**Teledisk**

**Image File Format**

**Notes**
**Dave Dunfield**
**April 2, 2007**
**Last revised: September 23, 2024 by Mark Ogden**

# 1. Introduction

Teledisk is a program which reads non-PC format diskettes into image files for archival and can later recreate a copy of the original disk from the image file. Once popular for archival of classic computer
software, Teledisk has been withdrawn from the market by it's manufacturer and is no longer legally available. This presents a problem for those people who have data archived with Teledisk,
because the file format is proprietary and not documented rendering the data useless without the program.

In my development of ImageDisk (a replacement for Teledisk), I have created a utility to convert Teledisk .TD0 images into ImageDisk .IMD format - in doing so, I have researched the Teledisk format, read other peoples notes, done some reverse engineering myself, and come to what I believe is a somewhat complete understanding of the contents of a .TD0 image file.

This document presents my notes on the Teledisk .TD0 image file format.

This document is a work in progress. If you have any information to add, corrections etc. Please contact me. I can be reached via the contact information on my web site.

https://dunfield.themindfactory.com/contact.htm

## 1.1 Acknowlegements

The following people whom I have never met have saved me tons of time by making the results of their related efforts freely available.

**Will Krantz**: Wrote a program called wteledisk which converts TX50 .TD0 into binary files for an emulator. Published a web page with his notes and source.

**Sergey Erokhin**: Provided more details for Wills web page.

**Simon Owen**: Published source code to read some .TD0 files for his SimCoupe emulator.

**Haruhika Okumura**: Published LZSS-Huffman source code which has become "the reference" for many implementations.

**Mark Ogden**: Updated notes to reflect v2.23 of Teledisk, added information on the "old" advanced compression and multi volume disk sets and made some corrections.

# 2. Image file Format

The overall disk image file has this format:

```
Image header                (12 bytes)
;If the image was created using "Advanced Compression", everything
;below this line is compressed with LZW or LZSS-Huffman encoding.
Optional comment header     (10 bytes)
Optional comment data       (Variable size)
 ;For each track on the disk ...
 Track header               (4 bytes)
   ;For each sector within the track
   Sector header            (6 bytes)
   Optional data header     (3 bytes)
   Optional data block      (variable size)
;Image ends with Track header beginning with 255 (FF hex)
```

Although rarely used, the format also supports overflow of data into another file. This is described in the section Multi-volume support.

If the Teledisk image was generated with "Advanced Data Compression", all parts of the file following the "Image Header" are compressed as a single block as described in the section Advanced Compression.

# 3. Image Header

The image header describes global information about the disk image. It is never compressed, and is laid out in the following format:

```
Signature                   (2 bytes)
Sequence                    (1 byte)
Checksequence               (1 byte)
Teledisk version            (1 byte)
Density                     (1 byte)
Drive type                  (1 byte)
Stepping                    (1 byte)
DOS allocation flag         (1 byte)
Sides                       (1 byte)
Cyclic Redundancy Check     (2 byte little endian word)
```

## 3.1 Signature (2 bytes)

- Contains 'TD' for normal compression
- Contains 'td' for advanced compression

## 3.2 Sequence (1 byte)

- TDCHECK reports "bad header" if this value is set to anything other than 00. Other disks in a multi-volume set check the sequence matches, i.e. it is also 00.

  A consequence of this is that the signature can be treated as a simple C string.

## 3.3 CheckSequence (1 byte)

- This must be the same for the all disks in a multi-volume set. It is typically 0.

## 3.4 Teledisk file format version (1 byte)

- Encodes the version number of the file format used to create the image. It is in the form High-nibble.low-nibble. eg: 15 = 1.5
  Note v1.05 and possibly earlier of Teledisk do not support file formats earlier than 1.0.

## 3.5 Density (1 byte)

- The lower 3  bits encode the recording density of the source diskette.

```
000 = low density      - 250kbps
001 = low density      - 300kbps
01x = high density     - 500kbps
10x = extended density - 1000kkbps
```

- The high bit indicates single-density diskette  i.e. FM encoding.

