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



```
          $ TITLE(' CONCURRENT CP/M-86 2.0  ---  ED')
1         ED:
          DO;
              /* MODIFIED FOR .PRL OPERATION MAY, 1979 */
              /* MODIFIED FOR OPERATION WITH CP/M 2.0 AUGUST 1979 */
              /* modified for MP/M 2.0 June 1981 */
              /* modified for CP/M 1.1 Oct  1981 */
              /* modified for CONCURRENT CP/M 1.0 Jul 1982 */
              /* modified for CP/M 3.0 July 1982 */
              /* modified for CP/M 3.0 SEPT 1982 */
              /* modified for CONCURRENT CP/M-86 2.0 October 1982 */

          /* MODIFICATION LOG:
           *   July 1982 whf: some code cleanup (grouped logicals, declared BOOL);
           *        fixed disk full error handling; fixed read from null files;
           *        fixed (some) of the dirty fcb handling (shouldn't use settype
           *        function on open fcbs!).
           *   July 1982 dh: installed patches to change macro abort command from
           *        ^C to ^Y and to not print error message when trying to delete
           *        a file that doesn't exist.  Added PERROR: PROCEDURE to print
           *        error messages in a consistant format and modified error
           *        message handler at RESET: entry point.  Also corrected Invalid
           *        filename error to not abort ED if parsing a R or X command.
           *        Modified start (at PLM:) and SETDEST: to prompt for missing
           *        filenames.  Modified parse$fcb & parse$lib to set a global
           *        flag and break if it got an invalid filename for X or R commands.
           *        Start sets page size from the system control block (SCB) if
           *        ED is running under CP/M-80 (high(ver)=0).
           *        The H command now works with new files. (sets newfile=false)
           *     Sept 82
           *        Corrected bug in which ED file b: didn't work. Changed PLM:
           *        and SETDEST: routines.
           *     Oct 82  whf
           *        Changed version numbers for Concurrent, ED entry point changed to
           *        PLMSTART.
           */


          /**** VAX commands for ED generation.

              $ util := ED
              $ ccpmsetup             ! set up environment
              $ assign "f$directory()" f1:        ! use local dir for temp files
              $ plm86 "util".plm xref "p1" optimize(3) debug
              $ link86 f2:scd.obj, "util".obj  to "util".lnk
              $ loc86 "util".lnk od(sm(code,dats,data,stack,const)) -
                   ad(sm(code(0),dats(10000h))) ss(stack(+32)) to "util".
              $ h86 "util"

          ***** Then, on a micro:
              A>vax ed.h86 $fans
```

```
            A>gencmd ed data[b1000 m80 xfff]

          ***** Notes: Stack is increased for interrupts.  Const(ants) are last
                 to force hex generation.  'm80' is the min number of pgraphs,
                 and is found in ED.MP2.

          ****/

          $include (:f1:copyrt.lit)

  2   1   declare
              mpmproduct literally '01h', /* requires mp/m */
              cpm3       literally '30h'; /* requires 3.0 cp/m */

  3   1   declare plmstart label public;  /* entry point for plm86 interface */


          /* DECLARE                              8080 Interface
              JMP EDCOMMAND - 3 (TO ADDRESS LXI SP)
              EDJMP BYTE DATA(0C3H),
              EDADR ADDRESS DATA(.EDCOMMAND-3); */


            /****************************************
             *                                      *
             *       B D O S   INTERFACE            *
             *                                      *
             ****************************************/


  4   1     mon1:
            procedure (func,info) external;
  5   2       declare func byte;
  6   2       declare info address;
  7   2     end mon1;

  8   1     mon2:
            procedure (func,info) byte external;
  9   2       declare func byte;
 10   2       declare info address;
 11   2     end mon2;

 12   1     mon3:
            procedure (func,info) address external;
 13   2       declare func byte;
 14   2       declare info address;
 15   2     end mon3;

 16   1     declare fcb (1)   byte    external;   /* 1st default fcb    */
 17   1     declare fcb16 (1) byte    external;   /* 2nd default fcb    */
 18   1     declare tbuff (1) byte    external;   /* default dma buffer */


 19   1   DECLARE
              MAXB      ADDRESS EXTERNAL, /* MAX BASE 0006H */
              BUFF (128)BYTE    EXTERNAL, /* BUFFER 0080H */
              SECTSHF   LITERALLY '7',    /* SHL(1,SECTSHF) = SECTSIZE */
```

```
            SECTSIZE  LITERALLY '80H';  /* SECTOR SIZE */

20   1      BOOT: PROCEDURE ;
21   2          call mon1(0,0);          /* changed for MP/M-86 version */
               /* SYSTEM REBOOT */
22   2          END BOOT;
```

```
$ eject

 /* E D :  T H E  C P / M  C O N T E X T  E D I T O R */

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

           Revised:
             07 April 81  by Thomas Rolander
             21 July  81  by Doug Huskey
             29 Oct   81  by Doug Huskey
             10 Nov   81  by Doug Huskey
             08 July  82  by Bill Fitler
             26 July  82  by Doug Huskey
   */
/* DECLARE COPYRIGHT(*) BYTE DATA
   (' COPYRIGHT (C) 1983, DIGITAL RESEARCH ');
   **** this message should be in the header ***
*/
23  1    declare date(*) byte data ('2/83');

 /* COMMAND          FUNCTION
    -------          --------
     A               APPEND LINES OF TEXT TO BUFFER
     B               MOVE TO BEGINNING OR END OF TEXT
     C               SKIP CHARACTERS
     D               DELETE CHARACTERS
     E               END OF EDIT
     F               FIND STRING IN CURRENT BUFFER
     H               MOVE TO TOP OF FILE (HEAD)
     I               INSERT  CHARACTERS FROM KEYBOARD
                     UP TO NEXT <ENDFILE>
     J               JUXTAPOSITION OPERATION - SEARCH FOR FIRST STRING,
                     INSERT SECOND STRING, DELETE UNTIL THIRD STRING
     K               DELETE LINES
     L               SKIP LINES
     M               MACRO DEFINITION (SEE COMMENT BELOW)
     N               FIND NEXT OCCURRENCE OF STRING
                     WITH AUTO SCAN THROUGH FILE
     O               RE-EDIT OLD FILE
     P               PAGE AND DISPLAY (MOVES UP OR DOWN 24 LINES AND
                     DISPLAYS 24 LINES)
     Q               QUIT EDIT WITHOUT UPDATING THE FILE
     R<FILENAME>     READ FROM FILE <FILENAME> UNTIL <ENDFILE> AND
                     INSERT INTO TEXT
     S               SEARCH FOR FIRST STRING, REPLACE BY SECOND STRING
     T               TYPE LINES
     U               TRANSLATE TO UPPER CASE (-U CHANGES TO NO TRANSLATE)
     W               WRITE LINES OF TEXT TO FILE
     X<FILENAME>     TRANSFER (XFER) LINES TO FILE <FILENAME>
     Z               SLEEP FOR 1/2 SECOND (USED IN MACROS TO STOP DISPLAY)
     <CR>            MOVE UP OR DOWN AND PRINT ONE LINE

   IN GENERAL, THE EDITOR ACCEPTS SINGLE LETTER COMMANDS WITH OPTIONAL
```

INTEGER VALUES PRECEDING THE COMMAND. THE EDITOR ACCEPTS BOTH UPPER AND LOWER CASE COMMANDS AND VALUES, AND PERFORMS TRANSLATION TO UPPER CASE UNDER THE FOLLOWING CONDITIONS. IF THE COMMAND IS TYPED IN UPPER CASE, THEN THE DATA WHICH FOLLOWS IS TRANSLATED TO UPPER CASE. THUS, IF THE "I" COMMAND IS TYPED IN UPPER CASE, THEN ALL INPUT IS AUTOMATICALLY TRANSLATED (ALTHOUGH ECHOED IN LOWER CASE, AS TYPED). IF THE "A" COMMAND IS TYPED IN UPPER CASE, THEN ALL INPUT IS TRANSLATED AS READ FROM THE DISK. GLOBAL TRANSLATION TO UPPER CASE CAN BE CONTROLLED BY THE "U" COMMAND (-U TO NEGATE ITS EFFECT). IF YOU ARE OPERATING WITH AN UPPER CASE ONLY TERMINAL, THEN OPERATION IS AUTOMATIC. SIMILARLY, IF YOU ARE OPERATING WITH A LOWER CASE TERMINAL, AND TRANSLATION TO UPPER CASE IS NOT SPECIFIED, THEN LOWER CASE CHARACTERS CAN BE ENTERED.

A NUMBER OF COMMANDS CAN BE PRECEDED BY A POSITIVE OR NEGATIVE INTEGER BETWEEN 0 AND 65535 (1 IS DEFAULT IF NO VALUE IS SPECIFIED). THIS VALUE DETERMINES THE NUMBER OF TIMES THE COMMAND IS APPLIED BEFORE RETURNING FOR ANOTHER COMMAND.
THE COMMANDS

C D K L T P U <CR>

CAN BE PRECEDED BY AN UNSIGNED, POSITIVE, OR NEGATIVE NUMBER, THE COMMANDS

A F J N W Z

CAN BE PRECEDED BY AN UNSIGNED OR POSITIVE NUMBER, THE COMMANDS

E H O Q

CANNOT BE PRECEDED BY A NUMBER. THE COMMANDS

F I J M R S

ARE ALL FOLLOWED BY ONE OR MORE STRINGS OF CHARACTERS WHICH CAN BE OPTIONALLY SEPARATED OR TERMINATED BY EITHER <ENDFILE> OR <CR>. THE <ENDFILE> IS GENERALLY USED TO SEPARATE THE SEARCH STRINGS IN THE S AND J COMMANDS, AND IS USED AT THE END OF THE COMMANDS IF ADDITIONAL COMMANDS FOLLOW. FOR EXAMPLE, THE FOLLOWING COMMAND SEQUENCE SEARCHES FOR THE STRING 'GAMMA', SUBSTITUTES THE STRING 'DELTA', AND THEN TYPES THE FIRST PART OF THE LINE WHERE THE CHANGE OCCURRED, FOLLOWED BY THE REMAINDER OF THE LINE WHICH WAS CHANGED:

SGAMMA<ENDFILE>DELTA<ENDFILE>OTT<CR>

THE CONTROL-L CHARACTER IN SEARCH AND SUBSTITUTE STRINGS IS REPLACED ON INPUT BY <CR><LF> CHARACTERS. THE CONTROL-I KEY IS TAKEN AS A TAB CHARACTER.

THE COMMANDS R & X MUST BE FOLLOWED BY A FILE NAME (WITH default FILE TYPE OF 'LIB') WITH A TRAILING <CR> OR <ENDFILE>. THE COMMAND I IS FOLLOWED BY A STRING OF SYMBOLS TO INSERT, TERMINATED BY A <CR> OR <ENDFILE>. IF SEVERAL LINES OF TEXT ARE TO BE INSERTED, THE I CAN BE DIRECTLY FOLLOWED BY AN <ENDFILE> OR <CR> IN WHICH CASE THE EDITOR ACCEPTS LINES OF INPUT TO THE NEXT <ENDFILE>. THE COMMAND OT PRINTS THE FIRST PART OF THE CURRENT LINE, AND THE COMMAND OL MOVES THE REFERENCE TO THE BEGINNING OF THE CURRENT LINE. THE COMMAND OP PRINTS THE CURRENT PAGE ONLY, WHILE THE COMMAND OZ READS THE CONSOLE RATHER THAN WAITING (THIS IS USED AGAIN WITHIN MACROS TO STOP THE DISPLAY - THE MACRO EXPANSION STOPS UNTIL A CHARACTER IS READ. IF THE CHARACTER IS NOT A BREAK THEN THE MACRO EXPANSION CONTINUES NORMALLY).

NOTE THAT A POUND SIGN IS TAKEN AS THE NUMBER 65535, ALL UNSIGNED NUMBERS ARE ASSUMED POSITIVE, AND A SINGLE - IS ASSUMED -1

A NUMBER OF COMMANDS CAN BE GROUPED TOGETHER AND EXECUTED REPETITIVELY USING THE MACRO COMMAND WHICH TAKES THE FORM

<NUMBER>MC1C2...CN<DELIMITER>

WHERE <NUMBER> IS A NON-NEGATIVE INTEGER N, AND <DELIMITER> IS <ENDFILE> OR <CR>. THE COMMANDS C1 ... CN FOLLOWING THE M ARE EXECUTED N TIMES, STARTING AT THE CURRENT POSITION IN THE BUFFER. IF N IS 0, 1, OR OMITTED, THE COMMANDS ARE EXECUTED UNTIL THE END IF THE BUFFER IS ENCOUNTERED.

THE FOLLOWING MACRO, FOR EXAMPLE, CHANGES ALL OCCURRENCES OF THE NAME 'GAMMA' TO 'DELTA', AND PRINTS THE LINES WHICH WERE CHANGED:

MFGAMMA<ENDFILE>-5DIDELTA<ENDFILE>0LT<CR>

(NOTE: AN <ENDFILE> IS THE CP/M END OF FILE MARK - CONTROL-Z)

IF ANY KEY IS DEPRESSED DURING TYPING OR MACRO EXPANSION, THE FUNCTION IS CONSIDERED TERMINATED, AND CONTROL RETURNS TO THE OPERATOR.

ERROR CONDITIONS ARE INDICATED BY PRINTING ONE OF THE CHARACTERS:

SYMBOL	ERROR CONDITION
GREATER	FREE MEMORY IS EXHAUSTED - ANY COMMAND CAN BE ISSUED WHICH DOES NOT INCREASE MEMORY REQUIREMENTS.
QUESTION	UNRECOGNIZED COMMAND OR ILLEGAL NUMERIC FIELD
POUND	CANNOT APPLY THE COMMAND THE NUMBER OF TIMES SPECFIED (OCCURS IF SEARCH STRING CANNOT BE FOUND)
LETTER O	CANNOT OPEN <FILENAME>.LIB IN R COMMAND

