```
                        title   'CCP/M-86 Bdos Loader'
                    ;***************************************************************
                    ;***************************************************************
                    ;**                                                           **
                    ;**   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) 1983
                    ;       Digital Research
                    ;       Box 579, Pacific Grove
                    ;       California
                    ;
                    ;       January 1983
                    ;
                    ;*******************************************************
                    ;
0000                bdosoffset      equ     0000h   ; offset of BDOS-86
0906                ldroffset       equ     0906h   ; offset of CLDCCPM
0900                biosoffset      equ     0900h   ; offset of BIOS-86
                    ;
                    ;******************* BDOS symbols: *******************
                    ;
                    ;
FFFF                true    equ     0ffffh
0000                false   equ     not true
                    ;
                    ;       Special 8086 symbols:
                    ;
0000                b       equ     byte ptr 0
0000                w       equ     word ptr 0
                    ;
                    ;*******************************************************
                    ;
                    ;       BIOS Function numbers
                    ;
                    ;io_const       equ     0       ;console status function
                    ;io_conin       equ     1       ;console input function
                    ;io_conout      equ     2       ;console output function
                    ;io_listst      equ     3       ;list status function
                    ;io_list        equ     4       ;list output function
                    ;io_auxin       equ     5       ;aux input function
                    ;io_auxout      equ     6       ;aux output function
                    ;io_            equ     7
                    ;io_            equ     8
0009                io_seldsk       equ     9       ;select disk function
000A                io_read         equ     10      ;read disk function
                    ;io_write       equ     11      ;write disk function
                    ;io_flush       equ     12      ;flush buffers function
                    ;io_            equ     13
```

```
                        ;
                        ;       literal constants
                        ;
FFFF                    enddir  EQU     0ffffh          ;end of directory
                        ;
                        ;       file control block (fcb) constants
                        ;
0020                    fcblen  EQU     32              ;fcb length
                        ;empty  EQU     0e5h            ;empty directory entry
0080                    recsiz  EQU     128             ;record 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)
001F                    maxext  EQU     31              ;largest extent number
003F                    maxmod  EQU     63              ;largest module number
000F                    namlen  EQU     15              ;name length
                        ;lstfcb EQU     fcblen-1

                        ;drv    EQU     0               ;drive field
                        ;f1     EQU     1               ;file name byte 1 to 8
                        ;f2     EQU     2
                        ;f3     EQU     3
                        ;f4     EQU     4
                        ;f5     EQU     5
                        ;f6     EQU     6
0007                    f7      EQU     7
                        ;f8     EQU     8
                        ;
                        ;       reserved file indicators
                        ;
                        ;rofile EQU     9               ;t1' -> read/only file
                        ;sysfil EQU     10              ;t2' -> system file
                        ;ARCHIV EQU     11              ;t3' -> FILE HAS BEEN ARCHIVED
000C                    extnum  EQU     12              ;extent number field
000D                    chksum  EQU     13              ;unfilled bytes field
000E                    modnum  EQU     14              ;data module number
000F                    reccnt  EQU     15              ;record count field
0010                    dskmap  EQU     16              ;disk map field
0020                    nxtrec  EQU     fcblen
                        ;ranrec EQU     nxtrec+1        ;random record field (3 bytes)
                        ;
                        ;
                        ;       interrupt control indicators
                        ;
0200                    interruptbit    equ     200h    ; bit 9 of flag word
                        ;
                        ;
                        ;***************** end of BDOS symbols ****************

                        ;****************** BDOS entry module ******************
```

```
                        ;
                        ;       Perform branching, error handling and special
                        ;       80d6 handling.
                        ;
                        ;
                                cseg
                                org     ldroffset
                        ldr_entry:
                        ;
                                org     bdosoffset
                        os_init:
0000 E90500     0008            jmp user_init
                        os_entry:
0003 E92E00     0034            jmp user_entry

0006 0000                       sysdat  dw      0               ;system data segment

                        ;=========
                        user_init:
                        ;=========
0008 8CC8                       mov ax,cs
000A 8ED8                       mov ds,ax
000C 8EC0                       mov es,ax
000E 8ED0                       mov ss,ax
0010 BC8408                     mov sp,offset bdosstack
0013 8C0EAC07                   mov bioscs,cs
0017 FF1EAA07                   callf dword ptr iosentry
001B C706AA070309               mov biosco,bios_offset+3        ;set io call to entry
0021 33C0                       xor ax,ax
0023 1E8ED8                     push ds! mov ds,ax
0026 C70680030300               mov word ptr .0380h,offset os_entry
002C 8C0E8203                   mov .0382h,cs
0030 1F                         pop ds
0031 E9D208     0906            jmp ldr_entry

                        ;
                        ;-----------------------------------------------------
                        ;==========
                        user_entry:
                        ;==========
                        ;       User Entry Point - enter here from a INT 224
                        ;
                        ;               REGISTER USAGE
                        ;       ENTRY                   EXIT
                        ;       -----                   ----
                        ;  CL - Function Code      AX - Copy of BX
                        ;  DX - Param               BX - Return
                        ;  DS - Seg Addr            CX - Error Code
                        ;                           ES - Segment Return
                        ;
                        ;               DS,SI,DI,BP preserved through call
                        ;
```

```
                        ;       contents of users stack
                        ;               Flags
                        ;               CS
                        ;               IP              <- u_stack_ss,_sp
                        ;       DS =  Sysdat Segment
                        ;       u_wrkseg = user's DS
                        ;       u_retseg = user's ES

0034 FC8CD8                     cld ! mov ax,ds                 ;AX = user's DS
0037 0E1F                       push cs ! pop ds
0039 8C06F707                   mov u_retseg,es                 ;save user's ES

003D A3C308                     mov u_wrkseg,ax                 ;wipe out earlier work seg

0040 8BDC                       mov bx,sp                       ;enable interrupt if it was on
0042 36F747040002               test ss:word ptr 4[bx],interruptbit
0048 7401           0048        jz bdose1
004A FB                           sti
                        bdose1:
004B 1E07                       push ds! pop es
004D 565755                     push si! push di! push bp

0050 E80E00         0061        call func                       ;chk netwrk, returns BX, ES, CX

0053 5D5F5E                     pop bp! pop di! pop si          ;setup user's environment and return
0056 8E06F707                   mov es,u_retseg                 ;restore user's ES
005A 8E1EC308                   mov ds,u_wrkseg                 ;DS = user's entry DS
005E 8BC3                       mov ax,bx                       ;parameter return BX = AX
0060 CF                         iret                            ;back to user ...


                        func:
                        ;----
0061 80F90E7506     006C        cmp cl,14! jne fn1
                                  mov si,offset bdofunc+0
0069 E96400         00D0          jmp bdo_entry
006C 80F90F7506     0077 fn1:   cmp cl,15! jne fn2
                                  mov si,offset bdofunc+3
0074 E95900         00D0          jmp bdo_entry
0077 80F9147506     0082 fn2:   cmp cl,20! jne fn3
                                  mov si,offset bdofunc+6
007F E94E00         00D0          jmp bdo_entry
0082 80F91A7413     009A fn3:   cmp cl,26! je func26
0087 80F9207413     009F        cmp cl,32! je func32
008C 80F92C741F     00B0        cmp cl,44! je func44
0091 80F933742C     00C2        cmp cl,51! je func51
                        badfunc:
0096 BBFFFF                       mov bx,0ffffh                 ;return ffff for illegal function
0099 C3                           ret
                        ;
                        func26:         ;set the subsequent dma address to info
                        ;======
```

```
 213
 214   009A 8916C508                    mov dmaad,dx                    ;dmaad = info
 215   009E C3                          ret
 216
 217                            func32:         ;set user code
 218                            ;======
 219
 220   009F 8AC2                        mov al,dl
 221   00A1 3CFF                        cmp al,0ffh
 222   00A3 7505          00AA          jnz setusrcode
 223   00A5 8A1EC007                      mov bl,p_user                 ;interrogate user code instead
 224   00A9 C3                            ret
 225                            setusrcode:
 226   00AA 240F                        and al,0fh
 227   00AC A2C007                      mov p_user,al
 228   00AF C3                          ret                             ;jmp goback
 229
 230                            func44:         ;set multi-sector count
 231                            ;======
 232   00B0 8AC2                        mov al,dl
 233   00B2 33DB                        xor bx,bx
 234   00B4 0AC0                        or al,al
 235   00B6 7408          00C0          jz return_not_ok
 236   00B8 3C81                        cmp al,129
 237   00BA 7304          00C0          jnb return_not_ok
 238   00BC A2C307                        mov u_mult_cnt,al
 239   00BF C3                            ret
 240                            return_not_ok:
 241   00C0 4B                          dec bx                  ;BX = 0ffffh
 242   00C1 C3                          ret
 243
 244                            func51:         ;set segment base for disk I/O
 245                            ;======
 246   00C2 8916C708                    mov     dmabase,dx
 247   00C6 C3                          ret
 248
 249                            ;
 250                            ;*************** end BDOS entry module ***************
 251
 252                            ;***************** BDOS Disk Functions *****************
 253
 254                            ;*********************************************************
 255                            ;*
 256                            ;* bdos function table
 257                            ;*
 258                            ;*********************************************************
 259
 260                                    ; structure of entry in functab
 261
 262     0000                   btab_addr       equ     word ptr 0
 263     0002                   btab_flag       equ     byte ptr (btab_addr + word)
 264     0001                   bf_getmx        equ     001h            ;get mxdisk queue
 265     0002                   bf_cshell       equ     002h            ;conditional shell function
```

```
                                ; bdos function table

00C7                    bdofunc equ     offset $
00C7 A00501                     dw func14   ! db 0001b  ; fxi14  equ 01h  ;select disk
00CA AA0601                     dw func15   ! db 0001b  ; fxi15  equ 02h  ;open file
00CD D40503                     dw func20   ! db 0011b  ; fxi20  equ 07h  ;read sequential

                        ;========
                        bdoentry:               ; bdos module entry point
                        ;========
                        ;       entry:  SI = offset of bdos functab entry
                        ;               DX = argument
                        ;               DS = ES = system data area
                        ;       exit:   BX = return code

00D0 A18F07                     mov ax,word ptr p_dsk
00D3 A3C908                     mov word ptr seldsk,ax          ;set default disk and user code

00D6 B9070033C0                 mov cx,zerolength ! xor ax,ax   ;zero local variables
00DB BFAE07                     mov di,offset fcbdsk
00DE F3AA                       rep stosb

00E0 8916C108                   mov info,dx                     ;info=dx

00E4 803EC3070174 00F3          cmp mult_cnt,1 ! je noshell     ;if mult_cnt <> 1 and
     08           00F3
00EB 2EF584020002               test cs:btabflag[si],bf_cshell  ; func uses mult_cnt then
00F1 7508         00FB          jnz shell                       ; use bdos multi sector shell
                        noshell:
00F3 E86100       0157          call call_bdos
                        retmon:
00F6 8B1EB007                   mov bx,aret
00FA C3                         ret
                        ;
                        shell:
                        ;-----
                        ;       entry:  SI = offset of functab entry

00FB 8936F207                   mov shell_si,si
00FF A1C508A3F407               mov ax,dmaadr ! mov shell_dma,ax
0105 2E8AA40200                 mov ah,cs:btabflag[si]
010A E86E00       0178          call parsave33
010D C606F607FF                 mov shell_flag,true

