```
                        TITLE   'Bdos Interface, Bdos, Version 2.2 Feb, 1980'
                ;***************************************************************
                ;***************************************************************
                ;**                                                           **
                ;**   B A S I C   D I S K   O P E R A T I N G   S Y S T E M   **
                ;**             I N T E R F A C E   M O D U L E               **
                ;**                                                           **
                ;***************************************************************
                ;***************************************************************
                ;
                ;       COPYRIGHT (C) 1978, 1979, 1980
                ;       DIGITAL RESEARCH
                ;       BOX 579, PACIFIC GROVE
                ;       CALIFORNIA
                ;
                ;
                ;      20 JANUARY 1980
                ;
                ;
FFFF =          ON      EQU     0FFFFH
0000 =          OFF     EQU     00000H
0000 =          TEST    EQU     OFF
                ;
                        IF      TEST
                        ORG     0DC00H
                        ELSE
0800                    ORG     0800H
                        ENDIF
                ;       BIOS VALUE DEFINED AT END OF MODULE
                ;
0018 =          SSIZE   EQU     24              ;24 LEVEL STACK
                ;
                ;       LOW MEMORY LOCATIONS
0000 =          REBOOT  EQU     0000H           ;REBOOT SYSTEM
0003 =          IOLOC   EQU     0003H           ;I/O BYTE LOCATION
0006 =          BDOSA   EQU     0006H           ;ADDRESS FIELD OF JMP BDOS
                ;
                ;       BIOS ACCESS CONSTANTS
1600 #          BOOTF   SET     BIOS+3*0        ;COLD BOOT FUNCTION
1603 #          WBOOTF  SET     BIOS+3*1        ;WARM BOOT FUNCTION
1606 #          CONSTF  SET     BIOS+3*2        ;CONSOLE STATUS FUNCTION
1609 #          CONINF  SET     BIOS+3*3        ;CONSOLE INPUT FUNCTION
160C #          CONOUTF SET     BIOS+3*4        ;CONSOLE OUTPUT FUNCTION
160F #          LISTF   SET     BIOS+3*5        ;LIST OUTPUT FUNCTION
1612 #          PUNCHF  SET     BIOS+3*6        ;PUNCH OUTPUT FUNCTION
1615 #          READERF SET     BIOS+3*7        ;READER INPUT FUNCTION
1618 #          HOMEF   SET     BIOS+3*8        ;DISK HOME FUNCTION
161B #          SELDSKF SET     BIOS+3*9        ;SELECT DISK FUNCTION
161E #          SETTRKF SET     BIOS+3*10       ;SET TRACK FUNCTION
1621 #          SETSECF SET     BIOS+3*11       ;SET SECTOR FUNCTION
1624 #          SETDMAF SET     BIOS+3*12       ;SET DMA FUNCTION
1627 #          READF   SET     BIOS+3*13       ;READ DISK FUNCTION
162A #          WRITEF  SET     BIOS+3*14       ;WRITE DISK FUNCTION
162D #          LISTSTF SET     BIOS+3*15       ;LIST STATUS FUNCTION
1630 #          SECTRAN SET     BIOS+3*16       ;SECTOR TRANSLATE
                ;
```

```
                ;       EQUATES FOR NON GRAPHIC CHARACTERS
0003 =          CTLC    EQU     03H     ;CONTROL C
0005 =          CTLE    EQU     05H     ;PHYSICAL EOL
0008 =          CTLH    EQU     08H     ;BACKSPACE
0010 =          CTLP    EQU     10H     ;PRNT TOGGLE
0012 =          CTLR    EQU     12H     ;REPEAT LINE
0013 =          CTLS    EQU     13H     ;STOP/START SCREEN
0015 =          CTLU    EQU     15H     ;LINE DELETE
0018 =          CTLX    EQU     18H     ;=CTL-U
001A =          CTLZ    EQU     1AH     ;END OF FILE
007F =          RUBOUT  EQU     7FH     ;CHAR DELETE
0009 =          TAB     EQU     09H     ;TAB CHAR
000D =          CR      EQU     0DH     ;CARRIAGE RETURN
000A =          LF      EQU     0AH     ;LINE FEED
005E =          CTL     EQU     5EH     ;UP ARROW
                ;
0800 0000000000         DB      0,0,0,0,0,0
                ;
                ;       ENTER HERE FROM THE USER'S PROGRAM WITH FUNCTION NUMBER IN C,
                ;       AND INFORMATION ADDRESS IN D,E
0806 C31108             JMP     BDOSE   ;PAST PARAMETER BLOCK
                ;
                ;       ************************************************
                ;       *** RELATIVE LOCATIONS 0009 - 000E           ***
                ;       ************************************************
0809 9908       PERERR: DW      PERSUB  ;PERMANENT ERROR SUBROUTINE
080B A508       SELERR: DW      SELSUB  ;SELECT ERROR SUBROUTINE
080D AB08       RODERR: DW      RODSUB  ;RO DISK ERROR SUBROUTINE
080F B108       ROFERR: DW      ROFSUB  ;RO FILE ERROR SUBROUTINE
                ;
                ;
                BDOSE:  ;ARRIVE HERE FROM USER PROGRAMS
0811 EB22430BEB         XCHG! SHLD INFO! XCHG ;INFO=DE, DE=INFO
0816 7B32D615           MOV A,E! STA LINFO ;LINFO = LOW(INFO) - DON'T EQU
081A 2100002245         LXI H,0! SHLD ARET ;RETURN VALUE DEFAULTS TO 0000
                        ;SAVE USER'S STACK POINTER, SET TO LOCAL STACK
0820 39220F0B           DAD SP! SHLD ENTSP ;ENTSP = STACKPTR
0824 31410B             LXI SP,LSTACK ;LOCAL STACK SETUP
0827 AF32E01532         XRA A! STA FCBDSK! STA RESEL ;FCBDSK,RESEL=FALSE
082E 217415             LXI H,GOBACK ;RETURN HERE AFTER ALL FUNCTIONS
0831 E5                 PUSH H ;JMP GOBACK EQUIVALENT TO RET
0832 79FE29D0           MOV A,C! CPI NFUNCS! RNC ;SKIP IF INVALID #
0836 4B                 MOV C,E ;POSSIBLE OUTPUT CHARACTER TO C
0837 2147085F16         LXI H,FUNCTAB! MOV E,A! MVI D,0 ;DE=FUNC, HL=.CIOTAB
083D 19195E2356         DAD D! DAD D! MOV E,M! INX H! MOV D,M ;DE=FUNCTAB(FUNC)
0842 2A430B                     LHLD INFO       ;INFO IN DE FOR LATER XCHG
0845 EB[C]9             XCHG! PCHL ;DISPATCHED
                ;
                ;       DISPATCH TABLE FOR FUNCTIONS
                FUNCTAB:
0847 0316C80A90         DW      WBOOTF, FUNC1, FUNC2, FUNC3
084F 12160F16D4         DW      PUNCHF, LISTF, FUNC6, FUNC7
0857 F30AF80AE1         DW      FUNC8, FUNC9, FUNC10,FUNC11
000C =          DISKF   EQU     ($-FUNCTAB)/2   ;DISK FUNCS
085F 7E14831445         DW      FUNC12,FUNC13,FUNC14,FUNC15
0867 A514AB14C8         DW      FUNC16,FUNC17,FUNC18,FUNC19
```

```
086F E014E614EC          DW      FUNC20,FUNC21,FUNC22,FUNC23
0877 FE1404150A          DW      FUNC24,FUNC25,FUNC26,FUNC27
087F 2C0D17151D          DW      FUNC28,FUNC29,FUNC30,FUNC31
0887 2D15411547          DW      FUNC32,FUNC33,FUNC34,FUNC35
088F 0E14531504          DW      FUNC36,FUNC37,FUNC38,FUNC39             ;
0897 9B15                DW      FUNC40                                  ;
0029 =           NFUNCS  EQU     ($-FUNCTAB)/2
                 ;
                 ;
                 ;       ERROR SUBROUTINES
                 PERSUB: ;REPORT PERMANENT ERROR
0899 21CA08CDE5          LXI H,PERMSG! CALL ERRFLG ;TO REPORT THE ERROR
089F FE03CA0000          CPI CTLC! JZ REBOOT ;REBOOT IF RESPONSE IS CTLC
08A4 C9                  RET ;AND IGNORE THE ERROR
                 ;
                 SELSUB: ;REPORT SELECT ERROR
08A5 21D508C3B4          LXI H SELMSG! JMP WAIT$ERR ;WAIT CONSOLE BEFORE BOOT
                 ;
                 RODSUB: ;REPORT WRITE TO READ/ONLY DISK
08AB 21E108C3B4          LXI H,RODMSG! JMP WAIT$ERR ;WAIT CONSOLE
                 ;
                 ROFSUB: ;REPORT READ/ONLY FILE
08B1 21DC08              LXI H,ROFMSG ;DROP THROUGH TO WAIT FOR CONSOLE
                 ;
                 WAIT$ERR:
                         ;WAIT FOR RESPONSE BEFORE BOOT
08B4 CDE508C300          CALL ERRFLG! JMP REBOOT
                 ;
                 ;       ERROR MESSAGES
08BA 42646F7320DSKMSG: DB       'Bdos Err On '
08C6 203A2024    DSKERR: DB      ' : $'  ;FILLED IN BY ERRFLG
08CA 4261642053PERMSG: DB       'Bad Sector$'
08D5 53656C6563SELMSG: DB       'Select$'
08DC 46696C6520ROFMSG: DB       'File '
08E1 522F4F24    RODMSG: DB      'R/O$'
                 ;
                 ;
                 ERRFLG:
                         ;REPORT ERROR TO CONSOLE, MESSAGE ADDRESS IN HL
08E5 E5CDC909            PUSH H! CALL CRLF ;STACK MSSG ADDRESS, NEW LINE
08E9 3A420BC641          LDA CURDSK! ADI 'A'! STA DSKERR ;CURRENT DISK NAME
08F1 01BA08CDD3          LXI B,DSKMSG! CALL PRINT ;THE ERROR MESSAGE
08F7 C1CDD309            POP B! CALL PRINT ;ERROR MSSAGE TAIL
                         ;JMP CONIN ;TO GET THE INPUT CHARACTER
                         ;(DROP THROUGH TO CONIN)
                         ;RET
                 ;
                 ;
                 ;       CONSOLE HANDLERS
                 CONIN:
                         ;READ CONSOLE CHARACTER TO A
08FB 210E0B7E36          LXI H,KBCHAR! MOV A,M! MVI M,0! ORA A! RNZ
                         ;NO PREVIOUS KEYBOARD CHARACTER READY
0903 C30916              JMP CONINF ;GET CHARACTER EXTERNALLY
                         ;RET
                 ;
```

```
                CONECH:
                        ;READ CHARACTER WITH ECHO
0906 CDFB08CD14         CALL CONIN! CALL ECHOC! RC ;ECHO CHARACTER?
                        ;CHARACTER MUST BE ECHOED BEFORE RETURN
090D F54FCD9009         PUSH PSW! MOV C,A! CALL TABOUT! POP PSW
0913 C9                 RET ;WITH CHARACTER IN A
                ;
                ECHOC:
                        ;ECHO CHARACTER IF GRAPHIC
                        ;CR, LF, TAB, OR BACKSPACE
0914 FE0DC8             CPI CR! RZ ;CARRIAGE RETURN?
0917 FE0AC8             CPI LF! RZ ;LINE FEED?
091A FE09C8             CPI TAB! RZ ;TAB?
091D FE08C8             CPI CTLH! RZ ;BACKSPACE?
0920 FE20C9             CPI ' '! RET ;CARRY SET IF NOT GRAPHIC
                ;
                CONBRK: ;CHECK FOR CHARACTER READY
0923 3A0E0BB7C2         LDA KBCHAR! ORA A! JNZ CONB1 ;SKIP IF ACTIVE KBCHAR
                                ;NO ACTIVE KBCHAR, CHECK EXTERNAL BREAK
092A CD0616E601                 CALL CONSTF! ANI 1! RZ ;RETURN IF NO CHAR READY
                                ;CHARACTER READY, READ IT
0930 CD0916                     CALL CONINF ;TO A
0933 FE13C24209                 CPI CTLS! JNZ CONBO ;CHECK STOP SCREEN FUNCTION
                                ;FOUND CTLS, READ NEXT CHARACTER
0938 CD0916                     CALL CONINF ;TO A
093B FE03CA0000                 CPI CTLC! JZ REBOOT ;CTLC IMPLIES RE-BOOT
                                ;NOT A REBOOT, ACT AS IF NOTHING HAS HAPPENED
0940 AFC9                       XRA A! RET ;WITH ZERO IN ACCUMULATOR
                        CONBO:
                                ;CHARACTER IN ACCUM, SAVE IT
0942 320E0B                     STA KBCHAR
                        CONB1:
                                ;RETURN WITH TRUE SET IN ACCUMULATOR
0945 3E01C9                     MVI A,1! RET
                ;
                CONOUT:
                        ;COMPUTE CHARACTER POSITION/WRITE CONSOLE CHAR FROM C
                        ;COMPCOL = TRUE IF COMPUTING COLUMN POSITION
0948 3A0A0BB7C2         LDA COMPCOL! ORA A! JNZ COMPOUT
                                ;WRITE THE CHARACTER, THEN COMPUTE THE COLUMN
                                ;WRITE CONSOLE CHARACTER FROM C
094F C5CD2309                   PUSH B! CALL CONBRK ;CHECK FOR SCREEN STOP FUNCTION
0953 C1C5                       POP B! PUSH B ;RECALL/SAVE CHARACTER
0955 CD0C16                     CALL CONOUTF ;EXTERNALLY, TO CONSOLE
0958 C1C5                       POP B! PUSH B ;RECALL/SAVE CHARACTER
                                ;MAY BE COPYING TO THE LIST DEVICE
095A 3A0D0BB7C4                 LDA LISTCP! ORA A! CNZ LISTF ;TO PRINTER, IF SO
0961 C1                         POP B ;RECALL THE CHARACTER
                        COMPOUT:
0962 79                         MOV A,C ;RECALL THE CHARACTER
                                ;AND COMPUTE COLUMN POSITION
0963 210C0B                     LXI H,COLUMN ;A = CHAR, HL = .COLUMN
0966 FE7FC8                     CPI RUBOUT! RZ ;NO COLUMN CHANGE IF NULLS
0969 34                         INR M ;COLUMN = COLUMN + 1
096A FE20D0                     CPI ' '! RNC ;RETURN IF GRAPHIC
                                ;NOT GRAPHIC, RESET COLUMN POSITION
```

```
096D 35                         DCR M ;COLUMN = COLUMN - 1
096E 7EB7C8                     MOV A,M! ORA A! RZ ;RETURN IF AT ZERO
                                ;NOT AT ZERO, MAY BE BACKSPACE OR END LINE
0971 79                         MOV A,C ;CHARACTER BACK TO A
0972 FE08C27909                 CPI CTLH! JNZ NOTBACKSP
                                        ;BACKSPACE CHARACTER
0977 35                                 DCR M ;COLUMN = COLUMN - 1
0978 C9                                 RET
                                NOTBACKSP:
                                        ;NOT A BACKSPACE CHARACTER, EOL?
0979 FE0AC0                             CPI LF! RNZ ;RETURN IF NOT
                                        ;END OF LINE, COLUMN = 0
097C 3600                               MVI M,0 ;COLUMN = 0
097E C9                         RET
                ;
                CTLOUT:
                        ;SEND C CHARACTER WITH POSSIBLE PRECEDING UP-ARROW
097F 79CD1409           MOV A,C! CALL ECHOC ;CY IF NOT GRAPHIC (OR SPECIAL CASE)
0983 D29009             JNC TABOUT ;SKIP IF GRAPHIC, TAB, CR, LF, OR CTLH
                                ;SEND PRECEDING UP ARROW
0986 F50E5ECD48                 PUSH PSW! MVI C,CTL! CALL CONOUT ;UP ARROW
098C F1F640                     POP PSW! ORI 40H ;BECOMES GRAPHIC LETTER
098F 4F                         MOV C,A ;READY TO PRINT
                                ;(DROP THROUGH TO TABOUT)
                ;
                TABOUT:
                        ;EXPAND TABS TO CONSOLE
0990 79FE09C248         MOV A,C! CPI TAB! JNZ CONOUT ;DIRECT TO CONOUT IF NOT
                                ;TAB ENCOUNTERED, MOVE TO NEXT TAB POSITION
                        TABO:
0996 0E20CD4809                 MVI C,' '! CALL CONOUT ;ANOTHER BLANK
099B 3A0C0BE607                 LDA COLUMN! ANI 111B ;COLUMN MOD 8 = 0 ?
09A0 C29609                     JNZ TABO ;BACK FOR ANOTHER IF NOT
09A3 C9                 RET
                ;
                ;
                BACKUP:
                        ;BACK-UP ONE SCREEN POSITION
09A4 CDAC090E20         CALL PCTLH! MVI C,' '! CALL CONOUTF
                ;       (DROP THROUGH TO PCTLH)                         ;
                PCTLH:
                        ;SEND CTLH TO CONSOLE WITHOUT AFFECTING COLUMN COUNT
09AC 0E08C30C16         MVI C,CTLH! JMP CONOUTF
                        ;RET
                ;
                CRLFP:
                        ;PRINT #, CR, LF FOR CTLX, CTLU, CTLR FUNCTIONS
                        ;THEN MOVE TO STRTCOL (STARTING COLUMN)
09B1 0E23CD4809         MVI C,'#'! CALL CONOUT
09B6 CDC909             CALL CRLF
                        ;COLUMN = 0, MOVE TO POSITION STRTCOL
                        CRLFPO:
09B9 3A0C0B210B                 LDA COLUMN! LXI H,STRTCOL
09BF BED0                       CMP M! RNC ;STOP WHEN COLUMN REACHES STRTCOL
09C1 0E20CD4809                 MVI C,' '! CALL CONOUT ;PRINT BLANK
09C6 C3B909                     JMP CRLFPO
```

```
                ;;
                ;
                CRLF:
                        ;CARRIAGE RETURN LINE FEED SEQUENCE
09C9 0E0DCD4809         MVI C,CR! CALL CONOUT! MVI C,LF! JMP CONOUT
                        ;RET
                ;
                PRINT:
                        ;PRINT MESSAGE UNTIL M(BC) = '$'
09D3 0AFE24C8           LDAX B! CPI '$'! RZ ;STOP ON $
                                ;MORE TO PRI T
09D7 03C54F                     INX B! PUSH B! MOV C,A ;CHAR TO C
09DA CD9009                     CALL TABOUT ;ANOTHER CHARACTER PRINTED
09DD C1C3D309                   POP B! JMP PRINT
                ;
                READ:   ;READ TO INFO ADDRESS (MAX LENGTH, CURRENT LENGTH, BUFFER)
09E1 3A0C0B320B         LDA COLUMN! STA STRT OL ;SAVE STA T FOR CTL-X, CTL-H
09E7 2A430B             LHLD INFO
09EA 4E23E50600         MOV C,M! INX H! PUSH H! MVI B,0
                        ;B = CURRENT BUFFER LENGTH,
                        ;C = MAXIMUM BUFFER LENGTH,
                        ;HL= NEXT TO FILL - 1
                        READNX:
                                ;READ NEXT CHARACTER, BC, HL ACTIVE
09EF C5E5                       PUSH B! PUSH H ;BLEN, CMAX, HL SAVED
                                READNO:
09F1 CDFB08                             CALL CONIN ;NEXT CHAR IN A
09F4 E67F                               ANI 7FH ;MASK PARITY BIT
09F6 E1C1                               POP H! POP B ;REACTIVATE COUNTERS
09F8 FE0DCAC10A                         CPI CR! JZ READEN ;END OF LINE?
