```
;
; BDOS      CP/M REV. 2.2
;

;                       N O T E

; THIS BDOS IS A MODIFICATION OF THE DRI CP/M 2.2 BDOS.
; IT IS BUILT ON THE BDOS22.ASM FILE, A DISASSEMBLY OF
; THE DRI OBJECT FILE, SAID TO HAVE ORIGINATED AT
; BERKELEY CALIFORNIA.
; SOME Z80 OP-CODES ARE USED, MOSTLY RELATIVE JUMPS, TO
; REDUCE OBJECT CODE SIZE. THE SPACE SAVINGS ARE USED
; BY A SMALL AMOUNT OF CODE THAT ADDS 6 BITS TO THE
; FILE SYSTEM RECORD COUNTER. THIS ALLOWS DRIVES OF
; UP TO 512 MBYTES TO BE SUPPORTED (AS DOES MP/M II).
; THE ORIGINAL BDOS USED 16 BITS FOR RECORD MANAGEMENT.
; AT 128 BYTES PER RECORD, DRIVE AND FILE SIZES WERE
; THUS LIMITED TO 8 MBYTES MAXIMUM.
;
; THIS BDOS ALSO SUPPORTS MP/M II STYLE RECORD PASSING
; ON DISK READ AND WRITES, FROM "GETRECS".
;
; AN MP/M 1.1 VERSION OF THIS BDOS IS AVAILABLE, KNOWN
; AS BDOS17. IT MUST BE ASSEMBLED AND CONVERTED TO
; AN .SPR FILE FOR USE WITH MP/M 1.1 SYSGEN.
;
; CODE SECTIONS THAT IMPLEMENT THIS FEATURE ARE PRECEDED BY:
;
; ;++++++++ < BDOS CHANGE > ++++++++++++++++++++++++++++++
;



MACLIB      Z80




MSIZE EQU  56           ;Memory size
;DOSLOC     EQU  (MSIZE-20)*1024+3C00H    ;Where BDOS resides
;DOSLOC     EQU  (MSIZE-20)*1024+3500H    ;Where TT BDOS resides
;

DOSLOC      EQU   0DC00H                  ;0D800H


BOOTV EQU  0000H  ;Vector to BIOS warm boot routine
IOBYTE      EQU  0003H  ;System I/O device assignment
TBUFF EQU  0080H  ;Default DMA buffer location
;
FALSE EQU   0           ;
TRUE  EQU   1           ;
;
CR    EQU   0DH         ;
```

```
LF    EQU   0AH         ;
TAB   EQU   09H         ;Tab
BACKSP      EQU   08H         ;Backspace
CTRLC EQU   03H         ;Control-C
CTRLE EQU   05H         ;Control-E
CTRLP EQU   10H         ;Control-P
CTRLR EQU   12H         ;Control-R
CTRLS EQU   13H         ;Control-S
CTRLU EQU   15H         ;Control-U
CTRLX EQU   18H         ;Control-X
DELDAT      EQU   0E5H        ;Deleted data byte
;
      ORG   DOSLOC


; BIOS ENTRY POINTS

CBOOT EQU   DOSLOC+0E00H      ;(BOOT) Cold boot entry
WBOOTF:     EQU   DOSLOC+0E03H      ;Warm boot entry
CONSF:      EQU   DOSLOC+0E06H      ;Console status
CONIF:      EQU   DOSLOC+0E09H      ;Console char in
CONOF:      EQU   DOSLOC+0E0CH      ;Console char out
LISTF:      EQU   DOSLOC+0E0FH      ;List char out
PUNF: EQU   DOSLOC+0E12H      ;Punch char out
READF:      EQU   DOSLOC+0E15H      ;Reader char in
HOMF: EQU   DOSLOC+0E18H      ;Home disk
SELF: EQU   DOSLOC+0E1BH      ;Select disk
TRKF: EQU   DOSLOC+0E1EH      ;Set disk track addr
SECF: EQU   DOSLOC+0E21H      ;Set disk sector addr
DMAF: EQU   DOSLOC+0E24H      ;Set DMA buffer addr
DRDF: EQU   DOSLOC+0E27H      ;Read sector
DWRF: EQU   DOSLOC+0E2AH      ;Write sector
LISTST:     EQU   DOSLOC+0E2DH      ;PRINTER STATUS
SECTRN:     EQU   DOSLOC+0E30H      ;Sector translation routine
;




;
SERIAL:     DB    2,16H,0,0,11H,60H ;Room for serial number
;
;*********************************************************
;bdos entry point. data enters as follows:
;           C  = command
;           DE = address or 16 bit data word
;           E  = 8 bit data word
;
;      return from bdos data is as follows:
;           A  = status or value
;           HL = 16 bit value or address
;
ENTRY:      JMP   BDOS        ;Entry to disk monitor
```

```
;
;*****************************************************************
*******
; BDOS error message table - used by other CP/M utilities perhaps
;
PERERR:     DW    PERSUB            ;Permanent error subroutine
SELERR:     DW    SELSUB            ;Select error subroutine
ROERR:      DW    ROSUB       ;Read-only subroutine
RONERR:     DW    RONSUB            ;Read-only error subroutine
;
;*****************************************************************
*******
;bdos entry saves input data as follows:
;    DE -> FCB          0  -> OLDDSK
;    E  -> IDATA  0  -> OUT1
;    0  -> ODATA  sets return to BEXIT on stack
;    SP -> OLDSP
;
;    saves input data and jumps to command routine
;
BDOS: XCHG              ;swap DE and HL - entry FCB is now in HL
      SHLD  FCB         ;save data in FCB for now
      XCHG              ;put registers back
      MOV   A,E         ;move data byte into A
      STA   IDATA       ;save it
      LXI   H,0         ;zero out HL
      SHLD  ODATA       ;save this for the return
      DAD   SP          ;add in the user stack pointer
      SHLD  OLDSP       ;save this for the return also
      LXI   SP,USRCOD   ;point to BDOS stack
      XRA   A           ;zero out a
      STA   OLDDSK            ;save this as the old disk
      STA   OUT1        ;and at OUT1 for disk relog on exit
      LXI   H,BEXIT           ;get the return point address
      PUSH  H           ;and stuff it on the stack for exiting
      MOV   A,C         ;move the command into A
      CPI   41          ;Max cmds +1 - is it too big?
      RNC               ;return if so
      MOV   C,E         ;get input data from E into C
      LXI   H,DISTBL    ;point to the dispatch table
      MOV   E,A         ;move the command from A into E
      MVI   D,0         ;zero out D
      DAD   D           ;add in the dispatch table
      DAD   D           ;once more - over two bytes
      MOV   E,M         ;pull the jump address
      INX   H           ;  into
      MOV   D,M         ;    DE
      LHLD  FCB         ;get the info word into HL
      XCHG              ;swap DE and HL
      PCHL              ;jump to the command routine
;
;*****************************************************************
*******
; command dispatch table
```

```
;
DISTBL:     DW    WBOOTF            ; 0: System reset
      DW    REDCON            ; 1: Console input
      DW    WRTCON            ; 2: Console output
      DW    REDRDR            ; 3: Reader input
      DW    PUNF        ; 4: Punch output
      DW    LISTF       ; 5: List output
      DW    DIRTIO            ; 6: Direct console I/O
      DW    GETIOB            ; 7: Get I/O Byte
      DW    PUTIOB            ; 8: Set I/O Byte
      DW    PRNBUF            ; 9: Print string
      DW    REDBUF            ;10: Read console buffer
      DW    GCSTAT            ;11: Get console status
      DW    GETVER            ;12: Return version number
      DW    RESET       ;13: Reset disk system
      DW    LOGIN       ;14: Select disk
      DW    OPEN        ;15: Open file
      DW    CLOSE       ;16: Close file
      DW    SEAR1       ;17: Search for first
      DW    SEARN       ;18: Search for next
      DW    DELETE            ;19: Delete file
      DW    READ        ;20: Read sequential
      DW    WRITE       ;21: Write sequential
      DW    CREATE            ;22: Make file
      DW    RENAME            ;23: Rename file
      DW    GLOGIN            ;24: Return login vector
      DW    GETDRV            ;25: Return current disk
      DW    DMASET            ;26: Set DMA address
      DW    GALLOC            ;27: Get addr (alloc)
      DW    MAKRO       ;28: Write protect disk
      DW    GROVEC            ;29: Get R/O vector
      DW    SETATT            ;30: Set file attributes
      DW    GETPAR            ;31: Get addr (disk parms)
      DW    MODUSR            ;32: Set/Get user code
      DW    REDRND            ;33: Read random
      DW    WRTRND            ;34: Write random
      DW    FILSIZ            ;35: Compute file size
      DW    SETRND            ;36: Set random record
      DW    RESDRV            ;37: Reset drive
      DW    JR          ;38: Undefined - go back
      DW    JR          ;39: Undefined - go back
      DW    ZERRND            ;40: Fill random file w/ zeros
;
;*****************************************************************
*******
; Print the permanent error message
;
PERSUB:     LXI   H,PERMSG    ;point to bad sector message
      CALL  DOSERR            ;print out with disk and return
      CPI   CTRLC       ;return with a char - is it ctrlc?
      JZ    BOOTV       ;if so go reboot
      RET               ;otherwise just go back
;
;*****************************************************************
```

```
*******
; Print the select error message then die
;
SELSUB:     LXI   H,SELMSG    ;'Select' msg
      JR    BOMB        ;say too bad then abort
;
;*****************************************************************
*******
; Print read only error message then die
ROSUB:      LXI   H,ROMSG           ;Send R/O msg
      JR    BOMB        ; say too bad then abort
;
;*****************************************************************
*******
; Print file read-only error message, die
;
RONSUB:     LXI   H,ROEMSG    ;'File R/O' msg
;
;*****************************************************************
*******
; Print out the error message then reboot
;
BOMB: CALL  DOSERR            ;Print BDOS err on [dsk #]
      JMP   BOOTV       ;Reboot system
;
;*****************************************************************
*******
; Error messages
;
DOSMSG:     DB    'DOS ERR '
DSKCH:      DB    ' : $'
PERMSG:     DB    'SECTOR$'
SELMSG:     DB    'SLCT$'
ROEMSG:     DB    'FILE '
ROMSG:      DB    'R/O$'
;
;*****************************************************************
*******
; Print BDOS error msg with disk #
;
DOSERR:     PUSH  H           ;save the message pointer
      CALL  CRLF        ;Turn up a new line
      LDA   CURDSK            ;get the current disk
      ADI   'A'         ;add in the ascii bias
      STA   DSKCH       ;Form drive ID char
      LXI   B,DOSMSG    ;Print the
      CALL  PRINT       ; 'Bdos Err on' msg
      POP   B           ;restore the message pointer
      CALL  PRINT       ;Print specific error message
;
;*****************************************************************
*******
; Read next console character
;
```

```
CONIN:      LXI   H,CHRRDY    ;Point to the char ready flag
      MOV   A,M         ;move the ready byte into A
      MVI   M,0         ;and zero out the flag
      ORA   A           ;was anyone home
      RNZ               ;return if not
      JMP   CONIF       ;otherwise go get the thing
;
;*****************************************************************
*******
; Read console character, echo print if it is ok
;
CIECHO:     CALL  CONIN       ;get a character
      CALL  GRAFIC            ;check the control characters
      RC                ;return if the char is not printable
      PUSH  PSW         ;otherwise save it
      MOV   C,A         ;put a copy into C
      CALL  WRTCON            ;and print it out
      POP   PSW         ;restore it
      RET               ;and return to sender
;
;*****************************************************************
*******
;see if A is a good control char, ret with CY set if not
;
GRAFIC:     CPI   CR          ;test for carriage return
      RZ                ;return if it was
      CPI   LF          ;how about line feed
      RZ                ;that is ok
      CPI   TAB         ;as is tab
      RZ                ;so back with it
      CPI   BACKSP            ;Backspace in 2.0
      RZ                ;is known
      CPI   ' '         ;test all other control characters
      RET               ;and go back with carry set for no echo
;
;*****************************************************************
*******
; Check for console break and abort
;
CONBRK:     LDA   CHRRDY            ;get the character ready byte
      ORA   A           ;and check for waiting char
      JRNZ  CB1         ;non zero says something is there
      CALL  CONSF       ;check the console if none there
      ANI   01H         ;and in the status bit
      RZ                ;return if it came up zero
      CALL  CONIF       ;otherwise go get the waiting char
      CPI   CTRLS       ;was it a control-s
      JRNZ  CB0         ;if not return with ready flag set
      CALL  CONIF       ;if it was wait for the next char
      CPI   CTRLC       ;is this one control c?
      JZ    BOOTV       ;if so we reboot
      XRA   A           ;or else zero a
      RET               ;and simply return
;
```

```
CB0:  STA   CHRRDY            ;Save input char
CB1:  MVI   A,TRUE            ;and say true to the caller
      RET               ;and go back
;
;*****************************************************************
*******
; Output char in C to console, list also if LSTCPY = true
;
CONOUT:     LDA   CTEMP1            ;get the first temp column counter
      ORA   A           ;is it zero?
      JRNZ  CONOU1            ;if nonzero jump over
      PUSH  B           ;otherwise save input character
      CALL  CONBRK            ;Check for abort
      POP   B           ;recover the char
      PUSH  B           ;save it again
      CALL  CONOF       ;and print the character on the console
      POP   B           ;recover once more
      PUSH  B           ;would you believe save it again
      LDA   LSTCPY            ;get the list flag byte
      ORA   A           ;to set flags
      CNZ   LISTF       ; and to list if LSTCPY=true
      POP   B           ;restore char one last time
CONOU1:     MOV   A,C         ;put the byte into A
      LXI   H,COLUMN    ;point to the buffer column counter
      CPI   7FH         ;Rubout ?
