

TR cpm source

•TITLE CPM

•=\$2900 ;STANDARD 16K START ADDRESS

•UNIT.=4 ;WE TELL USER WHAT THE DEFAULT UNIT IS, HERE.

•BDOS.=5 ;VECTOR ADDRESS FOR BDOS CALLS

•FCB.=55C ;WE BUILD AN FCB FOR USER FROM CMD STRING, HERE

ASCII CHARACTERS

CH•NUL=0 ;NULL FOR ASCIIZ
 CH•ATN=\$03 ;CONTROL C FOR ATTENTION
 CH•TAB=\$09 ;HORIZ TAB CHAR
 CH•LF=\$0A ;LINEFEED
 CH•CR=\$0D ;CARRIAGE RETURN
 CH•XOF=\$13 ;STOP OUTPUT
 CH•EOF=\$1A ;END OF FILE (^Z)
 CH•CTL=\$20 ;CONTROL CHARS ARE THOSE LESS THAN THIS
 CH•SPC=\$20 ;SPACE
 CH•NUM=\$23 ;NUMBERSIGN
 CH•DOL=\$24 ;DOLLARSIGN FOR STRING TERMINATOR
 CH•CLN=\$3A ;COLON
 CH•PRM=\$3E ;PROMPT CHARACTER
 CH•CM=\$3F ;WILDCARD CHAR, QUESTIONMARK
 CH•A=\$41 ;CHARACTER CAPITAL A
 CH•UPA=\$5E ;UPARROW FOR TYPING CTL CHARS
 CH•LCA=\$0141 ;LOWER CASE A,Z
 CH•LCZ=\$0172 ;RUBOUT
 CH•RUB=\$7F ;MASK FOR 7-BIT CHARACTER

IO CORE ADDRESSES

DEFBFR=\$0280 ;DEFAULT DMA ADDR BUFFER
 •TPA.= \$0120 ;TRANSIENT PROGRAM AREA
 •CCP.= \$2900 ;CONSOLE COMMAND PROCESSOR

COMMANDS TO BDOS

BF•TYI=1 ;TYPE IN A CHAR
 BF•TYO=2 ;TYPE OUT A CHAR
 BF•RDR=3 ;GET CHAR FROM READER
 BF•PCH=4 ;SEND CHAR TO PUNCH
 BF•LPT=5 ;SEND CHAR TO LISTING DEVICE
 BF•ADR=6 ;GET ADDR OF BDOS
 BF•GST=7 ;GET I/O STATUS BYTE
 BF•SST=8 ;SET STATUS BYTE
 BF•SCU=9 ;STRING OUTPUT
 BF•SIN=10 ;STRING INPUT
 BF•VBS=11 ;KEYBOARD STATUS CHECK

IFOLLOWING ARE THE DISK ONES

BF•LHD=12 ;LIFT HEAD
 BF•INI=13 ;INIT BDOS, SET DMA TO 80, LOG IN DSK A
 BF•LOG=14 ;LOG IN DISK N

```

BF.CLS=16      ;CLOSE FILE
BF.FND=17      ;FIND FILE
BF.FNX=18      ;FIND NEXT FILE
BF.DEL=19      ;DELETE FILE
BF.RED=20      ;READ NEXT RECORD
BF.WRT=21      ;WRITE NEXT RECORD
BF.CRE=22      ;CREATE NEW FILE
BF.REN=23      ;RENAME FILE
BF.ILV=24      ;INTERROGATE LOGIN VECTOR
BF.ILD=25      ;INTERROGATE LOGGED DRIVE NUMBER
BF.DMA=26      ;SET DMA ADDRESS
BF.IAV=27      ;INTERROGATE ALLOCATION VECTOR FOR CURRENT DISK

```

LAYOUT OF AN FCB (FILE CONTROL BLOCK)

```

FCB.UN=$00     ;UNIT NUMBER, IF FORCED
FCB.NM=$01     ;NAME, 8 CHARS
FCB.EX=$09     ;EXTENSION, 3 CHARS
FCB.RC=$0F     ;RECORD COUNT
FCB.XP=$10     ;MAP AREA
FCB.NR=$20     ;NUMBER OF RECORD

NDISKS=4       ;MAX NUMBER OF DISKS ON SYSTEM

XCLUST=$100    ;CLUSTERS ON A DISK, ROUNDED UP
XALLOV=XCLUST/8 ;BYTES OF ALLOC VECTOR FOR A DISK. (8 BITS/BYTE)
NAMLEN=8       ;CHARACTERS IN A FILE NAME
EXTLEN=3       ;CHARACTERS IN A FILENAME EXTENSION
SECTOR=$00     ;CHARACTERS IN A SECTOR

```

```

CCP:    JMP     CCP000      ;MAIN ENTRY TO CCP
        JMP     CCPREE      ;RESTART, CLEARING CMD BUFFER

```

TEXT BUFFER USED FOR COMMAND TYPEIN.
IT IS PRE-LOADED WITH A COPYRIGHT STATEMENT

NCMDBF=\$D128 ;SIZE OF BUFFER

```

CMDBUF: .BYTE    NCMDBF-1      ;MAXIMUM COUNT OF CHARS IN BUFFER
CBUFCT: .BYTE    2             ;CURRENT COUNT IN BUFFER
CBUFTX: .ASCII   /             ;SIXTEEN SPACES
        .ASCII   /COPYRIGHT (C) 1978, DIGITAL RESEARCH /
        .BLKB    NCMDBF-CBUFTX-

```

CMDPTR: .ADDR CBUFTX ;PARSER SCAN POINTER

ERRPTR: .BLKW 1 ;POINTER TO TYPE OUT BAD PART IF ERROR

```

CCPTYO: MOV E,    A           ;CHAR TO TYPE
        MVI C,    BF.TYO      ;BDOS TYO FUNCTION
        JMP      .BDOS.       ;CALL BDOS

```

```

TYOSBC: PUSH     B           ;SAME, BUT SAVING BC PAIR
        CALL     CCPTYO      ;CALL TYO ROUTINE
        POP      B           ;RESTORE BC PAIR
        RET

```

TCRLF:	MVI A, CH.CR	;TYPE CARR RET, LINEFEED	
	CALL CCPTYO		
	MVI A, CH.LF	;LINEFEED	
	JMP CCPTYO		
CRSCUT:	PUSH B	;SAVE POINTER TO ASCII2	
	CALL TCRLF	;FIRST A CRLF	
	POP H	;RESTORE POINTER TO ASCII2	
CCPSOU:	MOV A, M	;GET A CHAR FROM STRING	
	ORA A	;IS IT A NULL?	
	RZ	;IF SO, END OF STRING	
	INX H	;NOT NULL, STEP PAST IT	
	PUSH H	;SAVE FOR NEXT LOOP	
	CALL CCPTYO	;TYPE THIS CHAR	
	POP H	;POINT TO NEXT ONE	
	JMP CCPSOU	;LOOP FOR MORE	
INIDOS:	MVI C, BF.INI	;FUNCTION = INIT BDOS	
	JMP .BDOS.		
LOGDSK:	MOV E, A	;THIS IS THE DESIRED DISK	
	MVI C, BF.LOG	;LOG IN THIS DISK NUMBER	
	JMP .BDOS.		
OPNFIL:	MVI C, BF.OPN	;OPEN A FILE	
	CALL .BDOS.		
	STA FCBNUM	;SAVE THE BYTE ADDR IN DIR	
	INR A	;IF WAS \$FF, SET Z FLAG	
	RET	;RETURN Z FLAG TO CALLER	
OPNCMD:	XRA A	;START OF FILE	
	STA CMDFCB+FCB.NR		
	LXI D, CMDFCB	;POINT TO FILE BLOCK	
	JMP OPNFIL	;OPEN THE FILE	
CLSFIL:	MVI C, BF.CLS	;CLOSE FILE	
	CALL .BDOS.		
	STA FCBNUM		; 29D7 32 C8 30
	INR A		; 29DA 3C
	RET		; 29DB C9
FNOFIL:	MVI C, BF.FND	;FIND FILE	
	CALL .BDOS.		
	STA FCBNUM		; 29E1 32 C8 30
	INR A		; 29E4 3C
	RET		; 29E5 C9
FNDNXT:	MVI C, BF.FNX	;FIND NEXT FILE IN WILDCARD GROUP	
	CALL .BDOS.		
	STA FCBNUM		; 29EB 32 C8 30
	INR A	;RETURN Z FLAG IF NO MORE.	
	RET		
FND CFL:	LXI D, CMDFCB		; 29F0 11 A7 30
	JMP FNOFIL	;FIND FILE	
DELFIL:	MVI C, BF.DEL	;DELETE FILE	
	JMP .BDOS.		
READFL:	MVI C, BF.RED	;READ NEXT RECORD	

CALL .BDOS.
ORA A
RET

RE CMD: LXI D, CMDFCB ;FCB OF CURRENT COMMAND
JMP READFL ;READ ANOTHER SECTOR

WRITEFL: MVI C, BF.WRT ;WRITE NEXT RECORD
CALL .BDOS.
ORA A
RET

CREATEF: MVI C, BF.CRE ;CREATE FILE
CALL .BDOS.
STA FCBNUM
RET

RENAMEF: MVI C, BF.REN ;RENAME FILE
JMP .BDOS.

CHANGE: CPI CH.LCA ;RANGE CHECK. LOWER CASE A?
RC ;RETURN IF SMALLER
CPI CH.LCZ+1 ;LOWER CASE Z?
RNC ;RETURN IF BIGGER
ANI \$5F ;MAKE UPPER CASE
RET

HERE TO GET A COMMAND, FROM TOP LEVEL COMMAND LOOP.

SELECT: LDA SUBFLG ;SUBMIT FILE IN PROGRESS?
ORA A
JZ COLEC2 ;JUMP IF NOT.
LDA CMDUNI ;LOGGED DISK AT COMMAND LEVEL
ORA A
MVI A, 0 ;SELECT "A"
CNZ LOGDSK ;IF NOT DISK A, SWITCH TO A.
LDA FCBSUB+FCB.RC ;RECORD COUNT
DCR A ;LESS ONE
STA FCBSUB+FCB.NR
LXI D, FCBSUB ;FCB OF SUBMIT NAME
CALL READFL ;READ ANOTHER SECTOR
JNZ COLEC2 ;IF ERROR ON READ, STOP.
LXI D, CBUFTX ;MOVE BUFFER TO HERE
LXI H, DEFBFR ;MOVE BUFFER FROM HERE
MVI B, NSECCH ;MOVE THIS MANY BYTES
CALL MOVNCH ;MOVE STRING
LXI H, FCBSUB+FCB.RC. ; 2A51 21 95 30
DCR H ; 2A54 35
LXI D, FCBSUB ;FCB FOR SUBMIT NAME
CALL CLSFIL ;CLOSE FILE
JZ COLEC2 ; 2A5B CA 77 2A
LDA CMDUNI ;COMMAND LEVEL DRIVE
ORA A ;SEE IF DRIVE A
CNZ LOGDSK ;IF NOT, SWITCH TO WHATEVER IT IS
LXI H, CBUFTX ;POINT TO THE TEXT
CALL CCPSOU ;SHOW THE SUB TEXT
CALL CHKKBS ;ANYTHING TYPED BY OPERATOR?
JZ COLEC3 ;IF NOT, USE SUBMIT STUFF
CALL DELSUB ;IF SO, DELETE SUB FILE.
JMP PROMPT ;AND GO TO CMD PROMPT


```

COLEC2: CALL    DELSUB      ;DELETE ANY SUB FILE
        MVI C,    BF.SIN    ;STRING INPUT FROM TTY
        LXI D,    CMDBUF    ;HERE'S WHERE WE WANT TEXT
        CALL     .BDOS.     ;DO TEXT
COLEC3: LXI H,    CBUFCT    ;SEE HOW MANY CHARS WE GOT
        MOV B,    M         ;HERE'S THE COUNT
COLEC4: INX      H         ;LOOP THRU TEXT, RAISING IT.
        MOV A,    B         ;COUNT LEFT
        ORA      A         ;ANY LEFT?
        JZ       CCLECS    ;QUIT WHEN ALL DONE
        MOV A,    M         ;GET A CHAR
        CALL     RAISE     ;IF LOWER CASE ALPHA, RAISE IT
        MOV M,    A         ;PUT UPPER CASE IN BUFFER
        DCR      B         ;COUNT THRU BUFFER
        JMP      COLEC4    ;LOOP

COLEC5: MOV M,    A         ;MAKE IT ASCIZ - NULL AFTER TEXT
        LXI H,    CBUFTX    ;POINT AT START OF TEXT
        SHLD     CMDPTR     ;SCAN STARTS HERE
        RET              ;LINE READY FOR PROCESSING

CHKKBS: MVI C,    BF.KBS    ;CHECK KEYBOARD STATUS
        CALL     .BDOS.
        ORA      A
        RZ              ;NOTHING TYPED
        MVI C,    BF.TYI    ;GET THE TYPED CHAR
        CALL     .BDOS.
        ORA      A         ;MAKE NON-ZERO FLAG (UNLESS NULL)
        RET

LIFTHD: MVI C,    BF.LHD    ;LIFT HEAD (NOP)
        JMP      .BDOS.

GETLDN: MVI C,    BF.ILD    ;GET LOGGED DISK NUMBER
        JMP      .BDOS.

;UNUSED ROUTINE TO MOVE (C) CHARS FROM (DE) TO (HL)

DOBLT:  LDAX     D
        MOV M,    A
        INX      D
        INX      H
        DCR      C
        JNZ     DOBLT
        RET

SETADR: MVI C,    BF.DMA    ;SET DMA ADDRESS
        JMP      .BDOS.
; 2ACD C3 05 00

DELSUB: LXI H,    SUBFLG    ;WAS SUB FILE BEING DONE?
        MOV A,    M
        ORA      A
        RZ              ;RETURN IF NOT
        MVI M,    0         ;IT WAS, BUT NO MORE.
        XRA      A         ;CHOOSE DRIVE A
        CALL     LOGDSK     ;LOG IT IN
        LXI D,    FCBSUB    ;SUBMIT FILE'S FCB

```

```

        JMP      LOGDSK      ;AND RETURN

CHKSER: LXI D, CCPSER      ;MAKE SURE SERIALS MATCH
        LXI H, DOSSER      ; BETWEEN CCP AND BDOS
        MVI B, SERSIZ      ;CHECK THIS MANY CHARS

CHKSRJ: LDAX D
        CMP M
        JNZ SERERR        ;IF NOT, GO STEP ON SELF
        INX D
        INX H
        DCR B
        JNZ CHKSRL        ;LOOP THRU ALL OF SERIAL
        RET               ;IF MATCHES, RETURN OK

;REJECT COMMAND

CMDREJ: CALL TCRLF        ;NEW LINE
        LHLD ERRPTR      ;TYPE OUT THE BAD GUY
        MVI A, N         ; ..
        CPI CH.SPC
        JZ CMDRJ1
        ORA A
        JZ CMDRJ1
        PUSH H
        CALL CCPTYO      ;TYPE CHAR OF BAD CMD
        POP H
        INX H
        JMP CMDRJL       ;LOOP TIL NULL

CMDRJ1: MVI A, '?'        ;STICK "?" ON END
        CALL CCPTYO
        CALL TCRLF
        CALL DELSUB      ;DELETE SUB FILE, IF ANY
        JMP PROMPT       ;RESTART CCP

;L
;CHECK FOR TERMINATORS
; RETURN "Z" IF LEGAL SEPARATOR, ABORT ON CONTROLS.

CHKTRM: LDAX D
        ORA A
        RZ               ;RETURN Z IF NULL
        CPI CH.CTL       ;IF CONTROLS,
        JC CMDREJ        ; ABORT
        RZ               ;SPACE IS OK, THOUGH
        CPI '='          ;SC ARE THE FOLLOWING

        CPI '_'
        RZ
        CPI '.'
        RZ
        CPI CH.CLN      ;" "
        RZ
        CPI '['
        RZ
        CPI ']'
        RZ
        RET              ;IF NONE OF THOSE, RETURN NON-ZERO

```

```

NOTSPC: LDAX    D           ;RETURN "Z" ON NULL
        ORA     A
        RZ
CPI     $20           ;SCAN FOR NON-SPACE
        RNZ
INX     D
        JMP     NOTSPC

HL..A:  ADD     L           ;DA + HL => HL
        MOV L,  A
        RNC
INR     H
        RET

-L
;ROUTINE TO SCAN A WORD OR FILENAME

PARSE:  MVI A,  0           ;START AT BEGINNING TO BUILD WORD
PARSE1: LXI H,  CMDFCB      ;MAY BE A FILENAME, OR A VERB, ETC
        CALL    HL..A      ;ADD A TO HL
        PUSH    H          ;SAVE TWO COPIES
        PUSH    H
        XRA     A          ;ASSUME NOT A DISK SPEC
        STA     PRSUNI
        LHLD    CMDPTR     ;COMMAND POINTER
        XCHG                    ;TO DE
        CALL    NOTSPC     ;SCAN FOR NON-SPACE
        XCHG                    ;BACK TO HL
        SHLD    ERRPTR     ;IF ERROR, WILL RETYPE FROM HERE
                                ; 2B5D EB
                                ; 2B5E E1
        PCP     H
        LDAX    D          ;GET BACK THE NON-SPACE
        ORA     A          ;WAS NON-SPACE A NULL?
        JZ      PARSE2     ;JUMP IF SO
        SBI     CH..A-1
        MOV B,  A          ;MAY BE A DRIVE NAME
        INX     D          ;SEE IF A COLON IS NEXT
        LDAX    D
        CPI     CH..CLN    ;COLON?
        JZ      PARSE3     ;IF SO, THIS WAS A DRIVE
        DCX     D          ;WASN'T. BACK BEFORE COLON
        PARSE2: LDA     CMDUNI ;PLUG DEFAULT IN
        MOV M,  A
        JMP     PARSE4

PARSE3: MOV A,  B          ;WE HAD A <UNIT>:
        STA     PRSUNI     ;SO PLUG IT IN
        MOV M,  B
        INX     D          ;..
        ;STEP PAST UNIT AND COLON

PARSE4: MVI B,  NNAMCH
PARSE5: CALL    CHKTRM     ;LOOK FOR SPECIFIC TERMINATORS
        JZ      PARSE9     ;END OF WORD IF SO
        INX     H          ;NOT ONE OF THOSE. STEP PAST IT

        CPI     'X'        ;WILDCARD?
        JNZ     PARSE6     ;JUMP IF NOT
        MVI M,  '?'        ;YES, MAKE A WILD CHAR
        JMP     PARSE7     ;AND DON'T STEP PAST IT.

PARSE6: MOV M,  A          ;STORE CHAR FROM TEXT INOUT

```

```

PARSE7: DCR      B
        JNZ      PARSE5
PARSE8: CALL     CHKTRM
        JZ       PARS10
        INX      D
        JMP      PARSE8

```

```

;COUNT CHARS IN A NAME
;SOME LEFT, LOOP.
;SKIP TRAILING NON-TERMS
;QUIT AT TERM
;NON-TERM. SKIP IT.