09FD FE0ACAC10A                         CPI LF! JZ READEN ;ALSO END OF LINE
0A02 FE08C2160A                         CPI CTLH! JNZ NOTH ;BACKSPACE?
                                        ;DO WE HAVE ANY CHARACTERS TO BACK OVER?
0A07 78B7CAEF09                         MOV A,B! ORA A! JZ READNX
                                        ;CHARACTERS REMAIN IN BUFFER, BACKUP ONE
0A0C 05                                 DCR B ;REMOVE ONE CHARACTER
0A0D 3A0C0B320A                         LDA COLUMN! STA COMPCOL ;COL > 0
                                        ;COMPCOL > 0 MARKS REPEAT AS LENGTH COMPUTE
0A13 C3700A                             JMP LINELEN ;USES SAME CODE AS REPEAT
                                NOTH:
                                        ;NOT A BACKSPACE
0A16 FE7FC2260A                         CPI RUBOUT! JNZ NOTRUB ;RUBOUT CHAR?
                                        ;RUBOUT ENCOUNTERED, RUBOUT IF POSSIBLE
0A1B 78B7CAEF09                         MOV A,B! ORA A! JZ READNX ;SKIP IF LEN=0
                                        ;BUFFER HAS CHARACTERS, RESEND LAST CHAR
0A20 7E052B                             MOV A,M! DCR B! DCX H ;A = LAST CHAR
                                        ;BLEN=BLEN-1, NEXT TO FILL - 1 DECREMENTED
0A23 C3A90A                             JMP RDECH1 ;ACT LIKE THIS IS AN ECHO
                        ;
                                NOTRUB:
                                        ;NOT A RUBOUT CHARACTER, CHECK END LINE
0A26 FE05C2370A                         CPI CTLE! JNZ NOTE ;PHYSICAL END LINE?
                                        ;YES, SAVE ACTIVE COUNTERS AND FORCE EOL
0A2B C5E5CDC909                         PUSH B! PUSH H! CALL CRLF
0A30 AF320B0B                           XRA A! STA STRTCOL ;START POSITION = 00
0A34 C3F109                             JMP READNO ;FOR ANOTHER CHARACTER
```

```
                        NOTE:
                                ;NOT END OF LINE, LIST TOGGLE?
0A37 FE10C2480A                 CPI CTLP! JNZ NOTP ;SKIP IF NOT CTLP
                                ;LIST TOGGLE - CHANGE PARITY
0A3C E5                         PUSH H ;SAVE NEXT TO FILL - 1
0A3D 210D0B                     LXI H,LISTCP ;HL=.LISTCP FLAG
0A40 3E0196                     MVI A,1! SUB M ;TRUE-LISTCP
0A43 77                         MOV M,A ;LISTCP = NOT LISTCP
0A44 E1C3EF09                   POP H! JMP READNX ;FOR ANOTHER CHAR
                        NOTP:
                                ;NOT A CTLP, LINE DELETE?
0A48 FE18C25F0A                 CPI CTLX! JNZ NOTX
0A4D E1                         POP H ;DISCARD START POSITION
                                ;LOOP WHILE COLUMN > STRTCOL
                                BACKX:
0A4E 3A0B0B210C                         LDA STRTCOL! LXI H,COLUMN
0A54 BED2E109                           CMP M! JNC READ ;START AGAIN
0A58 35                                 DCR M ;COLUMN = COLUMN - 1
0A59 CDA409                             CALL BACKUP ;ONE POSITION
0A5C C34E0A                             JMP BACKX
                        NOTX:
                                ;NOT A CONTROL X, CONTROL U?
                                ;NOT CONTROL-X, CONTROL-U?
0A5F FE15C26B0A                 CPI CTLU! JNZ NOTU ;SKIP IF NOT
                                ;DELETE LINE (CTLU)
0A64 CDB109                     CALL CRLFP ;PHYSICAL EOL
0A67 E1                         POP H ;DISCARD STARTING POSITION
0A68 C3E109                     JMP READ ;TO START ALL OVER
                        NOTU:
                                ;NOT LINE DELETE, REPEAT LINE?
0A6B FE12C2A60A                 CPI CTLR! JNZ NOTR
                        LINELEN:
                                ;REPEAT LINE, OR COMPUTE LINE LEN (CTLH)
                                ;IF COMPCOL > 0
0A70 C5CDB109                   PUSH B! CALL CRLFP ;SAVE LINE LENGTH
0A74 C1E1E5C5                   POP B! POP H! PUSH H! PUSH B
                                ;BCUR, CMAX ACTIVE, BEGINNING BUFF AT HL
                        REP0:
0A78 78B7CA8A0A                 MOV A,B! ORA A! JZ REP1 ;COUNT LEN TO 00
0A7D 234E                       INX H! MOV C,M ;NEXT TO PRINT
0A7F 05C5E5                     DCR B! PUSH B! PUSH H ;COUNT LENGTH DOWN
0A82 CD7F09                     CALL CTLOUT ;CHARACTER ECHOED
0A85 E1C1                       POP H! POP B ;RECALL REMAINING COUNT
0A87 C3780A                     JMP REP0 ;FOR THE NEXT CHARACTER
                        REP1:
                                ;END OF REPEAT, RECALL LENGTHS
                                ;ORIGINAL BC STILL REMAINS PUSHED
0A8A E5                         PUSH H ;SAVE NEXT TO FILL
0A8B 3A0A0BB7                   LDA COMPCOL! ORA A ;>0 IF COMPUTING LENGTH
0A8F CAF109                     JZ READNO ;FOR ANOTHER CHAR IF SO
                                ;COLUMN POSITION COMPUTED FOR CTLH
0A92 210C0B96                   LXI H,COLUMN! SUB M ;DIFF > 0
0A96 320A0B                     STA COMPCOL ;COUNT DOWN BELOW
                                ;MOVE BACK COMPCOL-COLUMN SPACES
                        BACKSP:
                                ;MOVE BACK ONE MORE SPACE
```

```
0A99 CDA409                             CALL BACKUP ;ONE SPACE
0A9C 210A0B35                           LXI H,COMPCOL! DCR M
0AA0 C2990A                             JNZ BACKSP
0AA3 C3F109                             JMP READNO ;FOR NEXT CHARACTER
                                NOTR:
                                        ;NOT A CTLR, PLACE INTO BUFFER
                                RDECHO:
0AA6 2377                               INX H! MOV M,A ;CHARACTER FILLED TO MEM
0AA8 04                                 INR B ;BLEN = BLEN + 1
                                RDECH1:
                                        ;LOOK FOR A RANDOM CONTROL CHARACTER
0AA9 C5E5                               PUSH B! PUSH H ;ACTIVE VALUES SAVED
0AAB 4F                                 MOV C,A ;READY TO PRINT
0AAC CD7F09                             CALL CTLOUT ;MAY BE UP-ARROW C
0AAF E1C17E                             POP H! POP B! MOV A,M ;RECALL CHAR
0AB2 FE03                               CPI CTLC ;SET FLAGS FOR REBOOT TEST
0AB4 78                                 MOV A,B ;MOVE LENGTH TO A
0AB5 C2BD0A                             JNZ NOTC ;SKIP IF NOT A CONTROL C
0AB8 FE01                               CPI 1 ;CONTROL C, MUST BE LENGTH 1
0ABA CA0000                             JZ REBOOT ;REBOOT IF BLEN = 1
                                        ;LENGTH NOT ONE, SO SKIP REBOOT
                                NOTC:
                                        ;NOT REBOOT, ARE WE AT END OF BUFFER?
0ABD B9DAEF09                           CMP C! JC READNX ;GO FOR ANOTHER IF NOT
                                READEN:
                                        ;END OF READ OPERATION, STORE BLEN
0AC1 E170                               POP H! MOV M,B ;M(CURRENT LEN) = B
0AC3 0E0DC34809                         MVI C,CR! JMP CONOUT ;RETURN CARRIAGE
                                        ;RET
                FUNC1:
                        ;RETURN CONSOLE CHARACTER WITH ECHO
0AC8 CD0609             CALL CONECH
0ACB C3010B             JMP STA$RET
                ;
0990 =          FUNC2:  EQU     TABOUT
                        ;WRITE CONSOLE CHARACTER WITH TAB EXPANSION
                ;
                FUNC3:
                        ;RETURN READER CHARACTER
0ACE CD1516             CALL READERF
0AD1 C3010B             JMP STA$RET
                ;
                ;FUNC4: EQUATED TO PUNCHF
                        ;WRITE PUNCH CHARACTER
                ;
                ;FUNC5: EQUATED TO LISTF
                        ;WRITE LIST CHARACTER
                        ;WRITE TO LIST DEVICE
                ;
                FUNC6:
                        ;DIRECT CONSOLE I/O - READ IF OFFH
0AD4 793CCAE00A         MOV A,C! INR A! JZ DIRINP ;0FFH => 00H, MEANS INPUT MODE
0AD9 3CCA0616                   INR A! JZ CONSTF        ;0FEH IN C FOR STATUS
                                ;DIRECT OUTPUT FUNCTION
0ADD C30C16                     JMP CONOUTF                     ;
                        DIRINP:
```

```
0AE0 CD0616                     CALL CONSTF ;STATUS CHECK
0AE3 B7CA9115                   ORA A! JZ RETMON ;SKIP. RETURN 00 IF NOT READY
                                ;CHARACTER IS READY, GET IT
0AE7 CD0916                     CALL CONINF ;TO A
0AEA C3010B                     JMP STA$RET
                    ;
                    FUNC7:
                            ;RETURN IO BYTE
0AED 3A0300                 LDA IOLOC
0AF0 C3010B                 JMP STA$RET
                    ;
                    FUNC8:
                            ;SET I/O BYTE
0AF3 210300                 LXI H,IOLOC
0AF6 71                     MOV M,C
0AF7 C9                     RET ;JMP GOBACK
                    ;
                    FUNC9:
                            ;WRITE LINE UNTIL $ ENCOUNTERED
0AF8 EB                     XCHG    ;WAS LHLD INFO
0AF9 4D44                   MOV C,L! MOV B,H ;BC=STRING ADDRESS
0AFB C3D309                 JMP PRINT ;OUT TO CONSOLE
                    ;
09E1 =              FUNC10: EQU     READ
                            ;READ A BUFFERED CONSOLE LINE
                    ;
                    FUNC11:
                            ;CHECK CONSOLE STATUS
0AFE CD2309                 CALL CONBRK
                            ;(DROP THROUGH TO STA$RET)
                    STA$RET:
                            ;STORE THE A REGISTER TO ARET
0B01 32450B                 STA ARET
                    FUNC$RET:
0B04 C9                     RET ;JMP GOBACK (POP STACK FOR NON CP/M FUNCTIONS)
                    ;
                    SETLRET1:
                            ;SET LRET = 1
0B05 3E01C3010B             MVI A,1! JMP STA$RET                    ;
                    ;
                    ;
                    ;
                    ;       DATA AREAS
                    ;
0B0A 00             COMPCOL:DB      0       ;TRUE IF COMPUTING COLUMN POSITION
0B0B 00             STRTCOL:DB      0       ;STARTING COLUMN POSITION AFTER READ
0B0C 00             COLUMN: DB      0       ;COLUMN POSITION
0B0D 00             LISTCP: DB      0       ;LISTING TOGGLE
0B0E 00             KBCHAR: DB      0       ;INITIAL KEY CHAR = 00
0B0F                ENTSP:  DS      2       ;ENTRY STACK POINTER
0B11                        DS      SSIZE*2 ;STACK SIZE
                    LSTACK:
                    ;       END OF BASIC I/O SYSTEM
                    ;
                    ;*****************************************************************
                    ;*****************************************************************
```

```
                ;
                ;       COMMON VALUES SHARED BETWEEN BDOSI AND BDOS
0B41 00         USRCODE:DB      0       ;CURRENT USER NUMBER
0B42 00         CURDSK: DB      0       ;CURRENT DISK NUMBER
0B43            INFO:   DS      2       ;INFORMATION ADDRESS
0B45            ARET:   DS      2       ;ADDRESS VALUE TO RETURN
0B45 =          LRET    EQU     ARET    ;LOW(ARET)
                ;
                ;***************************************************************
                ;***************************************************************
                ;**                                                           **
                ;**   B A S I C   D I S K   O P E R A T I N G   S Y S T E M   **
                ;**                                                           **
                ;***************************************************************
                ;***************************************************************
                ;
0022 =          DVERS   EQU     22H     ;VERSION 2.2
                ;       MODULE ADDRESSES
                ;
                ;       LITERAL CONSTANTS
00FF =          TRUE    EQU     0FFH    ;CONSTANT TRUE
0000 =          FALSE   EQU     000H    ;CONSTANT FALSE
FFFF =          ENDDIR  EQU     0FFFFH  ;END OF DIRECTORY
0001 =          BYTE    EQU     1       ;NUMBER OF BYTES FOR "BYTE" TYPE
0002 =          WORD    EQU     2       ;NUMBER OF BYTES FOR "WORD" TYPE
                ;
                ;       FIXED ADDRESSES IN LOW MEMORY
005C =          TFCB    EQU     005CH   ;DEFAULT FCB LOCATION
0080 =          TBUFF   EQU     0080H   ;DEFAULT BUFFER LOCATION
                ;
                ;       FIXED ADDRESSES REFERENCED IN BIOS MODULE ARE
                ;       PERERR (0009), SELERR (000C), RODERR (000F)
                ;
                ;       ERROR MESSAGE HANDLERS
                ;
                ;PER$ERROR:
                        ;REPORT PERMANENT ERROR TO USER
                ;       LXI H,PERERR  JMP GOERR
                ;
                ;ROD$ERROR:                                             ;
                        ;REPORT READ/ONLY DISK ERROR
                ;       LXI H,RODERR  JMP GOERR                         ;
                ;
                ;ROF$ERROR:                                             ;
                        ;REPORT READ/ONLY FILE ERROR                    ;
                ;       LXI H,ROFERR    ;JMP GOERR
                ;
                SEL$ERROR:
                        ;REPORT SELECT ERROR
0B47 210B08             LXI H,SELERR
                ;
                ;
                GOERR:
                        ;HL = .ERRORHANDLER, CALL SUBROUTINE
0B4A 5E2356             MOV E,M! INX H! MOV D,M ;ADDRESS OF ROUTINE IN DE
0B4D EBE9               XCHG! PCHL ;TO SUBROUTINE
```

```
                ;
                ;
                ;
                ;       LOCAL SUBROUTINES FOR BIOS INTERFACE
                ;
                MOVE:
                        ;MOVE DATA LENGTH OF LENGTH C FROM SOURCE DE TO
                        ;DESTINATION GIVEN BY HL
0B4F 0C                 INR C ;IN CASE IT IS ZERO
                        MOVE0:
0B50 0DC8                       DCR C! RZ ;MORE TO MOVE
0B52 1A77                       LDAX D! MOV M,A ;ONE BYTE MOVED
0B54 1323                       INX D! INX H ;TO NEXT BYTE
0B56 C3500B                     JMP MOVE0
                ;
                SELECTDISK:
                        ;SELECT THE DISK DRIVE GIVEN BY CURDSK, AND FILL
                        ;THE BASE ADDRESSES CURTRKA - ALLOCA, THEN FILL
                        ;THE VALUES OF THE DISK PARAMETER BLOCK
0B59 3A420B4F           LDA CURDSK! MOV C,A ;CURRENT DISK# TO C
                        ;LSB OF E = 0 IF NOT YET LOGGED - IN
0B5D CD1B16             CALL SELDSKF ;HL FILLED BY CALL
                        ;HL = 0000 IF ERROR, OTHERWISE DISK HEADERS
0B60 7CB5C8             MOV A,H! ORA L! RZ ;RETURN WITH 0000 IN HL AND Z FLAG
                                ;DISK HEADER BLOCK ADDRESS IN HL
0B63 5E235623                   MOV E,M! INX H! MOV D,M! INX H ;DE=.TRAN
0B67 22B3152323                 SHLD CDRMAXA! INX H! INX H ;.CDRMAX
0B6C 22B5152323                 SHLD CURTRKA! INX H! INX H ;HL=.CURREC
0B71 22B7152323                 SHLD CURRECA! INX H! INX H ;HL=.BUFFA
                                ;DE STILL CONTAINS .TRAN
0B76 EB22D015                   XCHG! SHLD TRANV ;.TRAN VECTOR
0B7A 21B915                     LXI H,BUFFA ;DE= SOURCE FOR MOVE, HL=DEST
0B7D 0E08CD4F0B                 MVI C,ADDLIST! CALL MOVE ;ADDLIST FILLED
                                ;NOW FILL THE DISK PARAMETER BLOCK
0B82 2ABB15EB                   LHLD DPBADDR! XCHG ;DE IS SOURCE
0B86 21C115                     LXI H,SECTPT ;HL IS DESTINATION
0B89 0E0FCD4F0B                 MVI C,DPBLIST! CALL MOVE ;DATA FILLED
                                ;NOW SET SINGLE/DOUBLE MAP MODE
0B8E 2AC615                     LHLD MAXALL ;LARGEST ALLOCATION NUMBER
0B91 7C                         MOV A,H ;00 INDICATES < 255
0B92 21DD1536FF                 LXI H,SINGLE! MVI M,TRUE ;ASSUME A=00
0B97 B7CA9D0B                   ORA A! JZ RETSELECT
                                ;HIGH ORDER OF MAXALL NOT ZERO, USE DOUBLE DM
0B9B 3600                       MVI M,FALSE
                        RETSELECT:
0B9D 3EFFB7C9           MVI A,TRUE! ORA A! RET ;SELECT DISK FUNCTION OK
                ;
                HOME:
                        ;MOVE TO HOME POSITION, THEN OFFSET TO START OF DIR
0BA1 CD1816             CALL HOMEF ;MOVE TO TRACK 00, SECTOR 00 REFERENCE
                        ;LXI H,OFFSET ;MOV C,M ;INX H ;MOV B,M ;CALL SETTRKF    ;
                        ;FIRST DIRECTORY POSITION SELECTED
0BA4 AF                 XRA A ;CONSTANT ZERO TO ACCUMULATOR
0BA5 2AB5157723         LHLD CURTRKA! MOV M,A! INX H! MOV M,A ;CURTRK=0000
0BAB 2AB7157723         LHLD CURRECA! MOV M,A! INX H! MOV M,A ;CURREC=0000
                        ;CURTRK, CURREC BOTH SET TO 0000
```

```
0BB1 C9                 RET
                ;
                RDBUFF:
                        ;READ BUFFER AND CHECK CONDITION
0BB2 CD2716             CALL READF ;CURRENT DRIVE, TRACK, SECTOR, DMA
0BB5 C3BB0B             JMP DIOCOMP     ;CHECK FOR I/O ERRORS
                ;
