long as a line can be drawn from left to right over the green area around any red regions. A complete vertical red section blocking the green area from top to bottom becomes unrecoverable.

However, if points **b**, **c** and **d** are located within areas that do not contain any actual data or positioned between actual sectors, usually right next to the start of a sector or at the very beginning or end of the track, the data should still be ok.

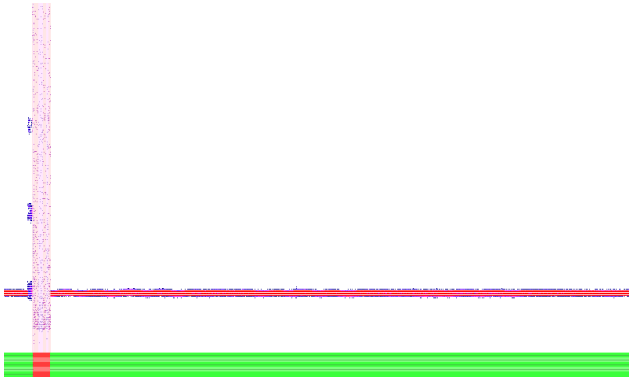
Write splices (often manifesting as incomplete, broken bit cells) occur when writing an entire track starts and stops - since the disk is rotating during the process, the start of the flux reversal data can be overwritten by the end of the data stream. Using the index signal to gating the write process may not be accurate enough to exactly control this, so tracks written at once often show a write splice. Writing less data than the track length will leave areas unformatted (on a new disk), or if the writing is not synchronised to the index, it will overwrite the previously written data (on a used disk). Again, the result is at least one write splice.

Amiga disk formats are normally designed to be written as an entire track; meaning that changing even a single sector on that track would require reading the track first (although usually it's buffered), changing the sector content and writing the track back to disk.

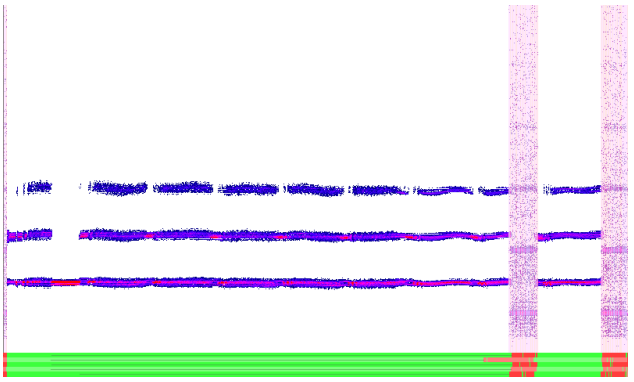
A common method of making it possible to write a single sector instead of an entire track is to separate sectors to header and data areas. The sector headers have enough gap leading to the sector data so that the hardware or software could quickly start writing new sector data once the correct sector position has been identified using the information stored in the header. Most disk systems use this approach by default, e.g. systems using hardware floppy disk controllers like PC or Atari ST, or hybrid controller approaches like the Commodore C1541 drive. The sector headers are written during formatting the track alongside with some predefined pattern for empty sector content. Once new sector data content is written by the user, the previous sector content will be overwritten starting with various sync marks, so as to identify the start of the new data area. Since this method is not entirely predictable (e.g. signal chain jitter, disk rotational differences, etc will introduce variability), this process will almost never overwrite the previously existing sector content exactly; it will start slightly sooner or later. Whenever that happens, new write splices may appear with broken bitcells at the start and the end of the new sector data area.

These unwritten or overwritten spots on a track or around sector data areas may cause inconsistencies and might show up as small inconsistent or unformatted areas. This is perfectly normal, however the presence of these artefacts means that the sector content was modified since formatting the track originally or in the case of duplicated disks, the original, factory new content was modified by the user.

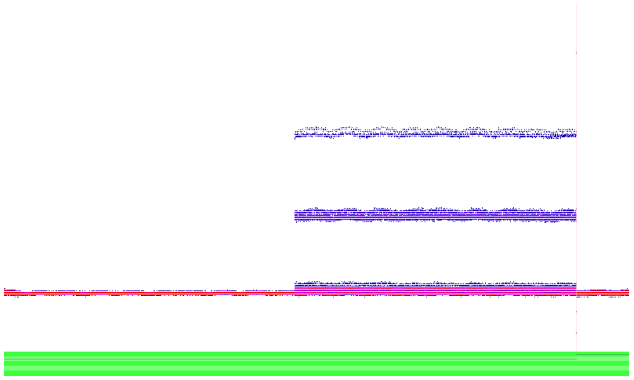
Understanding the various elements in the graph is important in interpreting the graphs and the overall condition of the disk. The next few pages contain example graphs from several different cases one might encounter.



