```
ISIS-II PL/M-86 V2.0 COMPILATION OF MODULE PIPMOD
OBJECT MODULE PLACED IN PIP.OBJ
COMPILER INVOKED BY:  :F0: PIP.PLM XREF OPTIMIZE(3) DEBUG DATE(2/9/83)


            $title('PERIPHERAL INTERCHANGE PROGRAM')
  1         PIPMOD:
              DO;
            /* P E R I P H E R A L  I N T E R C H A N G E  P R O G R A M

                    COPYRIGHT (C) 1976, 1977, 1978, 1979, 1980, 1981, 1982, 1983
                    DIGITAL RESEARCH
                    BOX 579
                    PACIFIC GROVE, CA
                    93950

                    Revised:
                       17 Jan 80  by  Thomas Kolander (MP/M 1.1)
                       05 Oct 81  by  Ray Pedrizetti  (MP/M-86 2.0)
                       18 Dec 81  by  Ray Pedrizetti  (CP/M-86 1.1)
                       29 Jun 82  by  Ray Pedrizetti  (CCP/M-86 3.0) */

            /*    Command lines used for CMD file generation         */

            /* (on VAX)
                asm86 scd1.a86
                    asm86 inpout.a86
                plm86 pip.plm debug xref optimize(3)
                link86 scd1.obj,inpout.obj,pip.obj, to pip.lnk
                loc86 pip.lnk od(sm(code,dats,data,const,stack)) -
                        ad(sm(code(0), dats(10000h)))  ss(stack(+32)) to pip.
                h86 pip

               (on a micro)
                vax pip.h86 $fans
                gencmd pip data[b1000 m280 xfff]

                    * note the beginning of the data segment will change when
                * the program is changed. see the 'MP2' file generated
                * by LOC86.  the constants are last to force hex generation
               */

               /* Compiler Directives */
            $set (mpm)
            $reset (cpm3)
            $cond

  2   1     declare    /* resets stack for error handling */
              reset label external;

  3   1     DECLARE
              MAXB  ADDRESS EXTERNAL,     /* ADDR FIELD OF JMP BDOS */
              FCB (33) BYTE EXTERNAL,     /* DEFAULT FILE CONTROL BLOCK */
              BUFF(128)BYTE EXTERNAL;     /* DEFAULT BUFFER */

  4   1     declare
```

CCP/M-86 2.0 V.5
COPYRIGHT © 1981, 1982, 1983
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. # CC-104

```
                retry byte initial(0);  /* true if error has occured */

 5   1      OUTD: PROCEDURE(B) external;
 6   2          DECLARE B BYTE;
            /* SEND B TO OUT: DEVICE */
 7   2          END OUTD;

 8   1      INPD: PROCEDURE BYTE external;
 9   2          END INPD;

10   1      MON1: PROCEDURE(F,A) EXTERNAL;
11   2          DECLARE F BYTE,
                   A ADDRESS;
12   2          END MON1;

13   1      MON2: PROCEDURE(F,A) BYTE EXTERNAL;
14   2          DECLARE F BYTE,
                   A ADDRESS;
15   2          END MON2;

16   1      MON3: PROCEDURE(F,A) ADDRESS EXTERNAL;
17   2          DECLARE F BYTE,
                   A ADDRESS;
18   2          END MON3;


19   1      plm: procedure public;

20   2      DECLARE
            $if mpm
                VERSION  LITERALLY '0031H', /* REQUIRED FOR BDOS 3.1 OPERATION */
            $else
                VERSION  LITERALLY '0022H', /* REQUIRED FOR OPERATION */
            $endif

                ENDFILE LITERALLY '1AH';    /* END OF FILE MARK */

21   2      DECLARE COPYRIGHT(*) BYTE DATA (
            $if cpm3
                  ' (02/07/83) CP/M 3 PIP VERS 3.1 ');
            $else
                  ' (02/07/83) CCP/M-86 PIP VERS 3.1 ');
            $endif


                /* LITERAL DECLARATIONS */
22   2      DECLARE
                LIT  LITERALLY 'LITERALLY',
                LPP  LIT '60',      /* LINES PER PAGE */
                TAB  LIT '09H',     /* HORIZONTAL TAB */
                FF   LIT '0CH',     /* FORM FEED */
                LA   LIT '05FH',    /* LEFT ARROW */
                LB   LIT '05BH',    /* LEFT BRACKET */
                RB   LIT '05DH',    /* RIGHT BRACKET */

                FSIZE  LIT '33',
                FRSIZE LIT '36',    /* SIZE OF RANDOM FCB */
```

```
            NSIZE  LIT '8',
            FNSIZE LIT '11',
            FEXT   LIT '9',
            FEXTL  LIT '3',

               /* scanner return type code */
            outt     LIT '0',   /* output device */
            PRNT     LIT '1',   /* PRINTER */
            LSTT     LIT '2',   /* list device */
            axot     lit '3',   /* auxilary output device */
            FILE     LIT '4',   /* file type */
            auxt     lit '5',   /* auxilary input/output device */
            CONS     LIT '6',   /* CONSOLE */
            axit     LIT '7',   /* auxilary input device */
            inpt     lit '8',   /* input device */
            NULT     LIT '9',   /* nul characters */
            EOFT     LIT '10',   /* EOF character */
            ERR      LIT '11',  /* error type */
            SPECL    LIT '12',  /* special character */
            DISKNAME LIT '13';  /* diskname letter */

23  2       DECLARE
            SEARFCB LIT 'FCB';   /* SEARCH FCB IN MULTI COPY */

24  2       DECLARE
            TRUE    LIT '1',
            FALSE   LIT '0',
            FOREVER LIT 'WHILE TRUE',
            cntrlc  lit '3',
            CR      LIT '13',
            LF      LIT '10',
            WHAT    LIT '63';

            $if mpm
25  2       declare
            maxmcnt lit '128',    /* maximum multi sector count */
            maxmbuf lit '16384';  /* maximum multi sector buffer size */
            $endif

26  2       DECLARE
            COLUMN   BYTE,      /* COLUMN COUNT FOR PRINTER TABS */
            LINENO   BYTE,      /* LINE WITHIN PAGE */
            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 */
            CDISK    BYTE,      /* CURRENT DISK */
            SBLEN    ADDRESS,   /* SOURCE BUFFER LENGTH */
            DBLEN    ADDRESS,   /* DEST BUFFER LENGTH */
            tblen    address,   /* temp 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 */
```

```
            /* source fcb, password and password mode */
        source structure (
            fcb(frsize)  byte,
$if mpm
            pwnam(nsize) byte,
            pwmode       byte,
$endif
            user         byte,
            type         byte ),

            /* temporary destination fcb, password and password mode */
        dest structure (
            fcb(frsize)  byte,
$if mpm
            pwnam(nsize) byte,
            pwmode       byte,
$endif
            user         byte,
            type         byte ),

            /* original destination fcb, password and password mode */
        odest structure (
            fcb(frsize)  byte,
$if mpm
            pwnam(nsize) byte,
            pwmode       byte,
$endif
            user         byte,
            type         byte ),

        filsize(3) byte,     /* file size random record number */

        DESTR    ADDRESS AT(.DEST.FCB(33)),    /* RANDOM RECORD POSITION */
        SOURCER  ADDRESS AT(.SOURCE.FCB(33)),  /* RANDOM RECORD POSITION */
        DESTR2   BYTE    AT(.DEST.FCB(35)),    /* RANDOM RECORD POSITION R2 */
        SOURCER2 BYTE    AT(.SOURCE.FCB(35)),  /* RANDOM RECORD POSITION R2 */

        extsave  byte,       /* temp extent byte for bdos bug */

        nsbuf    address,    /* next source buffer */
$if mpm
        bufsize address,     /* multsect buffer size */
        mseccnt  byte,       /* last multi sector count value */
$endif
        NSOURCE ADDRESS,     /* NEXT SOURCE CHARACTER */
        NDEST   ADDRESS;     /* NEXT DESTINATION CHARACTER */

27  2   DECLARE
        fastcopy byte,       /* true if copy directly to dbuf */
        dblbuf   byte,       /* true if both source and dest buffer used */
        concat   byte,       /* true if concatination command */
        ambig    byte,       /* true if file is ambig type */
        dfile    byte,       /* true if dest is file type */
        sfile    byte,       /* true if source is file type */
        made     byte,       /* true if destination file already made */
        opened   byte,       /* true if source file open */
        endofsrc byte,       /* true if end of source file */
```

```
            nendcmd  byte,      /* true if not end of command tail */
            insparc  byte,      /* true if in middle of sparce file */
            sparfil  byte,      /* true if sparce file being copied */
            MULTCOM  BYTE,      /* true if processing multiple commands */
            PUTNUM   BYTE,      /* SET WHEN READY FOR NEXT LINE NUM */
            CONCNT   BYTE,      /* COUNTER FOR CONSOLE READY CHECK */
            CHAR     BYTE,      /* LAST CHARACTER SCANNED */
            FLEN     BYTE;      /* FILE NAME LENGTH */

28  2       declare
            f1      byte,       /* f1 user attribute flag */
            f2      byte,       /* f2 user attribute flag */
            f3      byte,       /* f3 user attribute flag */
            f4      byte,       /* f4 user attribute flag */
            ro      byte,       /* read only attribute flag */
            sys     byte,       /* system attribute flag */
        $if mpm
            exten   byte,       /* extention error code */
            odcnt   byte,       /* saves dcnt for open dest file */
            eretry  byte,       /* error return flag */
        $endif
            dcnt    byte;       /* error code or directory code */

29  2       DECLARE CBUFF(130) BYTE,   /* COMMAND BUFFER */
            MAXLEN        BYTE AT (.CBUFF(0)),  /* MAX BUFFER LENGTH */
            COMLEN        BYTE AT (.CBUFF(1)),  /* CURRENT LENGTH */
            COMBUFF(128) BYTE AT (.CBUFF(2)),  /* COMMAND BUFFER CONTENTS */
            CBP           BYTE;                /* COMMAND BUFFER POINTER */

30  2       DECLARE
            CUSER BYTE, /* CURRENT USER NUMBER */
            last$user byte;

31  2       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  J  V  W  X  Y  Z   */
            archiv byte  at(.cont(0)),     /* file archive */
            confrm byte  at(.cont(2)),     /* confirm copy */
            DELET  BYTE  AT(.CONT(3)),     /* DELETE CHARACTERS */
            ECHO   BYTE  AT(.CONT(4)),     /* ECHO CONSOLE CHARACTERS */
            FORMF  BYTE  AT(.CONT(5)),     /* FORM FILTER */
            GETU   BYTE  AT(.CONT(6)),     /* GET FILE, USER # */
            HEXT   BYTE  AT(.CONT(7)),     /* HEX FILE TRANSFER */
            IGNOR  BYTE  AT(.CONT(8)),     /* IGNORE :00 RECORD ON FILE */
            kilds  byte  at(.cont(10)),    /* kill filename display */
            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 */
            PAGCNT BYTE  AT(.CONT(15)),    /* PAGE LENGTH */
            QUITS  BYTE  AT(.CONT(16)),    /* QUIT COPY */
            RSYS   BYTE  AT(.CONT(17)),    /* READ SYSTEM FILES */
            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 */
                WRROF  BYTE  AT(.CONT(22)),    /* WRITE TO R/O FILE */
                ZEROP  BYTE  AT(.CONT(25));    /* ZERO PARITY ON INPUT */

32   2      DECLARE ZEROSUP BYTE,  /* ZERO SUPPRESSION */
                (C3,C2,C1) BYTE;     /* LINE COUNT ON PRINTER  */


            $if mpm
33   2      retcodes: procedure(a);
34   3          declare a address;
35   3          dcnt = low(a);
36   3          exten = high(a);
37   3          end retcodes;
            $endif

38   2      BOOT: PROCEDURE;
                /* SYSTEM REBOOT */
39   3          CALL MON1(0,0);
40   3          END BOOT;


41   2      RDCHAR: PROCEDURE BYTE;
                /* READ CONSOLE CHARACTER */
42   3          RETURN MON2(1,0);
43   3          END RDCHAR;

44   2      PRINTCHAR: PROCEDURE(CHAR);
45   3          DECLARE CHAR BYTE;
46   3          CALL MON1(2,CHAR AND 7FH);
47   3          END PRINTCHAR;

48   2      CRLF: PROCEDURE;
49   3          CALL PRINTCHAR(CR);
50   3          CALL PRINTCHAR(LF);
51   3          END CRLF;

52   2      printx: procedure(a);
53   3          declare a address;
54   3          call mon1(9,a);
55   3          end printx;

56   2      PRINT: PROCEDURE(A);
57   3          DECLARE A ADDRESS;
                /* PRINT THE STRING STARTING AT ADDRESS A UNTIL THE
                NEXT DOLLAR SIGN IS ENCOUNTERED */
58   3          CALL CRLF;
59   3          CALL printx(A);
60   3          END PRINT;

61   2      RDCOM: PROCEDURE;
                /* READ INTO COMMAND BUFFER */
62   3          MAXLEN = 128;
63   3          CALL MON1(10,.MAXLEN);
64   3          END RDCOM;
```

```
65   2      CVERSION: PROCEDURE ADDRESS;
66   3          RETURN MON3(12,0); /* VERSION NUMBER */
67   3          END CVERSION;

68   2      SETDMA: PROCEDURE(A);
69   3          DECLARE A ADDRESS;
70   3          CALL MON1(26,A);
71   3          END SETDMA;

            $if mpm
72   2      setpw: procedure(fcba);
73   3          declare fcba address;
74   3          declare fcbs based fcba structure (
                    fcb(frsize) byte,
                    pwnam(nsize) byte );
75   3          call setdma(.fcbs.pwnam(0));
76   3          end setpw;
            $endif

77   2      OPEN: PROCEDURE(fcba);
78   3          DECLARE fcba ADDRESS;
79   3          declare fcb based fcba (frsize) byte;
            $if mpm
80   3          CALL SETPW(fcba);
81   3          call retcodes(mon3(15,fcba));
            $else
            dcnt = mon2(15,fcba);
            $endif
82   3          if dcnt <> 255 and rol(fcb(8),1) then
83   3              do; call mon1(16,fcba);
85   4              dcnt = 255;
            $if mpm
86   4              exten = 0;
            $endif
87   4              end;
88   3          END OPEN;

89   2      CLOSE: PROCEDURE(FCB);
90   3          DECLARE FCB ADDRESS;
            $if mpm
91   3          call retcodes(MON3(16,FCB));
            $else
            dcnt = MON2(16,FCB);
            $endif
92   3          END CLOSE;

93   2      SEARCH: PROCEDURE(FCB);
94   3          DECLARE FCB ADDRESS;
            $if mpm
95   3          call retcodes(MON3(17,FCB));
            $else
            dcnt = MON2(17,FCB);
            $endif
96   3          END SEARCH;

97   2      SEARCHN: PROCEDURE;
            $if mpm
```

```
 98   3          call retcodes(MON3(16,0));
                $else
                    dcnt = MON2(16,0);
                $endif
 99   3          END SEARCHN;

100   2      DELETE: PROCEDURE(FCB);
101   3          DECLARE FCB ADDRESS;
                $if mpm
102   3          CALL SETPW(FCB);
103   3          call retcodes(MON3(19,FCB));
                $else
                    CALL MON1(19,FCB);
                $endif
104   3          END DELETE;

105   2      DISKRD: PROCEDURE(FCB);
106   3          DECLARE FCB ADDRESS;
                $if mpm
107   3          call retcodes(MON3(20,FCB));
                $else
                    dcnt = MON2(20,FCB);
                $endif
108   3          END DISKRD;

109   2      DISKWRITE: PROCEDURE(FCB);
110   3          DECLARE FCB ADDRESS;
                $if mpm
111   3          call retcodes(MON3(21,FCB));
                $else
                    dcnt = MON2(21,FCB);
                $endif
112   3          END DISKWRITE;

113   2      MAKE: procedure(fcba);
114   3          declare fcba address;
                $if mpm
115   3          declare fcbs based fcba structure (
                        fcb(frsize) byte,
                        pwnam(nsize) byte );
116   3          if fcbs.pwnam(0) = 0 then   /* zero if no password */
117   3              fcbs.fcb(6) = fcbs.fcb(6) and 7fh;  /* reset password attribute */
118   3          else do;
119   4              fcbs.fcb(6) = fcbs.fcb(6) or 80h;  /* set password attribute */
120   4              call setdma(.fcbs.pwnam(0));  /* set password dma */
121   4              end;
122   3          call retcodes(mon3(22,fcba));
                $else
                    dcnt = mon2(22,fcba);
                $endif
123   3          END MAKE;

124   2      RENAME: PROCEDURE(FCB);
125   3          DECLARE FCB ADDRESS;
                $if mpm
126   3          CALL SETPW(FCB);
127   3          call retcodes(MON3(23,FCB)) ;
```

```
                $else
                    dcnt = MON2(23,FCB);
                $endif
128   3         END RENAME;

129   2       getdisk: procedure byte;
130   3           return mon2(25,0);
131   3           end getdisk;

132   2       SETIND: PROCEDURE(FCB);
133   3           DECLARE FCB ADDRESS;
                $if mpm
134   3           call retcodes(MON3(30,FCB));
                $else
                    dcnt = MON2(30,FCB);
                $endif
135   3           END SETIND;

136   2       GETUSER: PROCEDURE BYTE;
137   3           RETURN MON2(32,0FFH);
138   3           END GETUSER;

139   2       SETUSER: PROCEDURE(USER);
140   3           DECLARE USER BYTE;
141   3           if last$user <> user then
142   3               CALL MON1(32,(last$user:=USER));
143   3           END SETUSER;

144   2       SETCUSER: PROCEDURE;
145   3           CALL SETUSER(CUSER);
146   3           END SETCUSER;

147   2       setduser: procedure;
148   3           call setuser(odest.user);
149   3           end setduser;

150   2       SETSUSER: PROCEDURE;
151   3           CALL SETUSER(source.user);
152   3           END SETSUSER;

153   2       RD$RANDOM: PROCEDURE(FCB) BYTE;
154   3           DECLARE FCB ADDRESS;
                $if mpm
155   3           call retcodes(mon3(33,fcb));
                $else
                    dcnt = mon2(33,fcb);
                $endif
156   3           return dcnt;
157   3           END RD$RANDOM;

158   2       write$random: procedure(fcb) byte;
159   3           declare fcb address;
                $if mpm
160   3           call retcodes(mon3(34,fcb));
                $else
                    dcnt = mon2(34,fcb);
                $endif
```

```
161   3          return dcnt;
162   3          end write$random;

163   2      retfsize: procedure(fcb) byte;
164   3          declare fcb address;
165   3          return mon2(35,fcb);
166   3          end retfsize;

167   2      SET$RANDOM: PROCEDURE(FCB);
168   3          DECLARE FCB ADDRESS;
                  /* SET RANDOM RECORD POSITION */
169   3          CALL MON1(36,FCB);
170   3          END SET$RANDOM;

            $if mpm
171   2      multsect: procedure(cnt);
172   3          declare cnt byte;
173   3          if mseccnt <> cnt then
174   3              call mon1(44,(mseccnt := cnt));
175   3          end multsect;

176   2      flushbuf: procedure;
177   3          call mon1(48, 0ffh);    /* 0FFH = flush and discard buffers */
178   3          end flushbuf;

179   2      conatlst: procedure byte;
180   3          return mon2(161,0);
181   3          end conatlst;
            $endif


182   2      MOVE: PROCEDURE(S,D,N);
183   3          DECLARE (S,D) ADDRESS, N BYTE;
184   3          DECLARE A BASED S BYTE, B BASED D BYTE;
185   3              DO WHILE (N:=N-1) <> 255;
186   4              B = A; S = S+1; D = D+1;
189   4              END;
190   3          END MOVE;

                  /* errtype error messages */
191   2          declare er00(*) byte data ('DISK READ$');
192   2          declare er01(*) byte data ('DISK WRITE$');
193   2          declare er02(*) byte data ('VERIFY$');
194   2          declare er03(*) byte data ('INVALID DESTINATION$');
195   2          declare er04(*) byte data ('INVALID SOURCE$');
196   2          declare er05(*) byte data ('USER ABORTED$');
197   2          declare er06(*) byte data ('BAD PARAMETER$');
198   2          declare er07(*) byte data ('INVALID USER NUMBER$');
199   2          declare er08(*) byte data ('INVALID FORMAT$');
200   2          declare er09(*) byte data ('HEX RECORD CHECKSUM$');
201   2          declare er10(*) byte data ('FILE NOT FOUND$');
202   2          declare er11(*) byte data ('START NOT FOUND$');
203   2          declare er12(*) byte data ('QUIT NOT FOUND$');
204   2          declare er13(*) byte data ('INVALID HEX DIGIT$');
205   2          declare er14(*) byte data ('CLOSE FILE$');
206   2          declare er15(*) byte data ('UNEXPECTED END OF HEX FILE$');
207   2          declare er16(*) byte data ('INVALID SEPARATOR$');
```

```
208   2           declare er17(*) byte data ('NO DIRECTORY SPACE$');
209   2           declare er18(*) byte data ('INVALID FORMAT WITH SPARCE FILE$');
            $if mpm
210   2           declare er19(*) byte data ('MAKE FILE$');
211   2           declare er20(*) byte data ('OPEN FILE$');
212   2           declare er21(*) byte data ('PRINTER BUSY$');
213   2           declare er22(*) byte data ('CAN''T DELETE TEMP FILE$');
            $endif

214   2           declare errmsg(*) address data(
                    .er00,.er01,.er02,.er03,.er04,
                    .er05,.er06,.er07,.er08,.er09,
                    .er10,.er11,.er12,.er13,.er14,
                    .er15,.er16,.er17,.er18
            $if mpm
                    ,.er19,.er20,.er21,.er22
            $endif
                    );

215   2           declare sper00(*) byte data ('NO DIRECTORY SPACE$');
216   2           declare sper01(*) byte data ('NO DATA BLOCK$');
217   2           declare sper02(*) byte data ('CAN''T CLOSE CURRENT EXTENT$');
218   2           declare sper03(*) byte data ('SEEK TO UNWRITTEN EXTENT$');
219   2           declare sper05(*) byte data ('RANDOM RECORD OUT OF RANGE$');
220   2           declare sper06(*) byte data ('RECORDS DON''T MATCH$');
221   2           declare sper07(*) byte data ('RECORD LOCKED$');
222   2           declare sper08(*) byte data ('INVALID FILENAME$');
223   2           declare sper09(*) byte data ('FCB CHECKSUM$');

224   2           declare numspmsgs lit '10';  /* number of extended messages */
225   2           declare special$msg(numspmsgs) address data(
                    .sper00,.sper01,.sper02,.sper03,.sper00,
