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. 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. 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.

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.

As progress is made on imaging soft-sectored disks, the current plan is to have 2 additional disk formats similar to this for both 5 1/4” soft-sectored and 8” soft-sectored disks. Although, the pros and cons will need to be reviewed on whether to create a new format, or use one of the established formats. If created, the new formats will be called H37Disk and H47Disk. When/If emulation of the Z-89-67 is implemented, a new H67Disk will also be created to emulate the 10M Winchester drive.

1.1 File layout

The file starts with a 4 byte file tag “H17D” and 3 byte version number. Starting at offset 0x07, the remainder of the file consists of blocks of data. Each data block should appear in numerical order.

Name	Size	Required	Description
File Tag	4	Yes	ASCII “H17D” (0x48313744).
Version	3	Yes	The file format version.
Disk Format Block	Variable	No	General information about the disk such as Heads and tracks
Parameters Block	Variable	No	Parameters for the disk.
Comment Block	Variable	No	Free-form text describing the disk.
Data Block	Variable	Yes	The actual data from the disk.
Hole Block	Variable	No	Specifies Index/Sector hole information for the disk.
Raw Data Block	Variable	No	Raw data including timing bits, (as recorded from the FC5025 USB Floppy disk controller).
Undefined	Variable	No	Any other blocks which may be added in the future.

Table 1: Current File Layout

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.

In the future, H37Disk and H47Disk formats will have their own magic number, in order to make the which format easy to be determine.

The next 3 bytes are the Version number. The program that created the file, should set this field to version of the spec, which it is compliant to. Note: the decoder must not use the version field to determine if it can decode the file. It should only fail to decode a file, if it encounters an unknown block in which the “Mandatory” flag is set to True. Future extensions to this format, will try to be added in a way to maintain backward compatibility with older decoders.

2.2 Blocks

The remainder of the file consists of Blocks with the general format being a 1 byte identifier. The next byte contains the flags for this block. Currently, it only includes the mandatory flag. The next 4 bytes of the block is the length of the stored Block data, not including the 6 bytes of the header. After that the following 'Length' bytes contain the data for the block. The format of the block data is dependent on the Block ID type.

Position	Description
0	Block ID
1	Flags
2-5	Length
6	Block Data - Dependent on Block ID type
...	
5 + Length	

Table 2: Block Format

Block ID is a full byte, allowing for up to 256 different types of blocks. Although, only 5 are currently defined.

Bit	7	6	5	4	3	2	1	0
Description	M	X	X	X	X	X	X	X

Table 3: Flag Layout

M bit, Bit 7 (0x80), signifies if decoding and processing the block is mandatory to properly decode the file. This does not necessarily imply that the block is required in all image files. This flag is to allow new blocks to be added and flagged as optional if it doesn't impact the readability of the core part of the file. A 0 value means processing of this block is optional and can be ignored by older decoders, or even current decoders that do not care about the information. A value of 1 means that the decoder must be able to process the Block. If it can not process the block, it should report an error and abort the read of the file.

Bits 0-6 are currently unused and reserved. The should be set to 0 on write, and ignored on read.

Value	Name	Mandatory to Process if included	Required to be included	Description
0x00	Disk Format	Yes	No	General information about the disk such as Heads, tracks, encoding (FM/MFM).
0x01	Parameters	Yes	No	Parameters for the disk
0x02	Comment	No	No	Comment for the Disk – Free-form Text. Typically, what is on the disk label.
0x10	Data Block	Yes	Yes	The actual data from the disk
0x20	Hole Block	Yes	No	Specifies Index/Sector hole information for the disk.
0x30	Raw Data Block	No	No	Stores the raw data retrieved from disk imaging, this includes both data and clock bits.
All other values	Reserved	Determined by the Mandatory Flag.	No	Note: Any new IDs need to be reserved to avoid potential conflict.

Table 4: Block ID Type Values

2.2.1 Disk Format Block

The Disk Format Block describes the general layout of the physical disk and how the data is stored in the Data block. If this block is not included, the decoder should assume the default values listed in this table. If this block is included, the decoder must process it. This block may also be shorten, if the remaining fields should use the default values. Since only fields 1 through 3 are allowed a different value, there is no value to include fields 4 through 8.