                WRBUFF:
                        ;WRITE BUFFER AND CHECK CONDITION
                        ;WRITE TYPE (WRTYPE) IS IN REGISTER C
                        ;WRTYPE = 0 => NORMAL WRITE OPERATION
                        ;WRTYPE = 1 => DIRECTORY WRITE OPERATION
                        ;WRTYPE = 2 => START OF NEW BLOCK
0BB8 CD2A16             CALL WRITEF ;CURRENT DRIVE, TRACK, SECTOR, DMA
                DIOCOMP:        ;CHECK FOR DISK ERRORS
0BBB B7C8               ORA A! RZ                       ;
0BBD 210908             LXI     H,PERERR                        ;
0BC0 C34A0B             JMP GOERR
                ;
                SEEKDIR:
                        ;SEEK THE RECORD CONTAINING THE CURRENT DIR ENTRY
0BC3 2AEA15             LHLD DCNT ;DIRECTORY COUNTER TO HL
0BC6 0E02CDEA0C         MVI C,DSKSHF! CALL HLROTR ;VALUE TO HL
0BCB 22E51522EC         SHLD ARECORD! SHLD DREC ;READY FOR SEEK
                        ;  JMP SEEK
                        ;RET
                ;
                ;
                SEEK:
                        ;SEEK THE TRACK GIVEN BY ARECORD (ACTUAL RECORD)
                        ;LOCAL EQUATES FOR REGISTERS
0001 =                  ARECH   EQU B! ARECL   EQU C ;ARECORD = BC
0003 =                  CRECH   EQU D! CRECL   EQU E ;CURREC  = DE
0005 =                  CTRKH   EQU H! CTRKL   EQU L ;CURTRK  = HL
0005 =                  TCRECH EQU H! TCRECL EQU L ;TCURREC = HL
                        ;LOAD THE REGISTERS FROM MEMORY
0BD1 21E5154E23         LXI H,ARECORD! MOV ARECL,M! INX H! MOV ARECH,M
0BD7 2AB7155E23         LHLD CURRECA ! MOV CRECL,M! INX H! MOV CRECH,M
0BDD 2AB5157E23         LHLD CURTRKA ! MOV A,M! INX H! MOV CTRKH,M! MOV CTRKL,A
                        ;LOOP WHILE ARECORD < CURREC
                        SEEK0:
0BE4 7993789A                   MOV A,ARECL! SUB CRECL! MOV A,ARECH! SBB CRECH
0BE8 D2FA0B                     JNC SEEK1 ;SKIP IF ARECORD >= CURREC
                                        ;CURREC = CURREC - SECTPT
0BEB E52AC115                           PUSH CTRKH! LHLD SECTPT
0BEF 7B955F                             MOV A,CRECL! SUB L! MOV CRECL,A
0BF2 7A9C57                             MOV A,CRECH! SBB H! MOV CRECH,A
0BF5 E1                                 POP CTRKH
                                        ;CURTRK = CURTRK - 1
0BF6 2B                                 DCX CTRKH
0BF7 C3E40B                     JMP SEEK0 ;FOR ANOTHER TRY
                        SEEK1:
                        ;LOOK WHILE ARECORD >= (T:=CURREC + SECTPT)
0BFA E5                         PUSH CTRKH
0BFB 2AC11519                   LHLD SECTPT! DAD CRECH ;HL = CURREC+SECTPT
0BFF DA0F0C                       JC SEEK2              ;CAN BE > FFFFH
```

```
0C02 7995789C                   MOV A,ARECL! SUB TCRECL! MOV A,ARECH! SBB TCRECH
0C06 DA0F0C                     JC SEEK2 ;SKIP IF T > ARECORD
                                        ;CURREC = T
0C09 EB                                 XCHG
                                        ;CURTRK = CURTRK + 1
0C0A E123                               POP CTRKH! INX CTRKH
0C0C C3FA0B                     JMP SEEK1 ;FOR ANOTHER TRY
0C0F E1                 SEEK2:  POP CTRKH
                        ;ARRIVE HERE WITH UPDATED VALUES IN EACH REGISTER
0C10 C5D5E5             PUSH ARECH! PUSH CRECH! PUSH CTRKH ;TO STACK FOR LATER
                        ;STACK CONTAINS (LOWEST) BC=ARECORD, DE=CURREC, HL=CURTRK
0C13 EB2ACE1519         XCHG! LHLD OFFSET! DAD D ;HL = CURTRK+OFFSET
0C18 444DCD1E16         MOV B,H! MOV C,L! CALL SETTRKF ;TRACK SET UP
                        ;NOTE THAT BC - CURTRK IS DIFFERENCE TO MOVE IN BIOS
0C1D D1                 POP D ;RECALL CURTRK
0C1E 2AB5157323         LHLD CURTRKA! MOV M,E! INX H! MOV M,D ;CURTRK UPDATED
                        ;NOW COMPUTE SECTOR AS ARECORD-CURREC
0C24 D1                 POP CRECH ;RECALL CURREC
0C25 2AB7157323         LHLD CURRECA! MOV M,CRECL! INX H! MOV M,CRECH
0C2B C1                 POP ARECH ;BC=ARECORD, DE=CURREC
0C2C 79934F             MOV A,ARECL! SUB CRECL! MOV ARECL,A
0C2F 789A47             MOV A,ARECH! SBB CRECH! MOV ARECH,A
0C32 2AD015EB           LHLD TRANV! XCHG ;BC=SECTOR#, DE=.TRAN
0C36 CD3016             CALL SECTRAN ;HL = TRAN(SECTOR)
0C39 4D44               MOV C,L! MOV B,H ;BC = TRAN(SECTOR)
0C3B C32116             JMP SETSECF ;SECTOR SELECTED
                        ;RET
                ;
                ;       FILE CONTROL BLOCK (FCB) CONSTANTS
00E5 =          EMPTY   EQU     0E5H    ;EMPTY DIRECTORY ENTRY
007F =          LSTREC  EQU     127     ;LAST RECORD# IN EXTENT
0080 =          RECSIZ  EQU     128     ;RECORD SIZE
0020 =          FCBLEN  EQU     32      ;FILE CONTROL BLOCK SIZE
0004 =          DIRREC  EQU     RECSIZ/FCBLEN   ;DIRECTORY ELTS / RECORD
0002 =          DSKSHF  EQU     2       ;LOG2(DIRREC)
0003 =          DSKMSK  EQU     DIRREC-1
0005 =          FCBSHF  EQU     5       ;LOG2(FCBLEN)
                ;
000C =          EXTNUM  EQU     12      ;EXTENT NUMBER FIELD
001F =          MAXEXT  EQU     31      ;LARGEST EXTENT NUMBER
000D =          UBYTES  EQU     13      ;UNFILLED BYTES FIELD
000E =          MODNUM  EQU     14      ;DATA MODULE NUMBER
000F =          MAXMOD  EQU     15      ;LARGEST MODULE NUMBER
0080 =          FWFMSK  EQU     80H     ;FILE WRITE FLAG IS HIGH ORDER MODNUM
000F =          NAMLEN  EQU     15      ;NAME LENGTH
000F =          RECCNT  EQU     15      ;RECORD COUNT FIELD
0010 =          DSKMAP  EQU     16      ;DISK MAP FIELD
001F =          LSTFCB  EQU     FCBLEN-1
0020 =          NXTREC  EQU     FCBLEN
0021 =          RANREC  EQU     NXTREC+1;RANDOM RECORD FIELD (2 BYTES)
                ;
                ;       RESERVED FILE INDICATORS
0009 =          ROFILE  EQU     9       ;HIGH ORDER OF FIRST TYPE CHAR
000A =          INVIS   EQU     10      ;INVISIBLE FILE IN DIR COMMAND
                ;       EQU     11      ;RESERVED
                ;
```

```
                ;       UTILITY FUNCTIONS FOR FILE ACCESS
                ;
                DM$POSITION:
                        ;COMPUTE DISK MAP POSITION FOR VRECORD TO HL
0C3E 21C3154E           LXI H,BLKSHF! MOV C,M ;SHIFT COUNT TO C
0C42 3AE315             LDA VRECORD ;CURRENT VIRTUAL RECORD TO A
                        DMPOS0:
0C45 B71F0DC245                 ORA A! RAR! DCR C! JNZ DMPOS0
                        ;A = SHR(VRECORD,BLKSHF) = VRECORD/2**(SECT/BLOCK)
0C4B 47                 MOV B,A ;SAVE IT FOR LATER ADDITION
0C4C 3E0896             MVI A,8! SUB M ;8-BLKSHF TO ACCUMULATOR
0C4F 4F                 MOV C,A ;EXTENT SHIFT COUNT IN REGISTER C
0C50 3AE215             LDA EXTVAL ;EXTENT VALUE ANI EXTMSK
                        DMPOS1:
                                ;BLKSHF = 3,4,5,6,7, C=5,4,3,2,1
                                ;SHIFT IS 4,3,2,1,0
0C53 0DCA5C0C                   DCR C! JZ DMPOS2
0C57 B717C3530C                 ORA A! RAL! JMP DMPOS1
                        DMPOS2:
                        ;ARRIVE HERE WITH A = SHL(EXT AND EXTMSK,7-BLKSHF)
0C5C 80                 ADD B ;ADD THE PREVIOUS SHR(VRECORD,BLKSHF) VALUE
                        ;A IS ONE OF THE FOLLOWING VALUES, DEPENDING UPON ALLOC
                        ;BKS BLKSHF
                        ;1K   3      V/8 + EXTVAL * 16
                        ;2K   4      V/16+ EXTVAL * 8
                        ;4K   5      V/32+ EXTVAL * 4
                        ;8K   6      V/64+ EXTVAL * 2
                        ;16K  7      V/128+EXTVAL * 1
0C5D C9                 RET ;WITH DM$POSITION IN A
                ;
                GETDM:
                        ;RETURN DISK MAP VALUE FROM POSITION GIVEN BY BC
0C5E 2A430B             LHLD INFO ;BASE ADDRESS OF FILE CONTROL BLOCK
0C61 11100019           LXI D,DSKMAP! DAD D ;HL =.DISKMAP
0C65 09                 DAD B ;INDEX BY A SINGLE BYTE VALUE
0C66 3ADD15             LDA SINGLE ;SINGLE BYTE/MAP ENTRY?
0C69 B7CA710C           ORA A! JZ GETDMD ;GET DISK MAP SINGLE BYTE
0C6D 6E2600C9                   MOV L,M! MVI H,0! RET ;WITH HL=00BB
                        GETDMD:
0C71 09                         DAD B ;HL=.FCB(DM+I*2)
                                ;DOUBLE PRECISION VALUE RETURNED
0C72 5E2356EBC9                 MOV E,M! INX H! MOV D,M! XCHG! RET
                ;
                INDEX:
                        ;COMPUTE DISK BLOCK NUMBER FROM CURRENT FCB
0C77 CD3E0C             CALL DM$POSITION ;0...15 IN REGISTER A
0C7A 4F0600CD5E         MOV C,A! MVI B,0! CALL GETDM ;VALUE TO HL
0C80 22E515C9           SHLD ARECORD! RET
                ;
                ALLOCATED:
                        ;CALLED FOLLOWING INDEX TO SEE IF BLOCK ALLOCATED
0C84 2AE5157DB4         LHLD ARECORD! MOV A,L! ORA H! RET
                ;
                ATRAN:
                        ;COMPUTE ACTUAL RECORD ADDRESS, ASSUMING INDEX CALLED
0C8A 3AC315             LDA BLKSHF ;SHIFT COUNT TO REG A
```

```
0C8D 2AE515             LHLD ARECORD
                        ATRANO:
0C90 293DC2900C                 DAD H! DCR A! JNZ ATRANO ;SHL(ARECORD,BLKSHF)
0C95 22E715                     SHLD ARECORD1           ;SAVE SHIFTED BLOCK #
0C98 3AC4154F           LDA BLKMSK! MOV C,A ;MASK VALUE TO C
0C9C 3AE315A1           LDA VRECORD! ANA C ;MASKED VALUE IN A
0CA0 B56F               ORA L! MOV L,A ;TO HL
0CA2 22E515             SHLD ARECORD ;ARECORD=HL OR (VRECORD AND BLKMSK)
0CA5 C9                 RET
                ;
                GETEXTA:
                        ;GET CURRENT EXTENT FIELD ADDRESS TO A
0CA6 2A430B110C         LHLD INFO! LXI D,EXTNUM! DAD D ;HL=.FCB(EXTNUM)
0CAD C9                 RET
                ;
                GETFCBA:
                        ;COMPUTE RECCNT AND NXTREC ADDRESSES FOR GET/SETFCB
0CAE 2A430B110F         LHLD INFO! LXI D,RECCNT! DAD D! XCHG ;DE=.FCB(RECCNT)
0CB6 21110019           LXI H,(NXTREC-RECCNT)! DAD D ;HL=.FCB(NXTREC)
0CBA C9                 RET
                ;
                GETFCB:
                        ;SET VARIABLES FROM CURRENTLY ADDRESSED FCB
0CBB CDAE0C             CALL GETFCBA ;ADDRESSES IN DE, HL
0CBE 7E32E315           MOV A,M! STA VRECORD ;VRECORD=FCB(NXTREC)
0CC2 EB7E32E115         XCHG! MOV A,M! STA RCOUNT ;RCOUNT=FCB(RECCNT)
0CC7 CDA60C             CALL GETEXTA ;HL=.FCB(EXTNUM)
0CCA 3AC515             LDA EXTMSK ;EXTENT MASK TO A
0CCD A6                 ANA M ;FCB(EXTNUM) AND EXTMSK
0CCE 32E215             STA EXTVAL
0CD1 C9                 RET
                ;
                SETFCB:
                        ;PLACE VALUES BACK INTO CURRENT FCB
0CD2 CDAE0C             CALL GETFCBA ;ADDRESSES TO DE, HL
0CD5 3AD515             LDA SEQIO
0CD8 FE02C2DE0C         CPI 02! JNZ SETFCB1! XRA A      ;CHECK RANFILL
                SETFCB1:
0CDE 4F                 MOV C,A ;=1 IF SEQUENTIAL I/O
0CDF 3AE3158177         LDA VRECORD! ADD C! MOV M,A ;FCB(NXTREC)=VRECORD+SEQIO
0CE4 EB3AE11577         XCHG! LDA RCOUNT! MOV M,A ;FCB(RECCNT)=RCOUNT
0CE9 C9                 RET
                ;
                HLROTR:
                        ;HL ROTATE RIGHT BY AMOUNT C
0CEA 0C                 INR C ;IN CASE ZERO
0CEB 0DC8               HLROTR0: DCR C! RZ ;RETURN WHEN ZERO
0CED 7CB71F67                   MOV A,H! ORA A! RAR! MOV H,A ;HIGH BYTE
0CF1 7D1F6F                     MOV A,L! RAR! MOV L,A ;LOW BYTE
0CF4 C3EB0C                     JMP HLROTR0
                ;
                ;
                COMPUTE$CS:
                        ;COMPUTE CHECKSUM FOR CURRENT DIRECTORY BUFFER
0CF7 0E80               MVI C,RECSIZ ;SIZE OF DIRECTORY BUFFER
0CF9 2AB915             LHLD BUFFA ;CURRENT DIRECTORY BUFFER
```

```
0CFC AF                 XRA A ;CLEAR CHECKSUM VALUE
                        COMPUTECS0:
0CFD 86230D                     ADD M! INX H! DCR C ;CS=CS+BUFF(RECSIZ-C)
0D00 C2FD0C                     JNZ COMPUTECS0
0D03 C9                 RET ;WITH CHECKSUM IN A
                ;
                HLROTL:
                        ;ROTATE THE MASK IN HL BY AMOUNT IN C
0D04 0C                 INR C ;MAY BE ZERO
0D05 0DC8               HLROTL0: DCR C! RZ ;RETURN IF ZERO
0D07 29C3050D                   DAD H! JMP HLROTL0
                ;
                SET$CDISK:
                        ;SET A "1" VALUE IN CURDSK POSITION OF BC
0D0B C5                 PUSH B ;SAVE INPUT PARAMETER
0D0C 3A420B4F           LDA CURDSK! MOV C,A ;READY PARAMETER FOR SHIFT
0D10 210100             LXI H,1 ;NUMBER TO SHIFT
0D13 CD040D             CALL HLROTL ;HL = MASK TO INTEGRATE
0D16 C1                 POP B ;ORIGINAL MASK
0D17 79B56F             MOV A,C! ORA L! MOV L,A
0D1A 78B467             MOV A,B! ORA H! MOV H,A ;HL = MASK OR ROL(1,CURDSK)
0D1D C9                 RET
                ;
                NOWRITE:
                        ;RETURN TRUE IF DIR CHECKSUM DIFFERENCE OCCURRED
0D1E 2AAD153A42         LHLD RODSK! LDA CURDSK! MOV C,A! CALL HLROTR
0D28 7DE601C9           MOV A,L! ANI 1B! RET ;NON ZERO IF NOWRITE
                ;
                SET$RO:
                        ;SET CURRENT DISK TO READ ONLY
0D2C 21AD154E23         LXI H,RODSK! MOV C,M! INX H! MOV B,M
0D32 CD0B0D             CALL SET$CDISK ;SETS BIT TO 1
0D35 22AD15             SHLD RODSK
                        ;HIGH WATER MARK IN DIRECTORY GOES TO MAX
0D38 2AC81523EB         LHLD DIRMAX! INX H! XCHG ;DE = DIRECTORY MAX
0D3D 2AB315             LHLD CDRMAXA ;HL = .CDRMAX
0D40 732372             MOV M,E! INX H! MOV M,D ;CDRMAX = DIRMAX
0D43 C9                 RET
                ;
                CHECK$RODIR:
                        ;CHECK CURRENT DIRECTORY ELEMENT FOR READ/ONLY STATUS
0D44 CD5E0D             CALL GETDPTRA ;ADDRESS OF ELEMENT
                ;
                CHECK$ROFILE:
                        ;CHECK CURRENT BUFF(DPTR) OR FCB(0) FOR R/O STATUS
0D47 11090019           LXI D,ROFILE! DAD D ;OFFSET TO RO BIT
0D4B 7E17D0             MOV A,M! RAL! RNC ;RETURN IF NOT SET
0D4E 210F08C34A         LXI H,ROFERR! JMP GOERR                 ;
                ;       JMP ROF$ERROR ;EXIT TO READ ONLY DISK MESSAGE
                ;
                ;
                CHECK$WRITE:
                        ;CHECK FOR WRITE PROTECTED DISK
0D54 CD1E0DC8           CALL NOWRITE! RZ ;OK TO WRITE IF NOT RODSK
0D58 210D08C34A         LXI H,RODERR! JMP GOERR
                ;       JMP ROD$ERROR ;READ ONLY DISK ERROR
```

```
                ;
                GETDPTRA:
                        ;COMPUTE THE ADDRESS OF A DIRECTORY ELEMENT AT
                        ;POSITON DPTR IN THE BUFFER
0D5E 2AB9153AE9         LHLD BUFFA! LDA DPTR
                ADDH:
                        ;HL = HL + A
0D64 856FD0             ADD L! MOV L,A! RNC
                        ;OVERFLOW TO H
0D67 24C9               INR H! RET
                ;
                ;
                GETMODNUM:
                        ;COMPUTE THE ADDRESS OF THE MODULE NUMBER
                        ;BRING MODULE NUMBER TO ACCUMULATOR
                        ;(HIGH ORDER BIT IS FWF (FILE WRITE FLAG)
0D69 2A430B110E         LHLD INFO! LXI D,MODNUM! DAD D ;HL=.FCB(MODNUM)
0D70 7EC9               MOV A,M! RET ;A=FCB(MODNUM)
                ;
                CLRMODNUM:
                        ;CLEAR THE MODULE NUMBER FIELD FOR USER OPEN/MAKE
0D72 CD690D3600         CALL GETMODNUM! MVI M,0 ;FCB(MODNUM)=0
0D77 C9                 RET
                ;
                SETFWF:
0D78 CD690D             CALL GETMODNUM ;HL=.FCB(MODNUM), A=FCB(MODNUM)
                        ;SET FWF (FILE WRITE FLAG) TO "1"
0D7B F68077             ORI FWFMSK! MOV M,A ;FCB(MODNUM)=FCB(MODNUM) OR 80H
                        ;ALSO RETURNS NON ZERO IN ACCUMULATOR
0D7E C9                 RET
                ;
                ;
                COMPCDR:
                        ;RETURN CY IF CDRMAX > DCNT
0D7F 2AEA15EB           LHLD DCNT! XCHG ;DE = DIRECTORY COUNTER
0D83 2AB315             LHLD CDRMAXA ;HL=.CDRMAX
0D86 7B96               MOV A,E! SUB M ;LOW(DCNT) - LOW(CDRMAX)
0D88 23                 INX H ;HL = .CDRMAX+1
0D89 7A9E               MOV A,D! SBB M ;HIG(DCNT) - HIG(CDRMAX)
                        ;CONDITION DCNT - CDRMAX  PRODUCES CY IF CDRMAX>DCNT
0D8B C9                 RET
                ;
                SETCDR:
                        ;IF NOT (CDRMAX > DCNT) THEN CDRMAX = DCNT+1
0D8C CD7F0D             CALL COMPCDR
0D8F D8                 RC ;RETURN IF CDRMAX > DCNT
                        ;OTHERWISE, HL = .CDRMAX+1, DE = DCNT
0D90 13722B73           INX D! MOV M,D! DCX H! MOV M,E
0D94 C9                 RET
                ;
                SUBDH:
                        ;COMPUTE HL = DE - HL
0D95 7B956F7A9C         MOV A,E! SUB L! MOV L,A! MOV A,D! SBB H! MOV H,A
0D9B C9                 RET
                ;
                NEWCHECKSUM:
```

```
0D9C 0EFF               MVI C,TRUE ;DROP THROUGH TO COMPUTE NEW CHECKSUM
                CHECKSUM:
                        ;COMPUTE CURRENT CHECKSUM RECORD AND UPDATE THE
                        ;DIRECTORY ELEMENT IF C=TRUE, OR CHECK FOR = IF NOT
                        ;DREC < CHKSIZ?