## 3.6 Drive type (1 byte)

- Indicates the type of drive the disk was made on. TDCHECK provides descriptions based on this, the Density and the Stepping as follows

```
Type Condition            Description
0    (stepping & 2) != 0  5.25" disk
0    (stepping & 2) == 0  5.25" 96 tpi disk in 48 tpi drive
1                         5.25" disk
2    (not low density)    5.25" disk
2    (stepping & 1) != 0  5.25" 48 tpi disk (low density)
2    (stepping & 1) == 0  5.25" 96 tpi disk (low density)
3                         3.5" disk
4                         3.5" disk
5                         8" disk
6                         3.5" disk
```

- The density and FM/MFM encoding are appended to these descriptions. TDCHECK reports both 250kbps and 300kbps as low density.
- Note the 96 tpi disk in 48 tpi drive seems unusual as in the best case it would read alternate tracks
  but it is possible that it will read corrupt data due to flux transitions from adjacent tracks.

## 3.7 Stepping (1 byte)

- Encodes step type in lower 2 bits

```
0 = Single-Step
1 = Double-step
2 = Even-only step (96 tpi disk in 48 tpi drive)
```

- High bit indicates presence of optional comment block

### 3.8 DOS allocation flag (1 byte)

non-zero means the disk was read using the DOS FAT table  to  skip unallocated sectors.

### 3.9 Sides (1 byte)

Encodes the number of sides read from the disk.

```
01             = One
anything-else = Two
```

### 3.10 Cyclic Redundancy Check (2 byte little endian word)

The 16 bit CRC (see below) is calculated over  the  first 10 bytes of the header and should match the value stored in this field.

## 4. Comment Header / Data block

The comment block encodes an ASCII comment as well  as  the  creation date.  It's presence is indicated by the high bit of  the  "Stepping" field in the image header being set.

When present,  it occurs immediately after the Image  header  in  the following format:

```
 Cyclic Redundancy Check     (2 byte little endian word)
 Data length                 (2 byte little endian word)
 Year since 1900             (1 byte)
 Month                       (1 byte)
 Day                         (1 byte)
 Hour                        (1 byte)
 Minite                      (1 byte)
 Second                      (1 byte)
```

Following the comment header are comment line records,  consisting of ASCII text terminated by NUL (00) bytes.

### 4.1 Cyclic Redundancy Check (2 byte little endian word)

The 16 bit CRC (see below) is calculated over the comment block  from the "Data length" field through to the end of any following comment line records.

### 4.2 Data length (2 byte little endian word)

This is the  total length of the comment line records  which  follow  the comment header, maximum 584 bytes.

### 4.3 Year (1 byte)

Gives the year the image was created as an offset from  1900.  eg:  2007 is encoded as 2007 - 1900 = 107.

## 4.4 Month (1 byte)

Gives the month the image was created using a zero index. ie: 0=January, 11=December.

## 4.5 Day (1 byte)

Gives the day (of the month) the image was created using a range of 1-31.

## 4.6 Hour, Minute, Second (1 byte each)

Gives the time of day the image was created using 24-hour time.

## 4.7 Comment data block (variable size see 4.2)

Contains ASCII comment text with each line terminated by a NUL (00).

To display the comment data, read and output this many bytes following the header, translating NUL (00) bytes into newline sequences.

Note due to the way the comment is captured in Teledisk, it seems reasonable to remove trailing newlines and spaces to avoid unnecessary blank lines.

# 5. Track Header

Every disk track recorded in the image will begin with a track header, which has the following format:

```
Number of sectors           (1 byte)
Cylinder number             (1 byte)
Side/Head number            (1 byte)
Cyclic Redundancy Check     (1 byte)
```

## 5.1 Number of sectors (1 byte)

This field indicates how many sectors are recorded for this track. This also indicates how many sector headers to expect following the track header.

A number of sectors of 255 (FF hex) indicates the end of the track list. No other fields occur in this record, and the CRC is not checked.

Note, the number of sectors may not match the real number of sectors per track for the following reasons:

- An unreadable IDAM and sector data will not be included.
- Occasionally two copies of the sectors on a track are included. I believe this occurs when a 48tpi disk is read in a 96tpi drive when it reads the track from two head steppings. If the data read is different, both sets of sectors are used. This also possibly explains why occasionally apparent duplicate sectors are not marked.

