HardMaster

By Colin Christensen

Sector Editors Compared

	Disk Patch (TI)	Disk Utilities (J. Birdwell)	Sector One (R. Moore)	HardMaster (Asgard)
SECTOR FUNCTIONS				
Read	Yes	Yes	Yes	Yes
Edit	Yes	Yes	Yes	Yes
Display in ASCII or Hex	Yes	Yes	Yes	Yes
Step through	Yes	Yes	No	Yes
Write	Yes	Yes	Yes[1]	Yes
Output to Printer	No	Yes	Yes	Yes
Print to Disk	No	Yes	Yes	Yes
Compare	No	No	Yes	No
Copy	No	Yes[2]	Yes	Yes[2]
Search for Text	No	Yes	Yes	Yes
Sector 0 Mark	No	Yes	No	No
Sector 0 Free	No	Yes	No	No
FILE FUNTIONS				
Search	Yes	Yes	No	Yes
DISK FUNCTIONS				
Floppy Catalog	Yes	Yes	No	Yes
Floppy Directory	No	No	No	Yes
Hard Disk Tree Directory	No	No	No	Yes
4-SECTOR AT ONCE EDITOR				
Edit Screen	No	No	No	Yes
Print Screen	No	No	No	Yes
COMPATIBILITY				
Floppy	Yes	Yes	Yes	Yes
Hard Drive	No	No	Yes	Yes
40 Column display	Yes	Yes	Yes	Yes
80 Column display	No	No	Yes	Yes

Note 1. Sector One does not prompt you to see if you are sure before writing back a sector - a potential risk because memory buffer and sector number isn't always the same.

Note 2. Can only copy between two disks if you choose the same sector number

Source: Sector Editor, Jan Alexandersson, Programbiten Nitlinian

HardMaster, by Colin Christensen, is a modern sector editor for the TI-99/4A and Myarc Geneve 9640. This powerful program doesn't force you to make choices like some other editors do - you can edit sectors on floppy disks or hard disk drives, you can edit them one at a time or four at a time, in 40 columns or in 80-columns, on a 99/4A or on a Geneve. *HardMaster* is also relatively complete, with a wide range of functions - some found no-where else. Hard-drive users will appreciate the "tree directory" which makes tracking files to sub-directories simpler, or the ability to copy sectors from one part of the hard drive to another, and all users will find the "quad editor" useful, or that printing out the sector bitmap and the 'pack' command make restoring a bad disk easier. We like to think *HardMaster* is the best of the bunch. But don't just take our word for it - look over the comparison list between four popular sector editors compiled by a respected user group in Sweden and make up your own mind.

If you need a disk sector editor, for floppy disks or hard drives, remember *HardMaster.*

Suggested Retail
$14.95

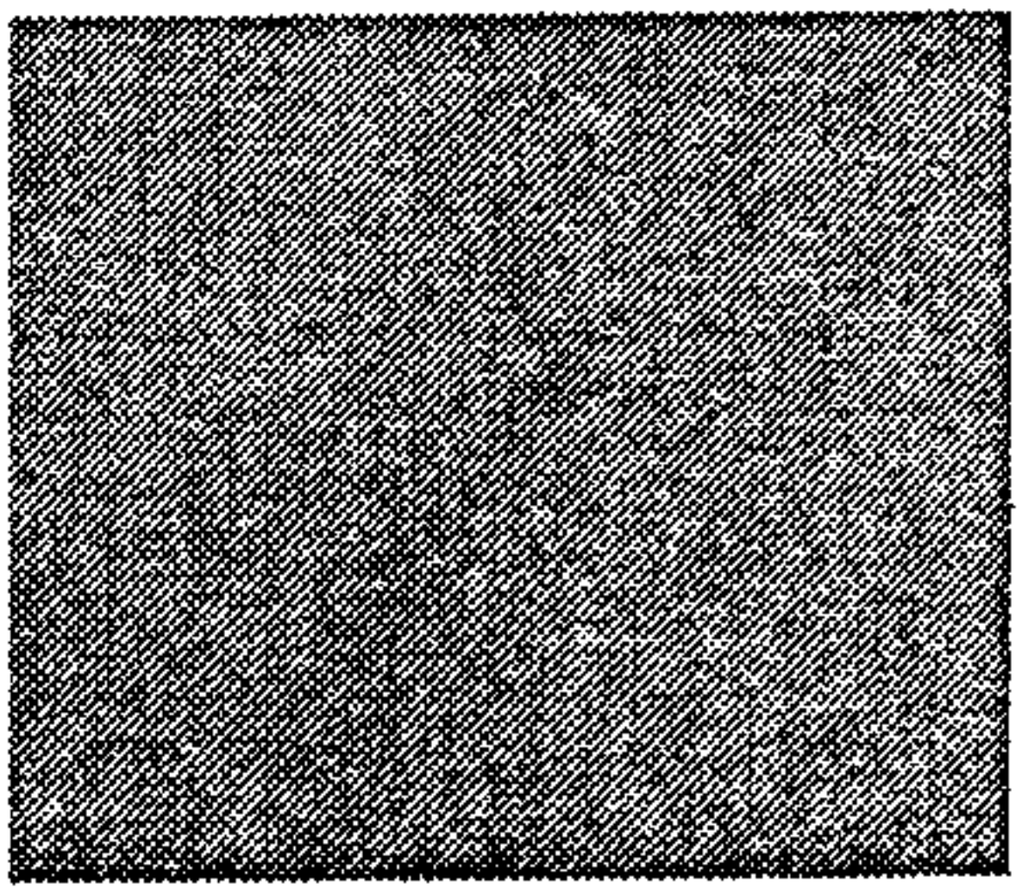
U.S. - Add $1.00 S&H
Canada - Add $1.50 S&H
Airmail - Add $3.00 S&H

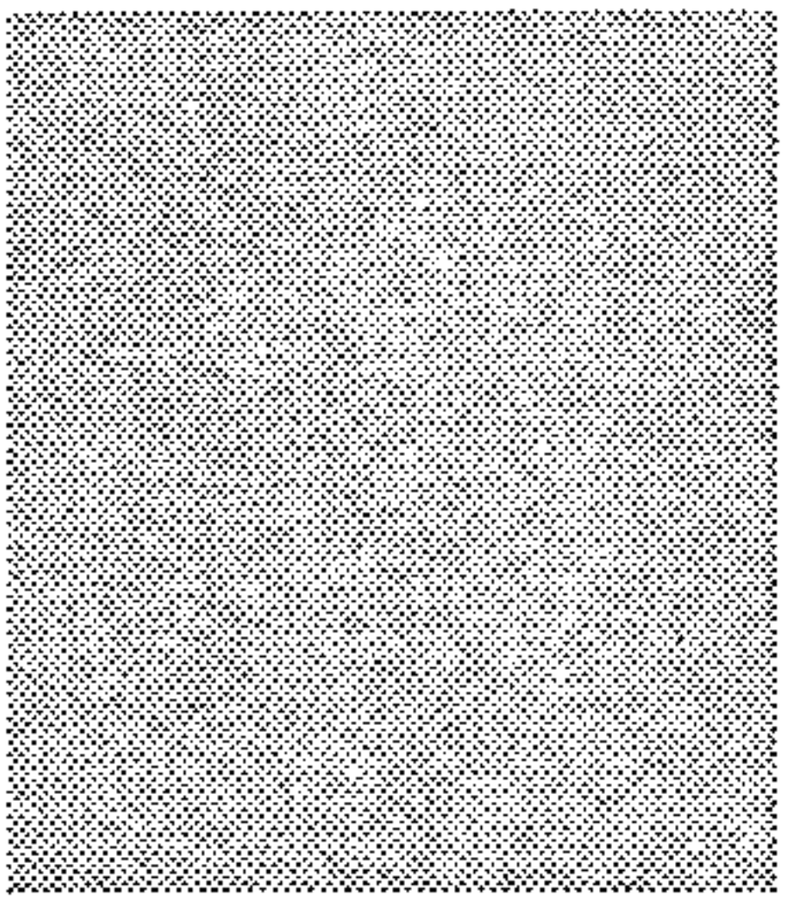
Asgard Software • P.O. Box 10306 • Rockville, MD 20849 • (703)255-3085

