# H17Disk

## New H17 Hard-sectored Disk Image

# 1.0 Overview

This document documents a new format for Heathkit H17 Hard-sectored Disk Images. The current format being used, “H8D”, does not preserve the header information on the disk. It only saves the raw user data (256 bytes per sector). Since it does not store the 5 byte header for each sector; recreation of a disk requires additional information such as whether the disk is for HDOS or CP/M, and if it's an HDOS disk, what volume number is used, etc. Having this information is critical to transparently supporting different OS (HDOS vs. CP/M) disks in an emulator. It also allows properly writing the data onto a new physical disk without having to know details of the disk. The current H8D format requires the user to specify the volume number when creating a new physical disk from the image or forces the volume number to 0.

The new format is called H17Disk. On modern OSes, which support long file names, such as Linux, Windows and Mac OS X, the file extension is “.h17disk”. On DOS, or any other system which limit extensions to 3 bytes, the extension is “.h17”. Note: the file extension is not critical to the format; internal checks are available to make sure the file is a valid H17Disk.

## 1.1 File layout

The file starts with a 4 byte file tag “H17D”, a 3 byte version number, and a 0xFF byte. Starting at offset 8, the remainder of the file consists of blocks of data.

| Name | Size | Description |
|---|---|---|
| File Tag | 4 | ASCII “H17D” (0x48313744). |
| Version | 3 | The file format version. |
| 8-bit check | 1 | Fixed value of 0xFF to verify 8-bit clean |

**Table 1: File Header**

# 2.0 Detailed Description

## 2.1 Header

The file starts with the 4 bytes 'H17D'. This is the file's magic number, which helps programs to confirm they are working with a valid H17Disk file.

The next 3 bytes are the Version number. It starts with major number, then minor number, and finally point release. Major number will increase when there is a significant change or addition to the standard. The minor number is for minor revisions, or corrections. Point release changes are for clarifications to the standard that do not impact the functionality. These numbers are ASCII, for example version 3.2.1 would be stored as 0x33,0x32,0x31.

The program that created the file, should set this field to version of the spec, which it is compliant to.

The current version of the standard is 2.0.0.

The next byte is 0xFF, which allows verification that file has not been transferred over a 7-bit link which has cleared the high-bit of each byte. If the byte is not 0xFF, then the file has been irreparably damaged.

## 2.2 Blocks

The remainder of the file consists of Blocks with the general format being a 4 byte identifier. The block ID will be 4 ASCII characters. The next 4 bytes of the block is the length of the stored Block data, not including the 8 bytes of the header. The block data immediately follows the length bytes. The format of the block data is dependent on the Block ID type.

| Position | Description |
|---|---|
| 0-3 | Block ID |
| 4-7 | Length (big endian) |
| 8 | Block Data<br>- Dependent on Block ID type |
| ... | |
| 8 + Length | |

**Table 2: Block Format**

| Value | Name | Mandatory to Process if included | Required to be included | Description |
|---|---|---|---|---|
| DskF | Disk Format | Yes | Yes | General information about the disk (must be first block). |
| Parm | Parameters | Yes | No | Parameters for the disk |
| Date | Date | No | No | Date image was created |
| Imgr | Imager | No | No | Person who imaged the disk |
| Prog | Program | No | No | Program used to create the image |
| Padd | Padding | No | No | Used to ensure H8D block data starts at byte 256 in the file. |
| H8DB | H8D Block | No | Yes | Raw data bytes, which are aligned and only contains the sector data. This is the same as the current H8D format. |
| SecM | Sector Metadata | No | No | Shows if sector was imaged successfully, and any errors, where location is located in the H8D block, and header and checksum bytes. |
| Labl | Label | No | No | Information from the physical label on the outside of the disk. Free-form Text. |
| Comm | Comment | No | No | Comment for the Disk, Free-form Text. |

**Table 3: Block ID Type Values**

### 2.2.1 Disk Format Block

The Disk Format Block specifies the number of sides and tracks for the image. Other drive characteristics are fixed for Heath hard-sectored disks. This include:

| Drive Characteristic | Value |
|---|---|
| Sectors per Track | 10 |
| Data bytes per sector | 256 |
| RPM | 300 |
| Encoding | FM |
| Bit rate | 250 bps |