## 5.2 Cylinder number (1 byte)

This field encodes the physical cylinder number (head position) for this track, in a range of 0-(#tracks on drive-1).

## 5.3 Side/Head number (1 byte)

This field encodes the disk side  (0 or 1)  that this track occurs on in it's lower bit.

The high bit of this field is  used  to  indicate  the  track  was recorded in single-density.  This allows mixed-density disks to be represented (FM on some tracks, and MFM on others).

I cannot confirm this,  but I suspect that the  FM bit in the header reflects the capabilities of the drive and not the recording density.

## 5.4 Cyclic Redundancy Check (1 byte)

This is the low 8 bits of the CRC (see below) calculated over the first three  bytes  of  the header and should match the forth byte.

Track headers and sector block lists occur until all tracks  on  the disk have been accounted for. When the last track record and  sector block list has been read,  a 255  (FF hex)  byte indicates the end of
the image.

# 6. Sector Header

Following the  Track  header  will  be  a  number  of  sector  blocks consisting of a sector header and optional  data  header/data  block. The number of sector blocks is indicated by the  "Number of  sectors"
field in the track header.

Each sector header has the following format:

```
Cylinder number             (1 byte)
Side/Head                   (1 byte)
Sector number               (1 byte)
Sector size                 (1 byte)
Flags                       (1 byte)
Cyclic Redundancy Check     (1 byte)
```

## 6.1 Cylinder number (1 byte)

This field indicates the logical cylinder number which is  written in the ID field of the disk  sector.  For  most  disk  formats  it  matches the Cylinder number indicated in the track header, however this  does  NOT  have  to  be  the  case  -  some  formats  encode non-physical cylinder numbers.

## 6.2 Side/Head (1 byte)

This field indicates the  logical  Side/Head  indicator  which  is  written in the ID field of the disk sector.  For most disk formats it matches the Side/Head number indicated  in  the  track  header, however this does NOT have to be the case -  some  formats  encode non-physical Side/Head numbers.

## 6.3 Sector number (1 byte)

This field indicates the logical sector number which is wrtten  in the ID field of the disk sector. Sector numbers do not have to be in any particular order  (the ordering of the  sectors determines the interleave factor of the track), do not necessarily begin at 0 or 1, and are not necessarily an unbroken series of numbers.  Some formats encode seemingly arbitrary sector numbers.

Teledisk sometimes creates bogus sectors headers to describe  data  that is not in a properly formatted sector or where the IDAM cannot be read.  These  extra  sectors appear to be created with sector numbers beginning at 100.

## 6.4 Sector size (1 byte)

Indicates the size of  the  sector,  according  to  the  following  table:

```
0 = 128 bytes           4 = 2048 bytes
1 = 256 bytes           5 = 4096 bytes
2 = 512 bytes           6 = 8192 bytes
3 = 1024 bytes          7 = 16384 bytes - not invalid but not sure if used
Sizes > 7 are treated as invalid
```

Note that disk formats exist which  have  different  sector  sizes  within the same track  and Teledisk will encode  them  this  way,  however many floppy  disk  controllers  cannot  format  such tracks.

## 6.5 Flags

This is a bit field indicating characteristics that Teledisk noted  about the sector when it  was recorded.   The  field  contain  the logical OR of the following hex byte values

```
01 = Sector was duplicated within a track
02 = Sector was read with a CRC error
04 = Sector has a "deleted-data" address mark
10 = Sector data was skipped based on DOS allocation [note]
20 = Sector had an ID field but not data [note]
40 = Sector had data but no ID field (bogus header)
```

Note:  Bit values 20H or 10H or Sector size > 8 indicate  that  NO  SECTOR  DATA  BLOCK FOLLOWS.

The meaning of some of these bits was taken  from  early  Teledisk documentation,  and may not be accurate. It is possible that 01H for duplicated sector is reserved for genuine duplicate sectors ids on a track. See section 5.1 as to why duplicates may appear that are not marked.

## 6.6 Cyclic Redundancy Check (1 byte)

This is the low 8 bits of the CRC (see below). If there is a sector data block, the CRC is calculated over the expanded sector, see section 7. Otherwise it is calculated over the first 5 bytes of the sector header.