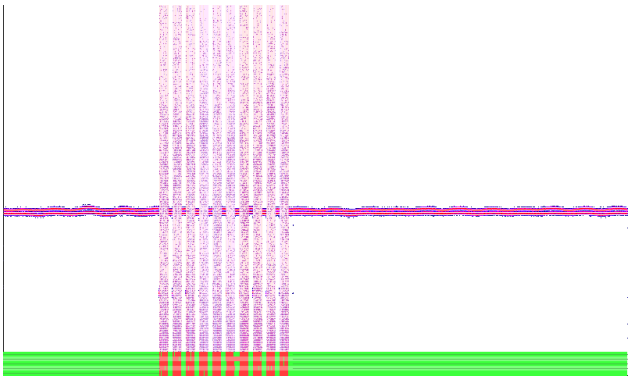
AcroJet [C64]
RapidLok protection, weakbits



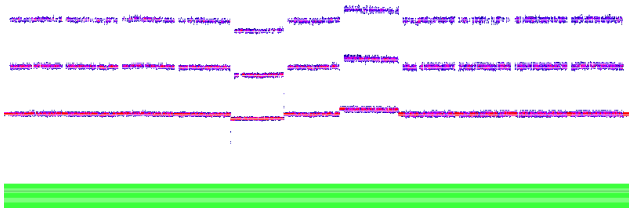
After Burner [CPC]
Weakbits



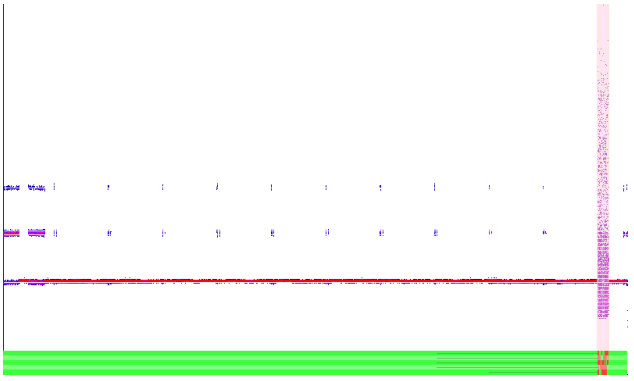
Asterix [C64]
Melbourne House protection



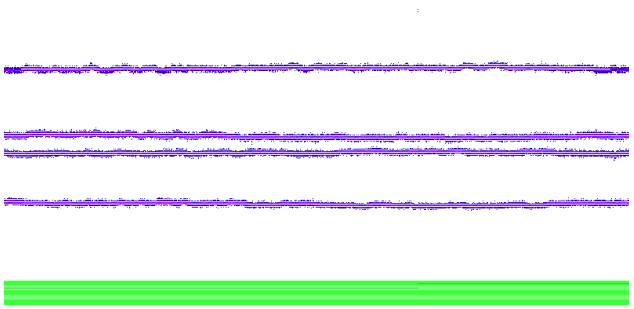
Asterix [C64]
Melbourne House protection, pattern & weakbits



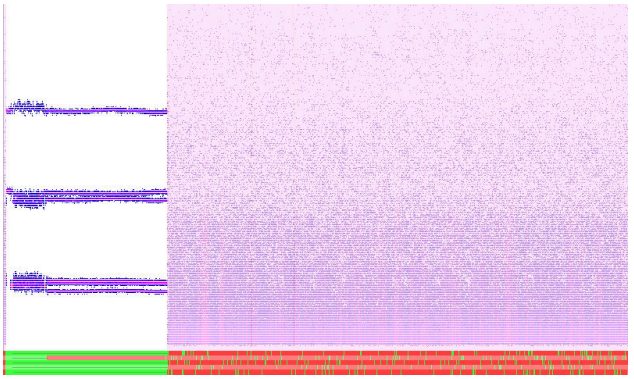
Batman The Movie [Amiga]
Variable densities on track, Copylock protection