```

```

PARSE9: INX      H
        MVI M, CH.SPC
        DCR      B
        JNZ      PARSE9

```

```

;FILL REST OF NAME WITH BLANKS

```

```

PARS10: MVI B, NEXTCH
        CPI      '.'
        JNZ      PARS15
        INX      D

```

```

;COUNT THRU EIGHT CHARS
;TILL ALL FILLED
;CHARS IN AN EXTENSION
;WAS TERM A DOT?
;JUMP IF NOT
;YES. PASS IT.

```

```

PARS11: CALL     CHKTRM
        JZ       PARS15
        INX      H
        CPI      '*'
        JNZ      PARS12
        MVI M, '?'
        JMP      PARS13

```

```

;CHECK CHARS IN EXTENSION
;GO IF TERMINATOR
;STEP DESTINATION
;WILD CARD EXTENSION?
;JUMP IF NOT
;YES, MAKE WILD CHAR
;AND DON'T SCAN PAST IT

```

```

PARS12: MOV M, A
        INX      D

```

```

;COPY CHAR FROM SOURCE
;STEP THRU EXTENSION

```

```

PARS13: DCR      B
        JNZ      PARS11

```

```

;COUNT EXTENSION
;LOOP TILL DONE

```

```

PARS14: CALL     CHKTRM
        JZ       PARS16
        INX      D
        JMP      PARS14

```

```

;SKIP TRAILING NON-TERMS
;JUMP ON TERM
;STEP SOURCE
;LOOK FOR TERM

```

```

PARS15: INX      H
        MVI M, CH.SPC
        DCR      B
        JNZ      PARS15

```

```

;STEP DESTINATION
;PAD WITH SPACES
;COUNT THRU EXTENSION
;FILL WHOLE EXT

```

```

PARS16: MVI B, $03
PARS17: INX      H
        MVI M, 0
        DCR      B
        JNZ      PARS17
        XCHG
        SHLD     CMDPTR
        POP      H

```

```

;FILL MORE OF SPARE BYTES OF FCB
;HERE
;WITH ZEROS
;COUNT THEM
;LOOP FOR THREE SPARE BYTES

```

```

        LXI B, NNAHCH+NEXTCH
PARS18: INX      H
        MOV A, M
        CPI      CH.OM
        JNZ      PARS19
        INR      B

```

```

;GET HOW MUCH READ FROM TEXT
;REMEMBER FOR NEXT SCAN
;BACK TO START OF FILENAME
;C GETS # TO SCAN, B GETS ANSWER

```

```

        INR      B
        INR      B
PARS19: DCR      C
        JNZ      PARS18
        MOV A, B
        ORA      A
        RET

```

```

;LOOK FOR WILD CARDS IN FILENAME
;GET A FILENAME CHAR
;WILD?
;IF NOT, DON'T COUNT IT
;YES, COUNT IT.

```

```

        DCR      C
        JNZ      PARS18
        MOV A, B
        ORA      A
        RET

```

```

;SCAN NAME AND EXT

```

```

;NUMBER OF WILD CHARS
;RETURN Z FLAG AS FAST CHECK

```

```

;CCP COMMAND TABLE

```

```

CCPCTB: .ASCII /DIR /
NCPCH=.CCPCTB

```

```

;CHARS IN A COMMAND NAME

```

.ASCII /ERA /
 .ASCII /TYPE/
 .ASCII /SAVE/
 .ASCII /REN /

COMMANDS IN TABLE

NCCPC=<.-CCPCTB>/NCCPCH

;SERIAL NUMBER OF COPY OF CCP
 ;SIZE OF SERIAL NUMBER

CCPSER: .BYTE 0,50E,0,0,1,2
 SERSIZ=-CCPSER

HERE TO LOOK UP A COMMAND VERB

FINDCM: LXI H, CCPCTB
 MVI C, 0

;POINT TO LEGAL COMMANDS
 ;THIS WILL BE INDEX INTO TABLE

FINDC1: MOV A, C
 CPI NCCPC
 RNC

;TRIED THEM ALL?
 ;GIVE UP IF SO
 ;LOOK AT USER'S STRING
 ;COMMAND IS THIS LONG

LXI D, CMDNAM
 MVI B, NCCPCH

FINDC2: LDAX D
 CMP M

;CHECK A CHARACTER
 ;NOT THIS COMMAND
 ;STEP THRU REAL AND TEST STRINGS

JNZ FINDC3
 INX D

INX H
 DCR B

;COUNT LENGTH OF COMMAND
 ;LOOP FOR WHOLE COMMAND
 ;CHECK USER'S CHAR AFTER COMMAND
 ;FOR SPACE

JNZ FINDC2
 LDAX D

CPI CH,SPC
 JNZ FINDC4

;JUMP IF NOT SPACE
 ;RETURN TABLE INDEX

MOV A, C
 RET

FINDC3: INX H
 DCR B

;NOT THIS COMMAND. SKIP REST OF IT.

JNZ FINDC3
 FINDC4: INR C

;TRY NEXT COMMAND

JMP FINDC1

CPNSUB: ORA A
 JNZ OPNSB1
 LXI D, FCBSUB
 CALL OPNFIL
 JZ OPNSB1
 MVI A, SFF
 JMP OPNSB2

; 2C3A B7
 ; 2C3B C2 4C 2C
 ;SUBMIT COMMAND'S FILENAME
 ;TRY TO FIND IT
 ;IF NOT THERE, SKIP.
 ;FLAG DOING SUB FILE

OPNSB1: XRA A
 OPNSB2: STA SUBFLG
 RET

;NOT DOING SUB FILE
 ;STORE WHETHER SUBMIT IN PROGRESS OR NOT

CCPREE: XRA A
 STA CBUFCT

; 2C51 AF
 ; 2C52 32 07 29

;JUMP HERE FROM BASE OF CCP

CCP000: LXI SP, CCPSTK
 PUSH B
 PUSH B
 CALL INIDOS
 POP B
 PUSH PSW

;SET UP STACK FOR CCP

; 2C58 C5

; 2C59 C5

;INIT BDOS

; 2C5D C1

; 2C5E F5

```

MOV A, C ;THIS DISK NUMBER
CALL LOGDSK ;LOG IT IN AND SELECT IT
POP PSM ; 2C63 F1
POP B ; 2C64 C1
INR A ; 2C65 3C
ORA C ; 2C66 B1
CALL OPNSUB ;OPEN SUB FILE, IF EXISTS
LDA CBUFT ; 2C6A 3A 07 29
ORA A ; 2C6D B7
JNZ NPROMT ; 2C6E C2 87 2C
PROMPT: LXI SP, CCPSTK ;INIT STACK
CALL TCRLF ; 2C74 CD 98 29
CALL GETLDN ;INTERROGATE DRIVE NUMBER
ADI CH,A ;MAKE IT A LETTER, A-D
CALL CCPTYO ;PROMPT CHAR
MVI A, CH,PRM ;GO COLLECT A COMMAND
CALL CCPTYO ;DEFAULT LOW CORE BUFFER
CALL COLECT ;SET DMA ADDRESS
NPROMT: LXI D, DEFBFR ;INTERROGATE DRIVE NUMBER
CALL SETADR ;SAVE COMMAND LEVEL DISK
CALL GETLDN ; 2C93 CD 44 2B
STA CMDUNI ; 2C96 C4 EF 2A
CALL PARSE ; 2C99 3A CA 32
CNZ CMOREJ ; 2C9C B7
LDA PRSUNI ;NO SUCH COMMAND
ORA A ; 2CA0 CD 10 2C
JNZ ZNOTFD ; 2CA3 21 B0 2C
CALL FINDCM ; 2CA6 5F
LXI H, CDSPTB ; 2CA7 16 00
MOV E, A ; 2CA9 19
MVI D, D ; 2CAA 19
DAD D ; 2CAB 7E
DAD D ; 2CAC 23
MOV A, M ; 2CAD 66
INX H ; 2CAE 6F
MOV H, M ; 2CAF E9
MOV L, A
PCHL

```

;DISPATCH TABLE FOR TYPED COMMANDS

```

CDSPTB: •ADDR ZDIR ;DIRECTORY COMMAND
        •ADDR ZERA ;ERASE COMMAND
        •ADDR ZTYP ;TYPE COMMAND
        •ADDR ZSAV ;SAVE COMMAND
        •ADDR ZREN ;RENAME COMMAND
        •ADDR ZNOTFD ;NO SUCH COMMAND

```

;HERE ONLY IF SERIAL NUMBER MISMATCHES

```

SERERR: LXI H, $76F3 ;MAKE A DI, HLT PAIR
        SHLD CCP ;PUT THEM AT START ADDRESS OF CCP
        LXI H, CCP ;GO DO THEM
        PCHL ;WHEN SERIAL NUMBERS DONT MATCH

RRDERR: LXI B, MRDERR ;TYPE READ ERROR MESSAGE
        JMP CRSOUT

```

MRDERR: •ASCIZ /READ ERROR/

RNTFND: LXI B, MNTFND
JMP CRSOUT

;TYPE NOT FOUND MESSAGE

MNTFND: ASCIZ /NOT FOUND/

GETNUM: LXI H, CMDNAM
LXI B, \$200B

;POINT AT WORD WHICH MAY BE NUMBER
;11 CHARS, ANS IN B STARTS AT 0

GETNML: MOV A, M

;GET WHAT MAY BE A DIGIT

CPI CH.SPC

;END OF WORD?

JZ GETNM1

;JUMP IF SO, CHECK TRAILING CHARS

INX H

;NO. STEP PAST IT

SUI 10

;SEE IF IT'S A DECIMAL DIGIT

CPI 10

;0-9?

JNC CMDREJ

;ERROR IF NOT

MOV D, A

;GOOD. KEEP A COPY

MOV A, B

;ACCUMULATE DECIMAL NUMBER

ANI \$E0

;PRE-CHECK FOR OVERFLOW

JNZ CMDREJ

;BAD IF SO

MOV A, B

;OK, MULTIPLY PARTIAL BY 10

RLC

;THREE SHIFTS FOR EIGHT

RLC

RLC

ADD

B

;ADD ONE MAKES NINE

JC CMDREJ

ADD B

;ONE MORE MAKES TEN

JC CMDREJ

;QUIT IF OVERFLOW

ADD D

;ADD IN NEW DIGIT

JC CMDREJ

;AGAIN, ERR IF OVERFLOW

MOV B, A

;NEW PARTIAL ANSWER

DCR C

;COUNT CHARS IN SOURCE LINE

JNZ GETNML

;LOOP IF MORE TO GO

RET

;MAKE SURE END OF WORD AND NO MORE BUT SPACE FILLS

GETNM1: MOV A, M

;ALL SPACES?

CPI CH.SPC

JNZ CMDREJ

;NOT A SPACE

INX H

DCR C

;COUNT AND SKIP ANOTHER

JNZ GETNM1

;LOOP IF MORE

MOV A, B

RET

-L

MOV3CH: MVI B, \$23

;MOVE 3 CHARS

MOVNCH: MOV A, M

;ENTER HERE TO MOVE (B) CHARS

STAX D

;FROM (HL) TO (DE)

INX H

INX D

DCR B

JNZ MOVNCH

RET

;GET A CHAR FROM DISK DIRECTORY AT (A) + (C) INTO THE BUFFER

GTDRCH: LXI H, DEFBFR

ADD C

;CURRENT CHAR + FCB BASE

CALL HL..A

;DA + HL => HL

MOV A, M

;GET THE CHAR

RET

ROUTINES TO SWITCH TO/FROM DISK NAMED IN A COMMAND, FROM TOP-LEVEL ONE.

```

TMPSEL: XRA      A           ;CLEAR DISK NUMBER BYTE IN FCB
        STA      CMDFCB
TMPSEL2: LDA      PRSUNI     ;DISK SPECIFIED?
        ORA      A
        RZ
        DCR      A           ;DONE IF NOT
        LXI H,    CMDUNI     ;YES, CONVERT TO 0-3
        CMP      M           ;WAS IT THE DEFAULT?
        RZ          ;SAME AS TOP LEVEL?
        JMP      LOGDSK      ;IF SO, DON'T RE-LOG IT
                                ;DIFFERENT. GO LOG IT.

```

ROUTINE TO SWITCH TO COMMAND LEVEL DISK.

```

DESEL:  LDA      PRSUNI     ;WAS THERE ONE IN COMMAND?
        ORA      A           ;..
        RZ          ;RETURN IF NOT
        DCR      A           ;YES. CONVERT TO 0-3 FORM
        LXI H,    CMDUNI     ; 2051 21 C9 30
        CMP      M           ;IS IT SAME AS TOP LEVEL ONE?
        RZ          ;IF SO, NO PROBLEM
        LDA      CMDUNI     ;IF DIFFERENT, SWITCH BACK TO THIS
        JMP      LOGDSK      ;GO LOG IT

```

DIRECTORY LISTING COMMAND. DIR, OR DIR AFN

```

DIR:    CALL      PARSE      ;GET THE TEXT ARGUMENT
        CALL      TMPSEL     ;SWITCH TO REQUESTED DISK
        LXI H,    CMDNAM     ;POINT TO ARGUMENT
        MOV A,    M           ;WAS IT BLANK?
        CPI      CH.SPC
        JNZ      ZDIR2      ;NO. DIR OF SPECIFIC FILES
        MVI B,    NNAMECH+NEXTCH ;YES. PLUG IN ALL WILD CARDS
ZDIR1:  MVI M,    CH.QM       ;FILL FCB NAME AND EXT
        INX
        DCR      B           ;COUNT THEM
        JNZ      ZDIR1      ;LOOP FOR WHOLE NAME/EXT
ZDIR2:  CALL      FNDCLF     ;FIND FILE
        CZ          ;IF NONE AT ALL, TYPE "NOT FOUND"
ZDIR3:  JZ          ZDIR7     ;JUMP IF NONE FOUND
        CALL      TCRLF      ;NEXT LINE
        CALL      GETLDN     ;FIND LOGGED DISK NUMBER
        ADI      CH.A        ;MAKE "A"- "D"
        CALL      CCPTYO     ;DISK NAME
        MVI A,    CH.CLN     ;COLON
        CALL      CCPTYO
        MVI A,    CH.SPC     ;SPACE FOR SEPARATOR
        CALL      CCPTYO
        LDA      FCBNUM      ;COMPUTE BYTE ADDR IN SECTOR
        RAL              ;FOR THIS FCB, FROM FCB NUMBER
        RAL
RAI
ANI      $60              ;32 BYTES PER FCB, FOUR FCB PER SECTOR
        MOV C,    A           ;HOLD HERE
        MVI B,    $01        ;FIRST CHAR OF FILENAME
ZDIR4:  MOV A,    B           ;CHARACTER NUMBER
        CALL      GDRCH      ;GET DEFBFR+C+A TO A
        CPI      CH.SPC     ;TRAILING SPACE IN NAME?