**Table 4: Fixed disk parameters**

All Heath hard-sectored disks, use FM encoding, 10 sectors per track, 256 bytes per sector, and 300 RPM.

| Description | Size (bytes) | Allowed Values | Default Value |
|---|---|---|---|
| Block ID | 4 | DskF | - |
| Block Length | 4 | 2 – 3 | - |
| Sides | 1 | 1 – 2 | 1 |
| Tracks | 1 | 40 or 80 | 40 |
| R/O | 1 | 0 – false / 1 – true | 0 – false |

**Table 5: Disk Format Block**

Sides – Number of recorded sides of data on the disk.

Tracks – Number of tracks per side

R/O Flag (boolean) - This signifies if the disk is write protected. If the value is 1, the disk is write protected. If 0, writes are allowed.

### 2.2.2 Parameters Block

The Parameters Block is an optional block in the file.

The parameters block currently defines 2 fields.

Fields are defined in the order specified in this document. New fields will be added at the end of the list.

| Description | Size (bytes) | Type | Allowed Values | Default |
|---|---|---|---|---|
| Block ID | 4 | | Parm | |
| Block Length | 4 | | 0-2 | |
| Distribution Disk | 1 | Value | 0-3 | 0 |
| Source of Header Data | 1 | Value | 0-5 | 0 |

**Table 6: Parameter Block**

Distribution Disk(value) - This flag signifies if the image was created from an original distribution disk.

| Value | Meaning |
|---|---|
| 0 | Status of source disk unknown. |
| 1 | Image was created from an original distribution disk. |
| 2 | Image was not an original distribution disk. |
| 3 | Image was created from a disk, which appears to be a copy of a distribution disk. |
| 4-127 | Reserved. |

**Table 7: Distribution Disk Flag**

Source of Header data(value) - This flag determines if the headers for the image were save from the actual disk or if it was created by a utility.

| Value | Meaning |
|---|---|
| 0 | Generated by an H8D to H17Disk conversion utility |
| 1 | Created by an emulator |
| 2 | Captured on an H8/H89 |
| 3 | Captured with an FC5025 |
| 4 | Captured by AppleSauce |
| 5-127 | Reserved. |

**Table 8: Source of Full Track Data Flag**

### 2.2.3 Label Block

The Label Block is an optional block in the file, but highly recommended. The block is free-form text, describing the label of the disk.

| Description | Size (bytes) | Value |
|---|---|---|
| Block ID | 4 | Labl |
| Length | 4 | |
| Text | <Length> | <Free-form text> |

**Table 9: Label Block**

### 2.2.4 Comment Block

The Comment Block is an optional block in the file, but highly recommended. The block is free-form text, allowing the user to make any comments about the disk. The user could include such things as the brand of the disk, how the disk was acquired, and any other details they want to record.

| Description | Size (bytes) | Values |
|---|---|---|
| Block ID | 4 | Comm |
| Length | 4 | |
| Text | <Length> | <Free-form text> |

**Table 10: Comment Block**

### 2.2.5 Date Block

The Date Block is an optional block in the file, that specifies the date/time when the image was generated. This should be in ISO 8601 format.

| Description | Size (bytes) | Values |
|---|---|---|
| Block ID | 4 | Date |
| Length | 4 | |
| Text | <Length> | <Should be in ISO 8601 format> |

**Table 11: Date Block**

### 2.2.6 Imager Block

The Imager Block is an optional block, which specifies the person who imaged the disk. They can provide as much detail as they wish, such as, name, email, website, phone number, etc. Or they may choose to provide no information, and have this block not included in the file.

| Description | Size (bytes) | Value |
|---|---|---|
| Block ID | 4 | Imgr |
| Length | 4 | |
| Text | <Length> | <Free-form text> |

**Table 12: Imager Block**

### 2.2.7 Program Block

The Program Block is an optional block, which specifies the program used to generate the file. This can be free-form text, with the program name, OS, version, and reference information (such as website to find the program).

| Description | Size (bytes) | Value |
|---|---|---|
| Block ID | 4 | Prog |
| Length | 4 | |
| Text | <Length> | <Free-form text> |

**Table 13: Program Block**

### 2.2.8 H8D Data Block

To make the transition to the new format easier, this block allows existing applications which are currently using H8D images, to use this block with very little code changes. Sectors are ordered in logical order. The data portion of this block must start at file offset 256. Working backward, this means this block must start at file offset 248.