THE ERROR CHARACTER IS ALSO ACCOMPANIED BY THE LAST CHARACTER SCANNED WHEN THE ERROR OCCURRED. */

```
              $ eject
              /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *


                           * * *  GLOBAL VARIABLES * * *


               * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

  24   1      DECLARE LIT LITERALLY 'LITERALLY',
                  DCL LIT 'DECLARE',
                  PROC LIT 'PROCEDURE',
                  ADDR LIT 'ADDRESS',
                  BOOLEAN LIT 'BYTE',
                  CTLL LIT '0CH',
                  CTLR LIT '12H',                 /* REPEAT LINE IN INSERT MODE */
                  CTLU LIT '15H',                 /* LINE DELETE IN INSERT MODE */
                  CTLX LIT '18H',                 /* EQUIVALENT TO CTLU */
                  CTLH LIT '08H',                 /* BACKSPACE */
                  TAB LIT '09H',                  /* TAB CHARACTER */
                  LCA  LIT '110$0001B',           /* LOWER CASE A */
                  LCZ  LIT '111$1010B',           /* LOWER CASE Z */
                  ESC  LIT '1BH',                 /* ESCAPE CHARACTER */
                  ENDFILE LIT '1AH';              /* CP/M END OF FILE */

  25   1      DECLARE
                  TRUE LITERALLY '1',
                  FALSE LITERALLY '0',
                  FOREVER LITERALLY 'WHILE TRUE',
                  CTRL$Y LITERALLY '19h',
                  CR LITERALLY '13',
                  LF LITERALLY '10',
                  WHAT LITERALLY '63';

  26   1      DECLARE
                  MAX ADDRESS,                    /* .MEMORY(MAX)=0 (END) */
                  MAXM ADDRESS,                   /* MINUS 1 */
                  HMAX ADDRESS;                   /* = MAX/2 */

  27   1      declare
                  i  byte;                        /* used by command parsing */

  28   1      DECLARE
                  us literally '8',               /* file from user 0   */
                  RO LITERALLY '9',               /* R/O FILE INDICATOR */
                  SY LITERALLY '10',              /* SYSTEM FILE ATTRIBUTE */
                  EX LITERALLY '12',              /* EXTENT NUMBER POSITION */
                  UB LITERALLY '13',              /* UNFILLED BYTES */
                  ck LITERALLY '13',              /* checksum */
                  MD LITERALLY '14',              /* MODULE NUMBER POSITION */
                  NR LITERALLY '32',              /* NEXT RECORD FIELD */
                  FS LITERALLY '33',              /* FCB SIZE */
                  RFCB (FS) BYTE                  /* READER FILE CONTROL BLOCK */
                      INITIAL(0, /* FILE NAME */ '        ',
                                 /* FILE TYPE */ 'LIB',0,0,0),
                  RBP BYTE,                       /* READ BUFFER POINTER */
```

```
            XFCB (FS) BYTE                  /* XFER FILE CONTROL BLOCK */
                INITIAL(0, 'X$$$$$$$','LIB',0,0,0,0,0,0,0),
            XFCBE BYTE AT(.XFCB(EX)),       /* XFCB EXTENT */
            XFCBR BYTE AT(.XFCB(NR)),       /* XFCB RECORD # */
            xfcbext byte initial(0),        /* save xfcb extent for appends */
            xfcbrec byte initial(0),        /* save xfcb record for appends */
            XBUFF (SECTSIZE) BYTE,          /* XFER BUFFER */
            XBP BYTE,                       /* XFER POINTER */

            NBUF BYTE,                      /* NUMBER OF BUFFERS */
            BUFFLENGTH ADDRESS,             /* NBUF * SECTSIZE */
            SFCB (FS) BYTE AT(.FCB),        /* SOURCE FCB = DEFAULT FCB */
            SDISK BYTE AT (.FCB),           /* SOURCE DISK */
            SBUFFADR ADDRESS,               /* SOURCE BUFFER ADDRESS */
            SBUFF BASED SBUFFADR (128) BYTE, /* SOURCE BUFFER */
            password (16) byte initial(0), /* source password */

            DFCB (FS) BYTE,                 /* DEST FILE CONTROL BLOCK */
            DDISK BYTE AT (.DFCB),          /* DESTINATION DISK */
            DBUFFADR ADDRESS,               /* DESTINATION BUFFER ADDRESS */
            DBUFF BASED DBUFFADR (128) BYTE, /* DEST BUFFER */
            NSOURCE ADDRESS,                /* NEXT SOURCE CHARACTER */
            NDEST   ADDRESS,                /* NEXT DESTINATION CHAR */

            tmpfcb (FS) BYTE;               /* temporary fcb for rename & deletes */

29  1       DECLARE                                 /**** some of the logicals *****/
            newfile      BOOLEAN initial (false), /* true if no source file */
            onefile      BOOLEAN initial (true),  /* true if output file=input file */
            XFERON       BOOLEAN initial (false), /* TRUE IF XFER ACTIVE */
            reading      BOOLEAN initial (false), /* TRUE IF reading RFCB */
            PRINTSUPPRESS BOOLEAN initial (false),/* TRUE IF PRINT SUPPRESSED */
            sys          BOOLEAN initial (false), /* true if system file */
            protection   BOOLEAN initial (false), /* password protection mode */
            INSERTING    BOOLEAN,                 /* TRUE IF INSERTING CHARACTERS */
            READBUFF     BOOLEAN,                 /* TRUE IF END OF READ BUFFER */
            TRANSLATE    BOOLEAN initial (false), /* TRUE IF XLATION TO UPPER CASE */
            UPPER        BOOLEAN initial (false), /* TRUE IF GLOBALLY XLATING TO UC */
            LINESET      BOOLEAN initial (true),  /* TRUE IF LINE #'S PRINTED */
            has$bdos3    BOOLEAN initial (false), /* true if BDOS version >= 3.0 */
            tail BOOLEAN initial (true),  /* true if reading from cmd tail */
            dot$found    BOOLEAN initial (false); /* true if dot found in fname parse*/

30  1       DECLARE
            dtype  (3)  byte,                   /* destination file type */
            libfcb (12) byte initial(0,'X$$$$$$$LIB'),/* default lib name */
            tempfl (3)  byte initial('$$$'), /* temporary file type */
            backup (3)  byte initial('BAK'); /* backup file type */

31  1       declare
              error$code address;

32  1       DECLARE
            COLUMN  BYTE initial(0),  /* CONSOLE COLUMN POSITION */
            SCOLUMN BYTE INITIAL(8),  /* STARTING COLUMN IN "I" MODE */
            TCOLUMN BYTE,             /* TEMP DURING BACKSPACE */
            QCOLUMN BYTE;             /* TEMP DURING BACKSPACE */
```

```
33  1   DECLARE DCNT BYTE;          /* RETURN CODE FROM MON? CALLS */

        /* COMMAND BUFFER */
34  1   DECLARE (MAXLEN,COMLEN) BYTE, COMBUFF(128) BYTE,
            CBP BYTE initial(0);

35  1   DECLARE /* LINE COUNTERS */
            BASELINE ADDRESS,          /* CURRENT LINE */
            RELLINE  ADDRESS;          /* RELATIVE LINE IN TYPEOUT */

36  1   DECLARE
            FORWARD LIT '1',
            BACKWARD LIT '0',
            RUBOUT LIT '07FH',
            POUND LIT '23H',
            MACSIZE LIT '128',         /* MAX MACRO SIZE */
            SCRSIZE LIT '100',         /* SCRATCH BUFFER SIZE */
            COMSIZE LIT 'ADDRESS';     /* DETERMINES MAX COMMAND NUMBER*/

37  1   DCL MACRO(MACSIZE) BYTE,
            SCRATCH(SCRSIZE) BYTE,     /* SCRATCH BUFFER FOR F,N,S */
            (WBP, WBE, WBJ)  BYTE,     /* END OF F STRING, S STRING, J STRING */
            (FLAG, MP, MI, XP) BYTE,
            MT COMSIZE;

38  1   DCL (START, RESTART, OVERCOUNT, OVERFLOW,
             disk$err, dir$err, RESET, BADCOM) LABEL;

        /* global variables used by file parsing routines */
39  1   dcl ncmd byte initial(0);

40  1   DCL (DISTANCE, TDIST) COMSIZE,
            (DIRECTION, CHAR) BYTE,
            ( FRONT, BACK, FIRST, LASTC) ADDR;

41  1   dcl LPP byte initial(23);          /* LINES PER PAGE */

        /* the following stucture is used near plm: to set
           the lines per page from the BDOS 3 SCB */
42  1   declare
            pb (2) byte data (28,0);

43  1   declare
            ver address;                   /* VERSION NUMBER */

44  1   declare
            err$msg          address initial(0),
            invalid (*)      byte data ('Invalid Filename$'),
            dirfull (*)      byte data ('DIRECTORY FULL$'),
            diskfull (*)     byte data ('DISK FULL$'),
            password$err(*)  byte data ('Creating Password$'),
            not$found (*)    byte data ('File not found$'),
            notavail (*)     byte data ('File not available$');
```

```
$ eject

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

                * * *  CP/M INTERFACE ROUTINES * * *

 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */


 /* IO SECTION */

45  1   READCHAR: PROCEDURE BYTE; RETURN MON2(1,0);
47  2       END READCHAR;

48  1    conin:
         procedure byte;
49  2       return mon2(6,0fdh);
50  2       end conin;

51  1   PRINTCHAR: PROCEDURE(CHAR);
52  2       DECLARE CHAR BYTE;
53  2       IF PRINTSUPPRESS THEN RETURN;
55  2       CALL MON1(2,CHAR);
56  2       END PRINTCHAR;

57  1   TTYCHAR: PROCEDURE(CHAR);
58  2       DECLARE CHAR BYTE;
59  2       IF CHAR >= ' ' THEN COLUMN = COLUMN + 1;
61  2       IF CHAR = LF THEN COLUMN = 0;
63  2       CALL PRINTCHAR(CHAR);
64  2       END TTYCHAR;

65  1   BACKSPACE: PROCEDURE;
         /* MOVE BACK ONE POSITION */
66  2       IF COLUMN <> 0 THEN RETURN;
68  2       CALL TTYCHAR(CTLH); /* COLUMN = COLUMN - 1 */
69  2       CALL TTYCHAR(' ' ); /* COLUMN = COLUMN + 1 */
70  2       CALL TTYCHAR(CTLH); /* COLUMN = COLUMN - 1 */
71  2       COLUMN = COLUMN - 2;
72  2       END BACKSPACE;

73  1   PRINTABS: PROCEDURE(CHAR);
74  2       DECLARE (CHAR,I,J) BYTE;
75  2       I = CHAR = TAB AND 7 - (COLUMN AND 7);
76  2       IF CHAR = TAB THEN CHAR = ' ';
78  2           DO J = 0 TO I;
79  3           CALL TTYCHAR(CHAR);
80  3           END;
81  2       END PRINTABS;

82  1   GRAPHIC: PROCEDURE(C) BOOLEAN;
83  2       DECLARE C BYTE;
         /* RETURN TRUE IF GRAPHIC CHARACTER */
84  2       IF C >= ' ' THEN RETURN TRUE;
86  2       RETURN C = CR OR C = LF OR C = TAB;
```

```
 87  2          END GRAPHIC;

 88  1      PRINTC: PROCEDURE(C);
 89  2          DECLARE C BYTE;
 90  2          IF NOT GRAPHIC(C) THEN
 91  2              DO; CALL PRINTABS('^');
 93  3              C = C + '@';
 94  3              END;
 95  2          CALL PRINTABS(C);
 96  2          END PRINTC;

 97  1      CRLF: PROCEDURE;
 98  2          CALL PRINTC(CR); CALL PRINTC(LF);
100  2          END CRLF;

101  1      PRINTM: PROCEDURE(A);
102  2          DECLARE A ADDRESS;
103  2          CALL MON1(9,A);
104  2          END PRINTM;

105  1      PRINT: PROCEDURE(A);
106  2          DECLARE A ADDRESS;
107  2          CALL CRLF;
108  2          CALL PRINTM(A);
109  2          END PRINT;

110  1      perror: procedure(a);
111  2          declare a address;
112  2          call print(.(tab,'ERROR - $'));
113  2          call printm(A);
114  2          call crlf;
115  2          end perror;

116  1      READ: PROCEDURE(A);
117  2          DECLARE A ADDRESS;
118  2          CALL MON1(10,A);
119  2          END READ;

                   /* used for library files */
120  1      OPEN: PROCEDURE(FCB);
121  2           DECLARE FCB ADDRESS;
122  2           if MON2(15,FCB) = 255 then do;
124  3              flag = '0';
125  3          err$msg = .not$found;
126  3              go to reset;
127  3              end;
128  2           END OPEN;

                   /* used for main source file */
129  1      OPEN$FILE: PROCEDURE(FCB) ADDRESS;
130  2           DECLARE FCB ADDRESS;
131  2           RETURN MON3(15,FCB);
132  2           END OPEN$FILE;

133  1      CLOSE: PROCEDURE(FCB);
134  2          DECLARE FCB ADDRESS;
135  2          DCNT = MON2(16,FCB);
```

```
136   2          END CLOSE;

137   1      DELETE: PROCEDURE(FCB);
138   2          DECLARE FCB ADDRESS;
139   2          DCNT = MON2(19,FCB);
140   2          END DELETE;

141   1      DISKREAD: PROCEDURE(FCB) BYTE;
142   2          DECLARE FCB ADDRESS;
143   2          RETURN MON2(20,FCB);
144   2          END DISKREAD;

145   1      DISKWRITE: PROCEDURE(FCB) BYTE;
146   2          DECLARE FCB ADDRESS;
147   2          RETURN MON2(21,FCB);
148   2          END DISKWRITE;

149   1      RENAME: PROCEDURE(FCB);
150   2          DECLARE FCB ADDRESS;
151   2          CALL MON1(23,FCB);
152   2          END RENAME;

153   1      READCON: PROCEDURE;
154   2          MAXLEN = 128; CALL READ(.MAXLEN);
156   2          END READCON;

157   1      BREAK$KEY: PROCEDURE BOOLEAN;
158   2          IF MON2(11,0) THEN
159   2              DO; /* CLEAR CHAR */
160   3              IF MON2(1,0) = CTRL$Y THEN
161   3                  RETURN TRUE;
162   3              END;
163   2          RETURN FALSE;
164   2          END BREAK$KEY;

165   1      CSELECT: PROCEDURE BYTE;
               /* RETURN CURRENT DRIVE NUMBER */
166   2          RETURN MON2(25,0);
167   2          END CSELECT;

168   1      SETDMA: PROCEDURE(A);
169   2          DECLARE A ADDRESS;
               /* SET DMA ADDRESS */
170   2          CALL MON1(26,A);
171   2          END SETDMA;

172   1      set$attribute: procedure(FCB);
173   2          declare fcb address;
174   2          call MON1(30,FCB);
175   2          end set$attribute;

           /* The PL/M built-in procedure "MOVE" can be used to move storage,
              its definition is:

           MOVE: PROCEDURE(COUNT,SOURCE,DEST);
                   DECLARE (COUNT,SOURCE,DEST) ADDRESS;
                   / MOVE DATA FROM SOURCE TO DEST ADDRESSES, FOR COUNT BYTES /
```

```
                END MOVE;
          */
                        /* this routine is included solely for
                           enconomy of space over the use of the
                           equivalent (in-line) code generated by
                           the built-in function */
176   1     move:   proc(c,s,d);
177   2         dcl (s,d) addr, c byte;
178   2         dcl a based s byte, b based d byte;

179   2             do while (c:=c-1)<>255;
180   3             b=a; s=s+1; d=d+1;
183   3             end;
184   2         end move;

185   1     write$xfcb: PROCEDURE(FCB);
186   2         DECLARE FCB ADDRESS;
187   2         call move(8,.password,.password(8));
188   2         if MON2(103,FCB)= 0ffh then
189   2            call perror(.password$err);
190   2         END write$xfcb;

191   1     read$xfcb: PROCEDURE(FCB);
192   2         DECLARE FCB ADDRESS;
193   2         call MON1(102,FCB);
194   2         END read$xfcb;

            /* 0ff => return BDOS errors */
195   1     return$errors:
            procedure(mode);
196   2         declare mode byte;
197   2           call mon1 (45,mode);
198   2         end return$errors;

199   1     REBOOT: PROCEDURE;
200   2         IF XFERON THEN
201   2             CALL DELETE(.libfcb);
202   2         CALL BOOT;
203   2         END REBOOT;

204   1     version: procedure address;
            /* returns current cp/m version # */
205   2         return mon3(12,0);
206   2         end version;
```

```
$ eject
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

                    * * *  SUBROUTINES  * * *

 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

  /*  INPUT / OUTPUT BUFFERING ROUTINES */


                    /* abort ED and print error message */
207   1     ABORT: PROCEDURE(A);
208   2         DECLARE A ADDRESS;
209   2         CALL perror(A);
210   2         CALL REBOOT;
211   2         END ABORT;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                    /* fatal file error */
212   1     FERR: PROCEDURE;
213   2         CALL CLOSE(.DFCB); /* ATTEMPT TO CLOSE FILE FOR LATER RECOVERY */
214   2         CALL ABORT (.dirfull);
215   2         END FERR;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                    /* set password if cpm 3*/
216   1     setpassword: procedure;
217   2         if has$bdos3 then
218   2             call setdma(.password);
219   2         end setpassword;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                    /* delete file at afcb */
220   1     delete$file: procedure(afcb);
221   2         declare afcb address;
222   2         call setpassword;
223   2         call delete(afcb);
224   2         end delete$file;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                    /* rename file at afcb */
225   1     rename$file: procedure(afcb);
226   2         declare afcb address;
227   2         call delete$file(afcb+16);      /* delete new file */
228   2         call setpassword;
229   2         call rename(afcb);
230   2         end rename$file;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
```

```
                        /* make file at afcb */
231   1     make$file: procedure(afcb);
232   2         declare afcb address;
233   2         call delete$file(afcb);      /* delete file */
234   2         call setpassword;
235   2         DCNT = MON2(22,afcb);        /* create file */
236   2         end make$file;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* fill string a s for c bytes with f */
237   1     fill:   proc(s,f,c);
238   2         dcl s addr,
                  (f,c) byte,
                  a based s byte;

239   2         do while (c:=c-1)<>255;
240   3         a = f;
241   3         s = s+1;
242   3         end;
243   2         end fill;
```

```
$ eject

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

                    * * *  FILE HANDLING ROUTINES  * * *

 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */


                        /* set destination file type to type at A */
244   1     SETTYPE: PROCEDURE(afcb,A);
245   2         DECLARE (afcb, A) ADDRESS;
246   2         CALL MOVE(3,A,aFCB+9);
247   2         END SETTYPE;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* set dma to xfer buffer */
248   1     SETXDMA: PROCEDURE;
249   2         CALL SETDMA(.XBUFF);
250   2         END SETXDMA;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* fill primary source buffer  */
251   1     FILLSOURCE: PROCEDURE;
252   2         DECLARE I BYTE;
253   2         ZN: PROCEDURE;
254   3             NSOURCE = 0;
255   3             END ZN;

256   2          CALL ZN;
257   2             DO I = 0 TO NBUF;
258   3             CALL SETDMA(SBUFFADR+NSOURCE);
259   3             IF (DCNT := DISKREAD(.FCB)) <> 0 THEN
260   3                 DO; IF DCNT > 1 THEN CALL FERR;
263   4                 SBUFF(NSOURCE) = ENDFILE;
264   4                 I = NBUF;
265   4                 END;
                  ELSE
266   3                 NSOURCE = NSOURCE + SECTSIZE;
267   3             END;
268   2         CALL ZN;
269   2         END FILLSOURCE;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* get next character in source file */
270   1     GETSOURCE: PROCEDURE BYTE;
271   2         DECLARE B BYTE;
272   2         if newfile then return endfile;   /* in case they try to #a */
274   2         IF NSOURCE >= BUFFLENGTH THEN CALL FILLSOURCE;
276   2         IF (B := SBUFF(NSOURCE)) <> ENDFILE THEN
277   2             NSOURCE = NSOURCE + 1;
```

```
278   2         RETURN B;
279   2         END GETSOURCE;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* try to free space by erasing backup */
280   1       erase$bak: PROCEDURE BOOLEAN;

281   2         if onefile then
282   2             if newfile then do;
284   3                 call move(fs,.dfcb,.tmpfcb);  /* can't diddle with open fcb */
285   3                 CALL SETTYPE(.tmpfcb,.BACKUP);
286   3                 CALL DELETE$file(.tmpfcb);
287   3                 if dcnt <> 255 then
288   3                     return true;
289   3                 end;
290   2         return false;
291   2         end erase$bak;


          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* write output buffer up to (not including)
                           ndest  (low 7 bits of ndest are 0  */
292   1       WRITEDEST: PROCEDURE;
293   2         DECLARE (I,N,save$ndest) BYTE;

294   2         n = shr(ndest,sectshf);   /* calculate number sectors to write */
295   2         if n=0 then return;       /* no need to write if we haven't filled sector*/
297   2         save$ndest = ndest;       /* save for error recovery */
298   2         ndest = 0;
299   2             DO I = 1 TO N;
300   3       retry:
                  CALL SETDMA(DBUFFADR+NDEST);
301   3                 IF DISKWRITE(.DFCB) <> 0 THEN
302   3                     if erase$bak then
303   3                         go to retry;
304   3                     else do;    /* reset buffer, let them take action (delete files) */
305   4                         if ndest <> 0 then
306   4                             call move(save$ndest-ndest, dbuffadr+ndest, dbuffadr);
307   4                         ndest = save$ndest-ndest;
308   4                         go to disk$err;
309   4                         end;
310   3                 NDEST = NDEST + SECTSIZE;
311   3                 END;
312   2         ndest = 0;
313   2         END WRITEDEST;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* put a character in output buffer */
314   1       PUTDEST: PROCEDURE(B);
315   2         DECLARE B BYTE;
316   2         IF NDEST >= BUFFLENGTH THEN CALL WRITEDEST;
318   2         DBUFF(NDEST) = B;
319   2         NDEST = NDEST + 1;
```

```
320   2         END PUTDEST;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* put a character in the xfer buffer */
321   1       PUTXFER: PROCEDURE(C);
322   2           DECLARE C BYTE;
323   2           IF XBP >= SECTSIZE THEN /* BUFFER OVERFLOW */
324   2               DO;
325   3       retry:
                    CALL SETXDMA;
326   3                   xfcbext = xfcbe;                /* save for appends */
327   3                   xfcbrec = xfcbr;
328   3                   IF DISKWRITE(.XFCB) <> 0 THEN
329   3                       if erase$bak then
330   3                           go to retry;
331   3                       else do;
          /********           call close(.xfcb);    ***  commented out whf 8/82 !!!! ********/
332   4                           go to disk$err;
333   4                           end;
334   3                   XBP = 0;
335   3                   END;
336   2           XBUFF(XBP) = C; XBP = XBP + 1;
338   2           END PUTXFER;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* empty xfer buffer and close file.
                           This routine is added to allow saving lib
                           files for future edits - DH 10/18/81 */
339   1       close$xfer: procedure;
340   2           dcl i byte;

341   2               do i = xbp to sectsize;
342   3               call putxfer(ENDFILE);
343   3               end;
344   2           call close(.xfcb);
345   2           end close$xfer;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* compare xfcb and rfcb to see if same */
346   1       compare$xfer: procedure BOOLEAN;
347   2           dcl i byte;

348   2           i = 12;
349   2               do while (i:=i-1) <> -1;
350   3               if xfcb(i) <> rfcb(i) then
351   3                   return false;
352   3               end;
353   2           return true;
354   2           end compare$xfer;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* restore xfer file extent and current
                           record, read record and set xfer pointer
```

```
                                to first ENDFILE */
355   1       append$xfer: procedure;

356   2           xfcbe = xfcbext;
357   2           call open(.xfcb);
358   2           xfcbr = xfcbrec;
359   2           call setxdma;
360   2           if diskread(.xfcb) = 0 then do;
362   3               xfcbr = xfcbrec;                /* write same record */
363   3                   do xbp = 0 to sectsize;
364   4                   if xbuff(xbp) = ENDFILE then
365   4                       return;
366   4                   end;
367   3               end;
368   2           end append$xfer;
```

```
            $ eject
            /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

                                * * *  END EDIT ROUTINE * * *

             * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

                                /* finish edit, close files, rename */
369   1     FINIS: PROCEDURE;
370   2         MOVEUP: PROCEDURE(afcb);
371   3         dcl afcb address;
                /* set second filename (new name) for rename function */
372   3         CALL MOVE(16,aFCB,aFCB+16);
373   3         END MOVEUP;

                /* * * * * * * *  WRITE OUTPUT BUFFER  * * * * * * * * * */
                /* SET UNFILLED BYTES - USED FOR ISIS-II COMPATIBILITY */
                /*  DFUB = 0 ; <<<< REMOVE FOR MP/M 2 , CP/M 3 */
374   2         DO WHILE (LOW(NDEST) AND 7FH) <> 0;
                /* COUNTS UNFILLED BYTES IN LAST RECORD */
                /* DFUB = DFUB + 1; */
375   3         CALL PUTDEST(ENDFILE);
376   3         END;
377   2         CALL WRITEDEST;

378   2         if not newfile then
379   2             call close(.sfcb);  /* close this to clean up for mp/m environs */

                /* * * * * *  CLOSE TEMPORARY DESTINATION FILE  * * * * * * */
380   2         CALL CLOSE(.DFCB);
381   2         IF DCNT = 255 THEN CALL FERR;
383   2         if sys then do;
385   3             dfcb(sy)=dfcb(sy) or 80h;
386   3             call setpassword;
387   3             call set$attribute(.dfcb);
388   3             end;

                /* * * * * *  RENAME SOURCE TO BACKUP IF ONE FILE  * * * * * * */
389   2         if onefile then do;
391   3             call moveup(.sfcb);
392   3             CALL SETTYPE(.sfcb+16,.BACKUP);   /* set new type to BAK */
393   3             CALL RENAME$FILE(.SFCB);
394   3             end;

                /* * * * * *  RENAME TEMPORARY DESTINATION FILE  * * * * * * */
395   2         CALL MOVEUP(.DFCB);
396   2         CALL SETTYPE(.DFCB+16,.DTYPE);
397   2         CALL RENAME$FILE(.DFCB);

398   2         END FINIS;
```

```
$ eject

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

                    * * *  COMMAND ROUTINES * * *

 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */


                    /* print a character if not macro expansion */
399  1     PRINTNMAC: PROCEDURE(CHAR);
400  2         DECLARE CHAR BYTE;
401  2         IF MP <> 0 THEN RETURN;
403  2         CALL PRINTC(CHAR);
404  2         END PRINTNMAC;
           /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                    /* return true if lower case character */
405  1     LOWERCASE: PROCEDURE(C) BOOLEAN;
406  2         DECLARE C BYTE;
407  2         RETURN C >= LCA AND C <= LCZ;
408  2         END LOWERCASE;
           /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                    /* translate character to upper case */
409  1     UCASE: PROCEDURE(C) BYTE;
410  2         DECLARE C BYTE;
411  2         IF LOWERCASE(C) THEN RETURN C AND 5FH;
413  2         RETURN C;
414  2         END UCASE;
           /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                    /* get password and place at fcb + 16 */
415  1     getpasswd:   proc;
416  2         dcl (i,c) byte;

417  2         call crlf;
418  2         call print(.('Password ? ','$'));
419  2     retry:
              call fill(.password,' ',8);
420  2             do i = 0 to 7;
421  3     nxtchr:
                  if (c:=ucase(conin)) >= ' ' then
422  3                 password(i)=c;
423  3                 if c = cr then
424  3                     go to exit;
425  3                 if c = CTLX then
426  3                     goto retry;
427  3                 if c = CTLH then do;
429  4                     if i<1 then
430  4                         goto retry;
```

```
431   4                 else do;
432   5                     password(i:=i-1)=' ';
433   5                     goto nxtchr;
434   5                     end;
435   4                 end;
436   3             if c = 3 then
437   3                 call reboot;
438   3             end;
439   2     exit:
              c = break$key;      /* clear raw I/O mode */
440   2         end getpasswd;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* translate to upercase if translate flag
                           is on (also translate ESC to ENDFILE) */
441   1     UTRAN: PROCEDURE(C) BYTE;
442   2         DECLARE C BYTE;
443   2         IF C = ESC THEN C = ENDFILE;
445   2         IF TRANSLATE THEN RETURN UCASE(C);
447   2         RETURN C;
448   2         END UTRAN;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* print the line number */
449   1     PRINTVALUE: PROCEDURE(V);
              /* PRINT THE LINE VALUE V */
450   2         DECLARE D BYTE,
                  ZERO BOOLEAN,
                      (K,V) ADDRESS;
451   2         K = 10000;
452   2         ZERO = FALSE;
453   2             DO WHILE K <> 0;
454   3             D = LOW(V/K); V = V MOD K;
456   3             K = K / 10;
457   3             IF ZERO OR  D <> 0 THEN
458   3                 DO; ZERO = TRUE;
460   4                 CALL PRINTC('0'+D);
461   4                 END;
                  ELSE
462   3                 CALL PRINTC(' ');
463   3             END;
