

DIGITAL RESEARCH

Post Office Box 579, Pacific Grove, California 93950, (408) 373-3403

BASIC DISK OPERATING SYSTEM (BDOS)

CP/M VERSION _____

COPYRIGHT © 1976

DIGITAL RESEARCH

P. O. BOX 579
PACIFIC GROVE, CA. 93950

SER. # _____

```

1>
2>NDECLARE FDOS LITERALLY '3200H',
3>    /* CP/M BDOS
4>        COPYRIGHT (C) 1976
5>        DIGITAL RESEARCH
6>        BOX 579, PACIFIC GROVE
7>        CALIFORNIA, 93950
8>    */
9>FDOS, DECLARE BOOT LITERALLY 'UBOOT';
10>/*
11>    */
12>/* CP/M BASIC I/O SYSTEM */ V1.05 */
13>
14>NDECLARE BASE LITERALLY '3E00H', /* DISK INTERFACE AND CONSOLE IO */
15>/* THE FOLLOWING SUBROUTINES ARE ASSUMED TO EXIST,
16>STARTING AT THE ADDRESS 'BASE' */
17>
18>    BASE:   BOOT      SYSTEM REBOOT OPERATION
19>    BASE+3  WBOOT     SYSTEM REBOOT - WARM START
20>    BASE+6  CONSTAT   CONSOLE STATUS - RETURNS
21>                0 IN REG-A IF NO CONSOLE DATA READY,
22>                FF IF CHARACTER IS READY
23>    BASE+9  CONIN     CONSOLE CHARACTER INTO ACCUMULATOR - 0 PARITY
24>    BASE+12 CONOUT    CONSOLE CHARACTER SENT FROM REGISTER C
25>    BASE+15 LIST      SEND CHARACTER FROM REGISTER C TO LIST DEVICE
26>    BASE+18 PUNCH     SEND CHARACTER FROM REGISTER C TO PUNCH DEVICE
27>    BASE+21 READER    READ CHARACTER TO REGISTER A WITH 0 PARITY
28>    BASE+24 HOME     MOVE DISK HEAD TO TRACK 0
29>    BASE+27 SELDSK    SELECT DISK DRIVE GIVEN BY REGISTER C (0,1,...)
30>    BASE+30 SETTRK    SET TRACK (0-76) GIVEN BY REGISTER C.
31>    BASE+33 SETSEC    SET SECTOR NUMBER GIVEN BY REG C (1-26)
32>    BASE+36 SETDMA   SET DMA ADDRESS GIVEN BY REG PAIR B,C (INITIALLY
33>Y
34>                DEFAULTED TO 86H)
35>    BASE+39 READ     READ DISK SECTOR (SETTRK, SETSEC, SELDSK ASSUME
36>HD)
37>
38>        ERROR RETURNS INN REGISTER A IN THREE
39>        LEAST SIGNIFICANT BITS (2,1, AND 0)
40>        0      HARDWARE MALFUNCTION
41>        1      DRIVE NOT READY
42>        2      COMMAND SEQUENCE ERROR
43>    BASE+42 WRITE    WRITE DISK SECTOR (SETTRK..SELDSK ASSUMED)
44>                ERROR RETURNS IN REGISTER A AS ABOVE
45>
46>CP/M ALSO PROVIDES A TEN BYTE AREA IMMEDIATELY AHEAD OF THE
47>DISK AND CONSOLE INTERFACE FOR TEMPORARY STORAGE IN CASE THE
48>INTERFACE IS IMPLEMENTED IN ROM
49>/*
50>
51>
52>
53>
54>NDISKMON: PROCEDURE(FUNC,INFO) ADDRESS;
55>    DECLARE COPYRIGHT DATA(
56>        'COPYRIGHT (C) 1976, DIGITAL RESEARCH');
57>
58>    DECLARE FUNC BYTE,
59>        LIINFO BYTE, /* LOW ORDER INFO */
60>        INFO ADDRESS,

```

CP/M VERSION 1.3 **1.0**
 COPYRIGHT © 1976
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950
 SER. # BDOS

```

61>    RET ADDRESS, RET BYTE,
62>
63>    /* FUNC IS THE DISK MONITOR FUNCTION NUMBER AS SHOWN BELOW */
64>    0:  SYSTEM RESET
65>    1:  READ CONSOLE DEVICE
66>    2:  WRITE CONSOLE DEVICE
67>    3:  READ READER DEVICE
68>    4:  WRITE PUNCH DEVICE
69>    5:  WRITE LIST DEVICE
70>    6:  INTERROGATE MEMORY SIZE
71>    7:  INTERROGATE DEVICE STATUS
72>    8:  CHANGE DEVICE STATUS
73>    9:  PRINT BUFFER ON CONSOLE
74>    10: READ BUFFER FROM CONSOLE
75>    11: CONSOLE CHARACTER READY
76>    12: LIFT HEAD (NO OPERATION ON CPM 1602JUN75)
77>    13: RESET DISK SYSTEM - SELECT DISK 0
78>    14: SELECT DISK 'INFO'
79>    15: OPEN FILE
80>    16: CLOSE FILE
81>    17: SEARCH FOR FIRST OCCURRENCE
82>    18: SEARCH FOR NEXT OCCURRENCE
83>    19: DELETE A FILE
84>    20: READ A FILE
85>    21: WRITE A FILE
86>    22: CREATE A FILE
87>    23: RENAME A FILE
88>    24: RETURN LOGIN VECTOR - EACH BIT CORRESPONDS TO
89>        A DISK NUMBER, FROM LSB TO MSB. 1 INDICATES
90>        THE DISK IS LOGGED IN.
91>    25: RETURN CURRENTLY SELECTED DISK NUMBER
92>    26: SET SUBSEQUENT DMA ADDRESS
93>    27: RETURN BASE ADDRESS OF ALLOCATION VECTOR
94>        (USED TO DETERMINE REMAINING SPACE)
95>    */
96>
97>NDECLARE
98>    EQU LITERALLY 'LITERALLY',
99>    BOOTF  EQU '3C$80H',
100>   H$00T  EQU '3E$03H',
101>   CUNSF  EQU '3E$06H',
102>   CONIF  EQU '3E$09H',
103>   CONOF  EQU '3E$0CH',
104>   LISTF  EQU '3E$0FH',
105>   PUNF   EQU '3E$12H',
106>   READF  EQU '3E$15H',
107>   HOMF   EQU '3E$18H',
108>   SELF   EQU '3E$1BH',
109>   TRKF   EQU '3E$1EH',
110>   SECf   EQU '3E$21H',
111>   DMAf   EQU '3E$24H',
112>   DRDF   EQU '3E$27H',
113>   DWRF   EQU '3E$2AH',
114>
115>NDECLARE TRUE LITERALLY '1',
116>    FALSE LITERALLY '0',
117>
118>NDECLARE CHAR$RDY BYTE INITIAL(FALSE), /* TRUE IF CHAR READ */
119>    KB$CHAR BYTE, /* VALUE OF CHARACTER WHEN CHAR$RDY IS TRUE */
120>    LISTCOPY BYTE, /* TRUE IF COPYING TO LIST DEVICE */

