KFPanzer Documentation

By Jean Louis-Guérin (DrCoolZic)
Revision 1.1 - August 2012

Table of Content

Presentation

This document describes the usage of the **KFPanzer** program. This program checks and prints the ***copy protection mechanisms*** found on an Atari Floppy Diskette imaged as a set of **Stream** files produced by a **KryoFlux** board from KryoFlux Products & Services Ltd.

The type of ***floppy disk protection mechanisms*** used on the Atari platform is very old and, with many years of development and the usage of sophisticated floppy disk hardware, has conducted to numerous protection methods frequently referred as **key disk protection**. The key disk protection method has at least two obvious qualities: first, a *key disk* can be simultaneously used as *protection* and *distribution* disk and second, this type of protection is very cheap but nevertheless hard to tamper with. So, key disks have been widely used for protection of Atari programs/games.

Floppy Disk Analysis Process Overview

You must follow this process to analyze the protection mechanisms of an Atari floppy:

1. You need to run the **DTC** program to produce a series of **Stream** files (one for each track and each side of each disk). The **KFPanzer** program expects the stream files to follow a specific organization and naming convention. To produce all the stream files as expected by the program you are advice to use the **AtariDump.bat** file or the **AtariDumpLog.bat** file. Usage of this batch files is detailed in Atari Dump Batch files.
2. You can then use the **KFPanzer** program to check all the protections mechanisms of the Atari FD. The program processes all the stream files and produces an output file that contains the list of all the protection mechanisms. There are several flags that can be specified during invocation to control the behavior of the analysis and the name of input and output files.

Running the KFPanzer Program

The program is invoked by typing:

```
KFPanzer <command_line_flags> game_name <output_file>
```

Where:

- **command_line_flags** are optional arguments that control several internal parameters of the **KFPanzer** program. The command line flags are described in the following section.
- **game_name** is the name of the floppy disk that you want to analyze. This parameter is mandatory.
- **Output_file** is an optional name for the output file. If no name is specified then the output of the analysis is displayed on the screen (standard out).

In order for the program to work all the stream files that compose the complete FD image must be organized as follow:

- All the stream files must be located in a directory named **fd_name**
- Each stream file in this directory must have the following name:
  game_name(disk_num-tot_disk).track_num.side_num.raw
  where:
  - disk_num is the number of the disk
  - tot_disk is the total number of disk for this game
  - track_num is the track number
  - side_num is the side number

For example if you want to check the protections for a game called BackToTheFuture, that is composed of 2 disks, the program expects the following stream files:

```
BackToTheFuture(1-2)00.0.raw
BackToTheFuture(1-2)00.1.raw
BackToTheFuture(1-2)01.0.raw
BackToTheFuture(1-2)01.1.raw
...
BackToTheFuture(1-2)83.0.raw
BackToTheFuture(1-2)83.1.raw
BackToTheFuture(2-2)00.0.raw
BackToTheFuture(2-2)00.1.raw
BackToTheFuture(2-2)01.0.raw
BackToTheFuture(2-2)01.1.raw
...
BackToTheFuture(2-2)83.0.raw
BackToTheFuture(2-2)83.1.raw
```

And all these files must be placed in a directory called

```
BackToTheFuture
```

A detail description of all the protections checked can be found in my document "Atari Floppy Disk Copy Protection Based on Key Disk". However a short description of the protection detected can be found in a later section Contents of the output File.

This program is very handy to check all the protections on a Floppy Disk, but it does not provide detail information on the protections. It is therefore advised to run the **KFAnalyze** program on the track(s) that contain protection mechanisms.

Note that the program fully implements a DPLL Data separator as described in section WD1772 DPLL Input Circuitry of this document and therefore the program should give results very close to an actual WD1772 FDC.

Command line flags

The **KFPanzer** program can be invoked with zero or several flags. The flags arguments must be specified before the **input_game_name**. Each argument consists of one letter preceded by a dash sign ("-"). When several options are used they must be separated by space and each of them needs to be preceded with a dash sign. The flags can be specified in any order.