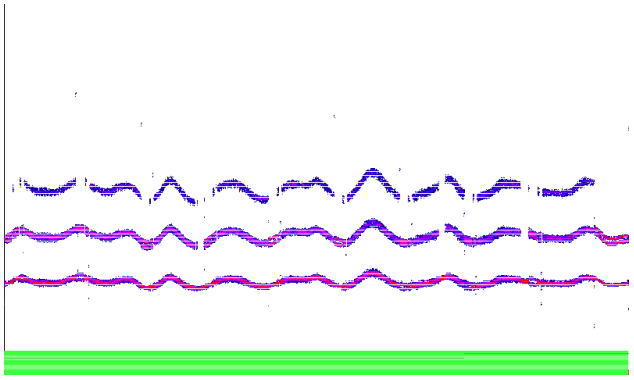
Behind The Iron Gate [Amiga]
Weakbits



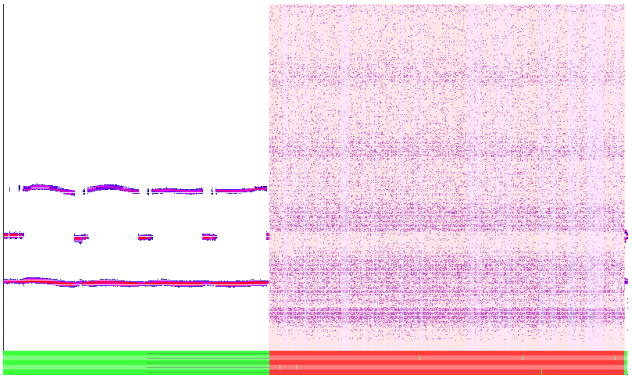
Big Trouble In Little China [C64]
Track written without precompensation



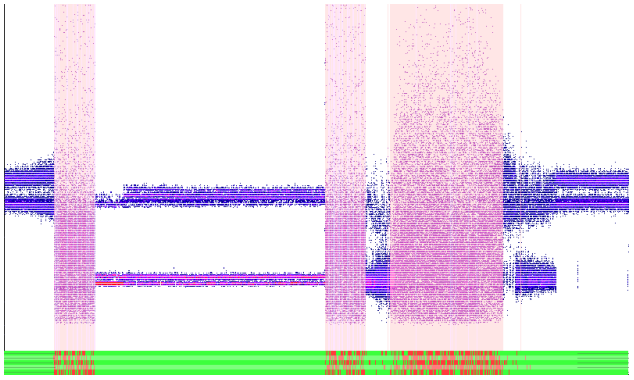
Black Thunder [C64]
Partially unformatted track



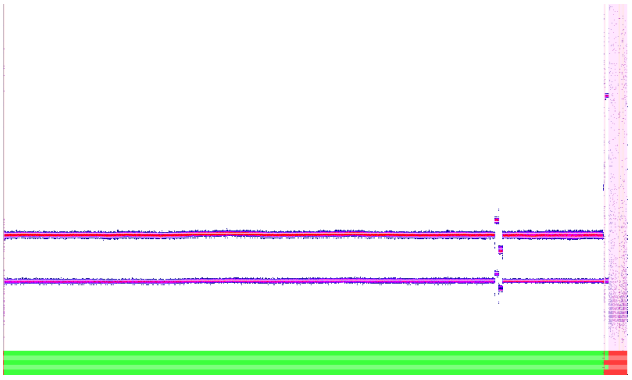
Bush 09 [Spectrum]
Bent/deformed disk or unstable drive speed (bent disk in this case)



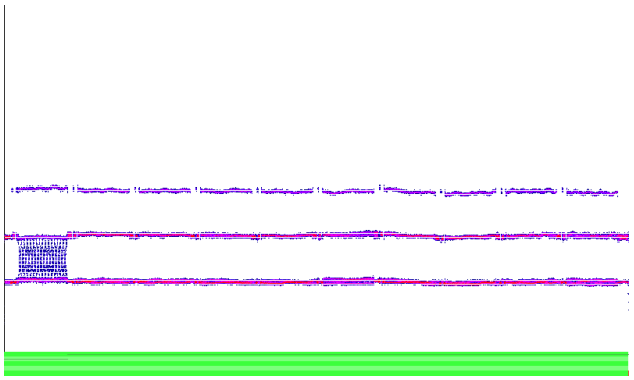
Chase H.Q. [Spectrum]
Partially unformatted track