```

CP/M VERSION _____
 COPYRIGHT © 1976
 DIGITAL RESEARCH
 P.O. BOX 579
 PACIFIC GROVE, CA 93950
 SER. #

```

121>CONRDY: PROCEDURE BYTE;
122>    /* RETURN TRUE IF CHAR READY AT CONSOLE */
123>    GO TO CONSF;
124>    END CONRDY;
125>
126>CONIN: PROCEDURE BYTE;
127>    /* READ NEXT CONSOLE CHARACTER */
128>    IF CHAR$RDY THEN /* CHARACTER IS READY */
129>        DO; CHAR$RDY = FALSE; RETURN KB$CHAR;
130>    END;
131>
132>    /* OTHERWISE READ THE CHARACTER */
133>    GO TO CONIF;
134>    END CONIN;
135>
136>CONBRK: PROCEDURE BYTE;
137>    DECLARE CTLC LITERALLY '83H';
138>    DECLARE CTLS LITERALLY '13H';
139>    IF CHAR$RDY THEN RETURN TRUE; /* CHARACTER ALREADY READ */
140>    IF CONRDY THEN /* CHECK FOR TYPE TERMINATION FUNCTION */
141>        DO;
142>            IF (KB$CHAR := CONIN) = CTLS THEN /* STOP TYPE */
143>                DO; IF CONIN = CTLA THEN, GO TO BOOT;
144>                RETURN FALSE;
145>            ENDJ;
146>        RETURN; (CHAR$RDY := TRUE);
147>    END;
148>    RETURN FALSE;
149>    END CONBRK;
150>
151>CONCHAR: PROCEDURE(CHAR);
152>    DECLARE CHAR BYTE;
153>    GO TO CONOF;
154>    END CONCHAR;
155>
156>CONOUT: PROCEDURE(CHAR);
157>    DECLARE CHAR BYTE;
158>    /* CHECK FOR BREAK CHARACTER */
159>    IF CONRK THEN;
160>    /* SEND CONSOLE CHARACTER */
161>    CALL CONCHAR(CHAR);
162>    END CONOUT;
163>
164>LSTOUT: PROCEDURE(CHAR);
165>    DECLARE CHAR BYTE;
166>    GO TO LISTF;
167>    END LSTOUT;
168>
169>PUNOUT: PROCEDURE(CHAR);
170>    DECLARE CHAR BYTE;
171>    GO TO PUNF;
172>    END PUNOUT;
173>
174>READIN: PROCEDURE BYTE;
175>    GO TO READF;
176>    END READIN;
177>
178>TRACK0: PROCEDURE;
179>    GO TO HOMF;
180>    END TRACK0;

```

CP/M VERSION
COPYRIGHT © 1976
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #

```

181>SELDISK: PROCEDURE(DISK);
182>    DECLARE DISK BYTE;
183>    GO TO SELF;
184>    END SELDISK;
185>
186>SELTRK: PROCEDURE(TRACK);
187>    DECLARE TRACK BYTE;
188>    GO TO TRKF;
189>    END SELTRK;
190>
191>SELSEC: PROCEDURE(SECTOR);
192>    DECLARE SECTOR BYTE;
193>    GO TO SECF;
194>
195>BUBBSSHPORTHNCBJUTQHORTPQHUSSSSH
CBSSHSSQCBSSSSNCBSSSSBYTE;
196>    GO TO DRDF;
197>    END READ$DISK;
198>
199>WRITE$DISK: PROCEDURE BYTE;
200>    GO TO DWRF;
201>    END WRITE$DISK;
202>
203> /* CONSOLE COMMUNICATIONN PROCEDURES */
204>
205>DECLARE
206>    /* SPECIAL CHARACTERS */
207>    ALT EQU '7DH';
208>    ESC EQU '16H';
209>    TAB EQU '08H';
210>    BEL EQU '07H';
211>    LF EQU '10';
212>    CR EQU '13';
213>
214>DECLARE COLUMN BYTE INITIAL(0); /* CURRENT CONSOLE COLUMN */
215>
216>
217>DECLARE IOSTATA ADDRESS INITIAL(3); /* IO STATUS BYTE LOCATION */
218>    IOSTAT BASED IOSTATA BYTE; /* VALUE OF STATUS BYTE */
219>    /* IOSTAT DEFINES THE CURRENT DEVICE ASSIGNMENT */
220>    0-1 CONSOLE
221>        0 TTY
222>        1 CRT
223>        2 BATCH (USE READER DEFINITION)
224>        3 USER (1)
225>    2-3 READER
226>        0 TTY
227>        1 PTR
228>        2 USER (1)
229>        3 USER (2)
230>    4-5 PUNCH
231>        0 TTY
232>        1 PTP
233>        2 USER (1)
234>        3 USER (2)
235>    6-7 LIST
236>        0 TTY
237>        1 CRT
238>        2 USER (1)
239>        3 USER (2)
240> */

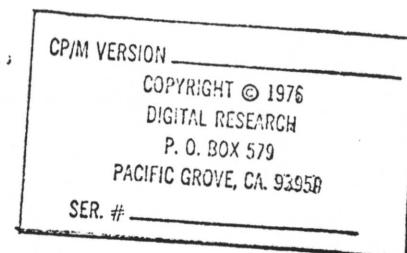
```

CP/M VERSION
COPYRIGHT © 1976
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```