464   2         END PRINTVALUE;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* print line with number V */
465   1     PRINTLINE: PROCEDURE(V);
466   2         DECLARE V ADDRESS;
467   2         IF NOT LINESET THEN RETURN;
469   2         CALL PRINTVALUE(V);
470   2         CALL PRINTC(':');
471   2         CALL PRINTC(' ');
472   2         IF INSERTING THEN
473   2             CALL PRINTC(' ');
              ELSE
```

```
474  2            CALL PRINTC('*');
475  2        END PRINTLINE;
         /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                      /* print current line (baseline) */
476  1    PRINTBASE: PROCEDURE;
477  2        CALL PRINTLINE(BASELINE);
478  2        END PRINTBASE;
         /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                      /* print current line if not in a macro */
479  1    PRINTNMBASE: PROCEDURE;
480  2        IF MP <> 0 THEN RETURN;
482  2        CALL PRINTBASE;
483  2        END PRINTNMBASE;
         /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                      /* get next character from command tail */
484  1    getcmd: proc byte;
485  2         if buff(ncmd+1) <> 0 then
486  2            return buff(ncmd := ncmd + 1);
487  2         return cr;
488  2        end getcmd;
         /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                /* read next char from command buffer */
489  1    READC: PROCEDURE BYTE;
             /* MAY BE MACRO EXPANSION */
490  2        IF MP > 0 THEN
491  2            DO;
492  3            IF BREAK$KEY THEN GO TO OVERCOUNT;
494  3            IF XP >= MP THEN
495  3                DO; /* START AGAIN */
496  4                IF MT <> 0 THEN
497  4                    DO; IF (MT:=MT-1) = 0 THEN
499  5                        GO TO OVERCOUNT;
500  5                    END;
501  4                XP = 0;
502  4                END;
503  3            RETURN UTRAN(MACRO(XP := XP + 1) - 1));
504  3            END;
505  2        IF INSERTING THEN RETURN UTRAN(READCHAR);

             /* GET COMMAND LINE */
507  2        IF READBUFF THEN
508  2            DO; READBUFF = FALSE;
510  3            IF LINESET AND COLUMN = 0 THEN
511  3                DO;
512  4                IF BACK >= MAXM THEN
513  4                    CALL PRINTLINE(0);
                 ELSE
514  4                    CALL PRINTBASE;
515  4                END;
```

```
                ELSE
516   3             CALL PRINTC('*');
517   3             CALL READCOM; CBP = 0;
519   3             CALL PRINTC(LF);
520   3             COLUMN = 0;
521   3             END;
522   2         IF (READBUFF := CBP = COMLEN ) THEN
523   2             COMBUFF(CBP) = CR;
524   2         RETURN UTRAN(COMBUFF((CBP := CBP +1) -1));
525   2         END READC;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* get upper case character from command
                           buffer or command line */
526   1       get$uc: proc;
527   2           if tail then
528   2               char = ucase(getcmd);
              else
529   2               char = ucase(readc);
530   2           end get$uc;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* parse file name
                           this routine requires a routine to get
                           the next character and put it in a byte
                           variable */
531   1       parse$fcb: proc(fcbadr) byte;
532   2           dcl fcbadr addr;
533   2           dcl afcb based fcbadr (33) byte;
534   2           dcl drive lit 'afcb(0)';
535   2           dcl (i,delimiter) byte;
536   2           dcl pflag boolean;

537   2           putc: proc;
538   3              afcb(i := i + 1) = char;
539   3              pflag = true;
540   3           end putc;

541   2           delim: proc boolean;
542   3              dcl del(*) byte data (CR,ENDFILE,' .,;=:<>_[]*?');
                                        /*   0 1        2345678901234 */
543   3                 do delimiter = 0 to last(del);
544   4                 if char = del(delimiter) then do;
546   5                    if delimiter > 12 then     /* * or ? */
547   5                       call perror(.('Cannot Edit Wildcard Filename$'));
548   5                    return (true);
549   5                    end;
550   4                 end;
551   3              return (false);
552   3           end delim;


553   2           pflag = false;
554   2           flag = true;    /* global flag set to false if invalid filename */
555   2           dot$found = false;  /* allow null extensions in 'parse$lib' */
```

```
556   2          call get$uc;
557   2          if char <> CR then
558   2              if char <> ENDFILE then do;
                              /* initialize fcb to srce fcb type & drive */
560   3                  call fill(fcbadr+12,0,21);
561   3                  call fill(fcbadr+1,' ',11);
                              /* clear leading blanks */
562   3                      do while char = ' ';
563   4                      call get$uc;
564   4                      end;
                              /* parse loop */
565   3                      do while not delim;
566   4                      i = 0;
                                  /* get name */
567   4                                do while not delim;
568   5                                if i > 8 then
569   5                                    go to err;      /* too long */
570   5                                call putc;
571   5                                call get$uc;
572   5                                end;
573   4                      if char = ':' then do;
                                  /* get drive from afcb(1) */
575   5                                if i <> 1 then
576   5                                    go to err;      /* invalid : */
577   5                                if (drive := afcb(1) - 'A' + 1) > 16 then
578   5                                    go to err;      /* invalid drive */
579   5                                afcb(1) = ' ';
580   5                                call get$uc;
581   5                                end;
582   4                      if char = '.' then do;
                                  /* get file type */
584   5                                i = 8;
585   5                                dot$found = true;         /* .ext specified (may be null)*/
586   5                                call get$uc;
587   5                                    do while not delim;
588   6                                    if i > 11 then
589   6                                        go to err;      /* too long */
590   6                                    call putc;
591   6                                    call get$uc;
592   6                                    end;
593   5                                end;
594   4              if char = ';' then do;
                          /* get password */
596   5                        call fill(fcbadr+16,' ',8); /* where fn #152 puts passwd */
597   5                        i = 15;          /* passwd is last field */
598   5                        call get$uc;
599   5                        do while not delim;
600   6                        if i > 23 then
601   6                            go to err;
602   6                        call putc;
603   6                        call get$uc;
604   6                        end;
605   5                        call move(8,fcbadr+16,.password); /* where ed wants it */
606   5                        end;
607   4                            end;                /* parse loop */
                          /* delimiter must be a comma or space */
608   3                        if delimiter > 3 then   /* not a CR,ENDFILE,SPACE,COMMA */
```

```
609   3                   go to err;
610   3               if not pflag then
611   3                   go to err;
612   3               end;

613   2       return (pflag);

614   2   err:
            call perror(.invalid);
615   2       return (flag:=false);
616   2       end parse$fcb;
        /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* set up destination FCB */
617   1   setdest: PROCEDURE;
618   2       dcl i byte;

            /* onefile = true; (initialized) */
619   2       if not tail then do;
621   3           call print(.('Enter Output file: $'));
622   3           call readcom;
623   3           cbp,readbuff = 0;
624   3           call crlf;
625   3           call crlf;
626   3           end;
627   2       if parse$fcb(.dfcb) then do;
629   3           onefile = false;
630   3           if dfcb(1) = ' ' then
631   3               call move(15,.sfcb+1,.dfcb+1);
632   3           end;
            else
633   2           CALL MOVE(16,.SFCB,.DFCB);
634   2       call move(3,.dfcb(9),.dtype);   /* save destination type */
635   2       end setdest;
        /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* set read lib file DMA address */
636   1   SETRDMA: PROCEDURE;
637   2       CALL SETDMA(.BUFF);
638   2       END SETRDMA;
        /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* read lib file routine */
639   1   READFILE: PROCEDURE BYTE;
640   2       IF RBP >= SECTSIZE THEN
641   2           DO; CALL SETRDMA;
643   3           IF DISKREAD(.RFCB) <> 0 THEN RETURN ENDFILE;
645   3           RBP = 0;
646   3           END;
647   2       RETURN UTRAN(BUFF(RBP := RBP + 1) - 1));
648   2       END READFILE;
```

```
$ eject

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

                    * * *  INITIALIZATION  * * *

 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

SETUP: PROCEDURE;

    /* * * * * * * * *  OPEN SOURCE FILE  * * * * * * * * * */

    sfcb(ex), sfcb(md), sfcb(nr) = 0;
    if has$bdos3 then do;
        call return$errors(0FEh);           /* set error mode */
        call setpassword;
        end;
    error$code = open$file (.SFCB);
    if has$bdos3 then do;                   /* extended bdos errors */
        call return$errors(0);              /* reset error mode */
        if low(error$code) = 0FFh and high(error$code) = 7 then do;
            call getpasswd;                 /* let them enter password */
            call crlf;
            call crlf;
            call setpassword;               /* set dma to password */
            error$code = open$file(.fcb);   /* reset error$code */
            end;
        if low(error$code)=0FFh and high(error$code)<>0 then
                call abort(.notavail);      /* abort anything but not found */
        end;
    dcnt=low(error$code);
    if onefile then do;
        IF ROL(FCB(RO),1) THEN
            CALL abort(.('FILE IS READ/ONLY$'));
        else IF ROL(FCB(SY),1) THEN   /* system attribute */
            do;
            if rol(FCB(us),1) then
                dcnt = 255;          /* user 0 file so create */
            else
                sys = true;
            end;
        end;

    /* * * * * *  NEW FILE IF NO SOURCE FILE  * * * * * * */

    IF DCNT = 255 THEN do;
        if not onefile then
            call abort(.not$found);
        newfile = true;
        CALL PRINT(.('NEW FILE$'));
        CALL CRLF;
        END;

    /* * * * * *  MAKE TEMPORARY DESTINATION FILE  * * * * * * */
```

```
691   2          CALL SETTYPE(.dfcb,.tempfl);
692   2          DFCB(EX)=0;
693   2          CALL MAKE$file(.DFCB);
694   2          if dcnt = 255 then
695   2              call ferr;
                 /* THE TEMP FILE IS NOW CREATED */

                 /* now create the password if any */
696   2          if protection <> 0 then do;
698   3              dfcb(ex) = protection or 1;  /* set password */
699   3              call setpassword;
700   3              call write$xfcb(.dfcb);
701   3              end;
702   2          dfcb(ex),DFCB(32) = 0;  /* NEXT RECORD IS ZERO */

                 /* * * * * * * * *  RESET BUFFER  * * * * * * * * * */

703   2          NSOURCE = BUFFLENGTH;
704   2          NDEST = 0;
705   2          BASELINE = 1;           /* START WITH LINE 1 */
706   2          END SETUP;
```

```
            $ eject

            /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

                                * * *  BUFFER MANAGEMENT * * *

             * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */


                                /* DISTANCE is the number of lines prefix
                                   to a command */
                                /* set maximum distance (OFFFFH) */
707   1     SETFF: PROCEDURE;
708   2         DISTANCE = OFFFFH;
709   2         END SETFF;
            /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                                /* return true if distance is zero */
710   1     DISTZERO: PROCEDURE BOOLEAN;
711   2         RETURN DISTANCE = 0;
712   2         END DISTZERO;
            /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                                /* set distance to zero */
713   1     ZERODIST: PROCEDURE;
714   2         DISTANCE = 0;
715   2         END ZERODIST;
            /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                                /* check for zero distance and decrement */
716   1     DISTNZERO: PROCEDURE BOOLEAN;
717   2         IF NOT DISTZERO THEN
718   2             DO; DISTANCE = DISTANCE - 1;
720   3             RETURN TRUE;
721   3             END;
722   2         RETURN FALSE;
723   2         END DISTNZERO;
            /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                                /* set memory limits of command from
                                   distance and direction */
724   1     SETLIMITS: PROC;
725   2         DCL (I,K,L,M) ADDR, (MID)LE,LOOPING) BYTE;
726   2         RELLINE = 1; /* RELATIVE LINE COUNT */
727   2         IF DIRECTION = BACKWARD THEN
728   2             DO; DISTANCE = DISTANCE+1; I = FRONT; L = 0; K = OFFFFH;
733   3             END;
                  ELSE
734   2             DO; I = BACK; L = MAXM; K = 1;
738   3             END;
```

```
739   2          LOOPING = TRUE;
740   2              DO WHILE LOOPING;
741   3                  DO WHILE (MIDDLE := I <> L) AND
                            MEMORY(M:=I+K) <> LF;
742   4                  I = M;
743   4                  END;
744   3              LOOPING = (DISTANCE := DISTANCE - 1) <> 0;
745   3              IF NOT MIDDLE THEN
746   3                  DO; LOOPING = FALSE;
748   4                  I = I - K;
749   4                  END;
750   3              ELSE do;
751   4                  RELLINE = RELLINE - 1;
752   4                  IF LOOPING THEN
753   4                      I = M;
754   4                  end;
755   3              END;

756   2          IF DIRECTION = BACKWARD THEN
757   2              DO; FIRST = I; LASTC = FRONT - 1;
760   3              END;
              ELSE
761   2              DO; FIRST = BACK + 1; LASTC = I + 1;
764   3              END;
765   2          END SETLIMITS;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* increment current position */
766   1      INCBASE: PROCEDURE;
767   2          BASELINE = BASELINE + 1;
768   2          END INCBASE;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* decrement current position */
769   1      DECBASE: PROCEDURE;
770   2          BASELINE = BASELINE - 1;
771   2          END DECBASE;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* increment limits */
772   1      INCFRONT: PROC; FRONT = FRONT + 1;
774   2          END INCFRONT;
775   1      INCBACK: PROCEDURE; BACK = BACK + 1;
777   2          END INCBACK;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* decrement limits */
778   1      DECFRONT: PROC; FRONT = FRONT - 1;
780   2          IF MEMORY(FRONT) = LF THEN
781   2              CALL DECBASE;
782   2          END DECFRONT;
783   1      DECBACK: PROC; BACK = BACK - 1;
```

```
785  2          END DECBACK;
           /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                      /* move current page in memory if move flag
                         true otherwise delete it */
786  1       MEM$MOVE: PROC(MOVEFLAG);
787  2           DECLARE (MOVEFLAG,C) BYTE;
                 /* MOVE IF MOVEFLAG IS TRUE */
788  2           IF DIRECTION = FORWARD THEN
789  2               DO WHILE BACK < LASTC; CALL INCBACK;
791  3               IF MOVEFLAG THEN
792  3                   DO;
793  4                   IF (C := MEMORY(BACK)) = LF THEN CALL INCBASE;
795  4                   MEMORY(FRONT) = C; CALL INCFRONT;
797  4                   END;
798  3               END;
                 ELSE
799  2               DO WHILE FRONT > FIRST; CALL DECFRONT;
801  3               IF MOVEFLAG THEN
802  3                   DO; MEMORY(BACK) = memory(front); CALL DECBACK;
805  4                   END;
806  3               END;
807  2           END MEM$MOVE;
           /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                      /* force a memory move */
808  1       MOVER: PROC;
809  2           CALL MEM$MOVE(TRUE);
810  2           END MOVER;
           /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                      /* reset memory limit pointers, deleting
                         characters (used by D command) */
811  1       SETPTRS: PROC;
812  2           CALL MEM$MOVE(FALSE);
813  2           END SETPTRS;
           /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                      /* set limits and force a move */
814  1       MOVELINES: PROC;
815  2           CALL SETLIMITS;
816  2           CALL MOVER;
817  2           END MOVELINES;
           /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                      /* set front to lower value deleteing
                         characters (used by S and J commands) */
818  1       setfront: proc(newfront);
819  2           dcl newfront addr;

820  2               do while front <> newfront;
821  3               call decfront;
```

```
822  3              end;
823  2          end setfront;
           /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                       /* set limits for memory move */
824  1      SETCLIMITS: PROC;
825  2          IF DIRECTION = BACKWARD THEN
826  2              DO; LASTC = BACK;
828  3              IF DISTANCE > FRONT THEN
829  3                  FIRST = 1;
                   ELSE
830  3                  FIRST = FRONT - DISTANCE;
831  3              END;
               ELSE
832  2              DO; FIRST = FRONT;
834  3              IF DISTANCE >= MAX - BACK THEN
835  3                  LASTC = MAXM;
                   ELSE
836  3                  LASTC = BACK + DISTANCE;
837  3              END;
838  2          END SETCLIMITS;
           /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                       /* read another line of input */
839  1      READLINE: PROCEDURE;
840  2          DECLARE B BYTE;
               /* READ ANOTHER LINE OF INPUT */
841  2          CTRAN: PROCEDURE(B) BYTE;
842  3              DECLARE B BYTE;
                   /* CONDITIONALLY TRANSLATE TO UPPER CASE ON INPUT */
843  3              IF UPPER THEN RETURN UTRAN(B);
845  3              RETURN B;
846  3              END CTRAN;
847  2          DO FOREVER;
848  3          IF FRONT >= BACK THEN GO TO OVERFLOW;
850  3          IF (B := CTRAN(GETSOURCE)) = ENDFILE THEN
851  3              DO; CALL ZERODIST; RETURN;
854  4              END;
855  3          MEMORY(FRONT) = B;
856  3          CALL INCFRONT;
857  3          IF B = LF THEN
858  3              DO; CALL INCBASE;
860  4              RETURN;
861  4              END;
862  3          END;
863  2          END READLINE;
           /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                       /* write one line out */
864  1      WRITELINE: PROCEDURE;
865  2          DECLARE B BYTE;
866  2              DO FOREVER;
867  3              IF BACK >= MAXM THEN /* EMPTY */
868  3                  DO; CALL ZERODIST; RETURN;
```

```
871   4                 END;
872   3               CALL INCBACK;
873   3               CALL PUTDEST(B:=MEMORY(BACK));
874   3               IF B = LF THEN
875   3                   DO; CALL INCBASE;
877   4                   RETURN;
878   4                   END;
879   3               END;
880   2           END WRITELINE;
            /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* write lines until at least half the
                           the buffer is empty */
881   1       WRHALF: PROCEDURE;
882   2           CALL SETFF;
883   2               DO WHILE DISTNZERO;
884   3               IF HMAX >= (MAXM - BACK) THEN
885   3                   CALL ZERODIST;
                    ELSE
886   3                   CALL WRITELINE;
887   3               END;
888   2           END WRHALF;
            /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* write lines determined by distance
                           called from W and E commands */
889   1       WRITEOUT: PROCEDURE;
890   2           DIRECTION = BACKWARD; FIRST = 1; LASTC = BACK;
893   2           CALL MOVER;
894   2           IF DISTZERO THEN CALL WRHALF;
                    /* DISTANCE = 0 IF CALL WRHALF */
896   2               DO WHILE DISTNZERO;
897   3               CALL WRTTELINE;
898   3               END;
899   2           IF BACK < LASTC THEN
900   2               DO; DIRECTION = FORWARD; CALL MOVER;
903   3               END;
904   2           END WRITEOUT;
            /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* clear memory buffer */
905   1       CLEARMEM: PROCEDURE;
906   2           CALL SETFF;
907   2           CALL WRITEOUT;
908   2           END CLEARMEM;
            /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* clear buffers, terminate edit */
909   1       TERMINATE: PROCEDURE;
910   2           CALL CLEARMEM;
911   2           if not newfile then
912   2               DO WHILE (CHAR := GETSOURCE) <> ENDFILE;
913   3               CALL PUTDEST(CHAR);
```

```
914   3             END;
915   2          CALL FINIS;
916   2          END TERMINATE;
```

```
            $ eject

            /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

                                * * *  COMMAND PRIMITIVES  * * *

             * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */


                                /* insert char into memory buffer */
917   1     INSERT: PROCEDURE;
918   2         IF FRONT = BACK THEN GO TO OVERFLOW;
920   2         MEMORY(FRONT) = CHAR; CALL INCFRONT;
922   2         IF CHAR = LF THEN CALL INCBASE;
924   2         END INSERT;
            /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                                /* read a character and check for endfile
                                   or CR  */
925   1     SCANNING: PROCEDURE BYTE;
926   2         RETURN NOT ((CHAR := READC) = ENDFILE OR
                              (CHAR = CR AND NOT INSERTING));
927   2         END SCANNING;
            /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                                /* read command buffer and insert characters
                                   into scratch 'til next endfile or CR for
                                   find, next, junt, or substitute commands
                                   fill at WBE and increment WBE so it
                                   addresses the next empty position of scratch */
928   1     COLLECT: PROCEDURE;

929   2         SETSCR: PROCEDURE;
930   3             SCRATCH(WBE) = CHAR;
931   3             IF (WBE := WBE + 1) >= SCRSIZE THEN GO TO OVERFLOW;
933   3             END SETSCR;

934   2             DO WHILE SCANNING;
935   3             IF CHAR = CTLL THEN
936   3                 DO; CHAR = CR; CALL SETSCR;
939   4                 CHAR = LF;
940   4                 END;
941   3             IF CHAR = 0 THEN GO TO BADCOM;
943   3             CALL SETSCR;
944   3             END;
945   2         END COLLECT;
            /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                                /* find the string in scratch starting at
                                   PA and ending at PB */
```

```
946   1       FIND: PROCEDURE(PA,PB) BYTE;
947   2           DECLARE (PA,PB) BYTE;
948   2           DECLARE J ADDRESS,
                      (K, MATCH) BYTE;
949   2           J = BACK ;
950   2           MATCH = FALSE;
951   2               DO WHILE NOT MATCH AND (MAXM > J);
952   3               LASTC,J = J + 1; /* START SCAN AT J */
953   3               K = PA ; /* ATTEMPT STRING MATCH AT K */
954   3                   DO WHILE SCRATCH(K) = MEMORY(LASTC) AND
                            NOT (MATCH := K = PB);
                        /* MATCHED ONE MORE CHARACTER */
955   4                   K = K + 1; LASTC = LASTC + 1;
957   4                   END;
958   3               END;
959   2           IF MATCH THEN /* MOVE STORAGE */
960   2               DO; LASTC = LASTC - 1; CALL MOVER;
963   3               END;
964   2           RETURN MATCH;
965   2           END FIND;
              /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* set up the search string for F,'N, and
                           S commands  */
966   1       SETFIND: PROCEDURE;
967   2           WBE = 0; CALL COLLECT; WBP = WBE;
970   2           END SETFIND;
              /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* check for found string in F and S commands */
971   1       CHKFOUND: PROCEDURE;
972   2           IF NOT FIND(0,WBP) THEN /* NO MATCH */ GO TO OVERCOUNT;
974   2           END CHKFOUND;
              /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* parse read / xfer lib FCB */
975   1       parse$lib: procedure(fcbadr) byte;
976   2           dcl fcbadr address;
977   2           dcl afcb based fcbadr (33) byte;
978   2           dcl b byte;

979   2           b = parse$fcb(fcbadr);
                  /* flag = false if invalid */
980   2           if not flag then do;
982   3               flag = '0';
983   3               goto reset;
984   3               end;
985   2           if afcb(9) = ' ' and not dot$found then
986   2               call move(3,.libfcb(9),fcbadr+9);
987   2           if afcb(1) = ' ' then
988   2               call move(8,.libfcb(1),fcbadr+1);
989   2           return b;
990   2           end parse$lib;
              /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
```

```
                        /* print relative position */
 991   1      PRINTREL: PROCEDURE;
 992   2          CALL PRINTLINE(BASELINE+RELLINE);
 993   2          END PRINTREL;
           /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

                        /* type lines command */
 994   1      TYPELINES: PROCEDURE;
 995   2          DCL I ADDR;
 996   2          DCL C BYTE;
 997   2          CALL SETLIMITS;
                  /* DISABLE THE * PROMPT */
 998   2          INSERTING = TRUE;
 999   2          IF DIRECTION = FORWARD THEN
1000   2              DO; RELLINE = 0; I = FRONT;
1003   3              END;
                  ELSE
1004   2              I = FIRST;
1005   2          IF (C := MEMORY(I-1)) = LF then  do;
1007   3              if COLUMN <> 0 THEN
1008   3                  CALL CRLF;
1009   3              end;
                  else
1010   2              relline = relline + 1;

1011   2              DO I = FIRST TO LASTC;
1012   3              IF C = LF THEN
1013   3                  DO;
1014   4                  CALL PRINTREL;
1015   4                  RELLINE = RELLINE + 1;
1016   4                  IF BREAK$KEY THEN GO TO OVERCOUNT;
1018   4                  END;
1019   3              CALL PRINTC(C:=MEMORY(I));
1020   3              END;
1021   2          END TYPELINES;
           /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

                        /* set distance to lines per page (LPP) */
1022   1      SETLPP: PROCEDURE;
1023   2          DISTANCE = LPP;
1024   2          END SETLPP;
           /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

                        /* save distance in TDIST */
1025   1      SAVEDIST: PROCEDURE;
1026   2          TDIST = DISTANCE;
1027   2          END SAVEDIST;
           /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

                        /* Restore distance from TDIST */
1028   1      RESTDIST: PROCEDURE;
```

```
1029  2          DISTANCE = TDTST;
1030  2          END RESTDIST;
                /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

                                /* page command (move n pages and print) */
1031  1        PAGE: PROCEDURE;
1032  2          DECLARE I BYTE;
1033  2          CALL SAVEDIST;
1034  2          CALL SETLPP;
1035  2          CALL MOVELINES;
1036  2          I = DIRECTION;
1037  2          DIRECTION = FORWARD;
1038  2          CALL SETLPP;
1039  2          CALL TYPELINES;
1040  2          DIRECTION = I;
1041  2          IF LASTC = MAXM OR FIRST = 1 THEN
1042  2              CALL ZERODIST;
                  ELSE
1043  2              CALL RESTDIST;
1044  2          END PAGE;
                /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

                                /* wait command (1/2 second time-out) */