Also Available from your local Asgard Software dealer - Call or Write for a free catalog

Hard Master

"The Sector Editor for the Myarc Hard and Floppy Disk Controller"





By Colin Christensen

Asgard Software

P.O. Box 10306 - Rockville, MD 20850

HARDMASTER

by Col Christensen

HARDMASTER is a sector editor and manager for use with the Myarc Hard-Floppy Disk Controller (HFDC) card. Two versions (2.04 and 2.08) are provided - they differ in that the latter runs on an 80 column screen while the former uses the 40 column screen.

This program is the most capable and complete program of its type for use with the HFDC. However, it is not for the faint of heart and requires at least a cursory knowledge of how disks are used by the computer. If you don't have this knowledge, however, you shouldn't attempt to use this type of program anyway because of the potential danger to your data.

ON READING THIS MANUAL

This manual is arranged in four parts. The first part is devoted to the use of this program, it's specific functions, and on using disk sector editors in general. The second part is devoted to an explanation of how the HFDC stores data on a hard-drive. Part three provides a complete overview of floppy disk storage techniques. Finally, the last section provides information for programmers and hardware enthusiasts on how disk drives are interfaced to the 99/4A in general and the HFDC in particular.

When reading this manual, all words that are underlined are command key presses used by this program. When the key suggested is pressed a command is executed. Any text in italics before a command key should be typed prior to pressing the command key for the action described.

GETTING STARTED

HARDMASTER is program-image assembly program that loads quickly using Editor/Assembler option 5 type loaders. To use the program one needs an expanded TI system or 9640 Geneve, the Myarc Hard Floppy Disk Controller Card and at least one hard drive or floppy drive. A printer is useful but not necessary.

LOADING

On the TI-99/4A

Place one of the following modules in the cartridge port, turn on your computer, and follow the instructions below.

From Editor/Assembler:

Place the program disk in drive one, select Option #5 (Run Program File) from the main menu, enter the filename *DSK1.HM* and press ENTER.

From TI-Writer:

After placing the program disk in drive one select the Utility option of TI-Writer (Option #3), type the filename *DSK1.HM* and press ENTER.

The program can be stored on hard disk or RAM-disk and loaded speedily by any of the above loaders or from a ROS menu or a FunnelWeb loader screen option.

On the Myarc Geneve 9640

Before you can load the program you have to turn on your computer and boot M-DOS. After M-DOS is up and running, you have a variety of loading options. Geneve users also have the option of using an 80-column version also provided on the program disk. This version may function on a TI-99/4A with an 80-column card, but it is not guaranteed.

From EXEC:

The easiest way to load the program is with EXEC, by Barry Boone. Place a disk containing EXEC in one drive (say on drive A), and the program disk in another (drive B). Type the following:

A> *EXEC B:HM* ENTER

The program will automatically load and run. The 80-column version of *HARDMASTER* may also be loaded in this fashion, except the program filename is HM80 and not HM.

From the GPL Interpreter:

Load the GPL Interpreter from the M-DOS command line. Either load the Editor/Assembler or TI-Writer modules, and follow the instructions above in the TI-99/4A section. Again, the 80-column version may be used in the GPL mode by using the program HM80 instead of HM.

USING HARDMASTER

HARDMASTER, unlike many disk sector editors, is command and not menu-driven. Instead of selecting an option by pressing a key (for instance), you type a command and the program performs it. This may seem to be a little "less friendly" than some programs, but allows for more capable programs.

On loading the program, you'll be asked to enter the number of the hard drive you wish to access. If you haven't yet purchased your hard drive and are using

floppy drives (and/or RAM-disks) with your HFDC, then enter Q. You will then be given the opportunity to input a floppy drive number instead.

After that drive is checked to be valid and available, a list of command options is displayed (as shown below) on the screen. At the completion of any command on the 40 column version, this list of options will be reprinted on the screen making it unnecessary to have to remember the commands and the parameters each needs. On the 80 column version, the list of commands available is listed on lines 25 and 26 at the bottom of the screen.

EDit s	**PRint s s**	**QUad s**	**DIrectory**		
FInd s s w/$	**PAck s b**	**AScii s s**	**MAp**		
HD n	**FD n**	**CO s s n**	**OutDev**	**TRee**	**EXit**

As you will note, many of the commands expect additional information (or post script data). The postscript, "s", stands for sector, the "w" for word of 2 bytes, the "b" for byte, the "$" for Ascii string and the "n" for number. This information must be included with the command so that it functions. For example, to edit sector hexadecimal 100 (decimal 256) you'd type:

ED 100 ENTER

Command inputs all must begin with a 2-letter command name and followed, in most cases by parameter numbers in hexadecimal but do not use the ">" sign before the numbers. The command and each parameter must be separated by ONE SPACE character.

All output to the screen or printer use the hexadecimal format with one exception. That is in the record length of fixed or variable files (e. g. D/V 80). We are used to seeing them as decimal numbers in all common directory listings and for convenience, this has been maintained.

A status line is displayed at the bottom of the screen during most phases of operation of the program. The status line shows which drive has been selected, the diskname, the disk size and the sector number being accessed. The status line is continually being updated as different sectors are being read.

The running of some functions can be interrupted by pressing a key to pause and press again to resume. Those functions with a pause facility also have been provided with the opportunity for the user to BREAK altogether. To do this press FCTN 4, FCTN 9 or CTRL C. If output is going through the RS232 card and you press FCTN 4, the DSR routine in the RS232 will detect this keypress, perform a break, and give an error code which will appear on the screen as "Device error". In this case the printer file is not closed and information may still remain in the printer buffer. The next time the printer is used, the first part of the printout may not be formatted as it should be. This is no fault of *HARDMASTER* and can be prevented by BREAKing a printout with the FCTN 9 or CTRL C combination that *HARDMASTER* looks for.

Lastly, most commands in this program are used to read the contents of disk sectors. The program does not allow you to change any of the contents of a disk

unless you specifically tell it to do so. Only a few commands allow changes to the disk but, before this happens, you must consciously change the default input from N to Y.