0112 A0C307                     mov al,mult_cnt
                        mult_io1:
0115 A28407                     MOV MULT_NUM,AL
0118 50                         push ax
0119 8B36F207                   mov si,shell_si
011D 8B16C108                   mov dx,info
0121 E83300       0157          call call_bdos
```

```
                        mov bl,byte ptr aret
0128 0ADB               or bl,bl
012A 7409       0135    jz no_shell_err
012C 8A3EC307             mov bh,mult_cnt
0130 58                   pop ax
0131 2AF8                 sub bh,al                     ;BH = # of successfull records
0133 EB0D       0142      jmps shell03
                      no_shell_err:
0135 8106C5088000         add dmaad,80h
013B 58FEC8               pop ax! dec al
013E 7505       0115      jnz mult_io1
0140 33DB                 xor bx,bx

                      shell03:
0142 891EB007             mov aret,bx
0146 A1F407A3C508         mov ax,shell_dma! mov dmaad,ax
014C C606F60700           mov shell_flag,false
0151 E84B00     019F      call parunsave
0154 E99FFF     00F6      jmp retmon
                      ;
                      call_bdos:
                      ;---------
                      ;       entry:  DX = argument
                      ;               SI = offset of bdos functab entry

0157 E8EE01     0348      call setdata                  ;ready to go to the function
015A 89268408             mov savesp,sp
015E 2EFF940000           call cs:btab_addr[si]
                      bdos_return:
0163 803EB20700           cmp resel,0
0168 7406       0170      je retmon5
016A A0AE07               mov al,fcbdsk
016D A28608               mov info_fcb,al
                      retmon5:
0170 803EAF07FF           cmp parcopfl,true
0175 7503       017A      jne retmon6
0177 E82500     019F      call parunsave                ;copy local vars to uda
                      retmon6:
017A C3                   ret
                      ;
                      parsave33:      ;copy 33 byte length FCB
                      ;---------
017B B121                 mov cl,33
                          ;jmps parsave
                      ;
                      parsave:        ;copy FCB from user segment to bdos segment
                      ;-------
                      ;       entry:  CL = length of FCB to save

017D F606F607FF           test shell_flag,true
0182 751A       019E      jnz parret
0184 C606AF07FF             mov parcopfl,true
```

```
                                ;
0189 880EF907                     mov parlg,cl
018D 32ED                         xor ch,ch
018F 8B36C108                     mov si,info
0193 BF8608                       mov di,offset info_fcb
0196 1E                           push ds
0197 8E1EC308                     mov ds,parametersegment
019B F3A4                         rep movsb
019D 1F                           pop ds
                        parret:
019E C3                           ret
                                ;
                        parunsave:      ;copy local FCB to user segment
                                ;---------
019F F605F607FF                   test shell_flag,true
01A4 75F8           019E          jnz parret
01A6 8A0EF907                     mov cl,parlg
01AA 32ED                         xor ch,ch
01AC BE8608                       mov si,offset info_fcb
01AF 8B3EC108                     mov di,info
01B3 06                           push es
01B4 8E06C308                     mov es,parametersegment
01B8 F3A4                         rep movsb
01BA 07                           pop es
01BB C3                           ret

                        setlret1:
01BC B001                         mov al,1
                        staret:
01BE A26007                       mov lret,al
01C1 C3                           ret

                                  ;these functions added for mpm interface

                                ;*********************************************************
                                ;*
                                ;*       bdos - xios interface
                                ;*
                                ;*********************************************************

                                ;======          ========================
                                xiosif:         ; xios interface routine
                                ;======          ========================
                                ;       entry:  AL = function number
                                ;               CX = argument 1
                                ;               DX = argument 2
                                ;       exit:   AX = BX = output

01C2 06                           push es
01C3 FF1EAA07                     callf dword ptr iosentry
01C7 FC07                         cldl pop es
01C9 C3                           ret
```

```
                        ;========       ================================
                        rwxiosif:       ; disk read/write xios interface
                        ;========       ================================
                        ;               entry:  AL = function number
                        ;               exit:   AX = BX = output
                        
01CA 8B16EA07                   mov dx,arecord
01CE 8A2EEC07                   mov ch,byte ptr arecord+2
01D2 8A1EC407                   mov bl,curdsk
01D6 B701                       mov bh,1
01D8 863ED507                   xchg bh,mult_sec        ;BH = multi sector count
                                                        ; stack on entry to the xios
01DC 53                         push bx                 ;    +C | DRV  | MCNT |
01DD FF36BB07                   push track              ;    +C |    TRACK     |
01E1 FF35BD07                   push sector             ;    +8 |   SECTOR     |
01E5 FF36CD07                   push curdmabase         ;    +6 |   DMA_SEG    |
01E9 FF36EF07                   push curdma             ;    +4 |   DMA_OFF    |
                                                        ;    +2 |   RET_SEG    |
                                                        ; SP+0 |   RET_OFF    |
01ED FF1EAA07                   callf dword ptr iosentry
01F1 83C40A                     add sp,10               ;remove parameters from stack
01F4 FC1E07                     cldl push dsl pop es
01F7 C3                         ret
                        
                        ;*****************************************************************
                        ;*****************************************************************
                        ;**                                                             **
                        ;**   b a s i c    d i s k   o p e r a t i n g   s y s t e m    **
                        ;**                                                             **
                        ;*****************************************************************
                        ;*****************************************************************
                        
                        ;****************** bdos file system ******************
                        
                        ;       error message handlers
                        
                        pererror:       ;report permanent error
                        ;--------
01F8 B501                       mov ch,1
01FA EB07               0203    jmps goerr
                        
                        selerror:       ;report select error
                        ;--------
01FC C606C407FF                 mov curdsk,0ffh
0201 B504                       mov ch,4
                        goerr:
0203 B1FF                       mov cl,0ffh
0205 890EB007                   mov aret,cx                     ;set aret
                        goback:
0209 8B268408                   mov sp,save_sp
020D E953FF             0163    jmp bdos_return
                        
```

```
                        ;       local subroutines for bios interface
                        ;-------------------------------------------------

                        move:           ;block move data
                        ;----
                        ;       entry:  CL = length of data to move
                        ;               DX = source offset
                        ;               BX = destination offset

0210 32ED                       xor ch,ch
0212 8BF2                       mov si,dx
0214 8BF8                       mov di,bx
0216 F3A4                       rep movsb
0218 C3                         ret

                        selectdisk:
                        ;----------
                        ;       select the disk drive given in AL, and fill
                        ;       the base addresses curtrka - alloca, then fill
                        ;       the values of the disk parameter block
                        ;       entry:  AL = disk to select
                        ;       exit:   C flag set if no error

0219 8AC8                       mov cl,al                       ;current disk# to cl
                                                                ;lsb of dl = 0 if not yet
021B B009                       mov al,io_seldsk                ;logged in
021D E8A2FF     01C2            call xiosif                     ;bx filled by call
                                                                ;bx = 0000 if error,
0220 0BDB                       or bx,bx                        ;otherwise disk headers
0222 7430       0254            jz ret4                         ;rz - carry flag reset
0224 83C308                     add bx,8
0227 8BF3                       mov si,bx
0229 BFC807                     MOV di,OFFSET DPBADDR           ;dx= source for move, bx=dest
022C B90A00                     mov cx,addlist                  ;addlist filled
022F F3A4                       rep movsb                       ;now fill the disk
0231 8B36C807                   mov si,dpbaddr                  ;parameter block
0235 BFD207                     mov di,offset sectpt            ;bx is destination
0238 B91100                     mov cx,dpblist
023B F3A4                       rep movsb                       ;data filled
023D 8A0EE107                   mov cl,physhf                   ;convert # sectors per track
0241 D326D207                   shl sectpt,cl                   ; from physical to logical
                                                                ;set single/double map mode
0245 A0D807                     mov al,byte ptr maxall+1        ;largest allocation number
0248 0AC0                       or al,al
024A 7402       024E            jz retselect
                                                                ;if high order of maxall not
024C B001                         mov al,1                      ;zero then use double disk map
                        retselect:
024E FEC8                       dec al
0250 A2E507                     mov single,al                   ;true if single disk map mode
0253 F9                         stc                             ;select disk function ok
```

```
                                ret4:
0254 C3                                 ret

                                rdbuff:         ;read buffer and check if ok
                                ;------
0255 B00A                               mov al,io_read
0257 E870FF     01CA                    call rwxiosif                   ;current drive, track,....

                                diocomp:        ;check for disk errors
                                ;-------
                                ;       entry:  AL = xios return code

025A 0AC074F6   0254                    or al,all jz ret4               ;rz
025E E997FF     01F8                      jmp pererror                  ; no

                                seak:           ;seek the track given by arecord (actual record)
                                ;----
0261 A1EA07                             mov ax,arecord                  ;compute track/sector
0264 3302                               xor dx,dx
0266 8A16EC07                           mov dl,byte ptr arecord+2
026A F7360D07                           div sectpt                      ;dx=sector, ax=track
026E 0306DF07                           add ax,offsetv
0272 A3B807                             MOV TRACK,ax                    ;save bios/xios track
0275 8A0EE107                           MOV CL,PHYSHF
0279 D3EA                               SHR dx,CL                       ;PHY SECTOR = SHR(LOG SECTOR,PHYSHF)
027B 8916BD07                           MOV SECTOR,dx                   ;save bios/xios sector
027F C3                                 ret

                                ;       utility functions for file access

                                dmposition:     ;compute disk map position for vrecord
                                ;----------
                                ;       exit:   AL = disk map position of vrecord

0280 8A0ED407                           mov cl,blkshf                   ;shift count to CL
0284 8A2EE807                           mov ch,vrecord                  ;current virtual record to a
0288 D2E0                               shr ch,cl                       ;CH = shr(vrecord,blkshf)
                                                                        ;   = vrecord/2**(sect/block)
028A F6D9                               neg cl
028C 80C107                             add cl,7                        ;CL = 7 - blkshf
028F A0E707                             mov al,extval                   ;extent value and extmsk
                                                                        ;blkshf = 3,4,5,6,7
                                                                        ;CL = 4,3,2,1,0
                                                                        ;shift is 4,3,2,1,0
0292 D2E0                               shl al,cl                       ;AL = shl(ext and extmsk,7-blkshf)
0294 02C5                               add al,ch                       ;add the previous
                                                                        ;shr(vrecord,blkshf) value
                                                                        ;AL 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
0296 C3                         ret                     ;with dm$position in a

                        getdm:          ;return disk map value from position given by cx
                        ;-----
                        ;       entry:  CX = index into disk map
                        ;       exit:   BX = disk map value at position CX

0297 8B9508                     mov bx,offset info_fcb+dskmap
029A 03D9                       add bx,cx               ;index by asingle byte value
029C 803EE50700                 cmp single,0            ;single byte/map entry?