                    .sper05,.sper06,.sper07,.sper08,.sper09);

            $if mpm
                /* extended error messages */
226   2           declare ex00(*) byte data ('$');  /* NO MESSAGE */
227   2           declare ex01(*) byte data ('NONRECOVERABLE$');
228   2           declare ex02(*) byte data ('R/O DISK$');
229   2           declare ex03(*) byte data ('R/O FILE$');
230   2           declare ex04(*) byte data ('INVALID DISK SELECT$');
231   2           declare ex05(*) byte data ('INCOMPATIBLE MODE$');
232   2           declare ex07(*) byte data ('INVALID PASSWORD$');
233   2           declare ex08(*) byte data ('ALREADY EXISTS$');
234   2           declare ex10(*) byte data ('LIMIT EXCEEDED$');

235   2           declare nummsgs lit '11';  /* number of extended messages */
236   2           declare extmsg(nummsgs) address data(
                    .ex00,.ex01,.ex02,.ex03,.ex04,
                    .ex05,.sper09,.ex07,.ex08,.sper08,
                    .ex10);
            $endif

237   2       error$cleanup: procedure;
            $if mpm
238   3           call multsect(1);
            $endif
```

```
239   3          eretry = 0;   /* initialize to no error retry */
240   3          if opened then   /* if source file opened */
241   3              do; call setsuser;
243   4              call close(.source);
244   4              opened = false;
245   4              end;
246   3          if made then
247   3              do; call setduser;
249   4              call close(.dest);
250   4              call delete(.dest);  /* delete destination scratch file */
251   4              end;
               /* Zero the command length in case this is a single command */
252   3             comlen = 0;
253   3             retry = true;
254   3             call print(.('ERROR: $'));
255   3          end error$cleanup;

256   2      error: procedure (errtype);
257   3          declare errtype byte;

258   3          call error$cleanup;
259   3          call printx(errmsg(errtype));
260   3          call crlf;
261   3          go to reset;
262   3          end error;

263   2      xerror: procedure (funcno,fileadr);
264   3          declare temp    byte,
                        i       byte,
                        sdcnt   byte,
                        sexten  byte,
                        funcno  byte,
                        fileadr address,
                        fcb based fileadr (fsize) byte;

265   3          declare message$index$tbl(17) byte data
                        (2,18,13,15,9,3,10,20,14,10,22,17,19,0,1,0,1);

266   3          sdcnt = dcnt;
267   3          sexten = exten;
268   3          call error$cleanup;

269   3          if (funcno < 6) or (sdcnt <> 0ffh) then
270   3             sexten = 0;
271   3          else sexten = sexten and 0fh;

272   3          call printx(errmsg(message$index$tbl(funcno)));

273   3          if (funcno > 12) and (funcno < 17) and
                   (sdcnt <> 0ffh) and (sdcnt <= numspmsgs) then
274   3             do; call printchar(' ');
276   4             call printx(special$msg(sdcnt-1));
277   4             sexten = 0;
278   4             end;

               $if mpm
279   3          if sexten < nummsgs then
```

```
280   3             do; call printchar(' ');
282   4             call printx(extmsg(sexten));
283   4             end;
            $endif

284   3         call printx(.(' - $'));
285   3         if fileadr <> 0 then
286   3             do; call printchar('A' + fcb(0) - 1);
288   4             call printchar(':');
289   4                 do i = 1 to fnsize;
290   5                 if (temp := fcb(i) and 07fh) <> ' ' then
291   5                     do; if i = fext then call printchar('.');
294   6                     call printchar(temp);
295   6                     end;
296   5                 end;
297   4             end;
298   3         call crlf;

299   3         if (sdcnt = 3) or (sdcnt = 4) or (sdcnt = 6) or (sdcnt = 8) then
300   3            eretry = ambig;
          else
301   3            if (sexten = 3) or ((sexten > 4) and (sexten < 9)) or (sexten > 9) then
302   3                 eretry = ambig;

          go to reset;
304   3       end xerror;

305   2       FORMERR: PROCEDURE;
306   3           call error(8);  /* invalid format */
307   3           END FORMERR;

308   2       CONBRK: PROCEDURE;
              /* CHECK CONSOLE CHARACTER READY */
309   3           if mon2(11,0) <> 0 then
310   3               if mon2(6,0fdh) = cntrlc then
311   3                  call error(5);
312   3           END CONBRK;

313   2       MAXSIZE: procedure byte;
              /* three byte compare of random record field
                 returns true if source.fcb.ranrec >= filesize */

314   3           if (source.fcb(35) < filsize(2)) then
315   3             return false;
316   3           if (source.fcb(35) = filsize(2)) then
317   3             do;
318   4             if (source.fcb(34) < filsize(1)) then
319   4               return false;
320   4             if (source.fcb(34) = filsize(1)) then
321   4               do;
322   5               if (source.fcb(33) < filsize(0)) then
323   5                 return false;
324   5               end;
325   4             end;
326   3           return true;
327   3           end maxsize;
```

```
328   2     SETUPDEST: PROCEDURE;
329   3         call setduser;  /* destination user */
            $if mpm
330   3         call move(.odest,.dest,(frsize + nsize + 1));  /* save original dest */
            $else
              call move(.odest,.dest,(frsize + 1));  /* save original dest */
            $endif
              /* MOVE THREE CHARACTER EXTENT INTO DEST FCB */
331   3         CALL MOVE(.('$$$'),.DEST.FCB(FEXT),FEXTL);
            $if mpm
332   3         odest.fcb(6) = odest.fcb(6) or 80h;
333   3         call open(.odest);  /* try to open destination file */
334   3         odcnt = dcnt;       /* and save error code */
335   3         if odcnt <> 255 then
336   3             call close(.odest);
337   3         else if (exten and 0fh) <> 0 then /* file exists */
338   3             call xerror(7,.odest);        /* but can't open - error */

              CALL DELETE(.DEST); /* REMOVE OLD $$$ FILE */
340   3         if dcnt = 255 and exten <> 0 then
                  /* cant delete temp file */
341   3             call xerror(10,.dest);
342   3         CALL MAKE(.DEST); /* CREATE A NEW ONE */
343   3         IF DCNT = 255 THEN
344   3             if (exten and 0fh) = 0 then
345   3                 call xerror(11,.dest);  /* no directory space */
346   3             else call xerror(12,.dest);  /* make file error */
            $else
              CALL DELETE(.DEST); /* REMOVE OLD $$$ FILE */
              CALL MAKE(.DEST); /* CREATE A NEW ONE */
              IF DCNT = 255 THEN
                call 17,.dest);  /* no directory space */
            $endif
347   3         DEST.FCB(32) = 0;
348   3         made = true;
349   3         END SETUPDEST;

350   2     SETUPSOURCE: PROCEDURE;
351   3         declare (i,j) byte;
352   3         CALL SETSUSER; /* SOURCE USER */
            $if mpm
353   3         source.fcb(6) = source.fcb(6) or 80h;
            $endif
354   3         CALL OPEN(.SOURCE);  /* open source */
355   3         if dcnt <> 255 then
356   3             opened = true;
357   3         IF (NOT RSYS) AND ROL(SOURCE.FCB(10),1) THEN
                  /* skip system file */
358   3             DCNT = 255;
359   3         IF DCNT = 255 THEN
            $if mpm
360   3             if (exten and 0fh) = 0 then
361   3                 call xerror(6,.source);  /* file not found */
                  else
362   3                 call xerror(7,.source);  /* open file error */
            $else
                  call xerror(6,.source);  /* file not found */
```

```
                $endif
363   3             f1 = source.fcb(1) and 80h;  /* save file atributes */
364   3             f2 = source.fcb(2) and 80h;
365   3             f3 = source.fcb(3) and 80h;
366   3             f4 = source.fcb(4) and 80h;
367   3             ro = source.fcb(9) and 80h;
368   3             sys = source.fcb(10) and 80h;
369   3             dcnt = retfsize(.source);
370   3             call move(.source.fcb(33),.filsize,3);
371   3             SOURCE.FCB(32) = 0;
372   3             source.fcb(33),source.fcb(34),source.fcb(35) = 0;
                    /* cause immediate read with no preceding write */
373   3             NSOURCE = 0ffffh;
374   3             END SETUPSOURCE;

375   2         WRITEDEST: PROCEDURE;
                    /* WRITE OUTPUT BUFFERS UP TO BUT NOT INCLUDING POSITION
                    NDEST - THE LOW ORDER 7 BITS OF NDEST ARE ZERO */
376   3             DECLARE (J,DATAOK) BYTE,
                        (tdest,n)      address;
377   3             if not made then call setupdest;
379   3             if (n := ndest and 0ff80h) = 0 then return;
381   3             tdest = 0;
382   3             call setduser;  /* destination user */
383   3             if (sparfil := (sparfil or insparc)) then
                        /* set up fcb from random record no. */
384   3                 do;
                $if mpm
385   4                 call multsect(1);
                $endif
386   4                 CALL SETDMA(.dbuff(tdest));
387   4                 if write$random(.dest) <> 0 then
388   4                     call xerror(16,.dest);  /* DISK WRITE ERROR */
389   4                 end;
                    else
390   3                 CALL SETRANDOM(.DEST); /* SET BASE RECORD FOR VERIFY */
                $if mpm
391   3             if fastcopy then
392   3                 do; bufsize = maxmbuf;
394   4                 call multsect(maxmcnt);
395   4                 end;
                    else
396   3                 do; bufsize = 128;
398   4                 call multsect(1);
399   4                 end;
                $endif

400   3                 do while n - tdest > 127;
                $if mpm
401   4                 if fastcopy and (n - tdest < maxmbuf) then
402   4                     do; bufsize = n - tdest;
404   5                     call multsect(low(shr(bufsize,7)));
405   5                     end;
                $endif
                        /* SET DMA ADDRESS TO NEXT BUFFER */
406   4                 CALL SETDMA(.dbuff(tdest));
407   4                 call diskwrite(.dest);
```

```
408   4               IF dcnt <> 0 THEN
409   4                   call xerror(14,.dest);  /* DISK WRITE ERROR */
              $if mpm
410   4               tdest = tdest + bufsize;
              $else
                      tdest = tdest + 128;
              $endif
411   4               END;

412   3           IF VERIF THEN /* VERIFY DATA WRITTEN OK */
413   3               DO;
414   4               call flushbuf;
415   4               tdest = 0;
              $if mpm
416   4               call multsect(1);
              $endif
417   4               CALL SETDMA(.BUFF); /* FOR COMPARE */
418   4                   do while tdest < n;
419   5                   DATAOK = (RDRANDOM(.DEST) = 0);
420   5                   if (DESTR := DESTR + 1) = 0 then  /* 3 byte inc for */
421   5                       destr2 = destr2 + 1;  /* next random record */
422   5                   J = 0;
                              /* PERFORM COMPARISON */
423   5                       DO WHILE DATAOK AND J < 80H;
424   6                       DATAOK = (BUFF(J) = DBUFF(tdest+J));
425   6                       J = J + 1;
426   6                       END;
427   5                   tdest = tdest + 128;
428   5                   IF NOT DATAOK THEN
429   5                       call xerror(0,.dest);  /* VERIFY ERROR */
430   5                   END;
431   4               call diskrd(.dest);
                      /* NOW READY TO CONTINUE THE WRITE OPERATION */
432   4               END;
433   3           CALL SETRANDOM(.DEST); /* set base record for sparce copy */
434   3           call move(.dbuff(tdest),.dbuff(0),low(ndest := ndest - tdest));
435   3           END WRITEDEST;

436   2       FILLSOURCE: PROCEDURE;
                  /* FILL THE SOURCE BUFFER */
437   3           call conbrk;
              $if mpm
438   3           if fastcopy then
439   3               do; bufsize = maxmbuf;
441   4               call multsect(maxmcnt);
442   4               end;
443   3           else do;
444   4               bufsize = 128;
445   4               call multsect(1);
446   4               end;
              $endif
447   3           CALL SETSUSER; /* SOURCE USER NUMBER SET */
448   3           nsource = nsbuf;
449   3               do while sblen - nsbuf > 127;
450   4               if fastcopy and (sblen - nsbuf < maxmbuf) then
451   4                   do; bufsize = (sblen - nsbuf) and 0ff80h;
453   5                   call multsect(low(shr(bufsize,7)));
```

```
454  5                  end;
                      /* SET DMA ADDRESS TO NEXT BUFFER POSITION */
455  4                  CALL SETDMA(.SBUFF(nsbuf));
456  4                  extsave = source.fcb(12);  /* save extent field */
457  4                  call diskrd(.source);
458  4                  IF dcnt <> 0 THEN
459  4                      DO; IF dcnt <> 1 THEN
461  5                          call xerror(13,.source);  /* DISK READ ERROR */
                          /* END - OF - FILE */
                  $if mpm
462  5                      if fastcopy then   /* add no. sectors copied */
463  5                          nsbuf = nsbuf + shl(double(exten),7);
                           /* nsbuf = nsbuf + shl(double(exten and 0f0h),3); */
                  $endif
                          /* check boundry condition for bug in bdos and correct */
464  5                      if (source.fcb(12) <> extsave) and (source.fcb(32) = 80h) then
465  5                          source.fcb(32) = 0;  /* zero current record */
466  5                      call set$random(.source);
467  5                      if (insparc := not maxsize) then
468  5                          do;
469  6                          if concat or (not fastcopy) then
                                  /* invalid format with sparce file */
470  6                              call xerror(1,.source);
471  6                          end;
                           else
472  5                          do;
473  6                          call close(.source);
474  6                          opened = false;
475  6                          end;
476  5                      endofsrc = true;  /* set end of source file */
477  5                      SBUFF(nsbuf) = ENDFILE; return;
479  5                      END;
                      ELSE
                  $if mpm
480  4                      nsbuf = nsbuf + bufsize;
                  $else
                          nsbuf = nsbuf + 128;
                  $endif
481  4                  END;
482  3              END FILLSOURCE;

483  2              PUTDCHAR: PROCEDURE(B);
484  3                  DECLARE B BYTE;
                      /* WRITE BYTE B TO THE DESTINATION DEVICE GIVEN BY ODEST.TYPE */
485  3                  IF B >= ' ' THEN
486  3                      DO; COLUMN = COLUMN + 1;
488  4                      IF DELET > 0 THEN /* MAY BE PAST RIGHT SIDE */
489  4                          DO; IF COLUMN > DELET THEN RETURN;
492  5                          END;
493  4                      END;
494  3                  if echo then call mon1(2,b);  /* echo to console */
496  3                  do case odest.type;
                          /* CASE 0 IS OUT */
497  4                      CALL OUTO(B);
                          /* CASE 1 IS PRN, TABS EXPANDED, LINES LISTED */
498  4                      call mon1(5,b);
                          /* CASE 2 IS LST */
```

```
499   4                   CALL MON1(5,B);
                        /* CASE 3 IS axo */
500   4           axocase:
                  $if not mpm
                        CALL MON1(4,B);
                  $else
                        call error(3);  /* invalid destination */
                  $endif
                        /* CASE 4 IS DESTINATION FILE */
501   4                   DO;
502   5                   IF NDEST >= DBLEN THEN CALL WRITEDEST;
504   5                   DBUFF(NDEST) = B;
505   5                   NDEST = NDEST+1;
506   5                   END;
                        /* CASE 5 IS AUX */
507   4                   goto axocase;
                        /* CASE 6 IS CON */
508   4                   CALL MON1(2,B);
509   4                 END; /* of case */
510   3               END PUTDCHAR;

511   2           PUTDESTC: PROCEDURE(B);
512   3               DECLARE (B,I) BYTE;
                      /* WRITE DESTINATION CHARACTER, TAB EXPANSION */
513   3               IF B <> TAB THEN CALL PUTDCHAR(B);
515   3               ELSE IF TABS = 0 THEN CALL PUTDCHAR(B);
                        ELSE  /* B IS TAB CHAR, TABS > 0 */
517   3                   DO; I = COLUMN;
519   4                       DO WHILE I >= TABS;
520   5                       I = I - TABS;
521   5                       END;
522   4                   I = TABS - I;
523   4                       DO WHILE I > 0;
524   5                       I = I - 1;
525   5                       CALL PUTDCHAR(' ');
526   5                       END;
527   4                   END;
528   3               IF B = CR THEN COLUMN = 0;
530   3               END PUTDESTC;

531   2           PRINT1: PROCEDURE(B);
532   3               DECLARE B BYTE;
533   3               IF (ZEROSUP := ZEROSUP AND B = 0) THEN
534   3                   CALL PUTDESTC(' ');
                        ELSE
535   3                   CALL PUTDESTC('0'+B);
536   3               END PRINT1;

537   2           PRINTDIG: PROCEDURE(D);
538   3               DECLARE D BYTE;
539   3               CALL PRINT1(SHR(D,4)); CALL PRINT1(D AND 1111B);
541   3               END PRINTDIG;

542   2           NEWLINE: PROCEDURE;
543   3               DECLARE ONE BYTE;
544   3               ONE = 1;
545   3               ZEROSUP = (NUMB = 1);
```

```
546   3           C1 = DEC(C1+ONE); C2 = DEC(C2 PLUS 0); C3 = DEC(C3 PLUS 0);
549   3           CALL PRINTDIG(C3); CALL PRINTDIG(C2); CALL PRINTDIG(C1);
552   3           IF NUMB = 1 THEN /* USUALLY PRINTER OUTPUT */
553   3               DO; CALL PUTDESTC(':'); CALL PUTDESTC(' ');
556   4               END;
                  ELSE
557   3               CALL PUTDESTC(TAB);
558   3           END NEWLINE;

559   2       PUTDEST: PROCEDURE(B);
560   3           DECLARE (I,B) BYTE;
                  /* WRITE DESTINATION CHARACTER, CHECK TABS AND LINES */
561   3           IF FORMF THEN /* SKIP FORM FEEDS */
562   3               DO; IF B = FF THEN RETURN;
565   4               END;
566   3           IF PUTNUM THEN /* END OF LINE OR START OF FILE */
567   3               DO;
568   4               IF (B <> FF) and (b <> endfile) THEN
569   4                   DO;    /* NOT FORM FEED or end of file */
570   5                   IF (I:=PAGCNT) <> 0 THEN /* PAGE EJECT */
571   5                       DO; IF I=1 THEN I=LPP;
574   6                       IF (LINENO := LINENO + 1) >= I THEN
575   6                           DO; LINENO = 0; /* NEW PAGE */
577   7                           CALL PUTDESTC(FF);
578   7                           END;
579   6                       END;
580   5                   IF NUMB > 0 THEN
581   5                       CALL NEWLINE;
582   5                   PUTNUM = FALSE;
583   5                   END;
584   4               END;
585   3           IF B = FF THEN LINENO = 0;
587   3           CALL PUTDESTC(B);
588   3           IF B = LF THEN PUTNUM = TRUE;
590   3           END PUTDEST;

591   2       UTRAN: PROCEDURE(B) BYTE;
592   3           DECLARE B BYTE;
                  /* TRANSLATE ALPHA TO UPPER CASE */
593   3           IF B >= 110$0001B AND B <= 111$1010B THEN /* LOWER CASE */
594   3               B = B AND 101$1111B; /* TO UPPER CASE */
595   3           RETURN B;
596   3           END UTRAN;

597   2       LTRAN: PROCEDURE(B) BYTE;
598   3           DECLARE B BYTE;
                  /* TRANSLATE TO LOWER CASE ALPHA */
599   3           IF B >= 'A' AND B <= 'Z' THEN
600   3               B = B OR 10$0000B; /* TO LOWER */
601   3           RETURN B;
602   3           END LTRAN;

603   2       GETSOURCEC: PROCEDURE BYTE;
                  /* READ NEXT SOURCE CHARACTER */
604   3           DECLARE (B,CONCHK) BYTE;
```

```
605   3          CONCHK = TRUE; /* CONSOLE STATUS CHECK BELOW */
606   3              DO CASE source.type;
                      /* CASE 0 IS out */
607   4                 go to notsource;
                      /* CASE 1 IS prn */
608   4                 go to notsource;
                      /* CASE 2 IS lst */
609   4                 notsource:
                            call error(4);  /* INVALID SOURCE */
                      /* CASE 3 IS axo */
610   4                 go to notsource;
                      /* CASE 4 IS SOURCE FILE */
611   4                 DO;
612   5                 IF NSOURCE >= SBLEN THEN
613   5                     do; if dblbuf or (not dfile) then
615   6                         nsbuf = 0;
616   6                     else if (nsource <> 0ffffh) then
617   6                         do; call writedest;
619   7                         nsbuf = ndest;
620   7                         end;
                          CALL FILLSOURCE;
622   6                     end;
623   5                 B = SBUFF(NSOURCE);
624   5                 NSOURCE = NSOURCE + 1;
625   5                 END;
                      /* CASE 5 IS AUX */