241>241>TABOUT: PROCEDURE(CHAR);
242>    DECLARE (I,CHAR) BYTE;
243>    I = (CHAR = TAB AND (7 - (COLUMN AND 7)));
244>    IF CHAR = TAB THEN CHAR = ' ';
245>    DO WHILE (I != I - 1) <> 254;
246>    IF CHAR = CR THEN COLUMN = 6;
247>    IF CHAR >= ' ' THEN COLUMN=COLUMN+1;
248>    CALL CONOUT(CHAR);
249>    IF LISTCOPY THEN CALL LSTOUT(CHAR);
250>    END;
251>    END TABOUT;
252>
253>
254>CRLF: PROCEDURE;
255>    CALL TABOUT(CR);
256>    CALL TABOUT(LF);
257>    END CRLF;
258>
259>PPPRINT: PROCEDURE(A);
260>    DECLARE A ADDRESS, (I, M BASED A) BYTE;
261>    /* PRINT THE STRING STARTING AT ADDRESS A UNTIL THE NEXT
262>    OCCURRENCE OF A DOLLAR SIGN */
263>    DO WHILE (I!=M) <> '$'; CALL TABOUT(I);
264>    A = A + 1;
265>    END;
266>    END PRINT;
267>
268>READ: PROCEDURE;
269>    /* READ CHARACTERS FROM THE CONSOLE DEVICE
270>    INTO THE MEMORY LOCATION GIVEN BY 'INFO'
271>    UNTIL THE FIRST CARRIAGE RETURN
272>    IS ENCOUNTERED. ALLOW BACKSPACE (RUBOUT),
273>    LINE ELIMINATE (CTL U), AND SYSTEM RE-BOOT
274>    (CTL C) */
275>    DECLARE
276>        SLASH EQU '5CH',
277>        CTLU EQU '03H',
278>        CTLU EQU '15H',
279>        CTL EQU '5EH',
280>        CTL EQU '05H',
281>        CTLU EQU '10H',
282>        CTLU EQU '1AH',
283>        CTLU EQU '0CH';
284>
285>    /* THE INFO POINTER IS ASSUMED TO ADDRESS AN
286>    AREA OF MEMORY CONTAINING TWO BYTE QUANTITIES.
287>    THE FIRST GIVES THE MAXIMUM BUFFER LENGTH, AND
288>    THE SECOND IS SET TO THE NUMBER OF CHARACTERS
289>    SCANNED UPON RETURN */
290>
291>    DECLARE MAXL BASED INFO BYTE, /* MAX LENGTH */
292>        COMLEN BYTE, /* SCANNED LENGTH */
293>        BUFFER BASED INFO BYTE, /* BUFFER */
294>        C BYTE;
295>
296>    CTLOUT: PROCEDURE;
297>        /* PRINT UP-ARROW IN FRONT OF LAST CHARACTER READ */
298>        CALL TABOUT(CTL); CALL TABOUT(C OR 40H);
299>    END CTLOUT;

```



```

301>    COMLEN = 0;
302>    DO WHILE COMLEN < MAXL;
303>        /* MAKE ALPHABETICS UPPER CASE */
304>        IF (C := CONIN) = CTLU THEN
305>            DO; CALL CTLOUT; CALL CRLF;
306>            GO TO BOOT;
307>        END; ELSE
308>        IF C = CTLU THEN /* PHYSICAL RETURN */
309>            CALL CRLF; ELSE
310>            IF C = CTLU THENN LISTCOPY = NOT LISTCOPY; ELSE
311>            IF C = CR THEN
312>                DO; BUFFER(1) = COMLEN;
313>                CALL TABOUT(CR);
314>                RETURN;
315>            END; ELSE
316>            IF C = CTLU THEN
317>                DO; CALL CTLOUT; CALL CRLF; COMLEN=0;
318>            END; ELSE
319>            IF C = 7FH THENN /* RUBOUT */
320>                DO;
321>                    IF COMLEN > 0 THEN
322>                        CALL TABOUT(BUFFER((COMLEN-1)+2));
323>                    END; ELSE
324>                DO;
325>                    IF (C AND 01100000B) = 0 THEN /* CONTROL CHARACTER */
326>                        CALL CTLOUT; ELSE
327>                        IF C = TAB THEN CALL TABOUT(TAB); ELSE
328>                            CALL TABOUT(C);
329>                            BUFFER ((COMLEN-1)+1) = C;
330>                    END;
331>                END;
332>            END;
333>        END READ;
334>
335>    DECLARE MAXDSK EQU '1', /* MAX DISK NUMBER 0, 1, ... */
336>        HDISK EQU '2' /* NUMBER OF DISKS = MAXDSK+1 */;
337>
338>    DECLARE (DPTR,DCNT) BYTE,
339>        BUFFA ADDRESS INITIAL(80H),
340>        BUFF BASED BUFFA (120) BYTE;
341>
342>    DECLARE DMX EQU '63',
343>        /* DMX IS THE LAST DIRECTORY ENTRY NUMBER
344>        (LISTED AS 0, 1, ... , DMX) */
345>
346>    OFFSET EQU '2', /* NUMBER OF TRACKS USED BY BOOT */
347>
348>    AL1 EQU '000H', /* FIRST ALLOCATION
349>    VECTOR ELEMENT. EACH BIT THAT IS '1' RESERVES
350>    A 1 K BLOCK FOR THE DIRECTORY. EACH BLOCK IS
351>    8 RECORDS BY 128 BYTES PER RECORD (NOTE THAT
352>    RESERVATIONS START ON THE LEFT OF THE WORD) */
353>
354>
355>    ALLOC0 (32) BYTE, /* ALLOCATION VECTOR FOR DISK 0 */
356>    ALLOC1 (32) BYTE, /* ALLOCATIONN VECTOR FOR DISK 1 */
357>    ALLOCA ADDRESS, /* POINTER TO CURRENTLY REFERENCED ALLOC */
358>    ALLOC BASED ALLOCA (32) BYTE; /* ALLOC VECTOR TEMPLATE */
359>
360>

```

```

361> DECLARE
362>   EMP EQU '0E5H',
363>
364>   MRD EQU '1B', /* NUMBER OF READ RE-TRY'S */
365>
366>   FOREVER EQU 'WHILE TRUE',
367>   MAL EQU '242', /* LARGEST BLOCK NUMBER */
368>   MRC EQU '127', /* LARGEST RECORD NUMBER */
369>   DSF EQU '2', /* AMOUNT TO SHIFT 128 BYTE RECORD
370>                 TO GET A SINGLE DISK ENTRY */
371>   DMK EQU '11B', /* MASK CORRESPONDING TO DSF */
372>   FLN EQU '32',
373>   FSL EQU '5', /* AMOUNT TO SHIFT TO MULTIPLY
374>                 BY THE FCB LENGTH (FLN) */
375>   FDM EQU '16', /* BEGINNING OF DISK MAP */
376>   FRL EQU '32', /* LOCATION OF REC TO R/W */
377>   FRC EQU '15', /* LOCATION OF RECORD COUNT
378>                 (MUST BE ONE BELOW DISK MAP) */
379>   FRE EQU '12', /* POSITION OF REEL NUMBER */
380>   LFB EQU '31',
381>   FNM EQU '13'; /* LENGTH OF FILE NAME */