02A1 7405               02A8    jz getdmd               ;get disk map single byte
02A3 8A1F                         mov bl,[bx]
02A5 32FF                         xor bh,bh
02A7 C3                           ret                   ;with bx=00bb
                        getdmd:
02A8 03D9                       add bx,cx               ;bx=.fcb(dm+1*2)
02AA 8B1F                       mov bx,[bx]             ;return double precision value
02AC C3                         ret

                        index:          ;compute disk block number from current fcb
                        ;-----
                        ;       exit:   BX = disk map value for vrecord in current fcb
                        ;               Z flag set according to value in BX

02AD E8D0FF             0280    call dmposition         ;0...15 in register al
02B0 A2E407                     MOV DMINX,AL
02B3 8AC8                       mov cl,al
02B5 32ED                       xor ch,ch
02B7 E8DDFF             0297    call getdm              ;value to bx
02BA 891EEA07                   mov arecord,bx
02BE 0BDB                       or bx,bx
02C0 C3                         ret

                        atran:          ;compute actual record address, assuming index called
                        ;-----
02C1 8A0ED407                   mov cl,blkshf           ;shift count to reg al
02C5 A1EA07                     mov ax,arecord
02C8 32FF                       xor bh,bh
02CA 8ADC                       mov bl,ah
02CC D3E0                       shl ax,cl
02CE D3E3                       shl bx,cl
02D0 93                         xchg ax,bx
02D1 A0E807                     mov al,vrecord
02D4 2206D507                   and al,blkmsk           ;masked value in al
02D8 A28607                     mov blkoff,al
02DB 0AD8                       or bl,al
02DD 891EEA07                   mov arecord,bx          ;arecord=bx or
                                                        ;(vrecord and blkmsk)
02E1 8826EC07                   mov byte ptr arecord+2,ah
```

```
                                ret

                        getfcb:         ;set local variables from currently addressed fcb
                        ;------
02E6 A0A508                     mov al,info_fcb+nxtrec
02E9 A2E807                     mov vrecord,al                  ;vrecord=fcb(nxtrec)
02EC 803E950800                 cmp info_fcb+reccnt,0
02F1 7508           02FB        jne getfcb0
02F3 E8A300         0399          call get_dir_ext
02F6 8AC8                         mov cl,al
02F8 E8A101         049C          call set_rc
                        getfcb0:
02FB A09508                     mov al,info_fcb+reccnt
02FE 3C817202       0304        cmp al,81h| jb getfcb1
0302 B080                         mov al,80h
                        getfcb1:
0304 A2E607                     mov rcount,al                   ;rcount=fcb(reccnt)
0307 A0D607                     mov al,extmsk                   ;extent mask to a
030A 22069208                   and al,info_fcb+extnum          ;fcb(extnum) and extmsk
030E A2E707                     mov extval,al
0311 C3                         ret

                        setfcb:         ;place local values back into current fcb
                        ;------
0312 B001                       mov al,1
0314 0206E807                   add al,vrecord
0318 A2A508                     mov info_fcb+nxtrec,al          ;fcb(nxtrec)=vrecord+seqio
031B 803E950880                 cmp info_fcb+reccnt,80h
0320 7306           0328        jnb ret41                       ;dont reset if fcb(rc) > 7fh
0322 A0E607                     mov al,rcount
0325 A29508                     mov info_fcb+reccnt,al          ;fcb(reccnt)=rcount
0328 C3                 ret41:  ret

                        getdptra:
                        ;--------
                        ; compute the address of a directory element at
                        ;  positon dptr in the buffer
                        ;       exit:   BX = buffa + dptr

0329 8A1EF107                   mov bl,dptr
032D 32FF                       xor bh,bh
032F 031EC607                   add bx,buffa            ;bx = buffa + dptr
0333 C3                         ret


                        rddir:          ;read the current directory record
                        ;-----
0334 A1C908                     MOV ax,DCNT                     ;seek the record containing
0337 B102D3E8                   MOV CL,DSKSHFT SHR ax,CL        ; the current dir entry
033B A3EA07                     MOV ARECORD,ax                  ;ARECORD = SHR(DCNT,DSKSHF)
033E C606EC0700                 MOV BYTE PTR ARECORD+2,0
0343 B403                       MOV AH,3                        ;LOCATE COMMAND
```

```
690
691   0345 E8A503       0DED          CALL DEBLOCK_DIR
692                                   ;JMPS SETDATA
693
694                           setdata:        ;set data dma address
695                           ;-------
696   0348 A1C708                     MOV ax,DMABASE
697   034B A3ED07                     MOV CUR_DMABASE,ax
698   034E A1C508                     MOV ax,DMAAD
699   0351 A3EF07                     MOV CURDMA,ax
700   0354 C3                         RET
701
702                           endofdir:       ;check if end of directory (dcnt = 0ffffh)
703                           ;--------
704                           ;       exit:   Z flag set if at end of directory
705
706   0355 BBCB08                     mov bx,offset dcnt
707
708                           test_ffff:
709                           ;---------
710   0358 833FFF                     cmp w[bx],0ffffh
711   035B C3                         ret
712
713                           setenddir:      ;set dcnt to the end of directory (dcnt = 0ffffh)
714                           ;---------
715   035C C706CB08FFFF               mov dcnt,enddir
716   0362 C3                         ret
717
718                           read_dir:       ;read next directory entry
719                           ;--------
720
721   0363 8B16D907                   mov dx,dirmax                   ;in preparation for subtract
722   0367 8B1ECB08                   mov bx,dcnt
723   036B 43                         inc bx
724   036C 891ECB08                   mov dcnt,bx                     ;dcnt=dcnt+1
725                                                                   ;continue while dirmax >= dcnt
726                                                                   ;(dirmax-dcnt no cy)
727   0370 2B03                       sub dx,bx
728   0372 72E8         035C          jb setenddir                    ;yes, set dcnt to end
729                                                                   ;of directory
730                           ;       not at end of directory, seek next element
731                                                                   ;cl=initialization flag
732   0374 A0CB08                     mov al,ldcnt
733   0377 2403                       and al,dskmsk                   ;low(dcnt) and dskmsk
734   0379 B105                       mov cl,fcbshf                   ;to multiply by fcb size
735   037B D2E0                       shl al,cl
736                                                                   ;a = (low(dcnt) and dskmsk)
737                                                                   ;shl fcbshf
738   037D A2F107                     mov dptr,al                     ;ready for next dir operation
739   0380 0AC0                       or al,al
740   0382 7503         0387          jnz ret71                       ;return if not a new record
741   0384 E8ADFF       0334          call rddir                      ;read the directory record
742                           ret71:
```

```
                                ret

                        compext:        ;compare extent$ in al with that in cl
                        ;-------
                        ;       entry:  AL,CL = extent numbers to compare
                        ;       exit:   Z flag set if extent numbers match
                        ;               BX,CX,DX = preserved

0388 51                         push cx                         ;save cx's original value
0389 8A2ED607                   mov ch,extmsk
038D F6D5                       not ch                          ;ch has negated form of
                                                                ;extent mask
038F 22C8                       and cl,ch                       ;low bits removed from cl
0391 22C5                       and al,ch                       ;low bits removed from al
0393 2AC1                       sub al,cl
0395 241F                       and al,maxext                   ;set flags
0397 59                         pop cx                          ;restore cx
0398 C3                         ret

                        get_dir_ext:
                        ;-----------
                        ;       compute directory extent from fcb
                        ;       scan fcb disk map backwards
                        ;       upon return dminx = 0 if no blocks are in fcb
                        ;       exit:   AL = directory extent number
                        ;               BX = .fcb(extnum)

0399 BBA508                     mov bx,offset info_fcb+nxtrec   ;BX = .fcb(vrecord)
039C BA0110                     mov dx,1001h                    ;DH = disk map position (rel to 1)
                                                                ;DL = no blocks switch
                        get_del:
039F FECE                       dec dh
03A1 4B                         dec bx                          ;decrement disk map ptr
03A2 803F00                     cmp b[bx],0                     ;is disk map byte non-zero ?
03A5 7506           03AD        jne get_de2                     ; yes
03A7 0AF6                         or dh,dh                      ; no - continue scan
03A9 75F4           039F          jnz get_del
03AB FECA                         dec dl                        ;DL = 0 if no blocks found

                        get_de2:
03AD 8816E407                   mov dminx,dl                    ;dminx = 0 if no blocks in fcb
03B1 803EE507FF                 cmp single,true                 ;are disk block indexes single byte ?
03B6 8AC6                       mov al,dh                       ;al = block offset in disk map
03B8 7402           03BC        jz get_de3                      ;yes
03BA D0E8                         shr al,1                      ;divide block offset by 2

                        ;       al = last non-zero block index in fcb (rel to 0)
                        ;       compute ext offset from last non-zero block index by
                        ;       shifting block index right 7-blkshf

                        get_de3:
03BC B107                       mov cl,7
03BE 2A0ED407                   sub cl,blkshf
```

```
                                ;
                                searchn:
                                ;-------
                                ;       search for the next directory element, assuming
                                ;       a previous call on search which sets searcha and searchhl
                                ;       exit:   Z flag = 0 for successful searches
                                ;
03EF E871FF       0363          call read_dir                   ;read next dir element
03F2 E860FF       0355          call endofdir
03F5 7417         040E          jz srchfin
                                                              ;skip to end if so
                                                              ;not end of directory,
                                                              ;scan for match
03F7 8B16C107                   mov dx,srcha                  ;dx=beginning of user fcb
03FB E823FF       0329          call getdptra                 ;bx = buffa+dptr
03FE 8A0ECD08                   mov cl,srchl                  ;length of search to cl
```

```
                                        xor ch,ch                       ;ch counts up, cl counts down
                                        mov al,[bx]                     ;is fcb an xfcb ?
0404 8A07                               and al,11101111b
0406 24EF                               cmp al,[bx]
0408 3A07                               je srch_loop                    ;no
040A 7408               0417            jmps search_n
040C EBE1               03EF        srchfin:                            ;end of directory, or empty name
                                    lret_eq_ff:
040E B0FF                               mov al,255
0410 8AE8                               mov ch,al
0412 FEC5                               inc ch                          ;zero flag set on unsuccessful
0414 E9A7FD             01BE            jmp staret                      ;searches

                                    srchloop:
0417 0AC9                               or cl,cl
0419 7432               044D            jz endsearch
041B 8BF2                               mov si,dx
041D AC                                 lods al                         ;fcb character
041E 247F                               and al,07fh

                                                                        ;scan next character if not
                                                                        ;chksum byte
0420 80FD0D                             cmp ch,chksum
0423 7413               0442            jz srchok

                                                                        ;not the ubytes field,
                                                                        ;extent field
0425 80FD0C                             cmp ch,extnum                   ;may be extent field
0428 740F               0439            jz srchext                      ;skip to search extent

042A 80FD0E                               cmp ch,modnum                 ;is field module # ?
