

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.
2 DO,
3 /* PERIPHERAL INTERCHANGE PROGRAM
4
5 COPYRIGHT (C) DIGITAL RESEARCH
6 NOVEMBER, 1976
7 */
8
9 1 DECLARE COPYRIGHT(0) BYTE DATA (
10   ' COPYRIGHT (C) 1976, DIGITAL RESEARCH. PIP VERS 1.5')
11
12 1 DECLARE INPLOC ADDRESS DATA (103H); /* ADDRESS OF INP. DEVICE */
13 1 DECLARE OUTLOC ADDRESS DATA (106H); /* ADDRESS OF OUT. DEVICE */
14
15 1 OUT, PROCEDURE(B);
16 2   DECLARE B BYTE;
17 2   /* SEND B TO OUT. DEVICE */
18 2   CALL OUTLOC;
19 2 END OUT;
20
21 1 INP, PROCEDURE BYTE;
22 2   CALL INPLOC;
23 2   /* PATCHED TO RETURN REG-A */
24 2   RETURN 0;
25 2 END INP;
26
27 1 TIMEOUT, PROCEDURE;
28 2   /* WAIT FOR 50 MSEC */
29 2   CALL TIME(250); CALL TIME(250);
30 2 END TIMEOUT;
31
32 /* LITERAL DECLARATIONS */
33 1 DECLARE
34   LIT LITERALLY 'LITERALLY',
35   ENDFILE LIT '1AH',
36   TAB LIT '9H',
37   LA LIT '5FH',
38   LB LIT '5BH', /* LEFT BRACKET */
39   RB LIT '5DH', /* RIGHT BRACKET */
40   XOFF LIT '13H', /* TRANSMIT BUFFER FUNCTION */
41
42   RDR LIT '5',
43   LST LIT '10',
44   PUMP LIT '15',
45   CONP LIT '19',
46   HULP LIT '19',
47   EOFP LIT '20',
48   HSRDR LIT 'RDR',
49   PRHT LIT '10',
50
51   FSIZE LIT '33',
52   HSIZE LIT '8',
53   FNSIZE LIT '11',
54   MDISK LIT '1',
55   FNAM LIT '8',
56   FEXT LIT '9',
57   FEXTL LIT '3',
58   FREEL LIT '12', /* REEL NUMBER FIELD OF FCB */

```

CP/M VERSION 1.3
 COPYRIGHT © 1976
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA. 93950
 SER. # PIP 15

```

18 1 HBUFS LIT '80', /* "HEX" BUFFER SIZE */
19 1
20 1 ERR LIT '0',
21 1   SPECL LIT '1',
22 1   FILE LIT '2',
23 1   PERIPH LIT '3',
24 1   DISKNAME LIT '4')

25 1
26 1 DECLARE
27   EXIT ADDRESS INITIAL(0H), /* REBOOT ADDRESS UPON COMPLETION */
28   COLUMN BYTE, /* COLUMN COUNT FOR PRINTER TABS */
29   AMBIG BYTE, /* SET FOR AMBIGUOUS FILE REFS */
30   PARSET BYTE, /* TRUE IF PARAMETERS PRESENT */
31   FEEDBASE BYTE, /* USED TO FEED SEARCH CHARACTERS */
32   FEEDLEN BYTE, /* LENGTH OF FEED STRING */
33   MATCHLEN BYTE, /* USED IN MATCHING STRINGS */
34   QUITLEN BYTE, /* USED TO TERMINATE QUIT COMMAND */
35   NBUF BYTE, /* NUM BUFFERS-1 IN SBUFF AND DBUFF */
36   CDISK BYTE, /* CURRENT DISK */
37   BUFFER (128) BYTE AT (80H), /* DEFAULT BUFFER */
38   SEARFCB (FSIZE) BYTE AT (5CH), /* SEARCH FCB IN MULTI COPY */
39   MEMSIZE ADDRESS AT (6H), /* MEMORY SIZE */
40   SBLEN ADDRESS, /* SOURCE BUFFER LENGTH */
41   DBLEN ADDRESS, /* DEST BUFFER LENGTH */
42   SBASE ADDRESS, /* SOURCE BUFFER BASE */
43
44   /* THE VECTORS DBUFF AND SBUFF ARE DECLARED WITH DIMENSION
45   1024, BUT ACTUALLY VARY WITH THE FREE MEMORY SIZE */
46   DBUFF(1024) BYTE AT (.MEMORY), /* DESTINATION BUFFER */
47   SBUFF BASED SBASE (1024) BYTE, /* SOURCE BUFFER */
48   SDISK BYTE, /* SOURCE DISK */
49   (SCOM, DHEX) BYTE, /* SOURCE IS 'COM' FILE IF TRUE */
50   SOURCE (FSIZE) BYTE, /* DEST IS 'HEX' FILE IF TRUE */
51   DEST (FSIZE) BYTE, /* SOURCE FCB */
52   DDISK BYTE, /* DESTINATION FCB */
53   HBUFS(HBUFS) BYTE, /* DESTINATION DISK */
54   HSOURCE BYTE, /* HEX FILE BUFFER */
55
56   NSOURCE ADDRESS, /* NEXT HEX SOURCE CHARACTER */
57   HARDEOF ADDRESS, /* SET TO NSOURCE ON REAL EOF */
58   NDEST ADDRESS, /* NEXT DESTINATION CHARACTER */

59 1
60 1 DECLARE
61   PDEST BYTE, /* DESTINATION DEVICE */
62   PSOURCE BYTE, /* CURRENT SOURCE DEVICE */

63 1
64 1 DECLARE
65   MULTCOM BYTE, /* FALSE IF PROCESSING ONE LINE */
66   PUTNUM BYTE, /* SET WHEN READY FOR NEXT LINE NM */
67   CONCNT BYTE, /* COUNTER FOR CONSOLE READY CHECK */
68   CHAR BYTE, /* LAST CHARACTER SCANNED */
69   TYPE BYTE, /* TYPE OF CHARACTER SCANNED */
70   FLEN BYTE, /* FILE NAME LENGTH */