```

```

JNZ      ZDIR5
MVI A,   $09
CALL     GTDRCH
MVI A,   CH.SPC
CMP      M
JZ       ZDIR6
ZDIR5:   CALL     TYOSBC
INR      B
MOV A,   B
CPI      $0C
JNC      ZDIR6
CPI      $09
JNZ      ZDIR4
MVI A,   CH.SPC
CALL     TYOSBC
JMP      ZDIR4

ZDIR6:   CALL     CHKKB5
JNZ      ZDIR7
CALL     FNDNXT
JMP      ZDIR3

ZDIR7:   JMP      CMDDUN

;ERASE FILE(S) COMMAND

ZERA:    CALL     PARSE
CPI      NNAMECH+NEXTCH
JNZ      ZERA1
LXI B,   ERAMSG
CALL     CRSOUT
CALL     COLECT
LXI H,   CBUFCT
DCR      M
JNZ      PROMPT
INX      H
MOV A,   M
CPI      'Y'
JNZ      PROMPT
INX      H
SHLD     CMOPTR
LXI H,   CMDFCB
MVI M,   CH.CH
CALL     TMPSL2
LXI D,   CMDFCB
CALL     DELFIL
CALL     INIDOS
JMP      CMDDUN

ZERA1:   CALL     TMPSL2
LXI D,   CMDFCB
CALL     DELFIL
JMP      CMDDUN

ERAMSG:  .ASCIZ  "ALL FILES (Y/N)?."      ;FOR VERIFYING *.*

;COMMAND TO TYPE OUT A FILE

ZTYP:    CALL     PARSE
IN7      CMDFCB

```

```

;JUMP IF NOT
;IF SO, SKIP TO EXTENSION
;GET EXTENSION'S FIRST CHAR
;IS EXT BLANK?
;..
;JUMP IF SO.
;TYO, SAVING BC
;STEP TO NEXT CHAR OF NAME/EXT

;END OF EXTENSION YET?
;JUMP IF SO
;END OF NAME?
;JUMP IF NOT
;YES, SPACE BETWEEN NAME AND EXT
;TYPE SPACE
;START ON EXT

;CHECK FOR TYPEIN
;IF SO, BREAK OUT
;FIND NEXT FILE IN * GROUP
;GO LIST NEXT FILENAME

;ON TYPEIN, DONE WITH COMMAND

;SCAN NAME(S)
;ALL FILES?
;NOT ALL WILDCARDS
;YES, BETTER CHECK THAT!
;ASK OPERATOR
;GET A Y OR N
;HOW MANY CHARS IN ANSWER
;BETTER HAVE BEEN ONE
;JUMP IF WASN'T ONE.
;WAS ONE CHAR. GET IT.
;..
;WAS IT A YES?
;QUIT IF NOT.
;REALLY WANTS ALL FILES DELETED
;POINT AT THE TRAILING SPACES
;AND RE-BUILD THE *.*
;OUT OF ALL WILD-CARDS
;SWITCH TO CORRECT DISK
;POINT TO THE WILD NAME
;DO THE DELETES
;RE-INIT THE DOS
;SCAN TO EOL AND RE-PROMPT

;SWITCH TO REQUESTED DISK
;POINT TO REQUESTED FILE NAME
;DO THE DELETES
;SCAN TO EOL AND RE-PROMPT

;CAN'T DO WILDCARDS

```

```

CALL    TMPSEL    ;SWITCH TO REQUESTED DISK
CALL    OPNCMD    ;OPEN THE FILE
JZ      ZTYPEF    ;JUMP IF NOT FOUND
CALL    TCRLF     ;MOVE BELOW COMMAND LINE
LXI H,  TYPNCH    ;CHAR COUNT IN SECTOR (USED CT)
MVI M,  $FF      ;MAKE HUGE
TYP1:   LXI H,  TYPNCH ;HOW MANY CHARS USED
        MOV A,  M  ;READ THIS MANY?
        CPI    NSECCH ;WHOLE SECTOR?
        JC     ZTYPE2 ;JUMP IF STILL OK
        PUSH   H    ;NEED TO READ A SECTOR. SAVE HL
        CALL   REDCMD ;READ NEXT SECTOR
        POP    H    ;RESTORE STACK / HL
        JNZ    TYEOFQ ;JUMP IF EOF OR ERR.
        XRA     A    ;CLEAR COUNT OF TYPED CHARS IN SECTOR
        MOV M,  A    ; ..
TYP2:   INR      M    ;COUNT A USED CHAR
        LXI H,  DEFBFR ;POINT INTO BUFFER
        CALL   HL..A  ;ADD IN USED COUNT
        MOV A,  M    ;GET THE CHAR AT THAT POINT
        CPI    CH.EOF ;END OF FILE?
        JZ     CHDDUN ;NO, TYPE THE CHAR.
        CALL   CCPTYC ;CHECK THE KEYBOARD
        CALL   CHKKBS ;IF TYPEIN, QUIT.
        JNZ    CHDDUN ;LOOP, TYPE MORE.
        JMP    ZTYP1

TYEOFQ: DCR      A    ;WAS IT ERR OR EOF?
        JZ     CHDDUN ;IF EOF, THAT'S ALL.
        CALL   RRDERR ;ERROR. SAY "READ ERROR"
TTYPE:  CALL    DFLSEL ;SWITCH BACK TO DEFAULT UNIT
        JMP    CMDREJ  ;AND ERROR PATH BACK TO IOP

;SAVE COMMAND

;SAV:   CALL    PARSE    ;SCAN A SIZE NUMBER
        LDA     PRSUNI   ;DISK SPECIFIED IN SIZE??
        ORA     A        ; ..
        JNZ    CMDREJ   ;IF SO, THAT'S WRONG.
        CALL   GETNUM    ;GET SIZE FROM COMMAND TEXT
        PUSH   PSW       ;SAVE THE COUNT
        CALL   PARSE     ;NOW GET THE FILENAME
        JNZ    CMDREJ   ;NO WILDCARDS ALLOWED
        CALL   TMPSEL    ;SWITCH TO REQUESTED DISK
        LXI D,  CMDFCB   ;DELETE ANY PREVIOUS COPY
        PUSH   D         ;SAVE ADDR OF CMDFCB
        CALL   DELFIL    ;DELETE
        POP    D         ;POINT AT NAME AGAIN
        CALL   CREATF    ;CREATE NEW COPY
        CALL   OPNCMD    ;OPEN NEW COPY
        JZ     ZSAV2     ;JUMP IF NO SPACE.
        POP    PSW       ;NUMBER OF PAGES TO SAVE
        MOV L,  A        ;CONVERT TO SECTORS
        MVI H,  0        ;WHICH MAY BE 9 BITS
        DAD    H         ;BY DOUBLING (NOT PARAMETERIZABLE?)
        LXI D,  .TPA.    ;START AT BASE OF TRANSIENT AREA
;SAV1:  MOV A,  H        ;SEE IF DONE ALL SECTORS
        ORA     L        ; ..
        JZ     ZSAV3     ;IF ZERO, DONE.
        DCX     H        ;COUNT ANOTHER SECTOR SAVED

```

```

        PUSH    H
        LXI H,  NSECCH
        DAD     D
        PUSH    H
        CALL    SETADR
        LXI D,  CMDFCB
        CALL    WRITFL
        POP     D
        POP     H
        JNZ     ZSAV2
        JMP     ZSAV1

ZSAV2:  LXI B,  ZSAVM1
        CALL    CRSOUT
ZSAV3:  LXI D,  CMDFCB
        CALL    CLSFIL
        JNZ     ZSAV4
        LXI B,  ZSAVM2
        CALL    CRSOUT
ZSAV4:  JMP     CMODUN

ZSAVM1:  .ASCIZ /NO SPACE/

ZSAVM2:  .ASCIZ /CANNOT CLOSE/
        .L
        .RENAME COMMAND - REN NEW=OLD

ZREN:   CALL    PARSE
        JNZ     CMOREJ
        LDA     PRSUNI
        PUSH    PSW
        CALL    TMPSEL
        CALL    FNDCFL
        JNZ     ZREN5
        LXI H,  CMDFCB
        LXI D,  CMFCB2
        MVI B,  $12
        CALL    MOYNCH
        LHLD    CMOPTR
        XCHG

CALL    NOTSPC
        CPI     '='
        JZ      ZREN1
        CPI     '-'
        JNZ     ZREN4

ZREN1:  XCHG
        INX

        SHLD    CMOPTR
        CALL    PARSE
        JNZ     ZREN4
        POP     PSW
        MOV B,  A
        LXI H,  PRSUNI
        MOV A,  H
        ORA     A
        JZ      ZREN2
        CMP     B
        MOV M,  B
        JNZ     ZREN4
        MOV M,  D

        .GET NEW FILE NAME
        .ERROR IF WILDCARDS
        .ANY DISK?
        .SAVE FOR LATER CHECK
        .SWITCH TO REQUESTED DISK
        .FIND NEW FILE
        .JUMP IF FOUND (SHOULDN'T)
        .COPY OUT THE FILE NAME
        .TO HERE
        .THIS MANY CHARACTERS
        .MOVE THAT TEXT
        .RESTART SCAN

        .SCAN FOR A NON-SPACE
        .LEGAL SEPARATORS
        .OK
        .OTHER LEGAL SEPARATOR
        .ERROR IF NOT

        .STEP PAST THE "="
        .RE-SCAN FROM THIS POINT
        .GET OLD FILE NAME
        .NO STARS ALLOWED
        .NEW DISK NUMBER
        .COMPARE WITH OLD FILE'S DISK
        .DISK IN SECOND NAME
        .WAS THERE ONE?
        .
        .JUMP IF NOT
        .YES. SAME AS OLD FILE?
        .AND UPDATE IT
        .CAN'T RENAME ACROSS DISKS
        .THIS IS THE ONE.

```

```

XRA      A          ;CLEAR DISK BYTE IN NAME
STA      CMDFCB      ; ..
CALL     FNDCLF      ;FIND OLD FILE.
JZ       ZREN3       ;IF NOT FOUND, QUIT
LXI D,   CMDFCB
CALL     RENAMF      ;DO THE RENAME!
JMP      CMDDUN      ;DONE

ZREN1:   CALL     RNTFND      ;OLD FILE NOT FOUND MSG
JMP      CMDDUN

ZREN2:   CALL     DFLSEL      ;SWITCH BACK TO DEFAULT UNIT
JMP      CMDDREJ

ZREN3:   LXI B,   FEXMSG      ;NEW FILE ALREADY EXISTS MSG
CALL     CRSOUT
JMP      CMDDUN      ;AND NO MORE TO DO

FEXMSG:  .ASCIZ  /FILE EXISTS/
        .L
ZNO1D:   CALL     CHKSER      ;NO SUCH CMD. IS VERSION OF SYSTEM OK?
        LDA      CHDNAM      ;YES. CMD START WITH A SPACE?
        CPI      CH.SPC
        JNZ      RCOMQ
        LDA      PRSUNI
        ORA      A
        JZ       CMDDN1      ;IF NOT, PROBABLY BLANK LINE
        DCR      A           ;THERE WAS A DRIVE. CONVERT TO 0-3
        STA      CMDDUN1     ;KEEP A COPY
        STA      .UNIT.      ;AND ONE FOR THE USER
        CALL     LOGDSK      ;GO LOG THAT DISK
        JMP      CMDDN1      ;CHECK FOR JUNK ON END OF LINE

RCOMQ:   LXI D,   CMDFCB+FCB.EX ;FIRST CHAR AFTER 8-LTR NAME
        LDAX     D
        CPI      CH.SPC
        JNZ      CMDDREJ     ;SPACE?
                                ;IF NOT, ERROR.
RCOM1:   PUSH     D           ;SAVE POINTER TO EXTENSION
        CALL     TMPSEL      ;SWITCH TO REQUESTED DISK
        POP      D           ;EXTENSION AGAIN
        LXI H,   COMEXT      ;PLUG IN .COM EXTENSION
        CALL     MOV3CH      ;COPY IT INTO FCB
        CALL     OPNCMD      ;TRY TO OPEN SUCH A FILE
        JZ       RCOME       ;JUMP IF FAILED. RE-LOG CMD DISK
        LXI H,   .TPA.       ;BASE OF TRANSIENT AREA
RCOM2:   PUSH     H           ;SAVE CORE ADDR
        XCHG              ;IN DE FOR SUBR ARG
        CALL     SETADR      ;SET DMA ADDRESS
        LXI D,   CMDFCB      ;THIS IS THE FILE
        CALL     READFL      ;READ ANOTHER SECTOR
        JNZ      RCOM2       ;IF EOF OR ERROR.
        POP      H           ;ADDRESS AGAIN
        LXI D,   WSECCH      ;SIZE OF SECTOR
        DAD      D           ;MOVE UP THAT MUCH
        LXI D,   CCP         ;BASE OF CCP
        MOV A,   L           ;MAKE SURE NOT OVERWRITING SELF
        SUB      E           ;16 BIT SUBTRACT
        MOV A,   H
        SBB      D
        INC      RCOMX      ;JUMP IF TOO BIG

```


	JMP	RCOM1	ELSE READ SOME MORE
RCOM2:	POP	H	CLEAR ADDR FROM STACK
	DCR	A	BETTER BE EOF
	JNZ	RCOMX	IF NOT, ERROR
	CALL	DFLSEL	SWITCH BACK TO DEFAULT UNIT
	CALL	PARSE	SEE IF ANY TRAILING ARGS
	LXI H,	PRSUNI	GET DISK DRIVE NUMBER
	PUSH	H	SAVE A COPY OF ADDR
	MOV A,	M	PICK UP DRIVE
	STA	CMDFCB	PUT A COPY IN FCB OF FILE
	MVI A,	\$10	Merge DOWN IN THE FCB THIS FAR
	CALL	PARSE1	AND PARSE ANY ARGUMENTS
	POP	H	DRIVE ADDRESS AGAIN
	MOV A,	M	MAY HAVE BEEN A DRIVE IN THE ARG
	STA	CMFCB2	
	XRA	A	START OF FILE
	STA	CMDFCB+FCB.NR	
	LXI D,	.FCB.	COMMAND FCB IN LOW CORE
	LXI H,	CMDFCB	POSSIBLE ARG OF FILENAME
	MVI B,	\$21	THIS MANY CHARS FOR USER
	CALL	MOVNCH	COPY THEM
	LXI H,	CBUFTX	RE-SCAN TO GIVE USER TEXT
RCOM3:	MOV A,	M	LOOK FOR SPACE OR NULL
	ORA	A	..
	JZ	RCOM4	QUIT ON NULL
	CPI	CH.SPC	QUIT ON SPACE
	JZ	RCOM4	..
	INX	H	KEEP LOOKING
	JMP	RCOM3	
RCOM4:	MVI B,	0	COUNT
	LXI D,	DEFBFR+1	COPY TRAILING STUFF FOR USER
RCOM5:	MOV A,	M	FROM COMMAND STRING
	STAX	D	TO USER COMM REGION
	ORA	A	QUIT ON NULL
	JZ	RCOM6	..
	INR	B	COUNT
	INX	H	SOURCE
	INX	D	DEST
	JMP	RCOM5	AND LOOP
RCOM6:	MOV A,	B	HOW MANY CHARS WE COPIED
	STA	DEFBFR	TELL USER. (HE GETS NO NULL)
	CALL	TCRLF	BEFORE GOING TO USER, CRLF
	LDA	CMDUNI	TELL USER THE CURRENT DRIVE
	STA	.UNIT.	..
	CALL	LIFTHD	LIFT HEAD (CCP) TO BDOS
	LXI D,	DEFBFR	PUT DMA BACK TO DEFAULT
	CALL	SETADR	SET DMA ADDRESS
	CALL	.TPA.	CALL THE USER PROGRAM
	LXI SP,	CCPSTK	IN CASE IT STEPPED ON STACK
	LDA	CMDUNI	IN CASE IT CHANGED DISKS
	CALL	LOGDSK	RE-LOG THE DEFAULT
	JMP	PROMPT	AND GO GET ANOTHER COMMAND
RCOME:	CALL	DFLSEL	SWITCH BACK TO DEFAULT UNIT
	JMP	CMDREJ	ERROR RETURN
RCOMX:	LXI B,	LDERRM	LOAD ERROR COMMENT

```

CALL    CRSOUT
JMP     CMDDUN                ;ASK FOR ANOTHER COMMAND

LDEIRM:  .ASCIZ /LOAD ERROR/

COMEXT:  .ASCII /COM/

CMNDUN:  CALL    (DFLSEL      ;SWITCH BACK TO DEFAULT UNIT
CMNDN1:  CALL    PARSE       ;ANY MORE TEXT?
        LDA     CMDNAM      ;SHOULD BE JUST BLANKS
        SUI     CH.SPC     ;..
        LXI H, PRSUNI      ;MAYBE A DISK GOT PARSED
        ORA     M          ;..
        JNZ     CMDREJ     ;IF EITHER, ERROR.
        JMP     PROMPT     ;OK, GET ANOTHER COMMAND

        .BLKB 510

CCP_TK:
CCP_FLG:  .BYTE 0           ;CCP'S STACK ABOVE HERE
                           ;FLAG FOR SUBMIT FILE BEING PROCESSED

FCB_CB:  .BYTE 2
        .ASCII /$$$      SUB/
        .BLKB 21           ;FCB FOR SUBMIT

CMFCB:  .BLKB 1             ;FCB STARTS HERE W/ DRIVE NUMBER
CMNAM:  .BLKB 15           ;USER'S COMMAND VERB OR FILENAME GOES HERE
CMV32:  .BLKB 311         ;ANOTHER ONE FOR RENAMES

FCMDM:  .BLKB 1
CMINI:  .BLKB 1           ;UNIT AT COMMAND LEVEL
PFINI:  .BLKB 1           ;UNIT RECOGNIZED BY PARSER
TYPECH: .BLKB 1           ;TEMP FOR TYPE COMMAND

;SPACE TO PAGE BOUNDARY
        .BLKB 52
;***** END OF CONSOLE COMMAND PROCESSOR *****
;***** BDOS STARTS HERE *****

DOSSER: .BYTE 0,30E,0,0,1,2 ;SERIAL NUMBER OF COPY OF DOS
SER1Z=.-DOSSER              ;SIZE OF SERIAL NUMBER

;ENTER HERE FOR ALL SERVICES. 5/ JMP BDOS

BDOS:    JMP     BDOSH      ;THIS IS A VECTOR. GO TO BODY.

VBSSEC:  .ADDR    BADSEC    ;HANDLER FOR BAD SECTOR
VSELER:  .ADDR    SELERR    ;HANDLER FOR SELECT ERROR
VROERR:  .ADDR    ROERR     ;HANDLER FOR R/O ERROR

;FOLLOWING ROUTINE AND CONSTANTS ARE MEDIUM-DEPENDANT
;ROUTINE TO PICK UP SKEWED SECTOR NUMBER

SKEWSC:  LXI H, SKEWTB      ;POINT TO TRANSLATION TABLE
        MVI B, 0           ;MAKE SECTOR INTO FULL WORD
        DAD     B          ;INDEX INTO TABLE
        MOV C, M           ;PICK UP TRANSLATED SECTOR
        JMP     CBSTSC     ;BIOS VECTOR TO SET SECTOR
; THIS IS THE ONLY SUCH CALL