042D 7502               0431              jnz $+4                       ;no
042F 243F                                   and al,3fh                  ;yes - mask off high order bits

0431 2A07                                 sub al,[bx]
0433 247F                                 and al,7fh                    ;mask-out flags/extent modulus
0435 7513               044A              jnz searchn_jmp
0437 EB09               0442              jmps srchok                   ;matched character

                                    srchext:                            ;AL = fcb character
                                                                        ;attempt an extent # match
0439 51                                 push cx                         ;save counters
043A 8A0F                               mov cl,[bx]                     ;directory character to c
043C E849FF             0388            call compext                    ;compare user/dir char
043F 59                                 pop cx                          ;recall counters
0440 7508               044A            jnz searchn_jmp                 ;skip if no match

                                    srchok:                             ;current character matches
0442 4243                               inc dx! inc bx
0444 FEC5FEC9                           inc ch! dec cl
0448 EBCD               0417            jmps srchloop

                                    searchn_jmp:
```

```
                                ;entire name matches, return dir position
044D 32C0                       xor al,al
044F A28D07                     mov lret,al
0452 8AE8                       mov ch,al
0454 FEC5                       inc ch                          ;zero flag reset on successful
0456 C3                         ret                             ;searches

                        check_wild:     ;check for ? in file name or type
                        ;----------
0457 BB8508                     mov bx,offset info_fcb
045A E80700             0464    call chk_wild
045D 7517               0476    jnz ret10
045F B009                       mov al,9                        ;extended error 9
0461 E97602             060A    jmp set_aret

                        chk_wild:       ;check fcb for ? marks
                        ;--------
                        ;       entry:  BX = .fcb(0)
                        ;       exit:   Z flag = 1 if ? mark found

0464 B90B3F                     mov cx,3f0bh                    ;ch = 3f, cl = 11
                        chk_wild1:
0467 43                         inc bx                          ;advance fcb ptr
0468 8AC5                       mov al,ch
046A 2A07                       sub al,[bx]
046C 22C5                       and al,ch
046E 7406               0476    jz ret10                        ;rtn with z flag set if ? fnd
0470 FEC9                       dec cl
0472 75F3               0467    jnz chk_wild1
0474 0AC0                       or al,al                        ;rtn with z flag reset - no ?s
0476 C3                 ret10:  ret

                        open:           ;search for the directory entry, copy to fcb
                        ;----
0477 E86DFF             03E7    call search_name
047A 74FA               0476    jz ret10                        ;return with lret=255 if end
                                                                ;not end of directory,copy fcb
                        opencopy:
                                                                ;(referenced below to copy fcb
047C 53                         push bx                         ;bx = .fcb(modnum)
047D 4B43                       dec bx! dec bx
047F 8A27                       mov ah,[bx]                     ;ah = extnum
0481 50                         push ax                         ;save extnum & modnum
0482 E8A4FE             0329    call getdptra
0485 8BD3                       mov dx,bx                       ;dx = .buff(dptr)
0487 BB8508                     mov bx,offset info_fcb          ;bx=.fcb(0)
048A B120                       mov cl,nxtrec                   ;length of move operation
048C E881FD             0210    call move                       ;from .buff(dptr) to .fcb(0)
                                                                ;note that entire fcb is
                                                                ;copied, including indicators
```

```
 955
 956   048F E807FF       0399          call get_dir_ext
 957   0492 8AC8                       mov cl,al                       ;cl = dir_ext
 958   0494 5853                       pop ax! pop bx
 959   0496 8807                       mov [bx],al                     ;restore modnum
 960   0498 4B4B                       dec bx! dec bx
 961   049A 8827                       mov [bx],ah                     ;restore extnum number
 962
 963                           ;       bx = .user extent#, cl = dir extent#
 964                           ;       if user ext < dir ext then user := 128 records
 965                           ;       if user ext = dir ext then user := dir records
 966                           ;       if user ext > dir ext then user := 0 records
 967
 968                           set_rc:
 969                           ;------
 970                           ;       entry:  BX = .user extent #
 971                           ;               CL = dir extent #
 972
 973   049C 32ED                       xor  ch,ch
 974   049E BE9508                     mov si,offset info_fcb+reccnt
 975   04A1 8A07                       mov al,[bx]                     ;al = current extent
 976   04A3 2AC1                       sub al,cl                       ;compare fcb ext to dir ext
 977   04A5 7403         04B2          jz set_rc2                      ;fcb ext = dir ext
 978                                                                   ;  actual_rc = 0, fcb(rc) = fcb(rc)
 979   04A7 8AC5                         mov al,ch
 980   04A9 7304         04AF            jae set_rc1                   ;fcb ext > dir ext
 981                                                                   ;  actual_rc = 0, fcb(rc) = 0
 982
 983   04AB B080                           mov al,128                  ;fcb ext < dir ext
 984   04AD 0A04                           or al,[si]                  ;  fcb(rc) = 128 | fcb(rc)
 985                           set_rc1:
 986   04AF 8804                           mov [si],al
 987   04B1 C3                 ret11:      ret
 988
 989                           set_rc2:
 990   04B2 3804                       cmp b[si],al                    ;is fcb(rc) = 0?
 991   04B4 75FB         04B1          jnz ret11                       ;no
 992   04B6 32C0                       xor al,al                       ;AL = 0
 993   04B8 8804                       mov b[si],al                    ;required by func 99
 994   04BA 3806E407                   cmp dminx,al                    ;do blocks exist in fcb?
 995   04BE 74F1         04B1          jz ret11                        ;no
 996   04C0 C60480                     mov b[si],80h                   ;fcb(rc) = 80h
 997   04C3 C3                         ret
 998
 999                           restore_rc:
1000                           ;----------
1001                           ;       if actual_rc <> 0 then fcb(rc) = actual_rc
1002                           ;       entry:  BX = .fcb(extnum)
1003
1004   04C4 A09508                     mov al,info_fcb+reccnt
1005   04C7 3C81                       cmp al,81h
1006   04C9 7205         04D0          jb restore_rc1
1007   04CB 247F                         and al,7fh
```

```
1008
1009  04CD A29508                       mov info_fcb+reccnt,al
1010                              restore_rcl:
1011  04D0 C3                           ret
1012
1013                              openreel:
1014                              ;--------
1015                              ;       close the current extent, and open the next one
1016                              ;       if possible.  rmf is true if in read mode.
1017                              ;       lret is set to 0 if successful
1018
1019  04D1 A09408                       mov al,info_fcb+modnum
1020  04D4 A2E307                       mov save_mod,al                 ;save current module #
1021  04D7 BB9208                       mov bx,offset info_fcb+extnum
1022  04DA 8A07                         mov al,[bx]
1023  04DC 8AC8                         mov cl,al
1024  04DE FEC1                         inc cl                          ;increment ext #
1025  04E0 E8A5FE       0388            call comp_ext                   ;did we cross a dir fcb boundary ?
1026  04E3 7503         04E8            jnz $+5
1027  04E5 E93A00       0522              jmp openr3                    ;no
1028  04E8 B01F                         mov al,maxext
1029  04EA 22C1                         and al,cl
1030  04EC 8807                         mov [bx],al                     ;update fcb extent field
1031  04EE 7508         04F8            jnz openr0                      ;branch if in same module
1032
1033                                  ;               extent number overflow, go to next module
1034
1035  04F0 83C302                       add bx,(modnum-extnum)          ;bx=.fcb(modnum)
1036  04F3 FE07                         inc b[bx]                       ;fcb(modnum)=++1
1037                                                                    ;module number incremented,
1038                                                                    ;check for overflow
1039  04F5 8A07                         mov al,[bx]
1040  04F7 243F                         and al,maxmod                   ;mask high order bits
1041  04F9 7413         050E            jz openerr                      ;cannot overflow to 0
1042                                                                    ;otherwise, ok to continue
1043                                                                    ;with new module
1044                              openr0:
1045  04FB E8E9FE       03E7            call search_name                ;next extent found?
1046  04FE 740E         050E            jz openerr                      ;end of file encountered
1047  0500 E879FF       047C            call opencopy
1048                              openr2:
1049  0503 E8E0FD       02E6            call getfcb                     ;set parameters
1050  0506 32C0                         xor al,al
1051  0508 A2E807                       mov vrecord,al
1052  050B E980FC       018E            jmp staret                      ;lret = 0
1053                                    ;ret
1054
1055                              openerr:
1056                              ;-------
1057  050E BB9208                       mov bx,offset info_fcb+extnum
1058  0511 A0E307                       mov al,save_mod
1059  0514 884702                       mov 2[bx],al
1060  0517 8A07                         mov al,[bx]
```

```
1061
1062   0519 FEC8                        dec al
1063   051B 241F                        and al,1fh
1064   051D 8807                        mov [bx],al
1065                                                        ;cannot move to next extent
1066                                                        ;of this file
1067   051F E99AFC       01BC           jmp setlret1       ;lret = 1
1068                                    ;ret
1069                          openr3:
1070   0522 880F                        mov [bx],cl        ;increment extent field
1071   0524 E872FE       0399           call get_dir_ext
1072   0527 8AC8                        mov cl,al
1073   0529 3A07                        cmp al,[bx]        ;is dir ext < fcb(ext)?
1074   052B 7305         0532           jae openr4         ;no
1075   052D FE0F                          dec b[bx]        ;decrement extent
1076   052F E98AFC       01BC             jmp set_lret1    ;yes - reading unwritten data
1077                          openr4:
1078   0532 E88FFF       04C4           call restore_rc
1079   0535 E864FF       049C           call set_rc
1080   0538 EBC9         0503           jmps openr2
1081
1082                          diskread:                     ;(may enter from seqdiskread)
1083                          ;--------
1084                                                        ;read the next record from
1085                                                        ;the current fcb
1086   053A E8A9FD       02E6           call getfcb        ;sets parameters for the read
1087   053D A0E807                      mov al,vrecord
1088   0540 3A06E607                    cmp al,rcount      ;vrecord-rcount
1089                                                        ;skip if rcount > vrecord
1090   0544 720E         0554           jb recordok
1091                                                        ;not enough records in extent
1092                                                        ;record count must be 128
1093                                                        ;to continue
1094   0546 3C80                          cmp al,128       ;vrecord = 128?
1095   0548 7528         0572             jnz diskeof      ;skip if vrecord<>128
1096   054A E884FF       04D1             call openreel    ;go to next extent if so
1097                                                        ;now check for open ok
1098   054D 803EB00700                    cmp lret,0       ;stop at eof
1099   0552 751E         0572             jnz diskeof
1100                          recordok:
1101   0554 E856FD       02AD           call index         ;fcb addresses a record to read
1102                                                        ;error 2 if reading
1103                                                        ;unwritten data
1104                                                        ;(returns 1 to be compatible
1105                                                        ;with 1.4)
1106   0557 7419         0572           jz diskeof         ;arecord=0000?