71 1   $INCLUDE(.F1.CPIO.PLB)
72 1   MOH1, PROCEDURE(F,A)
73 2   DECLARE F BYTE,
74 2   A ADDRESS,
75 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;
2 = A ADDRESS;
27 2 = L2, GO TO L2; /* PATCHED WITH JMP 0005 */
28 2 = RETURN 0;
29 2 = END MON2;
30 1 = READRDR: PROCEDURE BYTE;
2 = /* READ CURRENT READER DEVICE */;
31 2 = RETURN MON2(3,0);
32 2 = END READRDR;
33 1 = READCHAR: PROCEDURE BYTE;
2 = /* READ CONSOLE CHARACTER */;
34 2 = RETURN MON2(1,0);
35 2 = END READCHAR;
36 1 = DECLARE
2 = TRUE LITERALLY '1',
3 = FALSE LITERALLY '0',
4 = FOREVER LITERALLY 'WHILE TRUE',
5 = CR LITERALLY '13',
6 = LF LITERALLY '10',
7 = WHAT LITERALLY '63';
37 1 = PRINTCHAR: PROCEDURE(CHAR);
38 2 = DECLARE CHAR BYTE;
39 2 = CALL MON1(2,CHAR);
40 2 = END PRINTCHAR;
41 1 = CRLF: PROCEDURE;
42 2 = CALL PRINTCHAR(CR);
43 2 = CALL PRINTCHAR(LF);
44 2 = END CRLF;
45 1 = PRINT: PROCEDURE(A);
46 2 = DECLARE A ADDRESS;
4 = /* PRINT THE STRING STARTING AT ADDRESS A UNTIL THE
5 = NEXT DOLLAR SIGN IS ENCOUNTERED */;
47 2 = CALL CRLF;
48 2 = CALL MON1(9,A);
49 2 = END PRINT;
50 1 = DECLARE DCNT BYTE;
51 1 = INITIALIZE: PROCEDURE;
52 2 = CALL MON1(13,0);
53 2 = END INITIALIZE;
54 1 = SELECT: PROCEDURE(D);
55 2 = DECLARE D BYTE;
56 2 = CALL MON1(14,D);
57 2 = END SELECT;
58 1 = OPEN: PROCEDURE(FCB);
59 2 = DECLARE FCB ADDRESS;
60 2 = DCNT = MON2(15,FCB);
61 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 1 = DECLARE (TCBP,CBP) BYTE; /* TEMP CBP, COMMAND BUFFER POINTER */;
98 2 = READCOM: PROCEDURE;
99 2 = /* READ INTO COMMAND BUFFER */;
100 1 = MAXLEN = 128;
101 2 = CALL MON1(10,.MAXLEN);
102 2 = END READCOM;
103 1 = DECLARE IOBYTE BYTE AT(3H); /* INTEL IOBYTE */;
104 2 = END;

```

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 */
/* 80 81 82 83 84 85 86 87 88 89 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
0 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 */
QUIT 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 LIFTHEAD: PROCEDURE;
CALL MOH1(12,0);
END LIFTHEAD;

106 2
107 2
108 1 SETDMA: PROCEDURE(A);
DECLARE A ADDRESS;
CALL MOH1(26,A);
END SETDMA;

/* INTELIC 8 INTEL/ICOM READER INPUT */

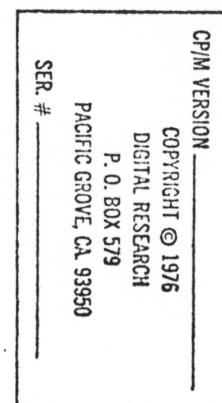
109 2 INTIH: PROCEDURE BYTE;
/* READ THE INTEL / ICOM READER */
DECLARE PTRI LITERALLY '3', /* DATA */
PTRS LITERALLY '1', /* STATUS */
PTRC LITERALLY '1', /* COMMAND */

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

110 2 /* STROBE THE READER */
OUTPUT(PTRC) = PTRG;
OUTPUT(PTRC) = PTRH;
DO WHILE NOT RD1(INPUT(PTRS),3); /* NOT READY */
END;
/* DATA READY */
111 2 RETURN INPUT(PTRI) AND 7FH;
END INTIH;

112 1
113 2
114 2
115 2
116 2
117 3
118 2
119 2
120 1
121 1
122 2
123 2
124 2
125 2
126 2
127 3
128 1
129 3

```



```

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

MOVE: PROCEDURE(S,D,N);
DECLARE (S,D) ADDRESS, N BYTE;
DECLARE A BASED S BYTE, B BASED D BYTE;
DO WHILE (N=N-1) <> 255;
B = A; S = S+1; D = D+1;
END;
END MOVE;

144 1 FILLSOURCE: PROCEDURE;
/* FILL THE SOURCE BUFFERS */
DECLARE (I,J) BYTE;
NSOURCE = 0;
CALL SELECT(SDISK);
DO I = 0 TO NBUF;
/* SET DMA ADDRESS TO NEXT BUFFER POSITION */
CALL SETDMA(.SBUFF(NSOURCE));
IF (J := DISKREAD(SOURCE)) <> 0 THEN
DO; IF J <> 1 THEN
CALL ERROR(.('DISK READ ERROR$'));
/* END - OF - FILE */
HARDEOF = NSOURCE; /* SET HARD END-OF-FILE */
SBUFF(NSOURCE) = ENDFILE; I = NBUF;
END; ELSE
NSOURCE = NSOURCE + 128;
END;
NSOURCE = 0;
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 */
DECLARE (I, J, N) BYTE;
DECLARE DMA ADDRESS;
DECLARE (EXTCHT, DATAOK) BYTE;
IF (N := LOW(SHR(NDEST,7)) - 1) = 255 THEN RETURN ;
EXTCHT, NDEST = 0;
CALL SELECT(DDISK);
DO I = 0 TO N;
/* SET DMA ADDRESS TO NEXT BUFFER */
DMA = .DBUFF(NDEST);
IF VERIF THEN /* VERIFY MODE */
DO;
IF DEST(32) = 127 THEN /* END OF EXTENT */
DO; CALL MOVE(DMA,80H,80H);
DMA = 80H; EXTCHT = EXTCHT + 1;
END;
CALL SETDMA(DMA);
IF DISKWRITE(.DEST) <> 0 THEN
CALL ERROR(.('DISK WRITE ERROR$'));
NDEST = NDEST + 128;