THE COMMANDS

Each command listed below is followed by the information it expects in the command string. The code in parenthesis refers to the type the data should be in. Again, the postscript, "s", stands for sector, the "w" for word of 2 bytes, the "b" for byte, the "$" for Ascii string and the "n" for number.

HD number (n)
FD number (n)

The HD and FD options allow you to change the drive numbers the program will access when a command is executed. The HD command sets the current Hard-drive, and the FD sets the floppy drive.

The HD command allows numbers from 1 to 3 while, FD allows for drives 1-9 as well as A-Z. The latter are included in anticipation of future RAM-disk partitioning with the new version of ROS. If you have a DSK1 emulation file on the hard disk and the HFDC card at >1100, this can be accessed as FD 1, and the first mechanical floppy drive would be DSK2. Horizon RAM-disks set to any drive number up to 9 and any CRU address can be accessed. The order of CRU polling by the DSRLINK is from >1200 to >2000 then >1000 and >1100.

ED sector (s)

The EDit s command is the most versatile and will be the most used. When you suffix the *ED* command with a sector number (note must be in hexadecimal) and press the ENTER key, the contents of the sector will be displayed in hexadecimal on the screen together with instructions on key usage. The CTRL key is used in combination with W to Window from hexadecimal to ASCII, with B to go Back one sector, with N for Next sector higher and with C for esCape to command mode.

The FCTN 5, 6, 4 and 9 keys are also active (being more convenient to use on the Geneve). In EDit you can skim the cursor over the sector display with the FCTN and arrow keys (E,X,S, and D) or you can type over the contents on either the hexadecimal or ASCII window. On the hexadecimal window, you are allowed to type only hexadecimal digits.

The only ways out of this command are CTRL C, the escape keystroke or FCTN 9, the BACK key combination. If you have previously typed over any of the sector contents on the screen purposely or accidentally, you will be given the option to save or not to save the changes to disk.

Sector number inputs when using the hard drive can be from one to five digits up to FFFFF and for the floppy drives up to 4 digits. In either case trying to

access a sector number greater than the disk size as shown on the status line at the bottom of the screen will result in an error message.

PR start_sector end_sector (s s)

PRint will print out to the output device whose name is in memory the contents of the sectors from the start sector to the end sector. You get a printout of both the hexadecimal contents and also its ASCII representation. Of course some values in hex have no ASCII counterpart so a dot is printed instead.

OD

OutDevice allows you to change the output device name from the default of PIO. This command needs no parameters. After typing OD, and pressing ENTER, you are asked to type in the name of the device to which you want the output to go. This can be through the RS232 card or wherever you wish.

QU sector (s)

QUad is based on the FORTH system of displaying 4 sectors at a time on the screen. You can step through sectors quite quickly this way if you are searching for some known visual characteristic of a disk sector. In the 40-column version the contents of four sectors won't fit onto the screen, so horizontal windowing is employed. No windowing is needed in the 80 column version as the whole sector is displayed on the screen. The same CTRL keys as for EDit command are active in QUad, including the FCTN 4, 5, 6 and 9 keys as well.

DI

The DIrectory command gives you a listing of the files on the currently active hard or floppy drive. The report contains quite a bit of information, including: the filename; the number of sectors that the file occupies; the file type abbreviated to 3 characters; length of record (if applicable - and the only numeral in the program shown in decimal notation); the sector number that the File Descriptor Record (FDR) is stored on; and the sector clusters where the actual file is stored on the disk. If the file on the disk is fractured, more than one cluster is shown.

If a hard-drive based file is severely fractured, then the pointers to its sector clusters may spill over onto another FDR called an offspring FDR. These clusters will also be reported by *HARDMASTER*.

DIrectory is a multi-purpose routine. It automatically categorizes your disk into one of three types. In all cases you will first have to indicate whether you want the output to go to the screen or to the printer. If you have changed the printer device name default (with the OD command above) from PIO to a disk filename, then all information will go there. What the routine does varies according to the circumstances in which it is used:

1. *With a hard drive.* You will have to type the full path name of the directory on the hard drive you wish to catalog. The hard drive caters for a root directory

as well as sub-directories, each of which has pointers to more sub-directories. To access the root directory, just type the path name, *ROOT*. To access a sub-directory such as ADV, type the full path name (e.g. TI.GAMES.ADV) placing a period between each, but no period at the end even though the last entry is a sub-directory. Including the device name in the path name is not required.

2. *With a floppy drive formatted by Myarc Disk Manager 5.* This format, a little like the hard drive, has a root directory and can have also up to three subdirectories. When using the command, DI, with this type of disk, *HARDMASTER* checks the disk and if there are sub-directories as well as the root directory, presents an option list from which to choose. If no sub-directories are found, then the root directory is automatically processed.

3. *With a floppy drive initialized TI-style.* When you want a directory of this type of disk, the program simply shows the listing of files on the disk.

FI start_sector end_sector word_string_to_search_for (s s w/$)

When the FInd command is executed, a search of the disk is made from the starting sector (the first "s"), to the ending sector (the second "s"), for the 2-byte word or the ASCII string that is entered. The ASCII string can be of any length, and must contain a "$" character (as in *$MY TEXT*) as its first character. The two byte word or string is looked for at both even and odd addresses and the search can even detect a match formed by bridging the last of one sector and the beginning of the next. The results of a search can be directed to either the screen or a printer (or other output device specified).

TR

The TRee command is a unique function of *HARDMASTER*, and one which you'll most likely find very useful. If works as a sort of extended DIrectory command.

TRee needs no parameters, but after entering the command you have the option of selecting whether to access the DIRECTORIES or DIRECTORIES AND FILENAMES. If you select the former, then a list of directory pathnames and the DDR (Directory Description Record) sector numbers of each are output. For the other option, directory pathnames and their DDRs, as well as lists of their filenames and FDRs (File Descriptor Records) are produced. The results can be directed to the screen or printer. When sending the output to the printer, the printout is tabbed over to start with a left margin of 10 by using the printer code of <27>"l"<10>. This margin makes the printout more suitable for filing in a ring binder or similar folder.

AS start_sector end_sector (s s)

The AScii command gives you an ASCII printout of the range of sectors specified in your command string. The two parameters following the AS command represent the first and last sectors of the ones to be accessed. Only the ASCII representable characters will be printed - each sector consisting of 4 lines of 64

characters. Byte values below 32 and above 127 are printed as spaces.

A printout of this information can be very useful - one day you may not be able to access a text data file in its entirety. Using this command, however, you can print out most of the accessible sectors and so recover on paper or disk much of the file. The disk file can then be edited with TI-Writer.

MA

The MAp command directs to the output device (or to screen in the 80 column version) the sector allocation table or bit-map of the disk in the current drive. This command works intelligently - it takes into consideration the big differences in the bit-maps of hard drives and floppy drives, and automatically selects the form of printout suitable for the particular kind of drive you are accessing.

