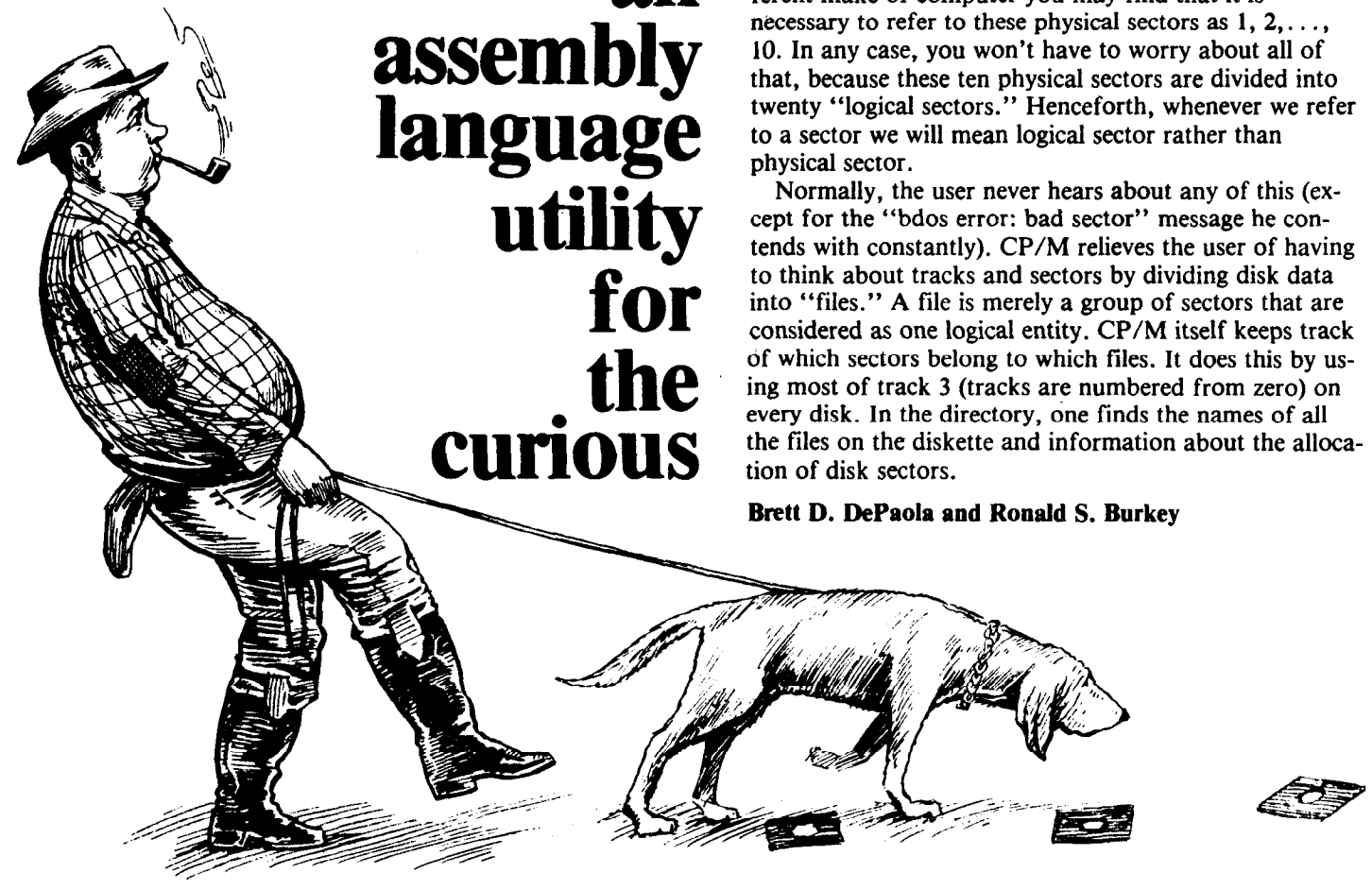
Disk snooping: an assembly language utility for the curious



(In preparing content for each issue of The Portable Companion, *we try to find something for both the novice and advance computer user. If you're a novice, be advised that the following article wasn't intended for you. This is an article for the computer literates among our readers.)*

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 thay 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 curiousity 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 are using some smaller system the initial "EQU" statements must be changed to reflect that fact.) Run DISKMON by typing:

DDT DISKMON.COM

(preceding either or both "DDT" and "DISKMON .COM" by the appropriate drive name.) At this point, you are in DDT and may perform any DDT function except those that will destroy DISKMON. which is located at hexadecimal addresses 0100-05FF. In particular, do not load any other files using the DDT I and R commands. To perform somedirect disk input/output type:

G100,105

which sends you to the DISKMON command menu.