Note. Previous documentation suggested that the CRC should have been calculated over the header and the expanded sector data. Internally Teledisk actually passes the CRC of the header as a third parameter to the CRC calculation of the expanded sector data, however the calculation routine ignores it.

# 7. Sector Data Header

The sector data header occurs following the sector header only when sector data is present. This is indicated by bits 10H and 20H of the Flags NOT being set and a sector size of < 8. When present it has the following format:

```
Data block size         (2 byte little endian word)
Encoded sector data     (Data block size bytes)
```

Sector headers and data blocks occur until all sectors for the track have been accounted for.

## 7.1 Data block size (2 byte little endian word)

This indicates the size of the encoded sector data block. Its content should expand to the expected sector size.

Note the max size supported is 4000h.

## 7.2 Encoded sector data (variable size)

The first byte of the encoded data sector determines how the data is encoded.

### 7.2.1 Raw sector data

The sector content is copied directly from the encoded sector data, it is encoded as:

```
0               (1 byte)
Sector data     (Data block size - 1)
```

### 7.2.2 Repeated 2-byte pattern

The sector content is generated as a repeated number of 2 byte values, it is encoded as:

```
1                   (1 byte)
Repeat count        (2 byte little endian word)
Repeat bytes        (2 bytes - copied as a byte pair)
```

Note, no check is done on whether there is residual data after the pattern.

### 7.2.3 Run Length Encoded data

The sector data is built up from concatenated fragments of RLE data each with two possible encodings. These fragments are processed until all of the Packed RLE data is used. It is encoded as:

```
2                   (1 byte)
Packed RLE data     (Data block size - 1)
```

#### 7.2.3.1 Raw Data Fragment

This is similar to raw sector data, except that the data length is one byte only and the data is appended to the sector, it is encoded as:

```
0                   (1 byte)
Length              (1 byte)
Sector data         (Length bytes)
```

#### 7.2.3.2 Repeated 2 byte pattern fragment

This is similar to repeated 2 byte pattern, except that the repeat count is one byte only and the data is appended to the sector, it is encoded as:

```
1                   (1 byte)
Repeat count        (1 byte)
Repeat bytes        (2 bytes)
```

# 8. Cyclic redundancy check

The error-checking 16 bit cyclic redundancy check is calculated with the polynomial value 0A097H using an input preset value of 0. Individual headers determine the data used in the calculation and whether the full 16 bits or low 8 bits are used.

# 9. Multi-volume support

If when reading the file, EOF is encountered, the specification supports continuation in another file. This file should have the same file name but with the last digit of the extent incremented by one.

The continuation file has its own file header which should have a valid CRC and the first four bytes must match the initial file header, other data is ignored. The check ensures that the compression and check byte are consistent. Data after this header is treated as a continuation of the data stream being processed when EOF was encountered.

# 10. Advanced Data Compression

There are two flavours of advanced compression used by Teledisk. The original "old" version uses a LZW compression scheme, but from file format v2.0 onwards it was replaced by a "new" version using the standard LZH compression scheme.

## 10.1 Old version

The LZW compression scheme used by Teledisk uses a fixed symbol size of 12 bits with table size of 4096, encoded in 6144 bytes. Data is divided into blocks, with a reset of the LZW tables for each new block. Each block is encoded as noted below.

```
Number of codes * 3 - stored as a little endian 16 bit word
12bit codes         - each pair of 12 bit codes is encoded as
                      code1 + (code2 >> 12) and stored
                      as a 24 bit little endian number
                      The last byte is omitted for an odd number of codes
```

The use of "number of codes * 3", simplifies the calculation of the bytes used for codes.

Note, Teledisk v2.23 appears to have dropped support for the "old" version of Advanced data compression.

## 10.2 New version

This compresses all data as a single block with LZH encoding with the string lookup buffer pre-set to all spaces (ASCII  20). The ring buffer size is 4096, the lookahead size is 60 and the threshold is 2. The frequency tables are rebuilt when the highest count is 8000H.

```
Updated 23-Sep-2024 - Mark Ogden
```