```

•BYTE 0

;TABLE OF SECTORS, LENGTH 26. FOR SKEW

```
SKFWTB: •BYTE $01,$07,$0D,$13,$19
        •BYTE $05,$0B,$11,$17
        •BYTE $03,$09,$0F,$15
        •BYTE $02,$08,$0E,$14,$1A
        •BYTE $06,$0C,$12,$18
        •BYTE $04,$0A,$10,$16

        •BYTE 0,0,0,0,0,0
```

;CONSTANTS (READ ONLY) DEFINING DISK LAYOUT

```
SECTPK: •BYTE $1A      ;SECTORS PER TRACK
MAXFCB: •BYTE $3F      ;MAX FCB IN DISK DIRECTORY
LSECPC: •BYTE 3         ;LOG2 OF CLUSTER SIZE. 8 SECTORS PER ALLOC UNIT
ALLMSK: •BYTE 7         ;MASK FOR SECTOR WITHIN ALLOC UNIT
MXCLUS: •BYTE $F2      ;N CLUSTERS FOR ALLOCATOR
INIALO: •BYTE $C0      ;INITIAL ALLOC. ALWAYS PRESERVE FIRST 2 CLUSTERS,
                        ; WHICH ARE THE DIRECTORY
TKBIAS: •BYTE $02      ;BIAS ON ALL COMPUTED TRACKS. SKIPS BOOT IMAGE.
```

~L
;HERE TO HANDLE CALLS THRU 5.

```
BDOSH:  XCHG                ;USER'S INFO ADDR TO HL
        SHLD USERDE        ;SAVE IT HERE
        LXI H, USRFCN      ;PUT USER'S FUNCTION HERE
        MOV M, C           ;..
        LXI H, 0           ;SAVE USER'S STACK POINTER
        DAD SP             ;WHICH TAKES A BIT OF EFFORT
        SHLD USERSP        ;STASH IT HERE
        LXI SP, BDOSSP     ;LOCAL STACK
        CALL BDOS00        ;CALL GUTS AS A SUBR
        LHLD USERSP        ;RESTORE USER STACK
        SPHL              ;..
        LHLD ANSWER        ;GET ANSWER TO USER
        MOV A, L           ;IN A AND B FOR PL/M CODE
        MOV B, H
        RET
```

```
BADSEC: LXI H, MSGADSC     ;BAD SECTOR MESSAGE
        CALL ERRPRT       ;ERROR MSG PRINTER
        CPI CH.ATN        ;USER TYPED ^C?
        JZ $0000          ;IF SO, REBOOT
        RET               ;IF NOT, ACCEPT BAD DATA
```

```
SFLERR: LXI H, MSELECT    ;'SELECT' MESSAGE
        JMP ERROR1        ;BIGGER ERROR MSG. AND REBOOT
```

```
ROERR:  LXI H, MRONLY     ;'R/O' MESSAGE
ERROR1: CALL ERRPRT       ;TYPE BIGGER ERR MSG
        JMP $0000         ;RE-BOOT AS IF ^C
```

```
ERRPRT: PUSH H           ;SAVE SPECIFIC MSG
        CALL BDCRLF       ;TYPE A CRLF
        LDA CURDSK        ;GET DRIVE NUMBER
        ADI CH.A          ;MAKE IT A LETTER. A-D
```

STA	MSG002	;STUFF IT INTO TEXT OF MSG
LXI B,	MSG001	;GET START OF MSG
CALL	DLSOUT	;TYPE THE GENERAL MSG
POP	B	;GET SPECIFIC MSG
CALL	DLSOUT	;TYPE IT, TOO
JMP	TYISVD	;GOBBLE A CHARACTER AND RETURN

~L
;TEXT FOR ABOVE MESSAGE PRINTERS

MSG001: .ASCII /BDOS ERR ON /
MSG002: .ASCII / : \$/
MBADSC: .ASCII /BAD SECTOR\$/
MSFLEC: .ASCII /SELECT\$/
MRONLY: .ASCII .R/O\$.

;ROUTINE TO READ A CHAR FROM TYI. FIRST SEE IF ONE SAVED ALREADY.

TYISVD:	LXI H,	SAVCH	;SEE IF THERE WAS A SAVED CH,
	MOV A,	M	; FROM STOPPING TYPEOUT
	MVI M,	D	;SEE IT ONLY ONCE
	ORA	A	;WAS THERE ONE?
	RNZ		;RETURN IT IF SO
	JMP	CBCTYI	;ELSE GO GET ONE FROM BIOS

;GET AND ECHO A CHARACTER

RDCCNS:	CALL	TYISVD	;GET SAVED, OR NEW, CHAR
	CALL	CKPRNT	;CHECK IT FOR CONTROLS
	RC		;RETURN IF NON-TYPING CTL
	PUSH	PSW	;ECHO IT.
	MOV C,	A	;DIFFERENT AC FOR TYO
	CALL	CTY02	;TYPE IT OUT
	POP	PSW	
	RET		

~L
;CHECK FOR PRINTABLE CHAR. SIMPLE CONTROLS ARE PRINTABLE.
; RETURN WITH CY FLAG ON IF ITS A NON-PRINTABLE CTL CHAR, NEEDING "™"

CKPRNT:	CPI	CH.CR
	RZ	
	CPI	CH.LF
	RZ	
	CPI	CH.TAB
	RZ	
	CPI	CH.CTL
	RET	

CHKKBD:	LDA	SAVCH	;GET ANY SAVED CHAR
	ORA	A	
	JNZ	CHKKB2	;RETURN IT
	CALL	CBTTCK	;CHECK KEYBOARD STAT, IN BIOS
	ANI	\$D1	
	RZ		;RETURN IF NONE THERE
	CALL	CBCTYI	;THERE'S SOMETHING READY. GET IT.
	CPI	CH.XOF	
	JNZ	CHKKB1	;JUMP UNLESS STOP REQUEST
	CALL	CBCTYI	;IN WHICH CASE WAIT FOR ANOTHER CH ← H/A
	CPI	CH.ATN	;IS IT A CONTROL C?
	JZ	\$C000	;IF SO, RE-BOOT
	XRA	A	;AFTER RESUMING, RETURN

RET

CHKKB1: STA SAVCH
 CHKKB2: MVI A, \$01
 RET

;SOMETHING BUT NOT XOFF. SAVE IT.
 ;SAY HAVE SOMETHING TO READ

~L

;CONSOLE TYO ROUTINE

CTYO: PUSH B
 CALL CHKKBD
 POP B
 PUSH B
 CALL CBCTYO
 POP B
 PUSH B
 LDA LPTFLG
 ORA A
 CNZ CBLPTO
 POP B
 MOV A, C
 LXI H, HPOS
 CPI CH.RUB
 RZ
 INR M
 CPI CH.CTL
 RNC
 DCR M
 CPI CH.LF
 RNZ
 MVI M, Z
 RET

;SAVE CHAR IN C
 ;CHECK FOR BEING FROZEN
 ;RESTORE AND SAVE CHAR AGAIN
 ;TTY OUTPUT IN BIOS
 ;GET BACK CHAR AND SAVE AGAIN
 ;COPYING ON PRINTER?
 ;IF SO, PRINT IT.
 ;GET THE CHAR AGAIN
 ;COPY IN A
 ;POINT TO COLUMN POSITION
 ;A RUBOUT DOESN'T SPACE
 ;SO DON'T. DONE.
 ;STEP HPOS
 ;CONTROL CHAR?
 ;IF NOT, ALL DONE.
 ;YES. DON'T SPACE AFTER ALL
 ;ON LF (SHOULD BE CR),

;CLEAR HPOS

~L

;HERE FOR PRETTY TYO, WITH ^X FORMAT AND TAB SIMULATION

CTYO1: MOV A, C
 CALL CKPRNT
 JNC CTYO2
 PUSH PSW
 MVI C, CH.UPAR
 CALL CTYO
 POP PSW
 ORI \$40
 MOV C, A
 CTYO2: MOV A, C
 CPI CH.TAB
 JNZ CTYO
 CTYOTL: MVI C, CH.SPC
 CALL CTYO
 LDA HPOS
 ANI \$07
 JNZ CTYOTL
 RET

;COPY CHAR
 ;CHECK FOR NEEDING UPARROW
 ;JUMP IF DON'T.
 ;SAVE CHAR IN A
 ;GET AN UPARROW
 ;TYPE UPARROW
 ;GET BACK CHAR
 ;MAKE UN-CONTROL.
 ;BACK TO RIGHT AC
 ;COMMON TYO ROUTINE
 ;TAB CHAR?
 ;IF NOT, TYPE IT AS IS.
 ;SIMULATE TAB WITH SPACES
 ;TYPE A SPACE
 ;ENOUGH TO MAKE MULT OF 8?
 ;IF NOT, TYPE SOME MORE.

NMCRLF: MVI C, CH.NUM
 CALL CTYO
 BDCRLF: MVI C, CH.CR
 CALL CTYO
 MVI C, CH.LF
 JMP CTYO

;NUMBER SIGN
 ;TYPE THE NUMBERSIGN
 ;TYPE CR
 ;AND LINEFEED
 ;TYPE AND RETURN

TYPE STRING UP TO TERMINATING DOLLARSIGN

DLSOUT:	LDAX	B	;GET CURRENT CHAR
	CPI	CH.DOL	;IS IT A DOLLAR?
	RZ		;IF SO, RETURN WITHOUT PRINTING IT
	INX	B	;ELSE, STEP FOR NEXT ONE,
	PUSH	B	;SAVE POINTER,
	MOV C,	A	;MOVE CURRENT CHAR TO C,
	CALL	CTY02	;AND TYPE CURRENT CHAR
	POP	B	;GET BACK MEMORY POINTER
	JMP	DLSOUT	;GO DO ANOTHER CHAR

HERE FOR READ STRING BUFFERED (PSIN) COMMAND

PSIN:	LHLD	USERDE		; 326B	2A 0A 33
	MOV C,	M		; 326E	4E
	INX	H		; 326F	23
	PUSH	H		; 3270	E5
	MVI B,	0		; 3271	66 00
PSIN1:	PUSH	B	;SAVE SOME REGS		
	PUSH	H			
PSIN2:	CALL	TYISVO	;RAW TYI, OR SAVED ONE FROM TYC/STOP		
	ANI	CH.MSK	;JUST 7 BITS		
	POP	H			
	POP	B			
	CPI	CH.CR	;HANDLE CAR RET		
	JZ	PSIN13	;TERMINATE STRING		
	CPI	CH.RUB	;RUBOUT?		
	JNZ	PSIN3	;JUMP IF NOT		
	MOV A,	B	;HANDLE CHAR DELETION		
	ORA	A		; 3287	B7
	JZ	PSIN1		; 3288	CA 73 32
	MOV A,	M		; 328B	7E
	DCR	B		; 328C	45
	DCX	H		; 328D	2B
	JMP	PSIN11		; 328E	C3 E6 32
PSIN3:	CPI	\$05	;CONTROL E? (PHYS CRLF)		
	JNZ	PSIN4		; 3293	C2 9E 32
	PUSH	B		; 3296	C5
	PUSH	H		; 3297	E5
	CALL	BDCRLF		; 3298	C0 53 32
	JMP	PSIN2		; 329B	C3 75 32
PSIN4:	CPI	\$10	;CONTROL P?		
	JNZ	PSIN5	;JUMP IF NOT		
	PUSH	H	;SAVE STRING POINTER		
	LXI H,	LPTFLG	;TOGGLE PRINTER-COPY FLAG		
	MVI A,	\$01	; ..		
	SUB	M	; ..		
	MOV H,	A	; ..		
	POP	H	;RESTORE POINTER		
	JMP	PSIN1	;GET ANOTHER CHAR.		
PSIN5:	CPI	\$18	;CONTROL X?		
	JZ	PSIN6		; 32B1	CA B9 32
	CPI	\$15	;OR CONTROL U?		
	JNZ	PSIN7		; 32B6	C2 C0 32
PSIN6:	CALL	NMCRLF	;HERE TO DELETE WHOLE LINE, TYPE ""		
	POP	H	; AND A CRLF		


```

      JMP      PSIN
PSIN7: CPI      $12
      JNZ      PSIN10
      PUSH     B
      CALL     NMCRLF
      POP      B
      POP      H
      PUSH     H
      PUSH     B
PSIN8: MOV A, B
      ORA      A
      JZ       PSIN9
      INX      H
      MOV C, M
      DCR      B
      PUSH     B
      PUSH     H
      CALL     CTY01
      POP      H
      POP      B
      JMP      PSIN8
PSIN9: POP      B
      JMP      PSIN1
;HERE ON ORDINARY CHAR.

PSIN10: INX      H
      MOV M, A
      INR      B
PSIN11: PUSH     B
      PUSH     H
      MOV C, A
      CALL     CTY01
      POP      H
      POP      B
      MOV A, M
      CPI      CH.ATN
      MOV A, B
      JNZ      PSIN12
      CPI      $01
      JZ       $0000
PSIN12: CMP      C
      JC       PSIN1
PSIN13: POP      H
      MOV M, B
      MVI C, CH.CR
      JHP      CTY0
HPCS:  .BYTE 0
LPTFLG: .BYTE 0
SAVCH:  .BYTE 0
CURDSK: .BYTE 0
USRFCN: .BYTE 0
USERDE: .BYTE 0,0
ANSWER: .BYTE 0,0
USERSP: .BYTE 0,0
      .BLKB 48
BDOSSP:

;TRY AGAIN
;CONTROL R?
;JUMP IF NOT
;YES. RETYPE LINE
;TYPE "##" CRLF FIRST
;RESTORE COUNT
;AND POINTER
;THEN RE-SAVE
; AND COUNT
;CHECK COUNT FOR DONE
; ..
;QUIT IF DONE
;NOT DONE. STEP TO ANOTHER CHAR
;PICK IT UP
;COUNT THEM
;SAVE COUNT AND POINTER OVER TYPEOUT
;TYPE CHAR, WITH " PROCESSING
;TYPE WHOLE LINE
;DONE RE-TYPING. RESTORE REAL COUNT
;AND GO GET ANOTHER CHAR

;SAVE IN BUFFER
;COUNT THEM
; 32E6 C5
; 32E7 E5
; 32E8 4F
;TYPE CHAR
; 32EC E1
; 32ED C1
; 32EE 7E
;WAS IT A CONTROL C?
; 32F1 78
; 32F2 C2 FA 32
;IF SO, WAS IT THE FIRST CHAR?
;IF SO, RE-BOOT.
;UP TO FULL BUFFER?
;IF NOT, GET MORE.
;BACK TO START OF BUFFER
;AND PUT THE COUNT THERE
;ECHO THE C-R AT END
;TYPE CR
;HORIZ POSITION OF TTY
;NONZERO TO COPY TYO TO PRINTER
;BKJFN'ED TYI CHAR
;CURRENT DISK # (0-3)
;CALLER'S AC C, = BDOS FUNCTION
;CALLER'S DE, = ARG ADDRESS
;ANSWER TO RETURN TO USER
;USER'S STACK POINTER
;STACK AREA
;BDOS HAS STACK ABOVE HERE

```

FRECHR: .BYTE \$E5

;FILL CHAR ON BLANK DISK, FOR DELETED FILES

;HOME THE DISK AND SET THE TRACK NUMBER IN CORE TO ZERO

HOMSET: CALL CBHOME

;HOME THE DISK (BIOS). THIS IS THE
; ONLY SUCH CALL

LHLD TKBIAS

;GO TO LOGICAL TRACK ZERO

MOV C, L

;TO C FOR BIOS ARGUMENT

CALL CBSTTK

;SET TRACK NUMBER (BIOS) FROM C

LHLD TKSEC0

;THAT'S SECTOR NUMBER ZERO, TOO.

MVI A, 0

; ..

MOV M, A

; ..

INX H

;ITS TWO BYTES EACH

MVI M, 0

LHLD TKPNTR

;POINT TO TRACK NUMBER

MOV M, 0

;CLEAR TRACK NUMBER

RET

COMPTK: LHLD TKSEC0

;SECTOR 0 ON CURRENT TRACK

LXI D, CURRMV

;THIS CLUSTER

CALL DEMIHL

;(DE)-(HL)=>HL

JNC CMPTK1

;JUMP IF TEST SECTOR .GE. TRACK'S SECTOR

LHLD TKSEC0

;BACK UP ONE TK. REDUCE (HL).