```

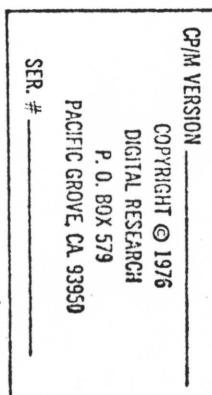
CP/M VERSION _____
 COPYRIGHT © 1976
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950
 SER. # _____

```

185 3      END;
186 2      IF VERIF THEN /* VERIFY DATA WRITTEN OK */
187 2          DOJ I = N + 1; /* NUMBER OF BUFFERS */
188 3          HDEST = 0;
189 3          IF EXTCNT < 0 THEN /* WENT OVER CURRENT EXTENT */
190 3              DO; CALL SETDMA(80H); CALL CLOSE(. DEST);
191 4                  DEST(FREEL) = DEST(FREEL) - EXTCNT;
192 4                  J = DEST(32); /* SAVE CURRENT RECORD NUMBER */
193 4                  CALL OPEN(. DEST);
194 4                  /* ONE OR TWO EXTENTS WHERE WRITTEN */
195 4                  I = I - 128;
196 4                  /* I MAY BE (TWO'S COMPLEMENT) NEGATIVE */
197 4                  DEST(32) = (J - I) AND 7FH;
198 4                  EHD; ELSE
199 4                      DEST(32) = DEST(32) - I; /* NEXT TO READ */
200 3                      CALL SETDMA(80H); /* FOR COMPARE */
201 3                      DO I = 0 TO N;
202 3                          DATAOK = DISKREAD(. DEST) = 0;
203 4                      J = 0;
204 4                      /* PERFORM COMPARISON */
205 4                      DO WHILE DATAOK AND J < 80H;
206 3                          DATAOK = BUFFER(J) = DBUFF(HDEST+J);
207 5                          J = J + 1;
208 5                          EHD;
209 4                          HDEST = HDEST + 128;
210 4                          IF NOT DATAOK THEN
211 4                              CALL ERROR(. ('VERIFY ERRORS'));
212 4                          END;
213 3                          IF DEST(32) = 128 THEN /* INTO NEXT EXTENT */
214 3                              DO; DEST(FREEL) = DEST(FREEL) + 1;
215 4                              CALL OPEN(. DEST);
216 4                          EHD;
217 4
218 3
219 2      END;
220 2      HDEST = 0;
221 2      EHD WRITEDEST;

221 1      PUTDESTC, PROCEDURE(B);
222 2      DECLARE (B,105) BYTE;
223 2      /* WRITE BYTE B TO THE DESTINATION DEVICE GIVEN BY PDEST */
224 2      IF B > ' ' THEN
225 2          DO; COLUMN = COLUMN + 1;
226 3          IF DELET > 0 THEN /* MAY BE PAST RIGHT SIDE */
227 3              DO; IF COLUMN > DELET THEN RETURN;
228 4              END;
229 3
230 2      END;
231 2      IOB = IOBYTE; /* IN CASE IT IS ALTERED */
232 2      DO CASE PDEST;
233 2          /* CASE 0 IS THE DESTINATION FILE */
234 3              DO;
235 4                  IF HDEST >= DBLEN THEN CALL WRITEDEST;
236 4                  DBUFF(HDEST) = B;
237 4                  HDEST = HDEST+1;
238 4                  END;
239 4
240 3          /* CASE 1 IS ARD (ADDMASTER) */
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 */

```



```

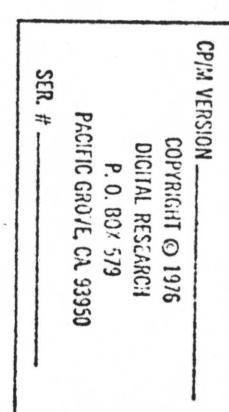
243 3          GO TO NOTDEST;
244 3          /* CASE 5 IS UR2 */
245 3          GO TO NOTDEST;
246 3          /* CASE 6 IS RDR */
247 3          NOTDEST;
248 3          CALL ERROR(. ('NOT A CHARACTER SINK$'));
249 3
250 4          /* CASE 7 IS OUT */
251 3          CALL OUT(B);
252 4          /* CASE 8 IS LPT */
253 3          DO; IOBYTE = 1000$0000B; GO TO LSTL;
254 4          END;
255 3          /* CASE 9 IS UL1 */
256 4          DO; IOBYTE = 1100$0000B; GO TO LSTL;
257 4          END;
258 3          /* CASE 10 IS PRN (TABS EXPANDED, LINES LISTED, CHANGED TO LST) */
259 4          DO; IOBYTE = 1000$0000B; GO TO LSTL;
260 4          END;
261 3          /* CASE 11 IS LST */
262 4          LSTL;
263 3          CALL MDN1(5,B);
264 3          /* CASE 12 IS PTP */
265 4          DO; IOBYTE = 0001$0000B; GO TO PUHL;
266 4          END;
267 3          /* CASE 13 IS UPI */
268 4          DO; IOBYTE = 0010$0000B; GO TO PUHL;
269 4          END;
270 3          /* CASE 14 IS UP2 */
271 4          DO; IOBYTE = 0011$0000B; GO TO PUHL;
272 4          END;
273 3          /* CASE 15 IS PUN */
274 4          PUHL;
275 3          CALL MDN1(4,B);
276 4          /* CASE 16 IS TTY */
277 3          DO; IOBYTE = B; GO TO CONL;
278 4          END;
279 3          /* CASE 17 IS CRT */
280 4          DO; IOBYTE = 1; GO TO CONL;
281 3          END;
282 4          /* CASE 18 IS UC1 */
283 4          DO; IOBYTE = 11B; GO TO CONL;
284 4          END;
285 3          /* CASE 19 IS CON */
286 4          CONL;
287 2          CALL MDN1(2,B);
288 2          END;
289 1          IOBYTE = IOB;
290 2          END PUTDESTC;

291 1      PRINT1, PROCEDURE(B);
292 2      DECLARE B BYTE;
293 2      IF (ZEROSUP != ZEROSUP AND B = 0) THEN CALL PUTDESTC(' ');
294 2      ELSE
295 1          CALL PUTDESTC('0'+B);
296 2
297 2      END PRINT1;

298 1      PRINTDIG, PROCEDURE(D);
299 2      DECLARE D BYTE;
299 2      CALL PRINT1(SHR(D,4)); CALL PRINT1(D AND 1111B);
299 2      END PRINTDIG;

300 1      NEWLINE, PROCEDURE;

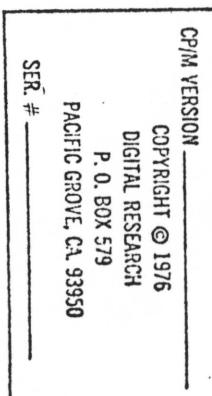
```



```

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      PUTHUM = FALSE;
306 2      END NEWLINE;
307 2
311 1      CLEARBUFF: PROCEDURE;
312 2          /* CLEAR OUTPUT BUFFER IN BLOCK MODE TRANSMISSION */
313 2          DECLARE HA ADDRESS;
314 2          DECLARE I BYTE;
315 2          I = LOW(HDEST) AND 7FH; /* REMAINING PARTIAL BUFFER LENGTH */
316 2          HA = HDEST AND OFF80H; /* START OF SEGMENT NOT WRITTEN */
317 2          CALL WRITEDEST; /* CLEARS BUFFERS */
318 2          CALL MOVE(.DBUFF(HA),.DBUFF,I);
319 2          /* DATA MOVED TO BEGINNING OF BUFFER */
320 1          HDEST = I;
321 2          END CLEARBUFF;
322 1      PUTDEST: PROCEDURE(B);
323 2          DECLARE (I,B) BYTE;
324 2          /* WRITE DESTINATION CHARACTER, CHECK TABS AND LINES */
325 2          IF HUMB AND PUTHUM THEN CALL NEWLINE;
326 2          IF BLOCK THEN /* BLOCK MODE TRANSFER */
327 3              DO;
328 3                  IF B = XOFF AND PIEST = 0 THEN
329 4                      DO; CALL CLEARBUFF; /* BUFFERS WRITTEN */
330 4                      RETURN; /* DON'T PASS THE X-OFF */
331 3                  END;
332 2          IF B = TAB THEN /* EXPAND TO NEXT TAB POSITION */
333 3              DO; IF TABS > 0 THEN
334 3                  DO; I = COLUMN;
335 3                  IF HUMB THEN I = I - 7; /* STARTING BIAS ON LINE */
336 4                  DO WHILE I > TABS;
337 5                      I = I - TABS;
338 5                  END;
339 4                  I = TABS - I;
340 5                  DO WHILE I > 0;
341 5                      I = I - 1; CALL PUTDESTC(' ');
342 5                  END;
343 4
344 5
345 5
346 5
347 4
348 3      ELSE CALL PUTDESTC(B);
349 3
350 2
351 3      CALL PUTDESTC(B);
352 3      IF B = CR THEN COLUMN = 0; ELSE
353 2          /* MAY NEED NEWLINE NEXT TIME AROUND */
354 3          PUTHUM = B = LF;
355 3
356 2      END PUTDEST;
357 1
358 2      UTRAN: PROCEDURE(B) BYTE;
359 2          DECLARE B BYTE;
360 2          /* TRANSLATE ALPHA TO UPPER CASE */
361 2          IF B >= 110$0001B AND B <= 111$1010B THEN /* LOWER CASE */
362 2              B = B AND 101$1111B; /* TO UPPER CASE */
363 2

```



```

363 1      LTRAN: PROCEDURE(B) BYTE;
364 2          DECLARE B BYTE;
365 2          /* TRANSLATE TO LOWER CASE ALPHA */
366 2          IF B >= 'A' AND B <= 'Z' THEN B = B OR 10$0000B; /* TO LOWER */
367 2
368 2          END LTRAN;
369 1
370 2      GETSOURCEC: PROCEDURE BYTE;
371 2          /* READ NEXT SOURCE CHARACTER */
372 2          DECLARE (IOB,B,CONCHK) BYTE;
373 2
374 3
375 4
376 4
377 4
378 4
379 4
380 3
381 2
382 2
383 2
384 3
385 4
386 4
387 4
388 4
389 4
390 3
391 3
392 3
393 4
394 3
395 4
396 3
397 4
398 3
399 4
400 3
401 4
402 3
403 4
404 3
405 3
406 3
407 3
408 3
409 3
410 3
411 3
412 3

```

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

```

413   3 /* CASE 15 IS PUN */
414     1   HOTSOURCE:
415       2     DO; CALL ERROR(.('NOT A CHARACTER SOURCE$'));
416       3     EHD;
417     4 /* CASE 16 IS TTY */
418       3     DO; IOBYTE = 0; GO TO COHL;
419       4     EHD;
420     3 /* CASE 17 IS CRT */
421       3     DO; IOBYTE = 01B; GO TO COHL;
422       4     EHD;
423     3 /* CASE 18 IS UC1 */
424       3     DO; IOBYTE = 11B; GO TO COHL;
425       4     EHD;
426     3 /* CASE 19 IS CON */
427       3     COHL:
428       4       DO; CONCHK = FALSE; /* DON'T CHECK CONSOLE STATUS */
429         5         B = MOH2(1,B);
430       4       EHD;
431     3 /* OF CASES */
432       2     IOBYTE = IOB; /* RESTORE IOBYTE */
433     2 IF ECHO THEN /* COPY TO CONSOLE DEVICE */
434       3       DO; IOB = PDEST; PDEST = CONP; CALL PUTDEST(B);
435       3       PDEST = IOB;
436       4     END;
437     2 IF CONCHK THEN /* TEST FOR CONSOLE CHAR READY */
438       3     DO;
439       3       IF SCOM THEN /* SOURCE IS A COM FILE */
440         4         CONCHK = (CONCHT := CONCHT + 1) = 0; ELSE /* ASCII */
441           5           CONCHK = B = LF;
442       3       IF CONCHK THEN
443         4         DO; IF CONBRK THEN
444           5           DO;
445             5             IF READCHAR = ENDFILE THEN RETURN ENDFILE;
446             5             CALL ERROR(.('ABORTED$'));
447             5           END;
448         4       END;
449       3     END;
450       2     IF ZEP0P THEN B = B AND 7FH;
451       2     IF UPPER THEN RETURN UTRAN(B);
452       2     IF LOWER THEN RETURN LTRAN(B);
453       2     RETURN B;
454     2 END GETSOURCEC;

455     3 GETSOURCE: PROCEDURE BYTE;
456       2   /* GET NEXT SOURCE CHARACTER */
457     2     DECLARE CHAR BYTE;
458     2     MATCH: PROCEDURE(B) BYTE;
459       3       /* MATCH START AND QUIT STRINGS */
460     3       DECLARE (B,C) BYTE;
461       3       IF (C:=COMBUFF(B,=(B+MATCHLEN))) = ENDFILE THEN /* END MATCH */
462         4         DO; COMBUFF(B) = CHAR; /* SAVE CURRENT CHARACTER */
463           5           RETURN TRUE;
464         4       END;
465       3       IF C = CHAR THEN MATCHLEN = MATCHLEN + 1; ELSE
466         4         MATCHLEN = 0; /* NO MATCH */
467       3       RETURN FALSE;
468       4     END MATCH;

469     2 IF QUITLEN > 0 THEN
470       3     DO; IF (QUITLEN := QUITLEN - 1) = 1 THEN RETURN LF;
471       3     RETURN ENDFILE; /* TERMINATED WITH CR,LF,ENDFILE */

```

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

SER. # _____

```

493    3
494    2
495    3
496    2
497    4
498    4
499    4
500    5
501    5
502    4
503    3
504    3
505    4
506    4
507    5
508    5
509    5
510    5
511    4
512    4
513    3
514    3
515    2
516    1
517    1
518    2
519    2
520    2
521    2
522    1
523    2
524    3
525    2
526    1
527    2
528    2
529    2
530    3
531    3
532    3
533    4
534    3
535    4
536    3
537    3
538    2
539    3
540    3

      END;
      DO FOREVER; /* LOOKING FOR START */
      IF FEEDLEN > 0 THEN /* GET SEARCH CHARACTERS */
        DO; FEEDLEN = FEEDLEN - 1;
        CHAR = COMBUFF(FEEDBASE);
        FEEDBASE = FEEDBASE + 1;
        RETURN CHAR;
      END;
      IF (CHAR := GETSOURCEC) = ENDFILE THEN RETURN ENDFILE;
      IF STARTS > 0 THEN /* LOOKING FOR START STRING */
        DO; IF MATCH(STARTS) THEN
          DO; FEEDBASE = STARTS; STARTS = 0;
          FEEDLEN = MATCHLEN + 1;
        END; /* OTHERWISE NO MATCH, SKIP CHARACTER */
      END; ELSE
        IF QUIT > 0 THEN /* PASS CHARACTERS TIL MATCH */
          DO; IF MATCH(QUIT) THEN
            DO; QUIT = 0; QUITLEN = 2;
            /* SUBSEQUENTLY RETURN CR, LF, ENDFILE */
            RETURN CR;
          END;
          RETURN CHAR;
        END; ELSE
        RETURN CHAR;
      END; /* OF DO FOREVER */
      END GETSOURCE;

      DECLARE DISK BYTE; /* SELECTED DISK */

      GNC: PROCEDURE BYTE;
      IF (CBP := CBP + 1) >= COMLEN THEN RETURN CR;
      RETURN UTRAN(COMBUFF(CBP));
      END GNC;

      DEBLANK: PROCEDURE;
      DO WHILE (CHAR := GNC) = ' ';
      END;
      END DEBLANK;

      SCAN: PROCEDURE(FCBA);
      DECLARE FCBA ADDRESS, /* ADDRESS OF FCB TO FILL */
      FCB BASED FCBA (FSIZE) BYTE; /* FCB TEMPLATE */
      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 */

      DELIMITER: PROCEDURE(C) BYTE;
      DECLARE (I,C) BYTE;
      DECLARE DEL(*) BYTE DATA
      (' =,,,<>',CR,LA,LB,RB);
      DO I = 0 TO LAST(DEL);
      IF C = DEL(I) THEN RETURN TRUE;
      END;
      RETURN FALSE;
      END DELIMITER;

      PUTCHAR: PROCEDURE;
      FCB(FLEN.=FLEN+1) = CHAR;
      IF CHAR = WHAT THEN AMBIG = TRUE; /* CONTAINS AMBIGUOUS REF */