The floppy drive printout will show each sector on the disk and whether it is marked off as used or free. One thing to be aware of in interpreting the printout is that the bytes on sector 0 that represent used and free sectors are bit-reversed from the order of the sector usage on the disk. e.g. the byte 3F (00111111) on sector 0 will indicate the following series of sectors to be used (X) and free (-) with XXXXXX--.

It is not practical to show the sector allocation table of a hard drive as individual sectors because of the immense amount of paper and time needed to print it. The printout has been confined to listing the hexadecimal bytes of the bit-map plotted against the sector numbers that they represent. Remember in interpreting the table to take into account the number of sectors your hard drive allocates for each AU (Allocation Unit). Unlike the TI bit-map, each byte representing 8 AUs is NOT bit-reversed.

Note that the MAp procedure for the hard drive begins by reading the bit-map sectors from >1F backwards towards sector 1 until it comes to a sector containing non-zero bytes. It then assumes (well this author does anyway) that this is the last of the used bit-map, and prints out the allocation table from sector 1 onwards till the end of this sector. Very good in theory but unfortunately your hard drive may have contained bad sectors when it was formatted, and parts of the bit-map towards the logical end of the drive accordingly could have been marked off as used. If MAp prints out the allocation table past the obvious end of used bit-map, then it can be interrupted with FCTN 9 or CTRL C.

PA sector byte_to_pack (s b)

PAck s b will not have a great deal of use but can be handy when rewriting a bad directory by first packing the DDR sector with zeros. The first argument of the command, the "s", is the sector which is to be packed, and the "b" is the byte to be written into each of the >100 (256) places within that sector. Before this command is executed, you must confirm your intention by changing the default option from N to Y. If you need to fill a group of sectors with the same byte, first PAck the lowest sector number, then use the COpy command to copy that sector to the next one above a given number of times.

CO source_sector destination_sector number (s s n)

This command allows you to copy one sector to another (or to more then one other) on the current drive's disk. Take care when using this command and make sure your command arguments are as you intend. As a precaution, you will be asked to confirm the command before the program allows you to proceed by changing the default input from N to Y.

The source sector is where the copy will start from, the destination sector is where the copy source sectors will be copied to, and number is how many consecutive sectors to copy.

The hard drive keeps a copy of the sector 0 VIB (and the bit-map that is on sectors 1 to 1F) on sectors 20 to 3F but only during initialization. Neither of these higher storages are updated as files are added or deleted from the disk. Using the CO command you can regularly and often make an up to date copy of these vital sectors by selecting the hard drive then typing:

CO 0 20 20 ENTER

If the data on sector 0 becomes somehow corrupted in the future, then it is an easy matter to restore it. Using this command to move masses of sectors might not be advisable, as the routine copies sector by sector and quite an amount of head stepping would take place.

FB b

The FB b command, which is not listed on the screen with the other commands, allows you to simultaneously set the foreground and background colors of the screen. The byte "b" contains both colors - with the first nibble representing the foreground color and the second the background color. To change the color to black text on a gray background, for instance, you would type:

FB 1E ENTER

The colors, which are also listed in the Editor/Assembler manual, are as follows:

0	Transparent	8	Medium red
1	Black	9	Light red
2	Medium green	A	Dark yellow
3	Light green	B	Light yellow
4	Dark blue	C	Dark green
5	Light blue	D	Magenta
6	Dark red	E	Gray
7	Cyan	F	White

The default colors are F4, with F the white foreground and 4 the blue background.

EX

EXit is provided as an "out" - or a way to quit from the program. This is

because the normal quit key has been disabled. If you entered HARDMASTER from FUNNELWEB then this program keeps F'WEB's colors and on exit, it returns to F'WEB. This feature makes it a suitable program to run through F'WEB's load option screens.

PART II -
HARD DISK INFORMATION
by Garry J. Christensen

Myarc did well with the manual that accompanies the Hard/Floppy Disk Controller Card, giving instructions for the use of MDM5, interfacing with Basic, hardware specifications and low level access. It is this final point that I wish to concentrate on here.

Unlike the documentation that accompanied much of the TI equipment, Myarc have provided considerable detail about accessing the hard disk with assembly language. The manual lists the parameters required and the addresses to be used when utilizing the on-board subprograms. It also details the layout of the data in the important sectors on the hard drive.

While this data is supplied, it must be considered as the "minimum necessary". If you intend to make any use of the data, there is some investigation left to be done. With this in mind, I have written this article to save hard disk users some work.

Before I go any further I must mention something called an Allocation Unit (AU). An allocation unit is a block of consecutive sectors. There may be up to 16 sectors in one allocation unit. The disk is considered as a series of allocation units, the size of each being determined by the size of the disk.

All pointers are one word so the maximum number of allocation unit's on the disk is >FFFF. If it has a large storage capacity, there is a larger number of sectors/allocation unit.

Allocation unit's are only used to find a file. All directory descriptor records (DDR) and file descriptor records (FDR) are located on the first sector of an allocation unit. The remainder of the allocation unit is rarely used. All files start at the first sector in an allocation unit and use consecutive sectors, unless the file is fractured.

Volume Information Block and Directory Descriptor Records.

OK, let's go through sector 0. Refer to page 62 in the manual for this section.

WDS1	HD		Sze>13380		Sec>0				
Addr	0 1	2 3	4 5	6 7	8 9	A B	C D	E F	
00	4844	2020	2020	2020	2020	99C0	2020	0000	HD .. ..
10	1300	13A5	B24F	2907	0020	029C	003D	0083	...I..).. ...=..
20	00DD	0105	01D8	01EE	0313	0000	0000	0000	.....
30	0000	0000	0000	0000	0000	0000	0000	0000	.....
40	0000	0000	0000	0000	0000 etc.				.....

The first 10 bytes are set aside for the name of the hard disk, that much is obvious.

The word at >0A is the total number of allocation units on the disk. Remember that the number of sectors on the disk is this number multiplied by the number of sectors per allocation unit (more about that in a minute).

Byte >0C is the number of sectors per track.

The following three bytes are a mystery. The manual says that they should be "WIN" but this is not the case. They have been used to store some data necessary for the operation of the disk.

Bytes >10 and >11 are also formatting parameters. These are listed in the manual. However only one part is of interest to us. The first nibble is the number of sectors per allocation unit minus 1.

Look at the example above. You will see that the word is >1300. The first 1 means that I have 2 sectors per allocation unit (1+1). All pointers and values must be multiplied by 2. If you have a calculator that will do hexadecimal arithmetic, get it out now. If you don't, here's a great chance to practice on paper.

You will see that I have >99C0 AU's on the disk and 2 sectors per AU. A total of >13380 sectors (78720 decimal).

The next two words, at >12 TO >15, contain the date and time of creation. The time is first. Write the value down in binary notation. The first 5 bits are the number of hours (24 hour clock), the next 6 are the minutes and the final 5 is the seconds divided by 2. When you catalogue your disk you will see that the seconds are always even.

Do the same with the date. The first 7 represent the year, the next 4 are the month and the final 5, the day.