      RZ                ;return if so
      INR   M           ;Increment column
      CPI   ' '         ;is the character a space?
      RNC               ;return if less - must be control
      DCR   M           ;Decrement column
      MOV   A,M         ;and get the character there
      ORA   A           ;check for a null
      RZ                ;return if it is
      MOV   A,C         ;move the char into C
      CPI   BACKSP            ;was it a backspace?
      JRNZ  CONOU2            ;if not jump over
      DCR   M           ;Decrement column
      RET               ;and return
;
CONOU2:     CPI   LF          ;was it a linefeed?
      RNZ               ;return if not again
      MVI   M,0         ;Set column to 0
      RET               ;and return
;
;*****************************************************************
*******
; Print char in C at console, convert Ctrl chars to ^[char]
;
CTLOUT:     MOV   A,C         ;move the char into A
      CALL  GRAFIC            ;see if it is printable
      JNC   WRTCON            ;jump past cause it is ok
      PUSH  PSW         ;otherwise save ctrl char on the stack
      MVI   C,5EH       ;get the "^" char
      CALL  CONOUT            ;and print it
```

```
      POP   PSW         ;get the char back
      ORI   40H         ;add in the bias to make it print
      MOV   C,A         ;put it into C for the next
;
;*****************************************************************
*******
; BDOS function 2: write to the system console char in C
;
WRTCON:     MOV   A,C         ;which puts it into A
      CPI   TAB         ;check for tab
      JRNZ  CONOUT            ;if not jump and print
TABOUT:     MVI   C,' '       ;Space gets printed to expand tabs
      CALL  CONOUT            ; to
      LDA   COLUMN            ;  next
      ANI   07H         ;   tab
      JRNZ  TABOUT            ;    stop
      RET               ;Return
;
DELAST:     CALL  BACKUP            ;back up one character
      MVI   C,' '       ;then get a blank
      CALL  CONOF       ;and print it out
BACKUP:     MVI   C,BACKSP    ;get a backspace
      JMP   CONOF       ;and print it as well to move into spot
;
;*****************************************************************
*******
; Print pound sign, CRLF, and fix columns
;
LBCRLF:     MVI   C,'#'       ;get a pound sign
      CALL  CONOUT            ;and print it
      CALL  CRLF        ;Turn up a new line
LB1:  LDA   COLUMN            ;get the column counter
      LXI   H,CTEMP2    ;point to the temp counter
      CMP   M           ;are they equal?
      RNC               ;not there yet
      MVI   C,' '       ;so get a space
      CALL  CONOUT            ;and print it out
      JR    LB1         ;loop til the counts are the same
;
;*****************************************************************
*******
; Print CR/LF at console
;
CRLF: MVI   C,CR        ; get a cr
      CALL  CONOUT            ;and print it out
      MVI   C,LF        ;followed by a lf
      JMP   CONOUT            ;which goes out too
;
;*****************************************************************
*******
; Print string at (BC) until '$' with TAB expansion
;
PRINT:      LDAX  B           ;get the byte at (BC)
      CPI   '$'         ;is it the end mark?
```

```
      RZ                ;if so we are done
      INX   B           ;otherwise bump the pointer
      PUSH  B           ;and save it in the stack
      MOV   C,A         ;get the character into C
      CALL  WRTCON            ;write it out expanding tabs
      POP   B           ;recover the pointer
      JR    PRINT       ;and loop til we hit the stop
;
;*********************************************************************
*******
; BDOS function 10: Read console buffer at
;     enter with BC -> console buffer address
;
REDBUF:     LDA   COLUMN            ;get the column counter into A
      STA   CTEMP2            ;save it at CTEMP2
      LHLD  FCB         ;get the information into HL
      MOV   C,M         ;get the buffer count byte
      INX   H           ;and bump the pointer to the next spot
      PUSH  H           ;save it for later
      MVI   B,0         ;zero B
RB0:  PUSH  B           ;save BC for later
      PUSH  H           ;and HL as well
RB1:  CALL  CONIN       ;go get a char from the console
      ANI   7FH         ;strip the parity bit
      POP   H           ;recover HL
      POP   B           ;and BC
      CPI   CR          ;is this character a CR?
      JZ    RBEXIT            ;if so jump over
      CPI   LF          ;how about a LF
      JZ    RBEXIT            ;jump with that as well
      CPI   BACKSP            ;do we have a backspace?
      JRNZ  CHKRUB            ;if not over we go
      MOV   A,B         ;put B into A
      ORA   A           ;is it still zero?
      JRZ   RB0         ;if so go get another character
      DCR   B           ;if not decrement the count in B
      LDA   COLUMN            ;get the column counter
      STA   CTEMP1            ;save it at CTEMP1 for later
      JR    RB65        ;jump over
;
; Check for Rubout (remove & echo last char.)
;
CHKRUB:     CPI   7FH         ;do we have a rubout?
      JRNZ  CHKEOL            ;if not jump over
      MOV   A,B         ;if so move b into A
      ORA   A           ;set the flags
      JRZ   RB0         ;if b was zero go get another
      MOV   A,M         ;get the char at (ODATA)
      DCR   B           ;decrement the count in B
      DCX   H           ;point back to FCB
      JMP   RB10        ;jump to echo the char
;
; Check for Control-E (physical end-of-line)
;
```

```
CHKEOL:     CPI   CTRLE       ; is it end of line?
      JRNZ  CKPTOG            ;jump over if not
      PUSH  B           ;if so save BC
      PUSH  H           ;and FCB address
      CALL  CRLF        ;Turn up a new line
      XRA   A           ;zero A
      STA   CTEMP2            ;set it into CTEMP2
      JR    RB1         ;and go get more
;
; Check for Control-P
;
CKPTOG:     CPI   CTRLP       ;is it the print toggle
      JRNZ  CKBOL       ;if not jump past
      PUSH  H           ;if so save FCB address
      LXI   H,LSTCPY    ;get the pointer to print toggle byte
      MVI   A,01H       ;put a 1 into A
      SUB   M           ;subtract it from the toggle
      MOV   M,A         ;and put it back
      POP   H           ;recover FCB
      JR    RB0         ;and go get more
;
; Check for Control-X (bac+space to beg. current line)
;
CKBOL:      CPI   CTRLX       ;do we back up?
      JRNZ  CKREML            ;if not on to the next choice
      POP   H           ;if so restore the stack pointer
BLOOP:      LDA   CTEMP2            ;get the byte at CTEMP2
      LXI   H,COLUMN    ;and point to the column counter
      CMP   M           ;are they the same?
      JNC   REDBUF            ;if so go try again for input
      DCR   M           ;if not decrement the CTEMP2 count
      CALL  DELAST            ;delete the character there
      JR    BLOOP       ;and loop until we are done
;
; check for control-U
;
CKREML:     CPI   CTRLU       ;do we remove the line after newline?
      JRNZ  CKRETL            ;if not try again
      CALL  LBCRLF            ;if so print a "#" and CR
      POP   H           ;restore the stack
      JMP   REDBUF            ;and try for input again
;
; Check for Control-R (retype current line after new line)
;
CKRETL:     CPI   CTRLR       ;want to retype?
      JRNZ  ECHOCC            ;if not onward for next
RB65: PUSH  B           ;if so save the count in B
      CALL  LBCRLF            ;print a "#" and CRLF
      POP   B           ;recover ODATA
      POP   H           ;and FCB
      PUSH  H           ;saving FCB again
      PUSH  B           ;and ODATA
RB7:  MOV   A,B         ;move the count into
      ORA   A           ;see if it is zero
```

```
      JRZ   FIXCOL            ;if so jump over
      INX   H           ;if not point to ODATA
      MOV   C,M         ;pull the byte there into C
      DCR   B           ;decrement the count
      PUSH  B           ;save it on the stack
      PUSH  H           ;and the ODATA pointer
      CALL  CTLOUT            ;print it out expanding control chars
      POP   H           ;recover the ODATA pointer
      POP   B           ;and the count in B
      JR    RB7         ;loop until the line is out
;
;fix up the column counters
;
FIXCOL:     PUSH  H           ;save the pointer
      LDA   CTEMP1            ;get CTEMP1
      ORA   A           ;set the flags
      JZ    RB1         ;if zero go get the next character
      LXI   H,COLUMN    ;point to the column counter
      SUB   M           ;subtract it from the value of CTEMP1
      STA   CTEMP1            ;and save this back in CTEMP1
FXLOOP:     CALL  DELAST            ;delete the last character
      LXI   H,CTEMP1    ;point to CTEMP1
      DCR   M           ;decrement it by one
      JRNZ  FXLOOP            ;loop to delete all of them
      JMP   RB1         ;go get the next character
;
;echo the control character
;
ECHOCC:     INX   H           ;must be some other control character
      MOV   M,A         ;put the character into ODATA+1
      INR   B           ;bump the count by one
RB10: PUSH  B           ;save it on the stack
      PUSH  H           ;and the pointer as well
      MOV   C,A         ;put the character into C
      CALL  CTLOUT            ;print it out with grafic control chars
      POP   H           ;recover the pointer
      POP   B           ;and the count
      MOV   A,M         ;put the byte at (HL) into A
      CPI   CTRLC       ;is it an abort?
      MOV   A,B         ;put the count into A
      JRNZ  RB11        ;if no abort jump over
      CPI   01H         ;is the count 1?
      JZ    BOOTV       ;if so boot
RB11: CMP   C           ;if not does it equal C
      JC    RB0         ;if less go get another char
RBEXIT:     POP   H           ;recover the pointer
      MOV   M,B         ;put the count in b there
      MVI   C,CR        ;get a CR
      JMP   CONOUT            ;and print it out
;
;*****************************************************************
*******
;BDOS function 1: Read console - return with byte in A
;
```

```
REDCON:     CALL  CIECHO            ;get a char echo if printable
      JR    GOBAK       ;and go back with it
;
;*****************************************************************
*******
; BDOS function 3: Read reader - return with byte in A
;
REDRDR:     CALL  READF       ;get byte from reader
      JR    GOBAK       ;and return with it
;
;*****************************************************************
*******
; BDOS function 6: Direct I/O
;     on entry, C=FF for input, C=char for output
;     (Book says E reg vice C)
;      appears can enter with FE or FF --- ????