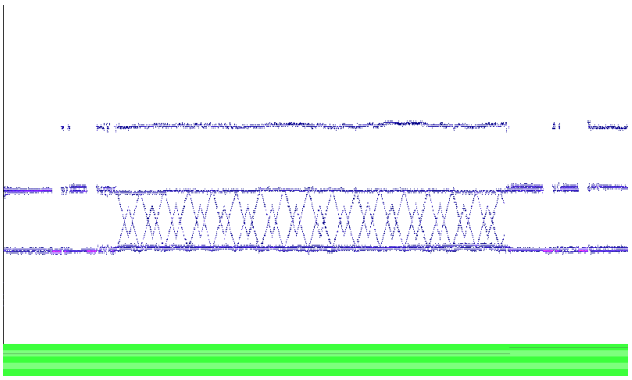
Choplifter [Apple II]
Partially used track and some crosstalk, Spiradisc protection



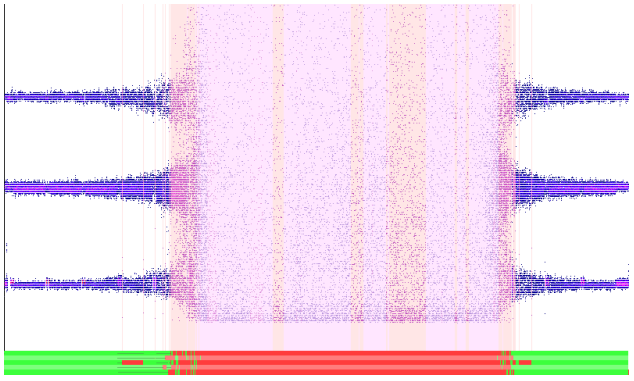
Dragons Breath [Amiga]
Short variable density areas on track, Speedlock protection



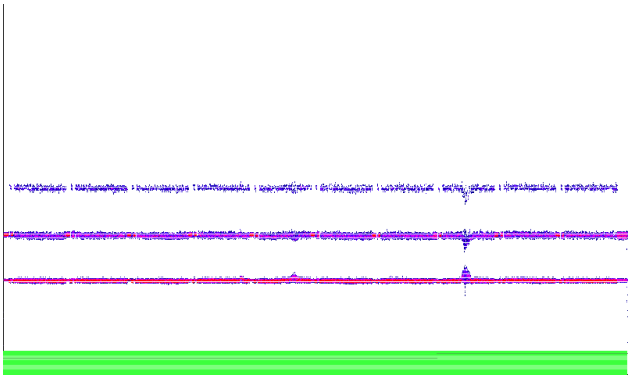
Dungeon Master [Amiga]
Programmed/predictable (deterministic) weakbits



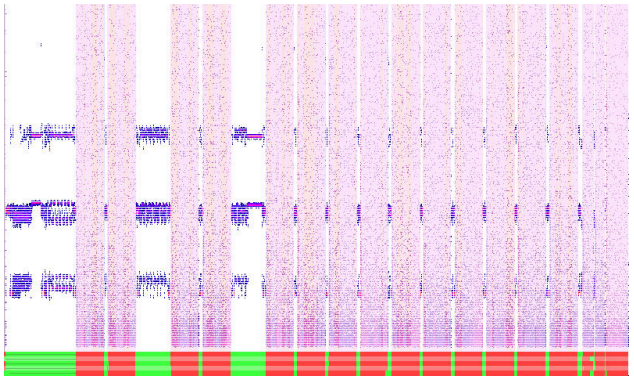
Dungeon Master [Amiga]
Programmed/predictable weakbits (closeup)



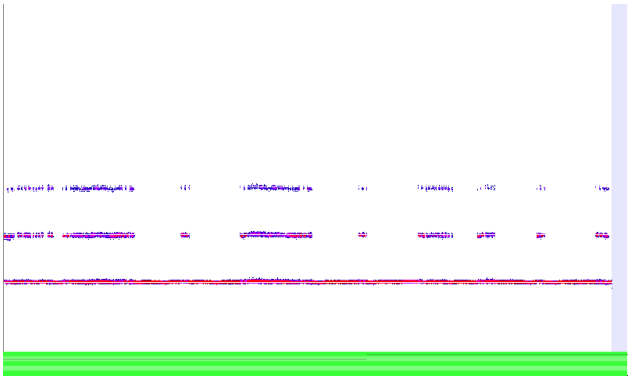
F-15 Strike Eagle [C64]
Crosstalk & unformatted area between normal tracks