626   4                goto axicase;
                      /* CASE 6 IS CON */
627   4                 DO; CONCHK = FALSE; /* DON'T CHECK CONSOLE STATUS */
629   5                 B = MON2(1,0);
630   5                 END;
                      /* CASE 7 IS axi */
631   4         axicase:
          $if not mpm
                          B = MON2(3,0) AND 7FH;
          $else
                          go to notsource;
          $endif
                      /* CASE 7 IS INP */
632   4                  B = INPD;
633   4                 END; /* OF CASES */

634   3             IF CONCHK THEN /* TEST FOR CONSOLE CHAR READY */
635   3                 DO;
636   4                 IF obj THEN /* SOURCE IS AN OBJECT FILE */
637   4                     CONCHK = ((CONCNT := CONCNT + 1) = 0);
                      ELSE /* ASCII */
638   4                     CONCHK = (B = LF);
639   4                 IF CONCHK THEN
640   4                     DO;
641   5                     call CONBRK;
642   5                     END;
643   4                 END;
644   3             IF ZEROP THEN B = B AND 7FH;
646   3             IF UPPER THEN RETURN UTRAN(B);
648   3             IF LOWER THEN RETURN LTRAN(B);
650   3             RETURN B;
```

```
651   3          END GETSOURCEC;

652   2      GETSOURCE: PROCEDURE BYTE;
                 /* GET NEXT SOURCE CHARACTER */
653   3          DECLARE CHAR BYTE;
654   3          MATCH: PROCEDURE(B) BYTE;
                     /* MATCH START AND QUIT STRINGS */
655   4              DECLARE (B,C) BYTE;
656   4              IF (C:=COMBUFF(B:=(B+MATCHLEN))) = ENDFILE THEN /* END MATCH */
657   4                  DO; COMBUFF(B) = CHAR; /* SAVE CURRENT CHARACTER */
659   5                  RETURN TRUE;
660   5                  END;
661   4              IF C = CHAR THEN MATCHLEN = MATCHLEN + 1;
                     ELSE
663   4                  MATCHLEN = 0; /* NO MATCH */
664   4              RETURN FALSE;
665   4              END MATCH;

666   3          IF QUITLEN > 0 THEN
667   3              DO; IF (QUITLEN := QUITLEN - 1) = 1 THEN RETURN LF;
670   4              RETURN ENDFILE; /* TERMINATED WITH CR,LF,ENDFILE */
671   4              END;
672   3          DO FOREVER; /* LOOKING FOR START */
673   4          IF FEEDLEN > 0 THEN /* GET SEARCH CHARACTERS */
674   4              DO; FEEDLEN = FEEDLEN - 1;
676   5              CHAR = COMBUFF(FEEDBASE);
677   5              FEEDBASE = FEEDBASE + 1;
678   5              RETURN CHAR;
679   5              END;
680   4          IF (CHAR := GETSOURCEC) = ENDFILE THEN RETURN ENDFILE;
682   4          IF STARTS > 0 THEN /* LOOKING FOR START STRING */
683   4              DO; IF MATCH(STARTS) THEN
685   5                  DO; FEEDBASE = STARTS; STARTS = 0;
688   6                  FEEDLEN = MATCHLEN + 1;
689   6                  matchlen = 0;
690   6                  END; /* OTHERWISE NO MATCH, SKIP CHARACTER */
691   5              END;
692   4          ELSE IF QUITS > 0 THEN /* PASS CHARACTERS TIL MATCH */
693   4              DO; IF MATCH(QUITS) THEN
695   5                  DO; QUITS = 0; QUITLEN = 2;
                         /* SUBSEQUENTLY RETURN CR, LF, ENDFILE */
698   6                  RETURN CR;
699   6                  END;
700   5              RETURN CHAR;
701   5              END;
                   ELSE
702   4              RETURN CHAR;
703   4          END; /* OF DO FOREVER */
704   3          END GETSOURCE;

705   2      RD$EOF: PROCEDURE BYTE;
                 /* RETURN TRUE IF END OF FILE */
706   3          CHAR = GETSOURCE;
707   3          IF obj THEN RETURN (endofsrc and (nsource > nsbuf));
709   3          RETURN (CHAR = ENDFILE);
710   3          END RD$EOF;
```

```
711   2       HEXRECORD: PROCEDURE;
712   3           DECLARE (h, hbuf, RL, CS, RT) BYTE,
                        zerorec byte,  /* true if last record had length of zero */
                        LOA ADDRESS;  /* LOAD ADDRESS WHICH FOLLOWS : */

713   3           ckhex: procedure byte;
714   4               IF H - '0' <= 9 THEN
715   4                   RETURN H-'0';
716   4               IF H - 'A' > 5 THEN
717   4                   CALL xerror(2,.source);  /* invalid hex digit */
718   4               RETURN H - 'A' + 10;
719   4               end ckhex;

720   3           rdhex: procedure byte;
721   4               call putdest(h := getsource);
722   4               return ckhex;
723   4               end rdhex;

724   3           RDCS: PROCEDURE BYTE;
                        /* READ BYTE WITH CHECKSUM */
725   4               RETURN CS := CS + (SHL(RDHEX,4) OR RDHEX);
726   4               END RDCS;

727   3           RDADDR: PROCEDURE ADDRESS;
                        /* READ DOUBLE BYTE WITH CHECKSUM */
728   4               RETURN SHL(DOUBLE(RDCS),8) OR RDCS;
729   4               END RDADDR;

                    /* READ HEX FILE AND CHECK EACH RECORD
                    FOR VALID DIGITS, AND PROPER CHECKSUM */
730   3           zerorec = false;
                    /* READ NEXT RECORD */
731   3           h = getsource;
732   3               do forever;
                        /* SCAN FOR THE ':' */
733   4                   DO WHILE h <> ':';
734   5                   IF (h = ENDFILE) THEN
735   5                       do; if zerorec then return;
738   6                       CALL xerror(3,.source);  /* unexpected end of hex file */
739   6                       end;
740   5                   call putdest(h);
741   5                   h = getsource;
742   5                   END;

                        /* ':' FOUND */
                        /* check for end of hex record */
743   4                   h = getsource;
744   4                   rl = shl(ckhex,4);
745   4                   hbuf = h; h = getsource;
747   4                   rl = rl or ckhex;
748   4                   if (rl = 0) then zerorec = true;
750   4                   else zerorec = false;
751   4                   if (zerorec and ignor) then
752   4                       do while (h <> ':') and (h <> endfile);
753   5                       h = getsource;
754   5                       end;
```

```
755   4               else do; call putdest(':');
757   5                   call putdest(hbuf);
758   5                   call putdest(h);
759   5                   cs = rl;
760   5                   LOA = LOADDR; /* LOAD ADDRESS */

                          /* READ WORDS UNTIL RECORD LENGTH EXHAUSTED */
761   5                   RT = RDCS; /* RECORD TYPE */
762   5                       DO WHILE RL <> 0; RL = RL - 1;
764   6                       hbuf = RDCS;
                              /* INCREMENT LA HERE FOR EXACT ADDRESS */
765   6                       END;

                          /* CHECK SUM */
766   5                   IF rdcs <> 0 THEN
767   5                       CALL xerror(4,.source);  /* hex record checksum */
768   5                   h = getsource;
769   5                   end;
770   4               end; /* do forever */
771   3           END HEXRECORD;

772   2       CK$STRINGS: PROCEDURE;
773   3           IF STARTS > 0 THEN
774   3               call error(11);  /* START NOT FOUND */
775   3           IF QUITS  > 0 THEN
776   3               call error(12);  /* QUIT NOT FOUND */
777   3           END CK$STRINGS;

778   2       CLOSEDEST: PROCEDURE;
779   3               DO WHILE (LOW(NDEST) AND 7FH) <> 0;
780   4               CALL PUTDEST(ENDFILE);
781   4               END;
782   3           CALL CK$STRINGS;
783   3           CALL WRITEDEST;
784   3           call setduser;  /* destination user */
785   3           CALL CLOSE(.DEST);
786   3           IF DCNT = 255 THEN
            $if mpm
787   3               call xerror(8,.dest);  /* CLOSE FILE */
788   3           IF odcnt <> 255 THEN /* FILE EXISTS */
789   3               do;
            $else
                    call xerror(8,.dest);  /* CLOSE FILE */
                call open(.odest);
                IF DCNT <> 255 THEN /* FILE EXISTS */
                    DO; call close(.odest);
            $endif
790   4               IF ROL(odest.fcb(9),1) THEN /* READ ONLY */
791   4                   DO;
792   5                   IF NOT WRROF THEN
793   5                       DO;
794   6                           do while ((dcnt <> 'Y') and (dcnt <> 'N'));
795   7                           CALL PRINT (.('DESTINATION IS R/O, DELETE (Y/N)? $'));
796   7                           dcnt = utran(rdchar);
797   7                           end;
798   6                       IF dcnt <> 'Y' THEN
799   6                           DO; CALL PRINT(.('**NOT DELETED**$'));
```

```
801   7                       CALL CRLF;
802   7                       CALL DELETE(.DEST);
803   7                       RETURN;
804   7                       END;
805   6                   CALL CRLF;
806   6                   END;
807   5               END;
                      /* reset r/o and sys attributes */
808   4               odest.fcb(9) = odest.fcb(9) and 7fh;
809   4               odest.fcb(10) = odest.fcb(10) AND 7FH;
810   4               CALL SETIND(.odest);
811   4               CALL DELETE(.odest);
812   4               END;
813   3           CALL MOVE(.odest.fcb,.dest.fcb(16),16); /* READY FOR RENAME */
814   3           CALL RENAME(.DEST);
                  /* set destination attributes same as source */
815   3           odest.fcb(1) = (odest.fcb(1) and 07fh) or f1;
816   3           odest.fcb(2) = (odest.fcb(2) and 07fh) or f2;
817   3           odest.fcb(3) = (odest.fcb(3) and 07fh) or f3;
818   3           odest.fcb(4) = (odest.fcb(4) and 07fh) or f4;
819   3           odest.fcb(8) = (odest.fcb(8) and 07fh);
820   3           odest.fcb(9) = (odest.fcb(9) and 07fh) or ro;
821   3           odest.fcb(10) = (odest.fcb(10) and 07fh) or sys;
822   3           odest.fcb(11) = (odest.fcb(11) and 07fh);
823   3           call setind(.odest);
824   3           if archiv then   /* set archive bit */
825   3               do; call setsuser;
827   4               source.fcb(11) = source.fcb(11) or 080h;
828   4               source.fcb(12) = 0;
829   4               call setind(.source);
830   4               end;
831   3           END CLOSEDEST;

832   2       SIZE$MEMORY: PROCEDURE;
                  /* SET UP SOURCE AND DESTINATION BUFFERS */
833   3           if not dblbuf then
834   3               do;  /* ABSORB THE SOURCE BUFFER INTO THE DEST BUFFER */
835   4               sbase = .memory;
836   4               sblen,dblen = ((maxb - .memory) and 0ff80h) - 128;
837   4               end;
838   3           else do;  /* may need to write destination buffer */
839   4               sblen,dblen = (shr((maxb - .memory),1) and 0ff80h) - 128;
840   4               sbase = .memory + dblen + 128;
841   4               if ndest >= dblen then call writedest;
843   4               nsbuf = 0;
844   4               end;
845   3           END SIZE$MEMORY;

846   2       setupeob: procedure;
                  /* sets nsbuf to end of source buffer */
847   3           declare i byte;
848   3           if (not obj) and (nsbuf <> 0) then
849   3               do; tblen = nsbuf - 128;
851   4                   do i = 0 to 128;
852   5                   if (sbuff(tblen + i)) = endfile then
853   5                       do; nsbuf = tblen + i;
855   6                       return;
```

```
856   6                     end;
857   5                 end;
858   4             end;
859   3         end setupeob;

860   2     SIMPLECOPY: PROCEDURE;
861   3         DECLARE 1 BYTE;
862   3         declare
                      fast lit '0',  /* fast file to file copy */
                      chrt lit '1',  /* character transfer option */
                      dubl lit '2';  /* double buffer required for file copy */
863   3         declare optype(26) byte data (
                  /* option type for each option character */
                      fast,  /* for A option */
                      fast,  /* for B option */
                      fast,  /* for C option */
                      dubl,  /* for D option */
                      chrt,  /* for E option */
                      dubl,  /* for F option */
                      fast,  /* for G option */
                      chrt,  /* for H option */
                      dubl,  /* for I option */
                      fast,  /* for J option */
                      fast,  /* for K option */
                      chrt,  /* for L option */
                      fast,  /* for M option */
                      dubl,  /* for N option */
                      fast,  /* for O option */
                      dubl,  /* for P option */
                      dubl,  /* for Q option */
                      fast,  /* for R option */
                      dubl,  /* for S option */
                      dubl,  /* for T option */
                      chrt,  /* for U option */
                      fast,  /* for V option */
                      fast,  /* for W option */
                      fast,  /* for X option */
                      fast,  /* for Y option */
                      chrt); /* for Z option */

864   3         chkrandom: procedure;
865   4             call setsuser;
866   4             call set$random(.source);
          $if mpm
867   4             call multsect(1);
          $endif
868   4             call setdma(.buff);
869   4                 do forever;
870   5                 if (((dcnt := rd$random(.source)) = 0) or maxsize) then
871   5                     do; destr = sourcer;
873   6                     destr2 = sourcer2;
874   6                     endofsrc = false;
875   6                     return;
876   6                     end;
877   5                 if dcnt = 1 then
878   5                     do; if (sourcer := sourcer + 1) = 0 then
880   6                         sourcer2 = sourcer2 + 1;
```

```
881   6                   end;
882   5               else if dcnt = 4 then
883   5                   do;
884   6                   if (sourcer := (sourcer + 128) and 0ff80h) = 0 then
885   6                       sourcer2 = sourcer2 + 1;
886   6                   end;
                      else
887   5                   call xerror(15,.source);
888   5               end;
889   4           end chkrandom;

890   3       fastcopy = (sfile and dfile);
891   3       endofsrc = false;
892   3       dblbuf = false;
893   3       sparfil = false;
894   3       insparc = false;
                  /* LOOK FOR PARAMETERS */
895   3           DO I = 0 TO 25;
896   4           IF CONT(I) <> 0 THEN
897   4               DO;
898   5               IF optype(i) = chrt THEN
899   5                   FASTCOPY = FALSE;
                    else
900   5                   if optype(i) = dubl then
901   5                       do; dblbuf = (sfile and dfile);
903   6                       fastcopy = false;
904   6                       end;
                    END;
906   4               END;

907   3       CALL SIZE$MEMORY;
908   3       if sfile then
909   3           CALL SETUPSOURCE;
                  /* FILES READY FOR COPY */

910   3       if fastcopy then
911   3           do while not endofsrc;
912   4           CALL FILLSOURCE;
913   4           if endofsrc and concat then
914   4               do; call setupeob;
916   5               ndest = nsbuf;
917   5               if nendcmd then return;
919   5               end;
920   4           ndest = nsbuf;
921   4           CALL WRITEDEST;
922   4           nsbuf = ndest;
923   4           if (endofsrc and insparc) then
924   4               call chkrandom;
925   4           end;

926   3       else do;
                  /* PERFORM THE ACTUAL COPY FUNCTION */
927   4           IF HEXT OR IGNOR THEN /* HEX FILE */
928   4               call hexrecord;
                  ELSE
929   4               DO WHILE NOT RD$EOF;
930   5               CALL PUTDEST(CHAR);
```

```
931   5                 END;
932   4               if concat and nendcmd then
933   4                   do; nsbuf = ndest;
935   5                   return;
936   5                   end;
937   4               end;

938   3           if dfile then
939   3               CALL CLOSEDEST;
940   3           END SIMPLECOPY;

941   2       MULTCOPY: PROCEDURE;
942   3           DECLARE (NEXTDIR, NDCNT, NCOPIED) ADDRESS;

943   3           PRNAME: PROCEDURE;
                    /* PRINT CURRENT FILE NAME */
944   4               DECLARE (I,C) BYTE;
945   4               CALL CRLF;
946   4                   DO I = 1 TO FNSIZE;
947   5                   IF (C := odest.fcb(I)) <> ' ' THEN
948   5                       DO; IF I = FEXT THEN CALL PRINTCHAR('.');
951   6                       CALL PRINTCHAR(C);
952   6                       END;
953   5                   END;
954   4               END PRNAME;

955   3           archck: procedure byte;
                    /* check if archive bit is set in any extent of source file */
956   4               if not archiv then
957   4                   return 1;
958   4               call setsuser;
959   4               source.fcb(12) = what;
960   4               call search(.source);
961   4                   do while dcnt <> 255;
962   5                   call move(.buff+shl(dcnt and 11b,5)+1,.source.fcb(1),15);
963   5                   if not rol(source.fcb(11),1) then
964   5                       return 1;
965   5                   call searchn;
966   5                   end;
967   4               return 0;
968   4               end archck;

            $if mpm
                /* initialize counters if not error retry */
969   3           if eretry = 0 then NEXTDIR, NCOPIED = 0;
            $else
                /* initialize counters */
                NEXTDIR, NCOPIED = 0;
            $endif

971   3                   DO FOREVER;
                        /* FIND A MATCHING ENTRY */
972   4                   CALL SETSUSER; /* SOURCE USER */
973   4                   CALL SETDMA(.BUFF);
974   4                   searfcb(12) = 0;
975   4                   CALL SEARCH(.SEARFCB);
976   4                   NDCNT = 0;
```

```
 977   4                DO WHILE (DCNT <> 255) AND NDCNT < NEXTDIR;
 978   5                NDCNT = NDCNT + 1;
 979   5                CALL SEARCHN;
 980   5                END;
                      /* FILE CONTROL BLOCK IN BUFFER */
 981   4                IF DCNT = 255 THEN
 982   4                    DO; IF NCOPIED = 0 THEN
 984   5                        call xerror(9,.searfcb);  /* file not found */
 985   5                    if not kilds then
 986   5                        CALL CRLF;
 987   5                    RETURN;
 988   5                    END;
 989   4                NEXTDIR = NDCNT + 1;
                      /* GET THE FILE CONTROL BLOCK NAME TO DEST */
 990   4                CALL MOVE(.BUFF + SHL(DCNT AND 11B,5)+1,.odest.fcb(1),15);
 991   4                CALL MOVE(.odest.fcb(1),.SOURCE.FCB(1),15); /* FILL BOTH FCB'S */
 992   4                if archck then
 993   4                    do; odest.fcb(12) = 0;
 995   5                    source.fcb(12) = 0;
 996   5                    IF RSYS OR NOT ROL(odest.fcb(10),1) THEN /* OK TO READ */
 997   5                        DO; if not kilds then    /* kill display option */
 999   6                    /       do; IF NCOPIED = 0 THEN
1001   7                               CALL PRINT(.('COPYING -$'));
1002   7                           dcnt = false;
1003   7                               do while ((dcnt <> 'Y') and (dcnt <> 'N'));
1004   8                               call prname;
1005   8                               if confrm then
1006   8                                   do; call printx(.(' (Y/N)? $'));
1008   9                                   dcnt = utran(rdchar);
1009   9                                   end;
                                        else
1010   8                                   dcnt = 'Y';
1011   8                               end;
1012   7                           end;
1013   6                       ncopied = ncopied + 1;
1014   6                       made = false;  /* destination file not made */
1015   6                       if (dcnt = 'Y') or (kilds) then
1016   6                           CALL SIMPLECOPY;
1017   6                       END;
1018   5                   end;
1019   4                END;
1020   3            END MULTCOPY;

1021   2        CK$DISK: PROCEDURE;
                      /* error if same user and same disk */
1022   3            IF (odest.user = source.user) and (odest.fcb(0) = source.fcb(0)) THEN
1023   3                CALL FORMERR;
1024   3            END CK$DISK;

1025   2        GNC: PROCEDURE BYTE;
1026   3            IF (CBP := CBP + 1) >= COMLEN THEN RETURN CR;
1028   3            RETURN UTRAN(COMBUFF(CBP));
1029   3            END GNC;

1030   2        DEBLANK: PROCEDURE;
1031   3                DO WHILE (CHAR := GNC) = ' ';
1032   4                END;
```

```
1033  3          END DEBLANK;

1034  2      CK$EOL: PROCEDURE;
1035  3          CALL DEBLANK;
1036  3          IF CHAR <> CR THEN CALL FORMERR;
1038  3          END CK$EOL;

1039  2      SCAN: PROCEDURE(FCBA);
1040  3          DECLARE FCBA ADDRESS,          /* ADDRESS OF FCB TO FILL */
                    fcbs based fcba structure (    /* FCB STRUCTURE */
                        fcb(frsize) byte,
              $if mpm
                        pwnam(nsize) byte,
                        pwmode byte,
              $endif
                        user byte,
                        type byte );
1041  3          DECLARE (I,K) BYTE;  /* TEMP COUNTERS */

                    /* SCAN LOOKS FOR THE NEXT DELIMITER, DEVICE NAME, OR FILE NAME.