;
;     return with char or status in A
;
DIRTIO:     MOV   A,C         ;Get request
      INR   A           ;Test for FF=input request
      JRZ   INREQ       ;Skip down if input request
      INR   A           ;if FF adding one will set zero flag
      JZ    CONSF       ;if it was go get console status
      JMP   CONOF       ;otherwise go send it out
;
INREQ:      CALL  CONSF       ;get console status
      ORA   A           ;set flags
      JZ    REXIT       ;return if none - restore first
      CALL  CONIF       ;if someone is there go get it
      JR    GOBAK       ;and return with it
;
;*****************************************************************
*******
; BDOS function 7: get IO byte into A
;
GETIOB:     LDA   IOBYTE            ;get the iobyte
      JR    GOBAK       ;and go back with it
;
;*****************************************************************
*******
; BDOS function 8: set IO byte from C into place
;
PUTIOB:     LXI   H,IOBYTE    ;point to the iobyte
      MOV   M,C         ;put the new value in from C
      RET               ;and return
;
;*****************************************************************
*******
; BDOS function 9: Print console buffer until '$'
;     entry string address in DE
;
PRNBUF:     XCHG              ;swap DE and HL - HL points to buffer
      MOV   C,L         ;and get a copy
```

```
      MOV   B,H         ;  of HL into BC
      JMP   PRINT       ;go to the print routine
;
;*****************************************************************
*******
; BDOS function 11: check console status - return in A
;
GCSTAT:     CALL  CONBRK            ;Check for abort
GOBAK:      STA   ODATA       ;save result in return info byte
JR0:  RET               ;and go back to caller
;
JR1:  MVI   A,01H       ;get a 1 for return
      JR    GOBAK       ;set it in to rinfo and return
;
;*****************************************************************
*******
; console input buffer data storage area
;
CTEMP1:     DB    0           ;temp used for console input buffer
CTEMP2:     DB    0           ;temp count holder for buffer
COLUMN:     DB    0           ;column pointer for buffer
LSTCPY:     DB    0           ;list copy toggle byte
CHRRDY:     DB    0           ;char waiting flag
OLDSP:      DB    0,0         ;Caller's Stack ptr
      DB    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
      DB    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
      DB    0,0,0,0,0,0,0,0   ;48 bytes (room for stack)
USRCOD:     DB    0           ;user and disk code lives here
CURDSK:     DB    0           ;current disk holder
FCB:  DW    0           ;Storage for caller's parms
ODATA:      DW    0           ;return information storage
;
;*****************************************************************
*******
; Goes to Select error
;
PTSERR:     LXI   H,SELERR    ;HL=Select error sub addr
;
;*****************************************************************
*******
;routine branches to (HL)
;
HLGO: MOV   E,M         ;get (HL) into E
      INX   H           ;bump pointer
      MOV   D,M         ;and pull in (HL+1) into D
      XCHG              ;swap DE and HL
      PCHL              ;swap PC and HL and off we go
;
;*****************************************************************
*******
; routine to move C bytes DE to HL
;
MOVE: INR   C           ;add 1 to move count to start loop
MOVLOP:     DCR   C           ;decrement the loop count
```

```
      RZ                ;return if we are done
      LDAX  D           ;otherwise get the byte at (DE)
      MOV   M,A         ;and put it at (HL)
      INX   D           ;bump the source pointer
      INX   H           ;and the destination pointer
      JR    MOVLOP            ;and loop til you are done
;
;*****************************************************************
*******
;routine to set up disk identification data for access
;
DISKID:

      LDA   CURDSK            ;get the current logged disk
      MOV   C,A         ;and put it into C
      CALL  SELF        ;Select disk - HL comes back with DISP parm
      MOV   A,H         ;move H into A for the zero test
      ORA   L           ;or in L - zero says select error
      RZ                ;so return to LOGDSK with error

      PUSH  H           ;SAVE GOOD HL

      LXI   D,9         ;INDEX TO DIR BUFFER HI IN DPH
      DAD   D           ;HL HAS IT
      MOV   A,M         ;GET HI BYTE
      ORA   A           ;SEE IF PROCESSED
      JRZ   OLDLOG            ;IF SO SKIP NEXT
      PUSH  H
      DCX   H           ;POINT BACK TO LO BYTE

;     PULL DIRECTORY BUFFER ADDRESS OUT OF DPH

      MOV   E,M         ;LOW TO E
      INX   H           ;POINT HI
      MOV   H,M         ;HI TO H
      MOV   L,E         ;LOW TO L
      SHLD  DTEMP       ;SAVE DIR BUFF ADDRESS
      SHLD  000BH       ;AND HERE
      POP   H           ;POINT HI
      MVI   M,0         ;SET IT AS A FLAG

OLDLOG:
      LHLD  DTEMP       ;TEST FOR WARM BOOT
      MOV   A,L
      ORA   H
      JRNZ  OKLOG
      LHLD  000BH
      SHLD  DTEMP
OKLOG:
      POP   H
      MOV   E,M         ;otherwise move
      INX   H           ;    translation address
      MOV   D,M         ;       into DE
      INX   H           ;bump once more
```

```
      SHLD  SCRT0       ;and save this address at scratch 0
      INX   H           ;move the pointer
      INX   H           ;   two doors down
      SHLD  SCRT1       ;and save this address at scratch 1
      INX   H           ;move over
      INX   H           ;  two more
      SHLD  SCRT2       ;saving the result at scratch 2
      INX   H           ;two more
      INX   H           ;  for good measure

;++++++++ <BDOS CHANGE > ++++++++++++++++++++++++++++++++++++++
; SKIP RE-SAVING THE DIR BUFFER SINCE THE LOWER HALF OF
; IT IS NOW USED TO HOLD THE HI SECTOR COUNT
;

      XCHG              ;swap DE and HL - DE has DIRBUF pointer
      SHLD  TRANS       ;save the original disp parm header
      LXI   H,DTEMP+2   ;point PAST the temp DMA area
      MVI   C,6         ;Bytes to move
      INX   D
      INX   D

      CALL  MOVE        ;Move C bytes DIRBUF to DTEMP (dont know why)
      LHLD  DPB         ;get the disk parm block address
      XCHG              ;and put it into DE
      LXI   H,DPBLK           ;now set HL to point to storage area
      MVI   C,15        ;Bytes to move
      CALL  MOVE        ;Move disk data into storage area
      LHLD  DSIZE       ;get the size of this disk
      MOV   A,H         ;high byte zero says 1024 byte blocks
      LXI   H,BSIZE           ;point to block size byte
      MVI   M,0FFH            ;put in an FF
      ORA   A           ;set flags
      JRZ   JRFF        ;Return FF if blocks are 1024 bytes
      MVI   M,0         ;otherwise set in a ZERO
JRFF: MVI   A,0FFH            ;and get an FF into A
      ORA   A           ;set flags
      RET               ;and return
;
;*****************************************************************
*******
;routine to home the selected disk and reset data words
;
HOMDSK:     CALL  HOMF        ;home the selected drive
      XRA   A           ;get a zero
      LHLD  SCRT1       ;point to the current track storage
      MOV   M,A         ;put in the zero
      INX   H           ;point to the low half
      MOV   M,A         ;and zero this as well
      LHLD  SCRT2       ;point to starting sector count for this track
      MOV   M,A         ;zero it
      INX   H           ;bump pointer
      MOV   M,A         ;both halves of this word
```

```
;++++++++ <BDOS CHANGE > + AND NEW HIGH RECORD LOC

      INX   H
      MOV   M,A

      RET               ;and return
;
;*****************************************************************
*******
;routine reads indicated sector and checks status
;
RDSEC:
      CALL  GETRECS
      CALL  DRDF        ;Read sector
      JR    PERCHK            ;jump to check error and return
;
;*****************************************************************
*******
;routine to write indicated sector and check result
;
WRSEC:
      CALL  GETRECS
      CALL  DWRF        ;Write sector
;
PERCHK:     ORA   A           ;set flags
      RZ                ;return if all is well
      LXI   H,PERERR    ;HL=Permanent err sub addr
      JMP   HLGO        ;Go to routine


;++++++++ <BDOS CHANGE > ++++++++++++++++++++++++++++++++
;                    ADD FOR DE & B
;              B & DE HAVE RESULT OF HL+DE
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++

ADD$WORD:
      MOV   A,E
      ADD   L
      MOV   E,A
      MOV   A,D
      ADC   H
      MOV   D,A
      RNC
      INR   B
      RET

;++++++++ <BDOS CHANGE > ++++++++++++++++++++++++++++++++
;                  SUBTRACT FOR DE & B
;              B & DE HAVE RESULT OF DE-HL
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++

SUB$WORD:
      MOV   A,E
      SUB   L
```

```
      MOV   E,A
      MOV   A,D
      SBB   H
      MOV   D,A
      RNC
      DCR   B
      RET

GETRECS:
      LHLD  GROUP
      XCHG
      LXI   H,GROUP+2
      MOV   B,M
      RET



;******************************************************
; GET NEXT DIRECTORY SECTOR


GETDS:
      LHLD  COUNT ;DIR COUNT
      MVI   C,2   ;SHIFT 2
      CALL  SPINHL      ;HL GOES RIGHT 2 BITS
      SHLD  GROUP ;GROUP NOW HAS SECTOR FOR DIR ENTRY
      SHLD  LDIRPNT     ;SO DOES LDIRPNT
      XRA   A
      STA   GROUP+2     ;SET HIGH COUNTERS TO 0
      STA   LDIRPNT+2


;++++++++ <BDOS CHANGE > ++++++++++++++++++++++++++++++++
;::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;            MAKE ANY DISK RECORD TO ACCESS
;::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++

GETD1:
      LHLD  SCRT1             ;PULL CURRENT TRACK ADDRESS
      MOV   C,M               ;GET LOW SCR1 TO C
      INX   H                 ;POINT NEXT
      MOV   B,M               ;GET HI SCR1 TO B
      PUSH  B                 ;SAVE TRACK
      LHLD  SCRT2             ;PULL CURRENT SECTOR ADDRESS
      MOV   E,M               ;GET LOW TO E
      INX   H                 ;NEXT
      MOV   D,M               ;GET HI TO D
      INX   H                 ;NEXT
      MOV   B,M               ;GET SCR3 TO B
      LHLD  GROUP             ;HL HAS CURRENT RECORD LO & HI
      LDA   GROUP+2                 ;ACC HAS CUR. REC. HI
      MOV   C,A               ;C GETS HI RECORD
```

```
;NOW SET TRACK AND STARTING SECTOR NUMBER FOR DISK ACCESS

GET$DS2:
      MOV   A,L               ;ACC GETS LO RECORD
      SUB   E                 ;SUBTRACT FROM SCR2 LO
      MOV   A,H               ;MID REC TO ACC
      SBB   D                 ;SUB W/ BORROW D
      MOV   A,C               ;HI REC INTO ACC
      SBB   B                 ;SUB W/ BORROW B
      PUSH  H                 ;SAVE HL
      JNC   GET$DS3                 ;IF NO BORROW BRANCH
      LHLD  DPBLK             ;ELSE GET SECTORS/TRACK
      CALL  SUB$WORD          ;AND GO REMOVE THEM
      POP   H                 ;RECOVER HL
      XTHL                    ;STACK<>HL (GET SAVED TRACK TO HL)
      DCX   H                 ;COUNT DOWN
      XTHL                    ;STACK<>HL (UPDATED TRACK TO STACK)
      JR    GET$DS2                 ;DO NEXT

;     B,DE THE RECORD, PUSHED HL THE TRACK

GET$DS3:
      LHLD  DPBLK             ;GET SECTORS PER TRACK
      CALL  ADD$WORD          ;ADD TO TRACK COUNT
      POP   H                 ;GET TRACK
      MOV   A,L               ;LOW TO ACC
      SUB   E                 ;UPDATE RECORD
      MOV   A,H               ;GET MID
      SBB   D                 ;UPDATE RECORD
      MOV   A,C               ;GET HI
      SBB   B                 ;UPDATE RECORD
      JRC   GET$DS4                 ;BRANCH IF DONE HERE
      XTHL                    ;GET STACKED TRACK
      INX   H                 ;UPDATE IT
      XTHL                    ;RE-STACK
      PUSH  H                 ;STACK HL FOR CORRECT ENTRY
      JR    GET$DS3                 ;BACK TO START


;     POINT DISK TO RIGHT SPOT FOR ACCESS

GET$DS4:
      XTHL                    ;GET THAT STACKED TRACK
      PUSH  H                 ;SAVE IT
      LHLD  DPBLK             ;GET SECTORS/TRACK
      CALL  SUB$WORD          ;REMOVE
      POP   H                 ;RECOVER TRACK
      PUSH  D                 ;SAVE IT
      PUSH  B                 ;ALL
      PUSH  H                 ;ON STACK
      XCHG                    ;DE<>HL
      LHLD  TOFS              ;GET TRACK OFFSET
      DAD   D                 ;ADD IN DESIRED TRACK NUMBER
      MOV   B,H               ;PUT IN BC
```

```
      MOV   C,L               ;FOR NEXT CALL
      CALL  TRKF              ;HAVE BIOS SET IT
      POP   D                 ;RECOVER ENTRY TRACK
      LHLD  SCRT1             ;GET POINTER TO CURRENT TRACK
      MOV   M,E               ;AND PUT NEW TRACK
      INX   H                 ;BACK INTO
      MOV   M,D               ;THE SCRATCH
      POP   B                 ;RECOVER CURRENT SECTOR HI
      POP   D                 ;AND MID-LOW
      LHLD  SCRT2             ;GET POINTER TO SCRATCH
      MOV   M,E               ;AND SET IN THE
      INX   H                 ;NEW
      MOV   M,D               ;SECTOR
      INX   H                 ;FOR
      MOV   M,B               ;DISK ACCESS
      POP   B                 ;RECOVER NEW TRACK SECTOR START NUMBER
      MOV   A,C               ;SUBTRACT
      SUB   E                 ; TRACK SECTOR START NUMBER
      MOV   C,A               ;   FROM THE DESIRED NUMBER
      MOV   A,B               ;     AND LEAVE
      SBB   D                 ;        RESULT
      MOV   B,A               ;          IN BC
      LHLD  TRANS             ;GET SECTOR XLATE TABLE
      XCHG                    ;INTO DE
      CALL  SECTRN                  ;HAVE BIOS TRANSLATE IT
      MOV   C,L               ;GET IT INTO
      MOV   B,H               ;BC
      JMP   SECF              ;HAVE BIOS SET SECTOR



;
;*****************************************************************
*******
;routine gets extent sector count from storage and figures block number
;
COMBLK:     LXI   H,BLSHFT    ;point to block shift factor
      MOV   C,M         ;pull it into C
      LDA   ESCNT2            ; pull in the extent low byte
COMBL1:     ORA   A           ;set flags - clear carry
      RAR               ;rotate right through carry
      DCR   C           ;decrement count
      JRNZ  COMBL1            ;loop until done
      MOV   B,A         ;then save the byte into B
      MVI   A,08H       ;get bit 3 set
      SUB   M           ;subtract block shift from this value
      MOV   C,A         ;and save it into C
      LDA   ESCNT1            ;get the extent high byte
COMBL2:     DCR   C           ;decrement count again
      JRZ   COMBL3            ;if done jump
      ORA   A           ;otherwise set flags - clear carry
      RAL               ;rotate left through carry
      JR    COMBL2            ;loop until done
;
```

```
COMBL3:     ADD   B           ;add in the offset
      RET               ;and return
;
;*****************************************************************
*******
;routine points into the FCB block and returns with group pointer.