The H8D block is just the raw data blocks of in the same format as existing H8D files. The blocks start with Side 0, Track 0, Sector 0.

| File Offset | Description | Size (bytes) | Value |
|---|---|---|---|
| 248 | Block ID | 4 | H8DB |
| 252 | Length | 4 | <should be a multiple of 256> |
| 256 | Data | <Length> | <sector data> |

**Table 14: H8D Data Block**

### 2.2.9 Sector metadata

| Description | Size (bytes) | Values |
|---|---|---|
| Block ID | 4 | SecM |
| Length | 4 | |
| Metadata | <Length> | |

This block contains important information about each sector of the disk. The information for each sector is 16 bytes long. The information include the data needed to place the sectors in the original physical order, header/checksum bytes, and any error status for the sector. The sector metadata is laid out in physical order as imaged from the disk. For single-sided images, it starts at track 0, sector 0, continues in order for each sector on track 0. It continues in the same way for each track 1 through 39 (or 79 for 80 track images). For double-sided disks, for each track, it will have both sides alternating. First side 0, track 0, sectors 0 to 9, then side 1, track 0, sectors 0 to 9. This continues for each track (up to 39 or 79).

| Byte | |
|---|---|
| 0 – 3 | Offset to data block |
| 4 | Sector Status |

| | |
|---|---|
| 5 | Header Sync Byte (0xFD) |
| 6 | Volume |
| 7 | Track |
| 8 | Sector |
| 9 | Header Checksum |
| 10 | Data Sync Byte (0xFD) |
| 11 | Data Checksum |
| 12 – 13 | Valid Bytes in the block |
| 14 – 15 | Reserved |

#### 2.2.9.1 Offset to data block

The offset (from the start of the file) to the 256 byte sector data in the H8D block. This allows recreating the correct physical layout of the track (i.e. the interleave).

#### 2.2.9.2 Sector Status

Sector status is used to indicate whether or not the sector had any errors during the imaging.

| **Bit** | **Meaning** |
|---|---|
| 0 | Missing Header Sync |
| 1 | Wrong Track |
| 2 | Invalid Sector |
| 3 | Invalid Header Checksum |
| 4 | Missing Data Sync |
| 5 | Invalid Data Checksum |
| 6 | Sector was unreadable |
| 7 | Reserved |

#### 2.2.9.3 Header Sync Byte

The sync byte for the start of the header block. This should be 0xFD for all valid Heathkit hard-sectored disks.

#### 2.2.9.4 Volume

Volume number recorded in the header block. For Heathkit disks, volume number for track 0 is always 0. CP/M always uses a volume number of 0. For HDOS disks, track number for tracks great than track zero can be set to any value between 0 and 255. It should match the track number specified in the Label sector of track 0.

#### 2.2.9.5 Track

Track number recorded in the header block

#### 2.2.9.6 Sector

Sector number recorded in the header block.

#### 2.2.9.7 Header Checksum

Checksum recorded in the header block

#### 2.2.9.8 Data Sync Byte

The sync byte for the start of the data block. This should be 0xFD for all valid Heathkit hard-sectored disks.

#### 2.2.9.9 Data Checksum

Checksum recorded in the data block.

#### 2.2.9.10 Valid Bytes in the Block

A value between 0 and 256, signifying the number of bytes in the data block that appeared to have proper clock bits. (Some disk imaging setups will be able to detect where bad data starts in a sector due to errors in the expected clock bits.)

#### 2.2.9.11 Reserved