Now you can do any or all of the following:

A) Choose the drive on which future diskette input/output is to occur.
B) Choose the starting track number for such operations.
C) Choose the starting sector number for such operations.
D) Choose the starting memory address for such operations

Options A-D set parameters to be used when the diskette is actually read (option G) or written to (option H). The read and write commands, explained below, update the sector number, track number, and memory location as they work. These parameters are printed on the screen every time the command menu is presented. Further options are:

E) Save present parameters.
F) Restore old paramenters.

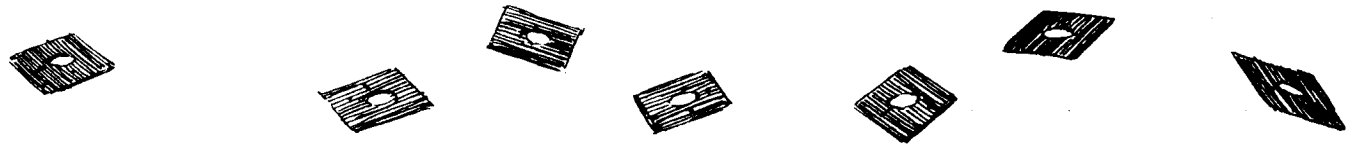
Options E-F are a convenience when you are continually using the same sector number, track number, and memory location. The final options are:

G) Read sector(s).
H) Write sector(s).
I) Go to DDT.
J) Go to CP/M command mode.

A typical session might go something like this: set into DISKMON as described above. Use options A-D to set drive = B, track = 00, sector = 00, memory address = 0600. (Incidentally, all numbers in all options are two-digit hexadecimal, except memory addresses, which are four-digit hexadecimal.) Now use option E to save this parameter pattern. Strictly speaking, theses steps would not be necessary in this example,since these are the default setting. Use option G to read 60 (decimal) sectors. (Option G prompts you for the number of sectors, so you request 3C, which is the hexadecimal equivalent of 60.) You would now have tracks 0-2 in memory, starting at 0600 (hex). Use option I to get into DDT. While in DDT, examine and modify. Type G100,105 to get back into DISKMON. Use option F to restore the parameter values *track = 00, sector = 00, memory address = 0600.* Use option H to save 60 sectors. Use option J to quit.