The following example:

```
KFAnalyze -d2 -l3 DungeonMaster dm.txt
```

Indicates that the **KFPanzer** program will analyze the protections of the *DungeonMaster* games using the **-d2** and **-l3** options and produces a *dm.txt* output file that list all the protections.

-v flag

By using this flag detailed information about the analysis of the stream file is produced (e.g. timing violation location, synch mark detection and, half-cell resynchronization, etc.). This flag is for debug purpose and in most cases should not be used as a huge number of information will be produced.

-d<num> flag

By default the program try to guess the number of disks of the game to analyze (up to 9 disks maximum). You can specify the number of disks of the game to analyze with this option. For example **-d3** indicates a game with 3 diskettes.

-I flag

This flag is used to filter the print out of the protection mechanisms found. The protections are classified in three levels:

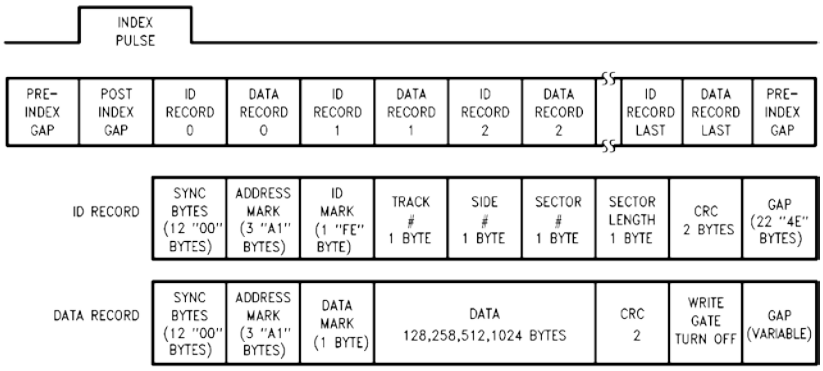
- level 1 : The protection mechanism is almost certainly used to copy protect the FD
- level 2 : The protection mechanism is probably used to copy protect the FD
- level 3 : The mechanism detected is informational and *may* indicate a protection.

The levels of the protection detected are detailed in a following section.

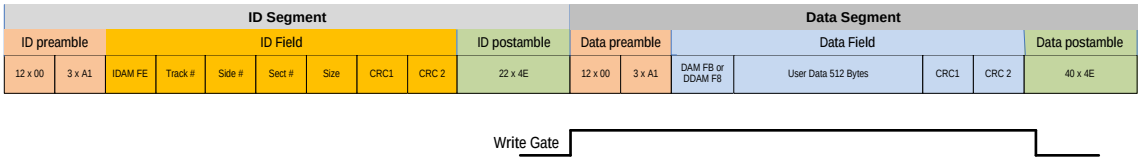
Terminology

The following terminology is used:

A **floppy diskette** has two **sides** read by two **heads**. Each side is composed of concentric **tracks**. Each track is made up of several **sectors** (or **records**). Each Sector contains several **fields (**or **blocks)** called: The **Gaps** (GAP1, GAP1a, GAP2, GAP3a, GAP3b, GAP4, and GAP5), the **SYNCH** bytes followed by an **Address Marks** (IAM, IDAM, DAM, and DDAM), the **ID** fields, and of course the **Data** fields.



A complete track



A sector

Contents of the output File

This section describes the contents of the output file produced by the **KFPanzer** program.

The purpose of the program is to check the images of one or several FD to find all possible copy protection mechanisms used.