382>
383> DECLARE S BASED INFO (33) BYTE; /* FILE CONTROL BLOCK
384> PASSED TO THE DISK MONITOR FROM THE USER */
385>
386> /* THE FILE CONTROL BLOCK FORMAT IS SHOWN BELOW:
387>
388> -----
389> / 1 BY / 8 BY / 3 BY / 1 BY /2BY/1 BY/ 16 BY /
390> /FILETYPE/ NAME / EXT / REEL NO/KXXX/RCNT/DMO .. DM15/
391> -----
392>
393> FILETYPE : 0EH IF AVAILABLE (OTHERWISE UNDEFINED NOW)
394> NAME : 8 CHARACTER PRIMARY NAME
395> EXT : 3 CHARACTER EXTENT
396>          COM IMPLIES COMMAND TYPE
397>          (OTHERWISE UNDEFINED NOW).
398> REEL NO : 'REEL NUMBER' FIRST REEL IS 0, SECOND IS 1,
399>          AND SO FORTH UNTIL 255.
400>
401> XXX : UNUSED FOR NOW
402> RCNT : RECORD COUNT IN FILE (0 TO 127)
403> DMO ... : DISK ALLOCATION MAP. 255 IF NOT ALLOCATED.
404> DM15 : OTHERWISE IT POINTS TO ALLOCATED DISK BLOCK
405>
406> THE FILE CONTROL BLOCK IS FOLLOWED BY ONE BYTE OF
407> INFORMATION WHICH GIVES THE NEXT RECORD TO BE READ
408> OR WRITTEN IN AN OPENED FILE. THIS INFORMATION
409> IS NOT A PART OF THE DIRECTORY. EACH READ OR WRITE
410> WILL INCREMENT THIS RECORD COUNT.
411>
412> */
413>
414>
415> DECLARE
416>   GLDDSK BYTE, /* DISK ON ENTRY TO DOS */
417>   FCBDISK BYTE, /* DISK NAMED IN FCB */
418>   CURDSK BYTE INITIAL(0), /* CURRENTLY ADDRESSED DISK */
419>   DLOG BYTE INITIAL(0), /* BIT VECTOR GIVING LOGGED-IN DISKS */
420>   CURTRKV(HDISK) BYTE, /* TRACK VECTOR */
421>   CURRECV(HDISK) ADDRESS, /* RECORD VECTOR */
422>   CURTRKA ADDRESS, /* POINTS TO CURRENT TRACK NUMBER */
423>   CURRECA ADDRESS, /* POINTS TO CURRENT RECORD NUMBER */
424>   CUREC BASED CURRECA ADDRESS, /* CURRENTLY ADDRESSED RECORD */
425>   CUPTRK BASED CURTRKA BYTE, /* CURRENT TRACK 0-76 */
426>   RCOUNT BYTE, /* RECORD COUNT IN CURRENTLY
427>                 ADDRESSED FCB */
428>   VRECORD BYTE, /* CURRENT VIRTUAL RECORD */
429>   ARECORD ADDRESS, /* CURRENT ACTUAL RECORD */
430>
431> PDISK: PROCEDURE;
432>   CALL PRINTC('DISK $');
433>   CALL TABOUT('A'+CURDSK);
434>   END PDISK;
435>
436> NHOME: PROCEDURE;
437>   /* MOVE TO HOME POSITION, THEN OFFSET BY DOS TRACKS */
438>   CALL TRACK0; /* AT HOME POSITION */
439>   CALL SELTRK(OFFSET); /* SELECT FIRST DIRECTQRY POSITION */
440>   CURREC, CURTRK = 0;
441>   END HOME;
442>
443> SEEK: PROCEDURE;
444>   /* SEEK THE TRACK GIVEN BY ARECORD (ACTUAL RECORD) */
445>   DECLARE TRAN. DATA /* SECTOR NUMBER TRANSLATE TABLE */
446>   (01H,07H,0DH,13H, 19H,05H,0BH,11H, 17H,03H,09H,0FH,
447>    15H,02H,08H,0EH, 14H,1AH,06H,BCH, 12H,18H,04H,0RH,
448>    10H,16H);
449>
450>   DECLARE T ADDRESS;
451>   DO WHILE ARECORD < CURREC;
452>     CURREC = CURREC - 26;
453>     CURTRK = CURTRK - 1;
454>   END;
455>   DO WHILE ARECORD >= (T + CURREC + 26);
456>     CURREC = T;
457>     CURTRK = CURTRK + 1;
458>   END;
459>
460>   /* WE ARE NOW POSITIONED OVER THE TRACK CONTAINING THE ACTUAL
461> RECORD. THE SECTOR TO BE READ IS ARECORD - CURREC + 1. THE
462> TRACK NUMBER IS CURTRK */
463>   CALL SELTRK(CURTRK+OFFSET);
464>   CALL SELSEEK(TRAN(ARECORD - CURREC));
465>   END SEEK;
466>
467> WAITIO: PROCEDURE(READING);
468>   DECLARE READING BYTE; /* TRUE IF READING, FALSE IF WRITING */
469>   DECLARE COND BYTE; /* CONDITION UPON RETURN */
470>   DECLARE CTLC LITERALLY '03H';
471>   IF READING THEN COND = REHD$DISK; ELSE
472>     COND = WRITE$DISK;
473>   IF COND = 0 THEN RETURN; /* DISK I/O SUCCESSFUL */
474>
475>   /* ARRIVE HERE AFTER TOO MANY READ WRITE FAILURES */
476>   CALL CRLF; CALL PRINTC('PERM ERR $');
477>   CALL PDISK; IF CONIN = CTLC THEN GO TO BOOT;
478>   /* ENSURE NOT BATCH PROCESSING */
479>   ...

```

```

481> IF (IOSTAT AND 11B) > 1 THEN HALT;
482> CALL CRLF;
483> END WAITIO;
484>
485>RDBUFF. PROCEDURE;
486> /* START AN I/O AND WAIT FOR IO FINISH */
487> CALL WAITIO(TRUE);
488> END RDBUFF;
489>
490>WRBUFF. PROCEDURE;
491> /* WRITE THE BUFFER, SELECT NON-DELETED DATA */
492> CALL WAITIO(FALSE);
493> END WRBUFF;
494>
495>INDEX. PROCEDURE;
496> /* COMPUTE DISK BLOCK NUMBER FROM CURRENT
   FCB ADDRESSED BY INFO */
497>
498> ARECORD = S(FDM+SHR(VRECORD,3));
499> END INDEX;
500>
501>ATRAH. PROCEDURE;
502> /* COMPUTE ACTUAL TRACK ADDRESS (ASSUMES
   PREVIOUS CALL TO INDEX */
503> ARECORD = SHL(ARECORD,3) OR (VRECORD AND 111B);
504> END ATRAH;
505>
506>GETFCB. PROCEDURE;
507> /* SET VARIABLES FROM CURRENTLY ADDRESSED FCB */
508>
509> VRECORD = S(FRL);
510> RCOUNT = S(FRC);
511> END GETFCB;
512>
513>SETFCB. PROCEDURE;
514> /* PLACE VALUES BACK INTO CURRENTLY ADDRESSED
   FCB, AND INCREMENT THE RECORD COUNT */
515>
516>FCB. PROCEDURE;
517> /* SEEK THE RECORD CONTAINING THE CURRENT DIRECTORY ENTRY */
518> FCB = SHR(DCNT, DSF);
519>
520> S(FRL) = VRECORD + 1;
521> S(FRC) = RCOUNT;
522> END SETFCB;
523>
524>SEEK$DIR. PROCEDURE;
525> /* SEEK THE RECORD CONTAINING THE CURRENT DIRECTORY ENTRY */
526> ARECORD = SHR(DCNT, DSF);
527> CALL SEEK;
528> END SEEK$DIR;
529>
530>READ$DIR. PROCEDURE;
531> /* READ NEXT DIRECTORY ENTRY (SET DCNT=255 INITIALLY) */
532> IF (DCNT = DCNT+1) > DMX THEN
533>   DO; DCNT = 255; RETURN;
534> END;
535> IF (DPTR.=SHL(DCNT AND DMK, FSL)) = 0 THEN
536>   DO; CALL SEEK$DIR;
537>   CALL RDBUFF;
538> END;
539> END READ$DIR;
540>

```

CP/M VERSION
 SER. #
 COPYRIGHT © 1976
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA. 93950