1045  1        WAIT: PROCEDURE;
1046  2          DECLARE I BYTE;
1047  2              DO I = 0 TO 19;
1048  3              IF BREAK$KEY THEN GO TO RESET;
1050  3              CALL TIME(250);
1051  3              END;
1052  2          END WAIT;
                /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

                                /* set direction to forward */
1053  1        SETFORWARD: PROCEDURE;
1054  2          DIRECTION = FORWARD;
1055  2          DISTANCE = 1;
1056  2          END SETFORWARD;
                /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

                                /* append 'til buffer is at least half full */
1057  1        APPHALF: PROCEDURE;
1058  2          CALL SETFF; /* DISTANCE = OFFFFH */
1059  2              DO WHILE DISTNZERO;
1060  3              IF FRONT >= HMAX THEN
1061  3                  CALL ZERODIST;
                    ELSE
1062  3                  CALL READLINE;
1063  3              END;
1064  2          END APPHALF;
                /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

                                /* insert CR LF characters */
```

```
1065   1       INSCRLF: PROCEDURE;
                   /* INSERT CR LF CHARACTERS */
1066   2           CHAR = CR; CALL INSERT;
1068   2           CHAR = LF; CALL INSERT;
1070   2           END INSCRLF;
               /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                              /* test if invalid delete or
                                 backspace at beginning of inserting */
1071   1       ins$error$chk: procedure;
1072   2           if (tcolumn = 255) or (front = 1) then
1073   2               go to reset;
1074   2           end ins$error$chk;
```

```
$ eject

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

                        * * *  COMMAND PARSING * * *

 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */



                        /* test for upper or lower case command
                           set translate flag (used to determine
                           if following characters should be translated
                           to upper case */
1075   1        TESTCASE: PROCEDURE;
1076   2            DECLARE T BYTE;
1077   2            TRANSLATE = TRUE;
1078   2            T = LOWERCASE(CHAR);
1079   2            CHAR = UTRAN(CHAR);
1080   2            TRANSLATE = UPPER OR NOT T;
1081   2            END TESTCASE;
               /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

                        /* set translate to false and read next
                           character */
1082   1        READCTRAN: PROCEDURE;
1083   2            TRANSLATE = FALSE;
1084   2            CHAR = READC;
1085   2            CALL TESTCASE;
1086   2            END READCTRAN;
               /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

                        /* return true if command is only character
                           not in macro or combination on a line */
1087   1        SINGLECOM: PROCEDURE(C) BOOLEAN;
1088   2            DECLARE C BYTE;
1089   2            RETURN CHAR = C AND COMLEN = 1 AND MP = 0;
1090   2            END SINGLECOM;
               /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

                        /* return true if command is only character
                           not in macro or combination on a line, and
                           the operator has responded with a 'Y' to a
                           Y/N request */
1091   1        SINGLERCOM: PROCEDURE(C) BOOLEAN;
1092   2            DECLARE (C,I) BYTE;
1093   2            IF SINGLECOM(C) THEN
1094   2                DO forever;
1095   3                CALL CRLF; CALL PRINTCHAR(C);
1097   3                CALL MON1(9,.('-(Y/N)',WHAT,'$'));
1098   3                I = UCASE(READCHAR); CALL CRLF;
```

```
1100  3             IF i = 'N' THEN GO TO START;
1102  3             if i = 'Y' then
1103  3                 RETURN TRUE;
1104  3             END;
1105  2         RETURN FALSE;
1106  2         END SINGLERCOM;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* return true if char is a digit */
1107  1     DIGIT: PROCEDURE BOOLEAN;
1108  2         RETURN (I := CHAR - '0') <= 9;
1109  2         END DIGIT;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* return with distance = number char =
                           next command */
1110  1     NUMBER: PROCEDURE;
1111  2         DISTANCE = 0;
1112  2             DO WHILE DIGIT;
1113  3             DISTANCE = SHL(DISTANCE,3) +
                               SHL(DISTANCE,1) + I;
1114  3             CALL READCTRAN;
1115  3             END;
1116  2         END NUMBER;
          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


                        /* set distance to distance relative to
                           the current line */
1117  1     RELDISTANCE: PROCEDURE;
1118  2         IF DISTANCE > BASELINE THEN
1119  2             DO; DIRECTION = FORWARD;
1121  3             DISTANCE = DISTANCE - BASELINE;
1122  3             END;
              ELSE
1123  2             DO; DIRECTION = BACKWARD;
1125  3             DISTANCE = BASELINE - DISTANCE;
1126  3             END;
1127  2         END RELDISTANCE;
```

```
            $ eject

            /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

                                * * *  MAIN PROGRAM * * *

             * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

1128   1    plmstart:               /* entry of MP/M-86 Interface */

             /* INITIALIZE THE SYSTEM */

                ver = version;
1129   1        if low(ver) >= cpm3 then
1130   1            has$bdos3 = true;       /* handles passwords & xfcbs */

                /* * * * * * * SET UP MEMORY BUFFER * * * * * * * * * * */

                /* I/O BUFFER REGION IS 1/8 AVAILABLE MEMORY */
1131   1        NBUF = SHR(MAX := MAXB - .MEMORY,SECTSHF+3) - 1;
                /* NBUF IS NUMBER OF BUFFERS - 1 */
1132   1        BUFFLENGTH = SHL(DOUBLE(NBUF+1),SECTSHF+1);
                /* NOW SET MAX AS REMAINDER OF FREE MEMORY */
1133   1        IF BUFFLENGTH + 1024 > MAX THEN
1134   1            DO; CALL perror(.('Insufficient memory$'));
1136   2            CALL BOOT;
1137   2            END;
                /* REMOVE BUFFER SPACE AND 00 AT END OF MEMORY VECTOR */
1138   1        MAX = MAX - BUFFLENGTH - 1;
                /* RESET BUFFER LENGTH FOR I AND O */
1139   1        BUFFLENGTH = SHR(BUFFLENGTH,1);
1140   1        SBUFFADR = MAXB - BUFFLENGTH;
1141   1        DBUFFADR = SBUFFADR - BUFFLENGTH;
1142   1        MEMORY(MAX) = 0; /* STOPS MATCH AT END OF BUFFER */
1143   1        MAXM = MAX - 1;
1144   1        HMAX = SHR(MAXM,1);

                /* * * * * *  SET UP SOURCE & DESTINATION FILES  * * * * * */

1145   1        if fcb(1)=' ' then do;
1147   2            call print(.('Enter Input  file: $'));
1148   2            call readcom;
1149   2            call crlf;
1150   2            tail = false;
1151   2            end;
1152   1        if not parse$fcb(.SFCB) then     /* parse source fcb */
1153   1            call reboot;

1154   1        if has$bdos3 then do;
1156   2            call read$xfcb(.sfcb);       /* get prot from source */
1157   2            protection = sfcb(ex);       /* password protection mode */
1158   2            sfcb(ex) = 0;
1159   2            if high(ver) = 0 then        /* CP/M-80 */
1160   2                if (lpp:=mon2(49,.pb)) = 0 then
```

```
1161  2                     lpp = 23;           /* get lines per page from SCB */
1162  2                 end;
1163  1             call setdest;               /* parse destination file */
1164  1             tail = false;               /* parse$fcb from ED command */

                    /* SOURCE AND DESTINATION DISKS SET */

                    /* IF SOURCE AND DESTINATION DISKS DIFFER, CHECK FOR
                    AN EXISTING SOURCE FILE ON THE DESTINATION DISK - THERE
                    COULD BE A FATAL ERROR CONDITION WHICH COULD DESTROY A
                    FILE IF THE USER HAPPENED TO BE ADDRESSING THE WRONG
                    DISK */
1165  1             IF (SDISK <> DDISK) or not onefile THEN
1166  1                 IF mon2(15,.dfcb) <> 255 THEN   /* try to open */
                        /* SOURCE FILE PRESENT ON DEST DISK */
1167  1                     CALL ABORT(.('Output File Exists, Erase It$'));


1168  1           RESTART:
                    CALL SETUP;
1169  1             MEMORY(0) = LF;
1170  1             FRONT = 1; BACK = MAXM;
1172  1             COLUMN = 0;
1173  1             GO TO START;

1174  1           OVERCOUNT: FLAG = POUND; GO TO RESET;

1176  1           BADCOM: FLAG = WHAT; GO TO RESET;

1178  1           OVERFLOW: /* ARRIVE HERE ON OVERFLOW CONDITION (I,F,S COMMAND) */
                    FLAG = '>'; go to reset;

1180  1           disk$err:
                    flag = 'F';
1181  1             err$msg = .diskfull;
1182  1             go to reset;

1183  1           dir$err:
                    flag = 'F';
1184  1             err$msg = .dirfull;

1185  1           RESET: /* ARRIVE HERE ON ERROR CONDITION */
                    PRINTSUPPRESS = FALSE;
1186  1             CALL PRINT(.(tab,'BREAK "$'));
1187  1             CALL PRINTC(FLAG);
1188  1             CALL PRINTM(.('" AT $'));
1189  1             if char = CR or char = LF then
1190  1                 call printm(.('END OF LINE$'));
                    else
1191  1                 CALL PRINTC(CHAR);
1192  1             if err$msg <> 0 then do;
1194  2                 call perror(err$msg);
1195  2                 err$msg = 0;
1196  2                 end;
1197  1             CALL CRLF;
```

```
1198   1      START:
                  READBUFF = TRUE;
1199   1          MP = 0;
```

```
            $ eject

1200   1        DO FOREVER; /* OR UNTIL THE POWER IS TURNED OFF */

                /* ***************************************************************
                SIMPLE COMMANDS (CANNOT BE PRECEDED BY DIRECTION/DISTANCE):
                          E       END THE EDIT NORMALLY
                          H       MOVE TO HEAD OF EDITED FILE
                          I       INSERT CHARACTERS
                          O       RETURN TO THE ORIGINAL FILE
                          R       READ FROM LIBRARY FILE
                          Q       QUIT EDIT WITHOUT CHANGES TO ORIGINAL FILE
                   *************************************************************** */


1201   2        INSERTING = FALSE;
1202   2        CALL READCTRAN;
1203   2        FLAG = 'E';
1204   2        MT = CBP; /* SAVE STARTING ADDRESS FOR <CR> COMMAND */
1205   2        IF SINGLECOM('E') THEN
1206   2            DO; CALL TERMINATE;
1208   3            CALL REBOOT;
1209   3            END;

1210   2        ELSE IF SINGLECOM('H') THEN /* GO TO TOP */
1211   2            DO; CALL TERMINATE;
1213   3            newfile = false;
1214   3            if onefile then do;
                        /* PING - PONG DISKS */
1216   4                CHAR  = DDISK;
1217   4                DDISK = SDISK;
1218   4                SDISK = CHAR;
1219   4                end;
1220   3            else do;
1221   4                call settype(.dfcb,.dtype);
1222   4                call move (16,.dfcb,.sfcb); /* source = destination */
1223   4                onefile = true;
1224   4                end;
1225   3            GO TO RESTART;
1226   3            END;

1227   2        ELSE IF CHAR = 'I' THEN /* INSERT CHARACTERS */
1228   2            DO;
1229   3            IF (INSERTING := (CBP = COMLEN) AND (MP = 0)) THEN do;
1231   4                tcolumn = 255;          /* tested in ins$error$chk routine */
1232   4                distance = 0;
1233   4                direction = backward;
1234   4                if memory(front-1) = LF then
1235   4                    call printbase;
                        else
1236   4                    call typelines;
1237   4                end;
1238   3            DO WHILE SCANNING;
1239   4                DO WHILE CHAR <> 0;
1240   5                IF CHAR=CTLU OR CHAR=CTLX OR CHAR=CTLR THEN
                            /* LINE DELETE OR RETYPE */
```

```
1241  5                 DO;
                        /* ELIMINATE OR REPEAT THE LINE */
1242  6                 IF CHAR = CTLR THEN
1243  6                     DO; CALL CRLF;
1245  7                     CALL TYPELINES;
1246  7                     END;
                        ELSE
                            /* LINE DELETE */
1247  6                     DO; CALL SETLIMITS; CALL SETPTRS;
1250  7                     IF CHAR = CTLU THEN
1251  7                         DO; CALL CRLF; CALL PRINTNMBASE;
1254  8                         END;
                            ELSE
                                /* MUST BE CTLX */
1255  7                         DO WHILE COLUMN > SCOLUMN;
1256  8                         CALL BACKSPACE;
1257  8                         END;
1258  7                     END;
1259  6                 END;
1260  5               ELSE IF CHAR = CTLH THEN
1261  5                 DO;
1262  6                 call ins$error$chk;
1263  6                 IF (TCOLUMN := COLUMN) > 0 THEN
1264  6                     CALL PRINTNMACC' '); /* RESTORE AFT BACKSP */
1265  6                 call decfront;
1266  6                 if tcolumn > scolumn then
1267  6                     DO; /* CHARACTER CAN BE ELIMINATED */
1268  7                     PRINTSUPPRESS = TRUE;
                            /* BACKSPACE CHARACTER ACCEPTED */
1269  7                     COLUMN = 0;
1270  7                     CALL TYPELINES;
1271  7                     PRINTSUPPRESS = FALSE;
                            /* COLUMN POSITION NOW RESET */
1272  7                     IF (QCOLUMN := COLUMN) < SCOLUMN THEN
1273  7                         QCOLUMN = SCOLUMN;
1274  7                     COLUMN = TCOLUMN; /* ORIGINAL VALUE */
1275  7                         DO WHILE COLUMN > QCOLUMN;
1276  8                         CALL BACKSPACE;
1277  8                         END;
1278  7                     END;
                        else
1279  6                     do;
1280  7                     if memory(front-1) = CR then
1281  7                         call decfront;
1282  7                     call crlf;
1283  7                     call typelines;
1284  7                     end;
1285  6                 CHAR = 0;
1286  6                 END;
1287  5               ELSE IF CHAR = RUBOUT THEN
1288  5                 DO; call ins$error$chk;
1290  6                 CALL DECFRONT; CALL PRINTC(CHAR:=MEMORY(FRONT));
1292  6                 CHAR = 0;
1293  6                 END;
1294  5               else if char = LF and memory(front-1) <> CR then
1295  5                 do;
1296  6                 call printc(CR);
```

```
1297   6                    call inscrlf;
1298   6                    end;
                        ELSE
                            /* NOT A SPECIAL CASE */
1299   5                    DO;
1300   6                    IF NOT GRAPHIC(CHAR) THEN
1301   6                        DO;
1302   7                        CALL PRINTNMAC('^');
1303   7                        CALL PRINTNMAC(CHAR + '@');
1304   7                        end;
                                /* COLUMN COUNT GOES UP IF GRAPHIC */
                                /* COMPUTE OUTPUT COLUMN POSITION */
1305   6                    if char = CTLL and not inserting then
1306   6                        call inscrlf;
1307   6                    else do;
1308   7                        IF MP = 0 THEN
1309   7                            DO;
1310   8                            IF CHAR >= ' ' THEN
1311   8                                COLUMN = COLUMN + 1;
1312   8                            ELSE IF CHAR = TAB THEN
1313   8                                COLUMN = COLUMN + (8 - (COLUMN AND 1118));
                                    END;
1315   7                        CALL INSERT;
1316   7                        END;
1317   6                    end;
1318   5                IF CHAR = LF THEN CALL PRINTNMBASE;
1320   5                IF CHAR = CR THEN
1321   5                    CALL PRINTNMAC(CHAR:=LF);
                        ELSE
1322   5                    CHAR = 0;
1323   5                tcolumn = 0;
1324   5                END; /* of while char <> 0 */
1325   4            END; /* of while scanning */
1326   3            IF CHAR <> ENDFILE THEN do; /* MUST HAVE STOPPED ON CR */
1328   4                CALL INSCRLF;
1329   4                column = 0;
1330   4                end;
1331   3            IF INSERTING AND LINESET THEN CALL CRLF;
1333   3            END;


1334   2        ELSE IF SINGLERCOM('D') THEN /* FORGET THIS EDIT */
1335   2            do;
1336   3            call close(.sfcb);
1337   3            GO TO RESTART;
1338   3            end;


1339   2        ELSE IF CHAR = 'R' THEN
1340   2            DO; DECLARE I BYTE;
                    /* READ FROM LIB FILE */
1342   3            CALL SETRDMA;
1343   3            IF (FLAG := parse$lib(.rfcb)) THEN
1344   3                reading = false;
1345   3            if not reading then do;
1347   4                if not flag then
                            /* READ FROM XFER FILE */
```

```
1348  4                 CALL MOVE(12,.XFCB,.RFCB);
1349  4             RFCB(12), RFCB(32) = 0; /* zero extent, next record */
1350  4             rbp = sectsize;
1351  4             CALL open(.RFCB);
1352  4             reading = true;
1353  4             end;

1354  3             DO WHILE (CHAR := READFILE) <> ENDFILE;
1355  4             CALL INSERT;
1356  4             END;
1357  3         reading = false;
1358  3         call close (.rfcb);
1359  3         END;


1360  2     ELSE IF SINGLERCOM('Q') THEN
1361  2         DO;
1362  3         CALL DELETE$file(.DFCB);
1363  3         if newfile or not onefile then do;
1365  4             call settype(.dfcb,.dtype);
1366  4             call delete$file(.dfcb);
1367  4             end;
1368  3         CALL REBOOT;
1369  3         END;


              ELSE
                  /* MAY BE A COMMAND WHICH HAS AN OPTIONAL DIRECTION AND DISTANCE */
1370  2             DO; /* SCAN A SIGNED INTEGER VALUE (IF ANY) */
1371  3             DCL I BYTE;

1372  3             CALL SETFORWARD;

1373  3             IF CHAR = '-' THEN
1374  3                 DO; CALL READCTRAN; DIRECTION = BACKWARD;
1377  4                 END;

1378  3             IF CHAR = POUND THEN
1379  3                 DO; CALL SETFF; CALL READCTRAN;
1382  4                 END;

1383  3             ELSE IF DIGIT THEN
1384  3                 DO; CALL NUMBER;
                      /* MAY BE ABSOLUTE LINE REFERENCE */
1386  4                 IF CHAR = ':' THEN
1387  4                     DO; CHAR = 'L';
1389  5                     CALL RELDISTANCE;
1390  5                     END;
1391  4                 END;

1392  3             ELSE IF CHAR = ':' THEN /* LEADING COLON */
1393  3                 DO; CALL READCTRAN; /* CLEAR THE COLON */
1395  4                 CALL NUMBER;
1396  4                 CALL RELDISTANCE;
1397  4                 IF DIRECTION = FORWARD THEN
1398  4                     DISTANCE = DISTANCE + 1;
1399  4                 END;
```

```
$ eject

            IF DISTZERO THEN
1401  3         DIRECTION = BACKWARD;
            /* DIRECTION AND DISTANCE ARE NOW SET */


        /* ***************************************************************
        MAY BE A COMMAND WHICH HAS DIRECTION AND DISTANCE SPECIFIED:
              B       BEGINNING/BOTTOM OF BUFFER
              C       MOVE CHARACTER POSITIONS
              D       DELETE CHARACTERS
              K       KILL LINES
              L       MOVE LINE POSITION
              P       PAGE UP OR DOWN (LPP LINES AND PRINT)
              T       TYPE LINES
              U       UPPER CASE TRANSLATE
              V       VERIFY LINE NUMBERS
             <CR>     MOVE UP OR DOWN LINES AND PRINT LINE
           *************************************************************** */


1402  3          IF CHAR = 'B' THEN
1403  3              DO; DIRECTION = 1 - DIRECTION;
1405  4              FIRST = 1; LASTC = MAXH; CALL MOVER;
1408  4              END;

1409  3          ELSE IF CHAR = 'C' THEN
1410  3              DO; CALL SETCLIMITS; CALL MOVER;
1413  4              END;

1414  3          ELSE IF CHAR = 'D' THEN
1415  3              DO; CALL SETCLIMITS;
1417  4              CALL SETPTRS; /* SETS BACK/FRONT */
1418  4              END;

1419  3          ELSE IF CHAR = 'K' THEN
1420  3              DO; CALL SETLIMITS;
1422  4              CALL SETPTRS;
1423  4              END;

1424  3          ELSE IF CHAR = 'L' THEN
1425  3              CALL MOVELINES;

1426  3          ELSE IF CHAR = 'P' THEN /* PAGE MODE PRINT */
1427  3              DO;
1428  4              IF DISTZERO THEN
1429  4                  DO; DIRECTION = FORWARD;
1431  5                  CALL SETLPP; CALL TYPELINES;
1433  5                  END;
                    ELSE
1434  4                  DO WHILE DISTNZERO; CALL PAGE;
```

```
1436  5                 CALL WAIT;
1437  5                 END;
1438  4             END;


1439  3         ELSE IF CHAR = 'T' THEN
1440  3             CALL TYPELINES;


1441  3         ELSE IF CHAR = 'U' THEN
1442  3             UPPER = DIRECTION = FORWARD;


1443  3         ELSE IF CHAR = 'V' THEN
1444  3             DO; /* OV DISPLAYS BUFFER STATE */
1445  4             IF DISTZERO THEN
1446  4                 DO; CALL PRINTVALUE(BACK-FRONT);
1448  5                 CALL PRINTC('/');
1449  5                 CALL PRINTVALUE(MAXM);
1450  5                 CALL CRLF;
1451  5                 END;
1452  4             ELSE if (LINESET := DIRECTION = FORWARD) then
1453  4                 scolumn = 8;
                      else
1454  4                 scolumn = 0;
1455  4             END;


1456  3         ELSE IF CHAR = CR THEN /* MAY BE MOVE/TYPE COMMAND */
1457  3             DO;
1458  4             IF MI = 1 AND MP = 0 THEN /* FIRST COMMAND */
1459  4                 DO; CALL MOVELINES; CALL SETFORWARD; CALL TYPELINES;
1463  5                 END;
1464  4             END;
```

```
$ eject

1465  3          ELSE IF DIRECTION = FORWARD OR DISTZERO THEN
1466  3              DO;

                /* ************************************************************
                COMMANDS WHICH ALLOW ONLY A PRECEDING NUMBER:
                        A       APPEND LINES
                        F       FIND NTH OCCURRENCE
                        M       APPLY MACRO
                        N       SAME AS F WITH AUTOSCAN THROUGH FILE
                        S       PERFORM N SUBSTITUTIONS
                        W       WRITE LINES TO OUTPUT FILE
                        X       TRANSFER (XFER) LINES TO TEMP FILE
                        Z       SLEEP
                   ************************************************************ */


1467  4          IF CHAR = 'A' THEN
1468  4              DO; DIRECTION = FORWARD;
1470  5              FIRST = FRONT; LASTC = MAXM; CALL MOVER;
                       /* ALL STORAGE FORWARD */
1473  5              IF DISTZERO THEN CALL APPHALF;
                       /* DISTANCE = 0 IF APPHALF CALLED */
1475  5                  DO WHILE DISTNZERO;
1476  6                  CALL READLINE;
1477  6                  END;
1478  5              DIRECTION = BACKWARD; CALL MOVER;
                       /* POINTERS REPOSITIONED */
1480  5              END;


1481  4          ELSE IF CHAR = 'F' THEN
1482  4              DO; CALL SETFIND; /* SEARCH STRING SCANNED
                       AND SETUP BETWEEN 0 AND WBP-1 IN SCRATCH */
1484  5                  DO WHILE DISTNZERO; CALL CHKFOUND;
1486  6                  END;
1487  5              END;


1488  4          ELSE IF CHAR = 'J' THEN /* JUXTAPOSITION OPERATION */
1489  4              DO; DECLARE T ADDRESS;
1491  5              CALL SETFIND; CALL COLLECT;
1493  5              WBJ = WBE; CALL COLLECT;
                       /* SEARCH FOR STRING 0 - WBP-1, INSERT STRING WBP TO WBJ-1,
                       AND THEN DELETE UP TO STRING WBJ TO WBE-1 */
1495  5              DO WHILE DISTNZERO; CALL CHKFOUND;
1497  6              /* INSERT STRING */ MI = WBP - 1;
1498  6                  DO WHILE (MI := MI + 1) < WBJ;
1499  7                  CHAR = SCRATCH(MI); CALL INSERT;
1501  7                  END;
1502  6              T = FRONT; /* SAVE POSITION FOR DELETE */
1503  6              IF NOT FIND(WBJ,WBE) THEN GO TO OVERCOUNT;
                       /* STRING FOUND, SO MOVE IT BACK */
1505  6              FIRST = FRONT - (WBE - WBJ);
1506  6              DIRECTION = BACKWARD; CALL MOVER;
```

```
                    /* NOW REMOVE THE INTERMEDIATE STRING */
1508   6            call setfront(t);
1509   6            END;
1510   5          END;

1511   4          ELSE IF CHAR = 'M' AND MP = 0 THEN /* MACRO DEFINITION */
1512   4            DO; XP = 255;
1514   5            IF DISTANCE = 1 THEN CALL ZERODIST;
1516   5              DO WHILE (MACRO(XP := XP + 1) := READC) <> CR;
1517   6              END;
1518   5            MP = XP; XP = 0; MI = DISTANCE;
1521   5            END;

1522   4          ELSE IF CHAR = 'N' THEN
1523   4            DO; /* SEARCH FOR STRING WITH AUTOSCAN */
1524   5            CALL SETFIND; /* SEARCH STRING SCANNED */
1525   5              DO WHILE DISTNZERO;
                      /* FIND ANOTHER OCCURRENCE OF STRING */
1526   6                DO WHILE NOT FIND(0,WBP); /* NOT IN BUFFER */
1527   7                IF BREAK$KEY THEN GO TO RESET;
1529   7                CALL SAVEDIST; CALL CLEARMEM;
                         /* MEMORY BUFFER WRITTEN */
1531   7                CALL APPHALF;
1532   7                DIRECTION = BACKWARD; FIRST = 1; CALL MOVER;
1535   7                CALL RESTDIST; DIRECTION = FORWARD;
                         /* MAY BE END OF FILE */
1537   7                IF BACK >= MAXM THEN GO TO OVERCOUNT;
1539   7                END;
1540   6              END;
1541   5            END;

1542   4          ELSE IF CHAR = 'S' THEN /* SUBSTITUTE COMMAND */
1543   4            DO; CALL SETFIND;
1545   5            CALL COLLECT;
                    /* FIND STRING FROM 0 TO WBP-1, SUBSTITUTE STRING
                    BETWEEN WBP AND WBE-1 IN SCRATCH */
1546   5              DO WHILE DISTNZERO;
1547   6              CALL CHKFOUND;
                      /* FRONT AND BACK NOW POSITIONED AT FOUND
                      STRING - REPLACE IT */
1548   6              call setfront(FRONT - (MI := WBP)); /* BACKED UP */
1549   6                DO WHILE MI < WBE;
1550   7                CHAR = SCRATCH(MI);
1551   7                MI = MI + 1; CALL INSERT;
1553   7                END;
1554   6              END;
1555   5            END;

1556   4          ELSE IF CHAR = 'W' THEN
1557   4            CALL WRITEOUT;

1558   4          ELSE IF CHAR = 'X' THEN /* TRANSFER LINES */
```

```
1559   4                DO;
1560   5                flag = parse$lib(.rfcb);
1561   5                xbp = 0;
1562   5                IF DISTZERO THEN
1563   5                    DO;             /* delete the file */
1564   6                    xferon  = false;
1565   6                    CALL DELETE(.rfcb);
1566   6                    if dcnt = 255 then
1567   6                        call perror(.not$found);
1568   6                    END;
                        ELSE
1569   5                    do;             /* transfer lines */
1570   6                    declare i address;

1571   6                    if xferon and compare$xfer then
1572   6                        call append$xfer;
                            else
1573   6                        DO;
1574   7                        XFERON = TRUE;
1575   7                        call move(12,.rfcb,.xfcb);
1576   7                        xfcbext, xfcbrec, xfcbe, xfcbr = 0;
1577   7                        CALL MAKE$file(.XFCB);
1578   7                        IF DCNT = 255 THEN
1579   7                            goto dir$err;
1580   7                        END;
1581   6                    CALL SETLIMITS;
1582   6                        DO I = FIRST TO LASTC;
1583   7                        CALL PUTXFER(MEMORY(I));
1584   7                        END;
1585   6                    call close$xfer;
1586   6                    END;
1587   5                END;


1588   4            ELSE IF CHAR = 'Z' THEN /* SLEEP */
1589   4                DO;
1590   5                IF DISTZERO THEN
1591   5                    DO; IF READCHAR = ENDFILE THEN GO TO RESET;
1594   6                    END;
1595   5                    DO WHILE DISTNZERO; CALL WAIT;
1597   6                    END;
1598   5                END;
1599   4            ELSE IF CHAR <> 0 THEN /* NOT BREAK LEFT OVER FROM STOP */
                     /* DIRECTION FORWARD, BUT NOT ONE OF THE ABOVE */
1600   4            GO TO BADCOM;


                     END;
                     ELSE /* DIRECTION NOT FORWARD */
1602   3                 GO TO BADCOM;
1603   3                 END;
1604   2            END;
1605   1       END;
```

