Emulation: DMK Format Documentation

Introduction

This page deals with TRS-80 software stored in David Keil's disk format.
Currently, files with a .DSK extension could be JV1, JV3, or DMK format. The DMK format is a format developed by David Keil which allows for the storing of TRS-80 disks in pure form, which would allow for the representation of copy protected disks into emulator images. The DMK format gives the ability to support any FM or MFM encoded format that is reasonably close to the IBM 3740 or IBM System 34 standards.
DMK format disks are easy to discern because of its 16 byte header. David Keil's Emulator, and Tim Mann's XTRS (post v3.7) will read the DMK format.
This page is made to prevent the fact that users of other emulators may become frustrated with intermixed files. Users of Tim's's or Dave's emulator can download and run any version but other emulator users whould have to be sure what version they download to be sure they have a copy that will work with their emulator.

Details of the DMK Format(*)

(*)Reprinted from Dave Keil's Site with Permission and modified in accordance with information from Tim Mann

This information on what has become known as the DMK virtual disk format is provided for users wanting to better understand the operation of the emulator and for programmers writing their own emulators wanting to add support for this format and/or the creation of PC utilities to work with the DMK virtual disk format.
This virtual disk format is as close to the way data on a real disk is stored as possible. There is very little added overhead and the data is easily examined and edited using PC based hex editors. The actual design is really quite simple and enables support of ALL the WD-17xx controller functions and formats. While the design is simple however the programming requirements for this format are much more extensive then for the JV1/JV3 formats.

Disk header:

Virtual disks have a 16 byte disk header which is initialized when the user creates a new virtual disk. This header may be modified before or after a virtual disk has been formatted to change some of its characteristics.

Byte 0	If this byte is set to FFH the disk is `write protected', 00H allows writing.
Byte 1	Number of tracks on virtual disk. Since tracks start at 0 this value will be one greater than the highest track written to the disk. So a disk with 40 tracks will have a value of 40 (28H) in this field after formatting while the highest track written would be 39. This field is updated after a track is formatted if the track formatted is greater than or equal to the current number of tracks. Re-formatting a disk with fewer tracks will NOT reduce the number of tracks on the virtual disk. Once a virtual disk has allocated space for a track it will NEVER release it. Formatting a virtual disk with 80 tracks then re-formatting it with 40 tracks would waste space just like formatting only 40 tracks on an 80 track drive. The emulator and TRS-80 operating system don't care. To re-format a virtual disk with fewer tracks use the /I option at start-up to delete and re-create the virtual disk first, then re-format to save space. Note: This field should NEVER be modified. Changing this number will cause TRS-80 operating system disk errors. (Like reading an 80 track disk in a 40 track drive)
Byte 2 & 3	This is the track length for the virtual disk. By default the value is 1900H, 80H bytes more than the actual track length, this gives a track length of 6272 bytes. A real double density track length is aprox. 6250 bytes. This is the default value when a virtual disk is created. Values for other disk