                    THE VALUE OF CDP MUST BE 255 UPON ENTRY THE FIRST TIME */

1042  3          DELIMITER: PROCEDURE(C) BYTE;
1043  4              DECLARE (I,C) BYTE;
1044  4              DECLARE DEL(*) BYTE DATA
                      (' =.:;,<>',CR,LA,LB,RB);
1045  4                  DO I = 0 TO LAST(DEL);
1046  5                  IF C = DEL(I) THEN RETURN TRUE;
1048  5                  END;
1049  4              RETURN FALSE;
1050  4              END DELIMITER;

1051  3          PUTCHAR: PROCEDURE;
1052  4              FCBS.FCB(FLEN:=FLEN+1) = CHAR;
1053  4              IF CHAR = WHAT THEN AMBIG = TRUE; /* CONTAINS AMBIGUOUS REF */
1055  4              END PUTCHAR;

1056  3          FILLQ: PROCEDURE(LEN);
                    /* FILL CURRENT NAME OR TYPE WITH QUESTION MARKS */
1057  4              DECLARE LEN BYTE;
1058  4              CHAR = WHAT; /* QUESTION MARK */
1059  4                  DO WHILE FLEN < LEN;
1060  5                  CALL PUTCHAR;
1061  5                  END;
1062  4              END FILLQ;

1063  3          SCANPAR: PROCEDURE;
1064  4              DECLARE (I,J) BYTE;
                    /* SCAN OPTIONAL PARAMETERS */
1065  4              CHAR = GNC; /* SCAN PAST BRACKET */
1066  4                  DO WHILE NOT(CHAR = CR OR CHAR = RB);
1067  5                  IF (I := CHAR - 'A') > 25 THEN /* NOT ALPHA */
                            DO; IF CHAR = ' ' THEN
1070  6                            CHAR = GNC;
                            ELSE
1071  6                            call error(6);  /* BAD PARAMETER */
1072  6                        END;
```

```
                    ELSE
1073  5                 DO; /* SCAN PARAMETER VALUE */
1074  6                 IF CHAR = 'S' OR CHAR = 'Q' THEN
1075  6                     DO; /* START OR QUIT COMMAND */
1076  7                     J = CBP + 1; /* START OF STRING */
1077  7                         DO WHILE NOT ((CHAR := GNC) = ENDFILE OR CHAR = CR);
1078  8                         END;
1079  7                      CHAR=GNC;
1080  7                     END;
1081  6                 ELSE IF (J := (CHAR := GNC) - '0') > 9 THEN
1082  6                     J = 1;
                          ELSE
1083  6                     DO WHILE (K := (CHAR := GNC) - '0') <= 9;
1084  7                     J = J * 10 + K;
1085  7                     END;
1086  6                 CONT(I) = J;
1087  6                 IF I = 6 THEN /* SET SOURCE USER */
1088  6                     DO;
1089  7                     IF J > 15 THEN
1090  7                         call error(7);  /* INVALID USER NUMBER */
1091  7                     fcbs.user = J;
1092  7                     END;
1093  6                 END;
1094  5               END;
1095  4           CHAR = GNC;
1096  4           END SCANPAR;


                /* scan procedure entry point */

                /* INITIALIZE FILE CONTROL BLOCK TO EMPTY */
1097  3         fcbs.type = ERR; CHAR = ' '; FLEN = 0;
            $if mpm
1100  3             DO WHILE FLEN < (FRSIZE + NSIZE);
1101  4             IF FLEN = FNSIZE THEN CHAR = 0;
1103  4             ELSE IF FLEN = FRSIZE THEN CHAR = ' ';
                      call putchar;
1106  4             END;
1107  3         fcbs.pwnam(0) = 0;
1108  3         fcbs.pwmode = 1;
            $else
                      DO WHILE FLEN < FRSIZE -1;
                      IF FLEN = FNSIZE THEN CHAR = 0;
                      CALL PUTCHAR;
                      END;
            $endif
1109  3         fcbs.fcb(0) = cdisk +1;    /* initialize to current disk */
1110  3         fcbs.user = cuser;    /* and current user */
                /* CLEAR PARAMETERS */
1111  3             DO I = 0 TO 25; CONT(I) = 0;
1113  4             END;
1114  3         FEEDLEN,MATCHLEN,QUITLEN = 0;

                /* DEBLANK COMMAND BUFFER */
1115  3         CALL DEBLANK;

                /* CHECK PERIPHERALS AND DISK FILES */
```

```
                /* SCAN NEXT NAME */
1116    3           DO FOREVER;
1117    4           FLEN = 0;
1118    4               DO WHILE NOT DELIMITER(CHAR);
1119    5               IF FLEN >= NSIZE THEN /* ERROR, FILE NAME TOO LONG */
1120    5                   RETURN;
1121    5               IF CHAR = '*' THEN CALL FILL$(NSIZE);
1123    5               ELSE CALL PUTCHAR;
1124    5               CHAR = GNC;
1125    5               END;

                    /* CHECK FOR DISK NAME OR DEVICE NAME */
1126    4               IF CHAR = ':' THEN
1127    4                   DO; IF FLEN = 1 THEN
                            /* MAY BE DISK NAME A ... P */
1129    5                   DO;
1130    6                   IF (fcbs.fcb(0) := fcbs.fcb(1) - 'A' + 1) > 16 THEN
1131    6                       RETURN;  /* ERROR, INVALID DISK NAME */
1132    6                   CALL DEBLANK; /* MAY BE DISK NAME ONLY */
1133    6                   IF DELIMITER(CHAR) THEN
1134    6                       DO; IF CHAR = LB THEN
1136    7                           CALL SCANPAR;
1137    7                       CBP = CBP - 1;
1138    7                       fcbs.type = DISKNAME;
1139    7                       RETURN;
1140    7                       END;
1141    6                   END;
                        ELSE
                        /* MAY BE A THREE CHARACTER DEVICE NAME */
1142    5                   IF FLEN <> 3 THEN /* ERROR, CANNOT BE DEVICE NAME */
1143    5                       RETURN;
                        ELSE
                            /* LOOK FOR DEVICE NAME */
1144    5                   DO; DECLARE (I,J,K) BYTE, M LITERALLY '10',
                            IO(*) BYTE DATA
                            ('OUTPRNLSTAXO',
                              0,0,0,        /* fake area for file type */
                             'AUX',
                             'CONAXT1NPNULEOF',0);

1146    6                   J = 255;
1147    6                       DO K = 0 TO M;
1148    7                       I = 0;
1149    7                           DO WHILE ((I:=I+1) <= 3) AND
                                    IO(J+I) = fcbs.fcb(I);
1150    8                           END;
1151    7                       IF I = 4 THEN /* COMPLETE MATCH */
1152    7                           DO; fcbs.type = k;
                                    /* SCAN PARAMETERS */
1154    8                           IF GNC = LB THEN CALL SCANPAR;
1156    8                           CBP = CBP - 1;
1157    8                           RETURN;
1158    8                           END;
1159    7                       J = J + 3;  /* OTHERWISE TRY NEXT DEVICE */
1160    7                       END;
1161    6                   RETURN;  /* ERROR, NO DEVICE NAME MATCH */
1162    6                   END;
```

```
1163  5                 IF CHAR = LB THEN /* PARAMETERS FOLLOW */
1164  5                     CALL SCANPAR;
1165  5                 END;
                      ELSE
                          /* CHAR IS NOT ":", SO FILE NAME IS SET. SCAN REMAINDER */
1166  4                 DO; IF FLEN = 0 THEN /* ERROR, NO PRIMARY NAME */
1168  5                     RETURN;
1169  5                 FLEN = NSIZE;
1170  5                 IF CHAR = "." THEN /* SCAN FILE TYPE */
1171  5                     DO WHILE NOT DELIMITER(CHAR := GNC);
1172  6                     IF FLEN >= FNSIZE THEN
1173  6                         RETURN;  /* ERROR, TYPE FIELD TOO LONG */
1174  6                     IF CHAR = "*" THEN CALL FILLQ(FNSIZE);
1176  6                     ELSE CALL PUTCHAR;
1177  6                     END;
          $if mpm
1178  5                 FLEN = 0;
1179  5                 IF CHAR = ";" THEN  /* SCAN PASSWORD */
1180  5                     DO WHILE NOT DELIMITER(CHAR := GNC);
1181  6                     IF FLEN >= NSIZE THEN
1182  6                         /* ERROR, PW TOO LONG */ RETURN;
                          ELSE  /* SAVE PASSWORD */
1183  6                         FCBS.PWNAM(FLEN) = CHAR;
1184  6                         FLEN = FLEN + 1;
1185  6                     END;
          $endif
1186  5                 IF CHAR = LB THEN
1187  5                     CALL SCANPAR;
                          /* RESCAN DELIMITER NEXT TIME AROUND */
1188  5                 CBP = CBP - 1;
1189  5                 fcbs.type = FILE;
1190  5                 FCBS.FCB(32) = 0;
1191  5                 RETURN;
1192  5                 END;
1193  4             END;
1194  3         END SCAN;


          /* PLM (PIP) ENTRY POINT */
              /* BUFFER AT 80H CONTAINS REMAINDER OF LINE TYPED
              FOLLOWING THE COMMAND "PIP" - IF ZERO THEN PROMPT TIL CR */

1195  2         if not retry then
1196  2             do; CALL MOVE(.BUFF,.COMLEN,80H);
1198  3             MULTCOM = (COMLEN = 0);

                      /* GET CURRENT CP/M VERSION */
1199  3             IF low(CVERSION) < VERSION THEN
1200  3                 DO;
          $if cpm3
                      CALL PRINTC.("REQUIRES CP/M 3$"));
          $else
1201  4                 CALL PRINTC.("REQUIRES CONCURRENT CP/M-86$"));
          $endif
1202  4                 CALL BOOT;
1203  4                 END;
```

```
1204  3             call mon1(45,255);  /* set return error mode */

          $if cpm3
                call mon1(109,1);  /* set CP/M 3 control-C status mode */
          $endif

1205  3             if multcom then
1206  3                 do;
          $if cpm3
                      call printx(.('CP/M 3 PIP VERSION 3.1$'));
          $else
1207  4                 call printx(.('CONCURRENT CP/M-86 PIP VERSION 3.1$'));
          $endif
1208  4                 call crlf;
1209  4                 end;

1210  3             cuser,last$user = getuser;  /* GET CURRENT USER */
1211  3             cdisk = getdisk;            /* GET CURRENT DISK */
          $if mpm
1212  3             mseccnt = 1;
          $endif
1213  3             eretry = false; /* need to initialize here for first time */
1214  3             end;


              /* START HERE ON RESET EXIT FROM THE PROCEDURE 'ERROR' */
          $if mpm
1215  2         if eretry <> 0 then
1216  2             do; call multcopy;
1218  3             comlen = multcom;
1219  3             end;
          $endif
              /* MAIN PROCESSING LOOP.  PROCESS UNTIL CR ONLY */
1220  2             DO FOREVER;
1221  3             C1, C2, C3 = 0;      /* LINE COUNT = 000000 */
1222  3             CONCNT,COLUMN = 0;  /* PRINTER TABS */
1223  3             ndest,nsbuf = 0;
1224  3             ambig = false;
1225  3             made = false;        /* destination file not made */
1226  3             opened = false;      /* source file not opened */
1227  3             concat = false;
1228  3             eretry = false;
1229  3             PUTNUM = TRUE;       /* ACTS LIKE LF OCCURRED ON ASCII FILE */
1230  3             dfile,sfile = true;
1231  3             nendcmd = true;
1232  3             LINENO = 254;        /* INCREMENTED TO 255 > PAGCNT */
                  /* READ FROM CONSOLE IF NOT A ONELINER */
1233  3             IF MULTCON THEN
1234  3                 DO; CALL PRINTCHAR('*'); CALL RDCOM;
1237  4                 CALL CRLF;
1238  4                 END;
1239  3             CBP = 255;
1240  3             IF COMLEN = 0 THEN       /* character = <CR> */
1241  3                 do; call setcuser;  /* restore current user */
1243  4                 CALL BOOT;           /* normal exit from pip here */
1244  4                 end;
```

```
                    /* LOOK FOR SPECIAL CASES FIRST */

1245   3            CALL SCAN(.odest);
1246   3            if ambig then
1247   3                call xerror(5,.odest);  /* invalid destination */
1248   3            call deblank;  /* check for equal sign or left arrow */
1249   3            if (char <> '=') and (char <> 1a) then call formerr;
1251   3            call scan(.source);

1252   3            IF odest.type = DISKNAME THEN
1253   3                DO;
1254   4                IF source.type <> file then call formerr;
1256   4                CALL CK$EOL;
1257   4                CALL CK$DISK;
1258   4                odest.type = file;  /* set for character transfer */
                        /* MAY BE MULTI COPY */
1259   4                IF AMBIG THEN  /* FORM IS A:=B:AFN */
1260   4                    DO;
1261   5                    CALL MOVE(.source.fcb(0),.searfcb(0),frsize);
1262   5                    CALL MULTCOPY;
1263   5                    END;
1264   4                ELSE DO;  /* FORM IS A:=B:UFN */
1265   5                    CALL MOVE(.source.fcb(1),.odest.fcb(1),frsize - 1);
1266   5                    CALL SIMPLECOPY;
1267   5                    END;
1268   4                END;

1269   3            else IF (odest.type = FILE) and (source.type = DISKNAME) THEN
1270   3                    DO;
1271   4                    CALL CK$EOL;
1272   4                    CALL CK$DISK;
1273   4                    source.type = file;  /* set for character transfer */
        $if mpm
1274   4                    call move(.odest.fcb(1),.source.fcb(1),(frsize+nsize));
        $else
                            call move(.odest.fcb(1),.source.fcb(1),(frsize - 1));
        $endif
1275   4                    CALL SIMPLECOPY;
1276   4                    END;

1277   3            else if (odest.type > cons) then
1278   3                call error(3);  /* invalid destination */
1279   3            else do;
1280   4                IF odest.type <> FILE THEN dfile = false;
        $if not mpm
                        /* no conditional attach list device */
        $else
1282   4                if (odest.type = prnt or odest.type = lstt) then
1283   4                    if conatlst = 255 then
1284   4                        call error(21);  /* printer busy */
        $endif
                        /* SCAN AND COPY UNTIL CR */
1285   4                    DO WHILE nendcmd;
1286   5                    sfile = true;
1287   5                    call deblank;
1288   5                    IF (CHAR <> ',' AND CHAR <> CR) THEN
1289   5                        call error(16);  /* invalid separator */
```

```
1290   5                    concat = concat or (nendcmd := (char = ','));
1291   5                    IF odest.type = PRNT THEN
1292   5                        DO; NUMB = 1;
1294   6                        IF TABS = 0 THEN TABS = 8;
1296   6                        IF PAGCNT = 0 THEN PAGCNT = 1;
1298   6                        END;
1299   5                    IF (source.type < file) or (source.type > eoft) or ambig THEN
1300   5                        call error(4);  /* invalid source */
1301   5                    IF source.type <> FILE THEN  /* NOT A SOURCE FILE */
1302   5                        sfile = false;
1303   5                    IF source.type = NULT THEN
                                /* SEND 40 NULLS TO OUTPUT DEVICE */
1304   5                        DO sfile = 0 TO 39; CALL PUTDEST(0);
1306   6                        END;
1307   5                    ELSE IF source.type = EOFT THEN
1308   5                        CALL PUTDEST(ENDFILE);
1309   5                    else call simplecopy;

1310   5                    CALL CK$STRINGS;
                            /* READ ENDFILE, GO TO NEXT SOURCE */

1311   5                    if nendcmd then call scan(.source);
1313   5                    END;
1314   4                end;

                        /* COMLEN SET TO 0 IF NOT PROCESSING MULTIPLE COMMANDS */
1315   3                COMLEN = MULTCOM;

1316   3                END; /* DO FOREVER */
1317   2            end plm;
1318   1            END;
```

CROSS-REFERENCE LISTING

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES
52	0004H	2	A. . . . . WORD PARAMETER AUTOMATIC 53 54
10	0000H	2	A. . . . . WORD PARAMETER 11
68	0004H	2	A. . . . . WORD PARAMETER AUTOMATIC 69 70
184	0000H	1	A. . . . . BYTE BASED(S) 186
13	0000H	2	A. . . . . WORD PARAMETER 14
33	0004H	2	A. . . . . WORD PARAMETER AUTOMATIC 34 35 36
56	0004H	2	A. . . . . WORD PARAMETER AUTOMATIC 57 59
16	0000H	2	A. . . . . WORD PARAMETER 17
27	008BH	1	AMBIG. . . . . BYTE 300 302 1054 1224 1246 1259 1299
955	15F7H	84	ARCHCK . . . . . PROCEDURE BYTE STACK=0012H 992
31	0158H	1	ARCHIV . . . . . BYTE AT 824 956
22			AUXT . . . . . LITERALLY
631	0E1CH		AXICASE. . . . . LABEL 626
22			AXIT . . . . . LITERALLY
500	08C1H		AXOCASE. . . . . LABEL 507
22			AXOT . . . . . LITERALLY
531	0004H	1	B. . . . . BYTE PARAMETER AUTOMATIC 532 533 535
597	0004H	1	B. . . . . BYTE PARAMETER AUTOMATIC 598 599 600 601
654	0004H	1	B. . . . . BYTE PARAMETER AUTOMATIC 655 656 658
184	0000H	1	B. . . . . BYTE BASED(D) 186
559	0004H	1	B. . . . . BYTE PARAMETER AUTOMATIC 560 563 568 585 587 588
591	0004H	1	B. . . . . BYTE PARAMETER AUTOMATIC 592 593 594 595
483	0004H	1	B. . . . . BYTE PARAMETER AUTOMATIC 484 485 495 497 498 499 504 508
511	0004H	1	B. . . . . BYTE PARAMETER AUTOMATIC 512 513 514 516 528
5	0000H	1	B. . . . . BYTE PARAMETER 6
604	0181H	1	B. . . . . BYTE 623 629 632 638 645 647 649 650
38	029DH	14	BOOT . . . . . PROCEDURE STACK=0008H 1202 1243
3	0000H	128	BUFF . . . . . BYTE ARRAY(128) EXTERNAL(3) 417 424 868 962 973 990 1197
26	000AH	2	BUFSIZE. . . . . WORD 393 397 403 404 410 440 444 452 453 480
655	0184H	1	C. . . . . BYTE 656 661
944	018EH	1	C. . . . . BYTE 947 951
1042	0004H	1	C. . . . . BYTE PARAMETER AUTOMATIC 1043 1046
32	0175H	1	C1 . . . . . BYTE 546 551 1221
32	0174H	1	C2 . . . . . BYTE 547 550 1221
32	0173H	1	C3 . . . . . BYTE 548 549 1221
29	0155H	1	CBP. . . . . BYTE 1026 1028 1076 1137 1156 1188 1239
29	0003H	130	CBUFF. . . . . BYTE ARRAY(130) 29
26	0025H	1	CDISK. . . . . BYTE 1109 1211
27	00C7H	1	CHAR . . . . . BYTE 706 709 930 1031 1036 1052 1053 1058 1065 1066 1067 1069 1070 1074 1077 1079 1081 1083 1095 1098 1102 1104 1118 1121 1124 1126 1133 1135 1163 1170 1171 1174 1179 1180 1183 1186 1249 1288 1290
44	0004H	1	CHAR . . . . . BYTE PARAMETER AUTOMATIC 45 46
653	0183H	1	CHAR . . . . . BYTE 658 661 676 678 680 700 702
864	13F2H	127	CHKRANDOM. . . . . PROCEDURE STACK=0026H 924
862			CHRT . . . . . LITERALLY 863 898
1021	164BH	26	CKDISK . . . . . PROCEDURE STACK=0028H 1257 1272
1034	1699H	18	CKEOL. . . . . PROCEDURE STACK=0028H 1256 1271
713	1096H	42	CKHEX. . . . . PROCEDURE BYTE STACK=0026H 722 744 747
772	1103H	31	CKSTRINGS. . . . . PROCEDURE STACK=0024H 732 1310

```
  89  0382H   20  CLOSE. . . . . PROCEDURE STACK=0000H          243  249  336  473  785
 778  1122H  315  CLOSEDEST. . . . PROCEDURE STACK=0050H          939
 171  0004H    1  CNT. . . . . BYTE PARAMETER AUTOMATIC        172  173  174
  24          CNTRLC . . . . . LITERALLY        310
  26  001FH    1  COLUMN . . . . . BYTE        487  490  518  529 1222
  29  00D5H  128  COMBUFF. . . . . BYTE ARRAY(128) AT         655  659  676 1028
  29  00D4H    1  COMLEN . . . . . BYTE AT          252 1026 1197 1199 1218 1240 1315
 179  0535H   15  CONATLST . . . . . PROCEDURE BYTE STACK=0008H         1283
 308  0715H   39  CONBRK . . . . . PROCEDURE STACK=0024H          437  641
  27  00BAH    1  CONCAT . . . . . BYTE         469  913  932 1227 1290
 604  0182H    1  CONCHK . . . . . BYTE         605  628  634  637  638  639
  27  00C6H    1  CONCNT . . . . . BYTE         637 1222
  31  015AH    1  CONFRM . . . . . BYTE AT         1005
  22          CONS . . . . . LITERALLY        1277
  31  0158H   26  CONT . . . . . BYTE ARRAY(26)          31  896 1086 1112
  21  0058H   34  COPYRIGHT. . . . BYTE ARRAY(34) DATA
  24          CR . . . . . LITERALLY         49  528  698 1027 1036 1044 1066 1077 1288