0D9E 2AEC15EB2A         LHLD DREC! XCHG! LHLD CHKSIZ! CALL SUBDH ;DE-HL
0DA8 D0                 RNC ;SKIP CHECKSUM IF PAST CHECKSUM VECTOR SIZE
                                ;DREC < CHKSIZ, SO CONTINUE
0DA9 C5                         PUSH B ;SAVE INIT FLAG
0DAA CDF70C                     CALL COMPUTE$CS ;CHECK SUM VALUE TO A
0DAD 2A[D]15                    LHLD CHECKA ;ADDRESS OF CHECK SUM VECTOR
0DB0 EB                         XCHG
0DB1 2AEC15                     LHLD DREC ;VALUE OF DREC
0DB4 19                         DAD D ;HL = .CHECK(DREC)
0DB5 C1                         POP B ;RECALL TRUE=0FFH OR FALSE=00 TO C
0DB6 0C                         INR C ;0FFH PRODUCES ZERO FLAG
0DB7 CA[4]0D                    JZ INITIAL$CS
                                        ;NOT INITIALIZING, COMPARE
0DBA BE                                 CMP M ;COMPUTE$CS=CHECK(DREC)?
0DBB C8                                 RZ ;NO MESSAGE IF OK
                                        ;CHECKSUM ERROR, ARE WE BEYOND
                                        ;THE END OF THE DISK?
0DBC CD7F0D                             CALL COMPCDR
0DBF D0                                 RNC ;NO MESSAGE IF SO
0DC0 CD2C0D                             CALL SET$RO ;READ/ONLY DISK SET
0DC3 C9                                 RET
                                INITIAL$CS:
                                        ;INITIALIZING THE CHECKSUM
0DC4 77C9                               MOV M,A! RET
                ;
                ;
                WRDIR:
                        ;WRITE THE CURRENT DIRECTORY ENTRY, SET CHECKSUM
0DC6 CD9C0D             CALL NEWCHECKSUM ;INITIALIZE ENTRY
0DC9 CDE00D             CALL SETDIR ;DIRECTORY DMA
0DCC 0E01               MVI C,1 ;INDICATES A WRITE DIRECTORY OPERATION
0DCE CDB80B             CALL WRBUFF ;WRITE THE BUFFER
0DD1 C3DA0D             JMP SETDATA ;TO DATA DMA ADDRESS
                        ;RET
                ;
                RD$DIR:
                        ;READ A DIRECTORY ENTRY INTO THE DIRECTORY BUFFER
0DD4 CDE00D             CALL SETDIR ;DIRECTORY DMA
0DD7 CDB20B             CALL RDBUFF ;DIRECTORY RECORD LOADED
                        ; JMP SETDATA TO DATA DMA ADDRESS
                        ;RET
                ;
                SETDATA:
                        ;SET DATA DMA ADDRESS
0DDA 21B115C3E3         LXI H,DMAAD! JMP SETDMA ;TO COMPLETE THE CALL
                ;
                SETDIR:
                        ;SET DIRECTORY DMA ADDRESS
0DE0 21B915             LXI H,BUFFA  ;JMP SETDMA TO COMPLETE CALL
                ;
                SETDMA:
```

```
                        ;HL=.DMA ADDRESS TO SET (I.E., BUFFA OR DMAAD)
0DE3 4E2346             MOV C,M! INX H! MOV B,M ;PARAMETER READY
0DE6 C32416             JMP SETDMAF
                ;
                ;
                DIR$TO$USER:
                        ;COPY THE DIRECTORY ENTRY TO THE USER BUFFER
                        ;AFTER CALL TO SEARCH OR SEARCHN BY USER CODE
0DE9 2AB915EB           LHLD BUFFA! XCHG ;SOURCE IS DIRECTORY BUFFER
0DED 2AB115             LHLD DMAAD ;DESTINATION IS USER DMA ADDRESS
0DF0 0E80               MVI C,RECSIZ ;COPY ENTIRE RECORD
0DF2 C34F0B             JMP MOVE
                        ;RET
                ;
                END$OF$DIR:
                        ;RETURN ZERO FLAG IF AT END OF DIRECTORY, NON ZERO
                         IF NOT AT END (END OF DIR IF DCNT = 0FFFFH)
0DF5 21EA157E           LXI H,DCNT! MOV A,M ;MAY BE 0FFH
0DF9 23BE               INX H! CMP M ;LOW(DCNT) = HIGH(DCNT)?
0DFB C0                 RNZ ;NON ZERO RETURNED IF DIFFERENT
                        ;HIGH AND LOW THE SAME, = 0FFH?
0DFC 3C                 INR A ;0FFH BECOMES 00 IF SO
0DFD C9                 RET
                ;
                SET$END$DIR:
                        ;SET DCNT TO THE END OF THE DIRECTORY
0DFE 21FFFF22EA         LXI H,ENDDIR! SHLD DCNT! RET
                ;
                READ$DIR:
                        ;READ NEXT DIRECTORY ENTRY, WITH C=TRUE IF INITIALIZING
0E05 2AC815EB           LHLD DIRMAX! XCHG ;IN PREPARATION FOR SUBTRACT
0E09 2AEA152322         LHLD DCNT! INX H! SHLD DCNT ;DCNT=DCNT+1
                        ;CONTINUE WHILE DIRMAX >= DCNT (DIRMAX-DCNT NO CY)
0E10 CD950D             CALL SUBDH ;DE-HL
0E13 D2190E             JNC READ$DIRO
                                ;YES, SET DCNT TO END OF DIRECTORY
0E16 C3FE0D                     JMP SET$END$DIR                 ;
                ;               RET                             ;
                        READ$DIRO:
                                ;NOT AT END OF DIRECTORY, SEEK NEXT ELEMENT
                                ;INITIALIZATION FLAG IS IN C
0E19 3AEA15E603                 LDA DCNT! ANI DSKMSK ;LOW(DCNT) AND DSKMSK
0E1E 0605                       MVI B,FCBSHF ;TO MULTIPLY BY FCB SIZE
                                READ$DIR1:
0E20 8705C2200E                         ADD A! DCR B! JNZ READ$DIR1
                                ;A = (LOW(DCNT) AND DSKMSK) SHL FCBSHF
0E25 32E915                     STA DPTR ;READY FOR NEXT DIR OPERATION
0E28 B7C0                       ORA A! RNZ ;RETURN IF NOT A NEW RECORD
0E2A C5                         PUSH B ;SAVE INITIALIZATION FLAG C
0E2B CDC30B                     CALL SEEK$DIR ;SEEK PROPER RECORD
0E2E CDD40D                     CALL RD$DIR ;READ THE DIRECTORY RECORD
0E31 C1                         POP B ;RECALL INITIALIZATION FLAG
0E32 C39E0D                     JMP CHECKSUM ;CHECKSUM THE DIRECTORY ELT
                                ;RET
                ;
                ;
```

```
                GETALLOCBIT:
                        ;GIVEN ALLOCATION VECTOR POSITION BC, RETURN WITH BYTE
                        ;CONTAINING BC SHIFTED SO THAT THE LEAST SIGNIFICANT
                        ;BIT IS IN THE LOW ORDER ACCUMULATOR POSITION.  HL IS
                        ;THE ADDRESS OF THE BYTE FOR POSSIBLE REPLACEMENT IN
                        ;MEMORY UPON RETURN, AND D CONTAINS THE NUMBER OF SHIFTS
                        ;REQUIRED TO PLACE THE RETURNED VALUE BACK INTO POSITION
0E35 79E6073C5F         MOV A,C! ANI 111B! INR A! MOV E,A! MOV D,A
                        ;D AND E BOTH CONTAIN THE NUMBER OF BIT POSITIONS TO SHIFT
0E3B 790F0F0FE6         MOV A,C! RRC! RRC! RRC! ANI 11111B! MOV C,A ;C SHR 3 TO C
0E42 7887878787         MOV A,B! ADD A! ADD A! ADD A! ADD A! ADD A ;B SHL 5
0E48 B14F               ORA C! MOV C,A ;BBBCCCCC TO C
0E4A 780F0F0FE6         MOV A,B! RRC! RRC! RRC! ANI 11111B! MOV B,A ;BC SHR 3 TO BC
0E51 2ABF15             LHLD ALLOCA ;BASE ADDRESS OF ALLOCATION VECTOR
0E54 097E               DAD B! MOV A,M ;BYTE TO A, HL = .ALLOC(BC SHR 3)
                        ;NOW MOVE THE BIT TO THE LOW ORDER POSITION OF A
0E56 071DC2560E         ROTL: RLC! DCR E! JNZ ROTL! RET
                ;
                ;
                SETALLOCBIT:
                        ;BC IS THE BIT POSITION OF ALLOC TO SET OR RESET.  THE
                        ;VALUE OF THE BIT IS IN REGISTER E.
0E5C D5CD350E           PUSH D! CALL GETALLOCBIT ;SHIFTED VAL A, COUNT IN D
0E60 E6FE               ANI 1111$1110B ;MASK LOW BIT TO ZERO (MAY BE SET)
0E62 C1B1               POP B! ORA C ;LOW BIT OF C IS MASKED INTO A
                ;       JMP ROTR ;TO ROTATE BACK INTO PROPER POSITION
                        ;RET
                ROTR:
                        ;BYTE VALUE FROM ALLOC IS IN REGISTER A, WITH SHIFT COUNT
                        ;IN REGISTER C (TO PLACE BIT BACK INTO POSITION), AND
                        ;TARGET ALLOC POSITION IN REGISTERS HL, ROTATE AND REPLACE
0E64 0F15C2640E         RRC! DCR D! JNZ ROTR ;BACK INTO POSITION
0E69 77                 MOV M,A ;BACK TO ALLOC
0E6A C9                 RET
                ;
                SCANDM:
                        ;SCAN THE DISK MAP ADDRESSED BY DPTR FOR NON-ZERO
                        ;ENTRIES, THE ALLOCATION VECTOR ENTRY CORRESPONDING
                        ;TO A NON-ZERO ENTRY IS SET TO THE VALUE OF C (0,1)
0E6B CD5E0D             CALL GETDPTRA ;HL = BUFFA + DPTR
                        ;HL ADDRESSES THE BEGINNING OF THE DIRECTORY ENTRY
0E6E 11100019           LXI D,DSKMAP! DAD D ;HL NOW ADDRESSES THE DISK MAP
0E72 C5                 PUSH B ;SAVE THE 0/1 BIT TO SET
0E73 0E11               MVI C,FCBLEN-DSKMAP+1 ;SIZE OF SINGLE BYTE DISK MAP + 1
                        SCANDM0:
                                ;LOOP ONCE FOR EACH DISK MAP ENTRY
0E75 D1                         POP D ;RECALL BIT PARITY
0E76 0DC8                       DCR C! RZ ;ALL DONE SCANNING?
                                ;NO, GET NEXT ENTRY FOR SCAN
0E78 D5                         PUSH D ;REPLACE BIT PARITY
0E79 3ADD15B7CA                 LDA SINGLE! ORA A! JZ SCANDM1
                                        ;SINGLE BYTE SCAN OPERATION
0E80 C5                         PUSH B ;SAVE COUNTER
0E81 E5                         PUSH H ;SAVE MAP ADDRESS
0E82 4E0600                     MOV C,M! MVI B,0 ;BC=BLOCK#
0E85 C38E0E                     JMP SCANDM2
```

```
                                SCANDM1:
                                        ;DOUBLE BYTE SCAN OPERATION
0E88 0D                                 DCR C ;COUNT FOR DOUBLE BYTE
0E89 C5                                 PUSH B ;SAVE COUNTER
0E8A 4E2346                             MOV C,M! INX H! MOV B,M ;BC=BLOCK#
0E8D E5                                 PUSH H ;SAVE MAP ADDRESS
                                SCANDM2:
                                        ;ARRIVE HERE WITH BC=BLOCK#, E=0/1
0E8E 79B0                               MOV A,C! ORA B ;SKIP IF = 0000
0E90 CA9D0E                             JZ SCANM3
0E93 2AC615                                     LHLD MAXALL     ;CHECK INVALID INDEX
0E96 7D917C98                           MOV A,L! SUB C! MOV A,H! SBB B  ;MAXALL - BLOCK
0E9A D45C0E                             CNC SET$ALLOC$BIT                       ;
                                        ;BIT SET TO 0/1
                                SCANM3:                                         ;
0E9D E123                               POP H! INX H ;TO NEXT BIT POSITION
0E9F C1                                 POP B ;RECALL COUNTER
0EA0 C3750E                             JMP SCANDM0 ;FOR ANOTHER ITEM
                        ;
                        INITIALIZE:
                                ;INITIALIZE THE CURRENT DISK
                                ;LRET = FALSE ;SET TO TRUE IF $ FILE EXISTS
                                ;COMPUTE THE LENGTH OF THE ALLOCATION VECTOR - 2
0EA3 2AC6150E03                 LHLD MAXALL! MVI C,3 ;PERFORM MAXALL/8
                                ;NUMBER OF BYTES IN ALLOC VECTOR IS (MAXALL/8)+1
0EA8 CDEA0C23                   CALL HLROTR! INX H ;HL = MAXALL/8+1
0EAC 444D                       MOV B,H! MOV C,L ;COUNT DOWN BC TIL ZERO
0EAE 2ABF15                     LHLD ALLOCA ;BASE OF ALLOCATION VECTOR
                                ;FILL THE ALLOCATION VECTOR WITH ZEROS
                                INITIAL0:
0EB1 360023                             MVI M,0! INX H ;ALLOC(I)=0
0EB4 0B                                 DCX B ;COUNT LENGTH DOWN
0EB5 78B1C2B10E                         MOV A,B! ORA C! JNZ INITIAL0
                                ;SET THE RESERVED SPACE FOR THE DIRECTORY
0EBA 2ACA15EB                   LHLD DIRBLK! XCHG
0EBE 2ABF15                     LHLD ALLOCA ;HL=.ALLOC()
0EC1 732372                     MOV M,E! INX H! MOV M,D ;SETS RESERVED DIRECTORY BLKS
                                ;ALLOCATION VECTOR INITIALIZED, HOME DISK
0EC4 CDA10B                     CALL HOME
                                ;CDRMAX = 3 (SCANS AT LEAST ONE DIRECTORY RECORD)
0EC7 2AB3153603                 LHLD CDRMAXA! MVI M,3! INX H! MVI M,0
                                ;CDRMAX = 0000
0ECF CDFE0D                     CALL SET$END$DIR ;DCNT = ENDDIR
                                ;READ DIRECTORY ENTRIES AND CHECK FOR ALLOCATED STORAGE
                                INITIAL2:
0ED2 0EFFCD050E                         MVI C,TRUE! CALL READ$DIR
0ED7 CDF50DC8                           CALL END$OF$DIR! RZ ;RETURN IF END OF DIRECTORY
                                        ;NOT END OF DIRECTORY, VALID ENTRY?
0EDB CD5E0D                             CALL GETDPTRA ;HL = BUFFA + DPTR
0EDE 3EE5BE                             MVI A,EMPTY! CMP M
0EE1 CAD20E                             JZ INITIAL2 ;GO GET ANOTHER ITEM
                                        ;NOT EMPTY, USER CODE THE SAME?
0EE4 3A410B                             LDA USRCODE
0EE7 BEC2F60E                           CMP M! JNZ PDOLLAR
                                        ;SAME USER CODE, CHECK FOR '$'
0EEB 237E                               INX H! MOV A,M ;FIRST CHARACTER
```

```
0EED D624                       SUI '$' ;DOLLAR FILE?