Description	Size (bytes)	Allowed Values	Allowed Values for H17D	Default Value for H17D
Format Block ID	1	0x00	0x00	
Flags	1	0x80 (mandatory)	0x80	
Block Length	4	0x02	0x01 - 0x02	
Sides	1	1 – 2	1, 2	1
Tracks	1	Should be 40 or 80	40, 80	40
The following parameters are fixed for the H17D format and shall not be included in the file.				
Disk Size	1	0x00 – 8” 0x01 – 5.25” 0x02 – 3.5”	0x01	0x01
Sectoring	1	0 – Hard-Sectored 1 – Soft-Sectored	0	0
RPM	1	0x00 – Unknown 0x01 – 300 RPM 0x02 – 360 RPM	0x01	0x01
Encoding	1	0x00 – Unknown 0x01 – FM 0x02 – MFM	0x01	0x01
Bit Rate	1	0x00 – Unknown 0x01 – 250 bps 0x02 – 500 bps	0x01	0x01

Table 5: Disk Format Block

2.2.2 Parameters Block

The Parameters Block is an optional block in the file. But if the block is present, decoders are required to process the block. Although the block is mandatory, individual values may not be. In fact, none of the current fields are mandatory. The decoder only has to handle fields with the mandatory bit set. The values may be boolean or values. A bool only uses bit 0. A value flag uses bits 0-6. For both types, bit 7 is defined to be the Mandatory flag. If the value is 1, the decoder must understand and decode the value. A value of 0, implies that the image can be correctly decoded without the decoder understanding the flag.

The parameters block currently defines 3 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	Mandatory	Allowed Values	Default
Format Block ID	1			0x01	
Flags	1			0x80 (mandatory)	
Block Length	4				0x00 0x00 0x00 0x03
R/O Flag	1	Bool	No	0/1	0
Distribution Disk	1	Value	No	0-2	0
Source of Track Data	1	Value	No	0-4	0

Table 6: Parameter Block

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.

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.
3-127	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. A value of 1 means that the header data (plus data block sync and checksum) was captured from the actual disk.

Value	Meaning
0	Generated by an H8D to H17Disk conversion utility
1	Created by an emulator
2	Captured on an H89
3	Captured with an FC5025
4-127	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)	Allowed Values	Default
Format Block ID	1	0x02	
Flags	1	0x00 (not mandatory)	
Block Length	4		<none>
Text	<length>	0/1	0

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)	Allowed Values	Default
Format Block ID	1	0x03	
Flags	1	0x00 (not mandatory)	
Block Length	4		
Text	<length>	Free-form text	<empty>

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.

Description	Size (bytes)	Allowed Values	Default
Format Block ID	1	0x04	
Flags	1	0x00 (not mandatory)	
Block Length	4		
Text	<length>	Free-form text	<empty>

Table 11: Comment 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)	Allowed Values	Default
Format Block ID	1	0x05	
Flags	1	0x00 (not mandatory)	
Block Length	4		
Text	<length>	Free-form text	<empty>

Table 12: Comment 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 (website to find the program).

Description	Size (bytes)	Allowed Values	Default
Format Block ID	1	0x06	
Flags	1	0x00 (not mandatory)	
Block Length	4		
Text	<length>	Free-form text	<empty>

Table 13: Comment Block

2.2.8 Data Block

The data block section stores data for each sector of the disk. This data has been processed so that the data is byte-aligned. This

		Description	Size (bytes)	Value
Block Header		Data Block ID	1	0x10
		Flags	1	0x80 (mandatory)
		Block Length	4	
Track information repeats 40 or 80 times (depending on the tracks field in the Disk Format Block).		Track Sub-block ID	1	0x11
		Head Number	1	0/1
		Track Number	1	0-39/0-79
		Track Length	2	3250
	Sector information repeats 10 times per track.	Sector Sub-block ID	1	0x12
		Sector Number (physical sector on the disk).	1	1-10
		Error Status	4	See description below.
		Sector Length	2	Should be 320
		Data (Byte aligned)	320	Dump of the physical sector

Table 14: Data Block

2.2.8.1 Error Status

This field specifies any errors that occurred while imaging and decoding the raw sector data. The error values are bit encoded for this sector.