  48  02CFH   17  CRLF . . . . . PROCEDURE STACK=000EH            58  260  298  801  805  945  986 1208
                  1237
 712  0188H    1  CS . . . . . BYTE         725  759
  30  0156H    1  CUSER. . . . . BYTE         145 1110 1210
  65  0314H   15  CVERSION . . . . PROCEDURE WORD STACK=0008H         1199
 182  0006H    2  D. . . . . WORD PARAMETER AUTOMATIC        183  184  186  188
 537  0004H    1  D. . . . . BYTE PARAMETER AUTOMATIC        538  539  540
 376  017DH    1  DATAOK . . . . . BYTE         419  423  424  428
  27  00B9H    1  DBLBUF . . . . . BYTE         614  833  892  902
  26  0002H    2  DBLEN. . . . . WORD         502  836  839  840  841
  26  0000H 1024  DBUFF. . . . . BYTE ARRAY(1024) AT         386  406  424  434  504
  28  00D2H    1  DCNT . . . . . BYTE          35   82   85  156  161  266  334  340  343  355  358  359
                                                369  408  458  460  786  794  796  798  870  877  882  961  962  977
                                                981  990 1002 1003 1008 1010 1015
1030  168AH   15  DEBLANK. . . . . PROCEDURE STACK=000CH         1035 1115 1132 1248 1287
                  DEC. . . . . BUILTIN          546  547  548
1044  0340H   12  DEL. . . . . BYTE ARRAY(12) DATA       1045 1046
  31  015BH    1  DELET. . . . . BYTE AT          488  490
 100  03BDH   26  DELETE . . . . . PROCEDURE STACK=0016H          250  339  802  811
1042  18E4H   46  DELIMITER. . . . PROCEDURE BYTE STACK=0004H         1118 1133 1171 1180
  26  0055H   47  DEST . . . . . STRUCTURE          26  249  250  330  331  339  341  342  345  346  347
                                                387  388  390  407  409  419  429  431  433  785  787  802  813  814
  26  0076H    2  DESTR. . . . . WORD AT          420  872
  26  0078H    1  DESTR2 . . . . . BYTE AT          421  873
  27  00BCH    1  DFILE. . . . . BYTE         614  890  902  938 1230 1281
  22          DISKNAME . . . . . LITERALLY       1138 1252 1269
 105  03D7H   20  DISKRD . . . . . PROCEDURE STACK=000AH          431  457
 109  03EBH   20  DISKWRITE. . . . PROCEDURE STACK=000AH          407
                  DOUBLE . . . . . BUILTIN          463  728
 862          DUBL . . . . . LITERALLY        863  900
  31  015CH    1  ECHO . . . . . BYTE AT          494
  20          ENDFILE. . . . . LITERALLY        477  568  656  670  680  681  709  734  752  780  852
                                                1077 1308
  27  00C0H    1  ENDOFSRC . . . . . BYTE         476  708  874  891  911  913  923
  22          EOFT . . . . . LITERALLY       1299 1307
 191  007AH   10  ER00 . . . . . BYTE ARRAY(10) DATA        214
 192  0084H   11  ER01 . . . . . BYTE ARRAY(11) DATA        214
 193  008FH    7  ER02 . . . . . BYTE ARRAY(7) DATA         214
 194  0096H   20  ER03 . . . . . BYTE ARRAY(20) DATA        214
 195  00AAH   15  ER04 . . . . . BYTE ARRAY(15) DATA        214
```

```
  196  00B9H    13  ER05 . . . . . .  BYTE ARRAY(13) DATA          214
  197  00C6H    14  ER06 . . . . . .  BYTE ARRAY(14) DATA          214
  198  00D4H    20  ER07 . . . . . .  BYTE ARRAY(20) DATA          214
  199  00E8H    15  ER08 . . . . . .  BYTE ARRAY(15) DATA          214
  200  00F7H    20  ER09 . . . . . .  BYTE ARRAY(20) DATA          214
  201  0108H    15  ER10 . . . . . .  BYTE ARRAY(15) DATA          214
  202  011AH    16  ER11 . . . . . .  BYTE ARRAY(16) DATA          214
  203  012AH    15  ER12 . . . . . .  BYTE ARRAY(15) DATA          214
  204  0139H    18  ER13 . . . . . .  BYTE ARRAY(18) DATA          214
  205  0148H    11  ER14 . . . . . .  BYTE ARRAY(11) DATA          214
  206  0156H    27  ER15 . . . . . .  BYTE ARRAY(27) DATA          214
  207  0171H    18  ER16 . . . . . .  BYTE ARRAY(18) DATA          214
  208  0183H    19  ER17 . . . . . .  BYTE ARRAY(19) DATA          214
  209  0196H    32  ER18 . . . . . .  BYTE ARRAY(32) DATA          214
  210  01B6H    10  ER19 . . . . . .  BYTE ARRAY(10) DATA          214
  211  01C0H    10  ER20 . . . . . .  BYTE ARRAY(10) DATA          214
  212  01CAH    13  ER21 . . . . . .  BYTE ARRAY(13) DATA          214
  213  01D7H    23  ER22 . . . . . .  BYTE ARRAY(23) DATA          214
   28  00D1H     1  ERETRY . . . . . .  BYTE          239  300  302  969 1213 1215 1228
   22             ERR. . . . . . .  LITERALLY       1097
  214  0000H    46  ERRMSG . . . . . .  WORD ARRAY(23) DATA          259  272
  256  0588H    26  ERROR. . . . . .  PROCEDURE STACK=0020H          306  311  500  609  774  776 1071 1090
                                      1278 1284 1289 1300
  237  0569H    79  ERRORCLEANUP . . .  PROCEDURE STACK=001AH          258  268
  256  0004H     1  ERRTYPE. . . . . .  BYTE PARAMETER AUTOMATIC      257  259
  226  029EH     1  EX00 . . . . . .  BYTE ARRAY(1) DATA          236
  227  029FH    15  EX01 . . . . . .  BYTE ARRAY(15) DATA          236
  228  02AEH     9  EX02 . . . . . .  BYTE ARRAY(9) DATA          236
  229  02B7H     9  EX03 . . . . . .  BYTE ARRAY(9) DATA          236
  230  02C0H    20  EX04 . . . . . .  BYTE ARRAY(20) DATA          236
  231  02D4H    18  EX05 . . . . . .  BYTE ARRAY(18) DATA          236
  232  02E6H    17  EX07 . . . . . .  BYTE ARRAY(17) DATA          236
  233  02F7H    15  EX08 . . . . . .  BYTE ARRAY(15) DATA          236
  234  0306H    15  EX10 . . . . . .  BYTE ARRAY(15) DATA          236
   28  00CFH     1  EXTEN. . . . . .  BYTE           36   86  267  337  340  344  360  463
  236  0042H    22  EXTMSG . . . . . .  WORD ARRAY(11) DATA          282
   26  00B6H     1  EXTSAVE. . . . . .  BYTE          456  464
   16  0000H     1  F. . . . . .  BYTE PARAMETER          17
   13  0000H     1  F. . . . . .  BYTE PARAMETER          14
   10  0000H     1  F. . . . . .  BYTE PARAMETER          11
   28  00C9H     1  F1 . . . . . .  BYTE          363  815
   28  00CAH     1  F2 . . . . . .  BYTE          364  816
   28  00CBH     1  F3 . . . . . .  BYTE          365  817
   28  00CCH     1  F4 . . . . . .  BYTE          366  818
   24             FALSE. . . . . .  LITERALLY        244  315  319  323  474  582  628  664  730  750  874
                                       891  892  893  894  899  903 1002 1014 1049 1213 1224 1225 1226 1227
                                      1228 1281 1302
  862             FAST . . . . . .  LITERALLY        863
   27  00B8H     1  FASTCOPY . . . . .  BYTE          391  401  438  450  462  469  890  899  903  910
  109  0004H     2  FCB. . . . . .  WORD PARAMETER AUTOMATIC       110  111
  105  0004H     2  FCB. . . . . .  WORD PARAMETER AUTOMATIC       106  107
  100  0004H     2  FCB. . . . . .  WORD PARAMETER AUTOMATIC       101  102  103
   93  0004H     2  FCB. . . . . .  WORD PARAMETER AUTOMATIC        94   95
   89  0004H     2  FCB. . . . . .  WORD PARAMETER AUTOMATIC        90   91
   74  0000H    36  FCB. . . . . .  BYTE ARRAY(36) MEMBER(FCBS)
    3  0000H    33  FCB. . . . . .  BYTE ARRAY(33) EXTERNAL(2)          974  975  984 1261
  264  0000H    33  FCB. . . . . .  BYTE BASED(FILEADR) ARRAY(33)       287  290
```

```
 167  0004H     2  FCB. . . . . .  WORD PARAMETER AUTOMATIC          168  169
  79  0000H    36  FCB. . . . . .  BYTE BASED(FCBA) ARRAY(36)        82
 115  0000H    36  FCB. . . . . .  BYTE ARRAY(36) MEMBER(FCBS)       117  119
 124  0004H     2  FCB. . . . . .  WORD PARAMETER AUTOMATIC          125  126  127
  26  0000H    36  FCB. . . . . .  BYTE ARRAY(36) MEMBER(DDEST)      332  790  808  809  813  815  816
                                    817  818  819  820  821  822    947  990  991  994  995 1022 1265 1274
  26  0000H    36  FCB. . . . . .  BYTE ARRAY(36) MEMBER(DEST)       26  331  347  813
 163  0004H     2  FCB. . . . . .  WORD PARAMETER AUTOMATIC          164  165
 158  0004H     2  FCB. . . . . .  WORD PARAMETER AUTOMATIC          159  160
 153  0004H     2  FCB. . . . . .  WORD PARAMETER AUTOMATIC          154  155
1040  0000H    36  FCB. . . . . .  BYTE ARRAY(36) MEMBER(FCBS)      1052 1109 1130 1149 1190
  26  0000H    36  FCB. . . . . .  BYTE ARRAY(36) MEMBER(SOURCE)     26  314  316  318  320  322  353
                                    357  363  364  365  366  367   368  370  371  372  456  464  465  827
                                    828  959  962  963  991  995  1022 1261 1265 1274
 132  0004H     2  FCB. . . . . .  WORD PARAMETER AUTOMATIC          133  134
1039  001CH     2  FCBA . . . . .  WORD PARAMETER      1040 1052 1091 1097 1107 1108 1109 1110 1130 1138
                                    1149 1153 1183 1189 1190
  77  0004H     2  FCBA . . . . .  WORD PARAMETER AUTOMATIC           78   79   80   81   82   84
  72  0004H     2  FCBA . . . . .  WORD PARAMETER AUTOMATIC           73   74   75
 113  0004H     2  FCBA . . . . .  WORD PARAMETER AUTOMATIC          114  115  116  117  119  120  122
  74  0000H    44  FCBS . . . . .  STRUCTURE BASED(FCBA)            75
1040  0000H    47  FCBS . . . . .  STRUCTURE BASED(FCBA)           1052 1091 1097 1107 1108 1109 1110 1130
                                    1138 1149 1153 1183 1189 1190
 115  0000H    44  FCBS . . . . .  STRUCTURE BASED(FCBA)           116  117  119  120
  26  0021H     1  FEEDBASE . . . .  BYTE        676  677  686
  26  0022H     1  FEEDLEN. . . . .  BYTE        673  675  688 1114
  22             FEXT . . . . . .  LITERALLY       292  331  949
  22             FEXTL. . . . . .  LITERALLY       331
  22             FF . . . . . .  LITERALLY       563  568  577  585
  22             FILE . . . . . .  LITERALLY      1189 1254 1258 1269 1273 1280 1299 1301
 263  0004H     2  FILEADR. . . . .  WORD PARAMETER AUTOMATIC       264  285  287  290
1056  1937H    25  FILLQ. . . . . .  PROCEDURE STACK=0003H        1122 1175
 436  0A5CH   273  FILLSOURCE . . . .  PROCEDURE STACK=0028H         621  912
  26  00B3H     3  FILSIZE. . . . .  BYTE ARRAY(3)        314  316  318  320  322  370
  27  00C8H     1  FLEN . . . . . .  BYTE      1052 1059 1099 1100 1101 1103 1117 1119 1128 1142 1167 1169
                                    1172 1178 1181 1183 1184
 176  0526H    15  FLUSHBUF . . . .  PROCEDURE STACK=0008H         414
  22             FNSIZE . . . . .  LITERALLY       289  946 1101 1172 1175
  24             FOREVER. . . . .  LITERALLY       672  732  869  971 1116 1220
 305  070AH    11  FORMERR. . . . .  PROCEDURE STACK=0024H        1023 1037 1250 1255
  31  015DH     1  FORMF. . . . . .  BYTE AT         561
  22             FRSIZE . . . . .  LITERALLY        26   74   79  115  330 1040 1100 1103 1261 1265 1274
  22             FSIZE. . . . . .  LITERALLY       264
 263  0006H     1  FUNCNO . . . . .  BYTE PARAMETER AUTOMATIC       264  269  272  273
 129  044AH    15  GETDISK. . . . .  PROCEDURE BYTE STACK=0008H      1211
 652  0E95H   160  GETSOURCE. . . . .  PROCEDURE BYTE STACK=0032H     706  721  731  741  743  746  753
                                    768
 603  0DA7H   238  GETSOURCEC . . . .  PROCEDURE BYTE STACK=002EH     680
  31  015EH     1  GETU . . . . . .  BYTE AT
 136  0460H    15  GETUSER. . . . .  PROCEDURE BYTE STACK=0008H      1210
1025  1665H    37  GNC. . . . . .  PROCEDURE BYTE STACK=0008H      1031 1065 1070 1077 1079 1081 1083
                                    1095 1124 1154 1171 1180
 712  0185H     1  H. . . . . . .  BYTE        714  715  716  718  721  731  733  734  740  741  743  745
                                     746  752  753  758  768
 712  0186H     1  HBUF . . . . . .  BYTE        745  757  764
 711  0FA5H   241  HEXRECORD. . . . .  PROCEDURE STACK=0060H        928
  31  015FH     1  HEXT . . . . . .  BYTE AT         927
```

```
                      HIGH . . . . . . . . .   BUILTIN          36
1064  0192H       1   I. . . . . . . . . . .   BYTE       1067 1086 1087
1043  0191H       1   I. . . . . . . . . . .   BYTE       1045 1046
1041  018FH       1   I. . . . . . . . . . .   BYTE       1111 1112
 944  018DH       1   I. . . . . . . . . . .   BYTE        946  947  949
 847  018BH       1   I. . . . . . . . . . .   BYTE        851  852  854
 560  0180H       1   I. . . . . . . . . . .   BYTE        570  572  573  574
 512  017EH       1   I. . . . . . . . . . .   BYTE        518  519  520  522  523  524
 351  017AH       1   I. . . . . . . . . . .   BYTE
 264  0177H       1   I. . . . . . . . . . .   BYTE        289  290  292
1145  0194H       1   I. . . . . . . . . . .   BYTE       1148 1149 1151
 861  018CH       1   I. . . . . . . . . . .   BYTE        895  896  898  900
  31  0160H       1   IGNOR. . . . . . . . .   BYTE AT          751  927
   8  0000H           INPD . . . . . . . . .   PROCEDURE BYTE EXTERNAL(5) STACK=0000H          632
  22                  INPT . . . . . . . . .   LITERALLY
  27  00C2H       1   INSPARC. . . . . . . .   BYTE        383  467  894  923
1145  034CH      34   IO . . . . . . . . . .   BYTE ARRAY(34) DATA        1149
1064  0193H       1   J. . . . . . . . . . .   BYTE       1076 1081 1082 1084 1086 1089 1091
1145  0195H       1   J. . . . . . . . . . .   BYTE       1146 1149 1159
 376  017CH       1   J. . . . . . . . . . .   BYTE        422  423  424  425
 351  017BH       1   J. . . . . . . . . . .   BYTE
1145  0196H       1   K. . . . . . . . . . .   BYTE       1147 1153
1041  0190H       1   K. . . . . . . . . . .   BYTE       1083 1084
  31  0162H       1   KILDS. . . . . . . . .   BYTE AT          985  998 1015
  22                  LA . . . . . . . . . .   LITERALLY       1044 1249
                      LAST . . . . . . . . .   BUILTIN         1045
  30  0157H       1   LASTUSER . . . . . . .   BYTE        141  142 1210
  22                  LB . . . . . . . . . .   LITERALLY       1044 1135 1154 1163 1186
 712  0014H       2   LDA. . . . . . . . . .   WORD        760
1056  0004H       1   LEN. . . . . . . . . .   BYTE PARAMETER AUTOMATIC        1057 1059
  24                  LF . . . . . . . . . .   LITERALLY         50  588  638  669
  26  0020H       1   LINENO . . . . . . . .   BYTE        574  576  586 1232
  22                  LIT. . . . . . . . . .   LITERALLY         22   23   24   25  224  235  862
                      LOW. . . . . . . . . .   BUILTIN           35  404  434  453  779 1199
  31  0163H       1   LOWER. . . . . . . . .   BYTE AT          648
  22                  LPP. . . . . . . . . .   LITERALLY        573
  22                  LSTT . . . . . . . . .   LITERALLY       1282
 597  0D8DH      26   LTRAN. . . . . . . . .   PROCEDURE BYTE STACK=0004H          649
1145                  M. . . . . . . . . . .   LITERALLY       1147
  27  00BEH       1   MADE . . . . . . . . .   BYTE        246  348  377 1014 1225
 113  03FFH      49   MAKE . . . . . . . . .   PROCEDURE STACK=0010H          342
 654  0F35H      66   MATCH. . . . . . . . .   PROCEDURE BYTE STACK=0004H          684  694
  26  0023H       1   MATCHLEN . . . . . . .   BYTE        656  662  663  688  689 1114
   3  0000H       2   MAXB . . . . . . . . .   WORD EXTERNAL(1)          836  839
  29  00D3H       1   MAXLEN . . . . . . . .   BYTE AT           62   63
  25                  MAXMBUF. . . . . . . .   LITERALLY        393  401  440  450
  25                  MAXMCNT. . . . . . . .   LITERALLY        394  441
 313  073CH      56   MAXSIZE. . . . . . . .   PROCEDURE BYTE STACK=0002H          467  870
      0000H           MEMORY . . . . . . . .   BYTE ARRAY(0)         26  835  836  839  840
 265  0315H      17   MESSAGEINDEXTBL. . . .   BYTE ARRAY(17) DATA        272
  10  0000H           MON1 . . . . . . . . .   PROCEDURE EXTERNAL(6) STACK=0000H          39   46   54   63   70   84
                                                 142  169  174  177  495  498  499  508 1204
  13  0000H           MON2 . . . . . . . . .   PROCEDURE BYTE EXTERNAL(7) STACK=0000H          42  130  137  165  180
                                                 309  310  629
  16  0000H           MON3 . . . . . . . . .   PROCEDURE WORD EXTERNAL(8) STACK=0000H          66   81   91   95   98
                                                 103  107  111  122  127  134  155  160
 182  0544H      37   MOVE . . . . . . . . .   PROCEDURE STACK=0008H          330  331  370  434  813  962  990  991
```

```
                                    1197 1261 1265 1274
   26  0087H     1  MSECCNT. . . . . BYTE          173  174 1217
   27  00C4H     1  MULTCOM. . . . . BYTE         1198 1205 1218 1233 1315
  941  1471H   330  MULTCOPY . . . . . PROCEDURE STACK=0068H          1217 1262
  171  050AH    28  MULTSECT . . . . . PROCEDURE STACK=000AH           238  385  394  398  404  416  441  445
                                     453  867
  376  0012H     2  N. . . . . . . . WORD          379  400  401  403  418
  182  0004H     1  N. . . . . . . . BYTE PARAMETER AUTOMATIC         183  185
  942  001AH     2  NCOPIED. . . . . WORD          970  983 1000 1013
  942  0018H     2  NDCNT. . . . . . WORD          976  977  978  989
   26  000EH     2  NDEST. . . . . . WORD          379  434  502  504  505  619  779  841  916  920  922  934
                                    1223
   27  00C1H     1  NENDCMD. . . . . BYTE          917  932 1231 1285 1290 1311
  542  0C98H    90  NEWLINE. . . . . PROCEDURE STACK=0046H           581
  942  0016H     2  NEXTDIR. . . . . WORD          970  977  989
  609  0DBCH        NOTSOURCE. . . . LABEL         607  608  610  631
   26  0008H     2  NSBUF. . . . . . WORD          448  449  450  452  455  463  477  480  615  619  708  843
                                     848  850  854  916  920  922  934 1223
   22                NSIZE. . . . . . LITERALLY        26   74  115  330 1040 1100 1119 1122 1169 1181 1274
   26  000CH     2  NSOURCE. . . . . WORD          373  448  612  616  623  624  708
   22                NULT . . . . . . LITERALLY      1303
   31  0165H     1  NUMB . . . . . . BYTE AT         545  552  580 1293
  235                NUMMSGS. . . . . LITERALLY      236  279
  224                NUMSPMSGS. . . . LITERALLY      225  273
   31  0166H     1  OBJ. . . . . . . BYTE AT         636  707  848
   28  00D0H     1  ODCNT. . . . . . BYTE          334  335  788
   26  0084H    47  ODEST. . . . . . STRUCTURE       148  330  332  333  336  338  496  790  808  809  810
                                     811  813  815  816  817  818  819  820  821  822  823  947  990  991
                                     994  996 1022 1245 1247 1252 1258 1265 1269 1274 1277 1280 1282 1291
  543  017FH     1  ONE. . . . . . . BYTE          544  546
   77  0344H    62  OPEN . . . . . . PROCEDURE STACK=0016H           333  354
   27  00BFH     1  OPENED . . . . . BYTE          240  244  356  474 1226
  863  0326H    26  OPTYPE . . . . . BYTE ARRAY(26) DATA         898  900
    5  0000H        OUTD . . . . . . PROCEDURE EXTERNAL(4) STACK=0000H         497
   22                OUTT . . . . . . LITERALLY
   31  0167H     1  PAGCNT . . . . . BYTE AT         570 1296 1297
    1  0000H        PIPMOD . . . . . PROCEDURE STACK=0000H
   19  0000H   648  PLM. . . . . . . PROCEDURE PUBLIC STACK=006CH
   56  02F0H    16  PRINT. . . . . . PROCEDURE STACK=0014H           254  795  800 1001 1201
  531  0C55H    40  PRINT1 . . . . . PROCEDURE STACK=003CH           539  540
   44  02BAH    21  PRINTCHAR. . . . PROCEDURE STACK=000AH            49   50  275  281  287  288  293  294
                                     950  951 1235
  537  0C7DH    27  PRINTDIG . . . . PROCEDURE STACK=0042H           549  550  551
   52  02E0H    16  PRINTX . . . . . PROCEDURE STACK=000AH            59  259  272  276  282  284 1007 1207
  943  15BBH    60  PRNAME . . . . . PROCEDURE STACK=0012H          1004
   22                PRNT . . . . . . LITERALLY       1282 1291
 1051  1912H    37  PUTCHAR. . . . . PROCEDURE STACK=0002H          1060 1105 1123 1176
  483  086DH   143  PUTDCHAR . . . . PROCEDURE STACK=0030H           514  516  525
  559  0CF2H   129  PUTDEST. . . . . PROCEDURE STACK=004CH           721  740  756  757  758  780  930 1305
                                    1308
  511  08FCH    89  PUTDESTC . . . . PROCEDURE STACK=0036H           534  535  554  555  557  577  587
   27  00C5H     1  PUTNUM . . . . . BYTE          566  582  589 1229
 1040  002CH     1  PWMODE . . . . . BYTE MEMBER(FCBS)          1108
   26  002CH     1  PWMODE . . . . . BYTE MEMBER(ODEST)
   26  002CH     1  PWMODE . . . . . BYTE MEMBER(DEST)
   26  002CH     1  PWMODE . . . . . BYTE MEMBER(SOURCE)
   74  0024H     8  PWNAM. . . . . . BYTE ARRAY(8) MEMBER(FCBS)           75
```

```
1040  0024H    8  PWNAM. . . . . .  BYTE ARRAY(8) MEMBER(FCBS)          1107 1183
  26  0024H    8  PWNAM. . . . . .  BYTE ARRAY(8) MEMBER(GDEST)
  26  0024H    8  PWNAM. . . . . .  BYTE ARRAY(8) MEMBER(DEST)
  26  0024H    8  PWNAM. . . . . .  BYTE ARRAY(8) MEMBER(SOURCE)
 115  0024H    8  PWNAM. . . . . .  BYTE ARRAY(8) MEMBER(FCBS)          116  120
  26  0024H    1  QUITLEN. . . . .  BYTE        666  668  697 1114
  31  0168H    1  QUITS. . . . . .  BYTE AT          692  694  696  775
  22             RB . . . . . .    LITERALLY       1044 1066
 727  10ECH   23  RDADDR . . . . .  PROCEDURE WORD STACK=005CH          760
  41  02ABH   15  RDCHAR . . . . .  PROCEDURE BYTE STACK=0008H          796 1008
  61  0300H   20  RDCOM. . . . . .  PROCEDURE STACK=0008H        1236
 724  10D2H   26  RDCS . . . . . .  PROCEDURE BYTE STACK=0056H          728  761  764  766
 705  0F77H   46  RDEOF. . . . . .  PROCEDURE BYTE STACK=0036H          929
 720  10C0H   18  RDHEX. . . . . .  PROCEDURE BYTE STACK=0050H          725
 153  04BCH   23  RDRANDOM . . . .  PROCEDURE BYTE STACK=000AH          419  870
 124  0430H   26  RENAME . . . . .  PROCEDURE STACK=0016H        814
   2  0000H       RESET. . . . . .  LABEL EXTERNAL(0)          261  303
  33  0288H   21  RETCODES . . . .  PROCEDURE STACK=0004H           81   91   95   98  103  107  111  122
                                     127  134  155  160
 163  04EAH   16  RETFSIZE . . . .  PROCEDURE BYTE STACK=000AH          369
   4  001EH    1  RETRY. . . . . .  BYTE INITIAL           253 1195
 712  0187H    1  RL . . . . . .    BYTE          744  747  748  759  762  763
  28  00C0H    1  RO . . . . . .    BYTE          367  820
                  ROL. . . . . .    BUILTIN        82  357  790  963  996
  31  0169H    1  RSYS . . . . . .  BYTE AT          357  996
 712  0189H    1  RT . . . . . .    BYTE          761
 182  0008H    2  S. . . . . . .    WORD PARAMETER AUTOMATIC          183  184  186  187
  26  0006H    2  SBASE. . . . . .  WORD           26  455  477  623  835  840  852
  26  0000H    2  SBLEN. . . . . .  WORD          449  450  452  612  836  839
  26  0000H 1024  SBUFF. . . . . .  BYTE BASED(SBASE) ARRAY(1024)          455  477  623  852
1039  16ABH  569  SCAN . . . . . .  PROCEDURE STACK=0028H        1245 1251 1312
1063  1950H  240  SCANPAR. . . . .  PROCEDURE STACK=0024H        1136 1155 1164 1187
 264  0178H    1  SDCNT. . . . . .  BYTE          266  269  273  276  299
  93  0396H   20  SEARCH . . . . .  PROCEDURE STACK=000AH         960  975
  97  03AAH   19  SEARCHN. . . . .  PROCEDURE STACK=0008H         965  979
  23             SEARFCB. . . . .  LITERALLY        975  984 1261
 144  0498H   12  SETCUSER . . . .  PROCEDURE STACK=000EH        1242
  68  0323H   16  SETDMA . . . . .  PROCEDURE STACK=000AH          75  120  386  406  417  455  868  973
 147  04A4H   12  SETDUSER . . . .  PROCEDURE STACK=000EH         248  329  382  784
 132  0459H   20  SETIND . . . . .  PROCEDURE STACK=000AH         810  823  829
  72  0333H   17  SETPW. . . . . .  PROCEDURE STACK=0010H          80  102  126
 167  04FAH   16  SETRANDOM. . . . .  PROCEDURE STACK=000AH       390  433  466  866
 150  04B0H   12  SETSUSER . . . .  PROCEDURE STACK=000EH         242  352  447  826  865  958  972
 328  0774H  160  SETUPDEST. . . .  PROCEDURE STACK=0026H         378
 846  1284H   70  SETUPEOB . . . .  PROCEDURE STACK=0002H         915
 350  0814H  176  SETUPSOURCE. . . .  PROCEDURE STACK=0026H       909
 139  047CH   28  SETUSER. . . . .  PROCEDURE STACK=000AH         145  148  151
 264  0179H    1  SEXTEN . . . . .  BYTE          267  270  271  277  279  292  301
  27  00B0H    1  SFILE. . . . . .  BYTE          890  902  908 1230 1286 1302 1304
                  SHL. . . . . .    BUILTIN        463  725  728  744  952  990
                  SHR. . . . . .    BUILTIN        404  453  539  839
 860  12FAH  248  SIMPLECOPY . . . .  PROCEDURE STACK=0064H       1016 1266 1275 1309
 832  125DH   87  SIZEMEMORY . . . .  PROCEDURE STACK=002EH        907
  26  0026H   47  SOURCE . . . . . .  STRUCTURE          26  151  243  314  316  318  320  322  353  354  357
                                      361  362  363  364  365  366  367  368  369  370  371  372  456  457
                                      461  464  465  466  470  473  606  717  738  767  827  828  829  866
                                      870  887  959  960  962  963  991  995 1022 1251 1254 1261 1265 1269
```

```
                                                1273 1274 1299 1301 1303 1307 1312
   26  0047H     2  SOURCER. . . . . .  WORD AT          872  879  884
   26  0049H     1  SOURCER2 . . . . .  BYTE AT          873  880  885
   27  00C3H     1  SPARFIL. . . . . .  BYTE        383  893
  225  002EH    20  SPECIALMSG . . . .  WORD ARRAY(10) DATA        276
   22             SPECL. . . . . . .  LITERALLY
  215  01EEH    19  SPER00 . . . . . .  BYTE ARRAY(19) DATA        225
  216  0201H    14  SPER01 . . . . . .  BYTE ARRAY(14) DATA        225
  217  020FH    27  SPER02 . . . . . .  BYTE ARRAY(27) DATA        225
  218  022AH    25  SPER03 . . . . . .  BYTE ARRAY(25) DATA        225
  219  0243H    27  SPER05 . . . . . .  BYTE ARRAY(27) DATA        225
  220  025EH    20  SPER06 . . . . . .  BYTE ARRAY(20) DATA        225
  221  0272H    14  SPER07 . . . . . .  BYTE ARRAY(14) DATA        225
  222  0280H    17  SPER08 . . . . . .  BYTE ARRAY(17) DATA        225  236
  223  0291H    13  SPER09 . . . . . .  BYTE ARRAY(13) DATA        225  236
   31  016AH     1  STARTS . . . . . .  BYTE AT          682  684  686  687  773
   28  00CEH     1  SYS. . . . . . . .  BYTE        368  821
   22             TAB. . . . . . . .  LITERALLY       513  557
   31  0168H     1  TABS . . . . . . .  BYTE AT          515  519  520  522 1294 1295
   26  0004H     2  TBLEN. . . . . . .  WORD        850  852  854
  376  0010H     2  TDEST. . . . . . .  WORD        381  386  400  401  403  406  410  415  418  424  427  434

  264  0176H     1  TEMP . . . . . . .  BYTE        290  294
   24             TRUE . . . . . . .  LITERALLY       253  326  348  356  476  589  605  659  672  732  749
                                        869  971 1047 1054 1116 1220 1229 1230 1231 1286
 1040  002EH     1  TYPE . . . . . . .  BYTE MEMBER(FCBS)          1097 1138 1153 1189
   26  002EH     1  TYPE . . . . . . .  BYTE MEMBER(DDEST)          496 1252 1258 1269 1277 1280 1282 1291
   26  002EH     1  TYPE . . . . . . .  BYTE MEMBER(DEST)
   26  002EH     1  TYPE . . . . . . .  BYTE MEMBER(SOURCE)         606 1254 1269 1273 1299 1301 1303 1307
   31  016CH     1  UPPER. . . . . . .  BYTE AT          646
 1040  002DH     1  USER . . . . . . .  BYTE MEMBER(FCBS)          1091 1110
   26  002DH     1  USER . . . . . . .  BYTE MEMBER(DDEST)          148 1022
   26  002DH     1  USER . . . . . . .  BYTE MEMBER(DEST)
   26  002DH     1  USER . . . . . . .  BYTE MEMBER(SOURCE)         151 1022
  139  0004H     1  USER . . . . . . .  BYTE PARAMETER AUTOMATIC          140  141  142
  591  0073H    26  UTRAN. . . . . . .  PROCEDURE BYTE STACK=0004H            647  796 1008 1028
   31  016DH     1  VERIF. . . . . . .  BYTE AT          412
   20             VERSION. . . . . .  LITERALLY      1199
   24             WHAT . . . . . . .  LITERALLY       959 1053 1058
  375  08C4H   408  WRITEDEST. . . . .  PROCEDURE STACK=002AH              503  618  783  842  921
  158  04D3H    23  WRITERANDOM. . . .  PROCEDURE BYTE STACK=000AH            387
   31  016EH     1  WRROF. . . . . . .  BYTE AT          792
  263  05D2H   312  XERROR . . . . . .  PROCEDURE STACK=0022H              338  341  345  346  361  362  388  409
                                        429  461  470  717  738  767  787  887  984 1247
   31  0171H     1  ZEROP. . . . . . .  BYTE AT          644
  712  018AH     1  ZEROREC. . . . . .  BYTE        730  736  749  750  751
   32  0172H     1  ZEROSUP. . . . . .  BYTE        533  545
```


MODULE INFORMATION:

```
  CODE AREA SIZE     = 1A40H   6720D
  CONSTANT AREA SIZE = 0402H   1026D
  VARIABLE AREA SIZE = 0197H    407D
  MAXIMUM STACK SIZE = 006CH    108D
  1964 LINES READ
```

0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION

```
ISIS-II MCS-86 MACRO ASSEMBLER V2.1 ASSEMBLY OF MODULE SCD1
OBJECT MODULE PLACED IN :F0:SCD1.OBJ
ASSEMBLER INVOKED BY:  :F0: SCD1.A86 XREF


LOC  OBJ                    LINE     SOURCE

                               1     name scd1
                               2     ;
                               3     ;       CP/M 3.0 MP/M-86 2.0 (BDOS version 3.0)
                               4     ;       Interface for PLM-86 with separate code and data
                               5     ;       Code org'd at 0
                               6     ;       December 18, 1981
                               7
                               8
                               9     dgroup  group   dats,stack
                              10     cgroup  group   code
                              11
                              12     assume  cs:cgroup, ds:dgroup, ss:dgroup
                              13
----                          14     stack   segment word stack 'STACK'
0000                          15     stack_base      label byte
----                          16     stack   ends
                              17
----                          18     dats     segment para public 'DATA'     ;CP/M page 0 - LOC86'd at 0H
                              19
0004                          20             org     4
0004 ??                       21     bdisk   db      ?
0006                          22             org     6
0006 ????                     23     maxb    dw      ?
0050                          24             org     50h
0050 ??                       25     cmdrv   db      ?
0051 ????                     26     pass0   dw      ?
0053 ??                       27     len0    db      ?
0054 ????                     28     pass1   dw      ?
0056 ??                       29     len1    db      ?
005C                          30             org     5ch
005C (16                      31     fcb     db      16 dup (?)
      ??
      )
006C (16                      32     fcb16   db      16 dup (?)
      ??
      )
007C ??                       33     cr      db      ?
007D ????                     34     rr      dw      ?
007F ??                       35     ro      db      ?
0080 (128                     36     buff    db      128 dup (?)
      ??
      )
  0080                        37     tbuff   equ     buff
                              38
                              39             public  bdisk,maxb,cmdrv,pass0,len0
                              40             public  pass1,len1,fcb,fcb16,cr,rr
                              41             public  ro,buff,tbuff
                              42
----                          43     dats ends
                              44
```

```
LOC  OBJ                  LINE     SOURCE

                            45
----                        46     code    segment public 'CODE'
                            47     public  reset,xdos,mon1,mon2,mon3,mon4
                            48     extrn   plm:near
                            49
0000                        50             org     0h              ; for separate code and data
0000 EB2C90                 51             jmp     reset
0003 434F5059524947         52             db 'COPYRIGHT (c) 1983 by DIGITAL RESEARCH INC.'
     485420286329
     313933833206279
     2044947495441
     4C205245534541
     524348204496E43
     2E
002E                        53     reset:
002E 9C                     54             pushf
002F 58                     55             pop     ax
0030 FA                     56             cli
0031 8CD9                   57             mov     cx,ds
0033 8ED1                   58             mov     ss,cx
0035 8D260000         R     59             lea     sp,stack_base
0039 50                     60             push    ax
003A 9D                     61             popf
003B E80000           E     62             call    plm
003E 33C9                   63             xor     cx,cx
0040 8BD1                   64             mov     dx,cx
0042 CDE0                   65             int     224
                            66
                            67
0044                        68     xdos    proc
0044 55                     69             push    bp
0045 8BEC                   70             mov     bp,sp
0047 8B5604                 71             mov     dx,[bp+4]
004A 8B4E06                 72             mov     cx,[bp+6]
004D CDE0                   73             int     224
004F 5D                     74             pop     bp
0050 C20400                 75             ret     4
                            76     xdos    endp
                            77
  0044                      78     mon1    equ     xdos            ; no returned value
  0044                      79     mon2    equ     xdos            ; returns byte in AL
  0044                      80     mon3    equ     xdos            ; returns address or word BX
  0044                      81     mon4    equ     xdos            ; returns pointer in BX and ES
                            82
007F                        83             org     07fh            ; reserve patch area
007F 00                     84             db      0
----                        85     code    ends
                            86     end
```

XREF SYMBOL TABLE LISTING
---- ------ ----- -------

```
NAME          TYPE      VALUE   ATTRIBUTES, XREFS

??SEG . . .   SEGMENT           SIZE=0000H PARA PUBLIC
BDISK . . .   V BYTE    0004H   DATA PUBLIC  21# 39
BUFF. . . .   V BYTE    0080H   DATA PUBLIC  35# 37 41
CGROUP. . .   GROUP             CODE   10# 12
CMDRV . . .   V BYTE    0050H   DATA PUBLIC  25# 39
CODE. . . .   SEGMENT           SIZE=0080H PARA PUBLIC 'CODE'  10# 46 85
CR. . . . .   V BYTE    007CH   DATA PUBLIC  33# 40
DATA. . . .   SEGMENT           SIZE=0100H PARA PUBLIC 'DATA'  9# 18 43
DGROUP. . .   GROUP             DATA STACK   9# 12 12
FCB . . . .   V BYTE    005CH   DATA PUBLIC  31# 40
FCB16 . . .   V BYTE    006CH   DATA PUBLIC  32# 40
LEN0. . . .   V BYTE    0053H   DATA PUBLIC  27# 39
LEN1. . . .   V BYTE    0056H   DATA PUBLIC  29# 40
MAXB. . . .   V WORD    0006H   DATA PUBLIC  23# 39
MON1. . . .   L NEAR    0044H   CODE PUBLIC  47 78#
MON2. . . .   L NEAR    0044H   CODE PUBLIC  47 79#
MON3. . . .   L NEAR    0044H   CODE PUBLIC  47 80#
MON4. . . .   L NEAR    0044H   CODE PUBLIC  47 81#
PASS0 . . .   V WORD    0051H   DATA PUBLIC  26# 39
PASS1 . . .   V WORD    0054H   DATA PUBLIC  28# 40
PLM . . . .   L NEAR    0000H   EXTRN  48# 62
RESET . . .   L NEAR    002EH   CODE PUBLIC  47 51 53#
RO. . . . .   V BYTE    007FH   DATA PUBLIC  35# 41
RR. . . . .   V WORD    007DH   DATA PUBLIC  34# 40
STACK . . .   SEGMENT           SIZE=0000H WORD  STACK 'STACK'
STACK_BASE.   V BYTE    0000H   STACK  15# 59
TBUFF . . .   V BYTE    0080H   DATA PUBLIC  37# 41
XDOS. . . .   L NEAR    0044H   CODE PUBLIC  47 68# 76 78 79 80 81

ASSEMBLY COMPLETE, NO ERRORS FOUND
```

```
ISIS-II MCS-86 MACRO ASSEMBLER V2.1 ASSEMBLY OF MODULE INPOUT
OBJECT MODULE PLACED IN :F0:INPOUT.OBJ
ASSEMBLER INVOKED BY:  :F0: INPOUT.A86 XREF
```



```
LOC  OBJ                  LINE     SOURCE

                             1     name inpout
                             2     ;
                             3     ;       CP/M-86 1.1 PIP Utility INP: / OUT:
                             4     ;       Interface module with separate code and data
                             5     ;       Code org'd at 080h
                             6     ;       December 18, 1981
                             7
                             8     cgroup  group   code
                             9
                            10     assume  cs:cgroup
                            11
----                        12     code    segment public  'CODE'
                            13     public  inploc,outloc,inpd,outd
                            14
0000                        15             org     00h             ; for separate code and data
0000                        16     inpd    proc
0000 55                     17             push bp
0001 E80F00                 18             call inploc
0004 5D                     19             pop bp
0005 C3                     20             ret
                            21     inpd    endp
                            22
0006                        23     outd    proc
0006 55                     24             push bp
0007 8BEC                   25             mov bp,sp
0009 8A4604                 26             mov al,[bp]+4
000C E80700                 27             call outloc
000F 5D                     28             pop bp
0010 C20200                 29             ret 2
                            30     outd    endp
                            31
0013                        32     inploc  proc
0013 B01A                   33             mov al,01Ah
0015 C3                     34             ret
                            35     inploc  endp
                            36
0016                        37     outloc  proc
0016 C3                     38             ret
0017 90                     39             nop
0018 90                     40             nop
                            41     outloc  endp
                            42
007F                        43             org     07fh
007F 00                     44             db      0
----                        45     code    ends
                            46     end
```

XREF SYMBOL TABLE LISTING
---- ------ ----- -------

```
NAME     TYPE      VALUE   ATTRIBUTES, XREFS

??SEG  . SEGMENT           SIZE=0000H PARA PUBLIC
CGROUP . GROUP             CODE    8# 10
CODE   . SEGMENT           SIZE=0080H PARA PUBLIC 'CODE'  8# 12 45
INPD   . L NEAR    0000H   CODE PUBLIC  13 16# 21
INPLOC . L NEAR    0013H   CODE PUBLIC  13 18 32# 35
OUTD   . L NEAR    0006H   CODE PUBLIC  13 23# 30
OUTLOC . L NEAR    0016H   CODE PUBLIC  13 27 37# 41
```

ASSEMBLY COMPLETE, NO ERRORS FOUND

```
ISIS-II MCS-86 LOCATER, V1.2 INVOKED BY:
:F0: PIP.LNK OD(SM(CODE,DATA,DATA,STACK,CONST)) AD(SM(CODE(0),DATA(10000H))) SS(STACK(+32)) TO PIP.

SYMBOL TABLE OF MODULE SCD1
READ FROM FILE PIP.LNK
WRITTEN TO FILE PIP.