;If BSIZE=0 blocks are>1024 and 16 bits come back
;
GETGRP:     LHLD  FCB         ;get the FCB address
      LXI   D,16        ;and a 16
      DAD   D           ;add in the offset
      DAD   B           ;add in BC as well
      LDA   BSIZE       ;get the block size byte
      ORA   A           ;set flags
      JRZ   HLPBC       ;if zero add HL and BC
      MOV   L,M         ;pull in this byte
      MVI   H,0         ;set upper byte 0
      RET               ;return with HL set
;
; routine sets HL = (HL+BC)
;
HLPBC:      DAD   B           ;add HL and BC
      MOV   E,M         ;move this address byte into E
      INX   H           ;point to next
      MOV   D,M         ;pull this one into D
      XCHG              ;swap DE and HL
      RET               ;and go back
;
;*****************************************************************
*******
;routine takes sector count for directory area and gets the group
pointer out
;
SECGRP:     CALL  COMBLK            ;get the block number
      MOV   C,A         ;and put it into C
      MVI   B,0         ;zero out B
      CALL  GETGRP            ;now go get the right group
      SHLD  GROUP       ;save it at group storage
      RET               ;and go back
;
;*****************************************************************
*******
; routine checks group and sets flags if zero - no group 0 should be
there
;
TSTGRP:     LHLD  GROUP       ;get the group pointer out
      MOV   A,L         ;move low into A
      ORA   H           ;or with H to set flags
      RET               ;and go back
;
;*****************************************************************
*******
; routine appears to figure actual sector number from group pointer
;++++++++ <BDOS CHANGE > + 24 BIT SUPPORT ADDED
```

```
COMSEC:     LDA   BLSHFT
      MOV   C,A
      LHLD  GROUP
      XRA   A
      CALL  SPINBIT
      SHLD  GROUP
      STA   GROUP+2

      SHLD  DIRPNT
      LDA   BLKMSK
      MOV   C,A
      LDA   ESCNT2
      ANA   C

      LXI   H,GROUP
      ORA   M
      MOV   M,A
      RET



;
;*****************************************************************
*******
;routine to point to information + 12 - the extent byte of the FCB
;
GETEX:      LHLD  FCB         ;get the FCB pointer
      LXI   D,12        ;and get a 12
      DAD   D           ;add in the offset
      RET               ;return with HL set to extent address
;
;*****************************************************************
*******
;routine sets HL to current record (CR) and DE to record count (RC)
;
SETPT:      LHLD  FCB         ;get the FCB address
      LXI   D,15        ;get an offset of 15
      DAD   D           ;HL now points to RC address
      XCHG              ;swap new value into DE
      LXI   H,17        ;now get an offset of 17
      DAD   D           ;add this in so HL points to CR
      RET               ;return with HL and DE set
;
;*****************************************************************
*******
;routine appears to get the extent byte from FCB and set up extent
pointers
;
FIXEXT:     CALL  SETPT       ;set record pointers
      MOV   A,M         ;get the current record for sequential I/O
      STA   ESCNT2            ;and save it
      XCHG              ;swap DE and HL
      MOV   A,M         ;pull in the record count for this extent
```

```
      STA   RECCNT            ;and save it
      CALL  GETEX             ;point to extent byte in FCB
      LDA   EXTMSK            ;get the extent mask
      ANA   M           ;and the extent byte extent mask
      STA   ESCNT1            ;saving the result
      RET               ;and return
;
;*****************************************************************
*******
;routine appears to set up record counters
;
FIXREC:     CALL  SETPT       ;set the record pointers
      LDA   DCODE       ;get the byte
      CPI   02H         ;is it a 2?
      JRNZ  FIXRC1            ;if not jump past
      XRA   A           ;zero out A
FIXRC1:     MOV   C,A         ;move A into C
      LDA   ESCNT2            ;get the byte
      ADD   C           ;add it to C
      MOV   M,A         ;put the result away at CR in FCB
      XCHG              ;swap DE and HL
      LDA   RECCNT            ;get the byte ( record count)
      MOV   M,A         ;place it at RC in FCB
      RET               ;and return
;
;*****************************************************************
*******
; spin HL by C bits right
;
SPINHL:     INR   C           ;set up for the loop decrement of C
SPH1: DCR   C           ;decrement loop count
      RZ                ;return if all is well
      MOV   A,H         ;otherwise get H into A
      ORA   A           ;clear carry
      RAR               ;rotate right through carry
      MOV   H,A         ;put the new byte back
      MOV   A,L         ;move L into A
      RAR               ;rotate it as well
      MOV   L,A         ;and put it back
      JR    SPH1        ;loop until the count quits
;
;*****************************************************************
*******
;routine to compute a check sum across a buffer of 128 bytes
;
SUM128:     MVI   C,128       ;we will sum 128 bytes in the temp buffer
      LHLD  DTEMP       ;get the address of the buffer from DTEMP
      XRA   A           ;zero A
SUM1: ADD   M           ;add in the byte at (HL)
      INX   H           ;bump pointer to next
      DCR   C           ;decrement count
      JRNZ  SUM1        ;loop if we are not finished
      RET               ;otherwise go back with sum in A
;
```

```
;*****************************************************************
*******
;routine to slide HL left C bits , CY & ACC
;
SPINBIT:INR C           ;bump for the decrement
SPB1: DCR   C           ;decrement the loop count
      RZ                ;return when done
      DAD   H           ;slide HL left one bit
      ADC   A
      JR    SPB1        ;spin until count zeroes


;
;*****************************************************************
*******
;routine to slide HL left C bits
;
SPINBITH:

      INR   C           ;bump for the decrement
SPBH1:      DCR   C           ;decrement the loop count
      RZ                ;return when done
      DAD   H           ;slide HL left one bit
      ADC   A
      JR    SPBH1       ;spin until count zeroes



;
;*****************************************************************
*******
;set the disk bit in the vector - BC has the current vector
;
SETDBT:     PUSH  B           ;save the BC pair
      LDA   CURDSK            ;get the current disk
      MOV   C,A         ;put it into C
      LXI   H,1         ;get a 1 into HL
      CALL  SPINBITH          ;spin HL into right spot to set RO
      POP   B           ;recover BC
      MOV   A,C         ;put C into A
      ORA   L           ;or in the RO flag low
      MOV   L,A         ;put it back
      MOV   A,B         ;pull in the high byte
      ORA   H           ;or in the RO flag high
      MOV   H,A         ;put it back
      RET               ;and return with new RO vector in HL
;
;*****************************************************************
*******
;routine to check for drive READ-ONLY state
;
ISRO: LHLD  ROVEC       ;point to the RO vector
      LDA   CURDSK            ;get the current disk
      MOV   C,A         ;and put it into C
```

```
      CALL  SPINHL            ;spin the RO vector into position
      MOV   A,L         ;and pull L into A
      ANI   01H         ;mask off all but bit 0
      RET               ;and return with flags zero for no,1 for yes
;
;*****************************************************************
*******
; BDOS function 28: makes drive read only
;
MAKRO:      LXI   H,ROVEC           ;point to the RO vector
      MOV   C,M         ;pull in the low half
      INX   H           ;point to high byte
      MOV   B,M         ;pull this one into B - got it all
      CALL  SETDBT            ;go set this drive RO
      SHLD  ROVEC       ;and save the new RO vector back for later
      LHLD  DIRMAX            ;get the directory max value
      INX   H           ;point to RO + 1
      XCHG              ;swap this address into DE
      LHLD  SCRT0       ;point to scratch0
      MOV   M,E         ;put RO + 1
      INX   H           ;   into
      MOV   M,D         ;     scratch0
      RET               ;and go back
;
;*****************************************************************
*******
;routine to check for file read only
;
CHKFRO:     CALL  PNTDIR            ;set up HL to DMA + offset
CHKFR1:     LXI   D,9         ;get 9 - point to file ro bit in t1
      DAD   D           ;add it to the pointer
      MOV   A,M         ;pull in this byte
      RAL               ;rotate it left through carry
      RNC               ;if bit 7 was not set return
      LXI   H,RONERR    ;HL=Read-only error sub addr
      JMP   HLGO        ;Go to routine
;
;*****************************************************************
*******
;see if the selected drive is read only - say we cant write if so
;
ROCHK:      CALL  ISRO        ;is the current drive read only?
      RZ                ;return if not - we can write
      LXI   H,ROERR           ;otherwise point to error routine
      JMP   HLGO        ;Go to routine
;
;*****************************************************************
*******
;routine to point to directory entry stored in buffer
;
PNTDIR:     LHLD  DTEMP       ;get the temp buffer address out
      LDA   DIROFF            ;get the offset from storage
PNTD1:      ADD   L           ;add it to A
      MOV   L,A         ;  and put it into L
```

```
      RNC               ;    if no overflow go back
      INR   H           ;otherwise add in the overflow
      RET               ;and then return
;
;*****************************************************************
*******
;routine gets the S2 byte at (FCB + 14)
;
GETS2:      LHLD  FCB         ;get the input FCB address
      LXI   D,14        ;point 14 downstream
      DAD   D           ;set up this address
      MOV   A,M         ;pull this byte into A
      RET               ;and go back
;
;*****************************************************************
*******
;routine sets the S2 byte at (FCB + 14) to zero
;
ZS2:  CALL  GETS2       ;get the address of S2
      MVI   M,0         ;and set in a zero
      RET               ;and go back
;
;*****************************************************************
*******
;routine sets bit 8 of S2
;
SETS28:     CALL  GETS2       ;point to S2
      ORI   80H         ;or in the eigth bit
      MOV   M,A         ;and put this value back
      RET               ;and return
;
;*****************************************************************
*******
;routine subtracts counter from word at scratch0
;
SUBCS0:     LHLD  COUNT       ;get the counter
      XCHG              ;swap it into DE
      LHLD  SCRT0       ;then get scratch0 data word
      MOV   A,E         ;and subtract
      SUB   M           ;  data at scratch 0
      INX   H           ;    from
      MOV   A,D         ;      the
      SBB   M           ;        counter
      RET               ;and go back with result set in flags?
;
;*****************************************************************
*******
;routine checks count against scratch0 and decrements count if ok
;
CHKINC:     CALL  SUBCS0            ;compare counter and (scratch0)
      RC                ;if (scratch0)>counter return
      INX   D           ;otherwise bump DE
      MOV   M,D         ;and place it
      DCX   H           ;  into
```

```
      MOV   M,E         ;    (scratch0)
      RET               ;and return
;
;*****************************************************************
*******
;routine to compute HL = DE - HL
;
SUBHD:      MOV   A,E         ;pull E into A
      SUB   L           ;subtract from L
      MOV   L,A         ;and put it back
      MOV   A,D         ;then move in the D value
      SBB   H           ;and subtract with borrow
      MOV   H,A         ;put back into H
      RET               ;and return
;
;*****************************************************************
*******
;routine to check directory area with check sum to see if it changed
;
CHKSUM:     MVI   C,0FFH            ;get the default return code
CHKCNG:     LHLD  LDIRPNT           ;point to the directory area
      XCHG              ;put the address into DE
      LHLD  CKSIZE            ;get the check size
      CALL  SUBHD       ;compute HL = DE - HL
      RNC               ;return if they were the same
      PUSH  B           ;otherwise save the BC regs
      CALL  SUM128            ;and compute the new check sum
      LHLD  CHANGE            ;point to the change storage area
      XCHG              ;swap DE and HL
      LHLD  LDIRPNT           ;point to the directory area
      DAD   D           ;add in the change word
      POP   B           ;recover BC
      INR   C           ;bump count by one
      JRZ   CHK1        ;if zero save A at (HL) and return
      CMP   M           ;if not check A and (HL)
      RZ                ;return if they are the same
      CALL  SUBCS0            ;otherwise check counter and (scratch0)
      RNC               ;return if they are ok
      CALL  MAKRO       ;make the disk read only if not
      RET               ;and return
;
CHK1: MOV   M,A         ;put A into (HL)
      RET               ;and return
;
;*****************************************************************
*******
;routine to write directory sector out
;
WRTDS:      CALL  CHKSUM            ;check the directory check sum
      CALL  SETEMP            ;if ok set DMA address up
      MVI   C,01H       ;get the code into C
      CALL  WRSEC       ;write the sector
      JR    SETDMA            ;reset the DMA address and return
;
```

```
;*****************************************************************
*******
;routine to read the next sector from the disk
;
GETNXT:     CALL  SETEMP            ;set the DMA address to (DTEMP)
      CALL  RDSEC       ;and read the next sector
;
;*****************************************************************
*******
;tell BIOS the correct DMA address
;
SETDMA:     LXI   H,DMAADR    ;point to the DMA address word
      JR    SET2        ;and tell BIOS about it
;
;*****************************************************************
*******
; set the dma address to DTEMP
;
SETEMP:     LXI   H,DTEMP           ;get the DMA address from DTEMP
;
SET2: MOV   C,M         ;move the low byte into C
      INX   H           ;point to the next
      MOV   B,M         ;and pull high into B
      JMP   DMAF        ;Set DMA buffer addr & return
;
;*****************************************************************
*******
;routine to move the data stored at (DTEMP) to (DMAADR)
;
MOVSEC:     LHLD  DTEMP       ;point to the  DMA buffer
      XCHG              ;put it into DE
      LHLD  DMAADR            ;get the DMA address for the load
      MVI   C,128       ;Bytes to move
      JMP   MOVE        ;Move C bytes DE to HL
;
;*****************************************************************
*******
;routine checks count - returns 1 if it is zero
;and the first byte at count if it is nonzero
;
CHKCNT:     LXI   H,COUNT           ;get count
      MOV   A,M         ;check to
      INX   H           ;   see
      CMP   M           ;     if it is zero
      RNZ               ; return if it is not
      INR   A           ;bump up to 1 if it is
      RET               ;and then return
;
;*****************************************************************
*******
;routine sets counter to -1 for starting loop
;
SETCNT:     LXI   H,-1        ;start off with a 16 bit -1
      SHLD  COUNT       ;and save it at count
```

```
      RET               ;return to sender
;
;*****************************************************************
*******
;routine moves to next directory entry - COUNT says the last one looked
at
;
NXTDIR:     LHLD  DIRMAX            ;get the number of directory entries
      XCHG              ;save it in DE
      LHLD  COUNT       ;now get the directory counter
      INX   H           ;add one to it
      SHLD  COUNT       ;and save it back for later
      CALL  SUBHD       ;looking too far?