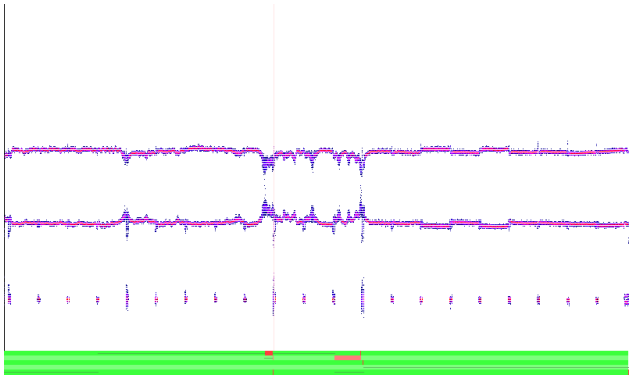
Fighter Bomber [Amiga]
Some crosstalk bleeding due to disk surface damage



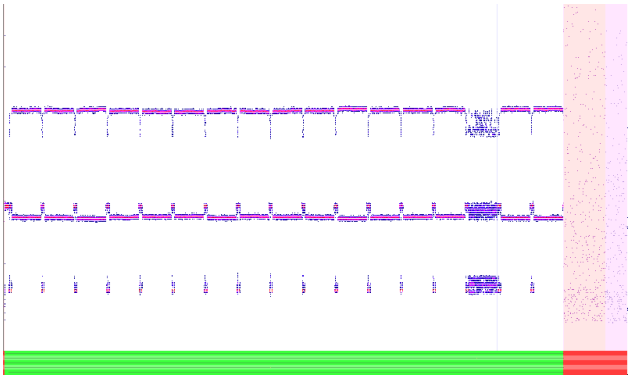
Flimbo's Quest [C64]
Very tiny regions of data between unformatted areas



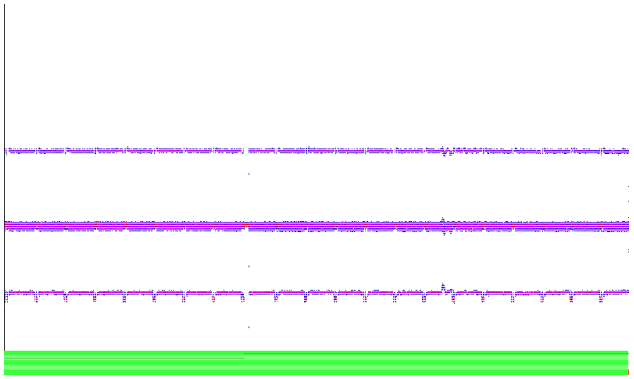
Guardian Angel [Atari ST]
Non-flux area



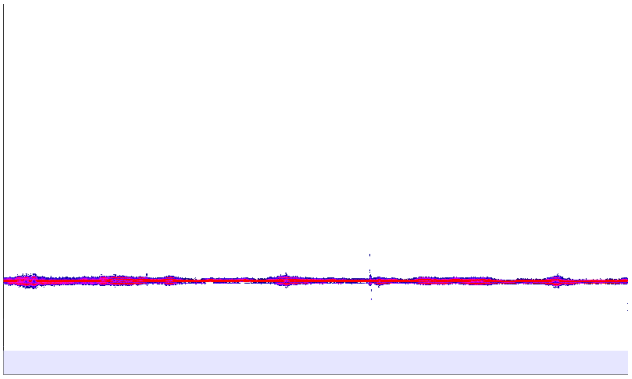
Halls Of Montezuma Master Disk [C64]
Severe crosstalk bleeding due to damaged disk surface



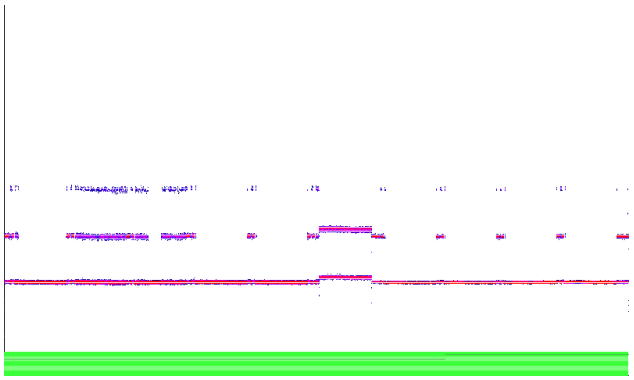
Hypaball [C64]
Protection, partially unformatted track