The protections are displayed as following:

```
D<disk_num> H<head_num> T<track_num> S<sec_num> <Prot_type> <detail> <(info)>
```

Where:

- disk_num is the disk number
- head_num is the head (side) number
- track_num is the track number
- sec_num is the sector number. For a track protection this field is absent.
- Prot_type is the type of protection. All the detected protections are listed below
- Detail is the detail name of the protection
- (info) is optional information displayed with certain protections. For example for a Short Track this field contains the number of bytes of the Track

For example:

```
D2 H0 T73 LGT Long Track (6396)
D2 H0 T74-79 TNF Track Not Found
D2 H1 T00-79 TNF Track Not Found
```

The following protection mechanisms (with their filter level) are detected by the program:

- DBI: Data Beyond Index (level = 1)
  The ID field of the last sector is placed **before** the index and the corresponding DATA field of this sector is placed **after** the index. This is the extreme version of the DOI protection described below. The position of the data sector is also displayed.
- DOI: Data Over Index (level = 1)
  Indicates that the DATA field of the last sector span over the index. The number of bytes placed after the index is also displayed.
- DSN: Duplicate Sector Number (level = 1)
  Two or more sectors have the same number.
- EXT: Extra Track (level = 2)
  Normal FD have 80 tracks numbered 00-79. EXT indicates tracks in the range 80-83 that contains information.
- FAA: Flux reversal in Ambiguous Area (level = 3)
  The flux reversal are placed at the border of the inspection window. This is used usually to produce fuzzy bits detected by the FZD protection. This field is therefore considered as informational. The total number of transitions in ambiguous area is also printed.
- FZD: Fuzzy bytes in Data field (level = 1)
  A specific sector contains fuzzy bytes. The number of the first bytes that differ from read to read is displayed.
- IBV: Intra-sector Bit rate variation. (level = 1)
  The same sector contains several regions with bit-rate **above** normal **and** bit-rate **below** normal (e.g. macrodos). The number of bytes that are 3% above or 3% below the normal value is also printed. It is recommended to use the **KFAnalyze** program on this track to analyze accurately the protection.
- IIF: Invalid ID field (level = 2)
  The ID field of the sector contain invalid values for the track number, and/or the side number, and/or the sector size.

- IDG: Invalid data in GAP field (level = 2)
  Some GAP fields (GAP1, GAP3, GAP4) contains data that cannot be written by a WD1772 FDC into GAP (range 245-247). The number of invalid data is also displayed.
- ISN: Invalid Sector Number (level = 1)
  ID fields contains a sector number that cannot be written by the WD1772 FDC (value in range 245-247).
- ISS: Invalid Synch Sequence (level = 3)
  A normal Synch sequence is 3 synch marks (A1 or C2) followed by an AM (address mark). Any sequence that does not follow this rule is detected and reported as an ISS. Placing a synch mark in a GAP allow to place “hidden data” in the GAP. Therefore this information can be useful to further analyze the track with **KFAnalyze**. The number of invalid synch sequence is also printed.
- ITN: Invalid Track Number (level = 2)
  Normally the track number in the ID field of the sector must be the same as the track analyzed. If the numbers are different this protection is displayed. The track value found in the ID field is also displayed.
- LGS: Long DATA Sector (level = 1)
  A normal DATA sector has a length of 16480 µs (515 * 32 µs). Long sector indicates a sector with a length 3% above normal. The length of the sector is also printed.
- LGT: Long Track (level = 1)
  A normal track contains 6250 bytes (200000 / 32). A long track is a track that contains about 3% more bytes. The number of bytes of the track is also displayed
- MTV: MFM Timing Violation (level = 3)
  Normal MFM flux reversals bands for a double density diskette are 4, 6, and 8 µs. If some flux transitions are above or below these bands a MTV is detected. In most cases this corresponds to partially unformatted DATA field. This is used usually to produce fuzzy bits detected by the FZD protection. This field is therefore considered as informational. The total number of violations is also printed.
- NFA: No Flux reversal Area (level = 1)
  This very difficult to reproduce protection that is obtained by not having any flux reversal in an area that can span for several milliseconds on a track. The position of the NFA on the track is also displayed.
- NOS: Number Of Sector (level = 3)
  The normal number of sectors per track is 9 or 10. If the actual number of sectors is above 10 or below 9 the NOS protection is detected. The number of sector found is also displayed. This is informational only
- NSD: Non Standard DAM (level = 2)
  The Address Mark placed at the beginning of the DATA field is not one of the two standard DAM. The value of the detected DAM is also printed.
- NSI: Non Standard IDAM (level = 2)
  The Address Mark placed at the beginning of the ID field is not the standard IDAM. The value of the detected IDAM is also printed.
- SBD: Sector with Bad Data (level = 1)
  The DATA field of the sector is read with a CRC error
- SBI: Sector with Bad ID (level = 1)
  The ID field of the sector is read with a CRC error
- SND: Sector with No Data (level = 1)
  A sector contains a normal ID segment which is not followed by a DATA segment.