Finally, a word of warning! This utility is very dangerous, particularly the sector-write operation. Never use DISKMON on a diskette for which you have no backup unless it is absolutely unavoidable, and only then 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.



```
;
;       DISKMONITOR UTILITY
;
RDSEC   EQU     0E527H          ;BIOS ENTRY POINT FOR READING A SECTOR
WRSEC   EQU     0E52AH          ;BIOS ENTRY POINT FOR WRITING TO A SECTOR

SELDSK  EQU     0E51BH          ;SELECT A DISK DRIVE
SETDMA  EQU     0E524H          ;SET READ/WRITE BUFFER ADDRESS
SETTRK  EQU     0E51EH          ;SET TRACK TO READ/WRITE
SETSEC  EQU     0E521H          ;SET SECTOR TO READ/WRITE
NTRY    EQU     0005H
        ORG     0100H
        JMP     COMM
        ORG     0105H
;
DDTGO:  RST     7               ;RETURN TO DDT
;
;       THE COMMAND MODE - CONTROLS THE MAIN MENU
;
COMM:   CALL    CHKPRM          ;PRINT THE EXISTING PARAMETER VALUES
        LXI     D,CMENU         ;READY THE COMMAND MENU
        MVI     C,9             ;PREPARE TO PRINT IT
        CALL    NTRY            ;DO IT!
        MVI     C,1
        CALL    NTRY            ;GET COMMAND
        SUI     41H
        CPI     26              ;IF A () 0,1,2, ... ,25
        JNC     COMM            ; GOTO COMM
        CMC                     ;CLEAR THE CARRY
        RAL                     ;A=0,2,4, ... ,50
        MOV     E,A             ;PUT A INTO DE
        MVI     D,0
        LXI     H,JMPTBL        ;HL=JUMP-TABLE
        DAD     D               ;HL=JUMP-ADDRESS
        XCHG                    ;DE=JUMP-ADDRESS
        LDAX    D
        MOV     L,A             ;PUT THE CONTENTS
        INX     D               ; OF THE JUMP-ADDRESS
        LDAX    D               ; INTO THE HL
        MOV     H,A             ; REGISTER PAIR
        PCHL                    ;JUMP BY SWITCHING HL AND SP
;
;       SUBROUTINE TO WRITE TO THE DISK
;
WRITE:  LXI     D,P5            ;SEND THE PROMPT
        CALL    STRSND
        CALL    BTGET           ;READ THE # OF SECTORS TO WRITE
        PUSH    PSW             ;SAVE THIS NUMBER!
LOOPW:  POP     PSW             ;REGET THIS NUMBER
        CPI     00              ;QUIT IF # OF SECTORS =0
        JZ      COMM
        DCR     A               ;DECREMENT # OF SECTORS TO WRITE
        PUSH    PSW             ;RESAVE THIS NUMBER
        CALL    PREPAR          ;"PREPARE" PARAMETERS FOR WRITING
        CALL    WRSEC           ;WRITE!
        CALL    MEMFIX          ;ADJUST THE BUFFER STARTING LOCATION
        CALL    PRFIX           ; AND THE SECTOR # FOR THE NEXT WRITE
        JMP     LOOPW           ;RETURN TO DDT
;
;       SUBROUTINE TO READ THE DISK
;
READ:   LXI     D,P6            ;SEND THE PROMPT
        CALL    STRSND
        CALL    BTGET           ;GET THE # OF SECTORS TO READ
        PUSH    PSW             ;SAVE THIS NUMBER
LOOPR:  POP     PSW             ;BEGIN READ LOOP
        CPI     00              ;IF THE # OF SECTORS LEFT TO READ
        JZ      COMM            ; IS 0 THEN RETURN TO COMMAND MODE
        DCR     A               ;REDUCE THE # OF SECTORS LEFT TO READ
        PUSH    PSW             ; BY 1 AND SAVE THIS NUMBER
        CALL    PREPAR          ;"PREPARE" PARAMETERS FOR READING
        CALL    MEMFIX          ;ADJUST THE BUFFER STARTING LOCATION
        CALL    PRFIX           ; AND THE SECTOR # FOR THE NEXT READ
LOOP:   CALL    RDSEC           ;READ'
