Here is a little utility that is both useful and instructive. At the same time, it is very dangerous to use; because it allows you to access and modify your Osborne 1 diskettes directly, without regard to CP/M file-structure. NOTE: if you are ignorant about the ins and outs of diskette input/output and prefer to remain that way, you'd better skip this article completely. If you do read on, be especially sure to heed the warnings at the end of the article.

A diskette is physically nothing more than a piece of plastic that has been coated with a type of magnetic material and then inserted into a protective jacket. When given the proper control signals, your Osborne 1 disk drives can read data from a diskette or write data to a diskette by influencing the magnetic coating of the diskette in much the same way a tape-recorder influences the magnetic coating on recording tape.

The arrangement of data on diskette is an elaborate matter. First, the diskette is divided into "tracks." Tracks are merely concentric circles on the disk. Osborne 1 diskettes have 40 tracks. Next, tracks are divided into "sectors." Sectors are slippery characters on the Osborne 1. Physically, there are ten sectors of 256 bytes each on a track of an Osborne single density diskette. The diskettes do not come this way: they are "soft-sectored," which means that the formatting program decides where the sectors are and how they are to be designated. Normally these physical sectors are numbered 0, 1, . . . , 9, although if you ever try to read your diskettes on a different make of computer you may find that it is necessary to refer to these physical sectors as 1, 2, . . . , 10. In any case, you won't have to worry about all of that, because these ten physical sectors are divided into twenty "logical sectors." Henceforth, whenever we refer to a sector we will mean logical sector rather than physical sector.

Normally, the user never hears about any of this (except for the "bdos error: bad sector" message he contends with constantly). CP/M relieves the user of having to think about tracks and sectors by dividing disk data into "files." A file is merely a group of sectors that are considered as one logical entity. CP/M itself keeps track of which sectors belong to which files. It does this by using most of track 3 (tracks are numbered from zero) on every disk. In the directory, one finds the names of all the files on the diskette and information about the allocation of disk sectors.

Brett D. DePaola and Ronald S. Burkey
So what's wrong with that? Nothing really, in a perfect world (counting curiosity as an imperfection). In a perfect world, data always comes in files, and there isn't the slightest reason to access the diskette sector by sector (which is what our utility, "DISKMON," allows you to do). The world, however, is not perfect and we must take into account several facts:

1) Try as we might, we cannot pretend that disk operations are always smooth. Even the most careful operator runs into a power failure during a critical disk write that makes lunchmeat of a valuable diskette. Lesser mortals suffer more frequent errors, since they occasionally do such things as erasing the current copy of their latest program, or turning off the computer before closing an open file. Worse, diskettes sometimes self-destruct for no apparent reason (though, fortunately, very seldom). The knowledgeable user can recover from these mishaps if he can access individual sectors of the disk. He can hunt through the diskette looking for this lost data or he can even repair damaged sectors (that is, sectors with improper data in them).

2) Some people have more curiosity than the proverbial cat and must know how data is laid out on the diskette simply because the user's manual strongly discourages doing so.

3) On a system disk, tracks 0, 1, and 2 hold the CP/M system. On a non-system disk, these tracks, of course, do not hold a system. In fact, they are not used at all. However, if you could gain access to these tracks, you could use this otherwise wasted space for your own purposes. Exploitation of these free tracks may be the subject of a future article.

We present the DISKMON (DISK MONitor) utility with the foregoing reservations. DISKMON works in conjunction with DDT, the useful machine-language monitor provided with the Osborne 1. Basically, DISKMON can read a sequence of sectors from disk into memory, or vice-versa, and can jump to DDT to allow disassembly or modification of what has been read.

How to use DISKMON: first, of course, you must enter the assembly-language listing included with this article using the non-document file creation function of WordStar, assemble it with ASM.COM, and load it with LOAD.COM. How to use DISKMON: first, of course, you must enter the assembly-language listing included with this article using the non-document file creation function of WordStar, assemble it with ASM.COM, and load it with LOAD.COM. (We have assumed a 60K system, and if you know precisely what you are doing. Some sectors of the diskette contain valuable information and access is not directly allowed by CP/M for just that reason. For example, the sample session above would make the diskette involved unbootable, unless you confined yourself to innocuous modifications such as changing the copyright notice.

In short, this program can help you fix—or destroy—diskettes depending on how you use it. DISKMON provides facilities, and though they are included in the Disk Doctor package, they are found nowhere in the utilities provided free with the Osborne 1.
A subroutine which changes the track to read/write.