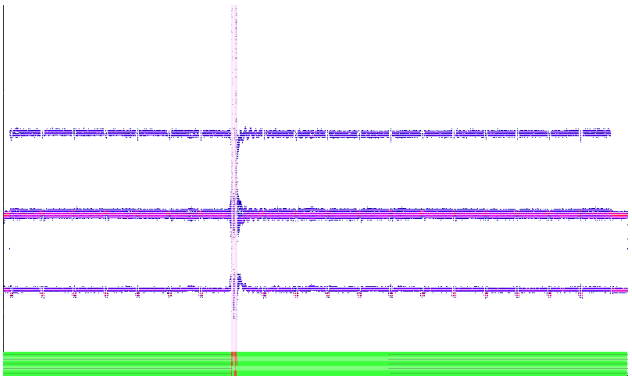
Labyrinth [C64]
Very slight crosstalk bleeding



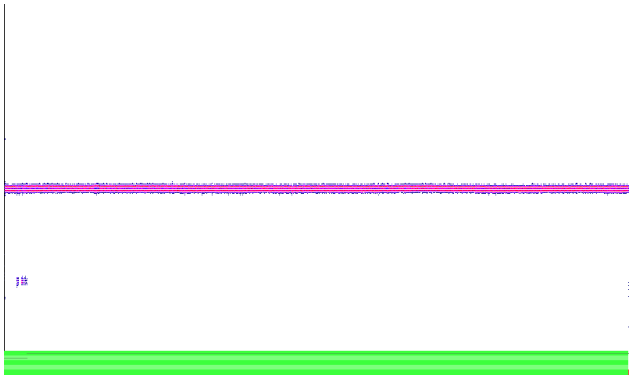
Lancelot [Atari ST]
Track filled (wiped) with a pattern



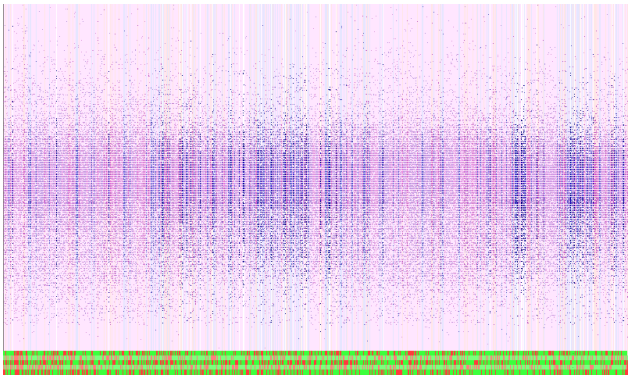
Midnight Resistance [Atari ST]
Variable densities on track, Copylock protection



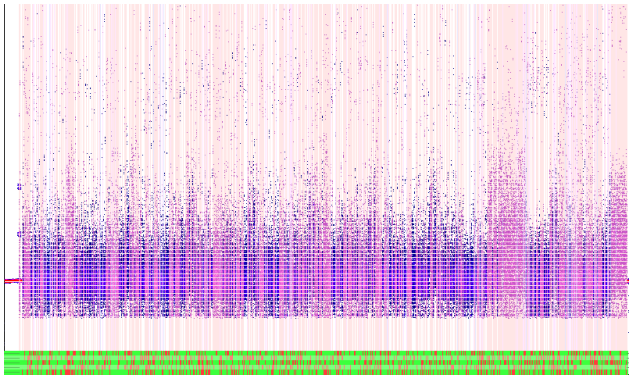
Milk Race [C64]
Severe crosstalk bleeding due to surface damage, etc.



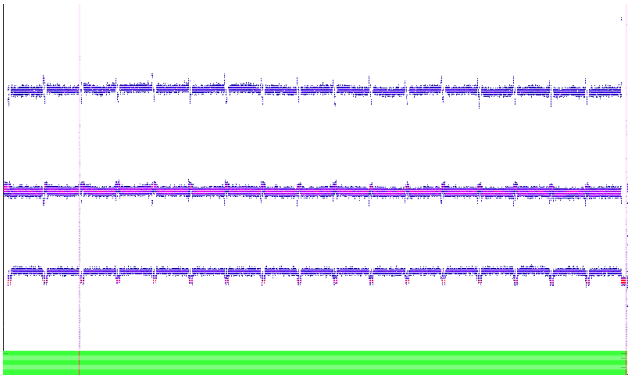
Milk Race [C64]
FM encoded Duplicator Info track (at the end of a disk)