1107   0559 E855FD       02C1           call atran         ;arecord now a disk address
1108   055C E81600       0575           CALL CHECK_NPRS
1109   055F 720E         056F           JC RECORDOK2
1110   0561 7503         0566             JNZ $+5
1111   0563 E97F01       06E5               JMP READ_DEBLOCK
1112   0566 E80FFD       0348             CALL SETDATA
1113   0569 E8F5FC       0261             call seek          ;to proper track,sector
```

```
1114
1115  056C E8E6FC       0255            call rdbuff                     ;to dma address
1116                          RECORDOK2:
1117  056F E9A0FD       0312            jmp setfcb                      ;replace parameter
1118
1119                          diskeof:
1120  0572 E947FC       01BC            jmp setlret1                    ;lret = 1
1121                                    ;ret
1122
1123                          CHECK_NPRS:
1124                          ;----------
1125                          ;       DIR_CNT CONTAINS THE NUMBER OF 128 BYTE RECORDS
1126                          ;       TO TRANSFER DIRECTLY.  THIS ROUTINE SET DIR_CNT
1127                          ;       WHEN INITIATING A SEQUENCE OF DIRECT PHYSICAL I/O
1128                          ;       OPERATIONS.  DIR_CNT IS DECREMENTED EACH TIME CHECK_NPRS
1129                          ;       IS CALLED DURING SUCH A SEQUENCE.
1130                          ;       exit:   ~C FLG & ~Z FLG - DIRECT PHYSICAL I/O OPERATION
1131                          ;               ~C FLG &  Z FLG - INDIRECT(DEBLOCK) I/O OPERATION
1132                          ;                 C FLG         - NO PHYSICAL I/O OPERATION
1133
1134  0575 8A2E8607                   MOV CH,blk_off                  ;CH = VRECORD & BLKMSK
1135  0579 A0B307                     MOV AL,DIR_CNT
1136  057C 3C02                       CMP AL,2                        ;IS DIR_CNT > 1?
1137  057E 7207         0587          JC CHECK_NPR1                   ;NO
1138  0580 FEC8                       DEC AL                          ;DIR_CNT = DIR_CNT - 1
1139  0582 A2B307                     MOV DIR_CNT,AL                  ;we are in mid-multi sector i/o
1140  0585 F9                         STC
1141  0586 C3                         ret                             ;RETURN WITH C FLAG SET
1142                          CHECK_NPR1:
1143  0587 A0E207                     MOV AL,PHYMSK                   ;CL = PHYMSK
1144  058A 8AC8                       MOV CL,AL
1145  058C 22C5                       AND AL,CH                       ;AL = VRECORD & PHYMSK
1146                                                                  ;ARE WE IN MID-PHYSICAL RECORD?
1147  058E 740A         059A          JZ CHECK_NPR11                  ;NO
1148                          CHECK_NPR1X:
1149  0590 0AC9                       OR CL,CL                        ;IS PHYMSK = 0?
1150  0592 7403         0597          JZ CHECK_NPR1Y                  ;YES
1151  0594 32C0                       XOR AL,AL                       ;RETURN WITH Z FLAG SET & C FLAG RESET
1152  0596 C3                         RET
1153                          CHECK_NPR1Y:
1154  0597 0C01                       OR AL,1                         ;RESET C & Z FLAGS
1155  0599 C3                         RET
1156                          CHECK_NPR11:
1157  059A 8AF1                       MOV DH,CL
1158  059C F6D6                       not DH                          ;DH = ~ PHYMSK
1159  059E A0B407                     MOV AL,MULTNUM
1160  05A1 3C02                       CMP AL,2                        ;IS MULT_NUM < 2?
1161  05A3 72E8         0590          JC CHECK_NPR1X                  ;YES
1162  05A5 BBE807                     MOV BX,OFFSET VRECORD
1163  05A8 8A27                       MOV AH,[BX]                     ;AH = VRECORD
1164  05AA 02C4                       ADD AL,ah
1165  05AC 3C80                       CMP AL,80H
1166  05AE 7202         05B2          JC CHECK_NPR2
```

```
1167
1168  05B0 B080                      MOV AL,80H
1169                           CHECK_NPR2:                    ;AL = MIN(VRECORD + MULT_NUM,80H) = X
1170  05B2 51                        PUSH CX                  ;SAVE VRECORD & BLKMSK, PHYMSK
1171  05B3 C6077F                    MOV BE[BX],7FH           ;VRECORD = 7F
1172  05B6 5350                      PUSH BX! PUSH AX
1173  05B8 8AD8                      MOV BL,AL                ;BL = X
1174  05BA A0D507                    MOV AL,BLKMSK
1175  05BD 8AD0                      MOV DL,AL
1176  05BF FEC2                      INC DL                   ;DL = BLKMSK + 1
1177  05C1 F6D0                      not AL
1178  05C3 22E0                      AND AH,AL                ;AH = VRECORD & ~BLKMSK
1179  05C5 A0E607                    MOV AL,RCOUNT
1180  05C8 22C6                      AND AL,DH                ;AL = RCOUNT & ~PHYMSK
1181  05CA 3AC3                      CMP AL,BL                ;IS AL < X?
1182  05CC 7202          05D0        JC CHECK_NPR23           ;YES
1183  05CE 8AC3                      MOV AL,BL                ;AL = MIN(VRECORD + MULT_NUM,80H) = X
1184                           CHECK_NPR23:
1185  05D0 2AC4                      SUB AL,AH                ;AL = AL - VRECORD & ~BLKMSK
1186  05D2 3AC2                      CMP AL,DL                ;IS AL < BLKMSK + 1?
1187  05D4 7243          0619        JC CHECK_NPR9            ;YES
1188  05D6 50                        PUSH AX                  ;AL = MAX # OF RECORDS
1189  05D7 E8A6FC        0280        CALL DMPOSITION          ;COMPUTE MAXIMUM DISK POSITION
1190  05DA 8AE8                      MOV CH,AL                ;CH = MAX DISK POS
1191  05DC A0E407                    MOV AL,DMINX             ;AL = CURRENT DISK POS
1192  05DF 3AC5                      CMP AL,CH                ;IS CURR POS = MAX POS?
1193  05E1 8AD0                      MOV DL,AL
1194  05E3 741E          0603        JZ CHECK_NPR5            ;YES
1195  05E5 8AC8                        MOV CL,AL
1196  05E7 51                          PUSH CX                ;CL = CURR POS, CH = MAX POS
1197  05E8 B500                        MOV CH,0
1198  05EA E8AAFC        0297          CALL GET_DM            ;BX = BLOCK NUMBER (CURR POS)
1199                           CHECK_NPR4:
1200  05ED 53                          PUSH BX
1201  05EE 41                          INC CX                 ;CURR POS = CURR POS + 1
1202  05EF E8A5FC        0297          CALL GET_DM            ;GET NEXT BLOCK(CURR POS)
1203  05F2 5A                          POP DX
1204  05F3 42                          INC DX
1205  05F4 3BDA                        CMP BX,DX              ;DOES CURR BLK = LAST BLK + 1?
1206  05F6 74F5          05ED          JZ CHECK_NPR4          ;YES
1207
1208  05F8 FEC9                        DEC CL                 ;CL = INDEX OF LAST SEQ BLK
1209  05FA 5A                          POP DX
1210  05FB 8AC6                        MOV AL,DH              ;AL = MAX POS
1211  05FD 3AC1                        CMP AL,CL              ;IS AL < LAST SEQ BLK POS
1212  05FF 7202          0603          JC CHECK_NPR5          ;YES
1213  0601 8AC1                          MOV AL,CL
1214                           CHECK_NPR5:                    ;AL = LAST BLK POS
1215  0603 2AC2                      SUB AL,DL                ;AL = AL - STARTING POS
1216  0605 8AE8                      MOV CH,AL
1217  0607 FEC5                      INC CH                   ;CH = # OF CONSECUTIVE BLOCKS
1218  0609 A0D507                    MOV AL,BLKMSK
1219  060C FEC0                      INC AL                   ;AL = BLKMSK + 1
```

```
1220
1221  060E F6E5                       MUL CH                          ;AL = # OF CONSECUTIVE LOGICAL RECS
1222  0610 59                         POP CX                          ;CL = MAXIMUM # OF RECORDS
1223  0611 86C1                       XCHG AL,CL
1224  0613 3AC1                       CMP AL,CL                       ;IS MAX < # OF CONS. RECS?
1225  0615 7202           0619        JC CHECK_NPR9                   ;YES
1226  0617 8AC1                       MOV AL,CL
1227                            CHECK_NPR9:                           ;AL = # OF CONSECUTIVE RECS
1228  0619 59                         POP CX
1229  061A 5B                         POP BX
1230  061B 882F                       MOV [BX],CH                     ;RESTORE VRECORD
1231  061D 59                         POP CX
1232  061E 8A368407                   MOV DH,MULT_NUM
1233  0622 2AC5                       SUB AL,CH                       ;AL = AL - VRECORD & blkmsk
1234  0624 3AC6                       CMP AL,DH                       ;IS AL < MULT_NUM
1235  0626 7202           062A        JC CHECK_NPR10                  ;YES
1236  0628 8AC6                         MOV AL,DH
1237                            CHECK_NPR10:                          ;AL = # OF CONSECUTIVE RECS
1238  062A F6D1                       not CL
1239  062C 22C1                       AND AL,CL                       ;IF DIR_CNT = 0 THEN
1240  062E 740E           063E        JZ RET13B                       ;RETURN WITH Z FLAG SET, C FLAG RESET
1241  0630 A28307                     MOV DIR_CNT,AL                  ;DIR_CNT = AL & ~PHYMSK
1242  0633 8A0EE107                   MOV CL,PHYSHF
1243  0637 D2E8                       SHR AL,CL                       ;AL = # OF CONSECUTIVE PHYSICAL RECS
1244  0639 A2B507           <         mov mult_sec,al                 ;save multisector count for r/w
1245  063C 0C01             '         OR AL,1                         ;RETURN WITH C AND Z FLAGS RESET
1246  063E C3                 RET13B: RET
1247
1248                                  tmp_select:
1249                                  ;----------
1250                                  ;       entry:  DL = drive to select (0-f)
1251
1252  063F 8816C908                   mov seldsk,dl
1253
1254                                  curselect:
1255                                  ;---------
1256  0643 A0C908                     mov al,seldsk
1257  0646 3A06C407                   cmp al,curdsk
1258  064A 7505           0651        jne select                      ;don't select if seldsk = curdsk
1259  064C FEC07405       0655          inc all jz select0
1260  0650 C3                         ret
1261
1262                                  SELECT:         ;select disk in info_fcb for subsequent input or output ops
1263                                  ;------
1264  0651 3C107203       0658        cmp al,16 ! jb disk_select
1265                                  select0:
1266  0655 E9A4F8         01FC            jmp selerror
1267                                  disk_select:
1268                                  ;-----------
1269                                  ;       entry:  AL = disk to select
1270
1271  0658 A2E907                     mov adrive,al
1272  0658 A2C407                     mov curdsk,al
```

```
1273
1274   065E 33D2                      xor dx,dx                       ;dl = 1 if drive logged in
1275   0660 E886FB       0219         call selectdisk
1276   0663 73F0         0655         jnc select0                     ;carry set if select ok
1277   0665 C3                        ret
1278
1279                              parsave33r:
1280                              ;----------
1281   0666 E812FB       017B         call parsave33
1282                              ;    jmps reselect
1283
1284                              reselect:
1285                              ;--------
1286   0669 B17F                      mov cl,07fh
1287   066B BB8D08                    mov bx,offset info_fcb+f7
1288   066E 200F                      and [bx],cl                     ;fcb(7) = fcb(7) & 7fh
1289   0670 204F01                    and 1[bx],cl
1290   0673 80670F1F                  and byte ptr extnum-f7[bx],1fh  ;fcb(ext) = fcb(ext) & 1fh
1291                                        ;check current fcb to see if reselection necessary
1292   0677 C6068207FF                mov resel,true                  ;mark possible reselect
1293   067C A08608                    mov al,info_fcb                 ;drive select code
1294   067F A2AE07                    mov fcbdsk,al                   ;save drive
1295   0682 241F                      and al,00011111b                ;non zero is auto drive select
1296   0684 FEC8                      dec al                          ;drive code normalized to
1297                                                                  ;0...30, or 255
1298   0686 3CFF                      cmp al,0ffh
1299   0688 7403         068D         jz noselect                     ;auto select function,
1300   068A A2C908                      mov seldsk,al                 ;save seldsk
1301                              noselect:
1302   068D E883FF       0643         call curselect
1303                                                                  ;set user code
1304   0690 A0CA08                    mov al,usrcode                  ;0...31
1305   0693 A28608                    mov info_fcb,al
1306   0696 C3                        ret
1307
1308                              compare:        ;compare strings
1309                              ;-------
1310                              ;       entry:  CL = length of strings
1311                              ;               BX,DX = offset of strings
1312                              ;       exit:   Z flag set if strings match
1313
1314   0697 B500                      mov ch,0
1315   0699 8BF3                      mov si,bx
1316   069B 8BFA                      mov di,dx
1317   069D F3A6                      repe cmps al,al
1318   069F C3                        ret
1319
1320
1321                              ;       individual function handlers
1322                              ;-----------------------------------------------------
1323
1324                              func14:         ;select disk info
1325                              ;======
```

```
1326
1327   06A0 E89CFF       063F          call tmp_select
1328   06A3 A0C908                     mov al,seldsk
1329   06A6 A2BF07                     mov p_dsk,al
1330   06A9 C3                         ret
1331
1332                             func15:        ;open file
1333                             ;======
1334   06AA E889FF       0666          call parsave33r                 ;copy fcb from user seg.