CROSS-REFERENCE LISTING

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES												
101	0004H	2	A.	WORD PARAMETER AUTOMATIC					102	103					
105	0004H	2	A.	WORD PARAMETER AUTOMATIC					106	108					
207	0004H	2	A.	WORD PARAMETER AUTOMATIC					208	209					
116	0004H	2	A.	WORD PARAMETER AUTOMATIC					117	118					
178	0000H	1	A.	BYTE BASED(S)				180							
244	0004H	2	A.	WORD PARAMETER AUTOMATIC					245	246					
110	0004H	2	A.	WORD PARAMETER AUTOMATIC					111	113					
168	0004H	2	A.	WORD PARAMETER AUTOMATIC					169	170					
238	0000H	1	A.	BYTE BASED(S)				240							
207	0C2EH	16	ABORT.	PROCEDURE STACK=0032H					214	669	675	686	1167		
24			ADDR	LITERALLY	40	177	238		532	725	819	995			
244	0006H	2	AFCB	WORD PARAMETER AUTOMATIC					245	246					
977	0000H	33	AFCB	BYTE BASED(FCBADR) ARRAY(33)						985	987				
220	0004H	2	AFCB	WORD PARAMETER AUTOMATIC					221	223					
370	0004H	2	AFCB	WORD PARAMETER AUTOMATIC					371	372					
225	0004H	2	AFCB	WORD PARAMETER AUTOMATIC					226	227	229				
533	0000H	33	AFCB	BYTE BASED(FCBADR) ARRAY(33)						538	577	579			
231	0004H	2	AFCB	WORD PARAMETER AUTOMATIC					232	233	235				
355	0F10H	77	APPENDXFER	PROCEDURE STACK=0012H					1572						
1057	1C63H	34	APPHALF.	PROCEDURE STACK=0046H					1474	1531					
978	0309H	1	B.	BYTE	979	989									
841	0004H	1	B.	BYTE PARAMETER AUTOMATIC					842	844	845				
178	0000H	1	B.	BYTE BASED(D)				180							
865	0306H	1	B.	BYTE	873	874									
840	0305H	1	B.	BYTE	850	855	857								
314	0004H	1	B.	BYTE PARAMETER AUTOMATIC					315	318					
271	02F4H	1	B.	BYTE	276	278									
40	001EH	2	BACK	WORD	512	735	762	776	784	789	793	803	827	834	836	848
				867	873	884	892	899	918	949	1171	1447	1537			
65	0991H	37	BACKSPACE.	PROCEDURE STACK=0014H					1256	1276					
30	0177H	3	BACKUP	BYTE ARRAY(3) INITIAL					285	392					
36			BACKWARD	LITERALLY	727	756	825		890	1124	1233	1376	1401	1478	1506	1532
38	0148H		BADCOM	LABEL	942	1176	1600	1602								
35	0012H	2	BASELINE	WORD	477	705	767	770	992	1118	1121	1125				
24			BOOLEAN.	LITERALLY	29	82	157		280	346	405	450	536	541	710	716
				1087	1091	1107										
20	0929H	14	BOOT	PROCEDURE STACK=0008H					202	1136					
157	0842H	39	BREAKKEY	PROCEDURE BYTE STACK=0008H						439	492	1016	1048	1527	
19	0000H	128	BUFF	BYTE ARRAY(128) EXTERNAL(7)						485	486	637	647		
28	0006H	2	BUFFLENGTH	WORD	274	316	703	1132	1133	1138	1139	1140	1141		
1087	0004H	1	C.	BYTE PARAMETER AUTOMATIC					1088	1089					
416	02FBH	1	C.	BYTE	421	422	423	425	427	436	439				
441	0004H	1	C.	BYTE PARAMETER AUTOMATIC					442	443	444	446	447		
1091	0004H	1	C.	BYTE PARAMETER AUTOMATIC					1092	1093	1096				
787	0304H	1	C.	BYTE	793	795									
321	0004H	1	C.	BYTE PARAMETER AUTOMATIC					322	336					
405	0004H	1	C.	BYTE PARAMETER AUTOMATIC					406	407					
996	030AH	1	C.	BYTE	1005	1012	1019								
82	0004H	1	C.	BYTE PARAMETER AUTOMATIC					83	84	86				

```
237  0004H     1  C. . . . . .   BYTE PARAMETER AUTOMATIC          238  239
176  0008H     1  C. . . . . .   BYTE PARAMETER AUTOMATIC          177  179
409  0004H     1  C. . . . . .   BYTE PARAMETER AUTOMATIC          410  411  412  413
 88  0004H     1  C. . . . . .   BYTE PARAMETER AUTOMATIC           89   90   93   95
 34  0201H     1  CBP. . . . . .  BYTE INITIAL            518  522  523  524  623 1204 1229
 40  02EFH     1  CHAR . . . . .  BYTE          528  529  538  544  557  558  562  573  582  594  912  913
                                  920  922  926  930  935  937  939  941 1066 1068 1078 1079 1084 1089
                                 1108 1189 1191 1216 1218 1227 1239 1240 1242 1250 1260 1285 1287 1291
                                 1292 1294 1300 1303 1305 1310 1312 1318 1320 1321 1322 1326 1339 1354
                                 1373 1378 1386 1388 1392 1402 1409 1414 1419 1424 1426 1439 1441 1443
                                 1456 1467 1481 1488 1499 1511 1522 1542 1550 1556 1558 1588 1599
 73  0004H     1  CHAR . . . . .  BYTE PARAMETER AUTOMATIC           74   75   76   77   79
399  0004H     1  CHAR . . . . .  BYTE PARAMETER AUTOMATIC          400  403
 57  0004H     1  CHAR . . . . .  BYTE PARAMETER AUTOMATIC           58   59   61   63
 51  0004H     1  CHAR . . . . .  BYTE PARAMETER AUTOMATIC           52   55
971  1AC1H    24  CHKFOUND . . . .  PROCEDURE STACK=001CH         1485 1496 1547
 28              CK . . . . . .  LITERALLY
905  1959H    11  CLEARMEM . . . .  PROCEDURE STACK=0032H          910 1530
133  0ADBH    19  CLOSE. . . . . .  PROCEDURE STACK=000AH          213  344  379  380 1336 1358
339  0EC1H    37  CLOSEXFER. . . .  PROCEDURE STACK=0022H         1585
928  19DEH    47  COLLECT. . . . .  PROCEDURE STACK=0038H          968 1492 1494 1545
 32  017AH     1  COLUMN . . . . .  BYTE INITIAL          60   62   66   71   75  510  520 1007 1172 1255
                                 1263 1269 1272 1274 1275 1311 1313 1329
 34  0181H   128  COMBUFF. . . . .  BYTE ARRAY(128)      523  524
 34  0180H     1  COMLEN . . . . .  BYTE        522 1089 1229
346  0EE6H    42  COMPAREXFER. . .  PROCEDURE BYTE STACK=0002H        1571
 36              COMSIZE. . . . .  LITERALLY         37   40
 48  0946H    15  CONIN. . . . . .  PROCEDURE BYTE STACK=0008H          421
  2              CPM3 . . . . . .  LITERALLY       1129
 25              CR . . . . . .  LITERALLY          86   98  423  487  523  542  557  926  937 1066 1189
                                 1280 1294 1296 1320 1456 1516
 97  0A51H    17  CRLF . . . . . .  PROCEDURE STACK=0020H          107  114  417  624  625  663  664  689
                                 1008 1095 1099 1149 1197 1244 1252 1282 1332 1450
165  0B69H    15  CSELECT. . . . .  PROCEDURE BYTE STACK=0008H
 24              CTLH . . . . . .  LITERALLY          68   70  427 1260
 24              CTLL . . . . . .  LITERALLY        935 1305
 24              CTLR . . . . . .  LITERALLY       1240 1242
 24              CTLU . . . . . .  LITERALLY       1240 1250
 24              CTLX . . . . . .  LITERALLY        425 1240
841  18A9H    25  CTRAN. . . . . .  PROCEDURE BYTE STACK=0018H          850
 25              CTRLY. . . . . .  LITERALLY        160
450  02FCH     1  D. . . . . .   BYTE         454  457  460
176  0004H     2  D. . . . . .   WORD PARAMETER AUTOMATIC          177  178  180  182
 23  0000H     4  DATE . . . . . .  BYTE ARRAY(4) DATA
 28  0000H   128  DBUFF. . . . . .  BYTE BASED(DBUFFADR) ARRAY(128)          318
 28  000AH     2  DBUFFADR . . . .  WORD          28  300  306  318 1141
 24              DCL. . . . . .  LITERALLY
 33  017EH     1  DCNT . . . . . .  BYTE        135  139  235  259  261  287  381  671  679  683  694 1566
                                 1578
 28  0114H     1  DDISK. . . . . .  BYTE AT         1165 1216 1217
783  176FH     9  DECBACK. . . . .  PROCEDURE STACK=0002H          804
769  173CH     9  DECBASE. . . . .  PROCEDURE STACK=0002H          781
778  1757H    24  DECFRONT . . . .  PROCEDURE STACK=0006H          800  821 1265 1281 1290
542  0064H    15  DEL. . . . . .  BYTE ARRAY(15) DATA       543  544
137  0AEEH    19  DELETE . . . . .  PROCEDURE STACK=000AH          201  223 1565
220  0C64H    16  DELETEFILE . . .  PROCEDURE STACK=0014H          227  233  286 1362 1366
541  1411H    54  DELIM. . . . . .  PROCEDURE BYTE STACK=0030H          565  567  587  599
```

```
  535  02FFH     1  DELIMITER. . . . .  BYTE          543  544  546  608
   28  0114H    33  DFCB . . . . . .  BYTE ARRAY(33)          28  213  284  301  380  385  387  395  396  397
                                        627  630  631  633  634  691  692  693  698  700  702 1166 1221 1222
                                       1362 1365 1366
 1107  1D66H    20  DIGIT. . . . . .  PROCEDURE BYTE STACK=0002H          1112 1383
   40  02EEH     1  DIRECTION. . . . .  BYTE          727  756  788  825  890  901  999 1036 1037 1040 1054 1120
                                       1124 1233 1376 1397 1401 1404 1430 1442 1452 1465 1469 1478 1506 1532
                                       1536
   38  0172H        DIRERR . . . . . .  LABEL       1183 1579
   44  0017H    15  DIRFULL. . . . . .  BYTE ARRAY(15) DATA          214 1184
   38  0160H        DISKERR. . . . . .  LABEL        308  332 1180
   44  0026H    10  DISKFULL . . . . .  BYTE ARRAY(10) DATA         1181
  141  0B01H    16  DISKREAD . . . . .  PROCEDURE BYTE STACK=000AH           259  360  643
  145  0B11H    16  DISKWRITE. . . . .  PROCEDURE BYTE STACK=000AH           301  328
   40  0018H     2  DISTANCE . . . . .  WORD          708  711  714  719  729  744  828  830  834  836 1023 1026
                                       1029 1055 1111 1113 1118 1121 1125 1232 1398 1514 1520
  716  1642H    24  DISTNZERO. . . . .  PROCEDURE BYTE STACK=0006H           883  896 1059 1434 1475 1484 1495
                                       1525 1546 1595
  710  1628H    15  DISTZERO . . . . .  PROCEDURE BYTE STACK=0002H           717  894 1400 1428 1445 1465 1473
                                       1562 1590
   29  0164H     1  DOTFOUND . . . . .  BYTE INITIAL           555  585  985
                    DOUBLE . . . . . .  BUILTIN         1132
  534               DRIVE. . . . . .  LITERALLY         577
   30  0165H     3  DTYPE. . . . . .  BYTE ARRAY(3)          396  634 1221 1365
    1  0002H  2343  ED . . . . . .  PROCEDURE STACK=0048H
   24               ENDFILE. . . . . .  LITERALLY         263  273  276  342  364  375  444  542  558  644  850
                                        912  926 1326 1354 1592
  280  0D87H    64  ERASEBAK . . . . .  PROCEDURE BYTE STACK=0018H           302  329
  614  13E1H        ERR. . . . . . .  LABEL        569  576  578  589  601  609  611
   44  0026H     2  ERRMSG . . . . . .  WORD INITIAL          125 1181 1184 1192 1194 1195
   31  0010H     2  ERRORCODE. . . . .  WORD          656  660  666  668  671
   24               ESC. . . . . . .  LITERALLY         443
   28               EX . . . . . . .  LITERALLY          28  650  692  698  702 1157 1158
  439  10C8H        EXIT . . . . . .  LABEL        424
  237  0006H     1  F. . . . . . .  BYTE PARAMETER AUTOMATIC          238  240
   25               FALSE. . . . . .  LITERALLY          29  163  290  351  452  509  551  553  555  615  629
                                        722  747  812  950 1083 1105 1150 1164 1185 1201 1213 1271 1344 1357
                                       1564
  149  0004H     2  FCB. . . . . .  WORD PARAMETER AUTOMATIC          150  151
  145  0004H     2  FCB. . . . . .  WORD PARAMETER AUTOMATIC          146  147
  141  0004H     2  FCB. . . . . .  WORD PARAMETER AUTOMATIC          142  143
  137  0004H     2  FCB. . . . . .  WORD PARAMETER AUTOMATIC          138  139
  133  0004H     2  FCB. . . . . .  WORD PARAMETER AUTOMATIC          134  135
  129  0004H     2  FCB. . . . . .  WORD PARAMETER AUTOMATIC          130  131
  120  0004H     2  FCB. . . . . .  WORD PARAMETER AUTOMATIC          121  122
   16  0000H     1  FCB. . . . . .  BYTE ARRAY(1) EXTERNAL(3)          28  259  666  674  676  678 1145
  191  0004H     2  FCB. . . . . .  WORD PARAMETER AUTOMATIC          192  193
  185  0004H     2  FCB. . . . . .  WORD PARAMETER AUTOMATIC          186  188
  172  0004H     2  FCB. . . . . .  WORD PARAMETER AUTOMATIC          173  174
   17  0000H     1  FCB16. . . . . .  BYTE ARRAY(1) EXTERNAL(4)
  975  0004H     2  FCBADR . . . . . .  WORD PARAMETER AUTOMATIC          976  977  979  985  986  987  988
  531  002AH     2  FCBADR . . . . . .  WORD PARAMETER          532  533  538  560  561  577  579  596  605
  212  0C3EH    19  FERR . . . . . .  PROCEDURE STACK=0036H          262  382  695
  237  0CAAH    32  FILL . . . . . .  PROCEDURE STACK=0008H          419  560  561  596
  251  0CEDH    91  FILLSOURCE . . . .  PROCEDURE STACK=003AH          275
  946  1A2DH   129  FIND . . . . . .  PROCEDURE BYTE STACK=0018H           972 1503 1526
  369  0F5DH   135  FINIS. . . . . .  PROCEDURE STACK=003AH          915
```

```
  40  0020H     2  FIRST. . . . . .  WORD       758  762  799  829  830  833  891 1004 1011 1041 1405 1470
                                      1505 1533 1582
  37  02E9H     1  FLAG . . . . . .  BYTE       124  554  615  980  982 1174 1176 1178 1180 1183 1187 1203
                                      1343 1347 1560
  25             FOREVER. . . . . .  LITERALLY        847  866 1094 1200
  36             FORWARD. . . . . .  LITERALLY        788  901  999 1037 1054 1120 1397 1430 1442 1452 1465
                                      1469 1536
  40  001CH     2  FRONT. . . . . .  WORD       730  759  773  779  780  795  799  803  820  828  830  833
                                       848  855  918  920 1002 1060 1072 1170 1234 1280 1291 1294 1447 1470
                                      1502 1505 1548
  28             FS . . . . . . . .  LITERALLY         28  284
  12  0000H     1  FUNC . . . . . .  BYTE PARAMETER          13
   8  0000H     1  FUNC . . . . . .  BYTE PARAMETER           9
   4  0000H     1  FUNC . . . . . .  BYTE PARAMETER           5
 484  11A0H    34  GETCMD . . . . . .  PROCEDURE BYTE STACK=0002H          528
 415  1048H   136  GETPASSWD. . . . .  PROCEDURE STACK=002AH          662
 270  0D53H    52  GETSOURCE. . . . .  PROCEDURE BYTE STACK=003EH          850  912
 526  1294H    27  GETUC. . . . . .  PROCEDURE STACK=0034H          556  563  571  580  586  591  598  603
  82  09F9H    53  GRAPHIC. . . . . .  PROCEDURE BYTE STACK=0006H           90 1300
  29  0162H     1  HASBDOS3 . . . . .  BYTE INITIAL          217  651  657 1130 1154
                   HIGH . . . . . .  BUILTIN          660  668 1159
  26  0004H     2  HMAX . . . . . .  WORD        884 1060 1144
  27  003CH     1  I. . . . . .  BYTE       1108 1113
1371  0310H     1  I. . . . . .  BYTE
  74  02F1H     1  I. . . . . .  BYTE         75   78
1341  030FH     1  I. . . . . .  BYTE
1092  030EH     1  I. . . . . .  BYTE       1098 1100 1102
1570  003AH     2  I. . . . . .  WORD       1582 1583
 618  0301H     1  I. . . . . .  BYTE
 725  002CH     2  I. . . . . .  WORD        730  735  741  742  748  753  758  763
 535  02FEH     1  I. . . . . .  BYTE        538  566  568  575  584  588  597  600
 416  02FAH     1  I. . . . . .  BYTE        420  422  429  432
 347  02F9H     1  I. . . . . .  BYTE        348  349  350
 340  02F8H     1  I. . . . . .  BYTE        341
 995  0036H     2  I. . . . . .  WORD       1002 1004 1005 1011 1019
 293  02F5H     1  I. . . . . .  BYTE        299
1046  030CH     1  I. . . . . .  BYTE       1047
 252  02F3H     1  I. . . . . .  BYTE        257  264
1032  030BH     1  I. . . . . .  BYTE       1036 1040
 775  174EH     9  INCBACK. . . . . .  PROCEDURE STACK=0002H          790  872
 766  1733H     9  INCBASE. . . . . .  PROCEDURE STACK=0002H          794  859  876  923
 772  1745H     9  INCFRONT . . . . .  PROCEDURE STACK=0002H          796  856  921
   8  0000H     2  INFO . . . . . .  WORD PARAMETER          10
  12  0000H     2  INFO . . . . . .  WORD PARAMETER          14
   4  0000H     2  INFO . . . . . .  WORD PARAMETER           6
1065  1C85H    21  INSCRLF. . . . . .  PROCEDURE STACK=000AH         1297 1306 1328
1071  1C9AH    25  INSERRORCHK. . . .  PROCEDURE STACK=0002H         1262 1289
 917  198BH    41  INSERT . . . . . .  PROCEDURE STACK=0006H         1067 1069 1315 1355 1500 1552
  29  015DH     1  INSERTING. . . . .  BYTE        472  505  926  998 1201 1229 1305 1331
  44  0006H    17  INVALID. . . . . .  BYTE ARRAY(17) DATA        614
  74  02F2H     1  J. . . . . .  BYTE         78
 948  0034H     2  J. . . . . .  WORD        949  951  952
 725  002EH     2  K. . . . . .  WORD        732  737  741  748
 948  0307H     1  K. . . . . .  BYTE        953  954  955
 450  0028H     2  K. . . . . .  WORD        451  453  454  455  456
 725  0030H     2  L. . . . . .  WORD        731  736  741
                   LAST . . . . . .  BUILTIN          543
```

```
  40  0022H     2  LASTC. . . . . .  WORD       759  763  789  827  835  836  892  899  952  954  956  961
                                      1011 1041 1406 1471 1582
  24                LCA. . . . . .  LITERALLY       407
  24                LCZ. . . . . .  LITERALLY       407
  25                LF . . . . . .  LITERALLY        61   86   99  519  741  780  793  857  874  922  939
                                      1005 1012 1068 1169 1189 1234 1294 1318 1321