      JNC   GOTDIR            ;if no carry still entries left
      JR    SETCNT            ;otherwise return with a 16 bit -1
;
;*****************************************************************
*******
;routine to get the next directory sector off of the disk
;
GOTDIR:     LDA   COUNT       ;get the directory count low byte
      ANI   03H         ;strip off all but low two (4 entries per sec)
      MVI   B,5         ;set to spin low 3 over 5
SPINAB:     ADD   A           ;move over 1 bit
      DCR   B           ;decrement count
      JRNZ  SPINAB            ;loop until done
      STA   DIROFF            ;save this for later
      ORA   A           ;set flags
      RNZ               ;return if not zero
      PUSH  B           ;otherwise save BC
      CALL  GETDS       ;looks like we set for getting the next dir
      CALL  GETNXT            ;and read the next sector
      POP   B           ;recover the BC pair
      JMP   CHKCNG            ;see if the disk changed and return
;
;*****************************************************************
*******
;routine gets an allocation bit set into position
; and sets pointer to allocation byte for marking blocks
; BC has the group pointer - Exit with BC -> allocation byte
;                             D  -> bit for this group
;                             A  -> allocation byte spun into place
;

GETBIT:
      MOV   A,C
      ANI   7
      INR   A
      MOV   E,A
      MOV   D,A
      MOV   H,B
      MOV   L,C
      MVI   C,3
      CALL  SPINHL
```

```
      MOV   B,H
      MOV   C,L
      LHLD  ALLOCA
      DAD   B
      MOV   A,M
L0811:      RLC
      DCR   E
      JRNZ  L0811
      RET


;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;
;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::

SETBIT:
      PUSH  D
      CALL  GETBIT
      ANI   0FEH
      POP   B
      ORA   C
PUTBAK:
      RRC
      DCR   D
      JRNZ  PUTBAK
      MOV   M,A
      RET


;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;              UPDATE ALLOCATION MAP
;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::

FIXALL:
      CALL  PNTDIR
L0829:
      LXI   D,10H
      DAD   D
      PUSH  B
      MVI   C,11H
L0830:      POP   D
      DCR   C
      RZ
      PUSH  D
      LDA   BSIZE
      ORA   A
      JRZ   L0843
      PUSH  B
      PUSH  H
      MOV   C,M
      MVI   B,0
      JR    L0849

L0843:
```

```
      DCR   C
      PUSH  B
      MOV   C,M
      INX   H
      MOV   B,M
      PUSH  H

L0849:      MOV   A,C
L084A:      ORA   B
      JRZ   L0858
      LHLD  DSIZE
      MOV   A,L
      SUB   C
      MOV   A,H
      SBB   B
      CNC   SETBIT

L0858:      POP   H
L0859:      INX   H
      POP   B
      JR    L0830

;
;*****************************************************************
*******
;routine to get the allocation vector
;
GETAL:      LHLD  DSIZE       ;get the max number of blocks
      MVI   C,03H       ;set to
      CALL  SPINHL            ;  divide by eight
      INX   H           ;add 1 for the map size
      MOV   B,H         ;and put map size
      MOV   C,L         ;  into BC - number of bytes in allocation map
      LHLD  ALLOCA            ;point to allocation storage area
CLRALL:     MVI   M,0         ;put in a zero
      INX   H           ;bump the pointer
      DCX   B           ;one less group to zero out
      MOV   A,B         ;put the new count into A
      ORA   C           ;check for done
      JRNZ  CLRALL            ;if not loop until so
      LHLD  AL01        ;if done get directory allocation bits
      XCHG              ;swap DE and AL01
      LHLD  ALLOCA            ;point to allocation stogare
      MOV   M,E         ;put directory allocation bits
      INX   H           ;  into that
      MOV   M,D         ;     storage area cause they are busy
      CALL  HOMDSK            ;home the selected disk
      LHLD  SCRT0       ;get the scratch 0 address
      MVI   M,03H       ;save a 3 at (scratch0)
      INX   H           ;point to (scratch0+1)
      MVI   M,0         ;and zero this out - it has 16 bit 3 now
      CALL  SETCNT            ;set a counter to -1
GETA1:      MVI   C,0FFH            ;set the exit code in
      CALL  NXTDIR            ;set the next directory pointers
```

```
      CALL  CHKCNT            ;was there one?
      RZ                ;return if not
      CALL  PNTDIR            ;point to the right spot
      MVI   A,DELDAT    ;get the file deleted mark
      CMP   M           ;is this what we see?
      JRZ   GETA1       ;keep looking for a nonempty block
      LDA   USRCOD            ;get the user byte
      CMP   M           ;test this against the first byte
      JRNZ  GETA2       ;if we don't match mark this block
      INX   H           ;bump pointer to file name entry
      MOV   A,M         ;pull the byte in
      SUI   '$'         ;is this a submit file name $$$.SUB
      JRNZ  GETA2       ;if not take this block also
      DCR   A           ;otherwise set A to FF
      STA   ODATA       ;and save it at ODATA for CCP to use
GETA2:      MVI   C,01H       ;set the bit into C
      CALL  FIXALL            ;fix this allocation bit
      CALL  CHKINC            ;test for too far - bump count if not
      JR    GETA1       ;spin until done with the scan


;
;*****************************************************************
*******
;routine loads return code and goes back to sender
;
RETCOD:     LDA   SCODE1            ;
      JMP   GOBAK       ;
;
;*****************************************************************
*******
;routine appears to check extent byte
;
CHKEXT:     PUSH  B           ;get the entry BC safe
      PUSH  PSW         ;and the accumulator
      LDA   EXTMSK            ;get the extent mask
      CMA               ;flip it over
      MOV   B,A         ;and save it into B
      MOV   A,C         ;move C into A
      ANA   B           ;and it with the flipped extent mask
      MOV   C,A         ;save this into C
      POP   PSW         ;get the entry A back
      ANA   B           ;and with flip extent mask
      SUB   C           ;subtract C
      ANI   1FH         ;mask off high bit
      POP   B           ;recover the entry BC
      RET               ;and go back confused
;
;*****************************************************************
*******
;search for a specified file - set flags to say how it went
;used by all file routines to set up file information
;
SEARCH:     MVI   A,0FFH            ;set return code up
```

```
      STA   SCODE1            ;and save it for later
      LXI   H,SCODE           ;point to search code area
      MOV   M,C         ;save entry code away
      LHLD  FCB         ;get the entry FCB address
      SHLD  SEARA       ;save a copy
      CALL  SETCNT            ;set the search counter up
      CALL  HOMDSK            ;home the selected disk
SERCHN:     MVI   C,0         ;
      CALL  NXTDIR            ;locate next directory entry
      CALL  CHKCNT            ;check to see if one was found
      JZ    NOTFND            ;jump over if not
      LHLD  SEARA       ;get the FCB address out
      XCHG              ;into DE
      LDAX  D           ;get the byte at (DE)
      CPI   DELDAT            ;is it a deleted file?
      JRZ   SERCH1            ;if so jump over
      PUSH  D           ;save this address for later
      CALL  SUBCS0            ;check count and increment
      POP   D           ;recover the address
      JNC   NOTFND            ;if no carry we didn't find the file
SERCH1:     CALL  PNTDIR            ;point to the indicated directory
entry
      LDA   SCODE       ;get the search code out
      MOV   C,A         ;stuff it into C
      MVI   B,0         ;zero out the high byte
SERCH2:     MOV   A,C         ;move it to A
      ORA   A           ;set the flags - check for zero
      JRZ   SEREXT            ;if so jump over
      LDAX  D           ;otherwise get the FCB file name char
      CPI   '?'         ;is it a wild card name?
      JRZ   MATCH       ;if so jump over
      MOV   A,B         ;check B
      CPI   0DH         ;is it past extent byte?
      JRZ   MATCH       ;if so this is a good file to go back with
      CPI   0CH         ;how about at the extent byte?
      LDAX  D           ;get the byte again
      JRZ   SERCH3            ;if so jump over
      SUB   M           ;otherwise subtract the two bytes
      ANI   7FH         ;strip off the special bits
      JRNZ  SERCHN            ;if not zero this doesnt match so try
again
      JR    MATCH       ;if it matches press on
;
SERCH3:     PUSH  B           ;save BC from harm
      MOV   C,M         ;pull the byte into C
      CALL  CHKEXT            ;check the extent byte
      POP   B           ;recover BC
      JRNZ  SERCHN            ;if not zero search again
MATCH:      INX   D           ;otherwise bump FCB pointer
      INX   H           ;and pointer into memory
      INR   B           ;increment the FCB count
      DCR   C           ;decrement the C counter
      JR    SERCH2            ;and keep going with the search
;
```

```
SEREXT:     LDA   COUNT       ;get the count
      ANI   03H         ;strip off the high 5 bits
      STA   ODATA       ;save this at ODATA for return
      LXI   H,SCODE1    ;get the return code address
      MOV   A,M         ;pull it into A
      RAL               ;slide it left one bit through carry
      RNC               ;if bit 7 not set return
      XRA   A           ;otherwise zero out A
      MOV   M,A         ;put it back at SCODE1
      RET               ;then go back
;
NOTFND:     CALL  SETCNT            ;set the counter for error code
      MVI   A,0FFH            ;and set the return byte into A
      JMP   GOBAK       ;then return to sender
;
;*****************************************************************
*******
;routine to delete specified file
;
DELETF:     CALL  ROCHK       ;see if the selected drive is read only
      MVI   C,0CH       ;look for name and extent
      CALL  SEARCH            ;go find the file
DELOOP:     CALL  CHKCNT            ;was it there
      RZ                ;return if not - all extents are marked
      CALL  CHKFRO            ;see if file was read only
      CALL  PNTDIR            ;point to directory entry
      MVI   M,DELDAT    ;put in the file deleted marker
      MVI   C,0         ;set C for the fix bit routine
      CALL  FIXALL            ;fix the allocation vector
      CALL  WRTDS       ;write out the directory sector
      CALL  SERCHN            ;search for the next extent
      JR    DELOOP            ;and loop until done
;
;*****************************************************************
*******
;routine to check for block availability
;
BLKAVL:     MOV   D,B         ;move BC
      MOV   E,C         ;  into DE
BLKA1:      MOV   A,C         ;save a copy of C into A
      ORA   B           ;  check if BC is zero
      JRZ   BLKA2       ;if so jump over and look again
      DCX   B           ;decrement the counter
      PUSH  D           ;save on the stack
      PUSH  B           ;save BC as well
      CALL  GETBIT            ;check the allocation bit
      RAR               ;slide it right through carry
      JNC   TAKBLK            ;if no carry it is free
      POP   B           ;restore
      POP   D           ;  stack
BLKA2:      LHLD  DSIZE       ;get the disk size block count
      MOV   A,E         ;subtract
      SUB   L           ;   HL
      MOV   A,D         ;    from
```

```
      SBB   H           ;        DE
      JNC   BLKA3       ;if no carry all is well - jump over
      INX   D           ;otherwise bump DE
      PUSH  B           ;save on stack
      PUSH  D           ;    for next look
      MOV   B,D         ;put DE
      MOV   C,E         ;   into BC
      CALL  GETBIT            ;get this allocation bit out
      RAR               ;slide right to test
      JNC   TAKBLK            ;if no carry take the block
      POP   D           ;recover the
      POP   B           ;  stack
      JR    BLKA1       ;and keep looking
;
TAKBLK:     RAL               ;restore the allocation bit
      INR   A           ;add one to set bit
      CALL  PUTBAK            ;write it back into the vector
      POP   H           ;recover
      POP   D           ;  the stack
      RET               ;and return to the caller
;
BLKA3:      MOV   A,C         ;is BC
      ORA   B           ;  zero?