        CPI     1               ;IF A=1 THEN THERE WAS A READ ERROR
        CZ      ERROR           ;IN THAT CASE PRINT ERROR MSG
        CPI     0FFH            ;IF A=FF THEN DISK WAS "BUSY"
        JZ      LOOP            ;IN THAT CASE TRY AGAIN
        JMP     LOOPR           ;RETURN TO DDT
;
;       ROUTINE TO INCREMENT THE BUFFER STARTING
;       LOCATION BY 128 BYTES
;
MEMFIX: LHLD    KDMA            ;PUT THE EXISTING STARTING LOCATION INTO
        LXI     D,128           ; HL
        DAD     D               ;THEN ADD 128 TO IT
        SHLD    KDMA            ;PUT THE NEW STARTING ADDRESS BACK INTO
        RET                     ; KDMA, THEN QUIT
;
;       SUBROUTINE WHICH UPDATES THE OTHER PARAMETERS
;
PRFIX:  LDA     KSEC            ;UPDATE CURRENT SECTOR NUMBER
        INR     A
        CPI     20              ;IF IT IS = TO 20 THEN SET IT
        JNZ     PRCONT          ; TO 0 AND INCREMENT THE TRACK #
        LDA     KTRK            ; BY 1 OTHERWISE JUMP TO PRCONT
        INR     A               ; AND THEN RETURN
        STA     KTRK
        MVI     A,00
PRCONT: STA     KSEC
        RET
;
;       SUBROUTINE WHICH PRINTS A READ ERROR MESSAGE AND THE
;       CURRENT PARAMETERS WHEN CALLED
;
ERROR:  LXI     D,MES           ;PREPARE TO SEND ERROR MESSAGE
        CALL    STRSND          ;SEND IT!
        CALL    CHKPRM          ;DISPLAY PARAMETERS
        RET
;
;       SUBROUTINE WHICH PREPARES FOR A DISK READ OR WRITE
;
PREPAR: LDA     KDRV            ;PUT DRIVE # IN A (0 IS A), 1 IS B))
        MOV     C,A             ;A -> C
        CALL    SELDSK          ;SET THE DISK!
        LDA     KTRK            ;PUT TRACK # IN A
        MOV     C,A             ;A -> C
        CALL    SETTRK          ;SET THE TRACK!
        LDA     KSEC            ;PUT SECTOR # IN A
        MOV     C,A             ;A -> C
        CALL    SETSEC          ;SET THE SECTOR!
        LHLD    KDMA            ;PUT THE STARTING ADDRESS OF READ/WRITE
        MOV     B,H             ; BUFFER INTO A
        MOV     C,L
        CALL    SETDMA          ;SET THE BUFFER!
        RET
;
;       SUBROUTINE WHICH DISPLAYS THE CURRENT PARAMETERS
;
CHKPRM: LXI     D,STR1          ;SEND DRIVE # MESSAGE
        CALL    STRSND
        LDA     KDRV            ;SEE WHAT'S IN KDRV
        CPI     00              ;IF IT'S A 00 THEN LEAP
        JZ      ALOOP           ; AHEAD TO ALOOP
        MVI     E,'B'           ;OTHERWISE PREPARE TO PRINT A "B"
        JMP     AOUT            ;SKIP AROUND THIS PART
ALOOP:  MVI     E,'A'           ;PREPARE TO PRINT AN "A"
AOUT:   MVI     C,2             ;PRINT WHATEVER WAS PREPARED
        CALL    NTRY
        LXI     D,STR2          ;SEND TRACK # MESSAGE
        CALL    STRSND
        LDA     KTRK            ;SEE WHAT'S IN KTRK
        CALL    GETR
        LXI     D,STR3          ;SEND SECTOR # MESSAGE
        CALL    STRSND
        LDA     KSEC            ;SEE WHAT'S IN KSEC
        CALL    GETR
        LXI     D,STR4          ;SEND BUFFER LOCATION MESSAGE
        CALL    STRSND
        LDA     KDMA+1          ;SEE WHAT'S IN KDMA+1
        CALL    GETR
        LDA     KDMA            ;SEE WHAT'S IN KDMA
GETR:   CALL    CNVRT           ;CONVERT IT TO 2 ASCII
        MVI     C,2             ; CHARACTERS
        MOV     E,H             ; AND SEND THEM TO THE CONSOLE
        PUSH    B               ;PRESERVE REGISTERS
        PUSH    H
        CALL    NTRY            ;PRINT!
        POP     H               ;REGET REGISTERS