1335   06AD C606940800                 mov info_fcb+modnum,0           ;fcb(modnum)=0
1336   06B2 E8A2FD       0457          call check_wild                 ;check for ?s in fcb
1337   06B5 E88FFD       0477          call open                       ;attempt to open fcb
1338   06B8 E80300       06BE          call openx                      ;returns if unsuccessful
1339   06BB 32C0                       xor al,al
1340   06BD C3                  ret30: ret                             ;no - open failed
1341
1342                             openx:
1343                             ;-----
1344   06BE E894FC       0355          call end_of_dir                 ;was open successful
1345   06C1 74FA         06BD          jz ret30                        ;no
1346   06C3 B8A608                     mov bx,offset info_fcb+nxtrec
1347   06C6 803FFF                     cmp b[bx],0ffh
1348   06C9 7505         06D0          jne openxa
1349   06CB A09308                     mov al,info_fcb+chksum
1350   06CE 8807                       mov [bx],al
1351                             openxa:
1352   06D0 5B                         pop bx                          ;discard return address
1353   06D1 B140                       MOV CL,40H
1354   06D3 C3                         ret
1355
1356                             func20:        ;read a file
1357                             ;======
1358   06D4 E88FFF       0666          call parsave33r
1359   06D7 E960FE       053A          jmp disk_read
1360                                   ;jmp goback
1361
1362                             set_aret:
1363                             ;--------
1364   06DA 8AC8                       mov cl,al                       ;save error index
1365   06DC A2B107                     mov byte ptr aret+1,al          ;aret+1 = extended errors
1366   06DF E82CFD       040E          call lret_eq_ff
1367
1368   06E2 E924FB       0209          jmp goback                      ;return physical & extended errors
1369
1370                                   ;       BLOCKING/DEBLOCKING BUFFER CONTROL BLOCK (BCB) FORMAT
1371
1372                                   ;           +-------+-------+-------+-------+-------+-------+
1373                                   ;       00h |  DRV  |        RECORD         | PEND  |  SEQ  |
1374                                   ;           +-------+-------+-------+-------+-------+-------+
1375                                   ;       06h |     TRACK     |     SECTOR    |  BUFFER_ADDR  |
1376                                   ;           +-------+-------+-------+-------+-------+-------+
1377                                   ;       0Ch |     LINK      |  PROCESS_OFF  |
1378                                   ;           +-------+-------+-------+-------+
```

```
1379
1380
1381                            READ_DEBLOCK:
1382                            ;------------
1383   06E5 B401                        mov ah,1                        ;AH = 1
1384   06E7 E80D00        06F7          CALL DEBLOCK_DTA
1385   06EA E925FC        0312          JMP SET_FCB
1386
1387                            DEBLOCK_DIR:
1388                            ;-----------
1389   06ED 8C1EED07                    MOV CUR_DMABASE,DS              ;BUFFERS IN SYSTEM DATA AREA
1390   06F1 881ECE07                    MOV BX,DIRBCBA
1391   06F5 EB0A          0701          jmps DEBLOCK                    ;PREVIOUS LOCATE CALL
1392
1393                            DEBLOCK_DTA:
1394                            ;-----------
1395   06F7 881ED007                    MOV BX,DTABCBA
1396   06FB C706ED0700 00               MOV CUR_DMABASE,0               ;get segment from BCB
1397
1398                            DEBLOCK:        ;BDOS BLOCKING/DEBLOCKING ROUTINE
1399                            ;-------
1400                            ;       entry:  Z flag reset -> get new BCB address
1401                            ;               AH = 1 -> READ COMMAND
1402                            ;                  = 2 -> WRITE COMMAND
1403                            ;                  = 3 -> LOCATE COMMAND
1404                            ;                  = 4 -> FLUSH COMMAND
1405                            ;                  = 5 -> DIRECTORY UPDATE
1406
1407   0701 88268707                    MOV DEBLOCK_FX,AH               ;SAVE DEBLOCK FX
1408   0705 8A0EE207                    mov cl,phymsk                   ;CL = PHYMSK
1409   0709 A0EA07                      MOV AL,BYTE PTR ARECORD
1410   070C 22C1                        AND AL,cl
1411   070E A2B807                      MOV PHY_OFF,AL                  ;PHY_OFF = LOW(ARECORD) & PHYMSK
1412   0711 F6D1                        not cl                          ;CL = ~PHYMSK
1413   0713 200EEA07                    and BYTE PTR ARECORD,cl         ;LOW(ARECORD)) = LOW(ARECORD)) & ~PHYMSK
1414   0717 8B1F                        mov bx,[bx]                     ;GET BCB ADDRESS
1415   0719 891E6907                    MOV CURBCBA,BX                  ;BX = 1st curbcb
1416   071D 88470A                      MOV ax,10[BX]                   ;dma address field - offset/segment
1417   0720 833EED0700                  cmp cur_dmabase,0               ;if cur_dmabase <> 0, deblocking is
1418   0725 7505          072C          jne deblock0                    ; for dir buf and addr is offset
1419   0727 A3ED07                        mov cur_dmabase,ax            ;else deblocking data buffer and
1420   072A 33C0                          xor ax,ax                     ; addr is segment, offset = 0
1421                            deblock0:
1422   072C A3EF07                      MOV CURDMA,ax                   ;save current buffer address
1423   072F E85200        0784          CALL DEBLOCK9                   ;BX=CURBCBA, DX=.ADRIVE, CL=4
1424   0732 803FFF7405    073C          cmp b[bx],0ffh| je deblock2
1425   0737 E85DFF        0697            CALL COMPARE                  ;DOES BCB(0-3) = ADRIVE || ARECORD?
1426   073A 7415          0751            JZ DEBLOCK45                  ;YES
1427                            DEBLOCK2:
1428   073C 881E6907                    mov bx,curbcba
1429   0740 C607FF                      mov b[bx],0ffh                  ;discard in case of error
1430   0743 B002                        MOV AL,2
1431   0745 E84600        078E          CALL DEBLOCK_IO                 ;READ PHYSICAL RECORD
```

```
1432
1433  0748 E83900       0784          CALL DEBLOCK9                   ;BX=CURBCBA, DX=.ADRIVE, CL=4
1434  074B E8C2FA       0210          CALL MOVE
1435  074E C60500                     MOV BEDIT,0                     ;CURBCBA->BCB(4) = 0
1436                            DEBLOCK45:
1437  0751 32C0                       xor al,al
1438  0753 8A26B807                   MOV ah,PHY_OFF
1439  0757 D1E8                       shr ax,1                        ;AX = phy_off * 080h
1440  0759 8B36EF07                   MOV SI,CURDMA
1441  075D 03F0                       ADD SI,AX                       ;SI = CURDMA + PHY_OFF*80H
1442  075F A0B707                     MOV al,DEBLOCK_FX
1443  0762 3C03                       CMP al,3                        ;IS DEBLOCK_FX = LOCATE?