```

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

```

542 3     END PUTCHAR;

543 2     FILLO: PROCEDURE(LEN);
544 3         /* FILL CURRENT NAME OR TYPE WITH QUESTION MARKS */
545 3         DECLARE LEN BYTE;
546 3         CHAR = WHAT; /* QUESTION MARK */
547 4         DO WHILE FLEN < LEN;
548 4             CALL PUTCHAR;
549 3         END;
549 3     END FILLO;

550 2     GETFCB: PROCEDURE(I) BYTE;
551 3         DECLARE I BYTE;
552 3         RETURN FCB(I);
553 3     END GETFCB;

554 2     SCAHPAR: PROCEDURE;
555 3         DECLARE (I,J) BYTE;
556 3         /* SCAH OPTIONAL PARAMETERS */
557 3         PARSET = TRUE;
558 3         CHAR = GNC; /* SCAN PAST BRACKET */
559 4             DO WHILE NOT(CHAR = CR OR CHAR = RB);
560 4             IF (I .= CHAR - 'A') > 25 THEN /* NOT ALPHA */
561 4                 DO; IF CHAR = ' ' THEN CHAR = GNC; ELSE
562 5                     CALL ERROR(.('BAD PARAMETER$'));
563 5                 END; ELSE
564 5                 DO; /* SCAN PARAMETER VALUE */
565 5                 IF CHAR = 'S' OR CHAR = 'Q' THEN
566 5                     DO; /* START OR QUIT COMMAND */
567 5                     J = CBP + 1; /* START OF STRING */
568 6                         DO WHILE NOT ((CHAR .= GNC) = ENDFILE OR CHAR = CR);
569 6                         END;
570 7                         CHAR=GNC;
571 6                     END; ELSE
572 6                     IF (J .= (CHAR .= GNC) - 'B') > 9 THEN J = 1;
573 5                 ELSE
574 5                     DO WHILE (K .= (CHAR .= GNC) - 'B') <= 9;
575 5                     J = J * 10 + K;
576 6                 END;
577 6                 CONT(I) = J;
578 5             END;
579 5         END;
580 4     END;
581 3         CHAR = GNC;
582 3     END SCAHPAR;

583 2     CHKSET: PROCEDURE;
584 3         IF CHAR = LA THEN CHAR = '"';
585 3     END CHKSET;

587 2     /* INITIALIZE FILE CONTROL BLOCK TO EMPTY */
588 2     AMBIG = FALSE; TYPE = ERR; CHAR = ' ' ; FLEN = 0;
589 2         DO WHILE FLEN < FSIZE-1;
590 3         IF FLEN = FNSIZE THEN CHAR = B;
591 3         CALL PUTCHAR;
592 3         END;

593 2     /* DEBLANK COMMAND BUFFER */
594 3         CALL DEBLANK;
595 3     END;

596 2     /* SAVE STARTING POSITION OF SCAH FOR DIAGNOSTICS */
597 2