0EEF C2F60E                     JNZ PDOLLAR
                                ;DOLLAR FILE FOUND, MARK IN LRET
0EF2 3D32450B                   DCR A! STA LRET ;LRET = 255
                        PDOLLAR:
                                ;NOW SCAN THE DISK MAP FOR ALLOCATED BLOCKS
0EF6 0E01                       MVI C,1 ;SET TO ALLOCATED
0EF8 CD6B0E                     CALL SCANDM
0EFB CD8C0D                     CALL SETCDR ;SET CDRMAX TO DCNT
0EFE C3D20E                     JMP INITIAL2 ;FOR ANOTHER ENTRY
                ;
                COPY$DIRLOC:
                        ;COPY DIRECTORY LOCATION TO LRET FOLLOWING
                        ;DELETE, RENAME, ... OPS
0F01 3AD415C301         LDA DIRLOC! JMP STA$RET                         ;
                ;       RET                                             ;
                ;
                COMPEXT:
                        ;COMPARE EXTENT# IN A WITH THAT IN C, RETURN NONZERO
                        ;IF THEY DO NOT MATCH
0F07 C5                 PUSH B ;SAVE C'S ORIGINAL VALUE
0F08 F53AC5152F         PUSH PSW! LDA EXTMSK! CMA! MOV B,A
                        ;B HAS NEGATED FORM OF EXTENT MASK
0F0E 79A04F             MOV A,C! ANA B! MOV C,A ;LOW BITS REMOVED FROM C
0F11 F1A0               POP PSW! ANA B ;LOW BITS REMOVED FROM A
0F13 91E61F             SUB C! ANI MAXEXT ;SET FLAGS
0F16 C1                 POP B ;RESTORE ORIGINAL VALUES
0F17 C9                 RET
                ;
                SEARCH:
                        ;SEARCH FOR DIRECTORY ELEMENT OF LENGTH C AT INFO
0F18 3EFF32D415         MVI A,0FFH! STA DIRLOC ;CHANGED IF ACTUALLY FOUND
0F1D 21D81571           LXI H,SEARCHL! MOV M,C ;SEARCHL = C
0F21 2A430B22D9         LHLD INFO! SHLD SEARCHA ;SEARCHA = INFO
0F27 CDFE0D             CALL SET$END$DIR ;DCNT = ENDDIR
0F2A CDA10B             CALL HOME ;TO START AT THE BEGINNING
                        ;(DROP THROUGH TO SEARCHN)                      ;
                ;
                SEARCHN:
                        ;SEARCH FOR THE NEXT DIRECTORY ELEMENT, ASSUMING
                        ;A PREVIOUS CALL ON SEARCH WHICH SETS SEARCHA AND
                        ;SEARCHL
0F2D 0E00CD050E         MVI C,FALSE! CALL READ$DIR ;READ NEXT DIR ELEMENT
0F32 CDF50DCA94         CALL END$OF$DIR! JZ SEARCH$FIN ;SKIP TO END IF SO
                                ;NOT END OF DIRECTORY, SCAN FOR MATCH
0F38 2AD915EB                   LHLD SEARCHA! XCHG ;DE=BEGINNING OF USER FCB
0F3C 1A                         LDAX D ;FIRST CHARACTER
0F3D FEE5                       CPI EMPTY ;KEEP SCANNING IF EMPTY
0F3F CA4A0F                     JZ SEARCHNEXT
                                ;NOT EMPTY, MAY BE END OF LOGICAL DIRECTORY
0F42 D5                         PUSH D ;SAVE SEARCH ADDRESS
0F43 CD7F0D                     CALL COMPCDR ;PAST LOGICAL END?
0F46 D1                         POP D ;RECALL ADDRESS
0F47 D2940F                     JNC SEARCH$FIN ;ARTIFICIAL STOP
                        SEARCHNEXT:
0F4A CD5E0D                     CALL GETDPTRA ;HL = BUFFA+DPTR
```

```
0F4D 3AD8154F                   LDA SEARCHL! MOV C,A ;LENGTH OF SEARCH TO C
0F51 0600                       MVI B,0 ;B COUNTS UP, C COUNTS DOWN
                                SEARCHLOOP:
0F53 79B7CA830F                         MOV A,C! ORA A! JZ ENDSEARCH
0F58 1AFE3FCA7C                         LDAX D! CPI '?'! JZ SEARCHOK ;? MATCHES ALL
                                        ;SCAN NEXT CHARACTER IF NOT UBYTES
0F5E 78FE0DCA7C                         MOV A,B! CPI UBYTES! JZ SEARCHOK
                                        ;NOT THE UBYTES FIELD, EXTENT FIELD?
0F64 FE0C                               CPI EXTNUM ;MAY BE EXTENT FIELD
0F66 1A                                 LDAX D ;FCB CHARACTER
0F67 CA730F                             JZ SEARCHEXT ;SKIP TO SEARCH EXTENT
0F6A 96E67F                             SUB M! ANI 7FH ;MASK-OUT FLAGS/EXTENT MODULUS
0F6D C22D0F                             JNZ SEARCHN ;SKIP IF NOT MATCHED
0F70 C37C0F                             JMP SEARCHOK ;MATCHED CHARACTER
                                SEARCHEXT:
                                        ;A HAS FCB CHARACTER
                                        ;ATTEMPT AN EXTENT # MATCH
0F73 C5                                 PUSH B ;SAVE COUNTERS
0F74 4E                                 MOV C,M ;DIRECTORY CHARACTER TO C
0F75 CD070F                             CALL COMPEXT ;COMPARE USER/DIR CHAR
0F78 C1                                 POP B ;RECALL COUNTERS
0F79 C22D0F                             JNZ SEARCHN ;SKIP IF NO MATCH
                                SEARCHOK:
                                        ;CURRENT CHARACTER MATCHES
0F7C 1323040D                           INX D! INX H! INR B! DCR C
0F80 C3530F                             JMP SEARCHLOOP
                                ENDSEARCH:
                                        ;ENTIRE NAME MATCHES, RETURN DIR POSITION
0F83 3AEA15E603                         LDA DCNT! ANI DSKMSK! STA LRET
                                        ;LRET = LOW(DCNT) AND 11B
0F8B 21D4157E17                         LXI H,DIRLOC! MOV A,M! RAL! RNC ;DIRLOC=0FFH?
                                        ;YES, CHANGE IT TO 0 TO MARK AS FOUND
0F91 AF77                               XRA A! MOV M,A ;DIRLOC=0
0F93 C9                                 RET
                                SEARCH$FIN:
                                        ;END OF DIRECTORY, OR EMPTY NAME
0F94 CDFE0D                             CALL SET$END$DIR ;MAY BE ARTIFICAL END
0F97 3EFFC3010B                         MVI A,255! JMP STA$RET          ;
                        ;
                        ;
                        DELETE:
                                ;DELETE THE CURRENTLY ADDRESSED FILE
0F9C CD540D                     CALL CHECK$WRITE ;WRITE PROTECTED?
0F9F 0E0CCD180F                 MVI C,EXTNUM! CALL SEARCH ;SEARCH THROUGH FILE TYPE
                                DELETE0:
                                        ;LOOP WHILE DIRECTORY MATCHES
0FA4 CDF50DC8                           CALL END$OF$DIR! RZ ;STOP IF END
                                        ;SET EACH NON ZERO DISK MAP ENTRY TO 0
                                        ;IN THE ALLOCATION VECTOR
                                        ;MAY BE R/O FILE
0FA8 CD440D                             CALL CHECK$RODIR ;RO DISK ERROR IF FOUND
0FAB CD5E0D                             CALL GETDPTRA ;HL=.BUFF(DPTR)
0FAE 36E5                               MVI M,EMPTY
0FB0 0E00CD6B0E                         MVI C,0! CALL SCANDM ;ALLOC ELTS SET TO 0
0FB5 CDC60D                             CALL WRDIR ;WRITE THE DIRECTORY
0FB8 CD2D0F                             CALL SEARCHN ;TO NEXT ELEMENT
```

```
0FBB C3A40F                     JMP DELETE0 ;FOR ANOTHER RECORD
                ;
                GET$BLOCK:
                        ;GIVEN ALLOCATION VECTOR POSITION BC, FIND THE ZERO BIT
                        ;CLOSEST TO THIS POSITION BY SEARCHING LEFT AND RIGHT.
                        ;IF FOUND, SET THE BIT TO ONE AND RETURN THE BIT POSITION
                        ;IN HL.  IF NOT FOUND (I.E., WE PASS 0 ON THE LEFT, OR
                        ;MAXALL ON THE RIGHT), RETURN 0000 IN HL
0FBE 5059               MOV D,B! MOV E,C ;COPY OF STARTING POSITION TO DE
                        LEFTTST:
0FC0 79B0CAD10F                 MOV A,C! ORA B! JZ RIGHTTST ;SKIP IF LEFT=0000
                                ;LEFT NOT AT POSITION ZERO, BIT ZERO?
0FC5 0BD5C5                     DCX B! PUSH D! PUSH B ;LEFT,RIGHT PUSHED
0FC8 CD350E                     CALL GETALLOCBIT
0FCB 1FD2EC0F                   RAR! JNC RETBLOCK ;RETURN BLOCK NUMBER IF ZERO
                                ;BIT IS ONE, SO TRY THE RIGHT
0FCF C1D1                       POP B! POP D ;LEFT, RIGHT RESTORED
                        RIGHTTST:
0FD1 2AC615                     LHLD MAXALL ;VALUE OF MAXIMUM ALLOCATION#
0FD4 7B957A9C                   MOV A,E! SUB L! MOV A,D! SBB H ;RIGHT=MAXALL?
0FD8 D2F40F                     JNC RETBLOCK0 ;RETURN BLOCK 0000 IF SO
0FDB 13C5D5                     INX D! PUSH B! PUSH D ;LEFT, RIGHT PUSHED
0FDE 424B                       MOV B,D! MOV C,E ;READY RIGHT FOR CALL
0FE0 CD350E                     CALL GETALLOCBIT
0FE3 1FD2EC0F                   RAR! JNC RETBLOCK ;RETURN BLOCK NUMBER IF ZERO
0FE7 D1C1                       POP D! POP B ;RESTORE LEFT AND RIGHT POINTERS
0FE9 C3C00F                     JMP LEFTTST ;FOR ANOTHER ATTEMPT
                        RETBLOCK:
0FEC 173C                       RAL! INR A ;BIT BACK INTO POSITION AND SET TO 1
                                ;D CONTAINS THE NUMBER OF SHIFTS REQUIRED TO REPOSITION
0FEE CD640E                     CALL ROTR ;MOVE BIT BACK TO POSITION AND STORE
0FF1 E1D1                       POP H! POP D ;HL RETURNED VALUE, DE DISCARDED
0FF3 C9                         RET
                        RETBLOCK0:
                                ;CANNOT FIND AN AVAILABLE BIT, RETURN 0000
0FF4 79                         MOV A,C                 ;
0FF5 B0C2C00F                   ORA B! JNZ LEFTTST      ;ALSO AT BEGINNING
0FF9 210000C9                   LXI H,0000H! RET
                ;
                COPY$FCB:
                        ;COPY THE ENTIRE FILE CONTROL BLOCK
0FFD 0E001E20           MVI C,0! MVI E,FCBLEN ;START AT 0, TO FCBLEN-1
                        ;       JMP COPY$DIR                    ;
                ;
                COPY$DIR:
                        ;COPY FCB INFORMATION STARTING AT C FOR E BYTES
                        ;INTO THE CURRENTLY ADDRESSED DIRECTORY ENTRY
1001 D5                 PUSH D ;SAVE LENGTH FOR LATER
1002 0600               MVI B,0 ;DOUBLE INDEX TO BC
1004 2A430B             LHLD INFO ;HL = SOURCE FOR DATA
1007 09EB               DAD B! XCHG ;DE=.FCB(C), SOURCE FOR COPY
1009 CD5E0D             CALL GETDPTRA ;HL=.BUFF(DPTR), DESTINATION
100C C1                 POP B ;DE=SOURCE, HL=DEST, C=LENGTH
100D CD4F0B             CALL MOVE ;DATA MOVED
                SEEK$COPY:
                        ;ENTER FROM CLOSE TO SEEK AND COPY CURRENT ELEMENT
```

```
1010 CDC30B             CALL SEEK$DIR ;TO THE DIRECTORY ELEMENT
1013 C3C60D             JMP WRDIR ;WRITE THE DIRECTORY ELEMENT
                        ;RET
                ;
                ;
                RENAME:
                        ;RENAME THE FILE DESCRIBED BY THE FIRST HALF OF
                        ;THE CURRENTLY ADDRESSED FILE CONTROL BLOCK. THE
                        ;NEW NAME IS CONTAINED IN THE LAST HALF OF THE
                        ;CURRENTLY ADDRESSED FILE CONROL BLOCK.  THE FILE
                        ;NAME AND TYPE ARE CHANGED, BUT THE REEL NUMBER
                        ;IS IGNORED.  THE USER NUMBER IS IDENTICAL
1016 CD540D             CALL CHECK$WRITE ;MAY BE WRITE PROTECTED
                        ;SEARCH UP TO THE EXTENT FIELD
1019 0E0CCD180F         MVI C,EXTNUM! CALL SEARCH
                        ;COPY POSITION 0
101E 2A430B7E           LHLD INFO! MOV A,M ;HL=.FCB(0), A=FCB(0)
1022 11100019           LXI D,DSKMAP! DAD D;HL=.FCB(DSKMAP)
1026 77                 MOV M,A ;FCB(DSKMAP)=FCB(0)
                        ;ASSUME THE SAME DISK DRIVE FOR NEW NAMED FILE
                        RENAME0:
1027 CDF50DC8                   CALL END$OF$DIR! RZ ;STOP AT END OF DIR
                                ;NOT END OF DIRECTORY, RENAME NEXT ELEMENT
102B CD440D                     CALL CHECK$RODIR ;MAY BE READ-ONLY FILE
102E 0E101E0CCD                 MVI C,DSKMAP! MVI E,EXTNUM! CALL COPY$DIR
                                ;ELEMENT RENAMED, MOVE TO NEXT
1035 CD2D0F                     CALL SEARCHN
1038 C32710                     JMP RENAME0
                ;
                INDICATORS:
                        ;SET FILE INDICATORS FOR CURRENT FCB
103B 0E0CCD180F         MVI C,EXTNUM! CALL SEARCH ;THROUGH FILE TYPE
                        INDIC0:
1040 CDF50DC8                   CALL END$OF$DIR! RZ ;STOP AT END OF DIR
                                ;NOT END OF DIRECTORY, CONTINUE TO CHANGE
1044 0E001E0C                   MVI C,0! MVI E,EXTNUM ;COPY NAME
1048 CD0110                     CALL COPY$DIR
104B CD2D0F                     CALL SEARCHN
104E C34010                     JMP INDIC0
                ;
                OPEN:
                        ;SEARCH FOR THE DIRECTORY ENTRY, COPY TO FCB
1051 0E0FCD180F         MVI C,NAMLEN! CALL SEARCH
1056 CDF50DC8           CALL END$OF$DIR! RZ ;RETURN WITH LRET=255 IF END
                        ;NOT END OF DIRECTORY, COPY FCB INFORMATION
                OPEN$COPY:
                        ;(REFERENCED BELOW TO COPY FCB INFO)
105A CDA60C7EF5         CALL GETEXTA! MOV A,M! PUSH PSW! PUSH H ;SAVE EXTENT#
1060 CD5E0DEB           CALL GETDPTRA! XCHG ;DE = .BUFF(DPTR)
1064 2A430B             LHLD INFO ;HL=.FCB(0)
1067 0E20               MVI C,NXTREC ;LENGTH OF MOVE OPERATION
1069 D5                 PUSH D ;SAVE .BUFF(DPTR)
106A CD4F0B             CALL MOVE ;FROM .BUFF(DPTR) TO .FCB(0)
                        ;NOTE THAT ENTIRE FCB IS COPIED, INCLUDING INDICATORS
106D CD780D             CALL SETFWF ;SETS FILE WRITE FLAG
1070 D1210C0019         POP D! LXI H,EXTNUM! DAD D ;HL=.BUFF(DPTR+EXTNUM)
```

```
1075 4E                 MOV C,M ;C = DIRECTORY EXTENT NUMBER
1076 210F0019           LXI H,RECCNT! DAD D ;HL=.BUFF(DPTR+RECCNT)
107A 46                 MOV B,M ;B HOLDS DIRECTORY RECORD COUNT
107B E1F177             POP H! POP PSW! MOV M,A ;RESTORE EXTENT NUMBER
                        ;HL = .USER EXTENT#, B = DIR REC CNT, C = DIR EXTENT#
                        ;IF USER EXT < DIR EXT THEN USER := 128 RECORDS
                        ;IF USER EXT = DIR EXT THEN USER := DIR RECORDS
                        ;IF USER EXT > DIR EXT THEN USER := 0 RECORDS
107E 79BE78                     MOV A,C! CMP M! MOV A,B ;READY DIR RECCNT
1081 CA8B10                     JZ OPEN$RCNT ;IF SAME, USER GETS DIR RECCNT
1084 3E00DA8B10                 MVI A,0! JC OPEN$RCNT ;USER IS LARGER
1089 3E80                       MVI A,128 ;DIRECTORY IS LARGER
                        OPEN$RCNT: ;A HAS RECORD COUNT TO FILL
108B 2A430B110F         LHLD INFO! LXI D,RECCNT! DAD D! MOV M,A
1093 C9                 RET
                ;
                MERGE$ERO:
                        ;HL = .FCB1(I), DE = .FCB2(I),
                        ;IF FCB1(I) = 0 THEN FCB1(I) := FCB2(I)
1094 7E23B62BC0         MOV A,M! INX H! ORA M! DCX H! RNZ ;RETURN IF = 0000
1099 1A771323           LDAX D! MOV M,A! INX D! INX H ;LOW BYTE COPIED
109D 1A771B2B           LDAX D! MOV M,A! DCX D! DCX H ;BACK TO INPUT FORM
10A1 C9                 RET
                ;
                CLOSE:
                        ;LOCATE THE DIRECTORY ELEMENT AND RE-WRITE IT
10A2 AF32450B32         XRA A! STA LRET! STA DCNT! STA DCNT+1    ;
10AC CD1E0DC0           CALL NOWRITE! RNZ ;SKIP CLOSE IF R/O DISK
                        ;CHECK FILE WRITE FLAG - 0 INDICATES WRITTEN
10B0 CD690D             CALL GETMODNUM ;FCB(MODNUM) IN A
10B3 E680C0             ANI FWFMSK! RNZ ;RETURN IF BIT REMAINS SET
10B6 0E0FCD180F         MVI C,NAMLEN! CALL SEARCH ;LOCATE FILE
10BB CDF50DC8           CALL END$OF$DIR! RZ ;RETURN IF NOT FOUND
                        ;MERGE THE DISK MAP AT INFO WITH THAT AT BUFF(DPTR)
10BF 011000CD5E         LXI B,DSKMAP! CALL GETDPTRA
10C5 09EB               DAD B! XCHG ;DE IS .BUFF(DPTR+16)
10C7 2A430B09           LHLD INFO! DAD B ;DE=.BUFF(DPTR+16), HL=.FCB(16)
10CB 0E10               MVI C,(FCBLEN-DSKMAP) ;LENGTH OF SINGLE BYTE DM
                        MERGE0:
10CD 3ADD15B7CA                 LDA SINGLE! ORA A! JZ MERGED ;SKIP TO DOUBLE
                                ;THIS IS A SINGLE BYTE MAP
                                ;IF FCB(I) = 0 THEN FCB(I) = BUFF(I)
                                ;IF BUFF(I) = 0 THEN BUFF(I) = FCB(I)
                                ;IF FCB(I) <> BUFF(I) THEN ERROR
10D4 7EB71AC2DB                 MOV A,M! ORA A! LDAX D! JNZ FCBNZERO
                                        ;FCB(I) = 0
10DA 77                                 MOV M,A ;FCB(I) = BUFF(I)
                                FCBNZERO:
10DB B7C2E110                   ORA A! JNZ BUFFNZERO
                                        ;BUFF(I) = 0
10DF 7E12                               MOV A,M! STAX D ;BUFF(I)=FCB(I)
                                BUFFNZERO:
10E1 BEC21F11                   CMP M! JNZ MERGERR ;FCB(I) = BUFF(I)?