In the printout of my sector 0 the words >13A5,>B24F are the time and date. Breaking them into binary gives 0001 0011 1010 0101, 1011 0010 0100 1111. When split they look like this:

00010	2 Hours
011101	29 Minutes
00101	10 Seconds (x2)
1011001	89 Year

0010 2 Month
01111 15 Date

Byte >16 is the number of files in this directory and byte >17 is the number of subdirectories. I doubt that this needs any further description.

There is no room in this sector to store the pointers to all the files that may be under this directory. Another sector is used just to store these pointers, called the file descriptor index record (FDIR). The word starting at byte >18 is the AU of this list. Remember that this number must by multiplied by the number of sectors/AU to get the physical sector number.

Check your disk and have a look at the FDIR. It should look something like this:

WDS1 HD Sze>13380 Sec>29E

Addr	0 1	2 3	4 5	6 7	8 9	A B	C D	E F	
00	026D	0267	01B4	01B5	01B6	01B7	01B9	0085	.P.....
10	0215	0216	0217	0218	0219	0085	0021	01BB	.....
20	0000	0000	0000	0000	0000	0000	0000	0000	.....

Each word is the AU of the file descriptor record (FDR) for each file. We will come to FDR's shortly. All entries are sorted into alphabetical order. When a new file is added, it must be placed in the correct place in this chain of pointers. The 128th entry in this record always points back to the DDR that it belongs to, sector 0 in this case.

As you are aware, the hard disk can emulate DSK1 in a number of ways. One method is called DSK1 file emulation. In this method, the entire floppy disk is copied sector by sector and is used on the hard disk just as it would be on a floppy.

The word at byte number >1A is the pointer to the emulate file that is active. The actual value given is the AU of the file descriptor record. If it is >0000, there is no file active.

Finally, from >1C onwards there is the list of pointers to the file descriptor records (DDR's) of all the subdirectories. Each pointer gives the AU of the DDR. In the example of the volume information block above, the first entry points to the first directory, and so on until a 0 is reached. These entries are also sorted in alphabetical order.

The DDR's are the same format as sector 0 (the volume information block) except that "DIR" is present at 0D to 0F and the emulate pointer is replaced with a pointer to the parent DDR. The parent DDR is the directory that contains this directory. Following this path of pointers will lead back to the VIB.

Sector Usage - the Bit Map.

All disks must keep record of which sectors are used. The hard disk is no dif-

ferent. The bit map starts on sector 1 and is quite simple to follow. Each bit represents an allocation unit. If the bit is 1, the AU is used. The bits are taken in order from most significant to least significant. To calculate the actual sector number represented there is a little math to do.

Take the sector number (S) that you are looking at and subtract 1. Multiply this number by >100 and add the number of the byte (B). Multiply that by 8. The result is the AU represented by the first bit in that byte. By means of an example, on sector 5 the >50th byte in the sector may be >80.

(((5-1)*>100)+>50)*8 = >2280

In this way you can calculate the position of any used or unused sector on the disk. Look for the end of your bit map, calculate the sector number and have a look to see if you are right.

Conversely you can find the bit-map byte in which a bit represents a particular AU on the disk. Divide the AU by 8, add >100 and modulo divide by >100. Any remainder in the division by 8 is the required bit in that byte.

The formulae are:
(((S-1)*>100)+B)*8=AU
(AU/8+>100)/modulo>100=SremB

An extra feature that the hard disk has over and above the floppy drives is a separate bit map for the DDRs and FDRs. On my disk, all the first sector has been set aside for DDRs and FDRs. The files do not begin until sector >1000. When a DDR or FDR is created, it is put in this area and the sector marked off in sector 1. Unless the disk has a large number of files, all the descriptor records are at the beginning.

File Descriptor Record.

If you have followed through this far you will be able to track down any file descriptor record on the hard disk. The FDR will direct you to the file itself. Refer to page 64 in the manual.

The following is an example of one of the FDRs on my hard disk.

WDS1 HD Sze>13380 Sec>2A0

Addr	0 1	2 3	4 5	6 7	8 9	A B	C D	E F	
00	5041	5241	4348	5554	4520	0000	3900	0028	PARACHUTE. . ...(
10	DB00	0000	1F0C	2C42	1F0C	2C42	4649	0000	.....,B..,BFI..
20	0000	0014	014F	0000	1B68	1B7B	0000	0000	.....O...h.{....
30	0000	0000	0000	0000	0000				etc.....

The first 10 bytes are the name of the file.

Bytes >0A and >0B give the record length if it is greater than 255 bytes.At this stage there are no programs for the TI99/4A that will take advantage of this, however MDOS for the Geneve 9640 does have support for this type of file.

Byte >0C describes the file attributes. I will not be listing all the functions of the bits here because they are well described in the manual.Please note that in the manual, bit 0 is the least significant bit.The example above indicates that the file is a program, protected, has been modified since the last back-up and is a DSK1 emulate file.

Byte >0D indicates the number of records per sector.

Bytes >0E and >0F tell you the number of sectors allocated to the file.A very large file may have more than FFFF sectors allocated to it so the most significant nibble is stored in the most significant nibble of the extended information area.This is the MSN of byte >26.Place this value in front of the value in bytes >0E and >0F to give the true file length.

The value of byte >26 is 0 and the length is listed as >0028.The total length in sectors is >00028.

The end of file offset is located at byte >10.Variable length and program files may not necessarily end at the end of a sector.This byte indicates where in the final sector the EOF marker byte is located. Byte 10 shows >DB, so the final sector of the file will contain BC bytes relating to the file and the EOF marker byte at >DA.There will be "garbage" in the remaining unused bytes.

The logical record length in byte >11 is set to the record length for fixed length records, the maximum length of a record for variable length records and 0 for program files and the data files that make use of extended record lengths.In the case of DV80 files, this byte is set to 80.

Bytes >12 and >13 and the second nibble in byte >26 indicate the number of sectors actually used in this file. If you open a file and specify its size, that number of sectors is allocated for its use. You may not have written to all the sectors at that stage.The value derived from this part of the FDR indicates the highest sector written to for variable length files and the highest record written to in the case of fixed length files.

The next 4 words are the date and time that the file was created and last updated. I have covered the encoding of time and date already.

Bytes >1C and >1D are always "FI".

On a hard disk, fractured files are quite common. Long files may be split into many fragments. I once assembled a large source code and sent the list file to disk. The assembler wrote a small portion of object code then some list file, then some more object code and so on. The resultant object and list files were tightly interwoven.

To allow for the case where there is not enough space for pointers to all the parts of the file in one FDR, the controller creates another FDR in which to continue the table.This is called an offspring and the original is called the parent.

Bytes >1E and >1F are the pointer to the AU of the parent FDR.If this is the parent, this value is 0. Bytes >20 and >21 point to the AU of the offspring FDR.This chain of parent and offspring FDRs can continue until the file is stored or the disk is filled.

Where there is more than 1 sector per allocation unit, the offspring FDRs may be placed on any unused sector within the AU. If this is the case, the most significant nibble of byte >27 points to the sector number within the AU of the parent FDR and the least significant nibble points to the sector of the offspring FDR.

The above example is the parent FDR and it has no offspring.