SUBROUTINE WHICH DISPLAYS THE CURRENT PARAMETERS

SUBROUTINE WHICH PRINTS A READ ERROR MESSAGE AND THE CURRENT PARAMETERS WHEN CALLED.
CALL BTGET ; READ THE TRACK CHOICE
STA KTRK ; STORE CHOICE IN KTRK
JMP COMM

; SUBROUTINE TO CHANGE THE SECTOR

CHSEC: LIX D, P3 ; SEND THE PROMPT
CALL STRSEND
CALL BTGET ; READ THE SECTOR CHOICE
STA KSEC ; STORE CHOICE IN KSEC
JMP COMM

SUBROUTINE TO CHANGE THE BUFFER

; STARTING LOCATION

CHMOD: LIX D, P4 ; SEND THE PROMPT
CALL STRSEND
CALL BTGET ; GET THE MSB OF THE CHOICE
STA KMOD+1 ; STORE IT IN KMOD+1
CALL BTGET ; GET THE LSB OF THE CHOICE
STA KMOD ; STORE IT IN KMOD
JMP COMM

SUBROUTINE TO READ A BYTE FROM THE KEYBOARD

; READ I : CHARACTER
CALL NTRY
CALL BKVRAT ; CONVET IT TO A HEX NUMBER
PUSH PSW ; SAVE IT
MVI C, I ; READ NEXT CHARACTER
CALL NTRY
CALL BKVRAT ; CONVET IT TO A HEX NUMBER
MOV L, A ; SAVE IT IN L
POP PSW ; RESET FIRST CHARACTER
RCL ; MOVE IT BY 1A
ALC
RCL 11111111B ; ZERO THE LOW NYBLE
ADC L ; ADD IT TO THE OTHER CHARACTER
RT

; SUBROUTINE TO CONVERT AN ASCII CHARACTER INTO
; A HEX DIGIT
BKVRAT: CPI 41H ; IF THE CHARACTER IS A NUMBER...
JC BNUM ; JUMP AHEAD
SUI 37H ; OTHERWISE SUBTRACT 37H
RET

; BNUM: SUI 38H ; SUBTRACT 30H FROM IT
RET

; SUBROUTINE TO CONVET THE PARAMETERS

SVAR: LDA KDRV ; GET DRIVE CHOICE
STA SDRV ; SAVE DRIVE CHOICE
LDA KTRK ; GET TRACK CHOICE
STA STRK ; SAVE TRACK CHOICE
LDA KSEC ; GET SECTOR CHOICE
STA SSEC ; SAVE SECTOR CHOICE
LDA KMOD+1 ; GET BUFFER LOCATION CHOICE (LSB)
STA SDMA ; GET BUFFER LOCATION CHOICE (LSB)
LDA KMOD+2 ; GET BUFFER LOCATION CHOICE (MSB)
STA SDMA+1 ; SAVE BUFFER LOCATION CHOICE (MSB)
JMP COMM

; SUBROUTINE WHICH RESTORES THE PARAMETERS

SJV: LDA KDRV ; GET SAVED DRIVE
STA KDRV ; RESTORE DRIVE
LDA KTRK ; GET SAVED TRACK
STA KTRK ; RESTORE TRACK
LDA KSEC ; GET SAVED SECTOR
STA KSEC ; RESTORE SECTOR

LDA SDMA ; GET SAVED BUFFER LOC. (LSB)
STA SDMA+1 ; RESTORE BUFFER LOC. (MSB)
JMP COMM

; QUIT: JMP 0000H

; START OF THE DATA BLOCK:

; COMMANDS:

DB 13, 10, 'READ ERROR', 13, 10, 'S'
DB 13, 10, 'SELECTED PARAMETERS:',
DB 13, 10, 'SELECTED DRIVE IS: ', 'S'
DB 13, 10, 'SELECTED TRACK IS: ', 'S'
DB 13, 10, 'SELECTED SECTOR IS: ', 'S'
DB 13, 10, 'BUFFER STARTING ADDRESS IS: ', 'S'
DB 13, 10, 'WHICH DRIVE (A OR B)?', 13, 10, 'S'
DB 13, 10, 'WHICH TRACK (IN 2 DIGIT HEX PLEASE)?', 13, 10, 'S'
DB 13, 10, 'WHICH SECTOR (IN 2 DIGIT HEX PLEASE)?', 13, 10, 'S'
DB 13, 10, 'HOW MANY SECTORS DO YOU WANT TO WRITE?',
DB 13, 10, 'HOW MANY SECTORS DO YOU WANT TO READ?',
DB 13, 10, 'HOW MANY SECTORS DO YOU WANT TO WRITE?',
DB 13, 10, 'WRITE SECTORS FROM DISK TO BUFFER?', 13, 10
DB 13, 10, 'RESTORE SECTOR(S) FROM BUFFER TO DISK?', 13, 10
DB 13, 10, 'RETURN TO COMMAND MODE?', 12, 10, 10
DB 13, 10, 'RETURN TO COMMAND MODE?', 12, 10, 10

; END: