

# DIGITAL RESEARCH

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

PERIPHERAL INTERCHANGE PROGRAM (PIP)

CP/M VERSION \_\_\_\_\_

COPYRIGHT © 1976

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA. 93950

SER. # \_\_\_\_\_

```

1  PIPMOD.
DO;
  /* PERIPHERAL INTERCHANGE PROGRAM

      COPYRIGHT (C) DIGITAL RESEARCH
      NOVEMBER, 1976

2  1  DECLARE COPYRIGHT(*) BYTE DATA (
      ' COPYRIGHT (C) 1976, DIGITAL RESEARCH. PIP VERS 1.5'),

3  1  DECLARE INPLOC ADDRESS DATA (103H); /* ADDRESS OF INP. DEVICE */
4  1  DECLARE OUTLOC ADDRESS DATA (106H); /* ADDRESS OF OUT. DEVICE */

5  1  OUT. PROCEDURE(B);
6  2  DECLARE B BYTE;
      /* SEND B TO OUT. DEVICE */
7  2  CALL OUTLOC;
8  2  END OUT;

9  1  INP. PROCEDURE BYTE;
10 2  CALL INPLOC;
      /* PATCHED TO RETURN REG-A */
11 2  RETURN 0;
12 2  END INP;

13 1  TIMEOUT. PROCEDURE;
      /* WAIT FOR 50 MSEC */
14 2  CALL TIME(250); CALL TIME(250);
16 2  END TIMEOUT;

/* LITERAL DECLARATIONS */
17 1  DECLARE
      LIT LITERALLY 'LITERALLY',
      ENDFILE LIT '1AH',
      TAB LIT '9H',
      LA LIT '5FH',
      LB LIT '5BH', /* LEFT BRACKET */
      RB LIT '5DH', /* RIGHT BRACKET */
      XOFF LIT '13H', /* TRANSMIT BUFFER FUNCTION */

      RDR LIT '5',
      LST LIT '10',
      PUNP LIT '15',
      CONP LIT '19',
      NULP LIT '19',
      EOFP LIT '20',
      HSRDR LIT 'RDR',
      PRNT LIT '10',

      FSIZE LIT '33',
      NSIZE LIT '8',
      FNSIZE LIT '11',
      MDISK LIT '1',
      FNAME LIT '8',
      FEXT LIT '9',
      FEXTL LIT '3',
      FREEL LIT '12', /* REEL NUMBER FIELD OF FCB */

      /* POSITION OF 'PUN' + 1 */
      /* CONSOLE */
      /* NUL, BEFORE INCREMENT */
      /* EOF, BEFORE INCREMENT */
      /* READER DEVICES */
      /* PRINTER */

```

CPM VERSION 1.3  
 COPYRIGHT © 1976  
 DIGITAL RESEARCH  
 P. O. BOX 579  
 PACIFIC GROVE, CA. 93950  
 SER. # PIP 1.5

```

HBUFS LIT '80', /* "HEX" BUFFER SIZE */

ERR LIT '0',
SPECL LIT '1',
FILE LIT '2',
PERIPH LIT '3',
DISKNAME LIT '4',

18 1  DECLARE
      EXIT ADDRESS INITIAL(0H), /* REBOOT ADDRESS UPON COMPLETION */
      COLUMN BYTE, /* COLUMN COUNT FOR PRINTER TABS */
      AMBIG BYTE, /* SET FOR AMBIGUOUS FILE REFS */
      PARSET BYTE, /* TRUE IF PARAMETERS PRESENT */
      FEEDBASE BYTE, /* USED TO FEED SEARCH CHARACTERS */
      FEEDLEN BYTE, /* LENGTH OF FEED STRING */
      MATCHLEN BYTE, /* USED IN MATCHING STRINGS */
      QUITLEN BYTE, /* USED TO TERMINATE QUIT COMMAND */
      NBUF BYTE, /* NUM BUFFERS-1 IN SBUFF AND IBUFF */
      CDISK BYTE, /* CURRENT DISK */
      BUFFER (128) BYTE AT (80H), /* DEFAULT BUFFER */
      SEARCHCB (FSIZE) BYTE AT (5CH), /* SEARCH FCB IN MULTI COPY */
      MEMSIZE ADDRESS AT (6H), /* MEMORY SIZE */
      SBLEN ADDRESS, /* SOURCE BUFFER LENGTH */
      DBLEN ADDRESS, /* DEST BUFFER LENGTH */
      SBASE ADDRESS, /* SOURCE BUFFER BASE */
      /* THE VECTORS DBUFF AND SBUFF ARE DECLARED WITH DIMENSION
      1024, BUT ACTUALLY VARY WITH THE FREE MEMORY SIZE */
      DBUFF(1024) BYTE AT (.MEMORY), /* DESTINATION BUFFER */
      SBUFF BASED SBASE (1024) BYTE, /* SOURCE BUFFER */
      SDISK BYTE, /* SOURCE DISK */
      (SCOM, DHEX) BYTE, /* SOURCE IS 'COM' FILE IF TRUE */
                          /* DEST IS 'HEX' FILE IF TRUE */
      SOURCE (FSIZE) BYTE, /* SOURCE FCB */
      DEST (FSIZE) BYTE, /* DESTINATION FCB */
      DDISK BYTE, /* DESTINATION DISK */
      HBUFF(HBUFS) BYTE, /* HEX FILE BUFFER */
      HSOURCE BYTE, /* NEXT HEX SOURCE CHARACTER */

      NSOURCE ADDRESS, /* NEXT SOURCE CHARACTER */
      HARDEOF ADDRESS, /* SET TO NSOURCE ON REAL EOF */
      NDEST ADDRESS, /* NEXT DESTINATION CHARACTER */

19 1  DECLARE
      PDEST BYTE, /* DESTINATION DEVICE */
      PSOURCE BYTE, /* CURRENT SOURCE DEVICE */

20 1  DECLARE
      MULTCOM BYTE, /* FALSE IF PROCESSING ONE LINE */
      PUTNUM BYTE, /* SET WHEN READY FOR NEXT LINE NUM */
      CONCNT BYTE, /* COUNTER FOR CONSOLE READY CHECK */
      CHAR BYTE, /* LAST CHARACTER SCANNED */
      TYPE BYTE, /* TYPE OF CHARACTER SCANNED */
      FLEN BYTE, /* FILE NAME LENGTH */

      #INCLUDE(F1, CPIO, PLB)
21 1  = MON1. PROCEDURE(F,A);
22 2  = DECLARE F BYTE,
      = A ADDRESS;
23 2  = L1, GO TO L1; /* PATCHED WITH JMP 0005 */

```

```

24 2 = END MON1;
25 1 = MON2: PROCEDURE(F,A) BYTE;
26 2 = DECLARE F BYTE;
27 2 = A ADDRESS;
28 2 = L2, GO TO L2; /* PATCHED WITH JMP 0005 */
29 2 = RETURN 0;
30 1 = END MON2;
31 1 = READRDR: PROCEDURE BYTE;
32 2 = /* READ CURRENT READER DEVICE */
33 2 = RETURN MON2(3,0);
34 2 = END READRDR;
35 1 = READCHR: PROCEDURE BYTE;
36 2 = /* READ CONSOLE CHARACTER */
37 2 = RETURN MON2(1,0);
38 2 = END READCHR;
39 1 = DECLARE
40 2 = TRUE LITERALLY '1',
41 2 = FALSE LITERALLY '0',
42 2 = FOREVER LITERALLY 'WHILE TRUE',
43 2 = CR LITERALLY '13',
44 2 = LF LITERALLY '10',
45 2 = WHAT LITERALLY '63';
46 1 = PRINTCHR: PROCEDURE(CHAR);
47 2 = DECLARE CHAR BYTE;
48 2 = CALL MON1(2,CHAR);
49 2 = END PRINTCHR;
50 1 = CRLF: PROCEDURE;
51 2 = CALL PRINTCHR(CR);
52 2 = CALL PRINTCHR(LF);
53 2 = END CRLF;
54 1 = PRINT: PROCEDURE(A);
55 2 = DECLARE A ADDRESS;
56 2 = /* PRINT THE STRING STARTING AT ADDRESS A UNTIL THE
57 2 = NEXT DOLLAR SIGN IS ENCOUNTERED */
58 2 = CALL CRLF;
59 2 = CALL MON1(9,A);
60 2 = END PRINT;
61 1 = DECLARE DCNT BYTE;
62 1 = INITIALIZE: PROCEDURE;
63 2 = CALL MON1(13,0);
64 2 = END INITIALIZE;
65 1 = SELECT: PROCEDURE(D);
66 2 = DECLARE D BYTE;
67 2 = CALL MON1(14,D);
68 2 = END SELECT;
69 1 = OPEN: PROCEDURE(FCB);
70 2 = DECLARE FCB ADDRESS;
71 2 = DCNT = MON2(15,FCB);
72 2 = END OPEN;

```

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

```

62 1 = CLOSE: PROCEDURE(FCB);
63 2 = DECLARE FCB ADDRESS;
64 2 = DCNT = MON2(16,FCB);
65 2 = END CLOSE;
66 1 = SEARCH: PROCEDURE(FCB);
67 2 = DECLARE FCB ADDRESS;
68 2 = DCNT = MON2(17,FCB);
69 2 = END SEARCH;
70 1 = SEARCHN: PROCEDURE;
71 2 = DCNT = MON2(18,0);
72 2 = END SEARCHN;
73 1 = DELETE: PROCEDURE(FCB);
74 2 = DECLARE FCB ADDRESS;
75 2 = CALL MON1(19,FCB);
76 2 = END DELETE;
77 1 = DISKREAD: PROCEDURE(FCB) BYTE;
78 2 = DECLARE FCB ADDRESS;
79 2 = RETURN MON2(20,FCB);
80 2 = END DISKREAD;
81 1 = DISKWRITE: PROCEDURE(FCB) BYTE;
82 2 = DECLARE FCB ADDRESS;
83 2 = RETURN MON2(21,FCB);
84 2 = END DISKWRITE;
85 1 = MAKE: PROCEDURE(FCB);
86 2 = DECLARE FCB ADDRESS;
87 2 = DCNT = MON2(22,FCB);
88 2 = END MAKE;
89 1 = RENAME: PROCEDURE(FCB);
90 2 = DECLARE FCB ADDRESS;
91 2 = CALL MON1(23,FCB);
92 2 = END RENAME;
93 1 = DECLARE CBUFF(130) BYTE; /* COMMAND BUFFER */
94 2 = MAXLEN BYTE AT (.CBUFF(0)); /* MAX BUFFER LENGTH */
95 2 = COMLEN BYTE AT (.CBUFF(1)); /* CURRENT LENGTH */
96 2 = COMBUFF (128) BYTE AT (.CBUFF(2)); /* COMMAND BUFFER CONTENTS */
97 2 = DECLARE (TCBP,CBP) BYTE; /* TEMP CBP, COMMAND BUFFER POINTER */
98 1 = READCOM: PROCEDURE;
99 2 = /* READ INTO COMMAND BUFFER */
100 2 = MAXLEN = 128;
101 2 = CALL MON1(10,MAXLEN);
102 2 = END READCOM;
103 1 = DECLARE IOBYTE BYTE AT(3H); /* INTEL IOBYTE */

```

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

```

104 1 DECLARE /* CONTROL TOGGLE VECTOR */
      CONT(26) BYTE, /* ONE FOR EACH ALPHABETIC */
      /* 00 01 02 03 04 05 06 07 08 09 10 11 12 13
         A B C D E F G H I J K L M N
         14 15 16 17 18 19 20 21 22 23 24 25
         O P Q R S T U V W X Y Z */
      BLOCK BYTE AT(.CONT(1)), /* BLOCK MODE TRANSFER */
      DELET BYTE AT(.CONT(3)), /* DELETE CHARACTERS */
      ECHO BYTE AT(.CONT(4)), /* ECHO CONSOLE CHARACTERS */
      HEXT BYTE AT(.CONT(7)), /* HEX FILE TRANSFER */
      IGNOR BYTE AT(.CONT(8)), /* IGNORE .00 RECORD ON FILE */
      LOWER BYTE AT(.CONT(11)), /* TRANSLATE TO LOWER CASE */
      NUMB BYTE AT(.CONT(13)), /* NUMBER OUTPUT LINES */
      OBJ BYTE AT(.CONT(14)), /* OBJECT FILE TRANSFER */
      QUITB BYTE AT(.CONT(16)), /* QUIT COPY */
      STARTS BYTE AT(.CONT(18)), /* START COPY */
      TABS BYTE AT(.CONT(19)), /* TAB SET */
      UPPER BYTE AT(.CONT(20)), /* UPPER CASE TRANSLATE */
      VERIF BYTE AT(.CONT(21)), /* VERIFY EQUAL FILES ONLY */
      ZEROP BYTE AT(.CONT(25)), /* ZERO PARITY ON INPUT */

105 1 LIFHEAD: PROCEDURE;
106 2 CALL MONI(12,0);
107 2 END LIFHEAD;

108 1 SETDMA: PROCEDURE(A);
109 2 DECLARE A ADDRESS;
110 2 CALL MONI(26,A);
111 2 END SETDMA;

/* INTELLEC 8 INTEL/ICOM READER INPUT */

112 1 INTIN: PROCEDURE BYTE;
      /* READ THE INTEL / ICOM READER */
113 2 DECLARE PTR1 LITERALLY '3', /* DATA */
      PTRS LITERALLY '1', /* STATUS */
      PTRC LITERALLY '1', /* COMMAND */

      PTPG LITERALLY '0CH', /* GO */
      PTRH LITERALLY '08H', /* STOP */

      /* STROBE THE READER */
      OUTPUT(PTRC) = PTRG;
      OUTPUT(PTRC) = PTRH;
      DO WHILE NOT ROL(INPUT(PTRC),3); /* NOT READY */
      END;
      /* DATA READY */
      RETURN INPUT(PTR1) AND 7FH;
      END INTIN;

120 1 DECLARE ZEROSUP BYTE, /* ZERO SUPPRESSION */
      (C3,C2,C1) BYTE, /* LINE COUNT ON PRINTER */

121 1 ERROR: PROCEDURE(A);
122 2 DECLARE A ADDRESS, I BYTE;
123 2 CALL PRINT(A); CALL PRINTCHAR(' ');
126 2 DO I = TCBP TO CBP;
127 3 IF I < COMLEN THEN CALL PRINTCHAR(COMBUFF(I));
129 3 END;

```

SER. #

PACIFIC GROVE, CA 93950

DIGITAL RESEARCH  
P. O. BOX 579

CP/M VERSION

```

130 2 /* ZERO THE COMLEN IN CASE THIS IS A SINGLE COMMAND */
      COMLEN = 0;
      /* DELETE ANY $$$ SUB FILES IN CASE BATCH PROCESSING */
131 2 CALL DELETE(.0,'$$$ SUB',0);
132 2 CALL CRLF;
133 2 GO TO RETRY;
134 2 END ERROR;

135 1 MOVE: PROCEDURE(S,D,N);
136 2 DECLARE (S,D) ADDRESS, N BYTE;
137 2 DECLARE A BASED S BYTE, B BASED D BYTE;
138 2 DO WHILE (N=N-1) <> 255;
139 3 B = A; S = S+1; D = D+1;
142 3 END;
143 2 END MOVE;

144 1 FILLSOURCE: PROCEDURE;
      /* FILL THE SOURCE BUFFERS */
145 2 DECLARE (I,J) BYTE;
146 2 NSOURCE = 0;
147 2 CALL SELECT(SDISK);
148 2 DO I = 0 TO NBUF;
      /* SET DMA ADDRESS TO NEXT BUFFER POSITION */
149 3 CALL SETDMA(SBUFF(NSOURCE));
150 3 IF (J := DISKREAD(SOURCE)) <> 0 THEN
151 3 DO; IF J <> 1 THEN
153 4 CALL ERROR(.'DISK READ ERROR');
      /* END - OF - FILE */
154 4 HARDEOF = NSOURCE; /* SET HARD END-OF-FILE */
155 4 SBUFF(NSOURCE) = ENDFILE; I = NBUF;
157 4 END; ELSE
158 3 NSOURCE = NSOURCE + 128;
159 3 END;
160 2 NSOURCE = 0;
161 2 END FILLSOURCE;

162 1 WRITEDEST: PROCEDURE;
      /* WRITE OUTPUT BUFFERS UP TO BUT NOT INCLUDING POSITION
      NDEST - THE LOW ORDER 7 BITS OF NDEST ARE ZERO */
163 2 DECLARE (I, J, N) BYTE;
164 2 DECLARE DMA ADDRESS;
165 2 DECLARE (EXTCNT, DATAOK) BYTE;
166 2 IF (N := LOW(SHR(NDEST,7)) - 1) = 255 THEN RETURN;
168 2 EXTCNT, NDEST = 0;
169 2 CALL SELECT(DDISK);
170 2 DO I = 0 TO N;
      /* SET DMA ADDRESS TO NEXT BUFFER */
171 3 DMA = DBUFF(NDEST);
172 3 IF VERIF THEN /* VERIFY MODE */
173 3 DO;
174 4 IF DEST(32) = 127 THEN /* END OF EXTENT */
175 4 DO; CALL MOVE(DMA,80H,80H);
177 5 DMA = 80H; EXTCNT = EXTCNT + 1;
179 5 END;
180 4 END;
181 3 CALL SETDMA(DMA);
182 3 IF DISKWRITE(.DEST) <> 0 THEN
183 3 CALL ERROR(.'DISK WRITE ERROR');
184 3 NDEST = NDEST + 128;

```

CP/M VERSION

COPYRIGHT © 1976

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. #

```

185 3      END;
186 2      IF VERIF THEN /* VERIFY DATA WRITTEN OK */
187 2      DO; I = N + 1; /* NUMBER OF BUFFERS */
189 3      NDEST = 0;
190 3      IF EXTCNT < 0 THEN /* WENT OVER CURRENT EXTENT */
191 3      DO; CALL SETDMA(80H); CALL CLOSE(.DEST);
194 4      DEST(FREEL) = DEST(FREEL) - EXTCNT;
195 4      J = DEST(32); /* SAVE CURRENT RECORD NUMBER */
196 4      CALL OPEN(.DEST);
197 4      /* ONE OR TWO EXTENTS WERE WRITTEN */
198 4      I = I - 128;
199 4      /* I MAY BE (TWO'S COMPLEMENT) NEGATIVE */
200 3      DEST(32) = (J - I) AND 7FH;
201 3      END; ELSE
202 3      DEST(32) = DEST(32) - 1; /* NEXT TO READ */
203 4      CALL SETDMA(80H); /* FOR COMPARE */
204 4      DO I = 0 TO N;
205 4      DATAK = DISKREAD(.DEST) = 0;
206 5      J = 0;
207 5      /* PERFORM COMPARISON */
208 5      DO WHILE DATAK AND J < 80H;
209 4      DATAK = BUFFER(J) = DBUFF(NDEST+J);
210 4      J = J + 1;
211 4      END;
212 4      NDEST = NDEST + 128;
213 3      IF NOT DATAK THEN
214 3      CALL ERROR(.'VERIFY ERROR');
215 3      END;
216 3      IF DEST(32) = 128 THEN /* INTO NEXT EXTENT */
217 3      DO; DEST(FREEL) = DEST(FREEL) + 1;
218 3      CALL OPEN(.DEST);
219 2      END;
220 2      NDEST = 0;
221 1      END WRITEDEST;
222 2
223 1      PUTDESTC, PROCEDURE(B);
224 2      DECLARE (B,IOB) BYTE;
225 2      /* WRITE BYTE B TO THE DESTINATION DEVICE GIVEN BY PDEST */
226 2      IF B >= 255 THEN
227 2      DO; COLUMN = COLUMN + 1;
228 3      IF DELET > 0 THEN /* MAY BE PAST RIGHT SIDE */
229 3      DO; IF COLUMN > DELET THEN RETURN;
230 3      END;
231 3      END;
232 2      IOB = IOBYTE; /* IN CASE IT IS ALTERED */
233 2      DO CASE PDEST;
234 3      /* CASE 0 IS THE DESTINATION FILE */
235 4      DO;
236 4      IF NDEST >= DBLEN THEN CALL WRITEDEST;
237 4      DBUFF(NDEST) = B;
238 4      NDEST = NDEST+1;
239 4      END;
240 3      /* CASE 1 IS ARD (ADDRESS) */
241 3      GO TO NOTDEST;
242 3      /* CASE 2 IS IRD (INTEL/ICOM) */
243 3      GO TO NOTDEST;
244 3      /* CASE 3 IS PTR */
245 3      GO TO NOTDEST;
246 3      /* CASE 4 IS UR1 */

```

SER. #

PACIFIC GROVE, CA 93950

COPYRIGHT © 1976  
DIGITAL RESEARCH  
P. O. BOX 579

CP/M VERSION

```

243 3      GO TO NOTDEST;
244 3      /* CASE 5 IS UR2 */
245 3      GO TO NOTDEST;
246 3      /* CASE 6 IS RDR */
247 3      NOTDEST;
248 3      CALL ERROR(.'NOT A CHARACTER SINK');
249 3      /* CASE 7 IS OUT */
250 3      CALL OUT(B);
251 3      /* CASE 8 IS LPT */
252 3      DO; IOBYTE = 1800*0000B; GO TO LSTL;
253 3      END;
254 3      /* CASE 9 IS UL1 */
255 3      DO; IOBYTE = 1100*0000B; GO TO LSTL;
256 3      END;
257 3      /* CASE 10 IS PRN (TABS EXPANDED, LINES LISTED, CHANGED TO LST) */
258 3      DO; IOBYTE = 1800*0000B; GO TO LSTL;
259 3      END;
260 3      /* CASE 11 IS LST */
261 3      LSTL;
262 3      CALL MONI(5,B);
263 3      /* CASE 12 IS PTP */
264 3      DO; IOBYTE = 8001*0000B; GO TO PUNL;
265 3      END;
266 3      /* CASE 13 IS UPI */
267 3      DO; IOBYTE = 8010*0000B; GO TO PUNL;
268 3      END;
269 3      /* CASE 14 IS UP2 */
270 3      DO; IOBYTE = 8011*0000B; GO TO PUNL;
271 3      END;
272 3      /* CASE 15 IS PUN */
273 3      PUNL;
274 3      CALL MONI(4,B);
275 3      /* CASE 16 IS TIY */
276 3      DO; IOBYTE = 0; GO TO CONL;
277 3      END;
278 3      /* CASE 17 IS CRT */
279 3      DO; IOBYTE = 1; GO TO CONL;
280 3      END;
281 3      /* CASE 18 IS UC1 */
282 3      DO; IOBYTE = 11B; GO TO CONL;
283 3      END;
284 3      /* CASE 19 IS CON */
285 3      CONL;
286 3      CALL MONI(2,B);
287 3      END;
288 2      IOBYTE = IOB;
289 2      END PUTDESTC;
290 1      PRINTI, PROCEDURE(B);
291 2      DECLARE B BYTE;
292 2      IF (ZEROSUP = ZERDSUP AND B = 0) THEN CALL PUTDESTC(' '); ELSE
293 2      CALL PUTDESTC('0'+B);
294 2      END PRINTI;
295 1      PRINTDIG, PROCEDURE(D);
296 2      DECLARE D BYTE;
297 2      CALL PRINTI(SHR(D,4)); CALL PRINTI(D AND 1111B);
298 2      END PRINTDIG;
299 1      NEWLINE, PROCEDURE;
300 1

```

SER. #

PACIFIC GROVE, CA 93950

COPYRIGHT © 1976  
DIGITAL RESEARCH  
P. O. BOX 579

CP/M VERSION

```

301 2 ZEROSUP = 1;
302 2 C1 = DEC(C1+ZEROSUP); C2 = DEC(C2 PLUS 0); C3 = DEC(C3 PLUS 0);
303 2 CALL PRINTDIG(C3); CALL PRINTDIG(C2); CALL PRINTDIG(C1);
304 2 CALL PUTDESTC(' ');
305 2 PUTNUM = FALSE;
306 2 END NEWLINE;

```

```

311 1 CLEARBUFF: PROCEDURE;
/* CLEAR OUTPUT BUFFER IN BLOCK MODE TRANSMISSION */
312 2 DECLARE NA ADDRESS;
313 2 DECLARE I BYTE;
314 2 I = LOW(NDEST) AND 7FH; /* REMAINING PARTIAL BUFFER LENGTH */
315 2 NA = NDEST AND 0FF00H; /* START OF SEGMENT NOT WRITTEN */
316 2 CALL WRITEST; /* CLEARS BUFFERS */
317 2 CALL MOVE(.DBUFF(NA),.DBUFF,I);
/* DATA MOVED TO BEGINNING OF BUFFER */
318 2 NDEST = I;
319 2 END CLEARBUFF;

```

```

320 1 PUTDEST: PROCEDURE(B);
321 2 DECLARE (I,B) BYTE;
/* WRITE DESTINATION CHARACTER, CHECK TABS AND LINES */
322 2 IF HUMB AND PUTNUM THEN CALL NEWLINE;
323 2 IF BLOCK THEN /* BLOCK MODE TRANSFER */
324 2 DO;
325 2 IF B = XOFF AND PDEST = 0 THEN
326 2 IF B = XOFF AND PDEST = 0 THEN
327 2 DO; CALL CLEARBUFF; /* BUFFERS WRITTEN */
328 2 RETURN; /* DON'T PASS THE X-OFF */
329 2 END;
330 2 END;
331 2 IF B = TAB THEN /* EXPAND TO NEXT TAB POSITION */
332 2 DO; IF TABS > 0 THEN
333 2 DO; I = COLUMN;
334 2 IF HUMB THEN I = I - 7; /* STARTING BIAS ON LINE */
335 2 DO WHILE I >= TABS;
336 2 I = I - TABS;
337 2 END;
338 2 I = TABS - I;
339 2 DO WHILE I > 0;
340 2 I = I - 1; CALL PUTDESTC(' ');
341 2 END;
342 2 ELSE CALL PUTDESTC(B);
343 2 END; ELSE
344 2 DO;
345 2 CALL PUTDESTC(B);
346 2 IF B = CR THEN COLUMN = 0; ELSE
347 2 /* MAY NEED NEWLINE NEXT TIME AROUND */
348 2 PUTNUM = B = LF;
349 2 END;
350 2 END PUTDEST;

```

```

357 1 UTRAN: PROCEDURE(B) BYTE;
358 2 DECLARE B BYTE;
/* TRANSLATE ALPHA TO UPPER CASE */
359 2 IF B >= 11000001B AND B <= 11110100B THEN /* LOWER CASE */
360 2 B = B AND 1011111B; /* TO UPPER CASE */
361 2 RETURN B;
362 2 END UTRAN;

```

```

363 1 LTRAN: PROCEDURE(B) BYTE;
364 2 DECLARE B BYTE;
/* TRANSLATE TO LOWER CASE ALPHA */
365 2 IF B >= 'A' AND B <= 'Z' THEN B = B OR 1000000B; /* TO LOWER */
366 2 RETURN B;
367 2 END LTRAN;

```

```

369 1 GETSOURCEC: PROCEDURE BYTE;
/* READ NEXT SOURCE CHARACTER */
370 2 DECLARE (IOB,B,CONCHK) BYTE;
371 2 IF PSOURCE - 1 <= RDR THEN /* 1 ... RDR+1 */
372 2 DO; IF (BLOCK OR HEX) AND CONBRK THEN
373 2 DO;
374 2 IF READCHAR = ENDFILE THEN RETURN ENDFILE;
375 2 CALL PRINTC('READER STOPPING',CR,LF,' ');
376 2 RETURN XOFF;
377 2 END;
378 2 END;
379 2 CONCHK = TRUE; /* CONSOLE STATUS CHECK BELOW */
380 2 IOB = IOBYTE; /* SAVE IT IN CASE IT IS ALTERED */
381 2 DO CASE PSOURCE;
382 2 /* CASE 0 IS SOURCE FILE */
383 2 DO; IF NSOURCE >= SBLEN THEN CALL FILLSOURCE;
384 2 B = SBUFF(NSOURCE);
385 2 NSOURCE = NSOURCE + 1;
386 2 END;
387 2 /* CASE 1 IS INP */
388 2 B = INP;
389 2 /* CASE 2 IS IRD (INTEL/ICOM) */
390 2 B = INTIN;
391 2 /* CASE 3 IS PTR */
392 2 DO; IOBYTE = 00000100B; GO TO RDRL;
393 2 END;
394 2 /* CASE 4 IS UR1 */
395 2 DO; IOBYTE = 00000100B; GO TO RDRL;
396 2 END;
397 2 /* CASE 5 IS UR2 */
398 2 DO; IOBYTE = 000001100B; GO TO RDRL;
399 2 END;
400 2 /* CASE 6 IS RDR */
401 2 RDRL:
402 2 B = MON2(3,B) AND 7FH;
403 2 /* CASE 7 IS OUT */
404 2 GO TO NOTSOURCE;
405 2 /* CASE 8 IS LPT */
406 2 GO TO NOTSOURCE;
407 2 /* CASE 9 IS UL1 */
408 2 GO TO NOTSOURCE;
409 2 /* CASE 10 IS PRN */
410 2 GO TO NOTSOURCE;
411 2 /* CASE 11 IS LST */
412 2 GO TO NOTSOURCE;
413 2 /* CASE 12 IS PTP */
414 2 GO TO NOTSOURCE;
415 2 /* CASE 13 IS UP1 */
416 2 GO TO NOTSOURCE;
417 2 /* CASE 14 IS UP2 */
418 2 GO TO NOTSOURCE;

```

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

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

```

413 3      /* CASE 15 IS PUN */
        NOTSOURCE;
        DO; CALL ERROR(,('NOT A CHARACTER SOURCE*'));
415 4      END;
        /* CASE 16 IS TTY */
416 3      DO; IOBYTE = 0; GO TO CONL;
419 4      END;
        /* CASE 17 IS CRT */
420 3      DO; IOBYTE = 01B; GO TO CONL;
423 4      END;
        /* CASE 18 IS UC1 */
424 3      DO; IOBYTE = 11B; GO TO CONL;
427 4      END;
        /* CASE 19 IS CON */
428 3      CONL;
        DO; CONCHK = FALSE; /* DON'T CHECK CONSOLE STATUS */
        B = MOH2(1,0);
430 4      END;
431 4      END; /* OF CASES */
432 3      IOBYTE = IOB; /* RESTORE IOBYTE */
433 2      IF ECHO THEN /* COPY TO CONSOLE DEVICE */
434 2      DO; IOB = PDEST; PDEST = CONP; CALL PUTDEST(B);
435 2      PDEST = IOB;
439 3      END;
440 3      IF CONCHK THEN /* TEST FOR CONSOLE CHAR READY */
441 2      DO;
442 2      IF SCOM THEN /* SOURCE IS A COM FILE */
443 3      CONCHK = (CONCNT = CONCNT + 1) = 0; ELSE /* ASCII */
444 3      CONCHK = B = LF;
445 3      IF CONCHK THEN
446 3      DO; IF CONBRK THEN
447 3      DO;
448 3      IF READCHAR = ENDFILE THEN RETURN ENDFILE;
449 3      CALL ERROR(,('ABORTED*'));
450 3      END;
451 3      END;
452 3      IF ZEPOP THEN B = B AND 7FH;
453 3      IF UPPER THEN RETURN UTRAN(B);
454 3      IF LOWER THEN RETURN LTRAN(B);
455 3      RETURN B;
456 3      END GETSOURCEC;
464 1      GETSOURCE: PROCEDURE BYTE;
465 2      /* GET NEXT SOURCE CHARACTER */
466 2      DECLARE CHAR BYTE;
        MATCH: PROCEDURE(B) BYTE;
        /* MATCH START AND QUIT STRINGS */
467 3      DECLARE (B,C) BYTE;
468 3      IF (C = COMBUFF(B, (B + MATCHLEN))) = ENDFILE THEN /* END MATCH */
469 3      DO; COMBUFF(B) = CHAR; /* SAVE CURRENT CHARACTER */
470 3      RETURN TRUE;
471 3      END;
472 3      IF C = CHAR THEN MATCHLEN = MATCHLEN + 1; ELSE
473 3      MATCHLEN = 0; /* NO MATCH */
474 3      RETURN FALSE;
475 3      END MATCH;
476 3      IF QUITLEN > 0 THEN
477 3      DO; IF (QUITLEN = QUITLEN - 1) = 1 THEN RETURN LF;
478 2      RETURN ENDFILE; /* TERMINATED WITH CR, LF, ENDFILE */
479 2
482 3

```

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

```

483 3      END;
484 2      DO FOREVER; /* LOOKING FOR START */
485 3      IF FEEDLEN > 0 THEN /* GET SEARCH CHARACTERS */
486 3      DO; FEEDLEN = FEEDLEN - 1;
487 3      CHAR = COMBUFF(FEEDBASE);
488 3      FEEDBASE = FEEDBASE + 1;
489 3      RETURN CHAR;
490 3      END;
491 3      IF (CHAR = GETSOURCEC) = ENDFILE THEN RETURN ENDFILE;
492 3      IF STARTS > 0 THEN /* LOOKING FOR START STRING */
493 3      DO; IF MATCH(STARTS) THEN
494 3      DO; FEEDBASE = STARTS; STARTS = 0;
495 3      FEEDLEN = MATCHLEN + 1;
496 3      END; /* OTHERWISE NO MATCH, SKIP CHARACTER */
497 3      END; ELSE
498 3      IF QUIT > 0 THEN /* PASS CHARACTERS TIL MATCH */
499 3      DO; IF MATCH(QUIT) THEN
500 3      DO; QUIT = 0; QUITLEN = 2;
501 3      /* SUBSEQUENTLY RETURN CR, LF, ENDFILE */
502 3      RETURN CR;
503 3      END; ELSE
504 3      RETURN CHAR;
505 3      END; /* OF DO FOREVER */
506 3      END GETSOURCEC;
516 1      DECLARE DISK BYTE; /* SELECTED DISK */
517 1      GNC: PROCEDURE BYTE;
518 2      IF (CBP = CBP + 1) >= COMLEN THEN RETURN CR;
519 2      RETURN UTRAN(COMBUFF(CBP));
520 2      END GNC;
521 2      DEBLANK: PROCEDURE;
522 1      DO WHILE (CHAR = GNC) = ' ';
523 2      END;
524 2      END DEBLANK;
525 2      SCAN: PROCEDURE(FCBA);
526 1      DECLARE FCBA ADDRESS; /* ADDRESS OF FCB TO FILL */
527 2      FCB BASED FCBA (FSIZE) BYTE; /* FCB TEMPLATE */
528 2      DECLARE (I,J,K) BYTE; /* TEMP COUNTERS */
        /* SCAN LOOKS FOR THE NEXT DELIMITER, DEVICE NAME, OR FILE NAME.
        THE VALUE OF CBP MUST BE 255 UPON ENTRY THE FIRST TIME */
529 2      DELIMITER: PROCEDURE(C) BYTE;
530 3      DECLARE (I,C) BYTE;
531 3      DECLARE DEL(*) BYTE DATA
532 3      (' ', '<', '>', CR, LA, LB, RB);
533 3      DO I = 0 TO LAST(DEL);
534 3      IF C = DEL(I) THEN RETURN TRUE;
535 3      END;
536 3      RETURN FALSE;
537 3      END DELIMITER;
538 2      PUTCHAR: PROCEDURE;
539 3      FCB(FLEN, FLEN+1) = CHAR;
540 3      IF CHAR = WHAT THEN AMBIG = TRUE; /* CONTAINS AMBIGUOUS REF */

```

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

```

542 3      END PUTCHAR;

543 2      FILLQ: PROCEDURE(LEN);
          /* FILL CURRENT NAME OR TYPE WITH QUESTION MARKS */
          DECLARE LEN BYTE;
          CHAR = WHAT; /* QUESTION MARK */
          DO WHILE FLEN < LEN;
            CALL PUTCHAR;
          END;
          END FILLQ;

550 2      GETFCB: PROCEDURE(1) BYTE;
          DECLARE 1 BYTE;
          RETURN FCB(1);
          END GETFCB;

554 2      SCANPAR: PROCEDURE;
          DECLARE (1,J) BYTE;
          /* SCAN OPTIONAL PARAMETERS */
          PARSET = TRUE;
          CHAR = GNC; /* SCAN PAST BRACKET */
          DO WHILE NOT(CHAR = CR OR CHAR = RB);
            IF (1 = CHAR - 'A') > 25 THEN /* NOT ALPHA */
              DO; IF CHAR = ' ' THEN CHAR = GNC; ELSE
                CALL ERROR( ('BAD PARAMETER'));
              END; ELSE
                DO; /* SCAN PARAMETER VALUE */
                  IF CHAR = 'S' OR CHAR = 'Q' THEN
                    DO; /* START OR QUIT COMMAND */
                      J = CBP + 1; /* START OF STRING */
                      DO WHILE NOT ((CHAR = GNC) = ENDFILE OR CHAR = CR);
                        CHAR = GNC;
                      END; ELSE
                        IF (J = (CHAR = GNC) - 'B') > 9 THEN J = 1;
                        ELSE
                          DO WHILE (K = (CHAR = GNC) - 'B') <= 9;
                            J = J * 10 + K;
                          END;
                          CONT(1) = J;
                        END;
                      END;
                    CHAR = GNC;
                  END SCANPAR;

583 2      CHKSET: PROCEDURE;
          IF CHAR = LA THEN CHAR = ' ';
          END CHKSET;

          /* INITIALIZE FILE CONTROL BLOCK TO EMPTY */
          ANBIG = FALSE; TYPE = ERR; CHAR = ' ' FLEN = 0;
          DO WHILE FLEN < FSIZE-1;
            IF FLEN = FSIZE THEN CHAR = 0;
            CALL PUTCHAR;
          END;

          /* DEBLANK COMMAND BUFFER */
          CALL DEBLANK;

          /* SAVE STARTING POSITION OF SCAN FOR DIAGNOSTICS */

```

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

```

597 2      TCBP = CBP;

          /* MAY BE A SEPARATOR */
          IF DELIMITER(CHAR) THEN
            DO; CALL CHKSET;
            TYPE = SPECL; RETURN;
          END;

          /* CHECK PERIPHERALS AND DISK FILES */
          DISK = 0;
          /* CLEAR PARAMETERS */
          DO I = 0 TO 25; CONT(1) = 0;
          END;
          PARSET = FALSE;
          FEEDLEN, MATCHLEN, QUITLEN = 0;
          /* SCAN NEXT NAME */
          DO FOREVER;
            FLEN = 0;
            DO WHILE NOT DELIMITER(CHAR);
              IF FLEN > NSIZE THEN /* ERROR, FILE NAME TOO LONG */
                RETURN;
              IF CHAR = '*' THEN CALL FILLQ(NSIZE); ELSE CALL PUTCHAR;
              CHAR = GNC;
            END;

          /* CHECK FOR DISK NAME OR DEVICE NAME */
          IF CHAR = '.' THEN
            DO; IF DISK < 0 THEN RETURN; /* ALREADY SET */
            IF FLEN = 1 THEN
              /* MAY BE DISK NAME A ... Z */
              DO;
                IF (DISK = GETFCB(1) - 'A' + 1) > 26 THEN
                  /* ERROR, INVALID DISK NAME */ RETURN;
                CALL DEBLANK; /* MAY BE DISK NAME ONLY */
                IF DELIMITER(CHAR) THEN
                  DO; IF CHAR = LB THEN
                      CALL SCANPAR;
                    CBP = CBP - 1;
                    TYPE = DISKNAME;
                    RETURN;
                  END; ELSE
                    END; ELSE

          /* MAY BE A THREE CHARACTER DEVICE NAME */
          IF FLEN < 3 THEN /* ERROR, CANNOT BE DEVICE NAME */
            RETURN; ELSE

          /* LOOK FOR DEVICE NAME */
          DO; DECLARE (1,J,K) BYTE; M LITERALLY '20';
            IO(*) BYTE DATA
            ('INIPRDPTRRUIURCRDRROUTLPTULIPRNLST',
             'PTPUIUP2PUNTTYCRTUC1CONHOLEOF',0);
            /* NOTE THAT ALL READER-LIKE DEVICES MUST BE
             PLACED BEFORE 'RDR', AND ALL LISTING-LIKE DEVICES
             MUST APPEAR BELOW LST, BUT ABOVE RDR. THE LITERAL
             DECLARATIONS FOR RDR, LST, AND PUMP MUST INDICATE
             THE POSITIONS OF THESE DEVICES IN THE LIST */
            J = 255;
            DO K = 0 TO M;
              I = 0;

```

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

```

645 6      DO WHILE ((I:=I+1) <= 3) AND
        IO(J+1) = GETFCB(I),
646 7      END;
647 6      IF I = 4 THEN /* COMPLETE MATCH */
648 6      DO; TYPE = PERIPH;
        /* SCAN PARAMETERS */
        IF GNC = LB THEN CALL SCANPAR;
        CBP = CBP - 1; CHAR = K;
        RETURN;
        END;
        /* OTHERWISE TRY NEXT DEVICE */ J = J + 3;
        END;

658 5      /* ERROR, NO DEVICE NAME MATCH */ RETURN;
659 5      END;
660 4      IF CHAR = LB THEN /* PARAMETERS FOLLOW */
661 4      CALL SCANPAR;
662 4      END; ELSE

663 3      /* CHAR IS NOT '.', SO FILE NAME IS SET. SCAN REMAINDER */
664 4      DO; IF FLEN = 0 THEN /* ERROR, NO PRIMARY NAME */
665 4      RETURN;
666 4      FLEN = FNAM;
667 4      IF CHAR = '.' THEN /* SCAN FILE TYPE */
668 4      DO WHILE NOT DELIMITER(CHAR, GNC);
669 5      IF FLEN >= FNSIZE THEN
670 5      /* ERROR, TYPE FIELD TOO LONG */ RETURN;
671 5      IF CHAR = '*' THEN CALL FILLQ(FNSIZE);
672 5      ELSE CALL PUTCHAR;
673 5      END;
674 5      END;

675 4      IF CHAR = LB THEN
676 4      CALL SCANPAR;
        /* RESCAN DELIMITER NEXT TIME AROUND */
        CBP = CBP - 1;
        TYPE = FILE;
        /* DISK IS THE SELECTED DISK (1 2 3 ... ) */
        IF DISK = 0 THEN DISK = CDISK + 1; /* DEFAULT */
        FCB(0), FCB(32) = 0;
        RETURN;
        END;
        END;
        END SCAN;

686 1      NULLS, PROCEDURE;
        /* SEND 40 NULLS TO OUTPUT DEVICE */
        DECLARE I BYTE;
        DO I = 0 TO 39; CALL PUTDEST(0);
        END;
        END NULLS;

692 1      DECLARE FEXTH(FEXTL) BYTE, /* HOLDS DESTINATION FILE TYPE */
        COPYING BYTE; /* TRUE WHILE COPYING TO DEST FILE */

693 1      MOVEXT, PROCEDURE(A);
694 2      DECLARE A ADDRESS;
        /* MOVE THREE CHARACTER EXTENT INTO DEST FCB */
        CALL MOVE(A, DEST(FEXT), FEXTL);
        END MOVEXT;

```

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

```

697 1      EQUAL, PROCEDURE(A, B) BYTE;
        /* COMPARE THE STRINGS AT A AND B UNTIL EITHER A MISMATCH OR
        A '$' IS ENCOUNTERED IN STRING B */
        DECLARE (A, B) ADDRESS;
        (SA BASED A, SB BASED B) BYTE;
        DO WHILE SB <> '$';
        IF SB <> SA THEN RETURN FALSE;
        A = A + 1; B = B + 1;
        END;
        RETURN TRUE;
        END EQUAL;

707 1      READ$EOF, PROCEDURE BYTE;
        /* RETURN TRUE IF END OF FILE */
        CHAR = GETSOURCE;
        IF SCOM THEN RETURN HARDEOF (= NSOURCE);
        RETURN CHAR = ENDFILE;
        END READ$EOF;

713 1      HEWRECORD, PROCEDURE BYTE;
        /* READ ONE RECORD INTO SBUFF AND CHECK FOR PROPER FORM
        RETURNS 0 IF RECORD OK
        RETURNS 1 IF END OF TAPE (.00000)
        RETURNS 2 IF ERROR IN RECORD */

714 2      DECLARE KOFFSET BYTE; /* TRUE IF KOFF RECVD */
715 2      DECLARE NOERRS BYTE; /* TRUE IF NO ERRORS IN THIS RECORD */

716 2      PRINTERR, PROCEDURE(A);
        /* PRINT ERROR MESSAGE IF NOERRS TRUE */
        DECLARE A ADDRESS;
        IF NOERRS THEN
        DO; NOERRS = FALSE;
        CALL PRINT(A);
        END;
        END PRINTERR;

724 2      CHECKXOFF, PROCEDURE;
725 3      IF XOFFSET THEN
726 3      DO; XOFFSET = FALSE;
727 4      CALL CLEARBUFF;
728 4      END;
729 4      END CHECKXOFF;

731 2      SAVECHAR, PROCEDURE BYTE;
        /* READ CHARACTER AND SAVE IN BUFFER */
        DECLARE I BYTE;
        IF NOERRS THEN
        DO;
        DO WHILE (I := GETSOURCE) = XOFF; XOFFSET = TRUE;
        END;
        HBUFF(NSOURCE) = I;
        IF (NSOURCE := NSOURCE + 1) >= LAST(HBUFF) THEN
        CALL PRINTERR( ('RECORD TOO LONG'));
        RETURN I;
        END;
        RETURN ENDFILE; /* ON ERROR FLAG */

```

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

```

744 3      END SAVECHAR;
745 2      DECLARE (M, RL, CS, RT) BYTE;
          LDA ADDRESS; /* LOAD ADDRESS WHICH FOLLOWS */
746 2      READHEX, PROCEDURE BYTE;
747 3      DECLARE H BYTE;
748 3      IF (H, = SAVECHAR) - '0' <= 9 THEN RETURN H-'0';
750 3      IF H - 'A' > 5 THEN
751 3          CALL PRINTER('INVALID DIGIT$');
752 3      RETURN H - 'A' + 10;
753 3      END READHEX;
754 2      READBYTE, PROCEDURE BYTE;
          /* READ TWO HEX DIGITS */
755 3      RETURN SHL(READHEX, 4) OR READHEX;
756 3      END READBYTE;
757 2      READCS, PROCEDURE BYTE;
          /* READ BYTE WITH CHECKSUM */
758 3      RETURN CS, = CS + READBYTE;
759 3      END READCS;
760 2      READADDR, PROCEDURE ADDRESS;
          /* READ DOUBLE BYTE WITH CHECKSUM */
761 3      RETURN SHL(DOUBLE(READCS), 8) OR READCS;
762 3      END READADDR;
763 2      NOERRS = TRUE; /* NO ERRORS DETECTED IN THIS RECORD */
          /* READ NEXT RECORD */
          /* SCAN FOR THE ',' */
764 2      HSOURCE = 0;
765 2      DO WHILE (CS, = SAVECHAR) <> ',';
766 3      HSOURCE = 0;
767 3      IF CS = ENDFILE THEN
768 3          DO; CALL PRINT('END OF FILE, CTL-Z', WHAT, '$');
769 4          IF READCHAR = ENDFILE THEN RETURN 1;
770 4          ELSE HSOURCE = 0;
771 4          END;
772 4          CALL CHECKXOFF;
773 4          END;
774 3          /* ', ' FOUND */
775 3          CS = 0;
776 2          IF (RL, = READCS) = 0 THEN /* END OF TAPE */
777 2          DO; DO WHILE (RL, = SAVECHAR) <> ENDFILE;
778 4          CALL CHECKXOFF;
779 4          END;
780 4          IF NOERRS THEN RETURN 1;
781 3          RETURN 2;
782 3          END;
783 3          /* RECORD LENGTH IS NOT ZERO */
784 2          LDA = READADDR; /* LOAD ADDRESS */
785 3          /* READ WORDS UNTIL RECORD LENGTH EXHAUSTED */
          RT = READCS; /* RECORD TYPE */
          DO WHILE RL <> 0 AND NOERRS; RL = RL - 1;
          M = READCS;
786 2
787 2
788 2
790 3

```

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

```

791 3      /* INCREMENT LA HERE FOR EXACT ADDRESS */
          END;
792 2      /* CHECK SUM */
793 2      IF CS + READBYTE <> 0 THEN
          CALL PRINTER('CHECKSUM ERROR$');
794 2      CALL CHECKXOFF;
795 2      IF NOERRS THEN RETURN 0;
797 2      RETURN 2;
798 2      END HEXRECORD;
799 1      READTAPE, PROCEDURE;
          /* READ HEX FILE FROM HIGH SPEED READER TO 'HEX' FILE.
          CHECK EACH RECORD FOR VALID DIGITS, AND PROPER CHECKSUM */
          DECLARE (I, A) BYTE;
          DO FOREVER;
          DO WHILE (I, = HEXRECORD) <= 1;
          IF NOT (I = 1 AND IGNORE) THEN
          DO A = 1 TO HSOURCE;
          CALL PUTDEST(HBUFF(A-1));
          END;
          CALL PUTDEST(CR); CALL PUTDEST(LF);
          IF I = 1 THEN /* END OF TAPE ENCOUNTERED */
          RETURN;
          END;
          CALL CRLF; HBUFF(HSOURCE) = '$';
          CALL PRINT(HBUFF);
          CALL PRINT('CORRECT ERROR, TYPE RETURN OR CTL-Z$');
          CALL CRLF;
          IF READCHAR = ENDFILE THEN RETURN;
          END;
          END READTAPE;
800 2
801 2
802 3
803 4
804 4
805 5
806 5
807 4
808 4
809 4
810 4
811 4
812 3
814 3
815 3
816 3
817 3
819 3
820 2
          END READTAPE;
821 1      FORMERR, PROCEDURE;
822 2      CALL ERROR('INVALID FORMAT$');
823 2      END FORMERR;
824 1      SETUPDEST, PROCEDURE;
825 2      CALL SELECT(DISK);
826 2      DHEX = EQUAL(.DEST(FEXT), 'HEX$');
827 2      CALL MOVE(.DEST(FEXT), FEXT, FEXTL); /* SAVE TYPE */
828 2      CALL MOVEXT('$$$'); CALL DELETE(.DEST); /* REMOVE OLD $$$ FILE */
829 2      CALL MAKE(.DEST); /* CREATE A NEW ONE */
830 2      IF DCNT = 255 THEN CALL ERROR('NO DIRECTORY SPACES$');
831 2      DEST(32), NDEST = 0;
832 2      DEST(32), NDEST = 0;
833 2      END SETUPDEST;
834 2
835 1      SETUPSOURCE, PROCEDURE;
836 2      HARDEOF = 0FFFFH;
837 2      CALL SELECT(SDISK);
838 2      CALL OPEN(.SOURCE);
839 2      IF DCNT = 255 THEN CALL ERROR('NO FILE$');
840 2      SOURCE(32) = 0;
841 2      /* CAUSE IMMEDIATE READ */
842 2      SCOM = EQUAL(.SOURCE(FEXT), 'COM$') OR
          EQUAL(.SOURCE(FEXT), 'CHI$');
843 2      HSOURCE = SBLN;
844 2      END SETUPSOURCE;

```

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

```

845 1  CHECK$SIRINGS, PROCEDURE;
846 2  IF STARTS > 0 THEN
847 2  CALL ERROR( ('START NOT FOUND$'));
848 2  IF QUIT$ > 0 THEN
849 2  CALL ERROR( ('QUIT NOT FOUND$'));
850 2  END CHECK$SIRINGS;

851 1  CLOSEDEST, PROCEDURE;
      /* CLEAR BUFFER */
852 2  DO WHILE (LOW(NDEST) AND 7FH) (<) 0;
853 3  CALL PUTDEST(ENDFILE);
854 3  END;
855 2  CALL CHECK$SIRINGS;
856 2  CALL WRITEDEST;
857 2  CALL SELECT(DDISK);
858 2  CALL CLOSE(.DEST);
859 2  IF DCNT = 255 THEN CALL ERROR( ('WRITE PROTECTED', WHAT, '$'));
860 2  CALL MOVEXT(.FEXTH); /* RECALL ORIGINAL TYPE */
861 2  CALL DELETE(.DEST); /* REMOVE OLD FILE */
862 2  CALL MOVE(.DEST, .DEST(16), 16); /* READY FOR RENAME */
863 2  CALL MOVEXT( ('$$$') );
864 2  CALL RENAME(.DEST);
865 2  END CLOSEDEST;

866 2

867 1  SIZE$NBUF, PROCEDURE;
      /* COMPUTE NUMBER OF BUFFERS - 1 FROM DBLEN */
868 2  NBUF = (SHR(DBLEN, 7) AND 0FFH) - 1;
      /* COMPUTED AS DBLEN/128-1, WHERE DBLEN <= 32K (AND THUS
869 2  NBUF RESULTS IN A VALUE <= 2**15/2**7-1 = 2**8-1 = 255) */
      END SIZE$NBUF;

870 1  SET$DBLEN, PROCEDURE;
      /* ABSORB THE SOURCE BUFFER INTO THE DEST BUFFER */
871 2  SBASE = .MEMORY;
872 2  IF DBLEN >= 4000H THEN DBLEN = 7F80H; ELSE
873 2  DBLEN = DBLEN + SBLEN;
874 2  CALL SIZE$NBUF;
875 2  END SET$DBLEN;

876 2

877 1  SIZE$MEMORY, PROCEDURE;
      /* SET UP SOURCE AND DESTINATION BUFFERS */
878 2  SBASE = .MEMORY + SHR(MEMSIZE - .MEMORY, 1);
879 2  SBLEN, DBLEN = SHR(MEMSIZE - .MEMORY) AND 0FF00H, 1;
880 2  CALL SIZE$NBUF;
881 2  END SIZE$MEMORY;

882 1  COPYCHAR, PROCEDURE;
      /* PERFORM THE ACTUAL COPY FUNCTION */
883 2  DECLARE RESIZED BYTE; /* TRUE IF SBUFF AND DBUFF COMBINED */
884 2  IF (RESIZED = (BLOCK AND PSOURCE (<) 0)) THEN /* BLOCK MODE */
885 2  CALL SET$DBLEN; /* ABSORB SOURCE BUFFER */
886 2  IF NEXT OR IGNORE THEN /* HEX FILE */
887 2  CALL READTAPE; ELSE
888 2  DO WHILE NOT READ$EOF;
889 3  CALL PUTDEST(CHAR);
890 3  END;
891 2  IF RESIZED THEN
892 2  DO; CALL CLEARBUFF;
893 2  CALL SIZE$MEMORY;
894 3  END;
895 3

```

2/.. VERSION \_\_\_\_\_  
 COPYR.G.I.F © 1976  
 DIGITAL RESEARCH  
 P. O. BOX 579  
 PACIFIC GROVE, CA. 93950  
 SER. # \_\_\_\_\_

```

896 2  END COPYCHAR;

897 1  SIMPLECOPY, PROCEDURE;
898 2  DECLARE (FASTCOPY, I) BYTE;
899 2  REAL$EOF, PROCEDURE BYTE;
900 3  RETURN HARDEOF (<) 0FFFFH;
901 3  END REAL$EOF;
902 2  CALL SIZE$MEMORY;
903 2  TCBP = MCBP; /* FOR ERROR TRACING */
904 2  CALL SETUPDEST;
905 2  CALL SETUPSOURCE;
      /* FILES READY FOR DIRECT COPY */
906 2  FASTCOPY = TRUE;
      /* LOOK FOR PARAMETERS */
907 2  DO I = 0 TO 25;
908 3  IF CONT(I) (<) 0 THEN
909 4  DO;
910 5  IF NOT(I = 14 OR I = 21) THEN
911 6  /* NOT OBJ OR VERIFY */
912 6  FASTCOPY = FALSE;
913 6  END;
914 5  END;
915 2  IF FASTCOPY THEN /* COPY DIRECTLY TO DBUFF */
916 3  DO; CALL SET$DBLEN; /* EXTEND DBUFF */
917 4  DO WHILE NOT REAL$EOF;
918 5  CALL FILLSOURCE;
919 5  IF REAL$EOF THEN
920 6  NDEST = HARDEOF; ELSE NDEST = DBLEN;
921 6  CALL WRITEDEST;
922 6  END;
923 4  END; ELSE
924 3  CALL COPYCHAR;
925 3  CALL CLOSEDEST;
926 3  END SIMPLECOPY;

927 2

928 1  MULTICOPY, PROCEDURE;
929 2  DECLARE (NEXTDIR, NCOPIED) BYTE;
930 2  PRNAME, PROCEDURE;
      /* PRINT CURRENT FILE NAME */
931 3  DECLARE (I, C) BYTE;
932 3  CALL CRLF;
933 3  DO I = 1 TO FNSIZE;
934 4  IF (C = DEST(I)) (<) ' ' THEN
935 5  DO; IF I = FEXT THEN CALL PRINTCHAR( '. ' );
936 5  CALL PRINTCHAR(C);
937 5  END;
938 4  END;
939 3  END PRNAME;
940 3  END MULTICOPY;

941 3

942 2  NEXTDIR, NCOPIED = 0;
943 2  DO FOREVER;
      /* FIND A MATCHING ENTRY */
944 3  CALL SELECT(SDISK);
945 3  CALL SETDMA(.BUFFER);
946 3  CALL SEARCH(.SEARFCB);
947 3  DO WHILE DCNT < NEXTDIR;
948 4  CALL SEARCHN;
949 4  END;
      /* FILE CONTROL BLOCK IN BUFFER */
950 3  IF DCNT = 255 THEN

```

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

```

951 3      DO; IF NCOPIED = 0 THEN
953 4      CALL ERROR('NOT FOUND'); CALL CRLF;
955 4      RETURN;
956 4      END;
957 3      NEXTDIR = DCNT + 1;
          /* GET THE FILE CONTROL BLOCK NAME TO DEST */
958 3      CALL MOVE(BUFFER+SHL(DCNT AND 11B,5),DEST,16);
959 3      CALL MOVE(DEST,SOURCE,16); /* FILL BOTH FCB'S */
960 3      IF (NCOPIED = NCOPIED + 1) = 1 THEN
961 3          CALL PRINT('COPYING -S');
962 3          CALL PRNAME;
963 3          CALL SIMPLECOPY;
964 3          END;
965 2      END MULTICOPY;

966 1      SET$SDISK: PROCEDURE;
967 2      IF DISK > 0 THEN SDISK = DISK - 1; ELSE SDISK = CDISK;
970 2      END SET$SDISK;

971 1      SET$DDISK: PROCEDURE;
972 2      IF PARSET THEN /* PARAMETERS PRESENT */ CALL FORMERR;
974 2      IF DISK > 0 THEN DDISK = DISK - 1; ELSE DDISK = CDISK;
977 2      END SET$DDISK;

978 1      CHECK$DISK: PROCEDURE;
979 2      IF DDISK = SDISK THEN CALL FORMERR;
981 2      END CHECK$DISK;

982 1      CHECK$EOL: PROCEDURE;
983 2      CALL DEBLANK;
984 2      IF CHAR <> CR THEN CALL FORMERR;
986 2      END CHECK$EOL;

987 1      SCANDEST: PROCEDURE(COPYFCB);
988 2      DECLARE COPYFCB ADDRESS;
989 2      CALL SET$SDISK;
990 2      CALL CHECK$EOL;
991 2      CALL MOVE(SOURCE,COPYFCB,33);
992 2      CALL CHECK$DISK;
993 2      END SCANDEST;

994 1      SCANEQL: PROCEDURE;
995 2      CALL SCAN(SOURCE);
996 2      IF NOT (TYPE = SPECL AND CHAR = '=') THEN CALL FORMERR;
998 2      MCBP = CBP; /* FOR ERROR PRINTING */
999 2      END SCANEQL;

/* BUFFER AT 80H CONTAINS REMAINDER OF LINE TYPED
FOLLOWING THE COMMAND 'PIP' - IF ZERO THEN PROMPT TIL CR */
1000 1      CALL MOVE(80H,COMLEN,80H);
1001 1      MULTCOM = COMLEN = 0;

/* GET CURRENT DISK */
1002 1      CDISK = MOH2(25,0);

1003 1      RETRY;
          /* ENTER HERE ON ERROR EXIT FROM THE PROCEDURE 'ERROR' */
          CALL SIZE$MEMORY;
          /* MAIN PROCESSING LOOP. PROCESS UNTIL CR ONLY */

```

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

```

1004 1      DO FOREVER;
1005 2      CALL LIFHEAD;
1006 2      CONCNT,COLUMN = 0; /* PRINTER TABS */
          /* READ FROM CONSOLE IF NOT A ONELINER */
          IF MULTCOM THEN
1007 2          DO; CALL PRINTCHAR('*'); CALL READCOM;
1008 2          CALL CRLF;
1009 3          END;
          CBP = 255;
          IF COMLEN = 0 THEN /* SINGLE CARRIAGE RETURN */
1010 2          DO; CALL SELECT(CDISK);
1011 3          CALL EXIT;
1012 3          END;

/* LOOK FOR SPECIAL CASES FIRST */
DDISK,SDISK,PSOURCE,PDEST = 0;
CALL SCAN(DEST);
IF TYPE = PERIPH THEN GO TO SIMPLECOM;
IF TYPE = DISKNAME THEN
1013 2      DO; DDISK = DISK - 1;
1014 3      CALL SCANEQL;
1015 3      CALL SCAN(SOURCE);
          /* MAY BE MULTI COPY */
          IF TYPE <> FILE THEN CALL FORMERR;
          IF AMBIG THEN
1016 3          DO; CALL SCANDEST(SEARFCB);
1017 4          CALL MULTICOPY;
1018 4          END; ELSE
1019 3          DO; CALL SCANDEST(DEST);
          /* FORM IS A:=B:UFN */
          CALL SIMPLECOPY;
          END;
          GO TO ENDCOM;
          END;

IF TYPE <> FILE OR AMBIG THEN CALL FORMERR;
CALL SET$DDISK;
CALL SCANEQL;
CALL SCAN(SOURCE);
IF TYPE = DISKNAME THEN
1020 2      DO;
1021 3      CALL SET$SDISK; CALL CHECK$DISK;
1022 3      CALL MOVE(DEST,SOURCE,33);
1023 3      CALL CHECK$EOL;
1024 3      CALL SIMPLECOPY;
1025 3      GO TO ENDCOM;
1026 3      END;
          /* MAY BE POSSIBLE TO DO A FAST DISK COPY */
          IF TYPE = FILE THEN /* FILE TO FILE */
1027 2          DO; CALL DEBLANK; IF CHAR <> CR THEN GO TO SIMPLECOM;
          /* FILE TO FILE */
1028 3          CALL SET$SDISK;
1029 3          CALL SIMPLECOPY;
1030 3          GO TO ENDCOM;
1031 3          END;

1032 2      SIMPLECOM;
          CBP = 255; /* READY FOR RESCAN */

```

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

```

1065 2 /* OTHERWISE PROCESS SIMPLE REQUEST */
1066 2 CALL SCAN(.DEST);
1067 2 IF (TYPE < FILE) OR AMBIG THEN /* DELIMITER OR ERROR */
    CALL ERROR(.'UNRECOGNIZED DESTINATION$');

1068 2 DHEX = FALSE;
1069 2 IF TYPE = FILE THEN
1070 2 DO; /* DESTINATION IS A FILE, SAVE EXTENT NAME */
1071 3 CALL SET$DDISK;
1072 3 CALL SETUPDEST;
1073 3 CHAR = 255;
1074 3 END; ELSE
/* PERIPHERAL NAME */
1075 2 IF CHAR >= NULP OR CHAR <= RDR THEN CALL ERROR(.'CANNOT WRITE$');

    IF (PDEST = CHAR + 1) = PUNP THEN CALL NULLS;

/* NOW SCAN THE DELIMITER */
1079 2 CALL SCAN(.SOURCE);
1080 2 IF TYPE <> SPECL OR CHAR <> '=' THEN
1081 2 CALL ERROR(.'INVALID PIP FORMATS$');

/* OTHERWISE SCAN AND COPY UNTIL CR */
1082 2 C1,C2,C3 = 0; /* CLEAR LINE COUNTERS */
1083 2 COPYING = TRUE;
1084 2 DO WHILE COPYING;
1085 3 CALL SCAN(.SOURCE);
1086 3 SCOM = FALSE;
1087 3 IF TYPE = FILE AND NOT AMBIG THEN /* A SOURCE FILE */
1088 4 DO;
1089 4 CALL SET$DDISK;
1090 4 CALL SETUPSOURCE;
1091 4 CHAR = 255;
1092 4 END; ELSE

1093 3 IF TYPE <> PERIPH OR (CHAR <= LST AND CHAR > RDR) THEN
1094 3 CALL ERROR(.'CANNOT READ$');

1096 3 COLUMN = 0;
1097 3 PUTHUM = TRUE; /* CAUSES IMMEDIATE NEWLINE IF HUMB SET */
1098 3 SCOM = SCOM OR OBJ; /* MAY BE ABSOLUTE COPY */
1099 3 PSOURCE = CHAR + 1;
1100 3 IF CHAR = NULP THEN CALL NULLS; ELSE
1101 3 IF CHAR = EDFP THEN CALL PUTDEST(ENDFILE); ELSE
1102 3 DO; /* DISK COPY */
1103 4 IF (CHAR < HSRDR AND DHEX) THEN NEXT = 1;
1104 4 /* HEX FILE SET IF SOURCE IS RDR AND DEST IS HEX FILE */
1105 4 IF PDEST = PRNT THEN
1106 5 DO; TABS = 0; HUMB = 1;
1107 5 END;
1108 5 CALL COPYCHAR;
1109 5 END;
1110 4
1111 3 CALL CHECK$STRINGS;
1112 3 /* READ ENDFILE, GO TO NEXT SOURCE */
1113 3 CALL SCAN(.SOURCE);
1114 3 IF TYPE <> SPECL OR (CHAR <> ',' AND CHAR <> CR) THEN
1115 3 CALL ERROR(.'INVALID SEPARATOR$');
1116 3

```

CP/M VERSION \_\_\_\_\_

COPYRIGHT © 1976

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA. 93950

SER. # \_\_\_\_\_

```

1117 3 COPYING = CHAR <> CR;
1118 3 END;

1119 2 /* IF NECESSARY, CLOSE FILE OR PUNCH TRAILER */
1120 2 IF PDEST = PUNP THEN
1121 3 DO; CALL PUTDEST(ENDFILE); CALL NULLS;
1122 3 END;
1123 2 IF PDEST = 0 THEN /* FILE HAS TO BE CLOSED AND RENAMED */
1124 2 CALL CLOSEDEST;

1125 2 /* COMLEN SET TO 0 IF NOT PROCESSING MULTIPLE COMMANDS */
1126 2 ENDCOM;
    COMLEN = MULTCOM;

1127 2 END; /* DO FOREVER */
1128 1 END;

```

#### MODULE INFORMATION.

|                    |         |       |
|--------------------|---------|-------|
| CODE AREA SIZE     | = 1947H | 6471D |
| VARIABLE AREA SIZE | = 01CAH | 458D  |
| MAXIMUM STACK SIZE | = 0032H | 50D   |
| 1396 LINES READ    |         |       |
| 0 PROGRAM ERROR(S) |         |       |