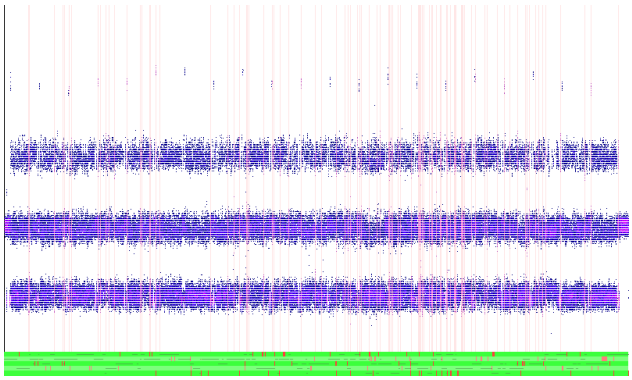
Milk Race [C64]
Unformatted track



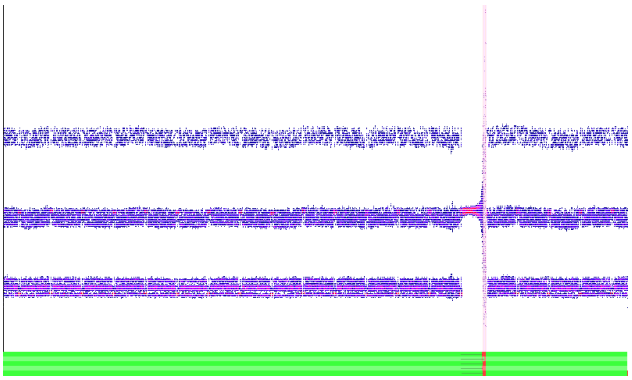
Powerdrift [Atari ST]
Very tiny region of data after the index signal



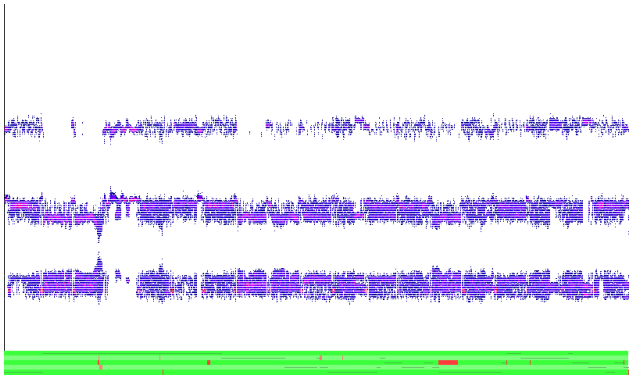
Re-Bounder [C64]
Cyan protection, weakbits at the end of second sector



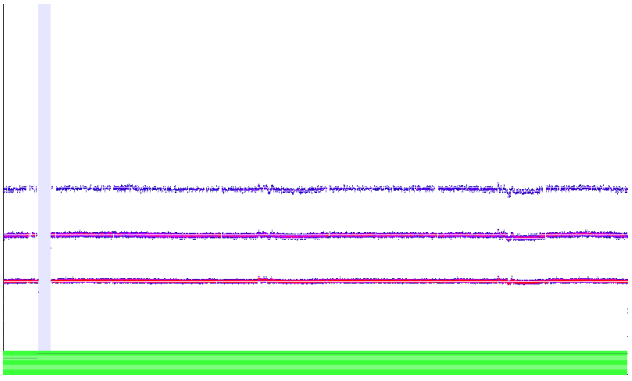
Retrograde [C64]
Very severe crosstalk bleeding, due to a bad disk



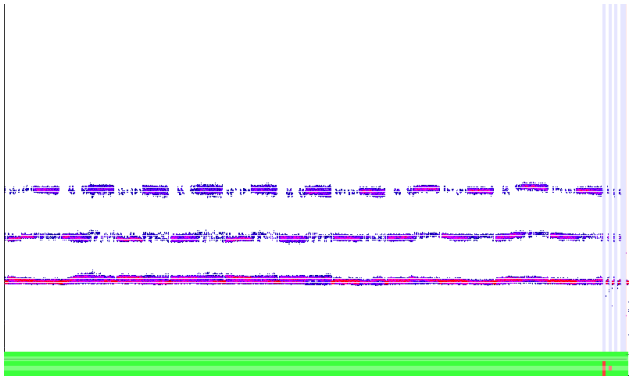
RoMuzak [C64]
Weakbits



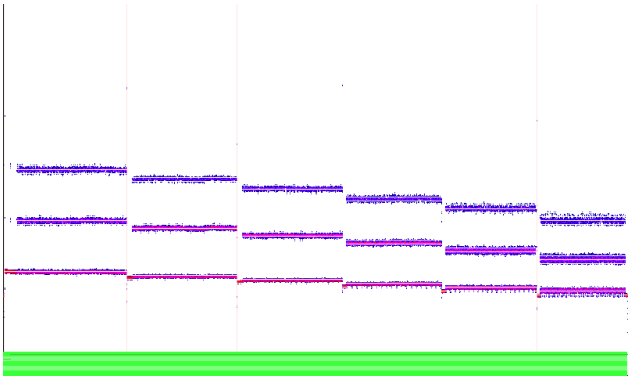
Stratton [C64]
Severe crosstalk bleeding, still recoverable