```

541>GET$ALLOC. PROCEDURE(I) BYTE;
542> DECLARE I BYTE;
543> RETURN ALLOC(I);
544> END GET$ALLOC;
545>
546>PPUT$ALLOC. PROCEDURE(I,X);
547> DECLARE (I,X) BYTE;
548> ALLOC(I) = X;
549> END PPUT$ALLOC;
550>
551>
552>GET$ALLOC$BIT. PROCEDURE(I) BYTE;
553> /* RETURN THE I-TH BIT OF ALLOC */
554> DECLARE I BYTE;
555> RETURN ROL(ALLOC(SHR(I,3)), (I AND 111B)+1);
556> END GET$ALLOC$BIT;
557>
558>SET$ALLOC$BIT. PROCEDURE(I,B);
559> /* SET THE I-TH BIT OF ALLOC TO THE LSB OF B */
560> DECLARE (I,B) BYTE;
561> CALL PUT$ALLOC(SHR(I,3));
562> ROR((GET$ALLOC$BIT(I) AND 0FEH) OR B, (I AND 111B) + 1);
563> END SET$ALLOC$BIT;
564>
565>
566>GETBUFF. PROCEDURE(I) BYTE;
567> DECLARE I BYTE;
568> RETURNN BUFF(I);
569> END GETBUFF;
570>
571>PPUTBUFF. PROCEDURE(I,X);
572> DECLARE (I,X) BYTE;
573> BUFF(I) = X;
574> END PPUTBUFF;
575>
576>SCANDM. PROCEDURE(BIT);
577> DECLARE (BIT, I, K) BYTE;
578> /* SCANDM SCANS THE DISK MAP ADDRESSED BY DPTR FOR NON-ZERO ENTRIES
   -- THE ALLOCATION VECTOR ENTRY CORRESPONDING TO A NON-ZERO ENTRY
   IS SET TO THE VALUE OF 'BIT' */
579> DO I = DPTR+FIM TO DPTR+LFB;
580>   IF (K = GETBUFF(I)) <> 0 THEN
581>     CALL SET$ALLOC$BIT(K,BIT);
582>   END;
583> END;
584> END SCANDM;
585>
586>
587>INITIALIZE. PROCEDURE;
588> DECLARE I BYTE;
589> /* INITIALIZE THE DISK SYSTEM */
590> RET = FALSE; /* SET TO TRUE IF $ FILE EXISTS */
591>
592> ALLDC = AL1;
593> DO I=1 TO 31; CALL PUT$ALLOC(I,0);
594> END;
595> CALL HOME;
596> DCNT = 255;
597> DO FOREVER;
598> CALL READ$DIR;
599> IF DCNT = 255 THEN RETURN;
600> IF GETBUFF(DPTR) <> EMP THEN

```

CP/M VERSION

COPYRIGHT © 1976
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA. 93950

SER. #

```

601> DO; /* CHECK FOR $ FILE (IN CASE OF SUBMIT) */
602> RET = RET OR GETBUFF(DPTR+1) = '$';
603> /* SET ALLOC BIT TO 1 FOR EACH NON-ZERO DM ENTRY */
604> CALL SCANDM(1);
605> END;
606> END;
607> END INITIALIZE;
608>
609>
610>DECLARE SEARCHL BYTE; /* SEARCH LENGTH SET BY SEARCH */
611> SEARCHA ADDRESS; /* SEARCH ADDRESS SET BY SEARCH */
612>
613>SEARCHN: PROCEDURE;
614> /* SEARCH FOR THE NEXT DIRECTORY ELEMENT, ASSUMING A PREVIOUS */
615> CALL DN SEARCH WHICH SETS SEARCHA AND SEARCHL */;
616> DECLARE (I,C) BYTE;
617> INFO = SEARCHA;
618> DO FOREVER;
619> CALL READ$DIR;
620> IF (RET := DCNT) = 255 THEN RETURN;
621> I = B;
622> DO WHILE (I < SEARCHL) AND
623> /* MATCH OR QUESTION MARK */
624> ((C, = S(I)) = GETBUFF(DPTR+I) OR C = 63);
625> I = I + 1;
626> END;
627> IF I = SEARCHL THEN RETURN;
628> END;
629> END SEARCHN;
630>
631>SEARCH: PROCEDURE(XL);
632> DECLARE XL BYTE;
633> SEARCHL = XL;
634> SEARCHA = INFO;
635> DCNT = 255;
636> CALL HOME;
637> /* NOW READY TO READ THE DISK */
638> CALL SEARCHN;
639> END SEARCH;
640>
641>HDELETE: PROCEDURE;
642> DECLARE (I,J,K) BYTE;
643> /* SEARCH ONLY UP THROUGH THREE CHARACTER EXTENT */
644> CALL SEARCH(FRE);
645> DO FOREVER;
646> IF DCNT = 255 THEN /* NO MORE ENTRIES MATCH */ RETURN;
647> /* SET EACH NON-ZERO DISK MAP ENTRY TO 0, IN ALLOC VECTOR */
648> CALL SCANDM(0);
649> CALL PUTBUFF(DPTR,EMP);
650> /* ARECORD HAS BEEN PREVIOUSLY SOUGHT BY READDIR */
651> CALL WRBUFF;
652> CALL SEARCHN;
653> END;
654> END DELETE;
655>
656>GET$BLOCK: PROCEDURE(L) BYTE;
657> /* FIND A BLOCK WHICH IS AVAILABLE ON THE DISK AND IS CLOSEST
658> TO THE BLOCK 'L'. RETURN A B IF NO BLOCK IS AVAILABLE */
659> DECLARE (L, R) BYTE;
660> R = L;

```

CP/M VERSION

COPYRIGHT © 1976
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA. 93950
SER. #