      JRNZ  BLKA1       ;if not jump back and continue
      LXI   H,0         ;otherwise get a 16 bit zero
      RET               ;and return with it
;
;*****************************************************************
*******
;routine to move the directory entry
;
MOVDIR:     MVI   C,0         ;set C to zero for the move
      MVI   E,32        ;get the directory size
;
;*****************************************************************
*******
;routine to set the directory onto the disk
;
SETDIR:     PUSH  D           ;save the count on the stack
      MVI   B,0         ;zero the high byte
      LHLD  FCB         ;get the FCB address
      DAD   B           ;add in the offset for the move
      XCHG              ;save it in DE
      CALL  PNTDIR            ;point to the directory area
      POP   B           ;recover the count
      CALL  MOVE        ;Move C bytes DE to HL
SETD1:      CALL  GETDS       ;go pull in the rest of the directory
      JMP   WRTDS       ;and write the current sector out
;
;*****************************************************************
*******
;routine to rename the selected file
;
RENAMF:     CALL  ROCHK       ;see if the selected disk is read only
```

```
      MVI   C,0CH       ;no- check name and extent
      CALL  SEARCH            ;by calling the search routine
      LHLD  FCB         ;get the information FCB address
      MOV   A,M         ;and pull the drivecode
      LXI   D,16        ;we will move down 16 bytes
      DAD   D           ;point to the new file name in FCB
      MOV   M,A         ;put in the new drive code
RENF1:      CALL  CHKCNT            ;check the loop counter
      RZ                ;return if all extents are renamed
      CALL  CHKFRO            ;see if the filw is read only
      MVI   C,10H       ;set to move 16 bytes of the file name
      MVI   E,0CH       ;get the search code ready
      CALL  SETDIR            ;set in the file name
      CALL  SERCHN            ;search for the next extent
      JR    RENF1       ;loop until no more extents
;
;*****************************************************************
*******
;routine sets up pointers to indicated file
;
SETFN:      MVI   C,0CH       ;look at name and extent
      CALL  SEARCH            ;search for the file
SETF1:      CALL  CHKCNT            ;see if there was an entry left
      RZ                ;return when all extents have been found
      MVI   C,0         ;
      MVI   E,0CH       ;
      CALL  SETDIR            ;
      CALL  SERCHN            ;look for the next extent
      JR    SETF1       ;and loop till all found
;
;*****************************************************************
*******
;routine to open selected file
;
OPENF:      MVI   C,0FH       ;look at FCB through S1 and S2 bytes
      CALL  SEARCH            ;search for the file
      CALL  CHKCNT            ;was it found
      RZ                ;return if not
OPEN1:      CALL  GETEX       ;get the extent byte
      MOV   A,M         ;pull it into A
      PUSH  PSW         ;save it for later
      PUSH  H           ;save the address as well
      CALL  PNTDIR            ;point to the directory area
      XCHG              ;and put the address into DE
      LHLD  FCB         ;get the FCB address
      MVI   C,32        ;Bytes to move
      PUSH  D           ;save the directory address
      CALL  MOVE        ;Move C bytes DE to HL
      CALL  SETS28            ;set bit 7 of S2
      POP   D           ;recover the directory address
      LXI   H,12        ;set to point downstream
      DAD   D           ;now pointing to extent byte of directory
      MOV   C,M         ;pull this byte in
      LXI   H,15        ;ready to move down again
```

```
      DAD   D           ;now point to
      MOV   B,M         ;put the extent byte back here
      POP   H           ;recover the FCB address
      POP   PSW         ;and the entry extent byte
      MOV   M,A         ;put the byte back
      MOV   A,C         ;move the directory extent byte into A
      CMP   M           ;are they the same
      MOV   A,B         ;
      JRZ   OPEN2       ;if so jump
      MVI   A,0         ;otherwise get a zero
      JRC   OPEN2       ;if it was too big jump over
      MVI   A,80H       ;now get bit 7 set
OPEN2:      LHLD  FCB         ;point to the FCB again
      LXI   D,15        ;ready to move it
      DAD   D           ;now point to record count
      MOV   M,A         ;pull this out
      RET               ;and go back with it
;
;*****************************************************************
*******
;routine to check if (HL) is zero - if so move (DE) to (HL)
;
CHKCPY:     MOV   A,M         ;test (HL)
      INX   H           ;  for
      ORA   M           ;    zero
      DCX   H           ;point back
      RNZ               ;return if not zero
      LDAX  D           ;otherwise get (DE)
      MOV   M,A         ;put it at (HL)
      INX   D           ;bump
      INX   H           ;bothof them
      LDAX  D           ;get (DE)
      MOV   M,A         ;and put it at (DE)
      DCX   D           ;restore both
      DCX   H           ;  pointers
      RET               ;and return
;
;*****************************************************************
*******
;routine to close selected file
;
CLOSEF:     XRA   A           ;cheap zero
      STA   ODATA       ;put it at return info byte
      STA   COUNT       ;and zero out the count
      STA   COUNT+1           ;   word
      CALL  ISRO        ;is the disk read only
      RNZ               ;return if so - can't close it
      CALL  GETS2       ;get the S2 byte from the FCB
      ANI   80H         ;strip off the low eight bits
      RNZ               ;return if not zero
      MVI   C,0FH       ;otherwise set to look through S2 in FCB
      CALL  SEARCH            ;search for the file
      CALL  CHKCNT            ;was it there
      RZ                ;zero says done
```

```
      LXI   B,16        ;get the pointer for group bytes in dir
      CALL  PNTDIR            ;point to the directory
      DAD   B           ;add in the offset - pointing right
      XCHG              ;swap DE and HL
      LHLD  FCB         ;point to the FCB
      DAD   B           ;add in the offset again
      MVI   C,10H       ;get set for 16 bytes to move
CLOSF1:     LDA   BSIZE       ;get the block size byte out
      ORA   A           ;is it zero
      JRZ   CLOSF4            ;if so blocks > 1024 so jump past
;
;small blocks
;
      MOV   A,M         ;pull the byte out
      ORA   A           ;test for zero
      LDAX  D           ;get the (DE) out
      JRNZ  CLOSF2            ;if not zero jump over
      MOV   M,A         ;otherwise put A into place
CLOSF2:     ORA   A           ;test the byte again
      JRNZ  CLOSF3            ;if not zero jump past
      MOV   A,M         ;otherwise pull in the byte
      STAX  D           ;stuff it back at (DE)
CLOSF3:     CMP   M           ;are they the same
      JRNZ  CLOSF7            ;if not jump
      JR    CLOSF5            ;otherwise skip over
;
;big blocks
;
CLOSF4:     CALL  CHKCPY            ;check and increment
      XCHG              ;exchange again
      CALL  CHKCPY            ;check again for next byte
      XCHG              ;exchange back - ready to continue
      LDAX  D           ;get the new byte from (DE)
      CMP   M           ;does it match (HL)
      JRNZ  CLOSF7            ;if not over we go
      INX   D           ;bump pointer
      INX   H           ;   data
      LDAX  D           ;get next byte
      CMP   M           ;compare it as well
      JRNZ  CLOSF7            ;again if no match jump
      DCR   C           ;decrement the count
CLOSF5:     INX   D           ;increment the
      INX   H           ;  pointers
      DCR   C           ;decrement the count
      JRNZ  CLOSF1            ;if not done spinn around again
      LXI   B,-20       ;point back
      DAD   B           ;  20 bytes
      XCHG              ;swap pointers
      DAD   B           ;and back this one up as well
      LDAX  D           ;pull in the byte at (DE)
      CMP   M           ;test against (HL)
      JRC   CLOSF6            ;if (HL)>(DE) jump
      MOV   M,A         ;if not put A into (HL)
      LXI   B,3         ;get a 3
```

```
      DAD   B           ;point down 3
      XCHG              ;swap pointers again
      DAD   B           ;down three also
      MOV   A,M         ;get the byte at (HL)
      STAX  D           ;stuff it in at (DE)
CLOSF6:     MVI   A,0FFH            ;get the flag data
      STA   RWFLG1            ;set flag to say reading
      JMP   SETD1       ;write out the directory block and return
;
CLOSF7:     LXI   H,ODATA           ;point to output data byte
      DCR   M           ;decrement it 1
      RET               ;and go back
;
;*****************************************************************
*******
;make a new directory entry for a new file
;
MAKE: CALL  ROCHK       ; is the disk read only
      LHLD  FCB         ;nope - so get the FCB address
      PUSH  H           ;and save it on the stack
      LXI   H,TINFO           ;point to the TINFO area
      SHLD  FCB         ;save this as the FCB address
      MVI   C,01H       ;set to look at only the ET byte in the FCB
      CALL  SEARCH            ;search for the new file name
      CALL  CHKCNT            ;was it there
      POP   H           ;get the old FCB back
      SHLD  FCB         ;restore it right
      RZ                ;return if it was there
      XCHG              ;save FCB in DE
      LXI   H,15        ;get offset to record count
      DAD   D           ;and set the pointer there
      MVI   C,17        ;get set to spin through 16 bytes
      XRA   A           ;get a zero ready
MAK1: MOV   M,A         ;and move zero
      INX   H           ;  into
      DCR   C           ;    all
      JRNZ  MAK1        ;     the group bytes
      LXI   H,13        ;get the offset to the S1 byte
      DAD   D           ;and point to it
      MOV   M,A         ;zero this as well
      CALL  CHKINC            ;see if we have more to do
      CALL  MOVDIR            ;copy the directory in place
      JMP   SETS28            ;set bit 7 of S2 and return
;
;*****************************************************************
*******
;routine sets up for the next extent access for either read or write
;
NEXTEX:     XRA   A           ;get a zero
      STA   RWFLG1            ;say we are writing
      CALL  CLOSEF            ;close the file
      CALL  CHKCNT            ;did it exist
      RZ                ;return if not
      LHLD  FCB         ;get the FCB address
```

```
      LXI   B,12        ;set to point to extent byte
      DAD   B           ;now it does
      MOV   A,M         ;pull the thing in
      INR   A           ;bump by one
      ANI   1FH         ;strip off bit 7
      MOV   M,A         ;and put it back
      JRZ   NEXTE1            ;if it was zero jump over
      MOV   B,A         ;otherwise save it in B
      LDA   EXTMSK            ;get the extent mask
      ANA   B           ;and use it
      LXI   H,RWFLG1    ;and point to the flag
      ANA   M           ;and in this byte as well
      JRZ   NEXTE2            ;if zero jump past
      JR    NEXTE3            ;otherwise off to
;
NEXTE1:     LXI   B,2         ;ready to bump 2 more
      DAD   B           ;add it in
      INR   M           ;increment this byte
      MOV   A,M         ;then pull it in
      ANI   0FH         ;strip off the high 4 bits
      JRZ   NEXTE5            ;if zero jump
NEXTE2:     MVI   C,0FH       ;look through S2 byte in FCB
      CALL  SEARCH            ;look for the indicated file
      CALL  CHKCNT            ;see if it was there
      JRNZ  NEXTE3            ;if so open the next extent
      LDA   RWFLG2            ;get the read/write flag
      INR   A           ;add one
      JRZ   NEXTE5            ;if it was FF then return
      CALL  MAKE        ;make a new file directory entry
      CALL  CHKCNT            ;was it ok
      JRZ   NEXTE5            ;if not set up exit routine
      JR    NEXTE4            ;otherwise set up extent pointers and
return
;
NEXTE3:     CALL  OPEN1       ;go open the next extent
NEXTE4:     CALL  FIXEXT            ;set the pointers up
      XRA   A           ;zero A for the return
      JMP   GOBAK       ;and go back
;
NEXTE5:     CALL  JR1         ;set to return with problems
      JMP   SETS28            ;set bit 7 of S2 to say so
;
;*****************************************************************
*******
;routine to read the disk
;
DSKRED:     MVI   A,01H       ;get a !
      STA   DCODE       ;save it for the disk code byte
RRERR:      MVI   A,0FFH            ;get a read flag
      STA   RWFLG2            ;set it in for the control
      CALL  FIXEXT            ;go fix the extent
      LDA   ESCNT2            ;get the extent low byte
      LXI   H,RECCNT    ;point to the record count
      CMP   M           ;are they the same?
```

```
      JRC   DSKR1       ;if record count was bigger, jump over
      CPI   80H         ;was the count 80 hex?