10E5 C3FD10                     JMP DMSET ;IF MERGE OK
                        MERGED:
                                ;THIS IS A DOUBLE BYTE MERGE OPERATION
```

```
10E8 CD9410             CALL MERGEZERO ;BUFF = FCB IF BUFF 0000
10EB EBCD9410EB         XCHG! CALL MERGEZERO! XCHG ;FCB = BUFF IF FCB 0000
                        ;THEY SHOULD BE IDENTICAL AT THIS POINT
10F0 1ABEC21F11         LDAX D! CMP M! JNZ MERGERR ;LOW SAME?
10F5 1323               INX D! INX H ;TO HIGH BYTE
10F7 1ABEC21F11         LDAX D! CMP M! JNZ MERGERR ;HIGH SAME?
                        ;MERGE OPERATION OK FOR THIS PAIR
10FC 0D                 DCR C ;EXTRA COUNT FOR DOUBLE BYTE
                    DMSET:
10FD 1323               INX D! INX H ;TO NEXT BYTE POSITION
10FF 0DC2CD10           DCR C' JNZ MERGEO ;FOR MORE
                        ;END OF DISK MAP MERGE, CHECK RECORD COUNT
                        ;DE = .BUFF(DPTR)+32, HL = .FCB(32)
1103 01ECFF09EB         LXI B,-(FCBLEN-EXTNUM)! DAD B! XCHG! DAD B
                        ;DE = .FCB(EXTNUM), HL = .BUFF(DPTR+EXTNUM)
1109 1A                 LDAX D ;CURRENT USER EXTENT NUMBER
                        ;IF FCB(EXT) >= BUFF(FCB) THEN
                        ;BUFF(EXT) := FCB(EXT), BUFF(REC) := FCB(REC)
110A BEDA1711           CMP M! JC ENDMERGE
                        ;FCB EXTENT NUMBER >= DIR EXTENT NUMBER
110E 77                 MOV M,A ;BUFF(EXT) = FCB(EXT)
                        ;UPDATE DIRECTORY RECORD COUNT FIELD
110F 01030009EB         LXI B,(RECCNT-EXTNUM)! DAD B! XCHG! DAD B
                        ;DE=.BUFF(RECCNT), HL=.FCB(RECCNT)
1115 7E12               MOV A,M! STAX D ;BUFF(RECCNT)=FCB(RECCNT)
                    ENDMERGE:
1117 3EFF32D215         MVI A,TRUE! STA FCB$COPIED ;MARK AS COPIED
111C C31010             JMP SEEK$COPY ;OK TO "WRDIR" HERE - 1.4 COMPAT
                    ;               RET                             ;
                    MERGERR:
                        ;ELEMENTS DID NOT MERGE CORRECTLY
111F 21450B35           LXI H,LRET! DCR M ;=255 NON ZERO FLAG SET
1123 C9                 RET
                ;
               MAKE:
                        ;CREATE A NEW FILE BY CREATING A DIRECTORY ENTRY
                        ;THEN OPENING THE FILE
1124 CD540D             CALL CHECK$WRITE ;MAY BE WRITE PROTECTED
1127 2A430BE5           LHLD INFO! PUSH H ;SAVE FCB ADDRESS, LOOK FOR E5
112B 21AC152243         LXI H,EFCB! SHLD INFO ;INFO = .EMPTY
1131 0E01CD180F         MVI C,1! CALL SEARCH ;LENGTH 1 MATCH ON EMPTY ENTRY
1136 CDF50D             CALL END$OF$DIR ;ZERO FLAG SET IF NO SPACE
1139 E1                 POP H ;RECALL INFO ADDRESS
113A 22430B             SHLD INFO ;IN CASE WE RETURN HERE
113D C8                 RZ ;RETURN WITH ERROR CONDITION 255 IF NOT FOUND
113E EB                 XCHG ;DE = INFO ADDRESS
                        :CLEAR THE REMAINDER OF THE FCB
113F 210F0019           LXI H,NAMLEN! DAD D ;HL=.FCB(NAMLEN)
1143 0E11               MVI C,FCBLEN-NAMLEN ;NUMBER OF BYTES TO FILL
1145 AF                 XRA A ;CLEAR ACCUMULATOR TO 00 FOR FILL
                        MAKEO:
1146 77230DC246             MOV M,A! INX H! DCR C! JNZ MAKEO
114C 210D0019           LXI H,UBYTES! DAD D ;HL = .FCB(UBYTES)
1150 77                 MOV M,A ;FCB(UBYTES) = 0
1151 CD8C0D             CALL SETCDR ;MAY HAVE EXTENDED THE DIRECTORY
                        ;NOW COPY ENTRY TO THE DIRECTORY
```

```
1154 CDFD0F             CALL COPY$FCB
                        ;AND SET THE FILE WRITE FLAG TO "1"
1157 C3780D             JMP SETFWF
                        ;RET
                ;
                OPEN$REEL:
                        ;CLOSE THE CURRENT EXTENT, AND OPEN THE NEXT ONE
                        ;IF POSSIBLE.  RMF IS TRUE IF IN READ MODE
115A AF32D215                   XRA A! STA FCB$COPIED ;SET TRUE IF ACTUALLY COPIED
115E CDA210                     CALL CLOSE ;CLOSE CURRENT EXTENT
                                ;LRET REMAINS AT ENDDIR IF WE CANNOT OPEN THE NEXT EXT
1161 CDF50DC8                   CALL END$OF$DIR! RZ ;RETURN IF END
                        ;INCREMENT EXTENT NUMBER
1165 2A430B010C         LHLD INFO! LXI B,EXTNUM! DAD B ;HL=.FCB(EXTNUM)
116C 7E3CE61F77         MOV A,M! INR A! ANI MAXEXT! MOV M,A ;FCB(EXTNUM)=++1
1171 CA8311             JZ OPEN$MOD ;MOVE TO NEXT MODULE.IF ZERO
                        ;MAY BE IN THE SAME EXTENT GROUP
1174 473AC515A0         MOV B,A! LDA EXTMSK! ANA B
                        ;IF RESULT IS ZERO, THEN NOT IN THE SAME GROUP
1179 21D215             LXI H,FCB$COPIED ;TRUE IF THE FCB WAS COPIED TO DIRECTORY
117C A6                 ANA M ;PRODUCES A 00 IN ACCUMULATOR IF NOT WRITTEN
117D CA8E11             JZ OPEN$REEL0 ;GO TO NEXT PHYSICAL EXTENT
                        ;RESULT IS NON ZERO, SO WE MUST BE IN SAME LOGICAL EXT
1180 C3AC11             JMP OPEN$REEL1 ;TO COPY FCB INFORMATION
                        OPEN$MOD:
                                ;EXTENT NUMBER OVERFLOW, GO TO NEXT MODULE
1183 01020009                   LXI B,(MODNUM-EXTNUM)! DAD B ;HL=.FCB(MODNUM)
1187 34                         INR M ;FCB(MODNUM)=++1
                                ;MODULE NUMBER INCREMENTED, CHECK FOR OVERFLOW
1188 7EE60F                     MOV A,M! ANI MAXMOD ;MASK HIGH ORDER BITS
118B CAB611                     JZ OPEN$R$ERR ;CANNOT OVERFLOW TO ZERO
                                ;OTHERWISE, OK TO CONTINUE WITH NEW MODULE
                        OPEN$REEL0:
118E 0E0FCD180F                 MVI C,NAMLEN! CALL SEARCH ;NEXT EXTENT FOUND?
1193 CDF50DC2AC                 CALL END$OF$DIR! JNZ OPEN$REEL1
                                        ;END OF FILE ENCOUNTERED
1199 3AD3153C                           LDA RMF! INR A ;0FFH BECOMES 00 IF READ
119D CAB611                             JZ OPEN$R$ERR ;SETS LRET = 1
                                        ;TRY TO EXTEND THE CURRENT FILE
11A0 CD2411                             CALL MAKE
                                        ;CANNOT BE END OF DIRECTORY
11A3 CDF50D                             CALL END$OF$DIR
11A6 CAB611                             JZ OPEN$R$ERR ;WITH LRET = 1
11A9 C3AF11                             JMP OPEN$REEL2
                                OPEN$REEL1:
                                        ;NOT END OF FILE, OPEN
11AC CD5A10                             CALL OPEN$COPY
                                OPEN$REEL2:
11AF CDBB0C                             CALL GETFCB ;SET PARAMETERS
11B2 AFC3010B                           XRA A! JMP STA$RET ;LRET = 0    ;
                ;                       RET ;WITH LRET = 0
                        OPEN$R$ERR:
                                ;CANNOT MOVE TO NEXT EXTENT OF THIS FILE
11B6 CD050B                     CALL SETLRET1 ;LRET = 1
11B9 C3780D                     JMP SETFWF ;ENSURE THAT IT WILL NOT BE CHOSEN
                                ;RET
```

```
                ;
                SEQDISKREAD:
                        ;SEQUENTIAL DISK READ OPERATION
11BC 3E0132D515         MVI A,1! STA SEQIO
                        ;DROP THROUGH TO DISKREAD
                ;
                DISKREAD:       ;(MAY ENTER FROM SEQDISKREAD)
11C1 3EFF32D315         MVI A,TRUE! STA RMF ;READ MODE FLAG = TRUE (OPEN$REEL)
                        ;READ THE NEXT RECORD FROM THE CURRENT FCB
11C6 CDBB0C             CALL GETFCB ;SETS PARAMETERS FOR THE READ
11C9 3AE31521E1         LDA VRECORD! LXI H,RCOUNT! CMP M ;VRECORD-RCOUNT
                        ;SKIP IF RCOUNT > VRECORD
11D0 DAE611             JC RECORDOK
                                ;NOT ENOUGH RECORDS IN THE EXTENT
                                ;RECORD COUNT MUST BE 128 TO CONTINUE
11D3 FE80                       CPI 128 ;VRECORD = 128?
11D5 C2FB11                     JNZ DISKEOF ;SKIP IF VRECORD<>128
11D8 CD5A11                     CALL OPEN$REEL ;GO TO NEXT EXTENT IF SO
11DB AF32E315                   XRA A! STA VRECORD ;VRECORD=00
                                ;NOW CHECK FOR OPEN OK
11DF 3A450BB7C2                 LDA LRET! ORA A! JNZ DISKEOF ;STOP AT EOF
                        RECORDOK:
                                ;ARRIVE WITH FCB ADDRESSING A RECORD TO READ
11E6 CD770C                     CALL INDEX
                                ;ERROR 2 IF READING UNWRITTEN DATA
                                ;(RETURNS 1 TO BE COMPATIBLE WITH 1.4)
11E9 CD840C                     CALL ALLOCATED ;ARECORD=0000?
11EC CAFB11                     JZ DISKEOF
                                ;RECORD HAS BEEN ALLOCATED, READ IT
11EF CD8A0C                     CALL ATRAN ;ARECORD NOW A DISK ADDRESS
11F2 CDD10B                     CALL SEEK ;TO PROPER TRACK,SECTOR
11F5 CDB20B                     CALL RDBUFF ;TO DMA ADDRESS
11F8 C3D20C                     JMP SETFCB ;REPLACE PARAMETER
                ;               RET                                     ;
                        DISKEOF:
11FB C3050B                     JMP SETLRET1 ;LRET = 1
                                ;RET
                ;
                SEQDISKWRITE:
                        ;SEQUENTIAL DISK WRITE
11FE 3E0132D515         MVI A,1! STA SEQIO
                        ;DROP THROUGH TO DISKWRITE
                ;
                DISKWRITE:      ;(MAY ENTER HERE FROM SEQDISKWRITE ABOVE)
1203 3E0032D315         MVI A,FALSE! STA RMF ;READ MODE FLAG
                        ;WRITE RECORD TO CURRENTLY SELECTED FILE
1208 CD540D             CALL CHECK$WRITE ;IN CASE WRITE PROTECTED
120B 2A430B             LHLD INFO ;HL = .FCB(0)
120E CD470D             CALL CHECK$ROFILE ;MAY BE A READ-ONLY FILE
1211 CDBB0C             CALL GETFCB ;TO SET LOCAL PARAMETERS
1214 3AE315FE80         LDA VRECORD! CPI LSTREC+1 ;VRECORD-128
                        ;SKIP IF VRECORD > LSTREC
                                ;VRECORD = 128, CANNOT OPEN NEXT EXTENT
1219 D2050B                     JNC SETLRET1    ;LRET=1                 ;
                        DISKWR0:
                        ;CAN WRITE THE NEXT RECORD, SO CONTINUE
```

```
121C CD770C             CALL INDEX
121F CD840C             CALL ALLOCATED
1222 0E00               MVI C,0 ;MARKED AS NORMAL WRITE OPERATION FOR WRBUFF
1224 C26E12             JNZ DISKWR1
                                ;NOT ALLOCATED
                                ;THE ARGUMENT TO GETBLOCK IS THE STARTING
                                ;POSITION FOR THE DISK SEARCH, AND SHOULD BE
                                ;THE LAST ALLOCATED BLOCK FOR THIS FILE, OR
                                ;THE VALUE 0 IF NO SPACE HAS BEEN ALLOCATED
1227 CD3E0C                     CALL DM$POSITION
122A 32D715                     STA DMINX ;SAVE FOR LATER
122D 010000                     LXI B,0000H ;MAY USE BLOCK ZERO
1230 B7CA3B12                   ORA A! JZ NOPBLOCK ;SKIP IF NO PREVIOUS BLOCK
                                        ;PREVIOUS BLOCK EXISTS AT A
1234 4F0B                               MOV C,A! DCX B ;PREVIOUS BLOCK # IN BC
1236 CD5E0C                             CALL GETDM ;PREVIOUS BLOCK # TO HL
1239 444D                               MOV B,H! MOV C,L ;BC=PREV BLOCK#
                                NOPBLOCK:
                                        ;BC = 0000, OR PREVIOUS BLOCK #
123B CDBE0F                             CALL GET$BLOCK ;BLOCK # TO HL
                                ;ARRIVE HERE WITH BLOCK# OR ZERO
123E 7DB4C24812                 MOV A,L! ORA H! JNZ BLOCKOK
                                        ;CANNOT FIND A BLOCK TO ALLOCATE
1243 3E02C3010B                         MVI A,2! JMP STA$RET    ;LRET=2
                                BLOCKOK:
                                ;ALLOCATED BLOCK NUMBER IS IN HL
1248 22E515                     SHLD ARECORD
124B EB                         XCHG ;BLOCK NUMBER TO DE
124C 2A430B0110                 LHLD INFO! LXI B,DSKMAP! DAD B ;HL=.FCB(DSKMAP)
1253 3ADD15B7                   LDA SINGLE! ORA A ;SET FLAGS FOR SINGLE BYTE DM
1257 3AD715                     LDA DMINX ;RECALL DM INDEX
125A CA6412                     JZ ALLOCWD ;SKIP IF ALLOCATING WORD
                                        ;ALLOCATING A BYTE VALUE
125D CD640D73                           CALL ADDH! MOV M,E ;SINGLE BYTE ALLOC
1261 C36C12                             JMP DISKWRU ;TO CONTINUE
                                ALLOCWD:
                                ;ALLOCATE A WORD VALUE
1264 4F0600                             MOV C,A! MVI B,0 ;DOUBLE(DMINX)
1267 0909                               DAD B! DAD B ;HL=.FCB(DMINX*2)
1269 732372                             MOV M,E! INX H! MOV M,D ;DOUBLE WD
                                DISKWRU:
                                ;DISK WRITE TO PREVIOUSLY UNALLOCATED BLOCK
126C 0E02                       MVI C,2 ;MARKED AS UNALLOCATED WRITE
                        DISKWR1:
                        ;CONTINUE THE WRITE OPERATION OF NO ALLOCATION ERROR
                        ;C = 0 IF NORMAL WRITE, 2 IF TO PREV UNALLOC BLOCK
126E 3A450BB7C0         LDA LRET! ORA A! RNZ ;STOP IF NON ZERO RETURNED VALUE
1273 C5                 PUSH B ;SAVE WRITE FLAG
1274 CD8A0C             CALL ATRAN ;ARECORD SET
1277 3AD5153D3D                 LDA SEQIO! DCR A! DCR A! JNZ DISKWR11   ;
127F C1C5793D3D                 POP B! PUSH B! MOV A,C! DCR A! DCR A    ;
1284 C2BB12                     JNZ DISKWR11            ;OLD ALLOCATION
1287 E5                         PUSH H          ;ARECORD IN HL, RET FROM ATRAN
1288 2AB91557                   LHLD BUFFA! MOV D,A     ;ZERO BUFFA & FILL
128C 772314F28C         FILL0:  MOV M,A! INX H! INR D! JP FILL0
1292 CDE00D2AE7                 CALL SETDIR! LHLD ARECORD1
```

```
1298 0E02                       MVI C,2                                 ;
129A 22E515C5CD         FILL1:  SHLD ARECORD! PUSH B! CALL SEEK! POP B  ;
12A2 CDB80B                     CALL WRBUFF     ;WRITE FILL RECORD      ;
12A5 2AE515                     LHLD ARECORD!   ;RESTORE LAST RECORD
12A8 0E00                       MVI C,0         ;CHANGE  ALLOCATE FLAG
12AA 3AC41547A5                 LDA BLKMSK! MOV B,A! ANA L! CMP B!INX H ;
12B1 C29A12                     JNZ FILL1       ;CONT UNTIL CLUSTER IS ZEROED
12B4 E122E515CD                 POP H! SHLD ARECORD! CALL SETDATA
                        DISKWR11:                                       ;
12BB CDD10B             CALL SEEK ;TO PROPER FILE POSITION
12BE C1C5               POP B! PUSH B ;RESTORE/SAVE WRITE FLAG (C=2 IF NEW BLOCK)
12C0 CDB80B             CALL WRBUFF ;WRITTEN TO DISK
12C3 C1                 POP B ;C = 2 IF A NEW BLOCK WAS ALLOCATED, 0 IF NOT
                        ;INCREMENT RECORD COUNT IF RCOUNT<=VRECORD
12C4 3AE31521E1         LDA VRECORD! LXI H,RCOUNT! CMP M ;VRECORD-RCOUNT
12CB DAD212             JC DISKWR2
                                ;RCOUNT <= VRECORD
12CE 7734                       MOV M,A! INR M ;RCOUNT = VRECORD+1
12D0 0E02                       MVI C,2 ;MARK AS RECORD COUNT INCREMENTED
                        DISKWR2:
                        ;A HAS VRECORD, C=2 IF NEW BLOCK OR NEW RECORD#
12D2 0D0DC2DF12         DCR C! DCR C! JNZ NOUPDATE      NOP ! NOP! LD HL,0
12D7 F5                         PUSH PSW ;SAVE VRECORD VALUE
12D8 CD690D                     CALL GETMODNUM ;HL=.FCB(MODNUM), A=FCB(MODNUM)
                                ;RESET THE FILE WRITE FLAG TO MARK AS WRITTEN FCB
12DB E67F                       ANI (NOT FWFMSK) AND 0FFH ;BIT RESET
12DD 77                         MOV M,A ;FCB(MODNUM) = FCB(MODNUM) AND 7FH
12DE F1                         POP PSW ;RESTORE VRECORD
                        NOUPDATE:
                        ;CHECK FOR END OF EXTENT, IF FOUND ATTEMPT TO OPEN
                        ;NEXT EXTENT IN PREPARATION FOR NEXT WRITE
12DF FE7F               CPI LSTREC ;VRECORD=LSTREC?