The word starting at byte >22 gives the total number of AU used for the FDRs for this file and the next word is the pointer to the file descriptor index record. This record contains pointers to all the FDRs in this directory and the pointer to it allows the user to follow the path back to directory. The example shows that the file descriptor index record is at AU >014F.

The function of the bytes in the extended information area have already been covered.

Up to this point, there have been few differences from the FDRs on a floppy disk where the functions are common. The pointers to the file are different. The hard disk points to the AU at the start of the file fragment and points to the last AU used. These two pointers are called a cluster.

The clusters are 2 words each and occupy the rest of the sector from byte >28. The chain of clusters is ended by a 0. Only one cluster is shown in the example. This file is continuous from AU number >1B68 to >1B7B.

With the information above you should be able to follow any valid path from sector 0 to the file that you are looking for. You will be able to use a sector editor to repair files that have been corrupted or modify files on the disk directly.

PART III -
FLOPPY DISK INFORMATION

by Colin Christensen

Disk Sector Organization

A single-density TI formatted disk has 360 sectors if it is one sided and 720 if it's double-sided. Double density disks have twice that number of sectors, and quad density have four times that number. Each sector has 256 (>100) bytes of accessible information. The sectors are numbered from >0 to >167 for SSSD, >0 to >2CF for DSSD, and >0 to >59F for DSDD.

DISK SECTOR PLAN

Sector >0	Sector >1	Sect. >2 to >21	Sect. >22 -->
Disk Info.	Pointers to Location of FDRs	File Directories	File Storage

Sector >0 contains general information about the disk itself.

Sector >1 contains pointer numbers to the sectors that contain the File Descriptor Records (FDR) that contain information about each file on the disk.

Sectors >2 to >21 are primarily reserved one for each FDR. If the disk already contains enough FDRs to fill all sectors from >2 to >21, and another file is saved to the disk, its FDR will be placed on the next available sector on the disk. That would generally be immediately following the sector where the last file storage ended.

Sectors >22 onwards to the capacity of the disk are used to store programs or files. If however, the disk is otherwise full and no other sectors are free, then actual files can be written to any vacant sectors from >2 to >21.

ALL NUMBERS FROM NOW ON WILL BE EXPRESSED IN HEXADECIMAL DIGITS but decimal numbers will be written in words. The term "file" will be used in a general sense to mean any batch storage of data on disk.

Sector 1
Pointers to File Descriptor Records

Sector 1 contains the pointers to indicate where each file's directory record, (FDR), is stored on the disk. Entering the command:

PR 1 1 ENTER

would show a printout similar to that below.

```
Drv5   RAMDISK5      Sze>2E0     Sec>1
       0 1    2 3    4 5    6 7    8 9    A B    C D    E F
00     0003   0004   0005   0006   0017   0015   0011   0010     ................
10     0012   0007   0008   0009   0014   0016   000A   000B     ................
20     000C   0002   0018   001C   000D   0013   0019   001A     ................
30     001B   000E   000F   0000   0000   0000   0000   0000     ................
40     0000   0000   0000   etc.
```

The first file listed has its FDR on sector 3, the next is on sector 4, and so on according to the entries above, until we find that the last file uses sector F for its FDR.

The individual FDRs for the files on this disk are stored on all sectors from 2 to 1C but they are not listed in numerical order.The disk controller arranges the files in the alphabetical order of their filenames, hence when a new file is added or deleted from a disk, Sector 1 is updated to conform to the alphabetic order

change.

When a file is deleted using the Basic command DELETE "DSK1.xxx" or by using Disk Manager, the FDR pointer only in Sector 1 is removed but the FDR itself remains intact. The sectors where the file is actually stored are also marked as vacant.

Sectors 2 to 21
File Descriptor Records

These sectors are primarily reserved for storage of file descriptor records.They contain all the information about each particular file.

Suppose we wish to look at the FDR of an Extended Basic program called FLIGHT. The easiest way is with the directory command, DI. A list of all files on the disk will be displayed as well as the FDR number for each.

Assume that FLIGHT's FDR is on Sector 14. Type the command

PR 14 14 ENTER

and the following could be a typical printout. (You could have also viewed it on the screen with the ED command).

```
Drv1      EXBASIC5        SZE>2D0     Sec>14
          01   23   45   67   89   A   CD   EF
00        464C 4947 4854 2020 2020 00  0100 0023  FLIGHT.....#
10        BA00 0000 0000 0000 0000 00  3121 0048  .............1!.H
20        C101 6A21 0200 0000 0000 00  0000 0000  ..j!............
30        0000 0000 0000 etc. .
```

Bytes 00 to 09, the first ten bytes, show the filename of the file.

Bytes 0A and 0B show here the record length if it is a data file and has a record length greater than FF. TI files leave this word unchanged.

0C is an indication of the file-type.

01 = PROGRAM
09 = PROGRAM and protection
00 = DIS/FIX
08 = DIS/FIX and protection
80 = DIS/VAR
88 = DIS/VAR and protection
02 = INT/FXD
0A = INT/FXD and protection
82 = INT/VAR
8A = INT/VAR and protection

Byte 0D is the number of records per disk sector (00 here if not applicable to that type of file).

Bytes 0E and 0F are together the number of sectors that the file actually occupies on the disk.

Byte 10 is the End of File Offset (not applicable for FIXED) in the last sector of the file. Byte 10 of FLIGHT happened to store the value BA. If we read the very last sector where FLIGHT is stored, we would find that byte at offset BA on this sector has the value AA. The EOF marker for program type files is AA while most other file types use FF.

Byte 11 is the record size. For example, a "DIS/VAR 80" file would show 50 here. This byte is >00 if record size is not applicable.

Bytes 12 and 13 show, for fixed length record files, the highest record actually written to. For variable length files it contains the highest sector actually written to. The bytes in this word are reversed.

The TI disk controller differs as it stores at 12 and 13 the number of records in this file.Not applicable for PROGRAM. The two bytes must be divided into 4 nibbles to get the correct number. The order of nibbles to read is 4th nibble, 1st nibble then 2nd nibble. (Disregard the 3rd one) .e.g. EF00 becomes 0EF and AA01 becomes 1AA.

Bytes 14 to 17 show the date and time of creation of this file. For more details on converting the values stored here, see the section, *Hard Disk Information*.

Bytes 18 to 1B show the date and time of the last update to this file. Please note: 14 to 1B are not used by the TI disk controller.

Bytes 1C onwards until >0000 occurs are the area that indicates where on the disk the file is actually stored. Sometimes when files have been deleted from the disk, previously used sectors become vacant and another program saved to disk will then occupy the lowest numbered unused sectors first. In this way a file becomes "fractured" if it is too large to fit into a recently vacated area.

Pointers to each section where the file parts are stored are recorded in byte 1C onwards in sets of 3 bytes. In FLIGHT these sets are 3121 00, 48 C101 and 6A21 02. Here again one must manipulate the 6 nibbles of each to obtain two details, the LOCATION on the disk and the SECTOR COUNT of the last consecutive sector stored in that part of the disk. Remember the SECTOR COUNT begins with 0.