	and format types are 0CC0H for single density 5.25• floppies, 14E0H for single density 8• floppies and 2940H for double density 8• floppies. The max value is 2940H. For normal formatting of disks the values of 1900H and 2940H for 5.25• and 8• are used. The emulator will write two bytes and read every second byte when in single density to maintain proper sector spacing, allowing mixed density disks. Setting the track length must be done before a virtual disk is formatted or the disk will have to be re-formatted and since the space for the disk has already been allocated no space will be saved. WARNING: Bytes are entered in reverse order (ex. 2940H would be entered, byte 2=40, byte 3=29). Note: No modification of the track length is necessary, doing so only saves space and is not necessary to normal operation. The values for all normal 5.25• and 8• disks are set when the virtual disk is created. DON'T modify the track length unless you understand these instructions completely. Nothing in the PC world can be messed up by improper modification but any other virtual disk mounted in the emulator with an improperly modified disk could have their data scrambled.
Byte 4	Virtual disk option flags. Bit 4 of this byte, if set, means this is a single sided ONLY disk. This bit is set if the user selects single sided during disk creation and should not require modification. This flag is used only to save PC hard disk space and is never required. Bit 6 of this byte, if set, means this disk is to be single density size and the emulator will access one byte instead of two when doing I/O in single density. Double density can still be written to a single density disk but with half the track length only 10 256 byte sectors can be written in either density. Mixed density is also possible but sector timing may be off so protected disks may not work, a maximum of 10 256 byte sectors of mixed density can be written to a single density disk. A program like "Spook House" which has a mixed density track 0 with 1 SD sector and 1 DD sector and the rest of the disk consisting of 10 SD sectors/track will work with this flag set and save half the PC hard disk space. The protected disk "Super Utility + 3.0• however has 6 SD and 6 DD sectors/track for a total of 12 256 byte sectors/track. This disk cannot be single density. This bit is set if the user selects single density during disk creation and should not require modification. This flag is used only to save PC hard disk space and is never required. Bit 7 of this byte, if set, means density is to be ignored when accessing this disk. The disk MUST be formatted in double density but the emulator will then read and write the sectors in either density. The emulator will access one byte instead of two when doing I/O in single density. This flag was an early way to support mixed density disks it is no longer needed for this purpose. It is now used for compatibility with old virtual disks created without the double byte now used when in single density. This bit can be set manually in a hex editor to access old virtual disks written in single density.
Byte 5-B	reserved for future options

Byte C-F	Must be zero if virtual disk is in emulator's native format. Must be 12345678h if virtual disk is a REAL disk specification file used to access REAL TRS-80 floppies in compatible PC drives.
Track Header	Each track has a 128 (80H) byte header which contains an offset to each IDAM in the track. This is created during format and should NEVER require modification. The actual track data follows this header and can be viewed with a hex editor showing the raw data on the track. Modification should not be done as each IDAM and sector has a CRC, this is just like a real disk, and modifying the sector data without updating the CRC value will cause CRC errors when accessing the virtual disk within the emulator. Note: Modification within MSDOS could however be done to emulate a protected disk in the TRS-80 emulator.

Track header:

Each side of each track has a 128 (80H) byte header which contains an offset pointer to each IDAM in the track. This allows a maximum of 64 sector IDAMs/track. This is more than twice what an 8 inch disk would require and 3.5 times that of a normal TRS-80 5 inch DD disk. This should more than enough for any protected disk also.

These IDAM pointers MUST adhere to the following rules.

- Each pointer is a 2 byte offset to the FEh byte of the IDAM. In double byte single density the pointer is to the first FEh.
- The offset includes the 128 byte header. For example, an IDAM 10h bytes into the track would have a pointer of 90h, 10h+80h=90h.
- The IDAM offsets MUST be in ascending order with no unused or bad pointers.
- If all the entries are not used the header is terminated with a 0000h entry. Unused entries must also be zero filled.
- Any IDAMs overwritten during a sector write command should have their entry removed from the header and all other pointer entries shifted to fill in.
- The IDAM pointers are created during the track write command (format). A completed track write MUST remove all previous IDAM pointers. A partial track write (aborted with the forced interrupt command) MUST have it's previous pointers that were not overwritten added to the new IDAM pointers.
- The pointer bytes are stored in reverse order (LSB/MSB).
- Each IDAM pointer has two flags. Bit 15 is set if the sector is double density. Bit 14 is currently undefined. These bits must be masked to get the actual sector offset. For example, an offset to an IDAM at byte 90h would be 0090h if single density and 8090h if double density.

Track data:

The actual track data follows the header and can be viewed with a hex editor showing the raw data on the track. If the virtual disk doesn't have bits 6 or 7 set of byte 4 of the disk header then each single density data byte is written twice, this includes IDAMs and CRCs (the CRCs are calculated as if only 1 byte was written however). The IDAM and sector data each have CRCs, this is just like on a real disk.

Modification should not be done since doing so without updating the CRCs would cause data errors. Modification could be done however to create protected tracks for importing protected disks to virtual disk format. Examples of disks created using this technique are "Super Utility+ 3.0• and "Forbidden City".