        POP     B
        MOV     E,L             ;PRINT THE SECOND BYTE
        CALL    NTRY
        RET
;
;       SUBROUTINE TO SEND STRINGS TO THE CONSOLE
;
STRSND: MVI     C,9             ;9 IS SYSTEM FUNCTION FOR SENDING
        CALL    NTRY            ; STRINGS
        RET
;
;       SUBROUTINE TO "CONVERT" 1 HEX BYTE INTO 2 ASCII BYTES
;       AND STORES THEM IN HL
;
CNVRT:  PUSH    PSW             ;SAVE THE BYTE
        ANI     00001111B       ;ZERO THE HIGH NYBBLE
        CALL    DOVRT           ;CONVERT THE LOW NYBBLE
        MOV     L,A             ;STORE IT IN L
        POP     PSW             ;REGET THE BYTE
        RRC                     ;MOVE THE HIGH NYBBLE
        RRC                     ; TO WHERE THE LOW NYBBLE WAS
        RRC
        RRC
        ANI     00001111B       ;ZERO THE HIGH NYBBLE
        CALL    DOVRT           ;CONVERT THE LOW NYBBLE
        MOV     H,A             ;STORE IT IN H
        RET
;
;       SUBROUTINE TO GIVE THE ASCII FORM OF
;       A HEX NUMBER
;
DOVRT:  CPI     0AH             ;IF THE NUMBER IS LESS THAN
        JC      NUM1            ; 0A THEN GOTO NUM1
        ADI     37H             ;OTHERWISE ADD 37H
        RET
NUM1:   ADI     30H             ;ADD 30H TO THE NUMBER
        RET
;
;       SUBROUTINE TO CHANGE THE DRIVE
;
CHDR:   LXI     D,P1            ;SEND THE PROMPT
        CALL    STRSND
        MVI     C,1             ;READ THE DRIVE CHOICE
        CALL    NTRY
        CPI     41H             ;IF IT'S AN 'A' ...
        JZ      CHDRA           ; THEN JUMP AHEAD
        MVI     A,01            ;OTHERWISE MAKE IT A 'B'
        JMP     SNDDSK
CHDRA:  MVI     A,00            ;MAKE IT AN 'A'
SNDDSK: STA     KDRV            ;STORE CHOICE IN KDRV
        JMP     COMM
;
;       SUBROUTINE TO CHANGE THE TRACK
;
CHTRK:  LXI     D,P2            ;SEND THE PROMPT
        CALL    STRSND
```

```
        CALL    BTGET           ;READ THE TRACK CHOICE
        STA     KTRK            ;STORE CHOICE IN KTRK
        JMP     COMM
;
;       SUBROUTINE TO CHANGE THE SECTOR
;
CHSEC:  LXI     D,P3            ;SEND THE PROMPT
        CALL    STRSND
        CALL    BTGET           ;READ THE SECTOR CHOICE
        STA     KSEC            ;STORE CHOICE IN KSEC
        JMP     COMM
;
;       SUBROUTINE TO CHANGE THE BUFFER
;       STARTING LOCATION
;
CHDMA:  LXI     D,P4            ;SEND THE PROMPT
        CALL    STRSND
        CALL    BTGET           ;GET THE MSB OF THE CHOICE
        STA     KDMA+1          ;STORE IT IN KDMA+1
        CALL    BTGET           ;GET THE LSB OF THE CHOICE
        STA     KDMA            ;STORE IT IN KDMA
        JMP     COMM
;
;       SUBROUTINE TO READ A BYTE FROM THE KEYBOARD
;
BTGET:  MVI     C,1             ;READ 1 CHARACTER
        CALL    NTRY
        CALL    BKVRT           ;CONVERT IT TO A HEX NUMBER
        PUSH    PSW             ;SAVE IT
        MVI     C,1             ;READ NEXT CHARACTER
        CALL    NTRY
        CALL    BKVRT           ;CONVERT IT TO A HEX NUMBER
        MOV     L,A             ;SAVE IT IN L
        POP     PSW             ;REGET FIRST CHARACTER
        RLC                     ;MULTIPLY IT BY 16
        RLC
        RLC
        RLC
        ANI     11110000B       ;ZERO THE LOW NYBBLE
        ADC     L               ;ADD IT TO THE OTHER CHARACTER
        RET
;
;       SUBROUTINE TO CONVERT AN ASCII CHARACTER INTO
;       A HEX DIGIT
;