```

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

```

597 2     TCBP = CBP;

598 2     /* MAY BE A SEPARATOR */
599 2     IF DELIMITER(CHAR) THEN
600 2         DO; CALL CHKSET;
601 3         TYPE = SPEC1; RETURN;
602 3     END;

604 2     /* CHECK PERIPHERALS AND DISK FILES */
605 2     DISK = 0;
606 2     /* CLEAR PARAMETERS */
607 3     DO I = 0 TO 25; CONT(I) = 0;
608 2     END;

609 2     PARSET = FALSE;
610 2     FEEDLEN, MATCHLEN, QUITLEN = 0;
611 2     /* SCAN NEXT NAME */
612 2         DO FOREVER;
613 3         FLEN = 0;
614 4             DO WHILE NOT DELIMITER(CHAR);
615 4             IF FLEN >= NSIZE THEN /* ERROR, FILE NAME TOO LONG */
616 4                 RETURN;
617 4             IF CHAR = '*' THEN CALL FILLO(NSIZE); ELSE CALL PUTCHAR;
618 2             CHAR = GNC;
619 4         END;

620 2     /* CHECK FOR DISK NAME OR DEVICE NAME */
621 2     IF CHAR = ',' THEN
622 2         DO; IF DISK < 0 THEN RETURN; /* ALREADY SET */
623 2         IF FLEN = 1 THEN
624 3             /* MAY BE DISK NAME A ... Z */
625 4             DO;
626 5             IF (DISK := GETFCB(I) - 'A' + 1) > 26 THEN
627 5                 /* ERROR, INVALID DISK NAME */ RETURN;
628 5             CALL DEBLANK; /* MAY BE DISK NAME ONLY */
629 5             IF DELIMITER(CHAR) THEN
630 5                 DO; IF CHAR = LB THEN
631 5                     CALL SCAHPAR;
632 6                 CBP = CBP - 1;
633 6                 TYPE = DISKNAME;
634 6                 RETURN;
635 6             END;
636 6         END; ELSE
637 5             /* MAY BE A THREE CHARACTER DEVICE NAME */
638 4             IF FLEN < 3 THEN /* ERROR, CANNOT BE DEVICE NAME */
639 4                 RETURN; ELSE
640 4                 /* LOOK FOR DEVICE NAME */
641 4                 DO; DECLARE (I,J,K) BYTE, M LITERALLY '20';
642 5                 IO(*) BYTE DATA
643 5                 ('INPIRDPTRUR1UR2RDROUTLPTUL1PRHLST',
644 6                 'PTPUPU1P2PUNTYCRTUC1CONHULEOF',0);
645 6                 /* NOTE THAT ALL READER-LIKE DEVICES MUST BE
646 6                 PLACED BEFORE 'RDR', AND ALL LISTING-LIKE DEVICES
647 6                 MUST APPEAR BELOW LST, BUT ABOVE RDR. THE LITERAL
648 6                 DECLARATIONS FOR RDR, LST, AND PUMP MUST INDICATE
649 6                 THE POSITIONS OF THESE DEVICES IN THE LIST */
650 6                 J = 255;
651 6                 DO K = 0 TO M;
652 6                 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
646   7              IO(J+I) = GETFCB(I)
647   6          END;
648   6          IF I = 4 THEN /* COMPLETE MATCH */
649   6              DO; TYPE = PERIPH;
650   7                  /* SCAN PARAMETERS */
651   7                  IF GNC = LB THEN CALL SCANPAR;
652   7                  CBP = CBP - 1; CHAR = K;
653   7                  RETURN;
654   7                  END;
655   7
656   6          /* OTHERWISE TRY NEXT DEVICE */ J = J + 3;
657   6          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 = FHAM;
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 FILLO(FNSIZE);
672   5                      ELSE CALL PUTCHAR;
673   5                  END;

674   5
675   4          IF CHAR = LB THEN
676   4              CALL SCANPAR;
677   4          /* RESCAN DELIMITER NEXT TIME AROUND */
678   4              CBP = CBP - 1;
679   4              TYPE = FILE;
680   4              /* DISK IS THE SELECTED DISK (1 2 3 ... ) */
681   4              IF DISK = 0 THEN DISK = CDISK + 1; /* DEFAULT */
682   4              FCB(0),FCB(32) = 0;
683   4              RETURN;
684   3          END;
685   2          END SCAN;

686   1          HULLS, PROCEDURE;
687   2              /* SEND 40 HULLS TO OUTPUT DEVICE */
688   2              DECLARE I BYTE;
689   2                  DO I = 0 TO 39; CALL PUTDEST(B);
690   3                  END;
691   2          END HULLS;

692   1          DECLARE FEXT(FEXTL) BYTE;      /* HOLDS DESTINATION FILE TYPE */
693   1          COPYBYTE;                /* TRUE WHILE COPYING TO DEST FILE */

694   1          MOVEXT, PROCEDURE(A);
695   2              DECLARE A ADDRESS;
696   2              /* MOVE THREE CHARACTER EXTENT INTO DEST FCB */
697   2              CALL MOVE(A,.DEST(FEXT),FEXTL);
698   2          END MOVEXT;

699   2          EQUAL, PROCEDURE(A,B) BYTE;
700   3              /* COMPARE THE STRINGS AT A AND B UNTIL EITHER A MISMATCH OR
701   3              A '$' IS ENCOUNTERED IN STRING B */
702   3              DECLARE (A,B) ADDRESS;
703   3                  (SA BASED A, SB BASED B) BYTE;
704   3                  DO WHILE SB <> '$';
705   3                  IF SB <> SA THEN RETURN FALSE;
706   3                  A = A + 1; B = B + 1;
707   3                  END;
708   3          RETURN TRUE;
709   2          END EQUAL;

710   1          READ$EOF, PROCEDURE BYTE;
711   2              /* RETURN TRUE IF END OF FILE */
712   2                  CHAR = GETSOURCE;
713   2                  IF SCOM THEN RETURN HARDEOF (= NSOURCE);
714   2                  RETURN CHAR = ENDFILE;
715   2          END READ$EOF;

716   1          HEKRECORD, PROCEDURE BYTE;
717   2              /* READ ONE RECORD INTO SBUFF AND CHECK FOR PROPER FORM
718   2                  RETURNS 0 IF RECORD OK
719   2                  RETURNS 1 IF END OF TAPE (.00000)
720   2                  RETURNS 2 IF ERROR IN RECORD */
721   2
722   2
723   2
724   2          DECLARE XOFFSET BYTE; /* TRUE IF XOFF RECVD */
725   3          DECLARE NOERRS BYTE; /* TRUE IF NO ERRORS IN THIS RECORD */
726   3
727   3
728   4
729   4
730   3
731   2
732   3
733   3
734   3
735   4
736   4
737   5
738   4
739   4
740   4
741   4
742   4
743   3

```

CP/A VERSION _____

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

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,
746 2       LDA ADDRESS; /* LOAD ADDRESS WHICH FOLLOWS : */
747 3       READHEX, PROCEDURE BYTE;
748 3         DECLARE H BYTE;
749 3         IF (H := SAVECHAR) = '0' <= 9 THEN RETURN H-'0';
750 3         IF H = 'A' > 5 THEN
751 3           CALL PRINTERR(.('INVALID DIGITS'));
752 3           RETURN H - 'A' + 10;
753 3         END READHEX;

754 2     READBYTE, PROCEDURE BYTE;
755 3       /* READ TWO HEX DIGITS */
756 3       RETURN SHL(READHEX,4) OR READHEX;
757 3     END READBYTE;

757 2     READCS, PROCEDURE BYTE;
758 3       /* READ BYTE WITH CHECKSUM */
759 3       RETURN CS := CS + READBYTE;
759 3     END READCS;

760 2     READADDR, PROCEDURE ADDRESS;
761 3       /* READ DOUBLE BYTE WITH CHECKSUM */
762 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 3     CALL CHECKOFF;
773 3   END;

/* ',' FOUND */
774 2     CS = 0;
775 2     IF (RL := READCS) = 0 THEN /* END OF TAPE */
776 2       DO; DO WHILE (RL := SAVECHAR) <> ENDFILE;
777 2         CALL CHECKOFF;
778 2       END;
779 3     IF NOERRS THEN RETURN 1;
780 3     RETURN 2;
781 3   END;

/* RECORD LENGTH IS NOT ZERO */
782 2     LDA = READADDR; /* LOAD ADDRESS */

/* READ WORDS UNTIL RECORD LENGTH EXHAUSTED */
783 2     RT = READCS; /* RECORD TYPE */
784 2     DO WHILE RL <> 0 AND NOERRS; RL = RL - 1;
785 3     M = READCS;

```

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

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

```

791 3     /* INCREMENT LA HERE FOR EXACT ADDRESS */
792 2     END;

/* CHECK SUM */
793 2     IF CS + READBYTE <> 0 THEN
794 2       CALL PRINTERR(.('CHECKSUM ERROR'));

CALL CHECKXOFF;
795 2     IF NOERRS THEN RETURN 0;
796 2     RETURN 2;
797 2     END HEXRECORD;

799 1     READTAPE, PROCEDURE;
799 1     /* READ HEX FILE FROM HIGH SPEED READER TO 'HEX' FILE,
799 1     CHECK EACH RECORD FOR VALID DIGITS, AND PROPER CHECKSUM */
799 1     DECLARE (I,A) BYTE;
799 1     DO FOREVER;
799 1       DO WHILE (I := HEXRECORD) <= 1;
799 1         IF NOT (I = 1 AND IGNOR) THEN
799 1           DO A = 1 TO HSOURCE;
799 1             CALL PUTDEST(HBUFF(A-1));
799 1           END;
799 1           CALL PUTDEST(CR); CALL PUTDEST(LF);
799 1           IF I = 1 THEN /* END OF TAPE ENCOUNTERED */
799 1             RETURN;
799 1           END;
799 1           CALL CRLF; HBUFF(HSOURCE) = '$';
799 1           CALL PRINT(.HBUFF);
799 1           CALL PRINT(.('CORRECT ERROR, TYPE RETURN OR CTL-Z'));
799 1           CALL CRLF;
799 1           IF READCHAR = ENDFILE THEN RETURN;
799 1         END;
799 1       END READTAPE;