To extract the information needed, take the 4th nibble, 1st nibble then the 2nd nibble to make the LOCATION. Take the 5th, the 6th then the 3rd to get the SECTOR COUNT.

e.g.
3121 00 becomes 131 002
48 C101 becomes 148 01C
6A21 02 becomes 16A 022.

FLIGHT, starting at a sector-count of 0, is stored beginning at Sector 131 for

three counts, 0,1,2 then jumps to another starting location of 148 to store count numbers 3 to 1C and then to location 16A from whence count numbers 1D to 22 are stored.

The number of sectors indicated in 0E-0F of the FDR shows 0023 sectors being occupied by FLIGHT. This number tallies with the 0-22 just seen (the 0 and 22 are inclusive). This file also uses one more sector, namely Sector 14, for its FDR, making a total of 24 or thirty-six, the number that the DISK MANAGER or a LOAD program would show when reading the disk's directory.

Sector 0
Volume Information Block

A sample printout of Sector 0 follows.

Drv1 EXBASIC5 Sze>2D0 Sec>0

	0 1	2 3	4 5	6 7	8 9		C D	E F	
00	5445	5354	2020	2020	2020	8	0944	534B	TEST...DSK
10	2028	0101	0000	0000	0000	0	0000	0000	(.....
20	0000	0000	0000	0000	0000	0	0000	0000	.....
30	0000	0000	0000	0000	DF01	0	FCFF	FFFF	.....
40	FFFF	C17F	0000	0000	0000	0	0000	0000	.....
50	0000	0000	0000	0000	0000	0	0000	0000	.....
60	0000	0000	00FF	FFFF	FFFF	F	FFFF	FFFF	.....
70	FFFF	FFFF	FFFF	FFFF	FFFF	F	FFFF	FFFF	.....
80	FFFF	FFFF	FFFF	FFFF	FFFF	F	FFFF	FFFF	.....
90	FFFF	FFFF	FFFF	FFFF	FFFF	F	FFFF	FFFF	.....
A0	FFFF	FFFF	FFFF	FFFF	FFFF	F	FFFF	FFFF	.....
B0	FFFF	FFFF	FFFF	FFFF	FFFF	F	FFFF	FFFF	.....
C0	FFFF	FFFF	FFFF	FFFF	FFFF	F	FFFF	FFFF	.....
D0	FFFF	FFFF	FFFF	FFFF	FFFF	F	FFFF	FFFF	.....
E0	FFFF	FFFF	FFFF	FFFF	FFFF	F	FFFF	FFFF	.....
F0	FFFF	FFFF	FFFF	FFFF	FFFF	F	FFFF	FFFF	.....

Bytes 00 to 09 are the diskname, and occupy ten bytes.

Bytes 0A and 0B show the number of sectors on the disk. The value 02D0 converts to seven hundred and twenty. Single-sided disks would show 0168 equivalent to three hundred and sixty.

Byte 0C gives the number of sectors per track. Always 09.

Bytes 0D to 0F are always 44 53 4B (DSK).

At byte 10, 50 is placed here for a protected disk and 20 otherwise.

Byte 11 is the number of tracks. TI disks are normally 28 (forty).

Byte 12 shows the number of sides. 01 equals single-sided, 02 is double-sided.
Byte 13 is the disk density. Again, 01 for single-density, and 02 for double-density.

Bytes 14 to 1D show the ASCII equivalent of a sub-directory name (similar to that stored in bytes 00 to 09).

Bytes 1E and 1F are together a pointer to the FDR-pointer for this sub-directory. If >0000 is stored here, there are no sub-directories. Explaining further: If, for example, 0156 is shown here, then sector 156 contains a list of pointers like those on sector 1. These pointers show which FDRs are accessed through the path name starting with this sub-directory.

Bytes 20 to 29 are together the name of sub-directory 2.

Bytes 2A and 2B are together a pointer to the FDR-pointer for this sub-directory. If >0000 then there's no sub-directory 2.

Bytes 2C to 35 are the name of sub-directory 3

Bytes 36 and 37 form a pointer to the FDR-pointer for sub-directory 3. Again, if >0000 then there is no sub-directory 3.

Note: TI controllers do not use the area from 14 to 37.

Finally, bytes 38 to 64 are the sector bit-map for a single-sided disk. 38 to 91 is the sector bit-map for a double-sided disk. Bytes 38 to FB are the sector bit-map for a double-density, double-sided disk.

Directory and Sector 0 Bit-map

(The MAp command will print out an expanded bit-map showing which individual sectors are used and which are free.)

Referring to bytes 38 onwards, the value of each byte (e.g.FF) must be converted to its binary equivalent 11111111. If the bit is a 1, that sector is used. If it is a 0, the sector it represents is vacant.

Byte 38 represents the first 8 sectors on the disk from 0 to 7, byte 39 shows whether Sectors 8 to F are used or free and so on.

Now comes the difficult part. First, the 8 binary bits of each byte must be reversed end for end. If byte 38 was 1F (00011111), then by reversal of the bits (11111000), the first five sectors 0 to 4 are used and 5, 6 and 7 are not.

Below is a sample printout of a directory and the sector bit-map effected separately by the commands DIrectory and MAp.

"Clusters" indicate the starting sector and ending sector where a contiguous block of data is stored on the disk. If a file is fractured, it will be stored on a number of clusters.

```
Diskname=GENEALOGY    Free=14        Used=2E7
Filename      Sze  Typ     Len   FdrNo   Clusters
-READ-ME-     11   D/V     80      2       22-31
ABBREV/I      18   D/V     80      3       32-48
APPEND-1      4D   D/V     80      4       49-94
BEG-GENE      2A   D/V     80      5       95-BD
GENE/4GEN     15   D/F     80      6       123-136
GENE/DOC-1    66   D/V     80      7       BE-122
GENE/PRNT     20   D/F     80      8       137-155
GENEALOGY     40   D/F     80      9       156-194
TREEDETAIL    C    D/V     80      A       195-19F
```

Sector 0 Bit-map

With reference to MAp's printout, the sectors are marked as Used (X) and Free (-). Beside and above the map are reference numbers that help to find a particular sector. e.g. GENEALOGY, with a cluster of 156-194, has a starting location beside 140 and below 16 (140+16=156) and the last sector is found on the map beside 180 and under 14 (180+14=194).