```

661> DO WHILE (R < MAL) OR (L > 0);
662> L = L - (1 AND L > 0);
663> R = R + (1 AND R < MAL);
664> IF NOT GET$ALLOC$BIT(R) THEN RETURN R;
665> IF NOT GET$ALLOC$BIT(L) THEN RETURN L;
666> END;
667> RETURN 0;
668> END GET$BLOCK;
669>
670>COPY$DIR: PROCEDURE(B,L);
671> DECLARE (B,L) BYTE;
672> /* COPY FCB INFORMATION STARTING AT BYTE B FOR L BYTES INTO
673> BEGINNING OF CURRENTLY ADDRESSED DIRECTORY ENTRY */
674> DO WHILE (L,=L-1) <> 255;
675> CALL PUTBUFF(L+DPTR,S(B+L));
676> END;
677> CALL SEEK$DIR;
678> CALL WRBUFF;
679> END COPY$DIR;
680>
681>COPY$FCB: PROCEDURE;
682> /* COPY THE ENTIRE FILE CONTROL BLOCK */
683> CALL COPY$DIR(0,FRL);
684> END COPY$FCB;
685>
686>RENAME: PROCEDURE;
687> /* RENAME THE FILE DESCRIBED BY THE FIRST HALF OF THE CURRENTLY
688> ADDRESSED FILE CONTROL BLOCK. THE NEW NAME IS CONTAINED IN THE
689> LAST HALF OF THE CURRENTLY ADDRESSED FILE CONTROL BLOCK. THE
690> FILE TYPE, FILE NAME, AND FILE EXT ARE CHANGED, BUT THE REEL
691> NUMBER FIELD IS IGNORED */
692>
693> /* SEARCH UP TO THE REEL NUMBER FIELD */
694> CALL SEARCH(FRN); S(16)=S; ← S(4)=S;
695> DO WHILE DCNT <> 255; CALL COPY$DIR(FDN,FRE);
696> CALL SEARCHN;
697> END;
698> END RENAME;
699>
700>OPEN: PROCEDURE;
701> DECLARE I BYTE;
702> /* SEARCH FOR DIRECTORY ENTRY, COPY TO FCB */
703> CALL SEARCH(FNM);
704> IF DCNT <> 255 THEN
705> DO I=FNM TO LFB;
706> S(I) = GETBUFF(DPTR+I);
707> END;
708> END OPEN;
709>
710>CLOSE: PROCEDURE;
711> /* LOCATE THE DIRECTORY ELEMENT AND RE-WRITE */
712> CALL SEARCH(FNM);
713> IF DCNT <> 255 THEN
714> CALL COPY$FCB;
715> END CLOSE;
716>
717>MAKE: PROCEDURE;
718> /* CREATE A NEW FILE; FIRST CREATE ENTRY IN
719> THE DIRECTORY. FILE IS OPENED UPON RETURN */
720> DECLARE I BYTE;

```

CP/M VERSION

COPYRIGHT © 1976
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CAL 93950
SER. #

```

721> FCB ADDRESS;
722>
723> FCB = INFO; INFO = .EMP,
724> /* LOOK FOR AN EMPTY DIRECTORY ENTRY */
725> CALL SEARCH(1);
726> IF DCNT <> 255 THEN
727>   DO; /* SET ELEMENTS TO ZERO */
728>     INFO = FCB;
729>     DO I=FNM TO LFB;
730>       S(I) = 0;
731>     END;
732>   /* COPY INTO DIRECTORY ENTRY */
733>   CALL COPY$FCB;
734>   END;
735> END MAKE;
736>
737>OPEN$REEL: PROCEDURE(READING);
738>   DECLARE READING BYTE;
739>   /* CLOSE CURRENT REEL AND OPEN THE NEXT ONE, IF POSSIBLE
    READING IS TRUE IF WE ARE IN READ MODE */
740>   CALL CLOSE;
741>   /* RET REMAINS AT 255 IF WE CANNOT OPEN THE NEXT REEL */
742>   IF DCNT = 255 THEN RETURN;
743>   /* INCREMENT THE REEL NUMBER */
744>   S(FRE) = S(FRE) + 1;
745>   CALL SEARCH(FNM);
746>   IF DCNT = 255 THEN
747>     DO; IF READING THEN RETURN;
748>     CALL MAKE;
749>   END; ELSE
750>   CALL OPEN;
751>   IF DCNT = 255 THEN
752>     DO; RET = 1; /* END OF FILE IN DISK READ */
753>     RETURN;
754>   END;
755>   CALL GETFCB;
756>   RET = 0;
757> END OPEN$REEL;
758>
759>HDISKREAD: PROCEDURE;
760>   CALL GETFCB;
761>
762>   IF RCOUNT <= VRECORD THEN
763>     DO; RET = 1;
764>     IF VRECORD = 128 THENN CALL OPEN$REEL(TRUE);
765>     VRECORD = 0;
766>     IF RET <> 0 THEN RETURN;
767>   END;
768>   DO; CALL INDEX;
769>
770>   /* ERROR 2 IF READING UNWRITTEN DATA */
771>   IF LOW(ARECORD) = 0 THEN RET = 1; ELSE
772>     DO; CALL ATRAN;
773>     /* ARECORD IS NOW ACTUAL DISK ADDRESS */
774>     CALL SEEK;
775>     /* NOW READ THE BUFFER */
776>     CALL RDBUFF;
777>     CALL SETFCB;
778>   END;
779>
780> END;

```

CP/M VERSION
 COPYRIGHT © 1976
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA. 93950
 SER. #

```

781> END DISKREAD;
782>
783>DDISKURITE: PROCEDURE;
784>   DECLARE (I,L) BYTE;
785>   CALL GETFCB;
786>
787>   IF VRECORD > MRC THEN /* PAST EOF, NEXT REEL NOT OPENED */
788>     RET = 1; ELSE
789>     DO; CALL INDEX;
790>     IF LOW(ARECORD) = 0 THEN /* NOT ALLOCATED */
791>       DO; /* THE ARGUMENT TO GET$BLOCK IS THE STARTING POSITION
        FOR THE DISK SEARCH - THIS SHOULD BE THE LAST ALLOCATED
        BLOCK FOR THIS FILE, OR THE VALUE 0 IF NO SPACE HAS BEEN
        ALLOCATED TO THIS FILE */
792>       I = 0;
793>     IF (L := FDM + SHR(VRECORD,3)) > FDM THEN
794>       /* THERE IS A PREVIOUS BLOCK ALLOCATED */ I = S(L-1);
795>     IF (I := GET$BLOCK(I)) = 0 THEN /* NO MORE SPACE */
796>       RET = 2; ELSE
797>       DO; CALL SET$ALLOC$BIT(I,1);
798>       /* BLOCK IS ALLOCATED */
799>       ARECORD, S(L) = I;
800>     END;
801>   /* CONTINUE IF NO ERROR IN ALLOCATION */
802>   IF RET = 0 THEN
803>     DO; CALL ATRAN;
804>     CALL SEEK;
805>     CALL WRBUFF;
806>     IF RCOUNT <= VRECORD THEN RCOUNT = VRECORD+1;
807>     /* CHECK FOR END-OF-REEL, IF FOUND ATTEMPT TO OPEN
        NEXT REEL IN PREPARATION FOR THE NEXT WRITE */
808>     IF VRECORD = MRC THEN
809>       DO;
810>         /* UPDATE CURRENT FCB BEFORE GOING TO THE NEXT REEL */
811>         CALL SETFCB; CALL OPEN$REEL(FALSE);
812>         /* VRECORD REMAINS AT MRC CAUSING END-OF-FILE
            IF NO MORE DIRECTORY SPACE IS AVAILABLE */
813>         IF RET = 0 THEN VRECORD = 255; /* GOES TO ZERO */
814>       RET = 0;
815>     END;
816>   CALL SETFCB;
817>   /* VRECORD REMAINS AT MRC CAUSING END-OF-FILE
      IF NO MORE DIRECTORY SPACE IS AVAILABLE */
818>   IF RET = 0 THEN VRECORD = 255; /* GOES TO ZERO */
819>   RET = 0;
820>   END;
821>   CALL SETFCB;
822>   END;
823> END;
824> END;
825> END DISKWRITE;
826>
827>SELECT: PROCEDURE;
828>   /* SELECT DISK 'INFO' FOR SUBSEQUENT
      INPUT OR OUTPUT OPERATIONS */
829>
830>
831>   IF CURDSK > MAXDSK THEN /* SELECTIONN ERROR */
832>     DO; CALL CRLF; CALL PRINT(. 'SELECT ERROR $');
833>     CALL PDISK; CALL CRLF; GO TO BOOT;
834>   END;
835>   ALLOCA = .ALLOC0(SHL(CURDSK,5));
836>   /* NOTE THAT THIS ASSUMES THERE ARE NO MORE
      THAN 8 DISKS ON THE SYSTEM - OTHERWISE
      REPLACE BY .ALLOC0(SHL(DOUBLE(CURDSK),5)) */
837>   CURTRKA = .CURTRKV(CURDSK);
838>
839>
840>

```