- SHS: Short Data Sector (level = 1)
  A normal DATA sector has a length of 16480 µs (515 * 32 µs). Short sector indicates a sector with a length 3% below normal. The length of the sector is also printed.
- SHT: Short Track (level = 1)
  A normal track contains 6250 bytes (200000 / 32). A Short track is a track that contains about 3% less bytes. The number of bytes of the track is also displayed
- SID: Synch mark in ID/DATA field (level = 3)
  The presence of synch mark inside data may indicate some tricks that can be further analyzed by **KFAnalyze**. This is especially true in presence of overlapping sectors (SWS) where it is possible for example to read clock flux reversals as data flux reversal by placing an appropriate synch mark. The number of bad synch mark detected is also printed.
- SSZ: Sector Size (level = 2)
  Normal Atari FD uses 512 or 1024 bytes per sector. Any other value is detected as SSZ mechanisms. The value of the size found in the ID field is also displayed
- SWS: Sector Within Sector (level = 1)
  A sector is placed inside another sector (overlapping sectors) usually in the DATA field. This allows to have 12 or more pseudo sectors in a track.
- TNF: Track Not Found (level = 2)
  Normal FD have 80 tracks numbered 00-79. TNF indicates missing tracks (unformatted) in the range 00-79. Often several consecutive tracks are missing and therefore in this case the program uses a range notation. For example **T45-60 TNF** indicates that all the tracks between 45 and 60 are not formatted or empty.
  The indication Dx H1 T00-79 TNF indicates that all the tracks for head 1 are missing and therefore this indicates that the FD is single sided

Atari Dump Batch files

Two batch files are provided to help imaging Atari FD with the KryoFlux board. These two batch files are called **AtariDump.bat** and **AtariDumpLog.bat**.

The only difference between the two batch files is that the first one display the output from the DTC program on the screen, while the second store this information in a file. The advantage of the first version is that you can see what is going on the screen but the drawback is that you lose a lot of useful information unless you stay stuck in front of your screen. The second version is the opposite you do not see what is going on during imaging but all the information is kept in a file and therefore you can review later all the problems detected by the DTC program.

The scripts are very easy to use:

- You are first requested to provide the name of the Floppy disk to image. This is a mandatory input that you must provide.
- Then you can enter the File/directory name that you want to use to store the files. By default the batch file propose the name of the floppy disk you have just entered with spaces removed. For example if you have entered “Back To The Future” the proposed file name is “BackToTheFuture”. You can accept this name by pressing the return key or enter a name of your choice.
- Then you need to enter the type of program (Game / Utility / Music /etc.). The default is “Game”. You can accept this type by pressing the return key or enter a type of your choice
- Then you need to enter the number of disks that are part of the game/program. The default is “1”. You can accept this number by pressing the return key or enter a different number.
- Then you need to enter the name of the Publisher and/or Developer. The default is “unknown”. You can accept this name by pressing the return key or enter a name of your choice.
- Then you need to specify the type of release “Retail, Budget or Compilation (with name)”. The default is “unknown”. You can accept this name by pressing the return key or enter a name of your choice.
- Then you need to specify the year of release. The default is “unknown”. You can accept this value by pressing the return key or enter a year of your choice.
- Then you are asked to enter the country of release. The default is “unknown”. You can accept this name by pressing the return key or enter a name of your choice.
- Then you are asked to specify the language(s) used in the program/game. The default is “English”. You can accept this language by pressing the return key or enter another language(s) of your choice.
- You then need to enter the recommended Atari model to run the program/game. The default is “Atari ST/STe”. You can accept this model by pressing the return key or enter a model of your choice.
- You need to enter the place of purchase. The default is “unknown”. You can accept this name by pressing the return key or enter a name of your choice.
- You have to specify if the program is known to be working. The default is “yes”. You can accept this name by pressing the return key or enter a “no” or “unknown”.
- You have to specify if the program run on NTSC or PAL machine. The default is “unknown”. You can accept this name by pressing the return key or enter a name of your choice.