XCHG

;PUT IT IN DE

LDA SECTPK

;SECTORS PER TRACK

CALL DEMIDA

;(DE) - DA => HL

XCHG

DCX

H

;CORRECT FOR MOD BY DEMIDA. BACK TO TKSEC0

MOV M, E

;PUT BACK REDUCED SECTOR NUMBER

INX H

MOV M, D

;AND STORE BACK IN (DE)

LHLD TKPNTR

;POINT TO TRACK NUMBER

DCR M

;REDUCE TRACK NUMBER

JMP COMPTK

;LOOP TIL ON RIGHT TRACK

CMPTK1: LHLD TKSEC0

;STEP UP ONE TRACK

LDA SECTPK

;SECTORS PER TRACK

CALL APLSMM

;DA + (HL) => HL

SHLD CTKTMP

;TEMP, NEW SECTOR

LXI D, CURRMV

;SECTOR NUMBER

CALL IDEMHL

;(DE) - HL => HL

JC CMPTK2

;JUMP IF DESIRED > TEST

LHLD TKSEC0

;CURRENT REAL SECTOR 0 OF THIS TK

PUSH H

LHLD CTKTMP

;TEST SECTOR

XCHG

POP

H

;STORE TEST AS CURRENT

MOV M, E

INX H

MOV M, D

LHLD TKPNTR

;POINT TO TRACK NUMBER

INR M

;MAKE IT CORRESPOND TO SECTOR

JMP CMPTK1

;SEE IF THAT'S HIGH ENOUGH

CMPTK2: LHLD TKPNTR

;POINT TO TRACK NUMBER

LDA TKBIAS

;OFFSET ALL LOGICAL TRACKS FOR BOOT

ADD M

MOV C, A

;ARG TO BIOS

CALL CBSTTK

;SET TRACK (BIOS) FROM C

LHLD TKSEC0

;SECTOR ZERO ON CHOSEN TRACK

```

LXI D, CURRMW      ;SECTOR NUMBER DESIRED
CALL DEMIHL        ;16 BIT SUBTRACT, TO HL
MOV C, L           ;LOGICAL SECTOR ON THE TRACK
CALL SKEWSC        ;GET CORR'ING SECTOR FROM SKEW TBL,
RET               ; CALL BIOS WITH IT (ONLY SUCH CALL)

;COMMON ENTRY FOR READ AND WRITE. IF READ, C/1. IF WRITE, C/2.

RWCOM: LXI H, RWARG ;SAVE READ OR WRITE COMMAND HERE
MOV M, C           ;READ OR WRITE COMMAND
LDA RWARG          ;PICK IT BACK UP
RAR               ;SEE WHICH IT IS
JNC RWCOMW         ;JUMP IF 0, FOR WRITE.
CALL CBREAD        ;CALL BIOS TO READ A SECTOR.
; (THIS IS THE ONLY SUCH CALL)
STA RWCOMT         ;STORE SUCCESS/FAIL CODE
JMP RWCOMF         ;COMMON CLEANUP FOR RD AND WRITE.

RWCOMW: CALL CBWRIT ;CALL BIOS TO WRITE A SECTOR.
; (THIS IS THE ONLY SUCH CALL)
STA RWCOMT         ;STORE SUCCESS/FAIL CODE
RWCOMF: LDA RWCOMT  ;SEE IF THERE WAS AN ERROR
CPI 0              ;IF ERROR, GO HANDLE IT.
JNZ RWCOME         ;IF NO ERROR, RETURN FROM RD/WRITE
RET

RWCOME: LXI H, RWCOMZ ;PLACE TO RETURN TO
PUSH H            ;SAVE FOR LATER POPJ
LHLD HEADSC       ;GET BAD SECTOR HANDLER (MAYBE PATCHED)
PCHL              ;GO TO IT.
RWCOMZ: RET        ;IF HANDLER COMES BACK, ACCEPT THE ERR

;ALL READS COME THROUGH HERE

RDSEC: MVI C, $01   ;FLAG FOR READ
CALL RWCOM          ;CALL COMMON I/O ROUTINE
RET

;ALL WRITES COME THROUGH HERE

WRSEC: MVI C, 0     ;FLAG FOR WRITE
CALL RWCOM          ;CALL COMMON I/O ROUTINE
RET

;GET MAP BYTE FOR CURRENT NR

GTHAPB: LHLD LSEPC  ;CONSTANT, LOG SEC PER CLUSTER
MOV C, L
LXI H, CURRNR      ; 33F8 21 D3 3D
CALL RARNM         ;GET M, DO (C) RAR'S, GIVING CLUSTER
ADI $10            ;STEP TO MAP AREA OF FCB
MOV C, A           ;FOR THIS CLUSTER
MVI B, 0           ;AS ADDRESS (16 BITS)
LHLD USERDE       ;IN FCB
DAD B              ;ADDR OF THIS MAP BYTE
MOV L, M           ;THIS IS THE CLUSTER NUMBER
MVI H, 0           ;TREAT AS THOUGH COULD BE 16 BITS
SHLD CURRMW        ;SAVE MAP BYTE (CLUSTER NUMBER)
RET

```

; COMPUTE SECTOR NUMBER FROM CURRNR AND CURRMW

```
COMPSN:  LHL D      LSECP C      ; SHIFT FACTOR, SECTORS/CLUSTER
          MOV C,    L
          LXI H,    CURRMW      ; GET THE CLUSTER NUMBER
          CALL     SHLMM        ; MAKE A SECTOR NUMBER
          LDA      ALLMSK      ; MASK FOR SECTOR IN CLUSTER
          PUSH     H
          LXI H,    CURRNR
          ANA      M           ; MASK SECTOR WITHIN CLUSTER
          POP      H           ; GET BACK CLUSTER AS SECTOR
          CALL     ADRHL        ; OR IN SECTOR WITHIN CLUSTER
          SHLD     CURRMW      ; NOW HAVE FULL SECTOR NUMBER
          RET
```

; GET NR AND RC FROM FCB

```
STNRRC:  LXI B,    $0020      ; GET NR BYTE
          LHL D      USERDE   ; IN FCB
          DAD      B
          MOV A,    M           ; PICK IT UP
          STA      CURRNR      ; SAVE RECORD NUMBER
          LXI B,    $000F      ; ALSO GET RC (REC COUNT)
          LHL D      USERDE   ; FROM FCB
          DAD      B
          MOV A,    M           ; PICK IT UP
          STA      CURRRC      ; SAVE RECORD COUNT
          RET
```

; UPDATE NR AND RC IN FCB

```
UDNRRC:  LDA      CURRNR      ; RECORD NUMBER
          INR      A           ; STEP, PREPARE FOR NEXT RECORD
          LXI B,    $0020      ; POINT TO NR FIELD
          LHL D      USERDE   ; IN USER'S FCB
          DAD      B           ; STEP TO NR BYTE
          MOV M,    A           ; STORE IT
          LXI B,    $000F      ; SAME FOR RC BYTE
          LHL D      USERDE   ; IN FCB
          DAD      B
          LDA      CURRRC      ; 3451 09
          MOV M,    A           ; 3452 3A D2 3D
          RET                  ; UPDATE IT.
```

```
GDIRSC:  LDA      DOSFCB      ; 3457 3A F3 3C
          ANI      $FE        ; DIVIDE BY FOUR, GETS SECTOR NUMBER
          RAR
          RAR                  ; FROM THE FCB NUMBER
          STA      DIRSEC      ; THIS IS THE DIRECTORY SECTOR NUMBER
          MOV C,    A           ; MAKE A DOUBLE BYTE OF IT
          MVI B,    0
          MOV H,    B
          MOV L,    C           ; COPY IT
          SHLD     CURRMW      ; READY TO COMPUTE TK FOR THIS SECTOR
          CALL     COMPTK      ; FIND TRACK AND SECTOR IN TRACK
          RET
```

; CHECKSUM ROUTINE FOR DIRECTORY SECTORS. VERY INEFFICIENT. COMPILED?

```
GETCKS:  LXI H,    CKSTM2      ; SUM GOES HERE
```

```

      MVI M, 0
      DCX H
      MVI M, 0
GTCKS1: MVI A, NSECCH-1
      LXI H, CKSTM1
      CMP M
      JC GTCKS2
      LHLD CKSTM1
      MVI H, 0
      XCHG
      LHLD SYSDMA
      DAD D
      LDA CKSTM2
      ADD M
      STA CKSTM2
      LXI H, CKSTM1
      INR M
      JNZ GTCKS1
GTCKS2: LDA CKSTM2
      RET

```

```

;START WITH ZERO
;HERE IS THE COUNT OF BYTES
;WHICH ALSO STARTS AT 0
;DONE ALL BYTES?
;CHECK COUNT
;QUIT WHEN DONE
;NO, GET ADDRESS
;WITHIN BLOCK

```

```

;COMPUTE ADDRESS IN BUFFER

```

```

;PARTIAL SUM
;ADD ANOTHER BYTE
;SAVE PARTIAL SUM
;AND COUNT ANOTHER ONE DONE
;LOOP
;GET ANSWER

```

```

;OR ARG WITH BIT FOR CURR DISK

```

```

ORCURRE: LXI H, ORCURT
      MOV M, C
      LDA CURDSK
      INR A
      MOV C, A
      MVI A, 301
      CALL RLCN
      RAL
      LXI H, ORCURT
      ORA M
      RET

```

```

;SAVE ARG
;..
;NUMBER OF CURR DISK
;SINGLE BIT PATTERN
;DO (C) RLC'S
;BACK ONE
;GET CALLER'S ARG
;OR IT IN WITH NEW BIT

```

```

;GET R/O BIT FOR CURRENT DISK

```

```

CHKRO: LDA CURDSK
      INR A
      MOV C, A
      LXI H, ROVEC
      CALL RALNM
      RLC
      RET

```

```

;READ ONLY DISKS
;SHIFT IN RIGHT BIT
;PUT IT IN B0

```

```

MAKERO: LHLD ROVEC
      MOV C, L
      CALL ORCURRE
      STA ROVEC
      RET

```

```

;DISKS ALREADY LOCKED
;OR IN BIT FOR CURRENT DISK
;NOW IT'S LOCKED, TOO.

```

```

;CHECK R/O BIT AND GIVE ERR IF SET.

```

```

ERRRO: CALL CHKRO
      RAR
      JNC ERRROX
      LXI H, ERRROX
      PUSH H

```

```

;CHECK IT
;BIT INTO CY
;JUMP IF OK TO WRITE
;ELSE SET THIS RETURN PC
;ON STACK FOR RET

```

ERRROX: RET

7L

VERDIR VERIFIES DIRECTORY SECTORS BY CHECKSUMS. CATCHES H/WARE ERRORS,
AND CATCHES CHANGING DISKS OUT FROM UNDER BDOS.
C/0 SAYS CHECK IT AGAINST LAST VALUE. C/1 SAYS SET NEW VALUE.

```

VERDIR: LXI H,   VERDIT           ;LOCAL TEMP, CHECK CKSM OR NOT
        MOV M,   C
        LDA     DIRSEC           ;SECTOR FOR DIRECTORY
        CPI     $10             ;LEGAL SECTOR FOR DIR?
        JC      VERDI1          ;JUMP IF SO.
        RET                     ;IF NOT, PROB FF (FILE NOT FOUND)

VERDI1: LDA     VERDIT           ;SEE IF CKSUM KNOWN
        RAR
        JNC     VERDI2          ;JUMP IF KNOWN
        CALL    GETCKS          ;COMPUTE CHECKSUM OF THIS BUFFER
        LHL     DIRSEC          ;SECTOR IT CAME FROM
        MVI H,   0              ;16 BIT SECTOR NUMBER
        XCHG
        LHL     VECCKS          ;POINTER TO CHKSUM TABLE
        DAD     D
        MOV M,   A              ;HERE'S THE CHECKSUM, NOW KNOWN
        JMP     VERDI3

VERDI2: LHL     DIRSEC          ;KNOWN. GET SECTOR NUMBER
        MVI H,   0              ;AS AN ADDR
        XCHG
        LHL     VECCKS          ;POINTER TO CHECKSUM TABLE
        DAD     D               ;WHERE CKS OF THIS DIR SEC GOES
        PUSH    H
        CALL    GETCKS          ;COMPUTE CHECKSUM OF THIS BUFFER
        POP     H               ;WHERE WE LAST PUT IT
        CMP     M               ;IS IT THE SAME?
        JZ      VERDI3          ;JUMP IF OK
        CALL    MAKERO          ;MAKE DRIVE UNWRITABLE, DIRECTORY ERRORS

VERDI3: RET

```

UPDATE DIRECTORY SECTOR ON DISK

```

MODIR: MVI C,   $21             ;SET NEW CKSUM INTO TABLE
        CALL    VERDIR          ; ..
        CALL    WRSEC           ;WRITE A SECTOR
        RET

```

SET NEXT DIR SLOT NUMBER. GET ITS SECTOR IN CORE.

```

MAPNXT: LXI H,   MAPNXV         ;SAVE CHECK OR SET CKSUM FLAG
        MOV M,   C
        LDA     DOSFCB          ;LAST FCB NUMBER
        INR     A               ;STEP TO NEXT ONE.
        STA     DOSFCB          ;PUT IT BACK IN CORE
        MOV C,   A              ;COPY IT
        LDA     MAXFCB          ;HIGHEST LEGAL NUMBER?
        CMP     C
        JNC     MAPNX1         ;JUMP IF OK
        LXI H,   DOSFCB        ;ELSE SAY NOT FOUND
        MVI M,   $FF
        RET

```



```

MAPNX1: LDA      DOSFCB      ;NEW FCB NUMBER
        ANI      $03         ;NUMBER WITHIN A SECTOR
        ADD      A           ;AS A BYTE NUMBER IN SECTOR
        ADD      A
        ADD      A
        ADD      A
        ADD      A
        STA      CHROFS      ;CHARACTER OFFSET OF FCB IN DIR BUFFER
        CPI      0           ;FIRST FCB IN A SECTOR?
        JNZ      MAPNX2     ;IF NOT, HAVE IT IN CORE ALREADY
        CALL     GDIRSC      ;COMPUTE DIRECTORY SECTOR NEEDED
        CALL     RDSEC       ;READ A SECTOR
        LHL      MAPNXV      ; 354C 2A DE 3D
        MOV C, L           ;CHECK OR SET CKSUM FLAG
        CALL     VERDIR      ;DO THE CHECKSUMMING
MAPNX2: RET

```

;GET ALLOC BIT FOR CLUSTER IN C

```

GETBTC: LXI H, GETBT      ;CLUSTER NUMBER
        MOV M, C           ;OVER 8 (BITS PER BYTE)
        LDA      GETBT
        ANI      $FC
        RAR
RAR
RAR
MOV C, A                  ;AS AN ADDRESS
        MVI B, 0
        LHL      CURALV    ;CURRENT DISK'S ALLOC VECTOR
        DAD      B         ;ADDR OF BIT IN TABLE
        LDA      GETBT
        ANI      $27       ;NOW THE BIT NUMBER IN BYTE
        INR      A         ;BITS IN A BYTE
        MOV C, A          ;OFFSET BY ONE
        CALL     RLCNM     ;PICK UP (HL) INTO A, AND DO (C) RLC'S
        RET

```

;MARK THE BIT TABLE WITH THE BIT IN E FOR CLUSTER IN C

```

MRKBTB: LXI H, MRKBTW      ; 3572 21 E1 3D
        MOV M, E           ;SAVE CALLER'S C
        DCX      H         ; 3576 2B
        MOV M, C           ;SAVE CLUSTER NUMBER
        LHL      MRKBTW    ;CLUSTER NUMBER AGAIN
        MOV C, L           ;AS AN ADDRESS
        CALL     GETBTC    ;GET ALLOCATION BIT
        ANI      $FE       ;MASK OUT OLD VALUE
        LXI H, MRKBTW      ;POINT TO NEW VALUE
        ORA      M         ;DEPOSIT THE NEW BIT
        PUSH     PSW        ;SAVE NEW BIT
        MVI A, $07         ;MASK FOR BIT NUMBER
        DCX      H         ;BACK TO THE CLUSTER NUMBER
        ANA      M         ;MASK FOR BIT IN BYTE
        INR      A         ;OFFSET BY ONE AGAIN
        MOV C, A          ;COPY FOR SUBR CONVENTION
        POP      PSW        ;NEW BITS AGAIN
        CALL     RALN      ;DO (C) RAL'S
        PUSH     PSW        ;SAVE IT, SHIFTED BACK
        MOV A, M           ;CLUSTER NUMBER AGAIN
        ANI      $FC       ;DIV BY 8 FOR A BYTE NUMBER