      JRNZ  DERR        ;if not jump out with an error
      CALL  NEXTEX            ;go get the next extent
      XRA   A           ;set a=0
      STA   ESCNT2            ;save this as the new extent low byte
      LDA   ODATA       ;get the output data
      ORA   A           ;set the flags
      JRNZ  DERR        ;if it was not zero exit with error
DSKR1:      CALL  SECGRP            ;otherwise group from the sector
count
      CALL  TSTGRP            ;see if it is ok
      JRZ   DERR        ;if not go back
      CALL  COMSEC            ;if co compute the right sector
      CALL  GETD1       ;get the directory set right
      CALL  RDSEC       ;read the sector
      JMP   FIXREC            ;fix up the record and return
;
DERR: JMP   JR1         ;set the error exit and return
;
;*****************************************************************
*******
;routine to write the disk
;
DSKWRT:     MVI   A,01H       ;set the disk
      STA   DCODE       ;   code up
RWERR:      MVI   A,0         ;set the read write flag for write
      STA   RWFLG2            ;and save it for later
      CALL  ROCHK       ;see if the drive is protected
      LHLD  FCB         ;get the FCB address
      CALL  CHKFR1            ;see if the file is read only
      CALL  FIXEXT            ;fix up the extent
      LDA   ESCNT2            ;and get the low extent byte
      CPI   80H         ;is it greater that 80h
      JNC   JR1         ;if so exit with errors
      CALL  SECGRP            ;compute the right group
      CALL  TSTGRP            ;see if it is ok
      MVI   C,0         ;get a zero
      JRNZ  DSKW05            ;if ok jump over
      CALL  COMBLK            ;compute the block
      STA   EXTBLK            ;save it at extent block area
      LXI   B,0         ;get a 16 bit 0
      ORA   A           ;set flags
      JRZ   DSKW01            ;if zero jump over
      MOV   C,A         ;
      DCX   B           ;
      CALL  GETGRP            ;get the group counter out
      MOV   B,H         ;save it
      MOV   C,L         ;  in BC
DSKW01:     CALL  BLKAVL            ;see if a block is available
      MOV   A,L         ;check for
      ORA   H           ;  zero returned
      JRNZ  DSKW02            ;if not zero jump over
      MVI   A,02H       ;set the disk full message
```

```
      JMP   GOBAK       ;and return
;
DSKW02:     SHLD  GROUP       ;save the group counter
      XCHG              ;put it inti DE as well
      LHLD  FCB         ;get the FCB address
      LXI   B,16        ;set the offset to the record count
      DAD   B           ;and point to it
      LDA   BSIZE       ;get the block size
      ORA   A           ;is it zero
      LDA   EXTBLK            ;get the extent block out
      JRZ   DSKW03            ;if blocks are greater than 1024 jump
      CALL  PNTD1       ;point to the directory entry
      MOV   M,E         ;save in the group low byte
      JR    DSKW04            ;and jump over
;
DSKW03:     MOV   C,A         ;save the extblk byte into C
      MVI   B,0         ;zero out B
      DAD   B           ;slide the offset
      DAD   B           ;   right 2 bits
      MOV   M,E         ;put the group low byte into place
      INX   H           ;point to the high byte
      MOV   M,D         ;and put this in too
DSKW04:     MVI   C,02H       ;set the exit code
DSKW05:     LDA   ODATA       ;and the exit data
      ORA   A           ;see if it is zero
      RNZ               ;return if not
      PUSH  B           ;otherwise save the extent block pointer
      CALL  COMSEC            ;go compute the sector
      LDA   DCODE       ;get the disk code out
      DCR   A           ;count it
      DCR   A           ;  down 2
      JRNZ  DSKW08            ;if not zero jump over
      POP   B           ;recover the block offset
      PUSH  B           ;save it again
      MOV   A,C         ;put low into A
      DCR   A           ;decrement it by 1
      DCR   A           ;   and 1 more
      JRNZ  DSKW08            ;if not zero then jump
      PUSH  H           ;save the address of the directory group
      LHLD  DTEMP       ;get the temp address
      MOV   D,A         ;save the group low byte into D
DSKW06:     MOV   M,A         ;and put it into memory
      INX   H           ;point to the high
      INR   D           ;add one to D
      JP    DSKW06            ;if positive jump past
      CALL  SETEMP            ;set the temp buffer
      LHLD  DIRPNT            ;get the directory pointer
      MVI   C,02H       ;put a 2 into C
DSKW07:     SHLD  GROUP       ;save the group pointer
      PUSH  B           ;push the counter onto the stack
      CALL  GETD1       ;get the next directory
      POP   B           ;recover the group pointer
      CALL  WRSEC       ;write the sector
      LHLD  GROUP       ;get the group data
```

```
      MVI   C,0         ;
      LDA   BLKMSK            ;get the block mask
      MOV   B,A         ;put it into B
      ANA   L           ;and in L
      CMP   B           ;compare with B
      INX   H           ;and bump to next
      JRNZ  DSKW07            ;if not the same jump over
      POP   H           ;recover HL
      SHLD  GROUP       ;and save it at GROUP
      CALL  SETDMA            ;set the dma address
DSKW08:     CALL  GETD1       ;and get the next directory
      POP   B           ;recover BC
      PUSH  B           ;and save it again
      CALL  WRSEC       ;write the sector
      POP   B           ;restore BC again
      LDA   ESCNT2            ;get the extent count low byte
      LXI   H,RECCNT    ;point to the record count
      CMP   M           ;are they the same
      JRC   DSKW09            ;if RECCNT>ESCNT2 jump
      MOV   M,A         ;otherwise save a new RECCNT
      INR   M           ;bump pointer by one
      MVI   C,02H       ;get a 2 to force fall through below

;     THE D.R.I. PATCH FOR DEBLOCKED BIOS TYPES

DSKW09:
      PUSH  PSW         ;save the flags
      CALL  GETS2       ;point to S2 byte
      ANI   7FH         ;strip off bit 7
      MOV   M,A         ;and put it back this way
      POP   PSW         ;recover the flags
DSKW10:     CPI   7FH         ;test the byte in A
      JRNZ  DSKW12            ;if not 7F jump over
      LDA   DCODE       ;get the disk code out
      CPI   01H         ;is it 1
      JRNZ  DSKW12            ;if not jump
      CALL  FIXREC            ;if so fix the record bytes
      CALL  NEXTEX            ;and get the next extent ready
      LXI   H,ODATA           ;point to the output data byte
      MOV   A,M         ;pull it in
      ORA   A           ;check for zero
      JRNZ  DSKW11            ;if not jump
      DCR   A           ;decrement it by one if it was zero
      STA   ESCNT2            ;and save this at ESCNT2
DSKW11:     MVI   M,0         ;zero out ODATA
DSKW12:     JMP   FIXREC            ;fix the records up and return
;
;*****************************************************************
*******
;random file access routine - C has the code for access type
;     FF for read, 0 for write
;
RANFIL:     XRA   A           ;zero the DCODE
      STA   DCODE       ;   byte
```

```
RANF1:      PUSH  B           ;save the entry code in C
      LHLD  FCB         ;point to the FCB address
      XCHG              ;and save it in DE for later
      LXI   H,33        ;get the offset to R0
      DAD   D           ;pointing to R0 byte
      MOV   A,M         ;pull it in
      ANI   7FH         ;and strip off the sector count for this ext
      PUSH  PSW         ;save it on the stack
      MOV   A,M         ;get R0 again
      RAL               ;slide it left
      INX   H           ;point to R1
      MOV   A,M         ;pull this in
      RAL               ;rotate this left also pulling in bit 7 of R0
      ANI   1FH         ;strip off the low 5 bits for extent count
      MOV   C,A         ;save this in C
      MOV   A,M         ;get R1 again
      RAR               ;rotate
      RAR               ;   it
      RAR               ;     right
      RAR               ;       4 bits
      ANI   0FH         ;and mask off the block counter
      MOV   B,A         ;put it into B
      POP   PSW         ;A has record, B has block, C has extent
                        ;   (0-127)      (0-15)       (0-31)
      INX   H           ;point to the overflow byte
      MOV   L,M         ;pull it into L
      INR   L           ;add one
      DCR   L           ;then sub 1 to set flag
      MVI   L,06H       ;get seek past end of disk error
      JRNZ  RANF5       ;if not zero exit with error
      LXI   H,32        ;offset to current record
      DAD   D           ;point to it
      MOV   M,A         ;put the current record byte into place
      LXI   H,12        ;now get the
      DAD   D           ;   extent pointer
      MOV   A,C         ;pull the new one into A
      SUB   M           ;compare with the one we are looking at
      JRNZ  RANF2       ;if not right, jump over
      LXI   H,14        ;otherwise check the
      DAD   D           ;   S2 byte
      MOV   A,B         ;put the new block into A
      SUB   M           ;and compare with what is there
      ANI   7FH         ;strip off bit 7 for the check
      JRZ   RANF3       ;if all is well jump back
RANF2:      PUSH  B           ;otherwise save the counters
      PUSH  D           ;and the FCB address
      CALL  CLOSEF            ;and close the current extent
      POP   D           ;recover the FCB address
      POP   B           ;and the counters
      MVI   L,03H       ;set the close error message
      LDA   ODATA       ;check the result
      INR   A           ;  should be >0 if ok
      JRZ   RANF4       ;if not we had problems so go back
      LXI   H,12        ;point to
```

```
      DAD   D           ;  the extent byte
      MOV   M,C         ;put in the new value
      LXI   H,14        ;now point to
      DAD   D           ;  the S2 byte
      MOV   M,B         ;put the new block in place
      CALL  OPENF       ;go open the new extent
      LDA   ODATA       ;get the result
      INR   A           ;and check if ok
      JRNZ  RANF3       ;if non zero the open was ok so go back
      POP   B           ;recover the entry code
      PUSH  B           ;then resave it
      MVI   L,04H       ;set the seek to unwritten data error
      INR   C           ;bump the entry code 1
      JRZ   RANF4       ;if it is zero we exit with the error
      CALL  MAKE        ;otherwise we are writing so open new extent
      MVI   L,05H       ;get the directory overflow error
      LDA   ODATA       ;get the result
      INR   A           ;add one to check
      JRZ   RANF4       ;if errors exit
RANF3:      POP   B           ;otherwise restore the entry code
      XRA   A           ;set zero into A
      JMP   GOBAK       ;and return with FCB set right
;
RANF4:      PUSH  H           ;save the current error byte in L
      CALL  GETS2       ;point to the S2 byte
      MVI   M,0C0H            ;set in a C0 hex
      POP   H           ;recover the error byte
RANF5:      POP   B           ;recover the entry code
      MOV   A,L         ;set the error byte into A
      STA   ODATA       ;save it for the exit
      JMP   SETS28            ;set bit 7 of S2 for later
;
;*****************************************************************
*******
;routine to read random record
;
RDREC:      MVI   C,0FFH            ;set the code for reading
      CALL  RANFIL            ;call the random routine to fix FCB up
      CZ    RRERR       ;If ok then read the record
      RET               ;and return
;
;*****************************************************************
*******
;routine to write random record
;
WRREC:      MVI   C,0         ;set the code for writing
      CALL  RANFIL            ;go fix the FCB up
      CZ    RWERR       ;if ok then write the sector
      RET               ;and return
;
;*****************************************************************
*******
;routine computes record count for random access
;
```

```
COMREC:     XCHG              ;put the FCB into DE - offset into HL
      DAD   D           ;set the pointer to the record byte
      MOV   C,M         ;pull the record count into C
      MVI   B,0         ;zero out B
      LXI   H,12        ;now set offset
      DAD   D           ;  to point to the extent byte
      MOV   A,M         ;pull it into A
      RRC               ;slide right through carry - carry has bit0
      ANI   80H         ;strip all but the bit in 7
      ADD   C           ;add in the current record count
      MOV   C,A         ;and save this in C
      MVI   A,0         ;zero A leaving carry set
      ADC   B           ;add in the carry and zero in B
      MOV   B,A         ;save this in B
      MOV   A,M         ;get the extent byte again
      RRC               ;slide right through carry
      ANI   0FH         ;mask all but low four
      ADD   B           ;add in the data in B
      MOV   B,A         ;and save back into B
      LXI   H,14        ;set the pointer
      DAD   D           ;  to the S2 byte
      MOV   A,M         ;pull it into A
      ADD   A           ;and slide
      ADD   A           ;   it
      ADD   A           ;    left
      ADD   A           ;     four bits
      PUSH  PSW         ;save the result on the stack
      ADD   B           ;add in B, if s2>=16 and B=17 carry is set
      MOV   B,A         ;save the result into B
      PUSH  PSW         ;save the flags and A
      POP   H           ;pull flags into L
      MOV   A,L         ;put then into A
      POP   H           ;recover the second flag set into L
      ORA   L           ;or the two together
      ANI   01H         ;and strip off all but carry
      RET               ;then return
;
;*****************************************************************
*******
;routine gets the record count from the last extent of a random file
;
GETSIZ:     MVI   C,0CH       ;look at file name ant extent
      CALL  SEARCH            ;go locate the file
      LHLD  FCB         ;point to the FCB
      LXI   D,33        ;get the offset to the
      DAD   D           ;   R0 byte
      PUSH  H           ;save the address on the stack
      MOV   M,D         ;put the 0 into position
      INX   H           ;point to R1
      MOV   M,D         ;0 goes there as well
      INX   H           ;point to R2 - overflow byte
      MOV   M,D         ;once more with zero
GSLOOP:     CALL  CHKCNT            ;check the counter
      JRZ   GSEXIT            ;if zero we exit now
```

```
      CALL  PNTDIR            ;otherwise point to the directory
      LXI   D,15        ;get the offset to extent byte
      CALL  COMREC            ;check extent overflow
      POP   H           ;recover the R0 address
      PUSH  H           ;save it again
      MOV   E,A         ;save the directory code in E
      MOV   A,C         ;and put the R0 byte into A
      SUB   M           ;subtract the byte from the FCB
      INX   H           ;point to the R1 byte
      MOV   A,B         ;put B in A
      SBB   M           ;subtract with borrow from here as well
      INX   H           ;once more with the overflow byte
      MOV   A,E         ;E into A
      SBB   M           ;subtract the overflow
      JRC   GETSZ1            ;if carry jump over - file is too big
      MOV   M,E         ;put R2 into place
      DCX   H           ;point back
      MOV   M,B         ;set R1 in as well
      DCX   H           ;back again
      MOV   M,C         ;and set R0 into place
GETSZ1:     CALL  SERCHN            ;go look for next extent
      JR    GSLOOP            ;loop until we hit the end
;
GSEXIT:     POP   H           ;recover the address of random file size
      RET               ;and return with it
;
;*****************************************************************
*******
; BDOS function 36 - set random record with data set by RANFIL
;
SETRND:     LHLD  FCB         ;get the FCB address
      LXI   D,32        ;and the offset to the record count
      CALL  COMREC            ;get the file pointers into A b and c
      LXI   H,33        ;offset to the R0 byte
      DAD   D           ;point to it
      MOV   M,C         ;put the byte in C into R0
      INX   H           ;point to R1
      MOV   M,B         ;put B in there
      INX   H           ;point to overflow byte
      MOV   M,A         ;stuff A there
      RET               ;and return
;
;*****************************************************************
*******
;routine to log the selected disk
;
LOGDSK:     LHLD  DLOG        ;point to the login vector
      LDA   CURDSK            ;get the current disk byte