BASE   OFFSET TYPE SYMBOL                BASE   OFFSET TYPE SYMBOL

0000H  0100H  PUB  PLM                   0000H  0093H  PUB  INPLOC
0000H  0096H  PUB  OUTLOC                0000H  0080H  PUB  INPO
0000H  0086H  PUB  OUTO                  0000H  002EH  PUB  RESET
0000H  0044H  PUB  XDOS                  0000H  0044H  PUB  MON1
0000H  0044H  PUB  MON2                  0000H  0044H  PUB  MON3
0000H  0044H  PUB  MON4                  1000H  0004H  PUB  BDTSK
1000H  0006H  PUB  MAXB                  1000H  0050H  PUB  CHDRV
1000H  0051H  PUB  PASS0                 1000H  0053H  PUB  LEN0
1000H  0054H  PUB  PASS1                 1000H  0056H  PUB  LEN1
1000H  005CH  PUB  FCB                   1000H  006CH  PUB  FCB16
1000H  007CH  PUB  CR                    1000H  007DH  PUB  RR
1000H  007FH  PUB  R0                    1000H  0080H  PUB  BUFF
1000H  0080H  PUB  TBUFF

PIPMOD:  SYMBOLS AND LINES
1000H  0730H  SYM  MEMORY                1000H  011EH  SYM  RETRY
0000H  0100H  SYM  PLM                   1000H  037CH  SYM  COPYRIGHT
1000H  011FH  SYM  COLUMN                1000H  0120H  SYM  LINENO
1000H  0121H  SYM  FEEDBASE              1000H  0122H  SYM  FEEDLEN
1000H  0123H  SYM  MATCHLEN              1000H  0124H  SYM  QUITLEN
1000H  0125H  SYM  CDTSK                 1000H  0100H  SYM  SBLEN
1000H  0102H  SYM  DBLEN                 1000H  0104H  SYM  TBLEN
1000H  0106H  SYM  SBASE                 1000H  0730H  SYM  DBUFF
1000H  0106H  BAS  SBUFF                 1000H  0126H  SYM  SOURCE
1000H  0155H  SYM  DEST                  1000H  0184H  SYM  ODEST
1000H  0183H  SYM  FILSIZE               1000H  0176H  SYM  DESTR
1000H  0147H  SYM  SOURCER               1000H  0178H  SYM  DESTR2
1000H  0149H  SYM  SOURCER2              1000H  01B6H  SYM  EXTSAVE
1000H  0108H  SYM  NSBUF                 1000H  010AH  SYM  BUFSIZE
1000H  01B7H  SYM  NSECCNT               1000H  010CH  SYM  NSOURCE
1000H  010EH  SYM  NDEST                 1000H  01B8H  SYM  FASTCOPY
1000H  01B9H  SYM  DBLBUF                1000H  018AH  SYM  CONCAT
1000H  01BBH  SYM  AMBIG                 1000H  018CH  SYM  OFILE
1000H  018DH  SYM  SFILE                 1000H  018EH  SYM  MADE
1000H  018FH  SYM  OPENED                1000H  01C0H  SYM  ENDOFSRC
1000H  01C1H  SYM  NENDCMD               1000H  01C2H  SYM  INSPARC
1000H  01C3H  SYM  SPARFIL               1000H  01C4H  SYM  MULTCOM
1000H  01C5H  SYM  PUTNUM                1000H  01C6H  SYM  CONCNT
1000H  01C7H  SYM  CHAR                  1000H  01C8H  SYM  FLEN
1000H  01C9H  SYM  F1                    1000H  01CAH  SYM  F2
1000H  01CBH  SYM  F3                    1000H  01CCH  SYM  F4
1000H  01CDH  SYM  R0                    1000H  01CEH  SYM  SYS
1000H  01CFH  SYM  EXTEN                 1000H  01D0H  SYM  ODCNT
1000H  01D1H  SYM  ERETRY                1000H  01D2H  SYM  OCNT
1000H  01D3H  SYM  CBUFF                 1000H  01D3H  SYM  MAXLEN
1000H  01D4H  SYM  COMLEN                1000H  01D5H  SYM  COMBUFF
1000H  0255H  SYM  CBP                   1000H  0256H  SYM  CUSER
1000H  0257H  SYM  LASTUSER              1000H  0258H  SYM  CONT
1000H  0258H  SYM  ARCHIV                1000H  025AH  SYM  CONFRM
1000H  025BH  SYM  DELET                 1000H  025CH  SYM  ECHO
1000H  025DH  SYM  FORMF                 1000H  025EH  SYM  GETU
```

```
1000H  025FH  SYM  NEXT
1000H  0262H  SYM  KILOS
1000H  0265H  SYM  NUMB
1000H  0267H  SYM  PAGCNT
1000H  0269H  SYM  RSYS
1000H  026BH  SYM  TABS
1000H  026DH  SYM  VERIF
1000H  0271H  SYM  ZEROP
1000H  0273H  SYM  C3
1000H  0275H  SYM  C1
STACK  0004H  SYM  A
0000H  03ABH  SYM  RDCHAR
STACK  0004H  SYM  CHAR
0000H  03E0H  SYM  PRINTX
0000H  03F0H  SYM  PRINT
0000H  0400H  SYM  RDCOM
0000H  0423H  SYM  SETDMA
0000H  0433H  SYM  SETPW
STACK  0004H  BAS  FCBS
STACK  0004H  SYM  FCBA
0000H  0482H  SYM  CLOSE
0000H  0496H  SYM  SEARCH
0000H  04AAH  SYM  SEARCHN
STACK  0004H  SYM  FCB
STACK  0004H  SYM  FCB
STACK  0004H  SYM  FCB
STACK  0004H  SYM  FCBA
0000H  0530H  SYM  RENAME
0000H  054AH  SYM  GETDISK
STACK  0004H  SYM  FCB
0000H  057CH  SYM  SETUSER
0000H  0598H  SYM  SETCUSER
0000H  05B0H  SYM  SETSUSER
STACK  0004H  SYM  FCB
STACK  0004H  SYM  FCB
STACK  0004H  SYM  FCB
STACK  0004H  SYM  FCB
STACK  0004H  SYM  CNT
0000H  0635H  SYM  CONATLST
STACK  0008H  SYM  S
STACK  0004H  SYM  N
STACK  0006H  BAS  B
1000H  03ABH  SYM  ER01
1000H  03BAH  SYM  ER03
1000H  03DDH  SYM  ER05
1000H  03F8H  SYM  ER07
1000H  041BH  SYM  ER09
1000H  043EH  SYM  ER11
1000H  045DH  SYM  ER13
1000H  047AH  SYM  ER15
1000H  04A7H  SYM  ER17
1000H  04DAH  SYM  ER19
1000H  04EEH  SYM  ER21
1000H  0324H  SYM  ERRMSG
1000H  0525H  SYM  SPER01
1000H  054EH  SYM  SPER03
1000H  0582H  SYM  SPER06
1000H  05A4H  SYM  SPER08
1000H  0352H  SYM  SPECIALMSG
1000H  05C3H  SYM  EX01
1000H  0250H  SYM  TOWER
1000H  0253H  SYM  LOWER
1000H  0266H  SYM  DSJ
1000H  0268H  SYM  QUITS
1000H  026AH  SYM  STARTS
1000H  026CH  SYM  UPPER
1000H  026EH  SYM  WRROF
1000H  0272H  SYM  ZEROSUP
1000H  0274H  SYM  C2
0000H  0388H  SYM  RETCODES
0000H  039DH  SYM  BDOS
0000H  03BAH  SYM  PRINTCHAR
0000H  03CFH  SYM  CRLF
STACK  0004H  SYM  A
STACK  0004H  SYM  A
0000H  0414H  SYM  GVERSION
STACK  0004H  SYM  A
STACK  0004H  SYM  FCBA
0000H  0444H  SYM  OPEN
STACK  0004H  BAS  FCB
STACK  0004H  SYM  FCB
STACK  0004H  SYM  FCB
0000H  04BDH  SYM  DELETE
0000H  04D7H  SYM  DISKRD
0000H  04E8H  SYM  DISKWRITE
0000H  04FFH  SYM  MAKE
STACK  0004H  BAS  FCBS
STACK  0004H  SYM  FCB
0000H  0559H  SYM  SETIND
0000H  056DH  SYM  GETUSER
STACK  0004H  SYM  USER
0000H  05A4H  SYM  SETDUSER
0000H  05BCH  SYM  RDRANDOM
0000H  05D3H  SYM  WRTTERANDOM
0000H  05EAH  SYM  RETFSIZE
0000H  05FAH  SYM  SETRANDOM
0000H  060AH  SYM  MULTSECT
0000H  0626H  SYM  FLUSHBUF
0000H  0644H  SYM  MOVE
STACK  0006H  SYM  D
STACK  0008H  BAS  A
1000H  039EH  SYM  ER00
1000H  03B3H  SYM  ER02
1000H  03CEH  SYM  ER04
1000H  03EAH  SYM  ER06
1000H  040CH  SYM  ER08
1000H  042FH  SYM  ER10
1000H  044EH  SYM  ER12
1000H  046FH  SYM  ER14
1000H  0495H  SYM  ER16
1000H  04BAH  SYM  ER18
1000H  04E4H  SYM  ER20
1000H  04FBH  SYM  ER22
1000H  0512H  SYM  SPER00
1000H  0533H  SYM  SPER02
1000H  0567H  SYM  SPER05
1000H  0596H  SYM  SPER07
1000H  05B5H  SYM  SPER09
1000H  05C2H  SYM  EX00
1000H  05D2H  SYM  EX02
```

1000H	05D8H	SYM	EX03	1000H	05E4H	SYM	EX04
1000H	05F8H	SYM	EX05	1000H	060AH	SYM	EX07
1000H	0618H	SYM	EX08	1000H	062AH	SYM	EX10
1000H	0366H	SYM	EXTHSG	0000H	0669H	SYM	ERRJRELEAMOP
0000H	0688H	SYM	ERROR	STACK	0004H	SYM	ERRTYPE
0000H	06D2H	SYM	XERROR	STACK	0006H	SYM	FUNCNO
STACK	0004H	SYM	FILEADR	1000H	0276H	SYM	TEMP
1000H	0277H	SYM	I	1000H	0278H	SYM	SDCNT
1000H	0279H	SYM	SEXTEN	STACK	0004H	BAS	FCB
1000H	0639H	SYM	MESSAGEINDEXTBL	0000H	080AH	SYM	FORMERR
0000H	0815H	SYM	CONBRK	0000H	083CH	SYM	MAXSIZE
0000H	0874H	SYM	SETUPDEST	0000H	0914H	SYM	SETUPSOURCE
1000H	027AH	SYM	I	1000H	027BH	SYM	J
0000H	09C4H	SYM	WRITEDEST	1000H	027CH	SYM	J
1000H	027DH	SYM	DATAOK	1000H	0110H	SYM	TDEST
1000H	0112H	SYM	N	0000H	085CH	SYM	FILLSOURCE
0000H	0C6DH	SYM	PUTDCHAR	STACK	0004H	SYM	B
0000H	0CC1H	SYM	AXOCASE	0000H	0CFCH	SYM	PUTDESTC
STACK	0004H	SYM	B	1000H	027EH	SYM	I
0000H	0D55H	SYM	PRINT1	STACK	0004H	SYM	B
0000H	0D7DH	SYM	PRINTDIG	STACK	0004H	SYM	D
0000H	0D98H	SYM	NEWLINE	1000H	027FH	SYM	ONE
0000H	0DF2H	SYM	PUTDEST	STACK	0004H	SYM	B
1000H	0280H	SYM	I	0000H	0E73H	SYM	UTRAN
STACK	0004H	SYM	B	0000H	0E8DH	SYM	LTRAN
STACK	0004H	SYM	B	0000H	0EA7H	SYM	GETSOURCEC
1000H	0281H	SYM	B	1000H	0282H	SYM	CONCHK
0000H	0EBCH	SYM	NOTSOURCE	0000H	0F1CH	SYM	AXICASE
0000H	0F95H	SYM	GETSOURCE	1000H	0283H	SYM	CHAR
0000H	1035H	SYM	MATCH	STACK	0004H	SYM	B
1000H	0284H	SYM	C	0000H	1077H	SYM	RDEOF
0000H	10A5H	SYM	HEXRECORD	1000H	0285H	SYM	H
1000H	0286H	SYM	HBUF	1000H	0287H	SYM	RL
1000H	0288H	SYM	CS	1000H	0289H	SYM	RT
1000H	028AH	SYM	ZEROREC	1000H	0114H	SYM	LDA
0000H	1196H	SYM	CKHEX	0000H	11C0H	SYM	RDHEX
0000H	11D2H	SYM	RDCS	0000H	11ECH	SYM	RDADDR
0000H	1203H	SYM	CKSTRINGS	0000H	1222H	SYM	CLOSEDEST
0000H	135DH	SYM	SIZEMEMORY	0000H	1384H	SYM	SETUPEOB
1000H	028BH	SYM	I	0000H	13FAH	SYM	SIMPLECOPY
1000H	028CH	SYM	I	1000H	064AH	SYM	OPTYPE
0000H	14F2H	SYM	CHKRANDOM	0000H	1571H	SYM	MULTCOPY
1000H	0116H	SYM	NEXTDIR	1000H	0118H	SYM	NDCNT
1000H	011AH	SYM	NCOPIED	0000H	16BBH	SYM	PRNAME
1000H	028DH	SYM	I	1000H	028EH	SYM	C
0000H	16F7H	SYM	ARCHCK	0000H	174BH	SYM	CKDISK
0000H	1765H	SYM	GNC	0000H	178AH	SYM	DEBLANK
0000H	1799H	SYM	CKEOL	0000H	17ABH	SYM	SCAN
1000H	011CH	SYM	FCBA	1000H	011CH	BAS	FCBS
1000H	028FH	SYM	I	1000H	0290H	SYM	K
0000H	19E4H	SYM	DELIMITER	STACK	0004H	SYM	C
1000H	0291H	SYM	I	1000H	0664H	SYM	DEL
0000H	1A12H	SYM	PUTCHAR	0000H	1A37H	SYM	FILLQ
STACK	0004H	SYM	LEN	0000H	1A50H	SYM	SCANPAR
1000H	0292H	SYM	I	1000H	0293H	SYM	J
1000H	0294H	SYM	I	1000H	0295H	SYM	J
1000H	0296H	SYM	K	1000H	0670H	SYM	IO
0000H	0100H	LIN	19	0000H	0388H	LIN	33
0000H	038BH	LIN	35	0000H	0391H	LIN	36
0000H	0399H	LIN	37	0000H	0390H	LIN	38

```
0000H  03A0H  LIN    39
0000H  03ABH  LIN    41
0000H  03BAH  LIN    43
0000H  03BDH  LIN    46
0000H  03CFH  LIN    48
0000H  03D8H  LIN    50
0000H  03E0H  LIN    52
0000H  03ECH  LIN    55
0000H  03F3H  LIN    58
0000H  03FCH  LIN    60
0000H  0403H  LIN    62
0000H  0412H  LIN    64
0000H  0417H  LIN    66
0000H  0423H  LIN    68
0000H  042FH  LIN    71
0000H  0436H  LIN    75
0000H  0444H  LIN    77
0000H  044DH  LIN    81
0000H  046DH  LIN    84
0000H  0479H  LIN    86
0000H  0482H  LIN    89
0000H  0492H  LIN    92
0000H  0499H  LIN    95
0000H  04AAH  LIN    97
0000H  04BBH  LIN    99
0000H  04C0H  LIN   102
0000H  04D3H  LIN   104
0000H  04DAH  LIN   107
0000H  04EBH  LIN   109
0000H  04FBH  LIN   112
0000H  0502H  LIN   116
0000H  0511H  LIN   119
0000H  051FH  LIN   122
0000H  0530H  LIN   124
0000H  0539H  LIN   127
0000H  054AH  LIN   129
0000H  0559H  LIN   131
0000H  055CH  LIN   134
0000H  056DH  LIN   136
0000H  057CH  LIN   138
0000H  057FH  LIN   141
0000H  0594H  LIN   143
0000H  059BH  LIN   145
0000H  05A4H  LIN   147
0000H  05AEH  LIN   149
0000H  05B3H  LIN   151
0000H  05BCH  LIN   153
0000H  05CCH  LIN   156
0000H  05D3H  LIN   158
0000H  05E3H  LIN   161
0000H  05EAH  LIN   163
0000H  05FAH  LIN   166
0000H  05FDH  LIN   169
0000H  060AH  LIN   171
0000H  0616H  LIN   174
0000H  0626H  LIN   176
0000H  0633H  LIN   178
0000H  0638H  LIN   180
0000H  0644H  LIN   182
0000H  0653H  LIN   186
0000H  0659H  LIN    40
0000H  05AEH  LIN    42
0000H  03BAH  LIN    44
0000H  03C8H  LIN    47
0000H  03D2H  LIN    49
0000H  03DEH  LIN    51
0000H  03E3H  LIN    54
0000H  03F0H  LIN    56
0000H  03F6H  LIN    59
0000H  0400H  LIN    61
0000H  0408H  LIN    63
0000H  0414H  LIN    65
0000H  0423H  LIN    67
0000H  0426H  LIN    70
0000H  0433H  LIN    72
0000H  0440H  LIN    76
0000H  0447H  LIN    80
0000H  045AH  LIN    82
0000H  0474H  LIN    85
0000H  047EH  LIN    88
0000H  0485H  LIN    91
0000H  0496H  LIN    93
0000H  04A6H  LIN    96
0000H  04ADH  LIN    98
0000H  04BDH  LIN   100
0000H  04C6H  LIN   103
0000H  04D7H  LIN   105
0000H  04E7H  LIN   108
0000H  04EEH  LIN   111
0000H  04FFH  LIN   113
0000H  050BH  LIN   117
0000H  0518H  LIN   120
0000H  052CH  LIN   123
0000H  0533H  LIN   126
0000H  0546H  LIN   128
0000H  054DH  LIN   130
0000H  0559H  LIN   132
0000H  0569H  LIN   135
0000H  0570H  LIN   137
0000H  057CH  LIN   139
0000H  0588H  LIN   142
0000H  0598H  LIN   144
0000H  05A2H  LIN   146
0000H  05A7H  LIN   148
0000H  05B0H  LIN   150
0000H  05BAH  LIN   152
0000H  05BFH  LIN   155
0000H  05D3H  LIN   157
0000H  05D6H  LIN   160
0000H  05EAH  LIN   162
0000H  05EDH  LIN   165
0000H  05FAH  LIN   167
0000H  0606H  LIN   170
0000H  060DH  LIN   173
0000H  0622H  LIN   175
0000H  0629H  LIN   177
0000H  0635H  LIN   179
0000H  0644H  LIN   181
0000H  0647H  LIN   185
0000H  065DH  LIN   187
```

```
0000H  0660H  LIN   188
0000H  0665H  LIN   190
0000H  066CH  LIN   238
0000H  0677H  LIN   240
0000H  0681H  LIN   243
0000H  068DH  LIN   246
0000H  0697H  LIN   249
0000H  06A5H  LIN   252
0000H  06AFH  LIN   254
0000H  06B8H  LIN   256
0000H  06BEH  LIN   259
0000H  06CFH  LIN   261
0000H  06D5H  LIN   266
0000H  06E1H  LIN   268
0000H  06F1H  LIN   270
0000H  06FDH  LIN   272
0000H  0728H  LIN   275
0000H  0742H  LIN   277
0000H  074EH  LIN   281
0000H  0763H  LIN   284
0000H  0770H  LIN   287
0000H  0783H  LIN   289
0000H  07A4H  LIN   292
0000H  07AEH  LIN   294
0000H  07B8H  LIN   298
0000H  07D1H  LIN   300
0000H  0801H  LIN   302
0000H  080AH  LIN   305
0000H  0813H  LIN   307
0000H  0818H  LIN   309
0000H  0834H  LIN   311
0000H  083CH  LIN   313
0000H  0848H  LIN   316
0000H  085AH  LIN   320
0000H  086CH  LIN   323
0000H  0874H  LIN   327
0000H  0877H  LIN   329
0000H  0888H  LIN   331
0000H  0898H  LIN   333
0000H  08A8H  LIN   335
0000H  08B5H  LIN   337
0000H  08C6H  LIN   339
0000H  08DBH  LIN   341
0000H  08ECH  LIN   343
0000H  08FAH  LIN   345
0000H  0908H  LIN   347
0000H  0912H  LIN   349
0000H  0917H  LIN   352
0000H  091FH  LIN   354
0000H  092DH  LIN   356
0000H  0943H  LIN   358
0000H  094FH  LIN   360
0000H  095AH  LIN   362
0000H  096EH  LIN   364
0000H  097EH  LIN   366
0000H  098EH  LIN   368
0000H  09A0H  LIN   370
0000H  09B3H  LIN   372
0000H  09C2H  LIN   374
0000H  09C7H  LIN   377

0000H  0663H  LIN   187
0000H  0669H  LIN   237
0000H  0672H  LIN   239
0000H  067EH  LIN   242
0000H  0688H  LIN   244
0000H  0694H  LIN   248
0000H  069EH  LIN   250
0000H  06AAH  LIN   253
0000H  06B6H  LIN   255
0000H  06BBH  LIN   258
0000H  06CCH  LIN   260
0000H  06D2H  LIN   263
0000H  06DBH  LIN   267
0000H  06E4H  LIN   269
0000H  06F8H  LIN   271
0000H  0711H  LIN   273
0000H  0731H  LIN   276
0000H  0747H  LIN   279
0000H  0754H  LIN   282
0000H  076AH  LIN   285
0000H  077DH  LIN   288
0000H  078FH  LIN   290
0000H  07A8H  LIN   293
0000H  07B5H  LIN   296
0000H  07BEH  LIN   299
0000H  07DAH  LIN   301
0000H  0807H  LIN   303
0000H  080DH  LIN   306
0000H  0815H  LIN   308
0000H  0826H  LIN   310
0000H  083AH  LIN   312
0000H  083FH  LIN   314
0000H  0851H  LIN   318
0000H  0863H  LIN   322
0000H  0870H  LIN   326
0000H  0874H  LIN   328
0000H  087AH  LIN   330
0000H  0896H  LIN   332
0000H  08A2H  LIN   334
0000H  08ACH  LIN   336
0000H  08BCH  LIN   338
0000H  08CDH  LIN   340
0000H  08E5H  LIN   342
0000H  08F3H  LIN   344
0000H  08FEH  LIN   346
0000H  090DH  LIN   348
0000H  0914H  LIN   350
0000H  091AH  LIN   353
0000H  0926H  LIN   355
0000H  0932H  LIN   357
0000H  0948H  LIN   359
0000H  0956H  LIN   361
0000H  0964H  LIN   363
0000H  0976H  LIN   365
0000H  0986H  LIN   367
0000H  0996H  LIN   369
0000H  09AEH  LIN   371
0000H  09BCH  LIN   373
0000H  09C4H  LIN   375
0000H  09D0H  LIN   378
```

```
0000H 09D3H LIN  379
0000H 09E0H LIN  381
0000H 09E9H LIN  383
0000H 09FDH LIN  386
0000H 0A14H LIN  388
0000H 0A20H LIN  390
0000H 0A2EH LIN  393
0000H 0A38H LIN  395
0000H 0A3EH LIN  398
0000H 0A52H LIN  401
0000H 0A63H LIN  404
0000H 0A79H LIN  407
0000H 0A87H LIN  409
0000H 0A98H LIN  411
0000H 0AA4H LIN  414
0000H 0AADH LIN  416
0000H 0AB8H LIN  418
0000H 0AD4H LIN  420
0000H 0ADEH LIN  422
0000H 0AF5H LIN  424
0000H 0B16H LIN  426
0000H 0B1EH LIN  428
0000H 0B31H LIN  430
0000H 0B3AH LIN  433
0000H 0B5AH LIN  435
0000H 0B5FH LIN  437
0000H 0B69H LIN  440
0000H 0B73H LIN  442
0000H 0B79H LIN  445
0000H 0B82H LIN  448
0000H 0B99H LIN  450
0000H 0BAEH LIN  453
0000H 0BC6H LIN  456
0000H 0BD3H LIN  458
0000H 0BE4H LIN  461
0000H 0BF5H LIN  463
0000H 0C12H LIN  465
0000H 0C1EH LIN  467
0000H 0C37H LIN  470
0000H 0C43H LIN  473
0000H 0C4FH LIN  476
0000H 0C5FH LIN  478
0000H 0C61H LIN  480
0000H 0C68H LIN  482
0000H 0C70H LIN  485
0000H 0C7EH LIN  488
0000H 0C88H LIN  491
0000H 0C92H LIN  495
0000H 0CA8H LIN  497
0000H 0CC1H LIN  500
0000H 0CCAH LIN  503
0000H 0CD8H LIN  505
0000H 0CDEH LIN  507
0000H 0CEAH LIN  509
0000H 0CFCH LIN  511
0000H 0D05H LIN  515
0000H 0D14H LIN  518
0000H 0D23H LIN  520
0000H 0D29H LIN  522
0000H 0D3AH LIN  524
0000H 09DEH LIN  380
0000H 09E6H LIN  382
0000H 09F7H LIN  385
0000H 0A09H LIN  387
0000H 0A1EH LIN  389
0000H 0A27H LIN  391
0000H 0A34H LIN  394
0000H 0A38H LIN  397
0000H 0A44H LIN  400
0000H 0A5FH LIN  403
0000H 0A6DH LIN  406
0000H 0A80H LIN  408
0000H 0A91H LIN  410
0000H 0A9AH LIN  412
0000H 0AA7H LIN  415
0000H 0AB3H LIN  417
0000H 0AC3H LIN  419
0000H 0ADAH LIN  421
0000H 0AE3H LIN  423
0000H 0B12H LIN  425
0000H 0B18H LIN  427
0000H 0B27H LIN  429
0000H 0B33H LIN  431
0000H 0B41H LIN  434
0000H 0B5CH LIN  436
0000H 0B62H LIN  438
0000H 0B6FH LIN  441
0000H 0B73H LIN  444
0000H 0B7FH LIN  447
0000H 0B88H LIN  449
0000H 0BA6H LIN  452
0000H 0BB8H LIN  455
0000H 0BCCH LIN  457
0000H 0BDDH LIN  460
0000H 0BEEH LIN  462
0000H 0C02H LIN  464
0000H 0C17H LIN  466
0000H 0C2AH LIN  469
0000H 0C41H LIN  471
0000H 0C4AH LIN  474
0000H 0C54H LIN  477
0000H 0C61H LIN  479
0000H 0C68H LIN  481
0000H 0C6DH LIN  483
0000H 0C76H LIN  487
0000H 0C85H LIN  490
0000H 0C88H LIN  494
0000H 0C9EH LIN  496
0000H 0CB3H LIN  499
0000H 0CC1H LIN  502