- You now have the chance to enter any extra information that you want to pass to SPS people that will be building the IPF file. The default is “none”. You can accept this default by pressing the return key or enter information of your choice.
- You now have to specify the drive to use. The default is “0”. You can accept this value by pressing the return key or enter a “1”. Any other value entered will be considered as 0
- At this point the batch file asks you to confirm that the information entered is correct. If you answer yes the program continue, if you enter no the batch file offer you to change any of the above information.
- Now the batch file will guide you in the process of creating the images by asking you to place the FD in the drive and to press any key to start the process.
- When all the FD of the game/program have been imaged the batch file asks if you want to image another game or not…

The batch files invoke the DTC program with the following flags:

```
DTC -f<name> -i0 -i2 -i4 -v300 -d<drive>
```

The batch files must be placed in the same directory where the DTC program resides.

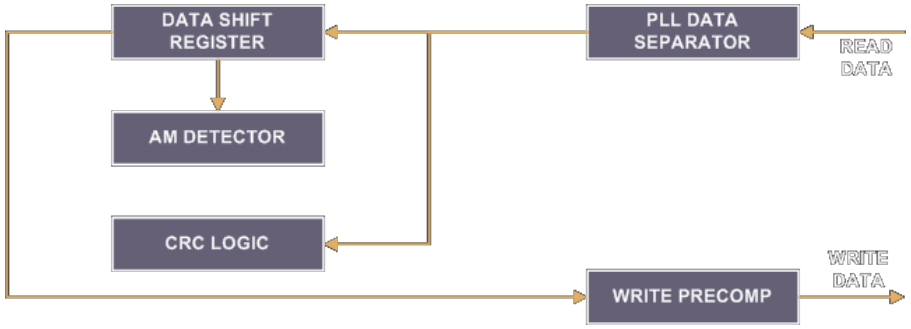
Beyond the stream files the **AtariDump.bat** batch file also produce a **game_name.txt** file in the game directory that contains all the information entered about the game. The **AtariDumpLog.bat** produces **game_name.txt** file as well as a **game_name(disk_num-total_disk).log** file for each disk that contains all the output of the DTC program (stdout as well as stderr).

WD1772 DPLL Input Circuitry

This section provides basic information on the DPLL of the WD1772 and how the decoded bits are entered in the shift register. It does not describe the data separator which is based on usage of the AM (Address Marks) detector to find a specific pattern in the shift register (usually during gaps) as it is pretty simple to understand and not useful in the context of this document.

The KFAnalyze program uses the same algorithm to process the data.

This is a simplified block diagram of the input circuitry of the FDC:

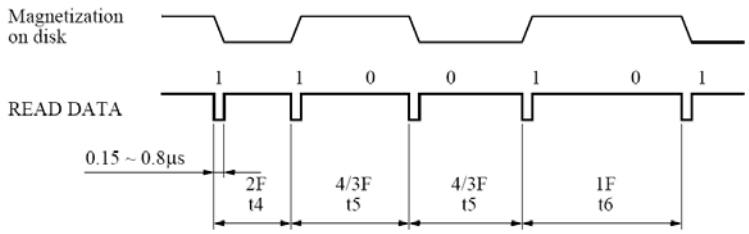


The WD1772 uses a digital phase lock loop (DPLL) circuit for reading the input data transmitted from FD media. Inspection windows are established that have duration proportional to the frequency of arrival of the data, and start/stop times that can be adjusted so that subsequent data bits will be received in the middle of the inspection windows. To achieve this, the DPLL apply **frequency** and **phase** corrections that compensate the input data frequency drift due to unsteadiness of the motor drive speed (frequency drift), and the migrations of the magnetic transition area (phase drift).