To the left of the bit-map printed below are shown the sector 0 offset addresses and the words stored there that represent the used and free sectors. If you are going to adjust the bit-map, then the offset on sector 0 is the place to do it.

```
Drv1 TI-WRITER2 Sze>2D0 Sec>0
DISK-SECTOR-BIT-MAP X = Used      - = Free
                SEC00 02 04 06 08 0A 0C 0E 10 12 14 16 18 1A 1C 1E
                TOR +-+-+-+--+-+-+-+--+-+-+-+--+-+-+-+-
38=FF3F0000     000|XXXXXXXX|XXXXXX--|--------|--------
3C=FCFFFFFF     020|--XXXXXX|XXXXXXXX|XXXXXXXX|XXXXXXXX
40=FFFFFFFF     040|XXXXXXXX|XXXXXXXX|XXXXXXXX|XXXXXXXX
44=FFFFFFFF     060|XXXXXXXX|XXXXXXXX|XXXXXXXX|XXXXXXXX
48=FFFFFFFF     080|XXXXXXXX|XXXXXXXX|XXXXXXXX|XXXXXXXX
4C=FFFFFFFF     0A0|XXXXXXXX|XXXXXXXX|XXXXXXXX|XXXXXXXX
50=FFFF0100     0C0|XXXXXXXX|XXXXXXXX|X-------|--------
54=00000000     0E0|--------|--------|--------|--------
58=00000000     100|--------|--------|--------|--------
5C=00000000     120|--------|--------|--------|--------
60=00000000     140|--------|--------|--------|--------
64=00000000     160|--------|--------|--------|--------
68=00000000     180|--------|--------|--------|--------
6C=00000000     1A0|--------|--------|--------|--------
70=00000000     1C0|--------|--------|--------|--------
74=00000000     1E0|--------|--------|--------|--------
78=00000000     200|--------|--------|--------|--------
7C=00000000     220|--------|--------|--------|--------
80=00000000     240|--------|--------|--------|--------
84=00000000     260|--------|--------|--------|--------
88=00000000     280|--------|--------|--------|--------
8C=00000000     2A0|--------|--------|--------|--------
90=0000FFFF     2C0|--------|--------|XXXXXXXX|XXXXXXXX
```

The bit-map of a quad density disk will differ from the above.There are more

sectors on a quad density disk than there are bits available on sector 0. To overcome this, each bit on sector 0 is made to represent 2 sectors. Therefore a pair of sectors is marked used or free by just one bit on the bit-map.

To further understand the workings of the bit-map, make a printout of the contents of one of your disks using the DIrectory and MAp commands.Then find on the bit-map where each file and its directory is located. Cross out the position on the bit-map where each sector is used by the file. Sectors 0 and 1 are always marked off as used when the disk is being formatted.

PART IV - HARD DISK, FLOPPY DISK AND RAMDISK - MIX AND MATCH

by Garry J Christensen

You may already know that the hard/floppy disk controller (HFDC) will operate up to 4 floppy drives.What you may not understand is that what these drives are called will change and those changes are determined by the system configuration.

The HFDCC is the first card for the TI that will sense what CRU address it occupies.This self-awareness allows it to make changes within the the card that cause it to act according to the requirements of the system.

Let me firstly look at the case where the card is at >1100 and the floppy drives are attached to it.This is the usual address for a disk controller so it will behave quite normally.The jumpers in the floppy drives set the numbers from 1 to 4 and they will respond to those names.Then a DSK1 file emulation is made active.This file on the hard drive will now be accessed whenever 'DSK1' is given as the device name.

At first thought it would appear that floppy drive 1 is no longer operable because all requests go to the emulate file on the hard drive. This is not strictly true because if the file is not found in the hard drive emulation file, it will then look for it on the floppy.

The situation could arise where there is a file of the same name on both but it is the one on the floppy that is needed. No problem.Use the device name "DSK2". In this case all the floppy drive numbers have been moved up one, from drive 2 to drive 5. When the emulate file is no longer active the drives will be returned to numbers 1 to 4.

Now lets look at the situation when the HFDC is not at >1100. In this position

it assumes that it is being used in conjunction with a TI, Myarc or Corcomp floppy disk controller. The four floppy drives that are attached to it now become numbered from 5 to 8, allowing 1 to 4 for the drives on the other controller card. If you try to use 'DSK2' and the other controller is not present, you will get an error.

The following table will summarize what I have said above.

CRU=	>1100		any other
DSK#	Emulate inactive	Emulate active	
1	1	2	5
2	2	3	6
3	3	4	7
4	4	5	8

Now all this is good to know, and if you didn't already, it wouldn't take long to work it out, but it is vitally important when I introduce the topic of RAM-disk. A RAM-disk can be set to any CRU address, and one that will also respond to the 'DSK#' device name.Selection of the address and drive name (s) MUST take into account the effect and position of the HFDCC.

Firstly the order in which the cards are 'looked at'. The original DSRlink routines (the ones in the console, modules and older programs) started at >1000 and went to >1F00.Since RAM-disks have become popular, most programs now start polling the CRU at >1200, go to >1F00 then do >1000 and >1100. This ensures that the floppy drive controller is checked last. Keep the order of polling in mind when considering the next part.

Use the following system as an example.
FDCC at >1100 (drives 1 to 4)
HFDCC at >1400 (drives 5 to 8)
RAM-disk at >1600

What name can the RAM-disk be called. 1 to 4 are assigned to the FDCC, 5 to 8 are for the HFDCC so that leaves 9. Remember that even though a drive may not be attached, the name is reserved by the card.

That is only if the DSRlink starts at >1000. If the modern version is used, the RAM-disk can also be 1 to 4 but this will block out the floppy drive of that name.,

Now try this one.
HFDCC at >1100 (drive 1 to 4/5)
RAM-disk at >1000

No problems here because the RAM-disk will always be checked first. No problem that is provided that you don't want to access the floppy with the same

name. If the RAM-disk is at >1600 there could be a problem. Modern DSRlinks will work as before but old ones will look at the HFDCC first. A trap if the RAM-disk is set at number 5. Sometimes it will work, others not, depending on the status of the DSK1 emulate file.

I hope by now you are getting the idea and you can see that this argument is going in circles. I can not tell you what the answer is because that answer will be determined by what hardware you have, how it is set up, and how you intend to use it.

One obvious solution to the problem of the RAM-disks is to use the new ROS that allows the use of device names from DSKA to DSKZ as well as 1-9. A RAM-disk named in this way would always be available, regardless of the CRU position and the other cards in the system.

When you configure your system, keep these points in mind. If the system is not operating correctly, use this information to work out what is going on inside those bits of silicon. Once set up in an appropriate manner the HFDCC provides an enormously flexible environment for manipulating the floppy drives and RAM-disks.

Disclaimer

Asgard Software, the sole manufacturer and distributor of this program, hereafter referred to as "the product", does not guarantee that this program will be free from error, perform as stated in this manual, or meet the needs or expectations of the user.

Asgard Software is not liable for the use or misuse of this product or any damage that is the result of the improper or proper use thereof - not limited to the proscribed or actual function of the product. Asgard Software warranties the part of the product consisting of the diskette for a period not to exceed 90 days from the date of purchase, provided this part is not damaged by improper use, accident, intentional actions, or from any condition not arising from the quality of the original materials or craftsmanship. Asgard Software reserves the right to reject for service any returned materials.

Asgard Software will service free of charge any product that meets these conditions within 90 days of purchase, and for the cost of return postage after 90 days up to the lifetime of the product.

This product is provided unprotected so that users can legally create copies for their own use. This is not a license to distribute this product. This product is copyrighted in the manner described in this manual, and may not be reproduced by any means for the use of others. In transferring ownership of this software all copies must by similarly transferred in the same transaction. Ownership of this product carries the responsibility to control its use and distribution, and users will be liable to any damage incurred to Asgard Software that may be caused by not carrying out this responsibility.