12E1 C20013             JNZ DISKWR3 ;SKIP IF NOT
                        ;MAY BE RANDOM ACCESS WRITE, IF SO WE ARE DONE
                                                        ;CHANGE NEXT
12E4 3AD515FE01         LDA SEQIO! CPI 1! JNZ DISKWR3 ;SKIP NEXT EXTENT OPEN OP
                                ;UPDATE CURRENT FCB BEFORE GOING TO NEXT EXTENT
12EC CDD20C                     CALL SETFCB
12EF CD5A11                     CALL OPEN$REEL ;RMF=FALSE
                                ;VRECORD REMAINS AT LSTREC CAUSING EOF IF
                                ;NO MORE DIRECTORY SPACE IS AVAILABLE
12F2 21450B7EB7                 LXI H,LRET! MOV A,M! ORA A! JNZ NOSPACE
                                        ;SPACE AVAILABLE, SET VRECORD=255
12FA 3D32E315                           DCR A! STA VRECORD ;GOES TO 00 NEXT TIME
                                NOSPACE:
12FE 3600                       MVI M,0 ;LRET = 00 FOR RETURNED VALUE
                        DISKWR3:
1300 C3D20C             JMP SETFCB ;REPLACE PARAMETERS
                        ;RET
                    ;
                    RSEEK:
                        ;RANDOM ACCESS SEEK OPERATION, C=0FFH IF READ MODE
                        ;FCB IS ASSUMED TO ADDRESS AN ACTIVE FILE CONTROL BLOCK
                        ;(MODNUM HAS BEEN SET TO 1100$0000B IF PREVIOUS BAD SEEK)
1303 AF32D515           XRA A! STA SEQIO ;MARKED AS RANDOM ACCESS OPERATION
                    RSEEK1:
```

```
1307 C5              PUSH B ;SAVE R/W FLAG
1308 2A430BEB        LHLD INFO! XCHG ;DE WILL HOLD BASE OF FCB
130C 21210019                LXI H,RANREC! DAD D ;HL=.FCB(RANREC)
1310 7EE67FF5                MOV A,M! ANI 7FH! PUSH PSW ;RECORD NUMBER
1314 7E17                    MOV A,M! RAL ;CY=LSB OF EXTENT#
1316 237E17E61F              INX H! MOV A,M! RAL! ANI 11111B ;A=EXT#
131B 4F                      MOV C,A ;C HOLDS EXTENT NUMBER, RECORD STACKED
131C 7E1F1F1F1F              MOV A,M! RAR! RAR! RAR! RAR! ANI 1111B ;MOD#
1323 47                      MOV B,A ;B HOLDS MODULE#, C HOLDS EXT#
1324 F1                      POP PSW ;RECALL SOUGHT RECORD #
                             ;CHECK TO INSURE THAT HIGH BYTE OF RAN REC = 00
1325 236E                    INX H! MOV L,M ;L=HIGH BYTE (MUST BE 00)
1327 2C2D2E06                INR L! DCR L! MVI L,6 ;ZERO FLAG, L=6
                             ;PRODUCE ERROR 6, SEEK PAST PHYSICAL EOD
132B C28B13                  JNZ SEEKERR
                             ;OTHERWISE, HIGH BYTE = 0, A = SOUGHT RECORD
132E 21200019                LXI H,NXTREC! DAD D ;HL = .FCB(NXTREC)
1332 77                      MOV M,A ;SOUGHT REC# STORED AWAY
                     ;ARRIVE HERE WITH B=MOD#, C=EXT#, DE=.FCB, REC STORED
                     ;THE R/W FLAG IS STILL STACKED.  COMPARE FCB VALUES
1333 210C001979              LXI H,EXTNUM! DAD D! MOV A,C ;A=SEEK EXT#
1338 96C24713                SUB M! JNZ RANCLOSE ;TESTS FOR = EXTENTS
                             ;EXTENTS MATCH, CHECK MOD#
133C 210E001978              LXI H,MODNUM! DAD D! MOV A,B ;B=SEEK MOD#
                             ;COULD BE OVERFLOW AT EOF, PRODUCING MODULE#
                             ;OF 90H OR 10H, SO COMPARE ALL BUT FWF
1341 96E67FCA7F              SUB M! ANI 7FH! JZ SEEKOK ;SAME?
                     RANCLOSE:
1347 C5D5                    PUSH B! PUSH D ;SAVE SEEK MOD#,EXT#, .FCB
1349 CDA210                  CALL CLOSE ;CURRENT EXTENT CLOSED
134C D1C1                    POP D! POP B ;RECALL PARAMETERS AND FILL
134E 2E03                    MVI L,3 ;CANNOT CLOSE ERROR #3
1350 3A450B3CCA              LDA LRET! INR A! JZ BADSEEK
1357 210C001971              LXI H,EXTNUM! DAD D! MOV M,C ;FCB(EXTNUM)=EXT#
135C 210E001970              LXI H,MODNUM! DAD D! MOV M,B ;FCB(MODNUM)=MOD#
1361 CD5110                  CALL OPEN ;IS THE FILE PRESENT?
1364 3A450B3CC2              LDA LRET! INR A! JNZ SEEKOK ;OPEN SUCCESSFUL?
                             ;CANNOT OPEN THE FILE, READ MODE?
136B C1                      POP B ;R/W FLAG TO C (=0FFH IF READ)
136C C5                      PUSH B ;EVERYONE EXPECTS THIS ITEM STACKED
136D 2E04                    MVI L,4 ;SEEK TO UNWRITTEN EXTENT #4
136F 0C                      INR C ;BECOMES 00 IF READ OPERATION
1370 CA8413                  JZ BADSEEK ;SKIP TO ERROR IF READ OPERATION
                             ;WRITE OPERATION, MAKE NEW EXTENT
1373 CD2411                  CALL MAKE
1376 2E05                    MVI L,5 ;CANNOT CREATE NEW EXTENT #5
1378 3A450B3CCA              LDA LRET! INR A! JZ BADSEEK ;NO DIR SPACE
                             ;FILE MAKE OPERATION SUCCESSFUL
                     SEEKOK:
137F C1                      POP B ;DISCARD R/W FLAG
1380 AFC3010B                XRA A! JMP STA$RET      ;WITH ZERO SET
                     BADSEEK:
                             ;FCB NO LONGER CONTAINS A VALID FCB, MARK
                             ;WITH 1100$000B IN MODNUM FIELD SO THAT IT
                             ;APPEARS AS OVERFLOW WITH FILE WRITE FLAG SET
1384 E5                      PUSH H ;SAVE ERROR FLAG
```

```
1385 CD690D                     CALL GETMODNUM ;HL = .MODNUM
1388 36C0                       MVI M,1100$0000B
138A E1                         POP H ;AND DROP THROUGH
                        SEEKERR:
138B C1                         POP B ;DISCARD R/W FLAG
138C 7D32450B                   MOV A,L! STA LRET ;LRET=#, NONZERO
                                ;SETFWF RETURNS NON-ZERO ACCUMULATOR FOR ERR
1390 C3780D                     JMP SETFWF ;FLAG SET, SO SUBSEQUENT CLOSE OK
                                ;RET
                ;
                RANDISKREAD:
                        ;RANDOM DISK READ OPERATION
1393 0EFF               MVI C,TRUE ;MARKED AS READ OPERATION
1395 CD0313             CALL RSEEK
1398 CCC111             CZ DISKREAD ;IF SEEK SUCCESSFUL
139B C9                 RET
                ;
                RANDISKWRITE:
                        ;RANDOM DISK WRITE OPERATION
139C 0E00               MVI C,FALSE ;MARKED AS WRITE OPERATION
139E CD0313             CALL RSEEK
13A1 CC0312             CZ DISKWRITE ;IF SEEK SUCCESSFUL
13A4 C9                 RET
                ;
                COMPUTE$RR:
                        ;COMPUTE RANDOM RECORD POSITION FOR GETFILESIZE/SETRANDOM
13A5 EB19               XCHG! DAD D
                        ;DE=.BUF(DPTR) OR .FCB(0), HL = .F(NXTREC/RECCNT)
13A7 4E0600             MOV C,M! MVI B,0 ;BC = 0000 0000 ?RRR RRRR
13AA 210C00197E         LXI H,EXTNUM! DAD D! MOV A,M! RRC! ANI 80H ;A=E000 0000
13B2 814F3E0088         ADD C! MOV C,A! MVI A,0! ADC B! MOV B,A
                        ;BC = 0000 000? ERRRR RRRR
13B8 7E0FE60F80         MOV A,M! RRC! ANI 0FH! ADD B! MOV B,A
                        ;BC = 000? EEEE ERRRR RRRR
13BE 210E00197E         LXI H,MODNUM! DAD D! MOV A,M ;A=XXX? MMMM
13C3 87878787           ADD A! ADD A! ADD A! ADD A ;CY=? A=MMMM 0000
13C7 F58047             PUSH PSW! ADD B! MOV B,A
                        ;CY=?, BC = MMMM EEEE ERRR RRRR
13CA F5                 PUSH PSW ;POSSIBLE SECOND CARRY
13CB E1                 POP H ;CY = LSB OF L
13CC 7D                 MOV A,L ;CY = LSB OF A
13CD E1                 POP H ;CY = LSB OF L
13CE B5                 ORA L ;CY/CY = LSB OF A
13CF E601               ANI 1 ;A = 0000 000? POSSIBLE CARRY-OUT
13D1 C9                 RET
                ;
                GETFILESIZE:
                        ;COMPUTE LOGICAL FILE SIZE FOR CURRENT FCB
13D2 0E0C               MVI C,EXTNUM
13D4 CD180F             CALL SEARCH
                        ;ZERO THE RECEIVING RANREC FIELD
13D7 2A430B1121         LHLD INFO! LXI D,RANREC! DAD D! PUSH H ;SAVE POSITION
13DF 7223722372         MOV M,D! INX H! MOV M,D! INX H! MOV M,D;=00 00 00
                        GETSIZE:
13E4 CDF50D                     CALL END$OF$DIR
13E7 CA0C14                     JZ SETSIZE
```

```
                                ;CURRENT FCB ADDRESSED BY DPTR
13EA CD5E0D110F                 CALL GETDPTRA! LXI D,RECCNT ;READY FOR COMPUTE SIZE
13F0 CDA513                     CALL COMPUTE$RR
                                ;A=0000 000? BC = MMMM EEEE ERRR RRRR
                                ;COMPARE WITH MEMORY, LARGER?
13F3 E1E5                       POP H! PUSH H ;RECALL, REPLACE .FCB(RANREC)
13F5 5F                         MOV E,A ;SAVE CY
13F6 799623                     MOV A,C! SUB M! INX H ;LS BYTE
13F9 789E23                     MOV A,B! SBB M! INX H ;MIDDLE BYTE
13FC 7B9E                       MOV A,E! SBB M ;CARRY IF .FCB(RANREC) > DIRECTORY
13FE DA0614                     JC GETNEXTSIZE ;FOR ANOTHER TRY
                                ;FCB IS LESS OR EQUAL, FILL FROM DIRECTORY
1401 732B702B71                 MOV M,E! DCX H! MOV M,B! DCX H! MOV M,C
                        GETNEXTSIZE:
1406 CD2D0F                     CALL SEARCHN
1409 C3E413                     JMP GETSIZE
                        SETSIZE:
140C E1                 POP H ;DISCARD .FCB(RANREC)
140D C9                 RET
                ;
                SETRANDOM:
                        ;SET RANDOM RECORD FROM THE CURRENT FILE CONTROL BLOCK
140E 2A430B1120         LHLD INFO! LXI D,NXTREC ;READY PARAMS FOR COMPUTESIZE
1414 CDA513             CALL COMPUTE$RR ;DE=INFO, A=CY, BC=MMMM EEEE ERRR RRRR
1417 21210019           LXI H,RANREC! DAD D ;HL = .FCB(RANREC)
141B 7123702377         MOV M,C! INX H! MOV M,B! INX H! MOV M,A ;TO RANREC
1420 C9                 RET
                ;
                SELECT:
                        ;SELECT DISK INFO FOR SUBSEQUENT INPUT OR OUTPUT OPS
1421 2AAF153A42         LHLD DLOG! LDA CURDSK! MOV C,A! CALL HLROTR
142B E5EB               PUSH H! XCHG ;SAVE IT FOR TEST BELOW, SEND TO SELDSK
142D CD590BE1           CALL SELECTDISK! POP H ;RECALL DLOG VECTOR
1431 CC470B             CZ SEL$ERROR ;RETURNS TRUE IF SELECT OK
                        ;IS THE DISK LOGGED IN?
1434 7D1FD8             MOV A,L! RAR! RC ;RETURN IF BIT IS SET
                        ;DISK NOT LOGGED IN, SET BIT AND INITIALIZE
1437 2AAF154D44         LHLD DLOG! MOV C,L! MOV B,H ;CALL READY
143C CD0B0D22AF         CALL SET$CDISK! SHLD DLOG ;DLOG=SET$CDISK(DLOG)
1442 C3A30E             JMP INITIALIZE
                        ;RET
                ;
                CURSELECT:
1445 3AD6152142         LDA LINFO! LXI H,CURDSK! CMP M! RZ ;SKIP IF LINFO=CURDSK
144D 77                 MOV M,A ;CURDSK=INFO
144E C32114             JMP SELECT
                        ;RET
                ;
                RESELECT:
                        ;CHECK CURRENT FCB TO SEE IF RESELECTION NECESSARY
1451 3EFF32DE15         MVI A,TRUE! STA RESEL ;MARK POSSIBLE RESELECT
1456 2A430B7E           LHLD INFO! MOV A,M ;DRIVE SELECT CODE
145A E61F               ANI 1$1111B ;NON ZERO IS AUTO DRIVE SELECT
145C 3D                 DCR A ;DRIVE CODE NORMALIZED TO 0..30 OR 255
145D 32D615             STA LINFO ;SAVE DRIVE CODE
1460 FE1ED27514         CPI 30! JNC NOSELECT
```

```
                        ;AUTO SELECT FUNCTION, SAVE CURDSK
1465 3A420B32DF         LDA CURDSK! STA OLDDSK ;OLDDSK=CURDSK
146B 7E32E015           MOV A,M! STA FCBDSK ;SAVE DRIVE CODE
146F E6E077             ANI 1110$0000B! MOV M,A ;PRESERVE HI BITS
1472 CD4514             CALL CURSELECT
                    NOSELECT:
                        ;SET USER CODE
1475 3A410B             LDA USRCODE ;0...31
1478 2A430BB677         LHLD INFO! ORA M! MOV M,A
147D C9                 RET
                ;
                ;       INDIVIDUAL FUNCTION HANDLERS
                FUNC12:
                        ;RETURN VERSION NUMBER
147E 3E22C3010B         MVI A,DVERS! JMP STA$RET ;LRET = DVERS (HIGH = 00)
                ;       RET ;JMP GOBACK
                ;
                FUNC13:
                        ;RESET DISK SYSTEM - INITIALIZE TO DISK 0
1483 21000022AD         LXI H,0! SHLD RODSK! SHLD DLOG
148C AF32420B           XRA A! STA CURDSK ;NOTE THAT USRCODE REMAINS UNCHANGED
1490 21800022B1         LXI H,TBUFF! SHLD DMAAD ;DMAAD = TBUFF
1496 CDDA0D             CALL SETDATA ;TO DATA DMA ADDRESS
1499 C32114             JMP SELECT
                        ;RET ;JMP GOBACK
                ;
1445 =          FUNC14: EQU     CURSELECT               ;
                        ;SELECT DISK INFO
                        ;RET ;JMP GOBACK
                ;
                FUNC15:
                        ;OPEN FILE
149C CD720D             CALL CLRMODNUM ;CLEAR THE MODULE NUMBER
149F CD5114             CALL RESELECT
14A2 C35110             JMP OPEN
                        ;RET ;JMP GOBACK
                ;
                FUNC16:
                        ;CLOSE FILE
14A5 CD5114             CALL RESELECT
14A8 C3A210             JMP CLOSE
                        ;RET ;JMP GOBACK
                ;
                FUNC17:
                        ;SEARCH FOR FIRST OCCURRENCE OF A FILE
14AB 0E00               MVI C,0 ;LENGTH ASSUMING '?' TRUE
14AD EB                 XCHG            ;WAS LHLD INFO
14AE 7EFE3F             MOV A,M! CPI '?' ;NO RESELECT IF ?
14B1 CAC214             JZ QSELECT ;SKIP RESELECT IF SO
                        ;NORMAL SEARCH
14B4 CDA60C7EFE         CALL GETEXTA! MOV A,M! CPI '?'  ;
14BA C4720D             CNZ CLRMODNUM ;MODULE NUMBER ZEROED
14BD CD5114             CALL RESELECT
14C0 0E0F               MVI C,NAMLEN
                    QSELECT:
14C2 CD180F             CALL SEARCH
```

```
14C5 C3E90D            JMP DIR$TO$USER ;COPY DIRECTORY ENTRY TO USER
                       ;RET ;JMP GOBACK
                ;
                FUNC18:
                       ;SEARCH FOR NEXT OCCURRENCE OF A FILE NAME
14C8 2AD9152243        LHLD SEARCHA! SHLD INFO
14CE CD5114CD2D        CALL RESELECT! CALL SEARCHN
14D4 C3E90D            JMP DIR$TO$USER ;COPY DIRECTORY ENTRY TO USER
                       ;RET ;JMP GOBACK
                ;
                FUNC19:
                       ;DELETE A FILE
14D7 CD5114            CALL RESELECT
14DA CD9C0F            CALL DELETE
14DD C3010F            JMP COPY$DIRLOC
                       ;RET ;JMP GOBACK
                ;
                FUNC20:
                       ;READ A FILE
14E0 CD5114            CALL RESELECT
14E3 C3BC11            JMP SEQDISKREAD                    ;
                        ;JMP GOBACK .