Value	Meaning
0	No error.
0x01	Read Error – Problem with actual read from the disk. (No data available).
0x02	Invalid clock bits – Unable to properly extract data bits from the raw image.
0x04	Missing header sync – Sync byte was not seen for the header.
0x08	Wrong track – Track number in header does not match what was expected.
0x10	Invalid Sector – Sector value is not in the range of [1..10].
0x20	Invalid header checksum – Validation of header checksum failed.
0x40	Data sync missing – No data sync byte found.
0x80	Invalid data checksum – Validation of data checksum failed.
Bits 9-32	Reserved.

Table 15: Decoding Errors

2.2.9 Raw Data Block

The data block section stores full tracks of data, this includes both clock and data bits. The goal of this section, is to preserve everything possible from the disk and not have to ever image this disk again.

	Description	Size (bytes)	Value
Block Header	Data Block ID	1	0x20
	Flags	1	0x00 (not mandatory)
	Block Length	4	
Track information repeats 40, 80, or 160 times (depending on the tracks field in the Disk Format Block).	Track Sub-block ID	1	0x21
	Head Number	1	0/1
	Track Number	1	0-79
	Track Length	4	6440-?[1]
Sector information is included at least once for all 10 sectors of the track. It can repeat many more time if there are there are errors during the decoding of the raw data. All attempts at reading the disk should be stored, so that information which may prove useful in post-analysis is preserved.	Sector Sub-block ID	1	0x22
	Sector Number (physical sector on the disk).	1	1-10
	Sector Length	2	640
	Raw data	640	Dump of the physical sector

Table 16: Raw Disk Format Block

All sectors read from the physical disk should be saved into this block, whether or not decoding of the block was successful. This is to preserve as much of the data as possible, for further analysis and recovery.

1 The minimum size is 6440, but could be MUCH larger if read errors caused sectors to be re-read. All sector reads should be stored in the file, not just the last, or one that decodes properly.

2.2.10 Hole Block

The Hole Block contains information specifying where holes are located on the disk. The encoding is a modified Run-Length encoding. The data specifies the status of the hole detect for each byte of in the Data Block. The first byte in this block specifies the number of byte of data that have the hole detect status set. The next 16 bits specify the number of data bytes which does not have the hole detect status set. Either of these values can be zero, if more bytes have the same type of status, than fit in the value. This repeats until all the bytes on a given track has been specified. Based on documentation hole are 4 mSec, which equates to 64 bytes of data. Then about 12 mSec of "Not Hole", which equates to 256 bytes. Due to the index hole, the last sector, would have the 64 byte Hole, 96 "no hole", 64 bytes of Hole, and finally 96 bytes of no hole. The layout is the same as the Data Block with respect to tracks and sides.

Description	Size (bytes)	Allowed Values
Format Block ID	1	0x20
Flags	1	0x00
Block Length	4	Variable, see text.
Head	1	0/1
Track	1	0-39/0-79
Hole Data	Length	See text.

Table 17: Hole Block

For the H17 format, the default follows that pattern, Table 18 shows the values an emulator should use if this block is not included in the file.

Description	Size (bytes)	Values
Format Block ID	1	0x20
Flags	1	0x00
Block Length	4	0x0021 (33)
Disk Data	33	0x40 (64), 0x0100 (256), 0x40 (64), 0x0100 (256), 0x40 (64), 0x0100 (256), 0x40 (64), 0x0100 (256), 0x40 (64), 0x0100 (256), 0x40 (64), 0x0100 (256), 0x40 (64), 0x0100 (256), 0x40 (64), 0x0100 (256), 0x40 (64), 0x0100 (256), 0x40 (64), 0x0060 (96), 0x40 (64), 0x0060 (96)

Table 18: Hole Block

3.0 Possible future features

One of the goals for the new format is to make it easy for people to add new features without breaking backward compatibility. I haven't spent much time thinking of new ideas, but one of the features I would like to add (once the core format has been defined and implemented) is a way for the emulator to keep multiple versions of a disk encoded in one file.

3.1 Multiple disk version

When using disk images, even though it's possible to copy the images and use the copies, it would be nice to have some internal versioning so that the user has the flexibility to roll-back the disk image to an early state. By keeping the latest version in the standard Data Block, older programs will still work with these files. The older versions of changed sectors will be stored in a non-mandatory newly defined block. New programs/utilities that want to take advantage of the versioning information will read this new block and merge with the data from the Data Block to recreate older versions. There can be a tool to allow the user manipulate the versioning information. Some options for the tool would be to roll back to a specific version, delete specific versions, and delete all old versions.