The DPLL used in the WD1772, as well as many other FDC build in the 80s, implement the algorithm described in the public **US patent 4,870,844**. The patent is rather complex and this section will only highlight some of the important aspects of the DPLL algorithm that are needed to understand special behaviors like fuzzy bits, long/short track, etc. For a detail understanding of the DPLL please refer to the patent.

Note that the KFanalyze program fully implements the DPLL as described in this patent.

Let's first look at typical MFM data encoding:



Note : READ DATA pulse will be detected within t7 from is nominal position. (When PLL separator is used with recommended write pre-compensation.)

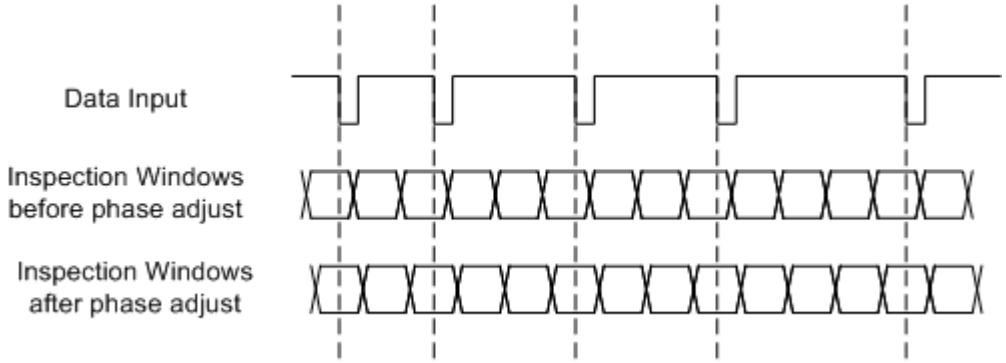
Density mode	rpm	t4	t5	t6	t7
2MB mode	300	2µs, Nom.	3µs, Nom.	4µs, Nom.	±350ns
1MB mode	300	4µs, Nom.	6µs, Nom.	8µs, Nom.	±700ns

As we can see the nominal values for possible transitions spacing in DD MFM are: 4µs, 6µs, or 8µs.

The data input circuit ensures that the data pulses received are converted into data bits and stored in a data shift register (DSR). For that matter the digital phase lock loop defines inspection windows that repeat every 2µs (half-cell size so we sample the clock bits). A one is input to the shift register if a data pulse is received at any time during one inspection windows; otherwise a zero will be stored in the shift register as the value of the current bit.

The period of the inspection windows is gradually adjusted (expanded or shortened) to compensate an eventual frequency shift affecting the input data transfer. This frequency correction is computed based on the history of the location (relative to the inspection window) of the last three flux transitions.

Ideally, individual pulses should be located in the middle of the inspection windows. To achieve this, the start/stop time of the inspection windows are adjusted to compensate for deviation, from ideal, in time of arrival of the most recently detected data pulse. This phase correction is done proportionally to the distance of the transition with the middle of the inspection window.



The proper ratio of phase and frequency correction provided in the loop is carefully balanced so that the DPLL is fast settling but stable. A large amount of phase correction cause the loop to settle faster but also make it more sensible to noise. On the other hand if too much frequency correction is used, the loop can become unstable.

With the above information it is now easy to understand that if a transition happens at the extreme border of an inspection window it will happen into one or the next inspection window based on small variation (for example of the drive rotation speed) and will therefore result in pseudo random values returned (fuzzy bits). For example having a transition at 5µs after the previous one can be interpreted as a transition after 4µs or a transition after 6µs based on small frequency variation of the input.

It is interesting to know that the DPLL as defined in the patent allow an input frequency variation of up to 9%. This corroborates the actual measurement made for the WD1772 that correctly interprets bits with a variation of 9 to 10 % in DD MFM (and about 100% in SD FM). Note that these values are well above the variation used by the protection mechanism like Copylock (about 5%) and therefore data in this kind of sector should be read correctly. However the Copylock mechanism not only varies the bit cell width but it also has uses fuzzy bits resulting in random values.