END OF PL/M-80 COMPILATION

CP/M VERSION \_\_\_\_\_

COPYRIGHT © 1976

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA. 93950

SER. # \_\_\_\_\_

|           |           |           |            |           |            |           |              |           |           |
|-----------|-----------|-----------|------------|-----------|------------|-----------|--------------|-----------|-----------|
| 1D4FH SYM | MEMORY    | 1C45H SYM | DCNT       | 1D03H SYM | N          | 1389H SYM | GETFCB       | 1D46H SYM | C         |
| 0200H SYM | COPYRIGHT | 074EH SYM | INITIALIZE | 0930H SYM | FILLSOURCE | 1D27H SYM | I            | 1A4EH SYM | SETSDISK  |
| 0236H SYM | INPLOC    | 0767H SYM | SELECT     | 1D04H SYM | I          | 1399H SYM | SCANPAR      | 1A68H SYM | SETDDISK  |
| 0230H SYM | OUTLOC    | 1C46H SYM | D          | 1D05H SYM | J          | 1D28H SYM | I            | 1A8CH SYM | CHECKDISK |
| 0736H SYM | OUT       | 07C7H SYM | OPEN       | 09ABH SYM | WRITEDST   | 1D29H SYM | J            | 1A9AH SYM | CHECKEOL  |
| 1E55H SYM | B         | 1C47H SYM | FCB        | 1D06H SYM | I          | 1488H SYM | CHKSET       | 1AA9H SYM | SCANDEST  |
| 0743H SYM | INP       | 07DAH SYM | CLOSE      | 1D07H SYM | J          | 1D2AH SYM | I            | 1D4DH SYM | COPYFCB   |
| 074EH SYM | TIMOUT    | 1C49H SYM | FCB        | 1D08H SYM | N          | 1D2BH SYM | J            | 1AC7H SYM | SCANEOL   |
| 1006H SYM | EXIT      | 07FDH SYM | SEARCH     | 1D09H SYM | DMA        | 1D2CH SYM | K            | 0475H SYM | RETRY     |
| 1E33H SYM | COLUMN    | 1C40H SYM | FCB        | 1D0BH SYM | EXTCNT     | 0245H SYM | IO           | 057CH SYM | SIMPLECOM |
| 1099H SYM | AMBIG     | 0000H SYM | SEARCHN    | 1D0CH SYM | DATAOK     | 1496H SYM | NULLS        | 072BH SYM | ENDCOM    |
| 109AH SYM | PARSET    | 000CH SYM | DELETE     | 002BH SYM | PUTDESTC   | 1D2DH SYM | I            |           |           |
| 109BH SYM | FEEDBASE  | 1C4DH SYM | FCB        | 1D0DH SYM | B          | 1D2EH SYM | FEXTH        |           |           |
| 100CH SYM | FEEDLEN   | 001CH SYM | DISKREAD   | 1D0EH SYM | IOB        | 1D31H SYM | COPYING      |           |           |
| 100DH SYM | MATCHLEN  | 1C4FH SYM | FCB        | 0090H SYM | NOTDEST    | 14B1H SYM | MOVEXT       |           |           |
| 100EH SYM | QUITLEN   | 002CH SYM | DISKWRITE  | 00CCH SYM | LSTL       | 1D32H SYM | A            |           |           |
| 109FH SYM | HBUF      | 1C51H SYM | FCB        | 00FBH SYM | PUNL       | 14C4H SYM | EQUAL        |           |           |
| 1090H SYM | CDISK     | 003CH SYM | MAKE       | 0C2AH SYM | CONL       | 1D34H SYM | A            |           |           |
| 0090H SYM | BUFFER    | 1C53H SYM | FCB        | 0C67H SYM | PRINT1     | 1D36H SYM | B            |           |           |
| 005CH SYM | SEAPFCB   | 004FH SYM | RENAME     | 1DAFH SYM | B          | 14FBH SYM | READEOF      |           |           |
| 0006H SYM | MEMSIZE   | 1C55H SYM | FCB        | 0C8EH SYM | PRINTDIG   | 151DH SYM | HEXRECORD    |           |           |
| 1091H SYM | SLEN      | 1C57H SYM | CBUFF      | 1D10H SYM | D          | 1D38H SYM | XOFFSET      |           |           |
| 1093H SYM | DBLEN     | 1C57H SYM | MAXLEN     | 0CA9H SYM | NEWLINE    | 1D39H SYM | HGERRS       |           |           |
| 1095H SYM | SBASE     | 1C58H SYM | COMLEN     | 0CE5H SYM | CLEARBUFF  | 15D4H SYM | PRINTERR     |           |           |
| 1D4FH SYM | DBUFF     | 1C59H SYM | COMBUFF    | 1D11H SYM | NA         | 1D3AH SYM | A            |           |           |
| 1097H SYM | SDISK     | 1CD9H SYM | TCBP       | 1D13H SYM | I          | 15EFH SYM | CHECKXOFF    |           |           |
| 1098H SYM | SCOM      | 1CD4H SYM | CBP        | 0D12H SYM | PUTDEST    | 15FFH SYM | SAVECHAR     |           |           |
| 1099H SYM | DHEX      | 005FH SYM | READCOM    | 1D14H SYM | B          | 1D3CH SYM | I            |           |           |
| 109AH SYM | SOURCE    | 1CDEH SYM | MCBP       | 1D15H SYM | I          | 1D3DH SYM | M            |           |           |
| 1E6BH SYM | DEST      | 006DH SYM | CONBRK     | 0DD1H SYM | UTRAN      | 1D3EH SYM | RL           |           |           |
| 1D0CH SYM | DDISK     | 0003H SYM | IOBYTE     | 1D16H SYM | B          | 1D3FH SYM | CS           |           |           |
| 1D0DH SYM | HBUFF     | 1CDCH SYM | CONT       | 0DF8H SYM | LTRAN      | 1D40H SYM | RT           |           |           |
| 1C2DH SYM | HSOURCE   | 1CDDH SYM | BLOCK      | 1D17H SYM | B          | 1D41H SYM | LDA          |           |           |
| 1C2EH SYM | NSOURCE   | 1CDFH SYM | DELET      | 0E1FH SYM | GETSOURCEC | 163FH SYM | READHEX      |           |           |
| 1C30H SYM | HARDEOF   | 1CE0H SYM | ECHO       | 1D18H SYM | IOB        | 1D43H SYM | H            |           |           |
| 1C32H SYM | NDEST     | 1CE3H SYM | HEXT       | 1D19H SYM | B          | 166EH SYM | READBYTE     |           |           |
| 1C34H SYM | PDEST     | 1CE4H SYM | IGNOR      | 1D1AH SYM | CONCHK     | 167DH SYM | READCS       |           |           |
| 1C35H SYM | PSOURCE   | 1CE7H SYM | LOWER      | 0EC3H SYM | RDRL       | 1696H SYM | READADDR     |           |           |
| 1C36H SYM | MULTCOM   | 1CE9H SYM | NUMB       | 0EEBH SYM | NOTSOURCE  | 169CH SYM | READTAPE     |           |           |
| 1C37H SYM | PUTNUM    | 1CEAH SYM | OBJ        | 0F15H SYM | CONL       | 1D44H SYM | I            |           |           |
| 1C38H SYM | CONCHT    | 1CECH SYM | QUITS      | 0FF0H SYM | GETSOURCE  | 1D45H SYM | A            |           |           |
| 1C39H SYM | CHAR      | 1CEEH SYM | STARTS     | 1D1BH SYM | CHAR       | 171FH SYM | FORMERR      |           |           |
| 1C3AH SYM | TYPE      | 1CEFH SYM | TABS       | 1093H SYM | MATCH      | 1726H SYM | SETUPDEST    |           |           |
| 1C3BH SYM | FLEN      | 1CF0H SYM | UPPER      | 1D1CH SYM | B          | 1771H SYM | SETUPSOURCE  |           |           |
| 0754H SYM | MON1      | 1CF1H SYM | VERIF      | 1D1DH SYM | C          | 17B7H SYM | CHECKSTRINGS |           |           |
| 1C3CH SYM | F         | 1CF5H SYM | ZEROP      | 1D1EH SYM | DISK       | 17D6H SYM | CLOSEDEST    |           |           |
| 1C3DH SYM | A         | 0076H SYM | LIFTHAD    | 1D0BH SYM | GNC        | 102FH SYM | SIZENRUF     |           |           |
| 075CH SYM | L1        | 007FH SYM | SETDMA     | 10F7H SYM | DEBLANK    | 1043H SYM | SETDBLEN     |           |           |
| 0760H SYM | MON2      | 1CF6H SYM | A          | 1106H SYM | SCAN       | 106DH SYM | SIZEMEMORY   |           |           |
| 1C3FH SYM | F         | 006FH SYM | INTIN      | 1D1FH SYM | FCBA       | 109DH SYM | COPYCHAR     |           |           |
| 1C40H SYM | A         | 1CF8H SYM | ZEROSUP    | 1D21H SYM | I          | 1D46H SYM | RESIZED      |           |           |
| 0768H SYM | L2        | 1CF9H SYM | C3         | 1D22H SYM | J          | 10E4H SYM | SIMPLECOPY   |           |           |
| 076EH SYM | READRDR   | 1CFAH SYM | C2         | 1D23H SYM | K          | 1D47H SYM | FASTCOPY     |           |           |
| 0777H SYM | READCHAR  | 1CFBH SYM | C1         | 1320H SYM | DELIMITER  | 1D48H SYM | I            |           |           |
| 0760H SYM | PRINTCHAR | 00A9H SYM | ERROR      | 1D24H SYM | C          | 1973H SYM | REALEOF      |           |           |
| 1C42H SYM | CHAR      | 1CFCH SYM | A          | 1D25H SYM | I          | 1901H SYM | MULTCOPY     |           |           |
| 0790H SYM | CRLF      | 1CFEH SYM | I          | 023AH SYM | DEL        | 1D49H SYM | NEXTDIR      |           |           |
| 079BH SYM | PRINT     | 0901H SYM | MOVE       | 134FH SYM | PUTCHAR    | 1D4AH SYM | NCOPIED      |           |           |
| 1C43H SYM | A         | 1CFFH SYM | S          | 136FH SYM | FILLQ      | 1A0FH SYM | PRNAME       |           |           |
|           |           | 1D01H SYM | D          | 1D26H SYM | LEN        | 1D4BH SYM | I            |           |           |