```

841> CURRECA = .CURRECV(CURDSK);
842> /* SET CONTROLLER */
843> CALL SELDISK(CURDSK);
844>
845> /* CHECK TO INSURE THAT DISK IS LOGGED IN */
846> IF NOT ROR(ROL(DLOG,1),CURDSK+1) THEN
847>   DO;
848>     DLOG = DLOG OR ROR(ROL(1,CURDSK+1),1);
849>     CALL INITIALIZE;
850>   END;
851> END SELECT;
852>
853> CURSELECT: PROCEDURE;
854>   IF LINFO <> CURDSK THEN
855>     DO; CURDSK = LINFO; CALL SELECT;
856>   END;
857> END CURSELECT;
858>
859> RESELECT: PROCEDURE;
860>   /* CHECK CURRENT FCB TO SEE IF RESELECTION NECESSARY */
861>   IF (LINFO := (S AND 1$1111B) - 1) < 30 THEN
862>     DO; OLDDSK = CURDSK; FCBDISK = S; S = S AND 1$110$0000B;
863>     CALL CURSELECT;
864>   END;
865> END RESELECT;
866>
867> SETDMA: PROCEDURE(A);
868>   DECLARE A ADDRESS;
869>   CALL SELDMA(BUFFA,A);
870> END SETDMA;
871>
872> /* ARRIVE HERE UPON ENTRY TO THE DISK MONITOR */
873> SAVE THE STACKPOINTER, PERFORM THE DESIRED FUNCTION,
874> RESTORE THE STACKPOINTER, AND RETURN TO THE CALLING
875> PROGRAM. */
876>
877> DECLARE STACK (16) ADDRESS,
878>   OLDSP ADDRESS;
879>
880> OLDSP = STACKPTR;
881> STACKPTR = .STACK(LENGTH(STACK));
882> /* CALLING PROGRAM'S STACK TOP ADDRESS NOW SAVED */
883>
884> LINFO = LOW(LINFO);
885> ARET, RET = 0;
886> FCBDISK = 0;
887>
888> DO CASE FUNC;
889>   /* 0: SYSTEM RE-BOOT */
890>   GO TO BOOT;
891>
892>   /* 1: READ CONSOLE */
893>   DO; /* READ CHARACTER, TEST FOR GRAPHICS */
894>     IF ((RET := COHIN) >= ' ') OR
895>       (RET = CR) OR (RET = LF) OR (RET = TAB) THEN
896>       CALL TABOUT(RET);
897>
898>   /* 2: WRITE CONSOLE */
899>   CALL TABOUT(LINFO);
900>
901>   /* 3: READ READER DEVICE */
902>   RET = READIN;

```

CP/M VERSION
 COPYRIGHT © 1976
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA. 93950
 SER. # _____

```

901> /* 4: WRITE PUNCH DEVICE */
902> CALL PUNOUT(LINFO);
903> /* 5: WRITE LIST DEVICE */
904> CALL LSTOUT(LINFO);
905> /* 6: INTERROGATE MEMORY SIZE */
906> ARET = FDOS;
907> /* 7: INTERROGATE DEVICE STATUS */
908> ARET = IOSTAT;
909> IOSTAT = INFO;
910> /* 8: CHANGE DEVICE STATUS */
911> CALL PRINT(INFO);
912> /* 9: PRINT BUFFER AT THE CONSOLE */
913> CALL READ;
914> /* 10: READ BUFFER FROM THE CONSOLE */
915> CALL READ;
916> /* 11: CHECK FOR CONSOLE INPUT READY */
917> RET = CONBRK;
918> /* 12: */
919> /* 13: RESET DISK SYSTEM, INITIALIZE TO DISK 0 */
920> DO; CURDSK.DLOG = 0;
921> CALL SETDMA(B0H);
922> CALL SELECT;
923> CHAR$RDY,LISTCOPY, = FALSE;
924> END;
925> /* 14: SELECT DISK 'INFO' */
926> CALL CURSELECT;
927> /* 15: OPEN */
928> DO; CALL RESELECT;
929> CALL OPEN;
930> END;
931> /* 16: CLOSE */
932> DO; CALL RESELECT;
933> CALL CLOSE;
934> END;
935> /* 17: SEARCH FOR FIRST OCCURRENCE OF A FILE */
936> DO; CALL RESELECT;
937> CALL SEARCH(FNM);
938> END;
939> /* 18: SEARCH FOR NEXT OCCURRENCE OF A FILE,NAME */
940> DO; INFO = SEARCHA; CALL RESELECT;
941> CALL SEARCHN;
942> END;
943> /* 19: DELETE A FILE */
944> DO; CALL RESELECT;
945> CALL DELETE;
946> END;
947> /* 20: READ A FILE */
948> DO; CALL RESELECT;
949> CALL DISKREAD;
950> END;
951> /* 21: WRITE A FILE */
952> DO; CALL RESELECT;
953> CALL DISKWRITE;
954> END;
955> /* 22: CREATE A FILE */
956> DO; CALL RESELECT;
957> CALL MAKE;
958> END;
959> /* 23: RENAME A FILE */
960> DO; CALL RESELECT;

```

CP/M VERSION
 COPYRIGHT © 1976
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA. 93950
 SER. # _____