800 2     FORMERR, PROCEDURE;
801 2       CALL ERROR(.('INVALID FORMAT'));
802 2     END FORMERR;

803 1     SETUPDEST, PROCEDURE;
804 2       CALL SELECT(DDISK);
805 2       DHEX = EQUAL(.DEST(FEXT), .('HEX$'));
806 2       CALL MOVE(.DEST(FEXT), FEXTH, FEXTL); /* SAVE TYPE */
807 2       CALL MOVEXT(.('$$$'));
808 2       CALL DELETE(.DEST); /* REMOVE OLD $$$ FILE */
809 2       CALL MAKE(.DEST); /* CREATE A NEW ONE */
810 2       IF DCNT = 255 THEN CALL ERROR(.('NO DIRECTORY SPACES'));
811 2       DEST(32), NDEST = 0;
812 2     END SETUPDEST;

813 1     SETUPSOURCE, PROCEDURE;
814 2       HARDEOF = 6FFFFH;
815 2       CALL SELECT(SDISK);
816 2       CALL OPEN(.SOURCE);
817 2       IF DCNT = 255 THEN CALL ERROR(.('NO FILE$'));
818 2       SOURCE(32) = 0;
819 2       /* CAUSE IMMEDIATE READ */
820 2       SCOM = EQUAL(.SOURCE(FEXT), .('COM$')) OR
821 2         EQUAL(.SOURCE(FEXT), .('CHI$'));
822 2       NSOURCE = SBLEN;
823 2     END SETUPSOURCE;

```

```

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

851 1 CLOSEDEST; PROCEDURE;
852 2   /* CLEAR BUFFER */
853 2   DO WHILE (LOW(HDEST) AND 7FH) <> 0;
854 3     CALL PUTDEST(ENDIFFILE);
855 2   END;
856 2   CALL CHECK$STRINGS;
857 2   CALL WRITEDEST;
858 2   CALL SELECT(SDISK);
859 2   CALL CLOSE(. DEST);
860 2   IF DCNT = 255 THEN CALL ERROR(.('WRITE PROTECTED',WHAT,'$'));
861 2   CALL MOVEXT(. FEXTH); /* RECALL ORIGINAL TYPE */
862 2   CALL DELETE(. DEST); /* REMOVE OLD FILE */
863 2   CALL MOVE(. DEST,. DEST(16),16); /* READY FOR RENAME */
864 2   CALL MOVEXT(.('$$'));
865 2   CALL RENAME(. DEST);
866 2 END CLOSEDEST;

867 1 SIZE$HBUF; PROCEDURE;
868 2   /* COMPUTE NUMBER OF BUFFERS - 1 FROM DBLEN */
869 2   HBUF = (SHR(DBLEN,7) AND 0FFH) - 1;
870 2   /* COMPUTED AS DBLEN/128-1, WHERE DBLEN <= 32K (AND THUS
871 2   HBUF RESULTS IN A VALUE <= 2**15/2**7-1 = 2**8-1 = 255) */
872 2 END SIZE$HBUF;

873 1 SET$DBLEN; PROCEDURE;
874 2   /* ABSORB THE SOURCE BUFFER INTO THE DEST BUFFER */
875 2   SBASE = .MEMORY;
876 2   IF DBLEH >= 4000H THEN DBLEN = 7F80H; ELSE
877 2     DBLEN = DBLEN + SBLEN;
878 2   CALL SIZE$HBUF;
879 2 END SET$DBLEN;

880 1 SIZE$MEMORY; PROCEDURE;
881 2   /* SET UP SOURCE AND DESTINATION BUFFERS */
882 2   SBASE = .MEMORY + SHR(MEMSIZE - .MEMORY,1);
883 2   SBLEH, DBLEN = SHR((MEMSIZE - .MEMORY) AND 0FF00H,1);
884 2   CALL SIZE$HBUF;
885 2 END SIZE$MEMORY;

886 1 COPYCHAR; PROCEDURE;
887 2   /* PERFORM THE ACTUAL COPY FUNCTION */
888 2   DECLARE RESIZED BYTE; /* TRUE IF SBUFF AND DBUFF COMBINED */
889 2   IF (RESIZED .= (BLOCK AND PSOURCE <> 0)) THEN /* BLOCK MODE */
890 2     CALL SET$DBLEN; /* ABSORB SOURCE BUFFER */
891 2     IF HEXT OR IGNOR THEN /* HEX FILE */
892 2       CALL READTAPE; ELSE
893 2       DO WHILE NOT READ$EOF;
894 2         CALL PUTDEST(CHAR);
895 2       END;
896 2     IF RESIZED THEN
897 2       DO; CALL CLEARBUFF;
898 2       CALL SIZE$MEMORY;
899 2     END;

```

1.. VERSION
COPYR.G.IF © 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$EOF; PROCEDURE BYTE;
900 3   RETURN HARDEOF (> 0FFFFH);
901 3   END HARDEOF;
902 2   CALL SIZE$MEMORY;
903 2   TCBP = MCBP; /* FOR ERROR TRACING */
904 2   CALL SETUPDEST;
905 2   CALL SETUPSOURCE;
906 2   /* FILES READY FOR DIRECT COPY */
907 2   FASTCOPY = TRUE;
908 3   /* LOOK FOR PARAMETERS */
909 3   DO I = 0 TO 25;
910 4   IF CONT(I) <> 0 THEN
911 4     DO;
912 4       IF NOT(I = 14 OR I = 21) THEN
913 3         /* NOT OBJ OR VERIFY */
914 2         FASTCOPY = FALSE;
915 2       END;
916 2     IF FASTCOPY THEN /* COPY DIRECTLY TO DBUFF */
917 3       DO; CALL SET$DBLEN; /* EXTEND DBUFF */
918 4       DO WHILE NOT REAL$EOF;
919 4       CALL FILLSOURCE;
920 4       IF REAL$EOF THEN
921 4         NDEST = HARDEOF; ELSE NDEST = DBLEN;
922 4       CALL WRITEDEST;
923 4     END;
924 3   END; ELSE
925 2     CALL COPYCHAR;
926 2     CALL CLOSEDEST;
927 2 END SIMPLECOPY;

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

941 3   NEXTDIR, NCOPIED = 0;
942 2   DO FOREVER;
943 2   /* 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;
950 3   /* FILE CONTROL BLOCK IN BUFFER */
951 3   IF DCNT = 255 THEN

```

CP/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$')); CALL CRLF;
955   4          RETURN;
956   4          END;
957   3      NEXTDIR = DCNT + 1;
/* GET THE FILE CONTROL BLOCK NAME TO DEST */
958   3      CALL MOVE(.BUFFER+SHL(DCNT AND 11B,5),.DEST,16);
959   3      CALL MOVE(.DEST,.SOURCE,16); /* FILL BOTH FCB'S */
960   3      IF (NCOPIED += NCOPIED + 1) = 1 THEN
961   3          CALL PRHT(.('COPYING -$'));
962   3          CALL PRHME;
963   3          CALL SIMPLECOPY;
964   3          END;
965   2      END MULTCOPY;