```

```

RAR
RAR
MOV C, A      ;BYTE NUMBER MAKES ADDRESS AGAIN
MVI B, 0
LHLD CURALV   ;CURRENT DISK'S ALLOC VECTOR
DAD B         ;WHERE THE CURRENT BYTE LIVES
POP B
MOV C, B      ;UPDATED BYTE
MOV M, C      ;TO VECTOR
RET

;L
;SET OR CLEAR THIS FCB'S ALLOCATION IN BITTABLE

ALOCB: LXI H, ALOCB      ;WILL SET BITS TO THIS (0 OR 1)
MOV M, C
LDA CHROFS               ;CHARACTER OFFSET OF FCB IN DIR BUFFER
ADI $10                  ;MAP AREA
STA ALOCB                ;SAVE MAP ADDR
ALOCB1: LDA CHROFS        ;CHARACTER OFFSET OF FCB IN DIR BUFFER
ADI $1F                  ;SEE IF DONE
LXI H, ALOCB             ;WHERE WE ARE BY NOW
CMP M                    ;DONE?
JC ALOCB3                ;JUMP IF ALL DONE
LHLD ALOCB               ;NOT YET. POINT TO MAP BYTE
MVI H, 0                 ;AS AN ADDRESS
XCHG
LHLD SYSDMA              ;IN BUFFER
DAD D                    ;..
MOV A, M                 ;GET A MAP BYTE
STA ALOCB                ;KEEP A COPY OF IT
CPI 0                    ;IS THIS CLUSTER FREE?
JZ ALOCB2                ;IF SO, DON'T (DE)ALLOCATE IT
LHLD ALOCB               ;NO, MUST (DE)ALLOCATE IT
MOV C, L                 ;CLUSTER NUMBER
LHLD ALOCB               ;0 OR 1 FOR THAT CLUSTER
XCHG
CALL MRKSTB              ;PUT BIT IN BIT TABLE
ALOCB2: LXI H, ALOCB      ;STEP TO NEXT BYTE OF MAP AREA
INR M
JNZ ALOCB1               ;LOOP UNTIL DONE
ALOCB3: RET

;L
;READ THE DIRECTORY AND ALLOCATE ALL FILES IN BIT TABLE

RDALOC: LXI H, ANSARL      ; 35E0 21 F0 3C
MVI M, 0                  ; 35E3 36 00
LXI H, ROVEC              ; 35E5 21 7C 3D
MVI M, 0
LHLD CURALV               ;CURRENT DISK'S ALLOC VECTOR
LDA INIALO
MOV M, A                  ;ALLOCATE THE DIRECTORY CLUSTERS
LXI H, RDALOV             ;ADDR WITHIN ALLOC VECTOR
MVI M, $01                ;START WITH BYTE AFTER ABOVE INIT
RDALOB1: MVI A, NALLOV-1   ;DONE THEM ALL?
LXI H, RDALOV             ;..
CMP M                     ;..
JC RDALOB2                ;JUMP IF SO
LHLD RDALOV               ;NO, DO ANOTHER

```

LHLD	XCHG CURALV		;CURRENT DISK'S ALLOC VECTOR
	DAD	D	;BYTE IN ALLOC VECTOR
	MVI M,	0	;CLEAR ALLOCATION
	LXI H,	RDALOV	;STEP THRU VECTOR
	INR	M	; ..
	JNZ	RDALO1	;LOOP TILL DONE WHOLE VECTOR THIS DSK
RDALO2:	CALL	HOMSET	;HOME THE DISK
	LXI H,	DOSFCB	;SAY NO FCB NOW OPEN
	MVI M,	\$FF	
RDALO3:	MVI C,	\$01	;MAP AN FCB, SETTING UP ALLOC VEC
	CALL	MAPNXT	;GET NEXT FCB NUM, MAP ITS SECTOR INTO CORE
	LDA	DOSFCB	;DONE ALL FCB'S?
	CPI	\$FF	;WHEN ALL DONE, RETURNS FF
	JNZ	RDALO4	;KEEP ON TILL HIGHEST LEGAL NUMBER
	RET		
RDALO4:	LHLD	CHROFS	;CHARACTER OFFSET OF FCB IN DIR BUFFER
	MVI H,	0	;COMPUTE ADDR IN BUFFER
	XCHG		
LHLD	SYSDMA		
	DAD	D	
	MOV A,	M	;GET FIRST CHAR OF FCB
	CPI	\$E5	;DISK FILL CHAR, = BLANK ENTRY
	JZ	RDALOS	;IF SO, THIS FCB IS A FREE ONE.
	LHLD	CHROFS	;CHARACTER OFFSET OF FCB IN DIR BUFFER
	MVI H,	0	;NOT FREE. COMPUTE UP THE NAME ADDR
	LXI B,	\$2001	;WHERE NAMES START
	DAD	B	
	XCHG		
LHLD	SYSDMA		
	DAD	D	
	MOV A,	M	;GET FIRST CHAR OF NAME
	SUI	CH.DOL	
	SUI	\$01	;MAKE 8-BIT MATCH FLAG
	SBB	A	;A/FF IF CHAR WAS DOLLAR, ELSE A/0
	LXI H,	ANSARL	
	CRA	M	;SET ANSWER TO ONES IF ANY \$'S FOUND
	MOV M,	A	; ..
	MVI C,	\$01	;IN ANY CASE, ALLOCATE THE SPACE
	CALL	ALOFCB	;SET ALLOCATION FOR THIS FCB IN BITTABLE
RDALOS:	JMP	RDALO3	;LOOP FOR ALL FCB'S
	RET		;EXTRA RET FROM COMPILER
;GET NEXT MATCHING FILE			
STPFIL:	LHLD	STPFIV	;USER'S FCB FOR PREV LOOKUP
	SHLD	USERDE	;RE-USE IT
STPFI5:	MVI C,	0	;REMEMBER TO CHECK CHECKSUMS
	CALL	MAPNXT	;GET NEXT FCB NUM, MAP ITS SECTOR INTO CORE.
	LDA	DOSFCB	;WHICH ONE DID IT FIND?
	STA	ANSARL	;SAVE THE ANSWER
	CPI	\$FF	;WERE ANY FOUND?
	JNZ	STPFI1	;JUMP IF SO
	RET		;RETURN IF NOT
STPFI1:	LXI H,	STPFI1	;NUMBER CHARS CHECKED SO FAR
	MVI M,	0	;INITIALLY, NONE CHECKED
STPFI2:	LXI H,	STPFI1	;# CHARS TO CHECK

```

LDA      STPFIX      ;NUMBER DONE
SUB      M            ;ALL DONE?
SBB      A            ;SET TO FF UNLESS ALL CHECKED, THEN 0
LHLD     STPFIX      ;CHAR POSITION IN FCB
MVI H,   0            ;AS AN ADDR
XCHG
LHLD     USERDE
DAD      D
PUSH     PSW          ;SAVE ALL-CHECKED FLAG
MOV A,   M            ;GET THIS CHAR FROM USER FCB
STA      STPFIX      ;SAVE USER'S CHAR
LDA      STPFIX      ;NUMBER CHECKED SO FAR
PUSH     H
LXI H,   CHROFS      ;CHARACTER OFFSET OF FCB IN DIR BUFFER
ADD      M            ;CURRENT CHAR NUMBER IN BUFFER
MOV C,   A            ;AS ADDRESS
MVI B,   0            ;AS ADDR IN BUFFER
LHLD     SYSDMA
DAD      B
POP      B            ;BACK TO USER FCB
LDAX     B            ;CHAR FROM USER'S FCB
SUB      M            ;COMPARE THEM
SUI      $D1          ;IF MATCH, SET CY
SBB      A            ;AND MAKE FF IF MATCH, ELSE 0
PUSH     PSW          ;SAVE MATCHED CHAR FLAG
LDA      STPFIX      ;USER'S CHAR
SUI      $3F          ;QUESTIONMARK? (WILDCARD)
SUI      $D1          ;MAKE FLAG FOR THAT
SBB      A
POP      B            ;MATCHED CHAR THIS TIME FLAG
MOV C,   B            ;ONES IF MATCHED OR WILDCARD
ORA      C
POP      B
MOV C,   B            ;MORE CHARS LEFT FLAG
ANA      C
RAR
JNC      STPF13        ;JUMP IF ALL DONE OR DIDNT MATCH
LXI H,   STPFIX      ;HAVE TO CHECK SOME MORE
INR      M            ;STEP TO NEXT CHAR NUMBER
JMP      STPF12        ;GO DO ANOTHER CHAR

```

HERE IF NON-MATCH OR ALL DONE.

```

STPF13: LXI H,   STPFIV      ;NUMBER CHARS TO CHECK
LDA      STPFIX      ;NUMBER CHECKED
CMP      M            ;ALL DONE?
JNZ      STPF14        ;IF NOT, NON-MATCH. QUIT.
RET       ;ELSE ALL OK, SO RETURN

```

```

STPF14: JMP      STPF15      ;GO TRY ANOTHER FCB
RET       ;EXTRA RET FROM COMPILER

```

ROUTINE TO LOOK FOR A FILENAME, OR SPECIFIC EXTENT.

```

LOOKUP: LXI H,   LOOKUV      ;SAVE NUMBER CHAR TO VERIFY
MOV M,   C            ;HERE
LDA      LOOKUV
STA      STPFIV      ;SAVE FOR LATER SCAN
LHLD     USERDE      ;COPY USER'S FCB ADDR

```

```

SHLD    STPFIL      ;FOR LATER
LXI H,  DOSFCB      ;ASSUME FAILURE
MVI M,  $FF         ;FLAGGED BY FF
CALL    HOMSET      ;HOME AND SET TRACK
CALL    STPFIL      ;GET NEXT MATCHING FILE
RET

```

;FUNCTION ROUTINE FOR DELETE FILE CALLS THIS TO DO THE WORK.

```

DDEL:   CALL    ERRRO      ;CHECK R/O AND ERROR OUT IF SET
        MVI C,  $0C      ;CHECK THIS MANY CHARS
        CALL    LOOKUP    ;FIND USER'S FILE
DDEL1:  LDA     DOSFCB      ;WAS THERE ONE?
        CPI     $FF
        JNZ     DDEL2     ;JUMP IF SO
        RET              ;ELSE JUST RETURN

```

;DOING A DELETE, AND THERE IS A FILE TO DO.

```

DDEL2:  MVI C,  0        ;DEALLOCATE FLAG FOR BITTABLE
        CALL    ALOFCB    ;CLEAR ALLOC FOR THIS FCB IN BITTABLE
        LHL    CHROFS    ;CHARACTER OFFSET OF FCB IN DIR BUFFER
        MVI H,  0        ;AS AN ADDR
        XCHG
LHLD    SYSDKA
        DAD     0
        MVI M,  $E5      ;DATA FILL CHAR, MAKING FREE FCB
        CALL    UPDIR     ;UPDATE DIR SEC ON DISK
        CALL    STPFIL    ;GET NEXT MATCHING FILE OR EXTENT
        JMP     DDEL1     ;LOOP FOR ALL
        RET              ;EXTRA RET FROM COMPILER

```

;FROM WRITE NEXT RECORD ROUTINE
; FIND A FREE CLUSTER NEAR CLUSTER IN C.

```

ALLOC:  LXI H,  ALO.LO    ;SAVE DESIRED CLUSTER
        MOV M,  C
        LDA     ALO.LO    ;AS BOTH LOW AND HIGH TESTS
        STA     ALO.HI    ; ..
ALLOC1: LXI H,  MXCLUS    ;NUMBER OF CLUSTERS ON DISK
        LDA     ALO.HI    ;TEST HIGH FOR BEING OFF TOP
        SUB     M
        SBB     A
        PUSH    PSW       ;SAVE HI LEGAL FLAG
        MVI A,  0        ;DOWN TO 0 ON LOW SIDE?
        LXI H,  ALO.LO
        SUB     M         ;I.E., IS LO LEGAL?
        SBB     A
        PCP     B         ;HIGH LEGAL FLAG
        MOV C,  B         ;INEFFICIENCY OF COMPILER
        ORA     C         ;SEE IF EITHER ONE LEGAL
        RAR
JNC     ALLOC4            ;JUMP IF NEITHER ONE GOOD
        MVI A,  0        ;NOW SEE WHICH IS OK.
        LXI H,  ALO.LO    ;BY WASTEFULLY RECOMPUTING
        SUB     M         ;WHETHER .LO IS LEGAL
        SBB     A
        ANI     $01       ;1 IF NOT ZERO
        PUSH    PSW

```

```

MOV A, M          ;LO
POP B
MOV C, B          ;1 IF NOT FREE
SUB C              ;DESIRED -(1 OR 0)
MOV M, A          ;STORE NEW TARGET
INX H              ;ALO.HI
MOV A, M
LXI H, MxCLUS     ;NUMBER OF CLUSTERS ON DISK
SUB M              ;IS HIGH AT MAX?
SBB A              ;IF LEGAL, MAKE FF
ANI $01            ;1 IF LEGAL, 0 IF NOT
LXI H, ALO.HI      ;STEP HI UP IF LEGAL
ADD M              ;..
MOV M, A           ;..
LHLD ALO.HI        ;SEE IF LO IS FREE
MOV C, L
CALL GETBTC        ;GET ITS ALLOCATION BIT
RAR                ;TEST IT
JC ALLOC2           ;JUMP IF NOT FREE
LDA ALO.HI         ;IT'S FREE, USE IT.
RET

```

7L
; MORE OF SEARCH FOR FREE CLUSTER

```

ALLOC2: LHLD ALO.LO      ;SEE IF THE LOW SIDE GUY IS FREE
MOV C, L
CALL GETBTC         ;GET ITS ALLOC BIT
RAR
JL ALLOC3           ;JUMP IF NOT FREE
LDA ALO.LO          ;FREE. USE IT.
RET

```

```

ALLOC3: JMP ALLOC1      ;GO BACK AND TRY AGAIN

```

```

ALLOC4: MVI A, 0        ;THERE ARE NO FREE CLUSTERS
RET

```

;COPY NAME STRING INTO SYSTEM DIRECTORY. FOR MAKE OR RENAME.

```

CPYNAM: LXI H, CPYNAM    ;TWO ARGUMENTS
MOV M, E              ;COUNT OF CHARS TO MOVE
DCX H
MOV M, C              ;WHERE IN USER FCB TO GET CHARS
CPYNM1: LDA CPYNAM       ;GET THE COUNT
DCR A                 ;COUNT IT DOWN
STA CPYNAM            ;IN CORE, TOO
CPI $FF               ;ALL DONE?
JZ CPYNM2              ;JUMP IF SO
LDA CPYNAM             ;IF NOT, GO GET ANOTHER CHAR FROM FCB
LXI H, CPYNAV          ;AT THIS OFFSET IN USER FCB
ADD M
MOV C, A              ;MAKE 16 BIT ADDR
MVI B, 0
LHLD USERDE           ;IN FCB
DAD B
LDA CHROFS             ;CHARACTER OFFSET OF FCB IN DIR BUFFER
PUSH H                ;SAVE WHERE USER'S CHAR IS
LXI H, CPYNAM          ;COMPUTE WHERE IT IS IN DIR BUFFER
ADD M
MOV C, A              ;AGAIN, 16 BIT ARITHMETIC

```



```

        MVI B, 0
        LHLD SYSDMA      ;IN SYSTEM BUFFER
        DAD B
        POP B            ;WHERE USER CHAR IS
        LDAX B           ;GET USER CHAR
        MOV M, A         ;PUT IN SYSTEM DIR
        JMP CPYNM1       ;SEE IF ALL DONE

CPYNM2: CALL 6DIRSC      ;COMPUTE DIR SECTOR AND 1K
        CALL UPDDIR      ;GO WRITE IT
        RET

;L
;COPY A WHOLE FCB

CPYFCB: MVI E, $20
        MVI C, 0
        CALL CPYNAM
        RET

;RENAME FILE ROUTINE

DOREN:  CALL ERRRO      ;CHECK R/O AND ERROR OUT IF SET
        MVI C, $0C      ;CHECK THIS MANY CHARS
        CALL LOOKUP     ;TRY TO FIND A FILE
        LHLD USERDE     ;POINT TO NEW NAME
        LXI B, $2016    ;IT STARTS HERE
        PUSH H          ;SAVE START OF FCB
        LHLD USERDE
        DAD B
        POP D
        LDAX D
        MOV M, A        ;GET IT
DOREN1: LDA DOSFCB      ;THERE WAS A FILE, WASN'T THERE?
        CPI $FF
        JZ DOREN2       ;JUMP IF NOT
        MVI E, $0C      ;COPY THIS MANY CHARS
        MVI C, $10      ;FROM HERE IN USER'S FCB
        CALL CPYNAM     ;COPY NEW NAME INTO DISK DIR
        CALL STPFIL     ;GET NEXT MATCHING FILE OR EXTENT
        JMP DOREN1      ;LOOP TILL ALL FCB'S FOUND

DOREN2: RET             ;SPARE RET FROM COMPILER

;L
;OPEN FILE ROUTINE. DO LOOKUP, COPY MAP AREA FROM DISK DIR TO FCB.

OPENFI: MVI C, $20
        CALL LOOKUP     ;CHECK THIS MANY CHARS
        LDA DOSFCB      ;GO FIND USER'S NAME
        CPI $FF         ;GET ANSWER
        JZ OPENNF       ;WAS IT THERE?
        LXI H, OPNFIV    ;IF NOT, GO GIVE ERROR
        MVI M, $0D       ;COPY MAP AREA INTO FCB IN USER SPACE
        MVI A, $1F      ;INITIAL INDEX
        LXI H, OPNFIV    ;END TEST. DONE THEM ALL?
        CMP M
        JC OPENNF       ;JUMP IF DONE
        LDA OPNFIV       ;MORE TO MOVE. BUILD ADDRESS
        LXI H, CHROFS    ;CHARACTER OFFSET OF FCB IN DIR BUFFER
        ADD M