  30  0168H    12  LIBFCB . . . . .  BYTE ARRAY(12) INITIAL          201  986  988
  29  0161H     1  LINESET. . . . .  BYTE INITIAL          467  510 1331 1452
  24                LIT. . . . . .  LITERALLY        24   36  534
 725  0303H     1  LOOPING. . . . .  BYTE       739  740  744  747  752
                   LOW. . . . . .  BUILTIN          374  454  660  668  671 1129
 405  1010H    29  LOWERCASE. . . .  PROCEDURE BYTE STACK=0006H          411 1078
  41  02F0H     1  LPP. . . . . .  BYTE INITIAL         1023 1160 1161
 725  0032H     2  M. . . . . .  WORD       741  742  753
  37  0202H   128  MACRO. . . . . .  BYTE ARRAY(128)       503 1516
  36                MACSIZE. . . . .  LITERALLY       37
 231  0C8EH    28  MAKEFILE . . . .  PROCEDURE STACK=001AH          693 1577
 948  0308H     1  MATCH. . . . . .  BYTE       950  951  954  959  964
  26  0000H     2  MAX. . . . . .  WORD       834 1131 1133 1138 1142 1143
  19  0000H     2  MAXB . . . . . .  WORD EXTERNAL(6)         1131 1140
  34  017FH     1  MAXLEN . . . . .  BYTE       154  155
  26  0002H     2  MAXM . . . . . .  WORD       512  736  835  867  884  951 1041 1143 1144 1171 1406 1449
                                      1471 1537
  28                MD . . . . . .  LITERALLY       650
 786  1778H   107  MEMMOVE. . . . .  PROCEDURE STACK=000CH          809  812
       0000H        MEMORY . . . . .  BYTE ARRAY(0)        741  780  793  795  803  855  873  920  954 1005
                                      1019 1131 1142 1169 1234 1280 1291 1294 1583
  37  02EBH     1  MI . . . . . .  BYTE      1204 1458 1497 1498 1499 1548 1549 1550 1551
 725  0302H     1  MIDDLE . . . . .  BYTE       741  745
 195  0004H     1  MODE . . . . . .  BYTE PARAMETER AUTOMATIC       196  197
   4  0000H        MON1 . . . . . .  PROCEDURE EXTERNAL(0) STACK=0000H          21   55  103  118  151  170
                                       174  193  197 1097
   8  0000H        MON2 . . . . . .  PROCEDURE BYTE EXTERNAL(1) STACK=0000H          46   49  122  135  139
                                       143  147  158  160  166  188  235 1160 1166
  12  0000H        MON3 . . . . . .  PROCEDURE WORD EXTERNAL(2) STACK=0000H        131  205
 176  0B98H    37  MOVE . . . . . .  PROCEDURE STACK=0008H          187  246  284  306  372  605  631  633
                                       634  986  988 1222 1348 1575
 786  0004H     1  MOVEFLAG . . . . .  BYTE PARAMETER AUTOMATIC       787  791  801
 814  17F9H    11  MOVELINES. . . . .  PROCEDURE STACK=0014H         1035 1425 1460
 808  17E3H    11  MOVER. . . . . .  PROCEDURE STACK=0010H          816  893  902  962 1407 1412 1472 1479
                                      1507 1534
 370  0FE4H    24  MOVEUP . . . . . .  PROCEDURE STACK=000EH          391  395
  37  02EAH     1  MP . . . . . .  BYTE       401  480  490  494 1089 1199 1229 1308 1458 1511 1518
   2                MPMPRODUCT . . . .  LITERALLY
  37  0016H     2  MT . . . . . .  WORD       496  498 1520
 293  02F6H     1  N. . . . . .  BYTE       294  295  299
  28  0103H     1  NBUF . . . . . .  BYTE       257  264 1131 1132
  39  02EDH     1  NCMD . . . . . .  BYTE INITIAL         485  486
  28  000EH     2  NDEST. . . . . .  WORD       294  297  298  300  305  306  307  310  312  316  318  319
                                       374  704
  29  0156H     1  NEWFILE. . . . .  BYTE INITIAL         272  282  378  687  911 1213 1363
 818  0004H     2  NEWFRONT . . . .  WORD PARAMETER AUTOMATIC       819  820
  44  0051H    19  NOTAVAIL . . . .  BYTE ARRAY(19) DATA       669
  44  0042H    15  NOTFOUND . . . .  BYTE ARRAY(15) DATA       125  686 1567
  28                NR . . . . . .  LITERALLY        28  650
  28  000CH     2  NSOURCE. . . . .  WORD       254  258  263  266  274  276  277  703
1110  1D7AH    49  NUMBER . . . . .  PROCEDURE STACK=0038H         1385 1395
```

```
  421  106EH        NXTCHR . . . . . LABEL      433
   29  0157H     1  ONEFILE. . . . . BYTE INITIAL          281  389  629  672  685 1165 1214 1223 1363
  120  0AA9H    34  OPEN . . . . . . PROCEDURE STACK=000AH          357 1351
  129  0ACBH    16  OPENFILE . . . . . PROCEDURE WORD STACK=000AH          656  666
   38  013CH        OVERCOUNT. . . . . LABEL      493  499  973 1017 1174 1504 1538
   38  0154H        OVERFLOW . . . . . LABEL      849  919  932 1178
  946  0006H     1  PA . . . . . . . BYTE PARAMETER AUTOMATIC       947  953
 1031  1BE7H    61  PAGE . . . . . . PROCEDURE STACK=0034H         1435
  531  12AFH   322  PARSEFCB . . . . . PROCEDURE BYTE STACK=0038H          627  979 1152
  975  1AD9H    89  PARSELIB . . . . . PROCEDURE BYTE STACK=003EH         1343 1560
   28  0104H    16  PASSWORD . . . . . BYTE ARRAY(16) INITIAL       187  218  419  422  432  605
   44  0030H    18  PASSWORDERR. . . . BYTE ARRAY(18) DATA      189
   42  0004H     2  PB . . . . . . . BYTE ARRAY(2) DATA      1160
  946  0004H     1  PB . . . . . . . BYTE PARAMETER AUTOMATIC       947  954
  110  0A82H    23  PERROR . . . . . . PROCEDURE STACK=002CH         189  209  547  614 1135 1194 1567
  536  0300H     1  PFLAG. . . . . . BYTE       539  553  610  613
    3  0010H        PLMSTART . . . . . LABEL PUBLIC       1128
   36               POUND. . . . . . LITERALLY      1174 1378
  105  0A72H    16  PRINT. . . . . . PROCEDURE STACK=0026H          112  418  621  688 1147 1186
   73  09B6H    67  PRINTABS . . . . . PROCEDURE STACK=0016H           92   95
  476  1183H    12  PRINTBASE. . . . . PROCEDURE STACK=002CH          482  514 1235
   88  0A2EH    35  PRINTC . . . . . . PROCEDURE STACK=001CH           98   99  403  460  462  470  471  473
                                        474  516  519 1019 1187 1191 1291 1296 1448
   51  0955H    26  PRINTCHAR. . . . . PROCEDURE STACK=000AH           63 1096
  465  1150H    51  PRINTLINE. . . . . PROCEDURE STACK=0028H          477  513  992
  101  0A62H    16  PRINTM . . . . . . PROCEDURE STACK=000AH          108  113 1188 1190
  399  0FFCH    20  PRINTNMAC. . . . . PROCEDURE STACK=0022H         1264 1302 1303 1321
  479  118FH    17  PRINTNMBASE. . . . PROCEDURE STACK=0030H         1253 1319
  991  1B32H    16  PRINTREL . . . . . PROCEDURE STACK=002CH         1014
   29  015AH     1  PRINTSUPPRESS. . . BYTE INITIAL          53 1185 1268 1271
  449  10F3H    93  PRINTVALUE . . . . PROCEDURE STACK=0022H          469 1447 1449
   24               PROC . . . . . . . LITERALLY       176  237  415  484  526  531  537  541  724  772  778
                                        783  786  808  811  814  818  824
   29  015CH     1  PROTECTION . . . . BYTE INITIAL          696  698 1157
  537  13F1H    32  PUTC . . . . . . PROCEDURE STACK=0002H          570  590  602
  314  0E53H    36  PUTDEST. . . . . . PROCEDURE STACK=0022H          375  873  913
  321  0E77H    74  PUTXFER. . . . . . PROCEDURE STACK=001EH          342 1583
   32  017DH     1  QCOLUMN. . . . . . BYTE      1272 1273 1275
   28  005EH     1  RBP. . . . . . . BYTE       640  645  647 1350
  116  0A99H    16  READ . . . . . . PROCEDURE STACK=000AH          155
   29  015EH     1  READBUFF . . . . . BYTE       507  509  522  623 1198
  489  11C2H   210  READC. . . . . . . PROCEDURE BYTE STACK=0030H          529  926 1084 1516
   45  0937H    15  READCHAR . . . . . PROCEDURE BYTE STACK=0008H          506 1098 1592
  153  0B31H    17  READCOM. . . . . . PROCEDURE STACK=000EH          517  622 1148
 1082  1CDDH    19  READCTRAN. . . . . PROCEDURE STACK=0034H         1114 1202 1375 1381 1394
  639  148AH    56  READFILE . . . . . PROCEDURE BYTE STACK=0016H         1354
   29  0159H     1  READING. . . . . . BYTE INITIAL         1344 1345 1352 1357
  839  1869H    64  READLINE . . . . . PROCEDURE STACK=0042H         1062 1476
  191  0BE6H    16  READXFCB . . . . . PROCEDURE STACK=000AH         1156
  199  0C09H    22  REBOOT . . . . . . PROCEDURE STACK=000EH          210  437 1153 1208 1368
 1117  1DABH    40  RELDISTANCE. . . . PROCEDURE STACK=0002H         1389 1396
   35  0014H     2  RELLINE. . . . . . WORD       726  751  992 1001 1010 1015
  149  0B21H    16  RENAME . . . . . . PROCEDURE STACK=000AH          229
  225  0C74H    26  RENAMEFILE . . . . PROCEDURE STACK=001AH          393  397
   38  017DH        RESET. . . . . . . LABEL      126  983 1049 1073 1175 1177 1179 1182 1185 1528 1593
   38  0120H        RESTART. . . . . . LABEL     1168 1225 1337
 1028  1BDCH    11  RESTDIST . . . . . PROCEDURE STACK=0002H         1043 1535
```

```
 419  1055H        RETRY. . . . . .   LABEL       426  430
 325  0E81H        RETRY. . . . . .   LABEL       330
 300  0DF7H        RETRY. . . . . .   LABEL       303
 195  0BF6H    19  RETURNERROKS . . .  PROCEDURE STACK=000AH          653  659
  28  003DH    33  RFCB . . . . . .   BYTE ARRAY(33) INITIAL         350  643 1343 1348 1349 1351 1358 1560
                                       1565 1575
  28               RO . . . . . .     LITERALLY       674
                   ROL. . . . . .     BUILTIN         674  676  678
  36               RUBOUT . . . . .   LITERALLY      1287
 237  0008H     2  S. . . . . .       WORD PARAMETER AUTOMATIC       238  240  241
 176  0006H     2  S. . . . . .       WORD PARAMETER AUTOMATIC       177  178  180  181
1025  18D1H    11  SAVEDIST . . . .   PROCEDURE STACK=0002H         1033 1529
 293  02F7H     1  SAVENDEST. . . .   BYTE        297  306  307
  28  0000H   128  SBUFF. . . . . .   BYTE BASED(SBUFFADR) ARRAY(128)      263  276
  28  0008H     2  SBUFFADR . . . .   WORD         28  258  263  276 1140 1141
 925  19B4H    42  SCANNING . . . .   PROCEDURE BYTE STACK=0034H         934 1238
  32  017BH     1  SCOLUMN. . . . .   BYTE INITIAL         1255 1266 1272 1273 1453 1454
  37  0282H   100  SCRATCH. . . . .   BYTE ARRAY(100)       930  954 1499 1550
  36               SCRSIZE. . . . .   LITERALLY        37  931
  28  0000H     1  SDISK. . . . . .   BYTE EXTERNAL(3) AT       1165 1217 1218
  19               SECTSHF. . . . .   LITERALLY       294 1131 1132
  19               SECTSIZE . . . .   LITERALLY        28  266  310  323  341  363  640 1350
 172  0B88H    16  SETATTRIBUTE . . .  PROCEDURE STACK=000AH          387
 824  1818H    81  SETCLIMITS . . . .  PROCEDURE STACK=0002H         1411 1416
 617  1447H   103  SETDEST. . . . . .  PROCEDURE STACK=003CH         1163
 168  0B78H    16  SETDMA . . . . . .  PROCEDURE STACK=000AH          218  249  258  300  637
 707  161DH    11  SETFF. . . . . .   PROCEDURE STACK=0002H          882  906 1058 1380
 966  1AAEH    19  SETFIND. . . . . .  PROCEDURE STACK=003CH         1483 1491 1524 1544
1053  1C53H    16  SETFORWARD . . . .  PROCEDURE STACK=0002H         1372 1461
 818  1804H    20  SETFRONT . . . . .  PROCEDURE STACK=000CH         1508 1548
 724  165AH   217  SETLIMITS. . . . .  PROCEDURE STACK=0004H          815  997 1248 1421 1581
1022  1BC4H    13  SETLPP . . . . . .  PROCEDURE STACK=0002H         1034 1038 1431
 216  0C51H    19  SETPASSWORD. . . .  PROCEDURE STACK=000EH          222  228  234  386  654  665  699
 811  17EEH    11  SETPTRS. . . . . .  PROCEDURE STACK=0010H         1249 1417 1422
 636  14AEH    12  SETRDMA. . . . . .  PROCEDURE STACK=000EH          642 1342
 929  1A0DH    32  SETSCR . . . . . .  PROCEDURE STACK=0002H          938  943
 244  0CCAH    23  SETTYPE. . . . . .  PROCEDURE STACK=0010H          285  392  396  691 1221 1365
 649  14F2H   299  SETUP. . . . . .   PROCEDURE STACK=003AH         1168
 248  0CE1H    12  SETXDMA. . . . . .  PROCEDURE STACK=000EH          325  359
  28  0000H    33  SFCB . . . . . .   BYTE ARRAY(33) EXTERNAL(3) AT       379  391  392  393  631  633  650
                                        656 1152 1156 1157 1158 1222 1336
                   SHL. . . . . .     BUILTIN        1113 1132
                   SHR. . . . . .     BUILTIN         294 1131 1139 1144
1087  1CF0H    46  SINGLECOM. . . . .  PROCEDURE BYTE STACK=0006H        1093 1205 1210
1091  1D1EH    72  SINGLERCOM . . . .  PROCEDURE BYTE STACK=0026H        1334 1360
  38  01D1H        START. . . . . .   LABEL      1101 1173 1198
  28               SY . . . . . .     LITERALLY       385  676
  29  015BH     1  SYS. . . . . .     BYTE INITIAL         383  680
1490  0038H     2  T. . . . . . .     WORD       1502 1508
1076  030DH     1  T. . . . . . .     BYTE       1078 1080
  24               TAB. . . . . .     LITERALLY        75   76   86  112 1186 1312
  29  0163H     1  TAIL . . . . . .   BYTE INITIAL          527  619 1150 1164
  18  0000H     1  TBUFF. . . . . .   BYTE ARRAY(1) EXTERNAL(5)
  32  017CH     1  TCOLUMN. . . . . .  BYTE       1072 1231 1263 1266 1274 1323
  40  001AH     2  TDIST. . . . . .   WORD       1026 1029
  30  0174H     3  TEMPFL . . . . . .  BYTE ARRAY(3) INITIAL          691
 909  1964H    39  TERMINATE. . . . .  PROCEDURE STACK=0042H         1207 1212
```

```
1075  1CB3H    42  TESTCASE . . . . .  PROCEDURE STACK=0016H          1085
                   TIME . . . . . .    BUILTIN        1050
  28  0135H    33  TMPFCB . . . . . .  BYTE ARRAY(33)        284  285  286
  29  015FH     1  TRANSLATE. . . . .  BYTE INITIAL          445 1077 1080 1083
  25                TRUE . . . . . .    LITERALLY          29   85  161  288  353  459  539  548  554  585  680
                                         687  720  759  809  847  866  998 1077 1094 1103 1130 1198 1200 1223
                                        1268 1352 1574
  57  096FH    34  TTYCHAR. . . . . .  PROCEDURE STACK=0010H           68   69   70   79
 994  1B42H   130  TYPELINES. . . . .  PROCEDURE STACK=0030H          1039 1236 1245 1270 1283 1432 1440 1462
  28                UB . . . . . .      LITERALLY
 409  102DH    27  UCASE. . . . . .  PROCEDURE BYTE STACK=000CH          421  446  528  529 1098
  29  0160H     1  UPPER. . . . . .  BYTE INITIAL          843 1080 1442
  28                US . . . . . .      LITERALLY        678
 441  10D0H    35  UTRAN. . . . . .  PROCEDURE BYTE STACK=0012H          503  506  524  647  844 1079
 465  0004H     2  V. . . . . .  WORD PARAMETER AUTOMATIC        466  469
 449  0004H     2  V. . . . . .  WORD PARAMETER AUTOMATIC        450  454  455
  43  0024H     2  VER. . . . . .  WORD       1128 1129 1159
 204  0C1FH    15  VERSION. . . . . .  PROCEDURE WORD STACK=0008H        1128
1045  1C24H    47  WAIT . . . . . .  PROCEDURE STACK=000CH          1436 1596
  37  02E7H     1  WBE. . . . . .  BYTE         930  931  967  969 1493 1503 1505 1549
  37  02E8H     1  WBJ. . . . . .  BYTE        1493 1498 1503 1505
  37  02E6H     1  WBP. . . . . .  BYTE         969  972 1497 1526 1548
  25                WHAT . . . . . .    LITERALLY       1097 1176
 881  18F3H    38  WRHALF . . . . . .  PROCEDURE STACK=002AH          895
 292  0DC7H   140  WRITEDEST. . . . .  PROCEDURE STACK=001CH          317  377
 864  18C2H    49  WRITELINE. . . . .  PROCEDURE STACK=0026H          886  897
 889  1919H    64  WRITEOUT . . . . .  PROCEDURE STACK=002EH          907 1557
 185  0BBDH    41  WRITEXFCB. . . . .  PROCEDURE STACK=0032H          700
  28  0102H     1  XBP. . . . . .  BYTE         323  334  336  337  341  363  364 1561
  28  0082H   128  XBUFF. . . . . .  BYTE ARRAY(128)       249  336  364
  28  005FH    33  XFCB . . . . . .  BYTE ARRAY(33) INITIAL          28  328  344  350  357  360 1348 1575
                                        1577
  28  006BH     1  XFCBE. . . . . .  BYTE AT           326  356 1576
  28  0080H     1  XFCBEXT. . . . . .  BYTE INITIAL          326  356 1576
  28  007FH     1  XFCBR. . . . . .  BYTE AT           327  358  362 1576
  28  0081H     1  XFCBREC. . . . . .  BYTE INITIAL          327  358  362 1576
  29  0158H     1  XFERON . . . . . .  BYTE INITIAL          200 1564 1571 1574
  37  02ECH     1  XP . . . . . .  BYTE         494  501  503 1513 1516 1518 1519
 450  02FDH     1  ZERO . . . . . .  BYTE         452  457  459
 713  1637H    11  ZERODIST . . . . .  PROCEDURE STACK=0002H          852  869  885 1042 1061 1515
 253  0D48H    11  ZN . . . . . .  PROCEDURE STACK=0002H          256  268
```


MODULE INFORMATION:

```
     CODE AREA SIZE     = 10D3H   7635D
     CONSTANT AREA SIZE = 013EH    318D
     VARIABLE AREA SIZE = 0311H    785D
     MAXIMUM STACK SIZE = 0048H     72D
     2639 LINES READ
     0 PROGRAM ERROR(S)
```

END OF PL/M-86 COMPILATION

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

SYMBOL TABLE OF MODULE SCD
READ FROM FILE ED.LNK
WRITTEN TO FILE ED.