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      SCANEOL: PROCEDURE;
995   2          CALL SCAN(.SOURCE);
996   2          IF NOT (TYPE = SPEC1 AND CHAR = '=') THEN CALL FORMERR;
998   2          MCBP = CBP; /* FOR ERROR PRINTING */
999   2          END SCANEOL;

/* 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 SIZESMEMORY;
/* MAIN PROCESSING LOOP. PROCESS UNTIL CR ONLY */

1004  1      DO FOREVER;
1005  2          CALL LIFTHEAD;
1006  2          CONCT,COLUMN = 0; /* PRINTER TABS */
/* READ FROM CONSOLE IF NOT A ONELINER */
IF MULTCOM THEN
    DO; CALL PRHTCHAR('+'); CALL READCOM;
    CALL CRLF;
    END;
    CBP = 255;
IF COMLEN = 0 THEN /* SINGLE CARRIAGE RETURN */
    DO; CALL SELECT(CDISK);
    CALL EXIT;
    END;

/* LOOK FOR SPECIAL CASES FIRST */
1019  2      DDISK,SDISK,PSOURCE,PDEST = 0;
1020  2      CALL SCAN(.DEST);
1021  2      IF TYPE = PERIPH THEN GO TO SIMPLECOM;
1023  2      IF TYPE = DISKNAME THEN
    DO; DDISK = DISK - 1;
    CALL SCANEOL;
    CALL SCAN(.SOURCE);
    /* MAY BE MULTI COPY */
    IF TYPE <> FILE THEN CALL FORMERR;
    IF AMBIG THEN
        DO; CALL SCANDEST(.SEARFCB);
        CALL MULTCOPY;
        END; ELSE
        DO; CALL SCANDEST(.DEST);
        /* FORM IS A:=B.UFM */
        CALL SIMPLECOPY;
        END;
    GO TO ENDCOM;
    END;

1041  2      IF TYPE <> FILE OR AMBIG THEN CALL FORMERR;
1043  2      CALL SET$DDISK;
1044  2      CALL SCANEOL;
1045  2      CALL SCAN(.SOURCE);
1046  2      IF TYPE = DISKNAME THEN
        DO;
        CALL SET$SDISK; CALL CHECK$DISK;
        CALL MOVE(.DEST,.SOURCE,33);
        CALL CHECK$EOL;
        CALL SIMPLECOPY;
        GO TO ENDCOM;
        END;
    /* MAY BE POSSIBLE TO DO A FAST DISK COPY */
1055  2      IF TYPE = FILE THEN /* FILE TO FILE */
    DO; CALL DEBLANK; IF CHAR <> CR THEN GO TO SIMPLECOM;
    /* FILE TO FILE */
    CALL SET$SDISK;
    CALL SIMPLECOPY;
    GO TO ENDCOM;
    END;

1064  2      SIMPLECOM;
    CBP = 255; /* READY FOR RESCAN */

```

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

SER. # _____

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

SEQ. #

```

1065 2 /* OTHERWISE PROCESS SIMPLE REQUEST */
1066 2 CALL SCAN(.DEST);
1067 2 IF (TYPE < FILE) OR AMBIG THEN /* DELIMITER OR ERROR */
1068 2   CALL ERROR(.('UNRECOGNIZED DESTINATIONS$'));
1069 2 DHEX = FALSE;
1070 2 IF TYPE = FILE THEN
1071 3   DO; /* DESTINATION IS A FILE, SAVE EXTENT NAME */
1072 3     CALL SET$SDISK;
1073 3     CALL SETUPDEST;
1074 3     CHAR = 255;
1075 2   END; ELSE
1076 2 /* PERIPHERAL NAME */
1077 2 IF CHAR >= HULP OR CHAR <= RDR THEN CALL ERROR(.('CANNOT WRITE$'));
1078 2 IF (PDEST != CHAR + 1) = PUMP THEN CALL NULLS;
1079 2 /* HOW SCAN THE DELIMITER */
1080 2 CALL SCAN(.SOURCE);
1081 2 IF TYPE <> SPEC1 OR CHAR <> '=' THEN
1082 2   CALL ERROR(.('INVALID PIP FORMATS$'));
1083 2 /* OTHERWISE SCAN AND COPY UNTIL CR */
1084 2 C1,C2,C3 = 0; /* CLEAR LINE COUNTERS */
1085 2 SCOPY = TRUE;
1086 2 DO WHILE COPYING;
1087 3   CALL SCAN(.SOURCE);
1088 3   SCOM = FALSE;
1089 3   IF TYPE = FILE AND NOT AMBIG THEN /* A SOURCE FILE */
1090 3     DO;
1091 3       CALL SET$SDISK;
1092 3       CALL SETUPSOURCE;
1093 3       CHAR = 255;
1094 3     END; ELSE
1095 3
1096 3     IF TYPE <> PERIPH OR (CHAR <= LST AND CHAR > RDR) THEN
1097 3       CALL ERROR(.('CANNOT READ$'));
1098 3
1099 3     COLUMN = 0;
1100 3     PUTNUM = TRUE; /* CAUSES IMMEDIATE NEWLINE IF NUMB SET */
1101 3     SCOM = SCOM OR OBJ; /* MAY BE ABSOLUTE COPY */
1102 3     PSOURCE = CHAR + 1;
1103 3     IF CHAR = NULP THEN CALL NULLS; ELSE
1104 3     IF CHAR = EDPP THEN CALL PUTDEST(ENDFILE); ELSE
1105 3       DO; /* DISK COPY */
1106 4         IF ((CHAR < HSRDR AND DHEX) THEN NEXT = 1;
1107 4         /* HEX FILE SET IF SOURCE IS RDR AND DEST IS HEX FILE */
1108 4         IF PDEST = PRNT THEN
1109 4           DO; TABS = 0; NUMB = 1;
1110 5         END;
1111 4         CALL COPYCHAR;
1112 4       END;
1113 3
1114 3     CALL CHECK$STRINGS;
1115 3     /* READ ENDFILE, GO TO NEXT SOURCE */
1116 3     CALL SCAN(.SOURCE);
1117 3     IF TYPE <> SPEC1 OR (CHAR <> ',' AND CHAR <> CR) THEN
1118 3       CALL ERROR(.('INVALID SEPARATOR$'));
1119 2
1120 2
1121 2
1122 2
1123 2
1124 2
1125 2
1126 2
1127 2
1128 1

```

CP/M VERSION _____

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

MODULE INFORMATION:

P. O. BOX 579
PACIFIC GROVE, CA 93950

CODE AREA SIZE = 1947H 6471D
VARIABLE AREA SIZE = 01CAH 4580
MAXIMUM STACK SIZE = 6032H 500
1396 LINES READ
0 PROGRAM ERRORS)

END OF PL/M-80 COMPILATION

CP/M VERSION _____

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