```

MVI B,	0	
LHLD	SYSDMA	; IN SYSTEM'S DIR COPY
DAD	B	
PUSH	H	
LHLD	OPNFIV	; MOVE IT TO SAME PLACE IN USER FCB
MVI H,	0	
XCHG		
LHLD	USERDE	; WHICH IS HERE
DAD	D	
POP	B	; GET POINTER TO SYS DIR CHAR
LDAX	B	; GET THE CHAR
MOV M,	A	; PUT IT IN USER FCB
LXI H,	OPNFIV	; STEP THE INDEX
INR	M	; ..
JNZ	OPNF11	; LOOP TILL ALL COPIED
CPENNF:	RET	
; CLOSE FILE LOGIC		
CLOSFI:	LXI H, ANSARL	; WHERE USER GETS ANSWER
	MVI M, 0	; SUCCESS FLAG
	CALL CHKRO	; IS DISK WRITABLE?
	RAR	; ..
	JNC CLSFI1	; JUMP IF SO
	RET	; RETURN IF NOT. IT'S CLOSED.
CLSFI1:	MVI C, 320	; THIS MANY CHARS
	CALL LOOKUP	; GET THE FCB
	LDA DOSFCB	; THERE WAS ONE, YES?
	CPI 3FF	
	JZ CLSFI2	; JUMP IF NO
	CALL CPYFCB	; COPY WHOLE FCB, UPDATING SIZE, ALLOC...
CLSFI2:	RET	
; MAKE FILE ROUTINE		
MAKEFI:	CALL ERRRO	; CHECK R/O AND ERROR OUT IF SET
	LHLD USERDE	; 3845 2A 0A 33
	SHLD MAKFIV	; SAVE NAME TO MAKE
	LXI H, FRECHR	; LOOK FOR A FREE FILE
	SHLD USERDE	; THAT'S A FILENAME, SORT OF.
	MVI C, 321	; ITS ONE CHAR LONG
	CALL LOOKUP	; LOOK FOR A FILE BY THAT NAME
	LDA DOSFCB	; ANY FREE?
	CPI 3FF	
	JZ MAKFI3	; JUMP IF NO ROOM
	LHLD MAKFIV	; GET BACK NAME TO MAKE
	SHLD USERDE	; USER'S FCB
	LXI H, MAKFIV	; HERE WE COUNT THE CHARS
	MVI M, 320	; INITIAL CHAR TO CHECK
MAKFI1:	MVI A, 31F	; DONE?
	LXI H, MAKFIV	
	CMP M	
	JC MAKFI2	; JUMP WHEN DONE
	LHLD MAKFIV	; COMPUTE ADDR IN USER'S FCB
	MVI H, 0	
	XCHG	
LHLD	USERDE	; USER'S FCB ADDR
	DAD D	
	MVI M, 0	; CLEAR THESE CHARS

```

        LXI H,  MAKFIV
        INR     M
        JNZ     MAKFI1
MAKFI2: CALL    CPYFCB
MAKFI3: RET

```

```

;DO THEM ALL
;STEP THRU THEM
;LOOP TILL ALL DONE
;COPY WHOLE FCB TO DISK BUFFER

```

```

;STEP TO THE NEXT EXTENT

```

```

STPXTN: LXI H,  STPXTV
        MOV H,  C
        CALL   CLOSFI
        LDA    DOSFCB
        CPI    $FF
        JNZ    STPXT1
        RET

```

```

;SAVE RD/WR FLAG
;CLOSE THE OLD EXTENT
;DID THAT WORK?
; ..
;JUMP IF SO
;FAILED TO CLOSE.

```

```

STPXT1: LXI B,  $200C
        LHLD   USERDE
        DAD    B
        INR    M
        MVI C,  $2D
        CALL   LOOKUP
        LDA    DOSFCB
        CPI    $FF
        JNZ    STPXT3
        LDA    STPXTV
        RAR
        JNC    STPXT2
        RET

```

```

;EXTENT BYTE
;IN USER'S FCB
; FOR FILE JUST CLOSED
;STEP TO NEXT EXTENT
;LOOK FOR THAT EXTENT
; ..
;FIND IT?

;JUMP IF SO
;NOT FOUND. WHAT NEXT?
;WERE WE READING OR WRITING?
;JUMP IF WRITING
;READING. JUST EOF, THEN.

```

```

STPXT2: CALL    MAKEFI
        JKP     STPXT4

```

```

;WRITING. MAKE ANOTHER EXTENT.
;IT'S OPEN.

```

```

STPXT3: CALL    OPENFI
STPXT4: LDA     DOSFCB
        CPI     $FF
        JNZ     STPXT5
        LXI H,  ANSARL
        MVI M,  $21
        RET

```

```

;OPEN THE NEXT EXTENT
;OPENED OK?

;JUMP IF SO
;NOT ABLE TO OPEN NEXT EXTENT.
;GIVE THIS ERROR RETVALUE
;RETURN TO USER

```

```

STPXT5: CALL    STNRRC
        LXI H,  ANSARL
        MVI M,  0
        RET

```

```

;GET NR AND RC FROM FCB
;USER ANSWER
;IS SUCCESS

```

```

SETDMA: LDA     FRZDMA
        RAR
JNC     SETDM1
        LHLD   USRDMA
        MOV B,  H
        MOV C,  L
        CALL   CBSTMA
SETDM1: RET

```

```

;DMA FROZEN?

;JUMP IF SO
;ALLOWED TO MOVE.
;SET TO WHAT USER LAST ASKED FOR
;BIOS SET DMA ADDRESS

```

```

;RESET DMA TO STANDARD

```

```

RSTDMA: LDA     FRZDMA
        RAR
        INC     RSTDMY

```

```

;ALLOWED TO MOVE DMA?

```

```

;IF FROZEN. SKIP THIS.

```

LHLD	SYSDMA	OK TO MOVE IT.	
MOV B,	H		38F0 44
MOV C,	L		38F1 4D
CALL	CBSTMA	BIOS SET DMA ADDRESS	
RSTDMX: RET			38F5 C9

HERE TO READ NEXT RECORD

RDNXTR: CALL	STNRRC	GET NR AND RC FROM FCB
LDA	CURRNR	WHERE WERE WE?
LXI H,	CURRRC	HOW FAR CAN WE GO?
CMP	M	OVERDONE IT?
JC	RDNXT2	JUMP IF OK
LXI H,	ANSARL	IF TOO FAR, GIVE EOF RETURN
MVI M,	\$01	TO USER'S ANSWER
LDA	CURRNR	WERE WE TO END OF EXTENT?
CPI	\$80	..
JNZ	RDNXT1	JUMP IF NOT
MVI C,	\$01	YES. SAY READING
CALL	STPXIN	AND TRY FOR ANOTHER EXTENT
RDNXT1: LXI H,	CURRNR	SAY FIRST RECORD IN EXTENT
MVI M,	0	
LDA	ANSARL	WHAT WAS ANSWER?
CPI	0	SUCCESS?
JZ	RDNXT2	IF SO, MORE TO DO.
RET		ELSE GIVE UP HERE.
RDNXT2: CALL	GTMAPB	GET MAP BYTE FOR CURRENT NR
LHLD	CURRMW	THAT'S CLUSTER NUMBER
MOV A,	L	AS 8 BIT NUMBER
CPI	0	IS IT ALLOCATED?
JNZ	RDNXT3	JUMP IF SO
LXI H,	ANSARL	NOT ALLOCATED. SAY EOF.
MVI M,	\$01	ALLOCATION CAN'T BE SPARSE.
JMP	RDNXT4	SO QUIT ON FIRST ZERO
RDNXT3: CALL	COMPSN	GET SECTOR NUMBER, FROM CURRMW, CURRNR
CALL	COMPTK	AND COMPUTE THE TRACK FROM THAT
CALL	SETDMA	SET UP TO USER'S (OR STD) DMA ADDR
CALL	RDSEC	READ A SECTOR
CALL	RSTDMA	BACK TO WHAT USER SAID
CALL	UDNRRC	UPDATE NR AND RC IN FCB
RDNXT4: RET		END OF READ-NEXT-RECORD

WRITE NEXT RECORD ROUTINE

WTNXTR: CALL	ERRRO	CHECK R/O AND ERROR OUT IF SET
CALL	STNRRC	GET NR AND RC FROM FCB
MVI A,	\$7F	ABOUT TO GO OFF END OF EXTENT?
LXI H,	CURRNR	WEREN'T ABLE TO MAKE A NEW EXTENT, THEN.
CMP	M	
JNC	WTNXT1	JUMP IF WITHIN AN EXT
LXI H,	ANSARL	REPLY FULL.
MVI M,	\$01	COULDN'T EXTEND FILE.
JMP	WTNXT8	RETURN TO USER
WTNXT1: CALL	GTMAPB	GET CLUSTER NUMBER FROM MAP
LHLD	CURRMW	PICK IT UP
MOV A,	L	

```

CPI      0
JNZ      WTNXT4
LXI H,   WTNXTV
MVI M,   0
LDA      CURRNR
ANJ      $FC
RAR

RAR
RAR
ADI      $10
INX      H
MOV M,   A
MOV C,   A
MVI A,   $10
CMP      C
JNC      WTNXT2
LDA      WTNXTV
DCR      A
MOV C,   A
MVI B,   0
LHLD     USERDE
DAD      B
MOV A,   M
STA      WTNXTV
WTNXT2: LHLD     WTNXTV
MOV C,   L
CALL     ALLOC
STA      WTNXTV
CPI      0
JNZ      WTNXT3
LXI H,   ANSARL
MVI M,   $02
JMP      WTNXT4

WTNXT3: LHLD     WTNXTV
MOV C,   L
MVI E,   $01
CALL     MRKBTB
LHLD     WTNXTV
MVI H,   0
SHLD     CURRMW
LHLD     WTNXTV
MVI H,   0
XCHG
LHLD     USERDE
DAD      0
LDA      WTNXTV
MOV M,   A
WTNXT4: LDA      ANSARL
CPI      0
JNZ      WTNXT8
CALL     COMPSN
CALL     COMPTK
CALL     SETDMA
CALL     WRSEC
CALL     RSTDMA
LDA      CURRNR
LXI H,   CURRRC
CMP      M
JIS IT ALLOCATED?
JUMP IF SO
;NO, MUST ALLOCATE IT
;PREV ALLOC = 0
;CONVERT TO CLUSTER NUMBER
;BY DIVIDING BY 8

;STEP INTO FCB TO MAP AREA
;TO WTNXTV
;SAVE FCB OFFSET FOR THIS CLUSTER
;IS IT THE INITIAL ONE OF FILE?

; ..
;JUMP IF SO
;AFTER FIRST CLUSTER
;BACK UP ONE, SEE WHERE IT WAS
;GET THAT MAP BYTE

;OFFSET INTO USER'S FCB
;FINALLY, GET CLUSTER NUMBER
;STORE IT
;NOW GET DESIRED NEIGHBOR
;IN RIGHT AC FOR SUBR
;ALLOCATE A CLUSTER NEAR IT
;HERE'S THE ONE WE CAN HAVE
;OR WAS IT FULL?
;JUMP IF OK
;FULL. USER'S ANSWER CELL
;PUT FULL FLAG IN IT
;GO FINISH UP

;HERE IS THE NEW CLUSTER
;PUT A ONE IN BIT TABLE
; ..
;SAVE AS CURRENT CLUSTER
; FULL WORD QUANTITY
; ..
;HERE'S THE FCB SLOT WHICH GETS IT
;FIGURE ADDRESS FOR IT

; IN USER'S FCB
;THE CLUSTER NUMBER AGAIN
;INTO THE FCB
;WAS THERE AN ERROR YET?
; ..
;YES, SO DON'T WRITE.
;GET SECTOR NUMBER, FROM CURRMW, CURRNR
;AND THE TRACK FROM THAT
;SET UP TO USER'S (OR STD) DMA ADDR
;WRITE A SECTOR
;BACK TO WHAT USER SAID
;THIS RECORD
;UP TO COUNT IN FCB?
; ..

```

```

LDA    CURRNR
INR    A
STA    CURRRC
NEXT5: LDA    CURRNR
CPI    $7F
JNZ    WINTXT7
CALL   UDNRRC
MVI C, 0
CALL   STPXTN
LDA    ANSARL
CPI    0
JNZ    WINTXT6
LXI H, CURRNR
MVI M, $FF
NEXT6: LXI H, ANSARL
MVI M, 0
NEXT7: CALL   UDNRRC
NEXT8: RET

```

;YES. CURRENT PLUS 1 TO MAX

```

;FOR LATER WRITING ON DISK
;DID WE JUST FILL EXTENT?

```

;NORMALLY, NO.

;UPDATE NR AND RC IN FCB

;SAY WRITING

;AND TRY FOR ANOTHER EXTENT

;WAS IT SUCCESSFUL?

;JUMP IF SO

;PUT A BAD NR, FOR NEXT WRITE.

;WILL GET EOF IF TRIES AGAIN.

;SAY THIS WRITE WAS OK, THOUGH.

;UPDATE NR AND RC IN FCB

LOG IN THE DISK IN CURDSK

```

LOGCUR: MVI A, $03
LXI H, CURDSK
CMP     M
JNC     LOGCU1
LXI H, LOGCU1
PUSH    H
LHLD    VSELER
PCHL

```

;RANGE CHECK THE DRIVE NUMBER

;CHECK USER'S ARG

;LEGAL?

;JUMP IF OK

; 3A21 21 29 3A

; 3A24 E5

;HANDLE BAD UNIT

; 3A28 E9

```

LOGCU1: LDA    CURDSK
ADD     A
ADD     A
ADD     A
ADD     A
ADD     A
MOV C, A
MVI B, 0
LXI H, ALLOCS
DAD     B
SHLD    CURALV
LDA     CURDSK
ADD     A
ADD     A
ADD     A
ADD     A
MOV C, A
MVI B, 0
LXI H, CKSTAB
DAD     B
SHLD    VECCKS
LHLD    CURDSK
MVI H, 0
LXI B, TKPTRS
DAD     B
SHLD    TKPNTR
LHLD    CURDSK
MVI H, 0
LXI B, TSEC0V

```

;DISK NUMBER

;TIMES 32 BYTES PER DISK

;MAKE ADDRESS

;BASE OF ALLOC AREA

;FOR THIS DISK

;CURRENT DISK'S ALLOC VECTOR

;DISK NUMBER * 16

;AS AN ADDRESS

;SPACE FOR CHECKSUMS

;BASE FOR CKSM'S OF THIS DISK'S DIR

;POINTER TO CKSUM TABLE

;COMPUTE POSITION IN TRK POINTERS TABLE

;FOR THIS DISK

; ..

; ..

;DEFINE WHERE TRACK NUMBER LIVES

;FOR CURRENT DISK

;SIMILARLY FOR SECTOR 0 ON THAT TK

;VECTOR OF SECTOR 0 ON TK

;DOUBLE THE DISK NUMBER. TO MAKE WORD


```

DAD      B
SHLD     TKSEC0
LHLD     CURDSK
MOV C,   L
CALL     CBSELE
; THE ONLY SUCH CALL
LDA      LOGVEC
RLC
PUSH     PSW
LDA      CURDSK
INR      A
MOV C,   A
POP      PSW
CALL     RALN
RAR
JC        LOGCU2
LHLD     LOGVEC
MOV C,   L
CALL     ORCURR
STA      LOGVEC
CALL     RDALOC

```

LOGCU2: RET

;LOG IN AND SELECT A DISK

```

LGNDISK: LXI H, CURDSK
LDA      ANSARG
CMP      M
JZ       LGNDS1
LDA      ANSARG
STA      CURDSK
CALL     LOGCUR

```

LGNDS1: RET

```

SETDSK: LHLD     USERDE
MVI A,   $1F
ANA      M
DCR      A
STA      ANSARG
CPI      $1E
JNC      SETDS1
LDA      CURDSK
STA      RELOGN
LHLD     USERDE
MOV A,   M
STA      RELOGF
LHLD     USERDE
MVI A,   $E0
ANA      M
MOV M,   A
CALL     LGNDISK

```

SETDS1: RET

;SET DMA FROM HERE. ONE CALL, SETS TO DEFDMR

```

DEFDMA: LHLD     USERDE
SHLD     SYSDMA
MOV B,   H
MOV C,   L
CALL     CBSTMA

```

;IN THIS TABLE

;NOW LET I/O PACKAGE KNOW IT

;BIOS - SELECT A DISK. THIS IS

;CURRENTLY LOGGED DISKS

;SEE IF IT WAS KNOWN ALREADY

; ..

;COMPUTE UP BIT NUMBER FOR DISK

;SAVE NUMBER

;GET BACK BITS

;DO (C) RAL'S

;CHECK THE BIT

;JUMP IF ALREADY KNOWN

;CURRENTLY LOGGED DISKS

;TO RIGHT AC

;OR IN BIT FOR CURRENT DISK

;ADD TO KNOWN DISKS

;SET UP BIT TABLE FROM ALL DIR FCB'S

;NEWLY REQUESTED DISK

;SAME DISK AS BEFORE?

;YES.

;NO, SO CHANGE TO IT

;UPDATE

;AND LOG IT IN

;GET FCB ADDR

;LOW 5 BITS FOR DRIVE

;ENTRY TYPE FIELD. USUALLY 0.

;CONVERT TO 0-3

;DISK USER HAS IN FCB

;DID USER SPECIFY ONE?

;NORMALLY ZERO, THAT'S ALL.