1444  0764 7505         0768          JNZ DEBLOCK6                    ;NO
1445  0766 8936C607                     MOV BUFFA,SI                  ;YES
1446  076A C3                           RET
1447                            DEBLOCK6:
1448  076B B94000                     MOV CX,40H                      ;transfer 40h words
1449  076E 8B3EC508                   MOV DI,DMAAD
1450  0772 A1C708                     mov ax,dmabase
1451  0775 8B16ED07                   mov dx,cur_dmabase
1452  0779 1E06                       push ds! push es
1453  077B 8EDA                       mov ds,dx                       ;setup source segment
1454  077D 8EC0                       mov es,ax                       ;setup destination segment
1455  077F F3A5                       rep movsw                       ;transfer data
1456  0781 071F                       pop es! pop ds
1457  0783 C3                         RET
1458
1459                            DEBLOCK9:                               ;SETUP FOR MOVE OR COMPARE
1460  0784 8B1EB907                   MOV BX,CURBCBA
1461  0788 BAE907                     MOV DX,OFFSET ADRIVE
1462  078B B104                       MOV CL,4
1463  078D C3                         RET
1464
1465                            DEBLOCK_IO:
1466                            ;----------
1467                            ;       entry:  AL = 0 -> SEEK ONLY
1468                            ;                  = 1 -> WRITE
1469                            ;                  = 2 -> READ
1470
1471  078E 50                         PUSH AX
1472  078F E8CFFA       0261          CALL SEEK
1473  0792 58                         POP AX
1474  0793 FEC8                       DEC AL
1475  0795 7803         079A          JS deblk_io2
1476                            ;         JNZ deblk_io1
1477                            ;           mov cl,1
1478                            ;           JMP WRBUFF
1479                            ;deblk_io1:
1480  0797 E8BBFA       0255          CALL RDBUFF
1481                            deblk_io2:
1482  079A BEB807                     mov si,offset track
1483  079D 8B3EB907                   mov di,curbcba
1484  07A1 83C706                     add di,6                        ;MOVE TRACK & SECTOR
```

```
1485
1486   07A4 B90200                    mov cx,2                        ;TO SCB
1487   07A7 F3A5                      rep movsw
1488   07A9 C3                        ret
1489
1490
1491     07AA                 endcode equ     offset $
1492
1493                          ;*************** end bdos file system ***************
1494
1495                          ;****************** BDOS data area ******************
1496                          ;
1497                          ;
1498                                  dseg
1499                          ;       Variables in data segment:
1500                                  org     endcode
1501
1502   07AA                   iosentry        rw      0
1503   07AA 0009              biosco          dw      biosoffset      ;offset and
1504   07AC 0000              bioscs          dw      0               ;segment for callf to BIOS
1505
1506                          ;
1507                          ;*******************************************************
1508                          ;
1509                          ;*******************************************************
1510                          ;*
1511                          ;*      bdos file system data area
1512                          ;*
1513                          ;*******************************************************
1514
1515                          ;       variables in data segment:
1516                          ;
1517
1518                          ;the following variables are set to zero upon entry to file system
1519
1520   07AE 00                fcbdsk          db      0               ;disk named in fcb
1521   07AF 00                parcopfl        db      0               ;true if parameter block copied
1522   07B0 0000              aret            dw      0               ;adr value to return
1523     07B0                 lret            equ     byte ptr aret   ;low(aret)
1524   07B2 00                resel           db      0               ;reselection flag
1525   07B3 00                DIR_CNT         DB      0               ;DIRECT I/O COUNT
1526   07B4 00                MULT_NUM        DB      0               ;MULTI-SECTOR COUNT used in bdos
1527     0007                 zerolength      equ     (offset $)-(offset fcbdsk)
1528   07B5 01                mult_sec        db      1               ;multi sector count passed to xios
1529
1530   07B6 00                BLK_OFF         DB      0               ;RECORD OFFSET WITHIN BLOCK
1531                          ;
1532
1533   07B7 00                DEBLOCK_FX      DB      0               ;DEBLOCK FUNCTION #
1534   07B8 00                PHY_OFF         DB      0               ;RECORD OFFSET WITHIN PHYSICAL RECORD
1535   07B9 0000              CURBCBA         DW      0               ;CURRENT BCB OFFSET
1536
1537   07BB 0000              TRACK           DW      0       ;BLB RECORD'S TRACK
```

```
1538
1539   078D 0000              SECTOR          DW      0       ;REG RECORD'S SECTOR
1540
1541                          ;       seldsk,usrcode are initialized as a pair
1542   078F 00                p_dsk           db      0       ;selected disk num
1543   07C0 00                p_user          db      0       ;curr user num
1544
1545                          ;info           dw      0       ;info adr
1546   07C1 0000              srcha           dw      0       ;search adr
1547
1548                          ;the following variable order is critical
1549
1550                          ;variables copied from uda for mp/m
1551
1552                          ;variables included in fcb checksum for mp/m and cp/m     x
1553
1554                          ;variables used to access system lock list for mp/m         x
1555
1556                          ;dmaad          dw      0       ;dma offset             1
1557                          ;dmabase                dw      0       ;dma base               2
1558                          ;srchl          db      0       ;search len             4
1559
1560                          ;dcnt           dw      0       ;directory counter      7
1561                          ;dblk           dw      0       ;directory block        8 ?? - not used - ??
1562   07C3                   u_mult_cnt      rb      0
1563   07C3 01                mult_cnt        db      1       ;bdos multi-sector cnt 10
1564                          ;df_password    rb      8       ;process default pw    11
1565
1566                          ;high_ext       db      0       ;fcb high extent bits   2
1567                          ;xfcb_read_only db      0       ;xfcb read only flag    3
1568   07C4 FF                curdsk          db      0ffh    ;current disk           4 1
1569
1570                                  org ((offset $) + 1) and 0fffeh
1571
1572                          ;       curtrka - alloca are set upon disk select
1573                          ;       (data must be adjacent)
1574
1575                          ;cdrmaxa                dw      0       ;ptr to cur dir max val
1576                          ;drvlbla                dw      0       ;drive label data byte addr
1577   07C6 0000              buffa           dw      0       ;ptr to dir dma addr
1578   07C8 0000              dpbaddr         dw      0       ;curr disk param block adar
1579   07CA 0000              checka          dw      0       ;curr checksum vector addr
1580   07CC 0000              alloca          dw      0       ;curr alloc vector addr
1581   07CE 0000              DIRBCBA         DW      0       ;DIRECTORY BUFFER CONTROL BLOCK ADDR
1582   07D0 0000              DTABCBA         dw      0       ;DATA BUFFER CONTROL BLOCK ADDR
1583     000A                 addlist         equ     10      ;"$-buffa" = addr list size
1584
1585                          ;       sectpt - offset obtained from disk parm block at dpbaddr
1586                          ;       (data must be adjacent)
1587
1588   07D2 0000              sectpt          dw      0       ;sectors per track
1589   07D4 00                blkshf          db      0       ;block shift factor
1590   07D5 00                blkmsk          db      0       ;block mask
```

```
1591
1592  07D6 00           extmsk      db      0       ;extent mask
1593  07D7 0000         maxall      dw      0       ;max alloc num
1594  07D9 0000         dirmax      dw      0       ;max dir num
1595  07DB 0000         dirblk      dw      0       ;reserved alloc bits for dir
1596  07DD 0000         chksiz      dw      0       ;size of checksum vector
1597  07DF 0000         offsetv     dw      0       ;offset tracks at beginning
1598  07E1 00           PHYSHF      DB      0       ;PHYSICAL RECORD SHIFT FACTOR
1599  07E2 00           PHYMSK      DB      0       ;PHYSICAL RECORD MASK
1600  07E3              endlist     rs      0       ;end of list
1601    0011            dpblist     equ     (offset endlist)-(offset sectpt)
1602                                                ;size
1603
1604                    ;       local variables
1605
1606  07E3 00           save_mod    db      0       ;open_reel module save field
1607
1608  07E4 00           dminx       db      0       ;local for diskwrite
1609  07E5 00           single      db      0       ;set true if single byte
1610                                                ;alloc map
1611  07E6 00           rcount      db      0       ;record count in curr fcb
1612  07E7 00           extval      db      0       ;extent num and extmsk
1613  07E8 00           vrecord     db      0       ;curr virtual record
1614  07E9 00           ADRIVE      DB      0       ;CURRENT DISK - must preceed arecord
1615  07EA 0000         arecord     dw      0       ;curr actual record
1616  07EC 00                       db      0       ;curr actual record high byte
1617  07ED 0000         CUR_DMABASE DW      0
1618  07EF 0000         CUR_DMA     DW      0
1619
1620                    ;       local variables for directory access
1621
1622  07F1 00           dptr        db      0       ;directory pointer 0,1,2,3
1623
1624                    ;       shell variables
1625
1626  07F2 0000         shell_si    dw      0       ;bdos command offset
1627  07F4 0000         shell_dma   dw      0       ;dmaad save area
1628  07F6 00           shell_flag  db      0       ;parsave shell flag
1629
1630                    ;       special 8086 variables:
1631
1632  07F7 0000         u_retseg    dw      0       ;user return segment
1633
1634  07F9 00           parlg       db      0       ;len of parameter block
1635
1636
1637                    ;       bdos stack switch variables and stack
1638                    ;       used for all bdos disk functions
1639
1640                            org     ((offset $) + 1) and 0fffeh
1641
1642                                    ; 69 word bdos stack
1643
```

```
1644
1645   07FA CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
1646   0800 CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
1647   0806 CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
1648   080C CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
1649   0812 CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
1650   0818 CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
1651   081E CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
1652   0824 CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
1653   082A CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
1654   0830 CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
1655   0836 CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
1656   083C CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
1657   0842 CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
1658   0848 CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
1659   084E CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
1660   0854 CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
1661   085A CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
1662   0860 CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
1663   0866 CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
1664   086C CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
1665   0872 CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
1666   0878 CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
1667   087E CCCCCCCCCCCC                    dw      0cccch,0cccch,0cccch
1668   0884                    bdosstack    rw      0
1669
1670   0884                    save_sp      rw      1
1671
1672                           ;       local buffer area:
1673