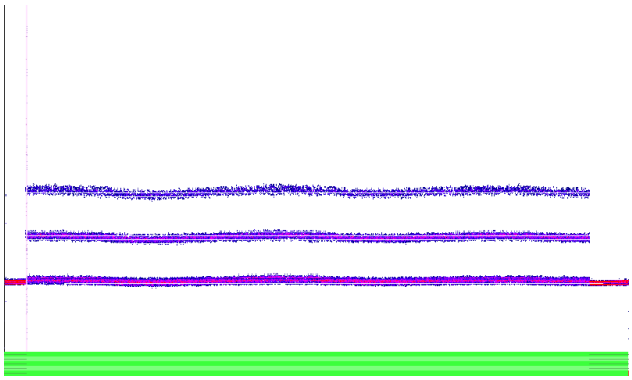
Turrican [Atari ST]
Non-flux area



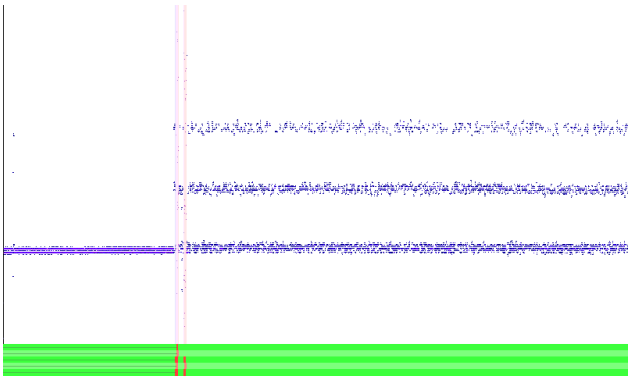
Bloodwych: Data Disks - Vol 1 [Amiga]
Non-flux areas



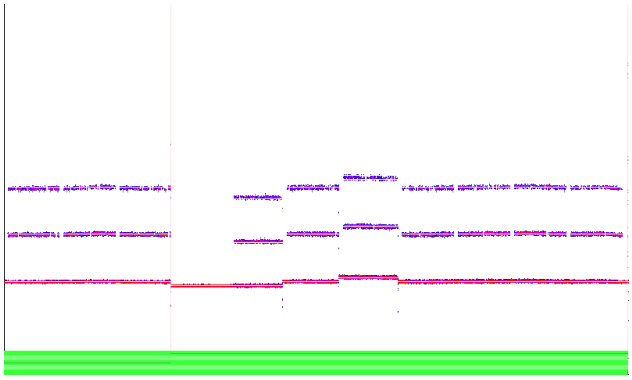
Merchant Colony [Amiga]
6 different densities used on a single track



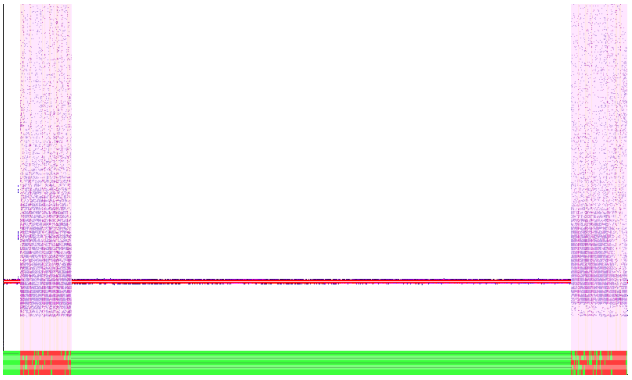
Flimbo's Quest [Amiga]
What looks like a simple weakbits protection...



Flimbo's Quest [Amiga]
...is actually a decryption key surrounded by weakbits (closeup)



Odyssey [Amiga]
Variable densities on track, Copylock protection alternate timing



Prison [Amiga]
Weakbits used as sector data

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 the 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 Fábián in 2001 as CAPS (the Classic Amiga Preservation Society), our highly specialised team has more than two decades 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 now using our own completely self-contained floppy disk 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 10.000 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

Contact: softpres.org/contact