      MOV   C,A         ;and put it into C
      CALL  SPINHL            ;spin the selected disk bit into place
      PUSH  H           ;save this vector on the stack
      XCHG              ;careful study says this is useless
      CALL  DISKID            ;go get the disk information set
      POP   H           ;recover the vector
```

```
      CZ    PTSERR            ;if trouble say so and quit
      MOV   A,L         ;otherwise put the low byte into A
      RAR               ;slide it right bit 0 into carry
      RC                ;return if bit was set - we already got it
      LHLD  DLOG        ;otherwise point to the login vector
      MOV   C,L         ;put the vector
      MOV   B,H         ;   into BC
      CALL  SETDBT            ;set the disk bit using setro
      SHLD  DLOG        ;save the vector into place again
      JMP   GETAL       ;get the new allocation vector
;
;*****************************************************************
*******
; BDOS function 14: select disk
;     enter with selected disk in IDATA
;
LOGIN:      LDA   IDATA       ;get the disk to log
      LXI   H,CURDSK    ;point to the current disk storage
      CMP   M           ;are they the same
      RZ                ;return if so
      MOV   M,A         ;otherwise make this current
      JR    LOGDSK            ;and log it in then return
;
;*****************************************************************
*******
;routine to select disk indicated in FCB drive code
;
RSELCT:     MVI   A,0FFH            ;get an FF
      STA   OUT1        ;save it to force disk reset on exit
      LHLD  FCB         ;point to the FCB
      MOV   A,M         ;pull in the new drive code
      ANI   1FH         ;strip off the user bits
      DCR   A           ;decrement to make it right
      STA   IDATA       ;save it for the login routines
      CPI   1EH         ;is it too big
      JNC   XIT         ;if so go back
      LDA   CURDSK            ;get the current disk out
      STA   OUTDSK            ;save it for the return
      MOV   A,M         ;pull in the new code again
      STA   OLDDSK            ;save the value as the olddisk
      ANI   0E0H        ;strip off the drive bits
      MOV   M,A         ;put the user bits back
      CALL  LOGIN       ;go log the new disk
XIT:  LDA   USRCOD            ;get the user code out
      LHLD  FCB         ;point back to the FCB
      ORA   M           ;set in the new user code bits
      MOV   M,A         ;and put the result back
      RET               ;then return
;
;*****************************************************************
*******
;BDOS function 12: routine returns CP/M version in A
;
GETVER:     MVI   A,22H       ;say this is CP/M version 2.2
```

```
      JMP   GOBAK       ;and return
;
;*****************************************************************
*******
; BDOS function 13: routine to reset the system disk
;
RESET:      LXI   H,0         ;get ready to reset the vectors
      SHLD  ROVEC       ;set both read only bits
      SHLD  DLOG        ;   and login bits to off
      XRA   A           ;zero A
      STA   CURDSK            ;  and make it the current disk
      LXI   H,TBUFF           ;Set DMA addr to 80H
      SHLD  DMAADR            ;save this as the new DMA value
      CALL  SETDMA            ;make it so for real
      JR    LOGDSK            ;go log in drive A and return
;
;*****************************************************************
*******
; BDOS function 15: open file
;     enter with FCB address in HL and INFO
;     exit with directory code in A
;
OPEN: CALL  ZS2         ;set S2 to zero for the new file
      CALL  RSELCT            ;select the right disk
      JMP   OPENF       ;then open the file in the FCB
;
;*****************************************************************
*******
; BDOS function 16: close the selected file
;     enter with FCB address
;     exit with directory code in A
;
CLOSE:      CALL  RSELCT            ;select the right drive
      JMP   CLOSEF            ;and finish off the close operation
;
;*****************************************************************
*******
; BDOS function 17: search for first file entry
;     enter with FCB address
;     exit with directory code in A
;
SEAR1:      MVI   C,0         ;set the search code into C
      XCHG              ;swap FCB into DE
      MOV   A,M         ;pull in the file name byte
      CPI   '?'         ;is it a wild card
      JRZ   SRF2        ;if so jump over
      CALL  GETEX       ;otherwise get the extent
      MOV   A,M         ;pull in the byte here
      CPI   '?'         ;is it a wild card
      CNZ   ZS2         ;if not set S2 to 0
      CALL  RSELCT            ;select the right drive
      MVI   C,0FH       ;look at name thru S2 byte
SRF2: CALL  SEARCH            ;search for the file
      JMP   MOVSEC            ;jump to move the sector and return
```

```
;
;*****************************************************************
*******
;BDOS function 18: search for next directory entry
;     enter with FCB set as before
;     exit with directory code in A 255 says not found
;
SEARN:      LHLD  SEARA       ;get the last search FCB out
      SHLD  FCB         ;save this as the FCB
      CALL  RSELCT            ;select the right drive
      CALL  SERCHN            ;then search for the nex extent
      JMP   MOVSEC            ;move the sector and return
;
;*****************************************************************
*******
; BDOS function 19: delete file
;     enter with FCB address set
;     exit with directory code in A
;
DELETE:     CALL  RSELCT            ;select the right drive
      CALL  DELETF            ;delete the indicated file
      JMP   RETCOD            ;set the return code and go back
;
;*****************************************************************
*******
; BDOS function 20: read sequential
;     enter with FCB address
;     exit with directory code in A
;
READ: CALL  RSELCT            ;select the indicated drive
      JMP   DSKRED            ;and go read the disk
;
;*****************************************************************
*******
; BDOS function 21: write sequential
;     enter with FCB address
;     exit with directory code in A
;
WRITE:      CALL  RSELCT            ;select the drive
      JMP   DSKWRT            ;and go write the disk
;
;*****************************************************************
*******
; BDOS function 22: make new file entry
;     enter with FCB
;     exit with directory code in A
;
CREATE:     CALL  ZS2         ;zero out S2
      CALL  RSELCT            ;select the drive
      JMP   MAKE        ;and make a new directory entry
;
;*****************************************************************
*******
; BDOS function 23: rename file
```

```
;     enter with FCB address
;     exit with directory code in A
;
RENAME:     CALL  RSELCT            ;select the drive
      CALL  RENAMF            ;and go rename the file
      JMP   RETCOD            ;set the return code and exit
;
;*****************************************************************
*******
; BDOS function 24: return login vector
;     exit with login vector in HL
;
GLOGIN:     LHLD  DLOG        ;get the login vector out
      JMP   RETHL       ;return with it
;
;*****************************************************************
*******
; BDOS function 25: return current disk
;     exit with current disk in A
;
GETDRV:     LDA   CURDSK            ;get the current disk out
      JMP   GOBAK       ;and go back with it
;
;*****************************************************************
*******
; BDOS function 26: set DMA address
;     enter with DMA address set
;
DMASET:     XCHG              ;save the new address into HL
      SHLD  DMAADR            ;save it into place
      JMP   SETDMA            ;then make it true
;
;*****************************************************************
*******
; BDOS function 27: get allocation vector address
;     return address in HL
;
GALLOC:     LHLD  ALLOCA            ;get the allocation vector out
      JMP   RETHL       ;and go back with it
;
;*****************************************************************
*******
; BDOS function 29: get read/only vector
;     return vector in HL
;
GROVEC:     LHLD  ROVEC       ;get the read only vector out
      JMP   RETHL       ;and go back with it
;
;*****************************************************************
*******
; BDOS function 30: set file attributes
;     enter with FCB address with new attributes set in F1..F7,T1..T3
;     exit with directory code in A
;
```

```
SETATT:     CALL  RSELCT            ;select the indicate drive
      CALL  SETFN       ;set in the new file name with bits set
      JMP   RETCOD            ;set the return code and exit
;
;*****************************************************************
*******
; BDOS function 31: get address of disk parameter table
;     return address in HL
;
GETPAR:     LHLD  DPB         ;get the disk parameter table address
RETHL:      SHLD  ODATA       ;save it at return data word
      RET               ;and go back with it
;
;*****************************************************************
*******
; BDOS function 32: get or set user code
;     enter with LINFO set to FF for get and to value for set
;
MODUSR:     LDA   IDATA       ;get the input argument
      CPI   0FFH        ;do we do a set
      JRNZ  SETCODE           ;or a get
      LDA   USRCOD            ;must be get so get
      JMP   GOBAK       ;and go back with it
;
SETCODE:ANI 1FH         ;otherwise strip off the high bits
      STA   USRCOD            ;and save the new value
      RET               ;then return
;
;*****************************************************************
*******
; BDOS function 33: read random record
;     enter with FCB address
;     exit with return code in A
;
REDRND:     CALL  RSELCT            ;select the right drive
      JMP   RDREC       ;and go read the sector
;
;*****************************************************************
*******
; BDOS function 34: write random record
;     enter with FCB address
;     exit with return code in A
;
WRTRND:     CALL  RSELCT            ;go select the drive
      JMP   WRREC       ;and write the sector
;
;*****************************************************************
*******
; BDOS function 35: compute file size
;     enter with FCB address
;     exit with random record set in FCB
;
FILSIZ:     CALL  RSELCT            ;select the right drive
      JMP   GETSIZ            ;and compute the file size
```

```
;
;*****************************************************************
*******
; BDOS function 37: Routine to reset drive
;     enter with drive vector for reset
;     return zero in A
;
RESDRV:     LHLD  FCB         ;has the drive vector for reset
      MOV   A,L         ;move low into A
      CMA               ;and flip it
      MOV   E,A         ;put the result back into E
      MOV   A,H         ;get the high vector byte
      CMA               ;  flip itr alse
      LHLD  DLOG        ;get the login vector
      ANA   H           ;and in the new bits in high
      MOV   D,A         ;save the result into D
      MOV   A,L         ;now on the the low bits
      ANA   E           ;and in the new bits
      MOV   E,A         ;and save the result into E
      LHLD  ROVEC       ;point to the ro vector
      XCHG              ;swap DE and HL
      SHLD  DLOG        ;put the new login vector into place
      MOV   A,L         ;get the read only low bits
      ANA   E           ;and in the new bits
      MOV   L,A         ;put then back into L
      MOV   A,H         ;now on the the high bits
      ANA   D           ;set the new ones in
      MOV   H,A         ;and put the result back
      SHLD  ROVEC       ;then save the new value into place
      RET               ;and return when done
;
;*****************************************************************
*******
; bdos exit routine
;
BEXIT:      LDA   OUT1        ;get the out flag for drive reset
      ORA   A           ;set the flags
      JRZ   REXIT       ;if zero we may exit as is
      LHLD  FCB         ;otherwise get the FCB address
      MVI   M,0         ;put a zero into drive select byte
      LDA   OLDDSK            ;get the old disk out
      ORA   A           ;was it zero
      JRZ   REXIT       ;if so exit
      MOV   M,A         ;otherwise set in the old disk byte
      LDA   OUTDSK            ;get the disk to relog
      STA   IDATA       ;save it for the login
      CALL  LOGIN       ;go log in the old disk
REXIT:      LHLD  OLDSP       ;get the old stack pointer out
      SPHL              ;put it back in place
      LHLD  ODATA       ;get the output data into place
      MOV   A,L         ;put L into A
      MOV   B,H         ;and H into B
      RET               ;then return from BDOS happy
;
```

```
;*****************************************************************
*******
; BDOS function 40: Routine to write random file with zero fill
;     enter with FCB address
;     exit with return code in A
;
ZERRND:     CALL  RSELCT            ;select the desired drive
      MVI   A,2         ;get the code for zero fill
      STA   DCODE       ;set it into place
      MVI   C,0         ;zero c as well
      CALL  RANF1       ;then off the finish the job set pointers
      CZ    RWERR       ;then write the sector if all is well
      RET               ;and return
;
;*****************************************************************
*******
;bdos data area follows
;
TINFO:      DB    0E5H        ;

LOGFLAG:DW  0           ;FIRST DRIVE SELECT

ROVEC:      DW    0           ;read only vector address storage
DLOG: DW    0           ;disk login areax
DMAADR:     DW    0080H       ;DMA address defaults to 80H
SCRT0:      DW    0           ;
SCRT1:      DW    0           ;holds current track
SCRT2:      DW    0           ;holds starting sector number of current
track
DTEMP:      DW    0           ;temp DMA address storage
DPB:  DW    0           ;disk parameter block address
CHANGE:     DW    0           ;scratch storage for disk change check
ALLOCA:     DW    0           ;allocation vector area
DPBLK:      DW    0           ;disk parm block - number of sectors/trk
BLSHFT:     DB    0           ;block shift factor
BLKMSK:     DB    0           ;block mask
EXTMSK:     DB    0           ;null mask
DSIZE:      DW    0           ;disk size
DIRMAX:     DW    0           ;directory max
AL01: DW    0           ;Allocation storage
CKSIZE:     DW    0           ;number of entries to check
TOFS: DW    0           ;track offset
TRANS:      DW    0           ;translation routine address holder
RWFLG1:     DB    0           ;flag indicates read or write
RWFLG2:     DB    0           ;flag also used to indicate read or write
SCODE1:     DB    0
DCODE:      DB    0           ;disk code byte
IDATA:      DB    0           ;input data storage area
EXTBLK:     DB    0
SCODE:      DB    0
SEARA:      DB    0
      DB    0
      DB    0
      DB    0
```

```
BSIZE:      DB    0           ;block size byte
OUT1: DB    0           ;used to force relog of disk on exit
OUTDSK:     DB    0           ;BDOS exit disk
OLDDSK:     DB    0           ;disk entered with
RECCNT:     DB    0           ;record count holder
ESCNT1:     DB    0           ;extent recort count high byte
ESCNT2:     DB    0           ;extent record count low byte
      DB    0
GROUP:      DB    0,0,0       ;group number storage

DIRPNT:     DB    0,0,0       ;directory pointer area

DIROFF:     DB    0           ;offset into the directory area

COUNT:      DB    0,0,0       ;holds 16 bit counter for looping

LDIRPNT:DB  0,0,0

      DB    'PATCH AREA START'

;
; BIOS Entry Vector Definitions
;
; (These are actually in CBIOS)
;




;     DS    3           ;(BOOT) Cold boot entry
;WBOOTF:    DS    3           ;Warm boot entry
;CONSF:     DS    3           ;Console status
;CONIF:     DS    3           ;Console char in
;CONOF:     DS    3           ;Console char out
;LISTF:     DS    3           ;List char out
;PUNF:      DS    3           ;Punch char out
;READF:     DS    3           ;Reader char in
;HOMF:      DS    3           ;Home disk
;SELF:      DS    3           ;Select disk
;TRKF:      DS    3           ;Set disk track addr
;SECF:      DS    3           ;Set disk sector addr
;DMAF:      DS    3           ;Set DMA buffer addr
;DRDF:      DS    3           ;Read sector
;DWRF:      DS    3           ;Write sector
;     DS    3           ;
;SECTRN:    DS    3           ;Sector translation routine
;



      END
```