1674   0886                    info_fcb     rb      0;40    ;local user FCB
1675
1676   0886 434F50595249               db      "COPYRIGHT(C)1983,"
1677        474854284329
1678        313938332C
1679   0897 444947495441               db      "DIGITAL RESEARCH(01/26/83)"
1680        4C2052455345
1681        415243482830
1682        312F32362F38
1683        3329
1684   08B1 585858582D30               db      "XXXX-0000-654321"
1685        3030302D3635
1686        34333231
1687
1688                           ;
1689   08C1 0000               info         dw      0               ;information address
1690                           ;
1691   08C3                    u_wrkseg     rw      0
1692   08C3 0000               parametersegment dw  0       ;user parameter segment
1693                           ;
1694                           ;
1695   08C5 0000               dmaad   dw      0               ;user DMA address
1696   08C7 0000               dmabase dw      0               ;user DMA base
```

```
1697
1698   08C9 00                  seldsk  db      0               ;current user disk
1699   08CA 00                  usrcode db      0               ;current user number
1700   08CB 0000                dcnt    dw      0               ;directory index
1701     08CB                   ldcnt   equ     byte ptr dcnt   ;low(dcnt)
1702   08CD 00                  srchl   db      0               ;search length
1703                            ;
1704                            ;
1705
1706   08CE 00                          db      0               ;end of this data area
1707
1708                            ;****************** end BDOS data area ****************
1709
1710                            ;********************* end of BDOS ********************
1711                            end
1712
1713
1714  END OF ASSEMBLY.  NUMBER OF ERRORS:   0.  USE FACTOR: 16%
```

```
ADDLIST          000A N   513  1583#
ADRIVE           07E9 V  1271  1461  1614#
ALLOCA           07CC V  1580#
ARECORD          07EA V   432   433   549   551   617   624   634   636   687   688
                        1409  1413  1615#
ARET             0780 V   299   320   334   473  1365  1522# 1523
ATRAN            02C1 L   621# 1107
B                0000 N    33#  777   990   993   996  1036  1075  1171  1347  1424
                        1429  1435
BADFUNC          0096 L   207#
BDOENTRY         0000 L   196   199   202   276#
BDOFUNC          00C7 N   195   198   201   270#
BDOSE1           0048 L   177   179#
BDOSOFFSET       0000 N    21#  117
BDOSRETURN       0163 L   348#  476
BDOSSTACK        0884 V   132  1668#
BFLSHELL         0002 N   265#  294
BFLGETMX         0001 N   264#
BIOSCO           07AA V   135  1503#
BIOSCS           07AC V   133  1504#
BIOSOFFSET       0900 N    23#  135  1503
BLKMSK           07D5 V   631  1174  1218  1590#
BLKOFF           0786 V   632  1134  1530#
BLKSHF           07D4 V   566   623   795  1589#
BTABADDR         0000 N   262#  263   347
BTABFLAG         0002 N   263#  294   308
BUFFA            07C6 V   679  1445  1577#
CALLBDOS         0157 L   297   318   340#
CHECKA           07CA V  1579#
CHECKNPR1        0587 L  1137  1142#
CHECKNPR10       062A L  1235  1237#
CHECKNPR11       059A L  1147  1156#
CHECKNPR1X       0590 L  1148# 1161
CHECKNPR1Y       0597 L  1150  1153#
CHECKNPR2        0582 L  1166  1169#
CHECKNPR23       05D0 L  1182  1184#
CHECKNPR4        05ED L  1199# 1206
CHECKNPR5        0603 L  1194  1212  1214#
CHECKNPR9        0619 L  1187  1225  1227#
CHECKNPRS        0575 L  1108  1123#
CHECKWILD        0457 L   912# 1336
CHKSIZ           07DD V  1596#
CHKSUM           000D N    91#  872  1349
CHKWILD          0464 L   915   920#
CHKWILD1         0467 L   926#  933
COMPARE          0697 L  1308# 1425
COMPEXT          0388 L   746#  892  1025
CS               SREG V   128   133   139   170   294   308   347
CURBCBA          0789 V  1415  1428  1460  1483  1535#
CURDMA           07EF V   442   699  1422  1440  1618#
CURDMABASE       07ED V   441   697  1389  1396  1417  1419  1451  1617#
CURDSK           07C4 V   434   469  1257  1272  1568#
CURSELECT        0643 L  1254# 1302
```

```
DCNT            08CB V   685   706   715   722   724  1700# 1701
DEBLK102        079A L  1475  1481#
DEBLOCK         0701 L  1391  1398#
DEBLOCK0        072C L  1413  1421#
DEBLOCK2        073C L  1424  1427#
DEBLOCK45       0751 L  1426  1436#
DEBLOCK6        076B L  1444  1447#
DEBLOCK9        0784 L  1423  1433  1459#
DEBLOCKDIR      06ED L   691  1387#
DEBLOCKDTA      06F7 L  1384  1393#
DEBLOCKFX       07B7 V  1407  1442  1533#
DEBLOCKIO       078E L  1431  1465#
DIDCOMP         025A L   540#
DIRBCBA         07CE V  1390  1581#
DIRBLK          07DB V  1595#
DIRCNT          07D3 V  1135  1139  1241  1525#
DIRMAX          07D9 V   721  1594#
DIRREC          0004 N    66#   68
DISKEOF         0572 L  1095  1099  1106  1119#
DISKREAD        053A L  1082# 1359
DISKSELECT      0658 L  1264  1267#
DMAAD           08C5 V   214   307   328   335   698  1449  1695#
DMABASE         08C7 V   246   696  1450  1696#
DMINX           07E4 V   613   783   994  1191  1608#
DMPOSITION      0280 L   562#  612  1189
DPBADDR         07C8 V   512   515  1578#
DPBLIST         0011 N   517  1601#
DPTR            07F1 V   677   738  1622#
DS              SREG V   129   137   137   140   169   170   180   187   377   378
                         380   447  1389  1452  1453  1456
DSKMAP          0010 N    94#  595
DSKMSK          0003 N    68#  733
DSKSHF          0002 N    67#  686
DTABCBA         07D0 V  1395  1582#
ENDCODE         07AA N  1491# 1500
ENDDIR          FFFF N    59#  715
ENDLIST         07E3 V  1600# 1601
ENDOFDIR        0355 L   702#  841  1344
ENDSEARCH       044D L   866   905#
ES              SREG V   130   171   180   186   392   393   395   420   422   447
                        1452  1454  1456
EXTMSK          07D6 V   655   753   799  1592#
EXTNUM          000C N    90#  656   805   876  1021  1035  1057  1290
EXTVAL          07E7 V   572   657  1612#
F7              0007 N    82#  1287  1290
FALSE           0000 N    29#  336
FCBDSK          07AE V   287   351  1294  1520# 1527
FCBLEN          0020 N    63#   66    95
FCBSHF          0005 N    69#  734
FN1             006C L   194   197#
FN2             0077 L   197   200#
FN3             0082 L   200   203#
FUNC            0061 L   183   192#
```

```
FUNC14          06A0 L   271  1324#
FUNC15          06AA L   272  1332#
FUNC20          06D4 L   275  1356#
FUNC26          009A L   203   211#
FUNC32          009F L   204   217#
FUNC44          00B0 L   205   230#
FUNC51          00C2 L   206   244#
GETDE1          039F L   774#  780   801
GETDE2          03AD L   778   782#
GETDE3          03BC L   786   793#
GETDIREXT       0399 L   646   763#  956  1071
GETDM           0297 L   590#  616  1198  1202
GETDMD          02A8 L   598   602#
GETDPTRA        0329 L   671#  847   948
GETFCB          02E6 L   640# 1049  1086
GETFCB0         02FB L   645   649#
GETFCB1         0304 L   651   653#
GOBACK          0209 L   474# 1368
GOERR           0203 L   465   471#
INDEX           02AD L   607# 1101
INFO            08C1 V   290   317   375   391  1689#
INFOFCB         0886 V   352   376   390   595   642   644   650   656   664   665
                         668   771   805   815   914   950   974  1004  1009  1019
                        1021  1057  1287  1293  1305  1335  1346  1349  1674#
INTERRUPTBIT    0200 N   101#  176
IOREAD          000A N    50#  537
IOSELDSK        0009 N    49#  505
IOSENTRY        07AA V   134   421   445  1502#
LDCNT           08CB V   732  1701#
LDRENTRY        0906 L   115#  141
LDROFFSET       0906 N    22#  114
LRET            07B0 V   401   907  1098  1523#
LRETEQFF        040E L   858# 1366
MAXALL          07D7 V   522  1593#
MAXEXT          001F N    70#  759   808  1028
MAXMOD          003F N    71# 1040
MODNUM          000E N    92#  879  1019  1035  1335
MOVE            0210 L   483#  952  1434
MULTCNT         07C3 V   292   312   323  1563#
MULTI01         0115 L   313#  330
MULTNUM         07B4 V   314  1159  1232  1526#
MULTSEC         07B5 V   436  1244  1528#
NAMLEN          000F N    72#  822
NOSELECT        068D L  1299  1301#
NOSHELL         00F3 L   292   296#
NOSHELLERR      0135 L   322   327#
NXTREC          0020 N    95#  642   664   771   951  1346
OFFSETV         07DF V   553  1597#
OPEN            0477 L   937# 1337
OPENCOPY        047C L   942# 1047
OPENERR         050E L  1041  1046  1055#
OPENR0          04FB L  1031  1044#
OPENR2          0503 L  1048# 1080
```

```
OPENR3              0522 L  1027  1069#
OPENR4              0532 L  1074  1077#
OPENREEL            04D1 L  1013#  1096
OPENX               068E L  1338  1342#
OPENXA              06D0 L  1348  1351#
OSENTRY             0003 L   120#   133
OSINIT              0000 L   118#
PARAMETERSEGMENT    08C3 V   378   393  1692#
PARCOPFL            07AF V   354   371  1521#
PARLG               07F9 V   373   388  1634#
PARRET              019E L   370   381#  387
PARSAVE             017D L   365#
PARSAVE33           017B L   309   360#  1281
PARSAVE33R          0666 L  1279#  1334  1358
PARUNSAVE           019F L   337   356   384#
PDSK                078F V   283  1329  1542#
PERERROR            01F8 L   462#   545
PHYMSK              07E2 V  1143  1408  1599#
PHYOFF              0788 V  1411  1438  1534#
PHYSHF              07E1 V   519   555  1242  1598#
PUSER               07C0 V   223   227  1543#
RCOUNT              07E6 V   654   667  1088  1179  1611#
RDBUFF              0255 L   535#  1115  1480
RDDIR               0334 L   683#   741
READDEBLOCK         06E5 L  1111  1381#
READDIR             0363 L   718#   840
RECCNT              000F N    93#   644   650   665   668   974  1004  1009
RECORDOK            0554 L  1090  1100#
RECORDOK2           056F L  1109  1116#
RECSIZ              0080 N    65#    66
RESEL               07B2 V   349  1292  1524#
RESELECT            0669 L  1284#
RESTORERC           04C4 L   999#  1078
RESTORERC1          04D0 L  1006  1010#
RET10               0476 L   916   931   935#  940
RET11               04B1 L   987#  991   995
RET13B              063E L  1240  1246#
RET30               068D L  1340#  1345
RET4                0254 L   509   532#  544
RET41               0328 L   666   669#
RET71               0387 L   740   742#
RETMON              00F6 L   298#  338
RETMON5             0170 L   350   353#
RETMON6             017A L   355   357#
RETSELECT           024E L   524   527#
RETURNNOTOK         00C0 L   235   237   240#
RWXIOSIF            01CA L   427#  538
SAVEMOD             07E3 V  1020  1058  1606#
SAVESP              0884 V   346   475  1670#
SEARCH              03E9 L   825#
SEARCHI             03DB L   813#  829
SEARCHN             03EF L   834#  856   903
SEARCHNAME          03E7 L   820#  939  1045
```

```
SEARCHNJMP      044A L   885   894   901#
SECTOR          07BD V   440   557  1539#
SECTPT          07D2 V   516   520   552  1588# 1601
SEEK            0261 L   547# 1113  1472
SELDSK          08C9 V   234  1252  1256  1300  1328  1693#
SELECT          0651 L  1258  1262#
SELECTO         0655 L  1259  1265# 1276
SELECTDISK      0219 L   495# 1275
SELERROR        01FC L   467# 1266
SETARET         06DA L   918  1362#
SETDATA         0348 L   345   694# 1112
SETENDDIR       035C L   713#  728   830
SETFCB          0312 L   660# 1117  1385
SETLRET1        018C L   398# 1067  1076  1120
SETRC           049C L   648   968# 1079
SETRC1          04BF L   980   985#
SETRC2          04B2 L   977   989#
SETUSRCODE      00AA L   222   225#
SHELL           00FB L   295   302#
SHELL03         0142 L   326   333#
SHELLDMA        07F4 V   307   335  1627#
SHELLFLAG       07F6 V   310   336   369   386  1628#
SHELLSI         07F2 V   306   316  1626#
SINGLE          07E5 V   529   597   784  1609#
SRCHA           07C1 V   816   846  1545#
SRCHEXT         0439 L   877   888#
SRCHFIN         040E L   842   857#
SRCHL           08CD V   817   848  1702#
SRCHLOOP        0417 L   855   864#  899
SRCHOK          0442 L   873   886   896#
SS              SREG V   131   176
STARET          01BE L   400#  862  1052
SYSDAT          0006 V   123#
TESTFFFF        0358 L   708#
TMPSELECT       063F L  1248# 1327
TRACK           07BB V   439   554  1482  1537#
TRUE            FFFF N    28#   29   310   354   369   371   386   784  1292
UMULTCNT        07C3 V   238  1562#
URETSEG         07F7 V   171   186  1632#
USERENTRY       0034 L   121   146#
USERINIT        0008 L   119   126#
USRCODE         08CA V  1304  1699#
UWRKSEG         08C3 V   173   187  1691#
VRECORD         07E8 V   567   630   643   663  1051  1087  1162  1613#
W               0000 N    34#   710
XIOSIF          01C2 L   413#  506
ZEROLENGTH      0007 N   286  1527#
```