                ;
                FUNC21:
                       ;WRITE A FILE
14E6 CD5114            CALL RESELECT
14E9 C3FE11            JMP SEQDISKWRITE                   ;
                        ;JMP GOBACK
                ;
                FUNC22:
                       ;MAKE A FILE
14EC CD720D            CALL CLRMODNUM
14EF CD5114            CALL RESELECT
14F2 C32411            JMP MAKE
                       ;RET ;JMP GOBACK
                ;
                FUNC23:
                       ;RENAME A FILE
14F5 CD5114            CALL RESELECT
14F8 CD1610            CALL RENAME
14FB C3010F            JMP COPY$DIRLOC
                       ;RET ;JMP GOBACK
                ;
                FUNC24:
                       ;RETURN THE LOGIN VECTOR
14FE 2AAF15C329        LHLD DLOG! JMP STHL$RET            ;
                ;      RET ;JMP GOBACK
                ;
                FUNC25:
                       ;RETURN SELECTED DISK NUMBER
1504 3A420BC301        LDA CURDSK! JMP STA$RET            ;
                ;      RET ;JMP GOBACK
                ;
                FUNC26:
                       ;SET THE SUBSEQUENT DMA ADDRESS TO INFO
150A EB                XCHG            ;WAS LHLD INFO
```

```
150B 22B115             SHLD DMAAD ;DMAAD = INFO
150E C3DA0D             JMP SETDATA ;TO DATA DMA ADDRESS
                        ;RET ;JMP GOBACK
                ;
                FUNC27:
                        ;RETURN THE LOGIN VECTOR ADDRESS
1511 2ABF15C329         LHLD ALLOCA! JMP STHL$RET               ;
                ;       RET ;JMP GOBACK
                ;
0D2C =          FUNC28: EQU     SET$RO                          ;
                        ;WRITE PROTECT CURRENT DISK
                        ;RET ;JMP GOBACK
                ;
                FUNC29:
                        ;RETURN R/O BIT VECTOR
1517 2AAD15C329         LHLD RODSK! JMP STHL$RET                ;
                ;       RET ;JMP GOBACK
                ;
                FUNC30:
                        ;SET FILE INDICATORS
151D CD5114             CALL RESELECT
1520 CD3B10             CALL INDICATORS
1523 C3010F             JMP COPY$DIRLOC ;LRET=DIRLOC
                        ;RET ;JMP GOBACK
                ;
                FUNC31:
                        ;RETURN ADDRESS OF DISK PARAMETER BLOCK
1526 2ABB15             LHLD DPBADDR
                STHL$RET:
1529 22450B             SHLD ARET
152C C9                 RET ;JMP GOBACK
                FUNC32:
                        ;SET USER CODE
152D 3AD615FEFF         LDA LINFO! CPI 0FFH! JNZ SETUSRCODE
                                ;INTERROGATE USER CODE INSTEAD
1535 3A410BC301                 LDA USRCODE! JMP STA$RET ;LRET=USRCODE
                ;               RET ;JMP GOBACK
                        SETUSRCODE:
153B E61F32410B                 ANI 1FH! STA USRCODE
1540 C9                         RET ;JMP GOBACK
                ;
                FUNC33:
                        ;RANDOM DISK READ OPERATION
1541 CD5114             CALL RESELECT
1544 C39313             JMP RANDISKREAD ;TO PERFORM THE DISK READ
                        ;RET ;JMP GOBACK
                ;
                FUNC34:
                        ;RANDOM DISK WRITE OPERATION
1547 CD5114             CALL RESELECT
154A C39C13             JMP RANDISKWRITE ;TO PERFORM THE DISK WRITE
                        ;RET ;JMP GOBACK
                ;
                FUNC35:
                        ;RETURN FILE SIZE (0-65536)
154D CD5114             CALL RESELECT
```

```
1550 C3D213            JMP GETFILESIZE
                       ;RET ;JMP GOBACK
                ;
140E =          FUNC36: EQU     SETRANDOM                       ;
                       ;SET RANDOM RECORD
                       ;RET ;JMP GOBACK
                FUNC37:
                ;
1553 2A430B            LHLD    INFO
1556 7D2F5F7C2F        MOV     A,L! CMA! MOV E,A! MOV A,H! CMA
155B 2AAF15A457        LHLD DLOG! ANA H! MOV D,A! MOV A,L! ANA E
1562 5F2AAD15EB        MOV E,A! LHLD RODSK! XCHG! SHLD DLOG
156A 7DA36F            MOV A,L! ANA E! MOV L,A
156D 7CA267            MOV A,H! ANA D! MOV H,A
1570 22AD15C9          SHLD RODSK! RET
                ;
                ;
                GOBACK:
                       ;ARRIVE HERE AT END OF PROCESSING TO RETURN TO USER
1574 3ADE15B7CA        LDA RESEL! ORA A! JZ RETMON
                               ;RESELECTION MAY HAVE TAKEN PLACE
157B 2A430B3600                LHLD INFO! MVI M,0 ;FCB(0)=0
1580 3AE015B7CA                LDA FCBDSK! ORA A! JZ RETMON
                               ;RESTORE DISK NUMBER
1587 77                        MOV M,A ;FCB(0)=FCBDSK
1588 3ADF1532D6                LDA OLDDSK! STA LINFO! CALL CURSELECT
                ;
                ;      RETURN FROM THE DISK MONITOR
                RETMON:
1591 2A0F0BF9          LHLD ENTSP! SPHL ;USER STACK RESTORED
1595 2A450B7D44        LHLD ARET! MOV A,L! MOV B,H ;BA = HL = ARET
159A C9                RET
0B04 =          FUNC38: EQU     FUNC$RET
0B04 =          FUNC39  EQU     FUNC$RET
                FUNC40:
                       ;RANDOM DISK WRITE WITH ZERO FILL OF UNALLOCATED BLOCK
159B CD5114            CALL RESELECT
159E 3E0232D515        MVI A,2! STA SEQIO
15A3 0E00              MVI     C,FALSE
15A5 CD0713            CALL    RSEEK1
15A8 CC0312            CZ      DISKWRITE       ;IF SEEK SUCCESSFUL
15AB C9                RET
                ;
                ;
                ;      DATA AREAS
                ;
                ;      INITIALIZED DATA
15AC E5         EFCB:  DB      EMPTY   ;0E5=AVAILABLE DIR ENTRY
15AD 0000       RODSK: DW      0       ;READ ONLY DISK VECTOR
15AF 0000       DLOG:  DW      0       ;LOGGED-IN DISKS
15B1 8000       DMAAD: DW      TBUFF   ;INITIAL DMA ADDRESS
                ;
                ;      CURTRKA - ALLOCA ARE SET UPON DISK SELECT
                ;      (DATA MUST BE ADJACENT, DO NOT INSERT VARIABLES)
                ;      (ADDRESS OF TRANSLATE VECTOR, NOT USED)
15B3            CDRMAXA:DS     WORD    ;POINTER TO CUR DIR MAX VALUE
```

```
15B5            CURTRKA:DS      WORD    ;CURRENT TRACK ADDRESS
15B7            CURRECA:DS      WORD    ;CURRENT RECORD ADDRESS
15B9            BUFFA:  DS      WORD    ;POINTER TO DIRECTORY DMA ADDRESS
15BB            DPBADDR:DS      WORD    ;CURRENT DISK PARAMETER BLOCK ADDRESS
15BD            CHECKA: DS      WORD    ;CURRENT CHECKSUM VECTOR ADDRESS
15BF            ALLOCA: DS      WORD    ;CURRENT ALLOCATION VECTOR ADDRESS
0008 =          ADDLIST EQU     $-BUFFA ;ADDRESS LIST SIZE
                ;
                ;       SECTPT - OFFSET OBTAINED FROM DISK PARM BLOCK AT DPBADDR
                ;       (DATA MUST BE ADJACENT, DO NOT INSERT VARIABLES)
15C1            SECTPT: DS      WORD    ;SECTORS PER TRACK
15C3            BLKSHF: DS      BYTE    ;BLOCK SHIFT FACTOR
15C4            BLKMSK: DS      BYTE    ;BLOCK MASK
15C5            EXTMSK: DS      BYTE    ;EXTENT MASK
15C6            MAXALL: DS      WORD    ;MAXIMUM ALLOCATION NUMBER
15C8            DIRMAX: DS      WORD    ;LARGEST DIRECTORY NUMBER
15CA            DIRBLK: DS      WORD    ;RESERVED ALLOCATION BITS FOR DIRECTORY
15CC            CHKSIZ: DS      WORD    ;SIZE OF CHECKSUM VECTOR
15CE            OFFSET: DS      WORD    ;OFFSET TRACKS AT BEGINNING
000F =          DPBLIST EQU     $-SECTPT        ;SIZE OF AREA
                ;
                ;       LOCAL VARIABLES
15D0            TRANV:  DS      WORD    ;ADDRESS OF TRANSLATE VECTOR
                FCB$COPIED:
15D2                    DS      BYTE    ;SET TRUE IF COPY$FCB CALLED
15D3            RMF:    DS      BYTE    ;READ MODE FLAG FOR OPEN$REEL
15D4            DIRLOC: DS      BYTE    ;DIRECTORY FLAG IN RENAME, ETC.
15D5            SEQIO:  DS      BYTE    ;1 IF SEQUENTIAL I/O
15D6            LINFO:  DS      BYTE    ;LOW(INFO)
15D7            DMINX:  DS      BYTE    ;LOCAL FOR DISKWRITE
15D8            SEARCHL:DS      BYTE    ;SEARCH LENGTH
15D9            SEARCHA:DS      WORD    ;SEARCH ADDRESS
15DB            TINFO:  DS      WORD    ;TEMP FOR INFO IN "MAKE"
15DD            SINGLE: DS      BYTE    ;SET TRUE IF SINGLE BYTE ALLOCATION MAP
15DE            RESEL:  DS      BYTE    ;RESELECTION FLAG
15DF            OLDDSK: DS      BYTE    ;DISK ON ENTRY TO BDOS
15E0            FCBDSK: DS      BYTE    ;DISK NAMED IN FCB
15E1            RCOUNT: DS      BYTE    ;RECORD COUNT IN CURRENT FCB
15E2            EXTVAL: DS      BYTE    ;EXTENT NUMBER AND EXTMSK
15E3            VRECORD:DS      WORD    ;CURRENT VIRTUAL RECORD
15E5            ARECORD:DS      WORD    ;CURRENT ACTUAL RECORD
15E7            ARECORD1:       DS      WORD    ;CURRENT ACTUAL BLOCK# * BLKMSK
                ;
                ;       LOCAL VARIABLES FOR DIRECTORY ACCESS
15E9            DPTR:   DS      BYTE    ;DIRECTORY POINTER 0,1,2,3
15EA            DCNT:   DS      WORD    ;DIRECTORY COUNTER 0,1,...,DIRMAX
15EC            DREC:   DS      WORD    ;DIRECTORY RECORD 0,1,...,DIRMAX/4
                ;
1600 =          BIOS    EQU     ($ AND 0FF00H)+100H     ;NEXT MODULE
15EE                    END
```

```
0D64 ADDH          0008 ADDLIST       15BF ALLOCA        0C84 ALLOCATED     1264 ALLOCWD
0000 ARECH         0001 ARECL         15E5 ARECORD       15E7 ARECORD1      0B45 ARET
0C8A ATRAN         0C90 ATRAN0        0A99 BACKSP        09A4 BACKUP        0A4E BACKX
1384 BADSEEK       0006 BDOSA         0811 BDOSE         1600 BIOS          15C4 BLKMSK
15C3 BLKSHF        1248 BLOCKOK       15B9 BUFFA         10E1 BUFFNZERO     0001 BYTE
15B3 CDRMAXA       15BD CHECKA        0D44 CHECKRODIR    0D47 CHECKROFILE
0D9E CHECKSUM      0D54 CHECKWRITE    15CC CHKSIZ        10A2 CLOSE         0D72 CLRMODNUM
0B0C COLUMN        0D7F COMPCDR       0B0A COMPCOL       0F07 COMPEXT       0962 COMPOUT
0CF7 COMPUTECS     0CFD COMPUTECS0    13A5 COMPUTERR     0942 CONB0         0945 CONB1
0923 CONBRK        0906 CONECH        08FB CONIN         0948 CONOUT        1001 COPYDIR
0F01 COPYDIRLOC    0FFD COPYFCB       000D CR            0002 CRECH         0003 CRECL
09C9 CRLF          09B1 CRLFP         09B9 CRLFP0        0003 CTLC          0005 CTLE
0008 CTLH          005E CTL           097F CTLOUT        0010 CTLP          0012 CTLR
0013 CTLS          0015 CTLU          0018 CTLX          001A CTLZ          0004 CTRKH
0005 CTRKL         0B42 CURDSK        15B7 CURRECA       1445 CURSELECT     15B5 CURTRKA
15EA DCNT          0F9C DELETE        0FA4 DELETE0       0BBB DIOCOMP       15CA DIRBLK
0AE0 DIRINP        15D4 DIRLOC        15C8 DIRMAX        0004 DIRREC        0DE9 DIRTOUSER
11FB DISKEOF       000C DISKF         11C1 DISKREAD      121C DISKWR0       126E DISKWR1
12BB DISKWR11      12D2 DISKWR2       1300 DISKWR3       1203 DISKWRITE     126C DISKWRU
15AF DLOG          15B1 DMAAD         15D7 DMINX         0C45 DMPOS0        0C53 DMPOS1
0C5C DMPOS2        0C3E DMPOSITION    10FD DMSET         15BB DPBADDR       000F DPBLIST
15E9 DPTR          15EC DREC          08C6 DSKERR        0010 DSKMAP        08BA DSKMSG
0003 DSKMSK        0002 DSKSHF        0022 DVERS         0914 ECHOC         15AC EFCB
00E5 EMPTY         FFFF ENDDIR        1117 ENDMERGE      0DF5 ENDOFDIR      0F83 ENDSEARCH
0B0F ENTSP         08E5 ERRFLG        15C5 EXTMSK        000C EXTNUM        15E2 EXTVAL
0000 FALSE         15D2 FCBCOPIED     15E0 FCBDSK        0020 FCBLEN        10DB FCBNZERO
0005 FCBSHF        128C FILL0         129A FILL1         0AC8 FUNC1         09E1 FUNC10
0AFE FUNC11        147E FUNC12        1483 FUNC13        1445 FUNC14        149C FUNC15
14A5 FUNC16        14AB FUNC17        14C8 FUNC18        14D7 FUNC19        14E0 FUNC20
14E6 FUNC21        14EC FUNC22        14F5 FUNC23        14FE FUNC24        1504 FUNC25
150A FUNC26        1511 FUNC27        0D2C FUNC28        0990 FUNC2         1517 FUNC29
151D FUNC30        1526 FUNC31        152D FUNC32        1541 FUNC33        1547 FUNC34
154D FUNC35        140E FUNC36        0ACE FUNC3         1553 FUNC37        0B04 FUNC38
0B04 FUNC39        159B FUNC40        0AD4 FUNC6         0AED FUNC7         0AF3 FUNC8
0AF8 FUNC9         0B04 FUNCRET       0847 FUNCTAB       0080 FWFMSK
0E35 GETALLOCBIT                      0FBE GETBLOCK      0C5E GETDM         0C71 GETDMD
0D5E GETDPTRA      0CA6 GETEXTA       0CAE GETFCBA       0CBB GETFCB
13D2 GETFILESIZE                      0D69 GETMODNUM     1406 GETNEXTSIZE
13E4 GETSIZE       1574 GOBACK        0B4A GOERR         0D04 HLROTL        0D05 HLROTL0
0CEA HLROTR        0CEB HLROTR0       0BA1 HOME          0C77 INDEX         1040 INDIC0
103B INDICATORS    0B43 INFO          0EB1 INITIAL0      0ED2 INITIAL2      0DC4 INITIALCS
0EA3 INITIALIZE    000A INVIS         0003 IOLOC         0B0E KBCHAR        0FC0 LEFTTST
000A LF            0A70 LINELEN       15D6 LINFO         0B0D LISTCP        0B45 LRET
0B41 LSTACK        001F LSTFCB        007F LSTREC        1124 MAKE          1146 MAKE0
15C6 MAXALL        001F MAXEXT        000F MAXMOD        10CD MERGE0        10E8 MERGED
111F MERGERR       1094 MERGEZERO     000E MODNUM        0B4F MOVE          0B50 MOVE0
000F NAMLEN        0D9C NEWCHECKSUM                      0029 NFUNCS        123B NOPBLOCK
1475 NOSELECT      12FE NOSPACE       0979 NOTBACKSP     0ABD NOTC          0A37 NOTE
0A16 NOTH          0A48 NOTP          0A26 NOTRUB        0AA6 NOTR          0A6B NOTU
0A5F NOTX          12DF NOUPDATE      0D1E NOWRITE       0020 NXTREC        0000 OFF
15CE OFFSET        15DF OLDDSK        FFFF ON            1051 OPEN          105A OPENCOPY
1183 OPENMOD       108B OPENRCNT      115A OPENREEL      118E OPENREEL0     11AC OPENREEL1
11AF OPENREEL2     11B6 OPENRERR      09AC PCTLH         0EF6 PDOLLAR       0809 PERERR
08CA PERMSG        0899 PERSUB        09D3 PRINT         14C2 QSELECT       1347 RANCLOSE
1393 RANDISKREAD                      139C RANDISKWRITE                     0021 RANREC
15E1 RCOUNT        0BB2 RDBUFF        0DD4 RDDIR         0AA9 RDECH1        0AA6 RDECHO
```

```
09E1 READ          0E05 READDIR       0E19 READDIR0      0E20 READDIR1      0AC1 READEN
09F1 READN0        09EF READNX        0000 REBOOT        000F RECCNT        11E6 RECORDOK
0080 RECSIZ        1016 RENAME        1027 RENAME0       0A78 REP0          0A8A REP1
1451 RESELECT      15DE RESEL         0FEC RETBLOCK      0FF4 RETBLOCK0     1591 RETMON
0B9D RETSELECT     0FD1 RIGHTTST      15D3 RMF           080D RODERR        08E1 RODMSG
15AD RODSK         08AB RODSUB        080F ROFERR        0009 ROFILE        08DC ROFMSG
08B1 ROFSUB        0E56 ROTL          0E64 ROTR          1303 RSEEK         1307 RSEEK1
007F RUBOUT        0E6B SCANDM        0E75 SCANDM0       0E88 SCANDM1       0E8E SCANDM2
0E9D SCANM3        0F18 SEARCH        15D9 SEARCHA       0F73 SEARCHEXT     0F94 SEARCHFIN
15D8 SEARCHL       0F53 SEARCHLOOP    0F2D SEARCHN       0F4A SEARCHNEXT    0F7C SEARCHOK
15C1 SECTPT        0BE4 SEEK0         0BFA SEEK1         0C0F SEEK2         1010 SEEKCOPY
0BC3 SEEKDIR       0BD1 SEEK          138B SEEKERR       137F SEEKOK        0B59 SELECTDISK
1421 SELECT        080B SELERR        0B47 SELERROR      08D5 SELMSG        08A5 SELSUB
11BC SEQDISKREAD                      11FE SEQDISKWRITE                     15D5 SEQIO
0E5C SETALLOCBIT                      0D0B SETCDISK      0D8C SETCDR        0DDA SETDATA
0DE0 SETDIR        0DE3 SETDMA        0DFE SETENDDIR     0CD2 SETFCB        0CDE SETFCB1
0D78 SETFWF        0B05 SETLRET1      140E SETRANDOM     0D2C SETRO         140C SETSIZE
153B SETUSRCODE    15DD SINGLE        0018 SSIZE         0B01 STARET        1529 STHLRET
0B0B STRTCOL       0D95 SUBDH         0009 TAB           0996 TAB0          0990 TABOUT
0080 TBUFF         0004 TCRECH        0005 TCRECL        0000 TEST          005C TFCB
15DB TINFO         15D0 TRANV         00FF TRUE          000D UBYTES        0B41 USRCODE
15E3 VRECORD       08B4 WAITERR       0002 WORD          0BB8 WRBUFF        0DC6 WRDIR
```