;SAVE CURRENT LOGGED DISK

;MRETN WILL RE-LOG FROM HERE

;PICK UP FCB'S ENTRY TYPE

;FLAG NEED TO RE-LOG

;NOW CLEAR WHAT USER SAID

;LOG IN AND SELECT REQUESTED DISK

;GET SYSTEM'S DMA ADDR ARG

;SAVE IN THIS SEMI-CONSTANT

;SWITCH REGISTERS FOR BIOS

;VECTOR SET DMA (BIOS)

RET

WORK ROUTINE FOR USER'S CALLS, FROM BDOSH, FROM 5

```
BDOS00: LHL    USERDE
        MOV A, L        ;USER'S "E"
        STA     ANSARG   ;SAVE IT
        LXI H, 0
        SHLD    ANSWER   ;DEFAULT ANSWER IS "0,0"
        MOV A, L        ; JAE0 7D
        STA     ANSARL   ;0 TO "A" ANSWER (SEE T353)
        STA     RELOGF   ;ASSUME NO NEED TO RE-LOG
        LHL     USRFCN   ;GET USER'S FUNCTION CODE
        MOV C, L        ; ..
        MVI B, 0        ;AS A FULL ADDRESS
        LXI H, FCNDTB   ;COMPUTE DISP TABLE ADDRESS
        DAD     B        ; ..
        DAD     B        ; ..
        MOV E, H        ;PICK UP THE ROUTINE ADDR
        INX     H
        MOV D, H        ;TWO BYTES
        XCHG         ;AND GO TO IT.
        PCHL          ; ..
```

THERE ARE THE FUNCTION HANDLERS, FCN0 THRU FCN30

```
FCN0:  CALL     CBWBUT   ;DO A WARM RE-BOOT, IN BIOS
        JMP      MRETN

FCN1:  CALL     RDCONS   ;READ CONSOLE CHAR
        STA     ANSARL   ;SAVE ANSWER
        JMP      MRETN

FCN2:  LHL     ANSARG
        MOV C, L        ;TYO ROUTINE
        CALL    CTY02    ;USER'S D
        JMP      MRETN   ;TYPE, WITH TAB SIMULATION

FCN3:  CALL     CBPTRI   ;READ FROM "READER", BIOS
        STA     ANSARL   ;SAVE ANSWER
        JMP      MRETN

FCN4:  LHL     ANSARG
        MOV C, L        ;WRITE ON "PUNCH"
        CALL    CBPTPO   ;CHAR TO WRITE
        JMP      MRETN   ;IN BIOS

FCN5:  LHL     ANSARG
        MOV C, L        ;WRITE ON "LISTING"
        CALL    CBLPTO   ;CHAR TO LIST
        JMP      MRETN   ;IN BIOS

FCN6:  LHL     •BDOS•+1  ;GET ADDRESS OF BDOS
        SHLD    ANSWER   ;FOR USER
        JMP      MRETN

FCN7:  LHL     $0003
        MVI H, 0        ;GET I/O STATUS BYTE
        SHLD    ANSWER   ;ONLY THE "L" PART
                        ;SAVE AS ANSWER
```

	JMP	MRETN		
FCN8:	LDA STA JMP	ANSARG \$0003 MRETN	;SET I/O STATUS BYTE ;EASY?	
FCN9:	LHLD MOV B, MOV C, CALL JMP	USERDE H L DLSOUT MRETN	;PRINT STRING, TERM BY "\$" ;SWAP AC'S ;WE HAVE A SUBROUTINE FOR THAT	
FCN10:	CALL JMP	PSIN MRETN	;READ STRING, BUFFERED (PSIN)	
FCN11:	CALL STA JMP	CHKKBD ANSARL MRETN	;CHECK KEYBOARD FLAG ;SAVE ANSWER	
FCN12:	JMP	MRETN	;LIFT HEAD (NOP)	
FCN13:	LXI H, MVI M, LXI H, MVI M, LXI H, MVI M, LXI H, SHLD CALL CALL JMP	CURDSK 0 LOGVEC 0 FRZDMA 0 \$0080 USERDE DEFDMA LOGCUR MRETN	;INITIALIZE BDOS ;SAY ALLOWED TO MOVE DMA AROUND ; .. ;DEFAULT DMA ADDR	; 3B6A 36 00 ; 3B6C 21 C1 3D ; 3B6F 36 00 ; 3B79 22 0A 33 ; 3B7C CD C7 3A ; 3B7F CD 18 3A
FCN14:	CALL JMP	LGNDISK MRETN	;SELECT AND LOG IN DISK	
FCN15:	CALL CALL JMP	SETDSK OPENFI MRETN	;OPEN FILE	
FCN16:	CALL CALL JMP	SETDSK CLOSEFI MRETN	;CLOSE FILE	; 3B97 CD 24 38
FCN17:	CALL MVI C, CALL JMP	SETDSK \$00 LOOKUP MRETN	;SEARCH FOR FILENAME ;CHECK 13 CHARS	
FCN18:	LHLD SHLD CALL CALL JMP	STPFIM USERDE SETDSK STPFIL MRETN	;SEARCH FOR NEXT OCCURENCE OF NAME ;USERS FCB FROM PREV LOOKUP ;PROCESS ENTRY TYPE FOR DRIVE ;GET NEXT MATCHING FILE ;ANSWER IS IN ANSARL	
FCN19:	CALL CALL JMP	SETDSK DODEL MRETN	;DELETE FILE	; 3BBA CD E7 36
FCN20:	CALL	SETDSK	;READ NEXT RECORD	

```

CALL RDNXTR      ; 3BC3 CD F6 38
JMP  MRETN

FCN21: CALL SETDSK      ;WRITE NEXT RECORD
CALL  WTNXTR      ; 3BCC CD 4A 39
JMP  MRETN

FCN22: CALL SETDSK      ;MAKE FILE
CALL  MAKEFI      ; 3BD5 CD 42 38
JMP  MRETN

FCN23: CALL SETDSK      ;RENAME FILE
CALL  DOREN       ; 3BDE CD B9 37
JMP  MRETN

FCN24: LDA  LOGVEC      ;INTERROGATE LOGIN VECTOR
STA  ANSARL        ;WHICH IS JUST THESE BITS
JMP  MRETN

FCN25: LDA  CURDSK      ;GET CURRENT DRIVE NUMBER
STA  ANSARL        ;SAVE AS ANSWER
JMP  MRETN

FCN26: LDA  FRZDMA      ;SET DMA ADDR. ALLOWED TO?
RAR
JNC  FCN26A        ;IF NOT, SKIP THIS.
LHLD USERDE
SHLD USRDMA        ;REMEMBER WHAT USER WANTED
JMP  FCN26B      ; 3C03 C3 09 3C

FCN26A: CALL DEFDMA      ;MAKE IT SYSTEM STANDARD
FCN26B: JMP  MRETN

FCN27: LHLD CURALV      ;GET CURRENT DRIVE'S ALLOCATION VECTOR
SHLD ANSWER        ; 3C0F 22 0C 33
JMP  MRETN

FCN28: CALL MAKERO      ;UNDOCUMENTED. SET R/O BIT FOR LGD DISK.
JMP  MRETN

FCN29: LDA  ROVEC      ;UNDOCUMENTED. GET R/O VECTOR
STA  ANSARL
JMP  MRETN

;FREEZE DMA ADDR. DON'T LET USER CHANGE IT.

FCN30: LXI H, FRZDMA      ;UNDOCUMENTED
MVI H, $01          ; 3C27 30 01
CALL DEFDMA
JMP  MRETN        ;AND FREEZE IT AT SYS STANDARD

;DISPATCH TABLE FOR USER FUNCTIONS.

FCNDB: .ADDR FCN0,FCN1,FCN2,FCN3,FCN4,FCN5
      .ADDR FCN6,FCN7,FCN8,FCN9,FCN10,FCN11
      .ADDR FCN12,FCN13,FCN14,FCN15,FCN16,FCN17
      .ADDR FCN18,FCN19,FCN20,FCN21,FCN22,FCN23
      .ADDR FCN24,FCN25,FCN26,FCN27,FCN28,FCN29
      .ADDR FCN30

```

;COMMON RETURN TO USER

```

MRETN:  LDA    RELOGF
        CPI    0
        JZ     MRETN1
        LHLD   USERDE
        LDA    RELOGF
        MOV    M, A
        LDA    RELOGN
        STA    ANSARG
        CALL   LGNSDK
MRETN1: LDA    ANSARL
        LXI    D, ANSWER
        CALL   AORIDE
        XCHG
        DCX    H
        MOV    M, E
        INX    H
        MOV    M, D
        RET

```

;NEED TO RE-LOG A DISK?

;JUMP IF NOT

;YES. POINT TO FCB

;PUT ENTRY TYPE BYTE BACK IN USER FCB

;HAVE TO RE-LOG THIS DISK

;LOG FROM HERE

;GO DO IT

;ANSWER TO USER

; 0A IOR (DE) => HL

; 3C88 11 0C 33

; 3C8E EB

; 3C8F 2B

; 3C90 73

; 3C91 23

; 3C92 72

; 3C93 C9

```

XCHG
APLSKM: MOV    E, A
        MVI    D, 0
        XCHG
        LDAX   D
        ADD    L
        MOV    L, A
        INX    D
        LDAX   D
        ADC    H
        MOV    H, A
        RET

```

; 0A + (HL) => HL

; 3C94 EB

; 3C96 16 00

; 3C98 EB

; 3C99 1A

; 3C9A 85

; 3C9B 6F

; 3C9C 13

; 3C9D 1A

; 3C9E 8C

; 3C9F 67

; 3CA0 C9

```

ACRHL:  MOV    E, A
        MVI    D, 0
        MOV    A, E
        ORA    L
        MOV    L, A
        MOV    A, D
        ORA    H
        MOV    H, A
        RET

```

; 0A IOR HL => HL

; 3CA2 16 00

; 3CA4 7B

; 3CA5 B5

; 3CA6 6F

; 3CA7 7A

; 3CA8 B4

; 3CA9 67

; 3CAA C9

```

AORIDE: XCHG
        MOV    E, A
        MVI    D, 0
        XCHG
        LDAX   D
        ORA    L
        MOV    L, A
        INX    D
        LDAX   D
        ORA    H
        MOV    H, A
        RET

```

; 0A IOR (DE) => HL

; 3CAC 5F

; 3CAD 16 00

; 3CAF EB

; 3CB0 1A

; 3CB1 B5

; 3CB2 6F

; 3CB3 13

; 3CB4 1A

; 3CB5 B4

; 3CB6 67

; 3CB7 C9

;MULTIPLE SHIFT ROUTINE, DOING RLC'S

```

RLCNH:  MOV    A, M

```

; 3CBB 7E

RLCN:	RLC			, 3CB9	07		
	DCR	C		, 3CBA	0D		
	JNZ	RLCN		, 3CBB	C2	B9	3C
	RET			, 3CBE	C9		
RALNM:	MOV A,	M		, 3CBF	7E		
RALN:	RAL			, 3CC0	0F		
	DCR	C		, 3CC1	0D		
	JNZ	RALN		, 3CC2	C2	C6	3C
	RET			, 3CC5	C9		

;PICK UP 16 BITS AT (HL), THEN DO (C) DAD H'S ON THEM

```

SHLMM:  MOV E,  M           ;PICK UP A DOUBLE BYTE
        INX          H
        MOV D,  M
        XCHG                ;PUT THEM IN HL
SHLMML:  DAD          H      ;SHIFT THEM LEFT
        OCR          C
        JNZ         SHLMML   ;(C) TIMES, AT LEAST ONCE
        RET

```

```

RARNM:  MOV A,  M           ;PICK UP MEM ARG
RARNL:  ORA      A           ;CLEAR CARRY
        RAR                ;ROTATE IT RIGHT
        DCR      C           ;THIS MANY TIMES
        JNZ      RARNL
        RET

```

MOV L, C		; 3CD8	69
MOV H, B			; 3CD9 60
DEMIHL: MOV C, M		; 16 BIT SUBTRACT 2 MEMS	TO HL
INX H			; 3CDB 23
MOV B, M			; 3CDC 46
LDAX D		; (DE) - (HL) => HL	
SUB C			; 3CDE 91
MOV L, A			; 3CDF 6F
INX D			; 3CE0 13
LDAX D			; 3CE1 1A
SBB B			; 3CE2 98
MOV H, A			; 3CE3 67
RET			; 3CE4 C9

DEMI3A:	MOV L, A	; 0A => HL		
	MVI H, 0		; 3CE6	26 00
IDEMHL:	LDAX D	; (DE) - HL => HL		
	SUB L		; 3CE9	95
	MOV L, A		; 3CEA	0F
	INX D		; 3CEB	13
	LDAX D		; 3CEC	1A
	SBB H		; 3CED	9C
	MOV H, A		; 3CEE	07
	RET		; 3CEF	C9

7L
VARIABLES FOR BDUS PORTION

```

ANSARL:  *BYTE 0      ;RETURN USER'S VALUE HERE, FOR A REG
ANSARG:  *BYTE 0      ; 3CF1  00
CHROFS:  *BYTE 0      ; CHARACTER OFFSET OF FCB IN DIR BUFFER
DOSEFCB: *BYTE 0      ; 3CF3  00

```



```

DIRSEC: .BYTE 0
SYSDMA: .ADDR DEFBFR
USRDMA: .ADDR DEFBFR
FRZDMA: .BYTE 0
→ ALLOCS: .BLKB NDISKS * NALLOV
CURLV: .BLKW 1
ROVEC: .BYTE 0
CKSTAB: .BLKB $40
VECKS: .BLKW 1
RELOGN: .BYTE 0
RELOGF: .BYTE 0
LOGVEC: .BYTE 0
TKPTRS: .BLKB NDISKS
TSECOV: .BLKW NDISKS
TKPNTR: .BLKW 1
TKSEC0: .BLKW 1
CURRRC: .BYTE 0
CURNR: .BYTE 0
CURRMW: .BYTE 0,0

```

; 3CF4 00
 ;OUR COPY OF CURRENT DMA ADDR
 ;USER'S REQUESTED DMA ADDR
 ;ALL THE ALLOCATION VECTORS
 ;CURRENT DISK'S ALLOC VECTOR
 ;BITS FOR DRIVES THAT ARE READ-ONLY
 ;NDISKS * NDIRECTORY-SEGMENTS
 ;POINTER TO CHECKSUM TABLE
 ;MRETN MUST RE-LOG THIS DISK
 ;FLAG NEED TO RE-LOG AT MRETN
 ;BITS FOR CURRENTLY LOGGED IN DISKS
 ; B0=A, B1=B, B2=C, B3=D
 ;CURRENT TRACK, FOR EACH DISK
 ;FOR ALLOCATING TRACKS
 ;POINTER TO TRACK NUMBER FOR CURRENT OP
 ;POINTER INTO TRACKV, FOR ALLOC TRACK
 ;RECORD COUNT
 ;RECORD NUMBER
 ;CURRENT MAP INFO, W/ 8 HIGH ORDER 0'S

;BELOW HERE ARE LOCAL SUBR ARG STORAGE

```

CTKTMP: .BLKW 1
RVARG: .BYTE 0
RWCOMT: .BYTE 0
CKSTM1: .BYTE 0
CKSTM2: .BYTE 0
CRCURT: .BYTE 0
VERDIT: .BYTE 0
MAPNXV: .BYTE 0
GETBIT: .BYTE 0
BRKBTV: .BYTE 0
BRKBTH: .BYTE 0
ALDFCV: .BYTE 0
ALDFCW: .BYTE 0
ALDFCX: .BYTE 0
RDALCV: .BYTE 0
STFFIV: .BYTE 0
STFFIW: .BYTE 0,0
STPFIW: .BYTE 0
STPFIY: .BYTE 0
LOOKUV: .BYTE 0,0,0,0
ALO.LO: .BYTE 0
ALO.HI: .BYTE 0
CPYNAY: .BYTE 0
CPYNAM: .BYTE 0
CPNFIV: .BYTE 0
MAKFIV: .BYTE 0
MAKFIW: .BYTE 0,0
SIPXTV: .BYTE 0
WINXTV: .BYTE 0
WINXTW: .BYTE 0

```

;TEMP IN ALLOCATION COMPUTATION

;A LITTLE SPACE TO END OF PAGE
 .BYTE 0,0,0,0,0,0

;HERE STARTS THE CUSTOMIZED BIOS

!FIRST THE VECTOR FOR DEFINED ENTRIES

BIOS:	JMP	0	!COLD BOOT
CBWBUT:	JMP	0	!WARM BOOT
CBTTCK:	JMP	0	!CONSOLE STATUS CHECK
CBCTYI:	JMP	0	!CONSOLE INPUT
CBCTYO:	JMP	0	!CONSOLE OUTPUT
CBLPFO:	JMP	0	!LIST OUTPUT
CBPTPO:	JMP	0	!PUNCH OUTPUT
CBPTRI:	JMP	0	!READER INPUT
CBHOME:	JMP	0	!HOME THE SELECTED DISK
CBSELE:	JMP	0	!SELECT A DISK
CBSTTK:	JMP	0	!SET TRACK
CBSTSC:	JMP	0	!SET SECTOR
CBSTMA:	JMP	0	!SET DMA ADDRESS
CBREAD:	JMP	0	!READ A SECTOR
CBWRIT:	JMP	0	!WRITE A SECTOR

!REST NOT IN THIS SOURCE FILE

•END