BKVRT:  CPI     41H             ;IF THE CHARACTER IS A NUMBER...
        JC      BKNUM           ; JUMP AHEAD
        SUI     37H             ;OTHERWISE SUBTRACT 37H
        RET
BKNUM:  SUI     30H             ;SUBTRACT 30H FROM IT
        RET
;
;       SUBROUTINE TO SAVE THE PARAMETERS
;
SAVPR:  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     KDMA            ;GET BUFFER LOCATION CHOICE (LSB)
        STA     SDMA            ;SAVE BUFFER LOCATION CHOICE (LSB)
        LDA     KDMA+1          ;GET BUFFER LOCATION CHOICE (MSB)
        STA     SDMA+1          ;SAVE BUFFER LOCATION CHOICE (MSB)
        JMP     COMM
;
;       SUBROUTINE WHICH RESTORES THE PARAMETERS
;       SAVED BY SAVPR
;
RESPR:  LDA     SDRV            ;GET SAVED DRIVE
        STA     KDRV            ;RESTORE DRIVE
        LDA     STRK            ;GET SAVED TRACK
        STA     KTRK            ;RESTORE TRACK
        LDA     SSEC            ;GET SAVED SECTOR
        STA     KSEC            ;RESTORE SECTOR
        LDA     SDMA            ;GET SAVED BUFFER LOC. (LSB)
        STA     KDMA            ;RESTORE BUFFER LOC. (LSB)
        LDA     SDMA+1          ;GET SAVED BUFFER LOC. (MSB)
        STA     KDMA+1          ;RESTORE BUFFER LOC. (MSB)
        JMP     COMM
;
QUIT:   JMP     0000H
;
;       START OF THE DATA BLOCK:
;
KDRV:   DB      1
KTRK:   DB      0
KSEC:   DB      0
KDMA:   DW      0600H
SDRV:   DB      1
STRK:   DB      0
SSEC:   DB      0
SDMA:   DW      0600H
MES:    DB      13,10,'READ ERROR',13,10,'$'
STR1:   DB      13,10,'    SELECTED PARAMETERS:'
        DB      10,10,13,'SELECTED DRIVE IS:  ','$'
STR2:   DB      13,10,'SELECTED TRACK IS:   ','$'
STR3:   DB      13,10,'SELECTED SECTOR IS:   ','$'
STR4:   DB      13,10,'BUFFER STARTING ADDRESS IS:   ','$'
P1:     DB      13,10,'WHICH DRIVE (A OR B)?',13,10,'$'
P2:     DB      13,10,'WHICH TRACK (IN 2 DIGIT HEX PLEASE)?',13,10,'$'
P3:     DB      13,10,'WHICH SECTOR (IN 2 DIGIT HEX PLEASE)?',13,10,'$'
P4:     DB      13,10,'BUFFER STARTING ADDRESS: ',13,10,'$'
P5:     DB      13,10,'HOW MANY SECTORS DO YOU WANT TO WRITE?'
        DB      13,10,'(2 DIGIT HEX PLEASE)',13,10,'$'
P6:     DB      13,10,'HOW MANY SECTORS DO YOU WANT TO READ?'
        DB      13,10,'(2 DIGIT HEX PLEASE)',13,10,'$'
CMENU:  DB      13,10,10,10,'COMMAND-MODE :  OPTIONS',13,10,10
        DB      'A=CHANGE DRIVE',13,10
        DB      'B=CHANGE TRACK',13,10
        DB      'C=CHANGE SECTOR',13,10
        DB      'D=CHANGE BUFFER STARTING ADDRESS',13,10
        DB      'E=PRESERVE PARAMETERS',13,10
        DB      'F=RESTORE PARAMETERS',13,10
        DB      'G=READ SECTOR(S) FROM DISK TO BUFFER',13,10
        DB      'H=WRITE SECTOR(S) FROM BUFFER TO DISK',13,10
        DB      'I=JUMP TO DDT',13,10
        DB      'J=RETURN TO CP/M COMMAND MODE',13,10,10
        DB      '$'
JMPTBL: DW      CHDR
        DW      CHTRK
        DW      CHSEC
        DW      CHDMA
        DW      SAVPR
        DW      RESPR
        DW      READ
        DW      WRITE
        DW      DDTGO
        DW      QUIT
        DW      COMM
        DW      COMM
        DW      COMM
        DW      COMM
        DW      COMM
        DW      COMM
        DW      COMM
        DW      COMM
        DW      COMM
        DW      COMM
        DW      COMM
        DW      COMM
        DW      COMM
        DW      COMM
        DW      COMM
        DW      COMM
        DW      COMM
        DW      COMM
        DW      COMM
        END
```