0000H 0CCDH LIN  504
0000H 0CDCH LIN  506
0000H 0CE6H LIN  508
0000H 0CF8H LIN  510
0000H 0CFFH LIN  513
0000H 0D0CH LIN  516
0000H 0D1AH LIN  519
0000H 0D27H LIN  521
0000H 0D33H LIN  523
0000H 0D3EH LIN  525
```

```
0000H  0D44H  LIN    526
0000H  0D4CH  LIN    529
0000H  0D55H  LIN    531
0000H  0D6CH  LIN    534
0000H  0D79H  LIN    536
0000H  0D80H  LIN    539
0000H  0D94H  LIN    541
0000H  0D98H  LIN    544
0000H  0DACH  LIN    546
0000H  0DBEH  LIN    548
0000H  0DCBH  LIN    550
0000H  0DD9H  LIN    552
0000H  0DE6H  LIN    555
0000H  0DEAH  LIN    557
0000H  0DF2H  LIN    559
0000H  0DFCH  LIN    563
0000H  0E02H  LIN    566
0000H  0E15H  LIN    570
0000H  0E26H  LIN    573
0000H  0E39H  LIN    576
0000H  0E44H  LIN    580
0000H  0E4EH  LIN    582
0000H  0E59H  LIN    586
0000H  0E64H  LIN    588
0000H  0E6FH  LIN    590
0000H  0E76H  LIN    593
0000H  0E86H  LIN    595
0000H  0E8DH  LIN    597
0000H  0E9CH  LIN    600
0000H  0EA7H  LIN    602
0000H  0EAAH  LIN    605
0000H  0EBCH  LIN    609
0000H  0EC4H  LIN    612
0000H  0EDAH  LIN    615
0000H  0EE9H  LIN    618
0000H  0EF2H  LIN    621
0000H  0F02H  LIN    624
0000H  0F08H  LIN    626
0000H  0F0DH  LIN    629
0000H  0F1CH  LIN    631
0000H  0F21H  LIN    633
0000H  0F3AH  LIN    636
0000H  0F4DH  LIN    638
0000H  0F61H  LIN    641
0000H  0F6BH  LIN    645
0000H  0F77H  LIN    647
0000H  0F87H  LIN    649
0000H  0F95H  LIN    651
0000H  1035H  LIN    654
0000H  1052H  LIN    658
0000H  105DH  LIN    661
0000H  106CH  LIN    663
0000H  1077H  LIN    665
0000H  0F9FH  LIN    668
0000H  0FAFH  LIN    670
0000H  0FB1H  LIN    673
0000H  0FBCH  LIN    676
0000H  0FCEH  LIN    678
0000H  0FDAH  LIN    681
0000H  0FE5H  LIN    684

0000H  0D46H  LIN    527
0000H  0D51H  LIN    530
0000H  0D58H  LIN    533
0000H  0D70H  LIN    535
0000H  0D7DH  LIN    537
0000H  0D8BH  LIN    540
0000H  0D98H  LIN    542
0000H  0DA0H  LIN    545
0000H  0DB5H  LIN    547
0000H  0DC7H  LIN    549
0000H  0DD2H  LIN    551
0000H  0DE0H  LIN    554
0000H  0DEAH  LIN    556
0000H  0DF0H  LIN    559
0000H  0DF5H  LIN    561
0000H  0E02H  LIN    564
0000H  0E09H  LIN    568
0000H  0E22H  LIN    572
0000H  0E28H  LIN    574
0000H  0E3EH  LIN    577
0000H  0E4BH  LIN    581
0000H  0E53H  LIN    585
0000H  0E5EH  LIN    587
0000H  0E6AH  LIN    589
0000H  0E73H  LIN    591
0000H  0E82H  LIN    594
0000H  0E8DH  LIN    596
0000H  0E90H  LIN    599
0000H  0EA0H  LIN    601
0000H  0EA7H  LIN    603
0000H  0EAFH  LIN    606
0000H  0EC4H  LIN    610
0000H  0ECDH  LIN    614
0000H  0EE2H  LIN    616
0000H  0EECH  LIN    619
0000H  0EF5H  LIN    623
0000H  0F06H  LIN    625
0000H  0F08H  LIN    628
0000H  0F1AH  LIN    630
0000H  0F1CH  LIN    632
0000H  0F33H  LIN    634
0000H  0F41H  LIN    637
0000H  0F5AH  LIN    639
0000H  0F64H  LIN    644
0000H  0F70H  LIN    646
0000H  0F80H  LIN    648
0000H  0F90H  LIN    650
0000H  0F95H  LIN    652
0000H  1038H  LIN    656
0000H  1059H  LIN    659
0000H  1066H  LIN    662
0000H  1071H  LIN    664
0000H  0F98H  LIN    666
0000H  0FA8H  LIN    669
0000H  0FB1H  LIN    672
0000H  0FB8H  LIN    675
0000H  0FCAH  LIN    677
0000H  0FD0H  LIN    680
0000H  0FDEH  LIN    682
0000H  0FF0H  LIN    686
```

```
0000H  0FF6H  LIN   687
0000H  1003H  LIN   689
0000H  100AH  LIN   692
0000H  101CH  LIN   696
0000H  1026H  LIN   698
0000H  102CH  LIN   701
0000H  1031H  LIN   703
0000H  1077H  LIN   705
0000H  1080H  LIN   707
0000H  1099H  LIN   709
0000H  10A5H  LIN   711
0000H  1199H  LIN   714
0000H  11A4H  LIN   716
0000H  11B7H  LIN   718
0000H  11C0H  LIN   720
0000H  11CDH  LIN   722
0000H  11D2H  LIN   724
0000H  11ECH  LIN   726
0000H  11EFH  LIN   728
0000H  10A8H  LIN   730
0000H  10B3H  LIN   732
0000H  10BAH  LIN   734
0000H  10C8H  LIN   737
0000H  10D4H  LIN   740
0000H  10DDH  LIN   742
0000H  10E3H  LIN   744
0000H  10F3H  LIN   746
0000H  1100H  LIN   748
0000H  110EH  LIN   750
0000H  111EH  LIN   752
0000H  1140H  LIN   754
0000H  114AH  LIN   757
0000H  1158H  LIN   759
0000H  1164H  LIN   761
0000H  1171H  LIN   763
0000H  117BH  LIN   765
0000H  1184H  LIN   767
0000H  1191H  LIN   770
0000H  1203H  LIN   772
0000H  120DH  LIN   774
0000H  121AH  LIN   776
0000H  1222H  LIN   778
0000H  122CH  LIN   780
0000H  1234H  LIN   782
0000H  123AH  LIN   784
0000H  1244H  LIN   786
0000H  1255H  LIN   788
0000H  1265H  LIN   792
0000H  126EH  LIN   794
0000H  1291H  LIN   796
0000H  129DH  LIN   798
0000H  12ABH  LIN   801
0000H  12B5H  LIN   803
0000H  12BAH  LIN   808
0000H  12C4H  LIN   810
0000H  12D2H  LIN   813
0000H  12E7H  LIN   815
0000H  1301H  LIN   817
0000H  1319H  LIN   819
0000H  1329H  LIN   821

0000H  0FF8H  LIN   688
0000H  1003H  LIN   691
0000H  1011H  LIN   694
0000H  1021H  LIN   697
0000H  102AH  LIN   700
0000H  102CH  LIN   702
0000H  1033H  LIN   704
0000H  107AH  LIN   706
0000H  1087H  LIN   708
0000H  10A5H  LIN   710
0000H  1196H  LIN   713
0000H  11A2H  LIN   715
0000H  11ADH  LIN   717
0000H  11C0H  LIN   719
0000H  11C3H  LIN   721
0000H  11D2H  LIN   723
0000H  11D5H  LIN   725
0000H  11ECH  LIN   727
0000H  1203H  LIN   729
0000H  10ADH  LIN   731
0000H  10B3H  LIN   733
0000H  10C1H  LIN   736
0000H  10CAH  LIN   738
0000H  10DBH  LIN   741
0000H  10DDH  LIN   743
0000H  10EDH  LIN   745
0000H  10F9H  LIN   747
0000H  1107H  LIN   749
0000H  1113H  LIN   751
0000H  113AH  LIN   753
0000H  1144H  LIN   756
0000H  1151H  LIN   758
0000H  115EH  LIN   760
0000H  116AH  LIN   762
0000H  1175H  LIN   764
0000H  117DH  LIN   766
0000H  118EH  LIN   768
0000H  1194H  LIN   771
0000H  1206H  LIN   773
0000H  1213H  LIN   775
0000H  1220H  LIN   777
0000H  1225H  LIN   779
0000H  1232H  LIN   781
0000H  1237H  LIN   783
0000H  123DH  LIN   785
0000H  124BH  LIN   787
0000H  125CH  LIN   790
0000H  126EH  LIN   793
0000H  128AH  LIN   795
0000H  129BH  LIN   797
0000H  12A4H  LIN   800
0000H  12AEH  LIN   802
0000H  12B7H  LIN   805
0000H  12BFH  LIN   809
0000H  12C8H  LIN   811
0000H  12E0H  LIN   814
0000H  12F5H  LIN   816
0000H  130DH  LIN   818
0000H  1310H  LIN   820
0000H  1335H  LIN   822
```

```
0000H  1339H  LIN    823      0000H  1340H  LIN    824
0000H  1347H  LIN    826      0000H  134AH  LIN    827
0000H  134FH  LIN    828      0000H  1354H  LIN    829
0000H  1358H  LIN    831      0000H  135DH  LIN    832
0000H  1360H  LIN    833      0000H  1369H  LIN    835
0000H  136FH  LIN    836      0000H  1381H  LIN    837
0000H  1383H  LIN    839      0000H  1397H  LIN    840
0000H  13A0H  LIN    841      0000H  13A9H  LIN    842
0000H  13ACH  LIN    843      0000H  13B2H  LIN    845
0000H  13B4H  LIN    846      0000H  1397H  LIN    848
0000H  13C7H  LIN    850      0000H  13D0H  LIN    851
0000H  13DCH  LIN    852      0000H  13EDH  LIN    854
0000H  13F0H  LIN    855      0000H  13F2H  LIN    857
0000H  13F8H  LIN    859      0000H  13FAH  LIN    860
0000H  14F2H  LIN    864      0000H  14F5H  LIN    865
0000H  14F8H  LIN    866      0000H  14FFH  LIN    867
0000H  1505H  LIN    868      0000H  150CH  LIN    869
0000H  150CH  LIN    870      0000H  1528H  LIN    872
0000H  152EH  LIN    873      0000H  1534H  LIN    874
0000H  1539H  LIN    875      0000H  153BH  LIN    877
0000H  1542H  LIN    879      0000H  1548H  LIN    881
0000H  1548H  LIN    882      0000H  154FH  LIN    884
0000H  155DH  LIN    885      0000H  1561H  LIN    886
0000H  1563H  LIN    887      0000H  156DH  LIN    888
0000H  156FH  LIN    889      0000H  13FDH  LIN    890
0000H  1407H  LIN    891      0000H  140CH  LIN    892
0000H  140FH  LIN    893      0000H  1412H  LIN    894
0000H  1415H  LIN    895      0000H  141FH  LIN    896
0000H  142AH  LIN    898      0000H  1431H  LIN    899
0000H  1431H  LIN    900      0000H  143EH  LIN    902
0000H  1448H  LIN    903      0000H  144DH  LIN    906
0000H  1453H  LIN    907      0000H  1456H  LIN    908
0000H  145DH  LIN    909      0000H  1460H  LIN    910
0000H  1467H  LIN    911      0000H  1470H  LIN    912
0000H  1473H  LIN    913      0000H  147EH  LIN    915
0000H  1481H  LIN    916      0000H  1487H  LIN    917
0000H  148EH  LIN    918      0000H  1490H  LIN    920
0000H  1496H  LIN    921      0000H  1499H  LIN    922
0000H  149FH  LIN    923      0000H  14AAH  LIN    924
0000H  14ADH  LIN    925      0000H  1481H  LIN    927
0000H  14BCH  LIN    928      0000H  14C1H  LIN    929
0000H  14CAH  LIN    930      0000H  14D1H  LIN    931
0000H  14D3H  LIN    932      0000H  14DEH  LIN    934
0000H  14E4H  LIN    935      0000H  14E6H  LIN    938
0000H  14EDH  LIN    939      0000H  14F0H  LIN    940
0000H  1571H  LIN    941      0000H  16BBH  LIN    943
0000H  16BEH  LIN    945      0000H  16C1H  LIN    946
0000H  16CDH  LIN    947      0000H  16DEH  LIN    949
0000H  16E2H  LIN    950      0000H  16F8H  LIN    951
0000H  16EFH  LIN    953      0000H  16F5H  LIN    954
0000H  16F7H  LIN    955      0000H  16FAH  LIN    956
0000H  1703H  LIN    958      0000H  1706H  LIN    959
0000H  1708H  LIN    960      0000H  1712H  LIN    961
0000H  1719H  LIN    962      0000H  1733H  LIN    963
0000H  173EH  LIN    964      0000H  1742H  LIN    965
0000H  1745H  LIN    966      0000H  1747H  LIN    967
0000H  1748H  LIN    968      0000H  1574H  LIN    969
0000H  1578H  LIN    970      0000H  1587H  LIN    971
0000H  1587H  LIN    972      0000H  158AH  LIN    973
0000H  1591H  LIN    974      0000H  1596H  LIN    975
```

```
0000H  159DH  LIN    976
0000H  15C2H  LIN    978
0000H  15C9H  LIN    980
0000H  15D2H  LIN    983
0000H  15E3H  LIN    985
0000H  15EFH  LIN    987
0000H  15F8H  LIN    990
0000H  1620H  LIN    992
0000H  162CH  LIN    995
0000H  1640H  LIN    998
0000H  1650H  LIN   1001
0000H  165CH  LIN   1003
0000H  1678H  LIN   1005
0000H  1689H  LIN   1008
0000H  1695H  LIN   1010
0000H  169CH  LIN   1013
0000H  16A5H  LIN   1015
0000H  16B6H  LIN   1019
0000H  1748H  LIN   1021
0000H  1760H  LIN   1023
0000H  1765H  LIN   1025
0000H  1776H  LIN   1027
0000H  178AH  LIN   1029
0000H  178DH  LIN   1031
0000H  1799H  LIN   1034
0000H  179FH  LIN   1036
0000H  17A9H  LIN   1038
0000H  19E4H  LIN   1042
0000H  19F3H  LIN   1046
0000H  1A06H  LIN   1048
0000H  1A12H  LIN   1050
0000H  1A15H  LIN   1052
0000H  1A30H  LIN   1054
0000H  1A37H  LIN   1056
0000H  1A3FH  LIN   1059
0000H  1A4AH  LIN   1061
0000H  1A50H  LIN   1063
0000H  1A59H  LIN   1066
0000H  1A88H  LIN   1069
0000H  1A95H  LIN   1072
0000H  1AA5H  LIN   1076
0000H  1ACEH  LIN   1079
0000H  1AD6H  LIN   1081
0000H  1AECH  LIN   1083
0000H  1B0DH  LIN   1085
0000H  1B1CH  LIN   1087
0000H  1B25H  LIN   1090
0000H  1B35H  LIN   1094
0000H  1B3EH  LIN   1096
0000H  17BCH  LIN   1098
0000H  17C6H  LIN   1100
0000H  17D4H  LIN   1102
0000H  17E2H  LIN   1104
0000H  17EAH  LIN   1106
0000H  17F4H  LIN   1108
0000H  17FFH  LIN   1110
0000H  1811H  LIN   1112
0000H  1822H  LIN   1114
0000H  1830H  LIN   1116
0000H  1835H  LIN   1118

0000H  15A3H  LIN    977
0000H  15C6H  LIN    979
0000H  15CBH  LIN    981
0000H  15D9H  LIN    984
0000H  15ECH  LIN    986
0000H  15F1H  LIN    989
0000H  1612H  LIN    991
0000H  1627H  LIN    994
0000H  1631H  LIN    996
0000H  1649H  LIN   1000
0000H  1657H  LIN   1002
0000H  1678H  LIN   1004
0000H  1682H  LIN   1007
0000H  1693H  LIN   1009
0000H  169AH  LIN   1011
0000H  16A0H  LIN   1014
0000H  16B3H  LIN   1016
0000H  1689H  LIN   1020
0000H  174EH  LIN   1022
0000H  1763H  LIN   1024
0000H  1768H  LIN   1026
0000H  177AH  LIN   1028
0000H  178AH  LIN   1030
0000H  1797H  LIN   1033
0000H  179CH  LIN   1035
0000H  17A6H  LIN   1037
0000H  17ABH  LIN   1039
0000H  19E7H  LIN   1045
0000H  1A02H  LIN   1047
0000H  1A0CH  LIN   1049
0000H  1A12H  LIN   1051
0000H  1A28H  LIN   1053
0000H  1A35H  LIN   1055
0000H  1A3AH  LIN   1058
0000H  1A47H  LIN   1060
0000H  1A4CH  LIN   1062
0000H  1A53H  LIN   1065
0000H  1A7CH  LIN   1067
0000H  1A8FH  LIN   1071
0000H  1A97H  LIN   1074
0000H  1AADH  LIN   1077
0000H  1AD4H  LIN   1080
0000H  1AE5H  LIN   1082
0000H  1AF8H  LIN   1084
0000H  1B0FH  LIN   1086
0000H  1B21H  LIN   1089
0000H  1B2BH  LIN   1091
0000H  1B38H  LIN   1095
0000H  17B4H  LIN   1097
0000H  17C1H  LIN   1099
0000H  17CDH  LIN   1101
0000H  17DBH  LIN   1103
0000H  17E7H  LIN   1105
0000H  17ECH  LIN   1107
0000H  17F8H  LIN   1109
0000H  1805H  LIN   1111
0000H  181CH  LIN   1113
0000H  182DH  LIN   1115
0000H  1830H  LIN   1117
0000H  1842H  LIN   1119
```

```
0000H  1849H  LIN  1120
0000H  1852H  LIN  1122
0000H  185DH  LIN  1124
0000H  1865H  LIN  1126
0000H  1876H  LIN  1130
0000H  1889H  LIN  1132
0000H  1897H  LIN  1135
0000H  18A1H  LIN  1137
0000H  18ADH  LIN  1139
0000H  18B2H  LIN  1142
0000H  18BBH  LIN  1146
0000H  18CCH  LIN  1148
0000H  1909H  LIN  1150
0000H  1910H  LIN  1153
0000H  1921H  LIN  1155
0000H  1928H  LIN  1157
0000H  192FH  LIN  1160
0000H  1937H  LIN  1163
0000H  1941H  LIN  1165
0000H  194BH  LIN  1168
0000H  1952H  LIN  1170
0000H  1969H  LIN  1172
0000H  1972H  LIN  1174
0000H  1981H  LIN  1176
0000H  1986H  LIN  1178
0000H  1992H  LIN  1180
0000H  19A9H  LIN  1182
0000H  19BDH  LIN  1184
0000H  19C3H  LIN  1186
0000H  19CDH  LIN  1188
0000H  19D9H  LIN  1190
0000H  19DFH  LIN  1193
0000H  0103H  LIN  1195
0000H  011AH  LIN  1198
0000H  012EH  LIN  1201
0000H  0138H  LIN  1204
0000H  0149H  LIN  1207
0000H  0153H  LIN  1210
0000H  0162H  LIN  1212
0000H  016CH  LIN  1215
0000H  0176H  LIN  1218
0000H  0185H  LIN  1221
0000H  0196H  LIN  1223
0000H  01A1H  LIN  1225
0000H  01A7H  LIN  1227
0000H  01ADH  LIN  1229
0000H  01B8H  LIN  1231
0000H  01C0H  LIN  1233
0000H  01CDH  LIN  1236
0000H  01D3H  LIN  1239
0000H  01DFH  LIN  1242
0000H  01E5H  LIN  1245
0000H  01F3H  LIN  1247
0000H  0200H  LIN  1249
0000H  0211H  LIN  1251
0000H  021FH  LIN  1254
0000H  0229H  LIN  1256
0000H  022FH  LIN  1258
0000H  023BH  LIN  1261
0000H  024CH  LIN  1263

0000H  1843H  LIN  1121
0000H  185AH  LIN  1123
0000H  1863H  LIN  1125
0000H  186FH  LIN  1128
0000H  1887H  LIN  1131
0000H  188CH  LIN  1133
0000H  189EH  LIN  1136
0000H  18A5H  LIN  1139
0000H  18AFH  LIN  1141
0000H  18B9H  LIN  1143
0000H  18C0H  LIN  1147
0000H  18D1H  LIN  1149
0000H  1909H  LIN  1151
0000H  191AH  LIN  1154
0000H  1924H  LIN  1156
0000H  192AH  LIN  1159
0000H  1935H  LIN  1161
0000H  193EH  LIN  1164
0000H  1944H  LIN  1167
0000H  194DH  LIN  1169
0000H  1959H  LIN  1171
0000H  1970H  LIN  1173
0000H  1979H  LIN  1175
0000H  1984H  LIN  1177
0000H  198BH  LIN  1179
0000H  19A2H  LIN  1181
0000H  19ABH  LIN  1183
0000H  19C1H  LIN  1185
0000H  19CAH  LIN  1187
0000H  19D1H  LIN  1189
0000H  19DDH  LIN  1191
0000H  19E2H  LIN  1194
0000H  010CH  LIN  1197
0000H  0127H  LIN  1199
0000H  0135H  LIN  1202
0000H  0142H  LIN  1205
0000H  0150H  LIN  1208
0000H  015CH  LIN  1211
0000H  0167H  LIN  1213
0000H  0173H  LIN  1217
0000H  017CH  LIN  1220
0000H  0190H  LIN  1222
0000H  019EH  LIN  1224
0000H  01A4H  LIN  1226
0000H  01AAH  LIN  1228
0000H  01B2H  LIN  1230
0000H  01BBH  LIN  1232
0000H  01C7H  LIN  1235
0000H  01D0H  LIN  1237
0000H  01D8H  LIN  1240
0000H  01E2H  LIN  1243
0000H  01ECH  LIN  1246
0000H  01FDH  LIN  1248
0000H  020EH  LIN  1250
0000H  0218H  LIN  1252
0000H  0226H  LIN  1255
0000H  022CH  LIN  1257
0000H  0234H  LIN  1259
0000H  0249H  LIN  1262
0000H  024CH  LIN  1265
```

```
0000H  0258H  LIN   1266
0000H  025AH  LIN   1269
0000H  0268H  LIN   1272
0000H  0273H  LIN   1274
0000H  0284H  LIN   1276
0000H  028DH  LIN   1278
0000H  029DH  LIN   1281
0000H  02B0H  LIN   1283
0000H  02BDH  LIN   1285
0000H  02C9H  LIN   1287
0000H  02DAH  LIN   1289
0000H  02F1H  LIN   1291
0000H  02FDH  LIN   1294
0000H  0309H  LIN   1296
0000H  0315H  LIN   1299
0000H  0330H  LIN   1301
0000H  033CH  LIN   1303
0000H  034FH  LIN   1305
0000H  035DH  LIN   1307
0000H  036CH  LIN   1309
0000H  0372H  LIN   1311
0000H  0380H  LIN   1313
0000H  0386H  LIN   1316
0000H  0100H  LIN   1318

0000H  0258H  LIN   1268
0000H  0268H  LIN   1271
0000H  026EH  LIN   1273
0000H  0281H  LIN   1275
0000H  0286H  LIN   1277
0000H  0296H  LIN   1280
0000H  02A2H  LIN   1282
0000H  02B7H  LIN   1284
0000H  02C4H  LIN   1286
0000H  02CCH  LIN   1288
0000H  02E0H  LIN   1290
0000H  02F8H  LIN   1293
0000H  0304H  LIN   1295
0000H  0310H  LIN   1297
0000H  032AH  LIN   1300
0000H  0337H  LIN   1302
0000H  0343H  LIN   1304
0000H  0355H  LIN   1306
0000H  0364H  LIN   1308
0000H  036FH  LIN   1310
0000H  0379H  LIN   1312
0000H  0383H  LIN   1315
0000H  0386H  LIN   1317
```


MEMORY MAP OF MODULE SCD1
READ FROM FILE PIP.LNK
WRITTEN TO FILE PIP.

SEGMENT MAP

START	STOP	LENGTH	ALIGN	NAME	CLASS
00000H	01B3FH	1B40H	G	CODE	CODE
10000H	100FFH	0100H	G	DATS	DATA
10100H	10296H	0197H	W	DATA	DATA
10298H	10323H	008CH	W	STACK	STACK
10324H	10725H	0402H	W	CONST	CONST
10730H	10730H	0000H	G	??SEG	
10730H	10730H	0000H	W	MEMORY	MEMORY

GROUP MAP

ADDRESS	GROUP OR SEGMENT NAME
00000H	CGROUP
	CODE
10000H	DGROUP
	DATS
	STACK
	CONST
	DATA
	MEMORY