BASE   OFFSET TYPE SYMBOL              BASE   OFFSET TYPE SYMBOL

0000H  0110H  PUB  PLMSTART           0000H  0050H  PUB  XDOS
0000H  0050H  PUB  MON1              0000H  0050H  PUB  MON2
0000H  0050H  PUB  MON3              0000H  0050H  PUB  MON4
1000H  0004H  PUB  BDISK             1000H  0006H  PUB  MAXB
1000H  0050H  PUB  CMDRV             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  RO
1000H  0080H  PUB  BUFF              1000H  0080H  PUB  TBUFF
1000H  0080H  PUB  BUFFA             1000H  005CH  PUB  FCBA
1000H  0100H  PUB  SAVEAX            1000H  0102H  PUB  SAVEBX
1000H  0104H  PUB  SAVECX            1000H  0106H  PUB  SAVEDX

ED:  SYMBOLS AND LINES
1000H  05C0H  SYM  MEMORY            0000H  0110H  SYM  PLMSTART
0000H  0A29H  SYM  BOOT              1000H  0482H  SYM  DATE
1000H  0108H  SYM  MAX               1000H  010AH  SYM  MAXM
1000H  010CH  SYM  HMAX              1000H  0144H  SYM  I
1000H  0145H  SYM  RFCB              1000H  0166H  SYM  RBP
1000H  0167H  SYM  XFCB              1000H  0173H  SYM  XFCBE
1000H  0187H  SYM  XFCBR             1000H  0188H  SYM  XFCBEXT
1000H  0189H  SYM  XFCBREC           1000H  018AH  SYM  XBUFF
1000H  020AH  SYM  XBP               1000H  020BH  SYM  NBUF
1000H  010EH  SYM  BUFFLENGTH        1000H  005CH  SYM  SFCB
1000H  005CH  SYM  SDISK             1000H  0110H  SYM  SBUFFADR
1000H  0110H  BAS  SBUFF             1000H  020CH  SYM  PASSWORD
1000H  021CH  SYM  DFCB              1000H  021CH  SYM  DDISK
1000H  0112H  SYM  DBUFFADR          1000H  0112H  BAS  DBUFF
1000H  0114H  SYM  NSOURCE           1000H  0116H  SYM  NDEST
1000H  023DH  SYM  TMPFCB            1000H  025EH  SYM  NEWFILE
1000H  025FH  SYM  ONEFILE           1000H  0260H  SYM  XFERON
1000H  0261H  SYM  READING           1000H  0262H  SYM  PRINTSUPPRESS
1000H  0263H  SYM  SYS               1000H  0264H  SYM  PROTECTION
1000H  0265H  SYM  INSERTING         1000H  0266H  SYM  READBUFF
1000H  0267H  SYM  TRANSLATE         1000H  0268H  SYM  UPPER
1000H  0269H  SYM  LINESET           1000H  026AH  SYM  HASBDOS3
1000H  026BH  SYM  TAIL              1000H  026CH  SYM  DOTFOUND
1000H  026DH  SYM  DTYPE             1000H  0270H  SYM  LIBFCB
1000H  027CH  SYM  TEMPFL            1000H  027FH  SYM  BACKUP
1000H  0118H  SYM  ERRORCODE         1000H  0282H  SYM  COLUMN
1000H  0283H  SYM  SCOLUMN           1000H  0284H  SYM  TCOLUMN
1000H  0285H  SYM  QCOLUMN           1000H  0286H  SYM  DCNT
1000H  0287H  SYM  MAXLEN            1000H  0288H  SYM  COMLEN
1000H  0289H  SYM  COMBUFF           1000H  0309H  SYM  CBP
1000H  011AH  SYM  BASELINE          1000H  011CH  SYM  RELLINE
1000H  030AH  SYM  MACRO             1000H  038AH  SYM  SCRATCH
1000H  03EEH  SYM  WBP               1000H  03EFH  SYM  WBE
1000H  03F0H  SYM  WBJ               1000H  03F1H  SYM  FLAG
1000H  03F2H  SYM  MP                1000H  03F3H  SYM  MI
1000H  03F4H  SYM  XP                1000H  011EH  SYM  MT
```

```
0000H  02D1H  SYM  START
0000H  023CH  SYM  OVERCOUNT
0000H  0260H  SYM  DISKERR
0000H  027DH  SYM  RESET
1000H  03F5H  SYM  NCMD
1000H  0122H  SYM  TDIST
1000H  03F7H  SYM  CHAR
1000H  0126H  SYM  BACK
1000H  012AH  SYM  LASTC
1000H  0486H  SYM  PB
1000H  012EH  SYM  ERRMSG
1000H  0499H  SYM  DIRFULL
1000H  04B2H  SYM  PASSWORDERR
1000H  04D3H  SYM  NOTAVAIL
0000H  0A46H  SYM  CONIN
STACK  0004H  SYM  CHAR
STACK  0004H  SYM  CHAR
0000H  0AB6H  SYM  PRINTABS
1000H  03F9H  SYM  I
0000H  0AF9H  SYM  GRAPHIC
0000H  0B2EH  SYM  PRINTC
0000H  0B51H  SYM  CRLF
STACK  0004H  SYM  A
STACK  0004H  SYM  A
STACK  0004H  SYM  A
STACK  0004H  SYM  A
STACK  0004H  SYM  FCB
STACK  0004H  SYM  FCB
STACK  0004H  SYM  FCB
STACK  0004H  SYM  FCB
STACK  0004H  SYM  FCB
STACK  0004H  SYM  FCB
STACK  0004H  SYM  FCB
0000H  0C42H  SYM  BREAKKEY
0000H  0C78H  SYM  SETDMA
0000H  0C88H  SYM  SETATTRIBUTE
0000H  0C98H  SYM  MOVE
STACK  0006H  SYM  S
STACK  0006H  BAS  A
0000H  0CBDH  SYM  WRITEXFCB
0000H  0CE6H  SYM  READXFCB
0000H  0CF6H  SYM  RETURNERRORS
0000H  0D09H  SYM  REBOOT
0000H  0D2EH  SYM  ABORT
0000H  0D3EH  SYM  FERR
0000H  0D64H  SYM  DELETEFILE
0000H  0D74H  SYM  RENAMEFILE
0000H  0D8EH  SYM  MAKEFILE
0000H  0DAAH  SYM  FILL
STACK  0006H  SYM  F
STACK  0008H  BAS  A
STACK  0006H  SYM  AFCB
0000H  0DE1H  SYM  SETXDMA
1000H  03FBH  SYM  I
0000H  0E53H  SYM  GETSOURCE
0000H  0E87H  SYM  ERASEBAK
1000H  03FDH  SYM  I
1000H  03FFH  SYM  SAVENDEST
0000H  0F53H  SYM  PUTDEST
0000H  0F77H  SYM  PUTXFER
0000H  0220H  SYM  RESTART
0000H  0254H  SYM  OVERFLOW
0000H  0272H  SYM  DIRERR
0000H  0248H  SYM  BADCOM
1000H  0120H  SYM  DISTANCE
1000H  03F6H  SYM  DIRECTION
1000H  0124H  SYM  FRONT
1000H  0128H  SYM  FIRST
1000H  03F8H  SYM  LPP
1000H  012CH  SYM  VER
1000H  0488H  SYM  INVALID
1000H  04A8H  SYM  DISKFULL
1000H  04C4H  SYM  NOTFOUND
0000H  0A37H  SYM  READCHAR
0000H  0A55H  SYM  PRTNTCHAR
0000H  0A6FH  SYM  TTYCHAR
0000H  0A91H  SYM  BACKSPACE
STACK  0004H  SYM  CHAR
1000H  03FAH  SYM  J
STACK  0004H  SYM  C
STACK  0004H  SYM  C
0000H  0B62H  SYM  PRINTM
0000H  0B72H  SYM  PRINT
0000H  0B82H  SYM  PERROR
0000H  0B99H  SYM  READ
0000H  0BA9H  SYM  OPEN
0000H  0BC8H  SYM  OPENFILE
0000H  0B0BH  SYM  CLOSE
0000H  0BEEH  SYM  DELETE
0000H  0C01H  SYM  DISKREAD
0000H  0C11H  SYM  DISKWRITE
0000H  0C21H  SYM  RENAME
0000H  0C31H  SYM  READCOM
0000H  0C69H  SYM  CSELECT
STACK  0004H  SYM  A
STACK  0004H  SYM  FCB
STACK  0008H  SYM  C
STACK  0004H  SYM  D
STACK  0004H  BAS  B
STACK  0004H  SYM  FCB
STACK  0004H  SYM  FCB
STACK  0004H  SYM  MODE
0000H  0D1FH  SYM  VERSION
STACK  0004H  SYM  A
0000H  0D51H  SYM  SETPASSWORD
STACK  0004H  SYM  AFCB
STACK  0004H  SYM  AFCB
STACK  0004H  SYM  AFCB
STACK  0008H  SYM  S
STACK  0004H  SYM  C
0000H  0DCAH  SYM  SETTYPE
STACK  0004H  SYM  A
0000H  0DEDH  SYM  FILLSOURCE
0000H  0E48H  SYM  ZN
1000H  03FCH  SYM  B
0000H  0EC7H  SYM  WRITEDEST
1000H  03FEH  SYM  N
0000H  0EF7H  SYM  RETRY
STACK  0004H  SYM  B
STACK  0004H  SYM  C
```

0000H	0F81H	SYM	RETRY
1000H	0400H	SYM	I
1000H	0401H	SYM	I
0000H	105DH	SYM	FINIS
STACK	0004H	SYM	AFCB
STACK	0004H	SYM	CHAR
STACK	0004H	SYM	C
STACK	0004H	SYM	C
1000H	0402H	SYM	I
0000H	1155H	SYM	RETRY
0000H	11C8H	SYM	EXIT
STACK	0004H	SYM	C
STACK	0004H	SYM	V
1000H	0405H	SYM	ZERO
0000H	1250H	SYM	PRINTLINE
0000H	1283H	SYM	PRINTBASE
0000H	12A0H	SYM	GETCMD
0000H	1394H	SYM	GETUC
1000H	0132H	SYM	FCBADR
1000H	0406H	SYM	I
1000H	0408H	SYM	PFLAG
0000H	1511H	SYM	DELIM
0000H	14E1H	SYM	ERR
1000H	0409H	SYM	I
0000H	158AH	SYM	READFILE
0000H	171DH	SYM	SETFF
0000H	1737H	SYM	ZERODIST
0000H	175AH	SYM	SETLIMITS
1000H	0136H	SYM	K
1000H	013AH	SYM	M
1000H	040BH	SYM	LOOPING
0000H	183CH	SYM	DECBASE
0000H	184EH	SYM	INCBACK
0000H	186FH	SYM	DECBACK
STACK	0004H	SYM	MOVEFLAG
0000H	18E3H	SYM	MOVER
0000H	18F9H	SYM	MOVELINES
STACK	0004H	SYM	NEWFRONT
0000H	1969H	SYM	READLINE
0000H	19A9H	SYM	CTRAN
0000H	19C2H	SYM	WRITELINE
0000H	19F3H	SYM	WRHALF
0000H	1A59H	SYM	CLEARMEM
0000H	1A88H	SYM	INSERT
0000H	1ADEH	SYM	COLLECT
0000H	1B2DH	SYM	FIND
STACK	0004H	SYM	PB
1000H	040FH	SYM	K
0000H	1BAEH	SYM	SETFIND
0000H	1BD9H	SYM	PARSELIB
STACK	0004H	BAS	AFCB
0000H	1C32H	SYM	PRINTREL
1000H	013EH	SYM	I
0000H	1CC4H	SYM	SETLPP
0000H	1CDCH	SYM	RESTDIST
1000H	0413H	SYM	I
1000H	0414H	SYM	I
0000H	1D63H	SYM	APPHALF
0000H	1D9AH	SYM	INSERRORCHK
1000H	0415H	SYM	T

0000H	0FC1H	SYM	CLOSEXFER
0000H	0FE5H	SYM	COMPAREXFER
0000H	1010H	SYM	APPENDXFER
0000H	10E4H	SYM	MOVEUP
0000H	10FCH	SYM	PRINTNMAC
0000H	1110H	SYM	LOWERCASE
0000H	112DH	SYM	UCASE
0000H	1148H	SYM	GETPASSWD
1000H	0403H	SYM	C
0000H	116EH	SYM	NXTCHR
0000H	1190H	SYM	UTRAN
0000H	11F3H	SYM	PRINTVALUE
1000H	0404H	SYM	D
1000H	0130H	SYM	K
STACK	0004H	SYM	V
0000H	128FH	SYM	PRINTNMBASE
0000H	12C2H	SYM	READC
0000H	13AFH	SYM	PARSEFCB
1000H	0132H	BAS	AFCB
1000H	0407H	SYM	DELIMITER
0000H	14F1H	SYM	PUTC
1000H	04E6H	SYM	DEL
0000H	1547H	SYM	SETDEST
0000H	15AEH	SYM	SETDMA
0000H	15F2H	SYM	SETUP
0000H	1728H	SYM	DISTZERO
0000H	1742H	SYM	DISTNZERO
1000H	0134H	SYM	I
1000H	0138H	SYM	L
1000H	040AH	SYM	MIDDLE
0000H	1833H	SYM	INCBASE
0000H	1845H	SYM	INCFRONT
0000H	1857H	SYM	DECFRONT
0000H	1878H	SYM	MEMMOVE
1000H	040CH	SYM	C
0000H	18EEH	SYM	SETPTRS
0000H	1904H	SYM	SETFRONT
0000H	1918H	SYM	SETCLIMITS
1000H	040DH	SYM	B
STACK	0004H	SYM	B
1000H	040EH	SYM	B
0000H	1A19H	SYM	WRITEOUT
0000H	1A64H	SYM	TERMINATE
0000H	1AB4H	SYM	SCANNING
0000H	1B0DH	SYM	SETSCR
STACK	0006H	SYM	PA
1000H	013CH	SYM	J
1000H	0410H	SYM	MATCH
0000H	1BC1H	SYM	CHKFOUND
STACK	0004H	SYM	FCBADR
1000H	0411H	SYM	B
0000H	1C42H	SYM	TYPELINES
1000H	0412H	SYM	C
0000H	1CD1H	SYM	SAVEDIST
0000H	1CE7H	SYM	PAGE
0000H	1D24H	SYM	WAIT
0000H	1D53H	SYM	SETFORWARD
0000H	1D85H	SYM	INSCRLF
0000H	1DB3H	SYM	TESTCASE
0000H	1DDDH	SYM	READCTRAN

```
0000H  1DF0H  SYM  SINGLECOM
0000H  1E1EH  SYM  SINGLERCOM
1000H  0416H  SYM  I
0000H  1E7AH  SYM  NUMBER
1000H  0417H  SYM  I
1000H  0140H  SYM  T
0000H  0A29H  LIN     20
0000H  0A35H  LIN     22
0000H  0A3AH  LIN     46
0000H  0A46H  LIN     48
0000H  0A55H  LIN     50
0000H  0A58H  LIN     53
0000H  0A68H  LIN     56
0000H  0A72H  LIN     59
0000H  0A7CH  LIN     61
0000H  0A87H  LIN     63
0000H  0A91H  LIN     65
0000H  0A98H  LIN     67
0000H  0AA3H  LIN     69
0000H  0AAFH  LIN     71
0000H  0AB6H  LIN     73
0000H  0AD3H  LIN     76
0000H  0ADBH  LIN     78
0000H  0AEFH  LIN     80
0000H  0AF9H  LIN     82
0000H  0B02H  LIN     85
0000H  0B2EH  LIN     87
0000H  0B31H  LIN     90
0000H  0B43H  LIN     93
0000H  0B4DH  LIN     96
0000H  0B54H  LIN     98
0000H  0B60H  LIN    100
0000H  0B65H  LIN    103
0000H  0B72H  LIN    105
0000H  0B78H  LIN    108
0000H  0B82H  LIN    110
0000H  0B8CH  LIN    113
0000H  0B95H  LIN    115
0000H  0B9CH  LIN    118
0000H  0BA9H  LIN    120
0000H  0BB9H  LIN    124
0000H  0BC4H  LIN    126
0000H  0BC8H  LIN    129
0000H  0BDBH  LIN    132
0000H  0BDEH  LIN    135
0000H  0BEEH  LIN    137
0000H  0BFDH  LIN    140
0000H  0C04H  LIN    143
0000H  0C11H  LIN    145
0000H  0C21H  LIN    148
0000H  0C24H  LIN    151
0000H  0C31H  LIN    153
0000H  0C39H  LIN    155
0000H  0C42H  LIN    157
0000H  0C53H  LIN    160
0000H  0C65H  LIN    163
0000H  0C69H  LIN    165
0000H  0C78H  LIN    167
0000H  0C78H  LIN    170
0000H  0C88H  LIN    172

STACK  0004H  SYM  C
STACK  0004H  SYM  C
0000H  1E56H  SYM  DATT
0000H  1E6BH  SYM  RELDISTANCE
1000H  0418H  SYM  I
1000H  0142H  SYM  I
0000H  0A2CH  LIN     21
0000H  0A37H  LIN     45
0000H  0A46H  LIN     47
0000H  0A49H  LIN     49
0000H  0A55H  LIN     51
0000H  0A5FH  LIN     55
0000H  0A6FH  LIN     57
0000H  0A78H  LIN     60
0000H  0A82H  LIN     62
0000H  0A8DH  LIN     64
0000H  0A94H  LIN     66
0000H  0A9DH  LIN     68
0000H  0AA9H  LIN     70
0000H  0AB4H  LIN     72
0000H  0AB9H  LIN     75
0000H  0AD7H  LIN     77
0000H  0AE9H  LIN     79
0000H  0AF5H  LIN     81
0000H  0AFCH  LIN     84
0000H  0B06H  LIN     86
0000H  0B2EH  LIN     88
0000H  0B3DH  LIN     92
0000H  0B47H  LIN     95
0000H  0B51H  LIN     97
0000H  0B5AH  LIN     99
0000H  0B62H  LIN    101
0000H  0B6EH  LIN    104
0000H  0B75H  LIN    107
0000H  0B7EH  LIN    109
0000H  0B85H  LIN    112
0000H  0B92H  LIN    114
0000H  0B99H  LIN    116
0000H  0BA5H  LIN    119
0000H  0BACH  LIN    122
0000H  0BBEH  LIN    125
0000H  0BC7H  LIN    128
0000H  0BCEH  LIN    131
0000H  0BDBH  LIN    133
0000H  0BEAH  LIN    136
0000H  0BF1H  LIN    139
0000H  0C01H  LIN    141
0000H  0C11H  LIN    144
0000H  0C14H  LIN    147
0000H  0C21H  LIN    149
0000H  0C2DH  LIN    152
0000H  0C34H  LIN    154
0000H  0C40H  LIN    156
0000H  0C45H  LIN    158
0000H  0C61H  LIN    161
0000H  0C69H  LIN    164
0000H  0C6CH  LIN    166
0000H  0C78H  LIN    168
0000H  0C84H  LIN    171
0000H  0C8BH  LIN    174
```

```
0000H  0C94H  LIN   175     0000H  0C98H  LIN   176
0000H  0C9BH  LIN   179     0000H  0CA7H  LIN   180
0000H  0CB1H  LIN   181     0000H  0CB4H  LIN   182
0000H  0CB7H  LIN   183     0000H  0CB9H  LIN   184
0000H  0CBDH  LIN   185     0000H  0CC0H  LIN   187
0000H  0CCEH  LIN   188     0000H  0CDBH  LIN   189
0000H  0CE2H  LIN   190     0000H  0CE6H  LIN   191
0000H  0CE9H  LIN   193     0000H  0CF2H  LIN   194
0000H  0CF6H  LIN   195     0000H  0CF9H  LIN   197
0000H  0D05H  LIN   198     0000H  0D09H  LIN   199
0000H  0D0CH  LIN   200     0000H  0D13H  LIN   201
0000H  0D1AH  LIN   202     0000H  0D1DH  LIN   203
0000H  0D1FH  LIN   204     0000H  0D22H  LIN   205
0000H  0D2EH  LIN   206     0000H  0D2EH  LIN   207
0000H  0D31H  LIN   209     0000H  0D37H  LIN   210
0000H  0D3AH  LIN   211     0000H  0D3EH  LIN   212
0000H  0D41H  LIN   213     0000H  0D48H  LIN   214
0000H  0D4FH  LIN   215     0000H  0D51H  LIN   216
0000H  0D54H  LIN   217     0000H  0D5BH  LIN   218
0000H  0D62H  LIN   219     0000H  0D64H  LIN   220
0000H  0D67H  LIN   222     0000H  0D6AH  LIN   223
0000H  0D70H  LIN   224     0000H  0D74H  LIN   225
0000H  0D77H  LIN   227     0000H  0D81H  LIN   228
0000H  0D84H  LIN   229     0000H  0D8AH  LIN   230
0000H  0D8EH  LIN   231     0000H  0D91H  LIN   233
0000H  0D97H  LIN   234     0000H  0D9AH  LIN   235
0000H  0DA6H  LIN   236     0000H  0DAAH  LIN   237
0000H  0DADH  LIN   239     0000H  0DB9H  LIN   240
0000H  0DC1H  LIN   241     0000H  0DC4H  LIN   242
0000H  0DC6H  LIN   243     0000H  0DCAH  LIN   244
0000H  0DCDH  LIN   246     0000H  0DDDH  LIN   247
0000H  0DE1H  LIN   248     0000H  0DE4H  LIN   249
0000H  0DEBH  LIN   250     0000H  0DEDH  LIN   251
0000H  0E48H  LIN   253     0000H  0E4BH  LIN   254
0000H  0E51H  LIN   255     0000H  0DF0H  LIN   256
0000H  0DF3H  LIN   257     0000H  0E01H  LIN   258
0000H  0E0CH  LIN   259     0000H  0E1AH  LIN   261
0000H  0E21H  LIN   262     0000H  0E24H  LIN   263
0000H  0E2FH  LIN   264     0000H  0E35H  LIN   265
0000H  0E37H  LIN   266     0000H  0E3DH  LIN   267
0000H  0E43H  LIN   268     0000H  0E46H  LIN   269
0000H  0E53H  LIN   270     0000H  0E56H  LIN   272
0000H  0E5DH  LIN   273     0000H  0E61H  LIN   274
0000H  0E6AH  LIN   275     0000H  0E6DH  LIN   276
0000H  0E7EH  LIN   277     0000H  0E82H  LIN   278
0000H  0E87H  LIN   279     0000H  0E87H  LIN   280
0000H  0E8AH  LIN   281     0000H  0E91H  LIN   282
0000H  0E98H  LIN   284     0000H  0EA6H  LIN   285
0000H  0EB1H  LIN   286     0000H  0EB8H  LIN   287
0000H  0EBFH  LIN   288     0000H  0EC3H  LIN   290
0000H  0EC7H  LIN   291     0000H  0EC7H  LIN   292
0000H  0ECAH  LIN   294     0000H  0ED4H  LIN   295
0000H  0EDBH  LIN   296     0000H  0EDDH  LIN   297
0000H  0EE3H  LIN   298     0000H  0EE9H  LIN   299
0000H  0EF7H  LIN   300     0000H  0F02H  LIN   301
0000H  0F0DH  LIN   302     0000H  0F14H  LIN   305
0000H  0F1BH  LIN   306     0000H  0F30H  LIN   307
0000H  0F3CH  LIN   308     0000H  0F3FH  LIN   310
0000H  0F45H  LIN   311     0000H  0F4BH  LIN   312
0000H  0F51H  LIN   313     0000H  0F53H  LIN   314
```