```

961>     CALL RENAME;
962>     END;
963> /* 24. RETURN THE LOGIN VECTOR */
964> RET = DLGIC;
965> /* 25. RETURN SELECTED DISK NUMBER */
966> RET = CURDSK;
967> /* 26. SET THE SUBSEQUENT DMA ADDRESS TO INFO */
968> CALL SETDMAINFO;
969> /* 27. RETURN THE LOGIN VECTOR ADDRESS */
970> APET = ALLOCATE;
971> END; /* OF CASES */
972>
973>
974>
975> GOBACK;
976> IF FCBDISK <> 0 THEN /* RESTORE DISK NUMBER */
977> DO; S = FCBDISK; LINFO = DLDDISK; CALL CURSELECT;
978> END;
979> /* RESTORE THE USER'S STACK AREA */
980> STACKPTR = OLDSPP;
981>
982> /* RETURN A SINGLE OR DOUBLE BYTE VALUE */
983> RETURNH ARET OR RET;
984> END DISKMON;
985> /* <><><><><><><><><> END OF DISK MONITOR <><><> */
986>
987>
988> //DECLARE TEMPDATA(10) BYTE; /* TEN LOCATIONS IN RAM FOR THE INTERFACE*/
989> EOF
990> EOF.
1>
2>      5 MEMORY 03E60H
3>      25 DISKMON 03206H
4>      26 FUNC 03D33H
5>      27 INFO 03D34H
6>      38 COPYRIGHT 03211H
7>      32 LINFO 03D37H
8>      33 HRET 03D38H
9>      34 RET 03D39H
10>     35 CHAPRDY 03D3CH
11>     37 K2CHAR 03D3DH
12>     38 LISTCOPY 03D3EH
13>     39 CONFDY 03D25H
14>     42 CONINH 03235H
15>     46 CONPRK 03244H
16>     56 CONCHAR 0327DH
17>     57 CHAR 0323FH
18>     60 CONOUT 03285H
19>     61 CHAR 03D40H
20>     64 LISTOUT 03298H
21>     65 CHAR 03D41H
22>     68 POUTOUT 03240H
23>     69 CHAP 03D42H
24>     72 READIN 03248H
25>     75 TRACKR 0324CH
26>     78 SELDISK 03260H
27>     79 DISK 03243H
28>     82 SELTRK 03288H
29>     83 TRACK 03D44H
30>     86 SELSEC 032C0H

```

CP/M VERSION
COPYRIGHT © 1976
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. #

```

    31>     87 SECTOR 03D45H
    32>     90 SELDMA 03200H
    33>     91 DMA 03D46H
    34>     94 READDISK 032D2H
    35>     97 WRITEDISK 032D6H
    36>     100 COLUMN 03D45H
    37>     101 IOSTATA 03D4AH
    38>     103 TABOUT 032DAH
    39>     104 CMAR 03D4CH
    40>     106 I 03D4DH
    41>     118 CRLF 03335H
    42>     121 PRINT 03340H
    43>     122 A 03D4EH
    44>     124 I 03D51H
    45>     129 READ 03362H
    46>     132 COMLEN 03D52H
    47>     134 C 03D53H
    48>     135 CTLOUT 03365H
    49>     164 DPTR 03D54H
    50>     165 DCNT 03D55H
    51>     166 BUFFA 03D56H
    52>     169 ALLOCB 03D56H
    53>     170 ALLOC1 03D178H
    54>     171 ALLOCA 03D98H
    55>     174 DLDDSK 03D9AH
    56>     175 FCBDISK 03D98H
    57>     176 CURDSK 03D9CH
    58>     177 DLGIC 03D9DH
    59>     178 CURTRKV 03D2EH
    60>     179 CURRECV 03D4DH
    61>     188 CURTRKA 03D44H
    62>     181 CURRECA 03D46H
    63>     184 RCOUNT 03D48H
    64>     185 VRECORD 03D49H
    65>     186 ARECORD 03D4AH
    66>     187 PDISK 0343AH
    67>     192 HOME 03454H
    68>     194 SEEK 03455H
    69>     197 I.RAN 03462H
    70>     213 T 03D4CH
    71>     218 WAITIO 03568H
    72>     219 READING 03D4EH
    73>     221 COND 03D4FH
    74>     229 RDBUFF 03560H
    75>     231 WRBUFF 03566H
    76>     233 INDEX 0356CH
    77>     235 ATRAHN 03580H
    78>     237 GETFCB 035A0H
    79>     240 SETFCB 035C1H
    80>     242 SEEKDIR 035BFH
    81>     244 READDIR 035F1H
    82>     250 GETALLOC 03614H
    83>     251 I 03DB0H
    84>     253 PUTALLOC 03620H
    85>     254 I 031B1H
    86>     255 X 03DB2H
    87>     257 GETALLOCBIT 03636H
    88>     258 I 03DB3H
    89>     260 SETALLOCBIT 03659H
    90>     261 I 03DB4H

```

CP/M VERSION

COPYRIGHT © 1976
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. #

```
21> 262 B 03DB5H  
22> 264 GETBUFF 03606H  
23> 265 I 03DE6H  
24> 267 PUTBUFF 03692H  
25> 268 I 03DB7H  
26> 269 X 03DE8H  
27> 271 SCHIDM 036A8H  
28> 272 BIT 03DB9H  
29> 274 I 03DBAH  
30> 275 K 03DBBH  
31> 290 INITIALIZE 036DBH  
32> 292 I 03IBCH  
33> 291 SEARCHL 03D9DH  
34> 292 SEARCHA 03DBEH  
35> 293 SEARCHN 03736H  
36> 295 I 03DC0H  
37> 296 C 03DC1H  
38> 303 SEARCH 0379BH  
39> 304 XL 03DC2H  
40> 306 DELETE 037B7H  
41> 309 I 03DC3H  
42> 310 J 03DC4H  
43> 313 K 03DC5H  
44> 314 GETBLOCK 037DBH  
45> 315 L 03DC6H  
46> 317 R 03IC7H  
47> 323 COPYDIR 03925H  
48> 324 B 03DC8H  
49> 325 L 03DC9H  
50> 329 COPYFCB 03955H  
51> 331 RENAME 0395EH  
52> 335 OPEN 03679H  
53> 337 I 03DCAH  
54> 341 CLOSE 03882H  
55> 344 MAYE 038C1H  
56> 346 I 03DCBH  
57> 347 FCB 03DCCH  
58> 352 OPENPEEL 03910H  
59> 353 PENDING 03DCFH  
60> 360 DISKREAD 03963H  
61> 367 DISKWRITE 039A7H  
62> 369 I 07106H  
63> 370 L 03DD1H  
64> 381 SELECT 03A5DH  
65> 387 CURSELECT 03ADEF  
66> 390 RESELECT 03AF2H  
67> 395 SETIMA 03818H  
68> 396 A 03DD2H  
69> 398 STACK 03DD4H  
70> 399 OLDSP 03DF4H  
71> 432 GOBACK 03CF1H  
72> 434 TEMPDATA 03DF6H
```

CP/M VERSION

COPYRIGHT © 1976
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA. 93950

SER. #