```
0000H  0F56H  LIN   316
0000H  0F62H  LIN   318
0000H  0F73H  LIN   320
0000H  0F7AH  LIN   323
0000H  0F81H  LIN   325
0000H  0F8AH  LIN   327
0000H  0F98H  LIN   329
0000H  0FA7H  LIN   332
0000H  0FACH  LIN   336
0000H  0FBDH  LIN   338
0000H  0FC4H  LIN   341
0000H  0FD7H  LIN   343
0000H  0FE4H  LIN   345
0000H  0FE9H  LIN   348
0000H  0FFAH  LIN   350
0000H  100CH  LIN   352
0000H  1010H  LIN   354
0000H  1013H  LIN   356
0000H  1020H  LIN   358
0000H  1029H  LIN   360
0000H  103AH  LIN   363
0000H  1053H  LIN   365
0000H  1058H  LIN   368
0000H  10E4H  LIN   370
0000H  10F8H  LIN   373
0000H  1067H  LIN   375
0000H  106FH  LIN   377
0000H  1078H  LIN   379
0000H  1089H  LIN   381
0000H  1093H  LIN   383
0000H  109FH  LIN   386
0000H  10A9H  LIN   389
0000H  10B7H  LIN   392
0000H  10C9H  LIN   395
0000H  10DBH  LIN   397
0000H  10FCH  LIN   399
0000H  1106H  LIN   403
0000H  1110H  LIN   405
0000H  112DH  LIN   408
0000H  1130H  LIN   411
0000H  1141H  LIN   413
0000H  1148H  LIN   415
0000H  114EH  LIN   418
0000H  1162H  LIN   420
0000H  117CH  LIN   422
0000H  1190H  LIN   425
0000H  119EH  LIN   429
0000H  1186H  LIN   433
0000H  118FH  LIN   437
0000H  11C8H  LIN   439
0000H  11D0H  LIN   441
0000H  11D9H  LIN   444
0000H  11E4H  LIN   446
0000H  11F3H  LIN   448
0000H  11F6H  LIN   451
0000H  1201H  LIN   453
0000H  1214H  LIN   455
0000H  122AH  LIN   457
0000H  123DH  LIN   460
0000H  1248H  LIN   462

0000H  0F5FH  LIN   317
0000H  0F6FH  LIN   319
0000H  0F77H  LIN   321
0000H  0F81H  LIN   324
0000H  0F84H  LIN   326
0000H  0F90H  LIN   328
0000H  0FA5H  LIN   330
0000H  0FA7H  LIN   334
0000H  0FB9H  LIN   337
0000H  0FC1H  LIN   339
0000H  0FD1H  LIN   342
0000H  0FDDH  LIN   344
0000H  0FE6H  LIN   346
0000H  0FEEH  LIN   349
0000H  1008H  LIN   351
0000H  100CH  LIN   353
0000H  101DH  LIN   355
0000H  1019H  LIN   357
0000H  1026H  LIN   359
0000H  1034H  LIN   362
0000H  1046H  LIN   364
0000H  1055H  LIN   366
0000H  105DH  LIN   369
0000H  10E7H  LIN   372
0000H  1060H  LIN   374
0000H  106DH  LIN   376
0000H  1072H  LIN   378
0000H  1082H  LIN   380
0000H  1090H  LIN   382
0000H  109AH  LIN   385
0000H  10A2H  LIN   387
0000H  10B0H  LIN   391
0000H  10C2H  LIN   393
0000H  10D0H  LIN   396
0000H  10E2H  LIN   398
0000H  10FFH  LIN   401
0000H  110CH  LIN   404
0000H  1113H  LIN   407
0000H  112DH  LIN   409
0000H  113AH  LIN   412
0000H  1148H  LIN   414
0000H  114BH  LIN   417
0000H  1155H  LIN   419
0000H  116EH  LIN   421
0000H  1189H  LIN   423
0000H  1197H  LIN   427
0000H  11A5H  LIN   432
0000H  11B8H  LIN   436
0000H  11C2H  LIN   438
0000H  11CEH  LIN   440
0000H  11D3H  LIN   443
0000H  11DDH  LIN   445
0000H  11ECH  LIN   447
0000H  11F3H  LIN   449
0000H  11FCH  LIN   452
0000H  1208H  LIN   454
0000H  121EH  LIN   456
0000H  1238H  LIN   459
0000H  1246H  LIN   461
0000H  124CH  LIN   463
```

```
0000H  124CH  LIN    464
0000H  1253H  LIN    467
0000H  1262H  LIN    470
0000H  126EH  LIN    472
0000H  1279H  LIN    474
0000H  1283H  LIN    476
0000H  128DH  LIN    478
0000H  1292H  LIN    480
0000H  129BH  LIN    482
0000H  12A0H  LIN    484
0000H  12B0H  LIN    486
0000H  12C2H  LIN    488
0000H  12C5H  LIN    490
0000H  12D6H  LIN    494
0000H  12E6H  LIN    498
0000H  12EFH  LIN    501
0000H  1308H  LIN    505
0000H  1315H  LIN    507
0000H  1321H  LIN    510
0000H  1338H  LIN    513
0000H  1344H  LIN    515
0000H  134CH  LIN    517
0000H  1354H  LIN    519
0000H  135FH  LIN    522
0000H  137DH  LIN    524
0000H  1394H  LIN    526
0000H  139EH  LIN    528
0000H  13A0H  LIN    530
0000H  14F1H  LIN    537
0000H  150AH  LIN    539
0000H  1511H  LIN    541
0000H  1520H  LIN    544
0000H  1532H  LIN    547
0000H  153DH  LIN    550
0000H  1547H  LIN    552
0000H  13BDH  LIN    554
0000H  13C7H  LIN    556
0000H  13D1H  LIN    558
0000H  13EBH  LIN    561
0000H  1400H  LIN    563
0000H  1405H  LIN    565
0000H  1416H  LIN    567
0000H  1426H  LIN    569
0000H  1429H  LIN    571
0000H  142EH  LIN    573
0000H  143FH  LIN    576
0000H  1450H  LIN    578
0000H  1458H  LIN    580
0000H  1462H  LIN    584
0000H  146CH  LIN    586
0000H  1478H  LIN    588
0000H  147FH  LIN    590
0000H  1484H  LIN    592
0000H  148BH  LIN    596
0000H  14A0H  LIN    598
0000H  14ACH  LIN    600
0000H  14B3H  LIN    602
0000H  14B8H  LIN    604
0000H  14C9H  LIN    607
0000H  14D3H  LIN    609

0000H  1250H  LIN    465
0000H  125CH  LIN    469
0000H  1268H  LIN    471
0000H  1275H  LIN    473
0000H  127FH  LIN    475
0000H  1286H  LIN    477
0000H  128FH  LIN    479
0000H  1299H  LIN    481
0000H  129EH  LIN    483
0000H  12A3H  LIN    485
0000H  12BEH  LIN    487
0000H  12C2H  LIN    489
0000H  12CCH  LIN    492
0000H  12DFH  LIN    496
0000H  12EFH  LIN    499
0000H  12F4H  LIN    503
0000H  130FH  LIN    506
0000H  131CH  LIN    509
0000H  132FH  LIN    512
0000H  1341H  LIN    514
0000H  1346H  LIN    516
0000H  134FH  LIN    518
0000H  135AH  LIN    520
0000H  1372H  LIN    523
0000H  1394H  LIN    525
0000H  1397H  LIN    527
0000H  13A3H  LIN    529
0000H  13AFH  LIN    531
0000H  14F4H  LIN    538
0000H  150FH  LIN    540
0000H  1514H  LIN    543
0000H  152EH  LIN    546
0000H  1539H  LIN    548
0000H  1543H  LIN    551
0000H  13B8H  LIN    553
0000H  13C2H  LIN    555
0000H  13CAH  LIN    557
0000H  13DBH  LIN    560
0000H  13F9H  LIN    562
0000H  1403H  LIN    564
0000H  1411H  LIN    566
0000H  141FH  LIN    568
0000H  1426H  LIN    570
0000H  142CH  LIN    572
0000H  1435H  LIN    575
0000H  143FH  LIN    577
0000H  1450H  LIN    579
0000H  1458H  LIN    582
0000H  1467H  LIN    585
0000H  146FH  LIN    587
0000H  147FH  LIN    589
0000H  1482H  LIN    591
0000H  1484H  LIN    594
0000H  1498H  LIN    597
0000H  14A3H  LIN    599
0000H  14B3H  LIN    601
0000H  14B6H  LIN    603
0000H  14B8H  LIN    605
0000H  14CCH  LIN    608
0000H  14D3H  LIN    610
```

```
0000H  14DCH  LIN    611
0000H  14E1H  LIN    614
0000H  14F1H  LIN    616
0000H  154AH  LIN    619
0000H  155AH  LIN    622
0000H  1567H  LIN    624
0000H  156DH  LIN    627
0000H  157DH  LIN    630
0000H  1590H  LIN    632
0000H  159EH  LIN    634
0000H  15AEH  LIN    636
0000H  15B8H  LIN    638
0000H  15BDH  LIN    640
0000H  15C7H  LIN    643
0000H  15D6H  LIN    645
0000H  15F2H  LIN    648
0000H  15F5H  LIN    650
0000H  1607H  LIN    653
0000H  1610H  LIN    656
0000H  1621H  LIN    659
0000H  1637H  LIN    662
0000H  163DH  LIN    664
0000H  1643H  LIN    666
0000H  165DH  LIN    669
0000H  166AH  LIN    672
0000H  167AH  LIN    675
0000H  168CH  LIN    678
0000H  169CH  LIN    680
0000H  16A8H  LIN    685
0000H  16B8H  LIN    687
0000H  16C4H  LIN    689
0000H  16D2H  LIN    692
0000H  16DEH  LIN    694
0000H  16E8H  LIN    696
0000H  16F7H  LIN    699
0000H  1701H  LIN    702
0000H  170FH  LIN    704
0000H  1718H  LIN    706
0000H  1720H  LIN    708
0000H  1728H  LIN    710
0000H  1737H  LIN    712
0000H  173AH  LIN    714
0000H  1742H  LIN    716
0000H  174EH  LIN    719
0000H  1756H  LIN    722
0000H  175AH  LIN    724
0000H  1763H  LIN    727
0000H  176EH  LIN    730
0000H  177AH  LIN    732
0000H  1782H  LIN    735
0000H  178EH  LIN    737
0000H  1799H  LIN    740
0000H  17CFH  LIN    742
0000H  17D5H  LIN    744
0000H  17EFH  LIN    747
0000H  17FBH  LIN    749
0000H  1801H  LIN    752
0000H  180EH  LIN    755
0000H  1817H  LIN    758
0000H  1823H  LIN    760

0000H  14DCH  LIN    613
0000H  14E8H  LIN    615
0000H  1547H  LIN    617
0000H  1553H  LIN    621
0000H  155DH  LIN    623
0000H  156AH  LIN    625
0000H  1578H  LIN    629
0000H  1584H  LIN    631
0000H  1590H  LIN    633
0000H  15ACH  LIN    635
0000H  15B1H  LIN    637
0000H  15BAH  LIN    639
0000H  15C4H  LIN    642
0000H  15D2H  LIN    644
0000H  15DBH  LIN    647
0000H  15F2H  LIN    649
0000H  1600H  LIN    651
0000H  160DH  LIN    654
0000H  161AH  LIN    657
0000H  1627H  LIN    660
0000H  163AH  LIN    663
0000H  1640H  LIN    665
0000H  164DH  LIN    668
0000H  1664H  LIN    671
0000H  1671H  LIN    674
0000H  1683H  LIN    676
0000H  1695H  LIN    679
0000H  16A1H  LIN    683
0000H  16B1H  LIN    686
0000H  16BDH  LIN    688
0000H  16C7H  LIN    691
0000H  16D7H  LIN    693
0000H  16E5H  LIN    695
0000H  16EFH  LIN    698
0000H  16FAH  LIN    700
0000H  1709H  LIN    703
0000H  1715H  LIN    705
0000H  171DH  LIN    707
0000H  1726H  LIN    709
0000H  1728H  LIN    711
0000H  1737H  LIN    713
0000H  1740H  LIN    715
0000H  1745H  LIN    717
0000H  1752H  LIN    720
0000H  175AH  LIN    723
0000H  175DH  LIN    726
0000H  176AH  LIN    729
0000H  1774H  LIN    731
0000H  1780H  LIN    733
0000H  1788H  LIN    736
0000H  1794H  LIN    739
0000H  17A0H  LIN    741
0000H  17D3H  LIN    743
0000H  17E6H  LIN    745
0000H  17F4H  LIN    748
0000H  17FDH  LIN    751
0000H  1808H  LIN    753
0000H  1810H  LIN    756
0000H  181DH  LIN    759
0000H  1823H  LIN    762
```

```
0000H  182AH  LIN    763        0000H  1831H  LIN    765
0000H  1833H  LIN    766        0000H  1836H  LIN    767
0000H  183AH  LIN    768        0000H  183CH  LIN    769
0000H  183FH  LIN    770        0000H  1843H  LIN    771
0000H  1845H  LIN    772        0000H  1848H  LIN    773
0000H  184CH  LIN    774        0000H  184EH  LIN    775
0000H  1851H  LIN    776        0000H  1855H  LIN    777
0000H  1857H  LIN    778        0000H  185AH  LIN    779
0000H  1861H  LIN    780        0000H  186AH  LIN    781
0000H  186DH  LIN    782        0000H  186FH  LIN    783
0000H  1872H  LIN    784        0000H  1876H  LIN    785
0000H  1878H  LIN    786        0000H  187BH  LIN    788
0000H  1882H  LIN    789        0000H  188BH  LIN    790
0000H  188EH  LIN    791        0000H  1895H  LIN    793
0000H  18A4H  LIN    794        0000H  18A7H  LIN    795
0000H  18B2H  LIN    796        0000H  18B5H  LIN    798
0000H  18B7H  LIN    799        0000H  18C0H  LIN    800
0000H  18C3H  LIN    801        0000H  18CAH  LIN    803
0000H  18DAH  LIN    804        0000H  18DDH  LIN    806
0000H  18DFH  LIN    807        0000H  18E3H  LIN    808
0000H  18E6H  LIN    809        0000H  18ECH  LIN    810
0000H  18EEH  LIN    811        0000H  18F1H  LIN    812
0000H  18F7H  LIN    813        0000H  18F9H  LIN    814
0000H  18FCH  LIN    815        0000H  18FFH  LIN    816
0000H  1902H  LIN    817        0000H  1904H  LIN    818
0000H  1907H  LIN    820        0000H  190FH  LIN    821
0000H  1912H  LIN    822        0000H  1914H  LIN    823
0000H  1918H  LIN    824        0000H  191BH  LIN    825
0000H  1922H  LIN    827        0000H  1928H  LIN    828
0000H  1931H  LIN    829        0000H  1939H  LIN    830
0000H  1943H  LIN    831        0000H  1945H  LIN    833
0000H  194BH  LIN    834        0000H  1958H  LIN    835
0000H  195DH  LIN    836        0000H  1967H  LIN    838
0000H  1969H  LIN    839        0000H  19A9H  LIN    841
0000H  19ACH  LIN    843        0000H  19B3H  LIN    844
0000H  19BBH  LIN    845        0000H  19C2H  LIN    846
0000H  196CH  LIN    847        0000H  196CH  LIN    848
0000H  1978H  LIN    849        0000H  1978H  LIN    850
0000H  1986H  LIN    852        0000H  1989H  LIN    853
0000H  198BH  LIN    855        0000H  1996H  LIN    856
0000H  1999H  LIN    857        0000H  19A0H  LIN    859
0000H  19A3H  LIN    860        0000H  19A5H  LIN    862
0000H  19A7H  LIN    863        0000H  19C2H  LIN    864
0000H  19C5H  LIN    866        0000H  19C5H  LIN    867
0000H  19CEH  LIN    869        0000H  19D1H  LIN    870
0000H  19D3H  LIN    872        0000H  19D6H  LIN    873
0000H  19E5H  LIN    874        0000H  19ECH  LIN    876
0000H  19EFH  LIN    877        0000H  19F1H  LIN    879
0000H  19F1H  LIN    880        0000H  19F3H  LIN    881
0000H  19F6H  LIN    882        0000H  19F9H  LIN    883
0000H  1A00H  LIN    884        0000H  1A0DH  LIN    885
0000H  1A12H  LIN    886        0000H  1A15H  LIN    887
0000H  1A17H  LIN    888        0000H  1A19H  LIN    889
0000H  1A1CH  LIN    890        0000H  1A21H  LIN    891
0000H  1A27H  LIN    892        0000H  1A2DH  LIN    893
0000H  1A30H  LIN    894        0000H  1A37H  LIN    895
0000H  1A3AH  LIN    896        0000H  1A41H  LIN    897
0000H  1A44H  LIN    898        0000H  1A46H  LIN    899
0000H  1A4FH  LIN    901        0000H  1A54H  LIN    902
0000H  1A57H  LIN    904        0000H  1A59H  LIN    905
```

```
0000H  1A5CH  LIN    906
0000H  1A62H  LIN    908
0000H  1A67H  LIN    910
0000H  1A73H  LIN    912
0000H  1A84H  LIN    914
0000H  1A89H  LIN    916
0000H  1A8EH  LIN    918
0000H  1AA5H  LIN    921
0000H  1AAFH  LIN    923
0000H  1AB4H  LIN    925
0000H  1ADEH  LIN    927
0000H  1B00H  LIN    929
0000H  1B1DH  LIN    931
0000H  1AE1H  LIN    934
0000H  1AEFH  LIN    937
0000H  1AF7H  LIN    939
0000H  1B06H  LIN    942
0000H  1B09H  LIN    944
0000H  1B2DH  LIN    946
0000H  1B36H  LIN    950
0000H  1B53H  LIN    952
0000H  1B63H  LIN    954
0000H  1B93H  LIN    956
0000H  1B99H  LIN    958
0000H  1BA0H  LIN    961
0000H  1BA7H  LIN    964
0000H  1BAEH  LIN    966
0000H  1BB6H  LIN    968
0000H  1BBFH  LIN    970
0000H  1BC4H  LIN    972
0000H  1BD9H  LIN    975
0000H  1BE5H  LIN    980
0000H  1BF3H  LIN    983
0000H  1C08H  LIN    986
0000H  1C1FH  LIN    988
0000H  1C32H  LIN    990
0000H  1C35H  LIN    992
0000H  1C42H  LIN    994
0000H  1C48H  LIN    998
0000H  1C54H  LIN   1001
0000H  1C5FH  LIN   1003
0000H  1C65H  LIN   1005
0000H  1C7CH  LIN   1008
0000H  1C81H  LIN   1010
0000H  1C94H  LIN   1012
0000H  1C9EH  LIN   1015
0000H  1CACH  LIN   1017
0000H  1CB8H  LIN   1020
0000H  1CC4H  LIN   1022
0000H  1CCFH  LIN   1024
0000H  1CD4H  LIN   1026
0000H  1CDCH  LIN   1028
0000H  1CE5H  LIN   1030
0000H  1CEAH  LIN   1033
0000H  1CF0H  LIN   1035
0000H  1CF9H  LIN   1037
0000H  1D01H  LIN   1039
0000H  1D0AH  LIN   1041
0000H  1D1FH  LIN   1043
0000H  1D24H  LIN   1045

0000H  1A5FH  LIN    907
0000H  1A64H  LIN    909
0000H  1A6AH  LIN    911
0000H  1A7DH  LIN    913
0000H  1A86H  LIN    915
0000H  1A8DH  LIN    917
0000H  1A9AH  LIN    920
0000H  1AA8H  LIN    922
0000H  1AB2H  LIN    924
0000H  1AB7H  LIN    926
0000H  1ADEH  LIN    928
0000H  1B10H  LIN    930
0000H  1B2BH  LIN    933
0000H  1AE8H  LIN    935
0000H  1AF4H  LIN    938
0000H  1AFCH  LIN    941
0000H  1B06H  LIN    943
0000H  1B0BH  LIN    945
0000H  1B30H  LIN    949
0000H  1B3BH  LIN    951
0000H  1B5DH  LIN    953
0000H  1B8FH  LIN    955
0000H  1B97H  LIN    957
0000H  1B99H  LIN    959
0000H  1BA4H  LIN    962
0000H  1BAEH  LIN    965
0000H  1BB1H  LIN    967
0000H  1BB9H  LIN    969
0000H  1BC1H  LIN    971
0000H  1BD7H  LIN    974
0000H  1BDCH  LIN    979
0000H  1BEEH  LIN    982
0000H  1BF6H  LIN    985
0000H  1C16H  LIN    987
0000H  1C28H  LIN    989
0000H  1C32H  LIN    991
0000H  1C40H  LIN    993
0000H  1C45H  LIN    997
0000H  1C4DH  LIN    999
0000H  1C5AH  LIN   1002
0000H  1C5FH  LIN   1004
0000H  1C75H  LIN   1007
0000H  1C7FH  LIN   1009
0000H  1C85H  LIN   1011
0000H  1C9BH  LIN   1014
0000H  1CA2H  LIN   1016
0000H  1CACH  LIN   1019
0000H  1CC2H  LIN   1021
0000H  1CC7H  LIN   1023
0000H  1CD1H  LIN   1025
0000H  1CDAH  LIN   1027
0000H  1CDFH  LIN   1029
0000H  1CE7H  LIN   1031
0000H  1CEDH  LIN   1034
0000H  1CF3H  LIN   1036
0000H  1CFEH  LIN   1038
0000H  1D04H  LIN   1040
0000H  1D1AH  LIN   1042
0000H  1D22H  LIN   1044
0000H  1D27H  LIN   1047
```

```
0000H  1D33H  LIN  1048
0000H  1D48H  LIN  1051
0000H  1D53H  LIN  1053
0000H  1D58H  LIN  1055
0000H  1D63H  LIN  1057
0000H  1D69H  LIN  1059
0000H  1D79H  LIN  1061
0000H  1D81H  LIN  1063
0000H  1D85H  LIN  1065
0000H  1D8DH  LIN  1067
0000H  1D95H  LIN  1069
0000H  1D9AH  LIN  1071
0000H  1DB1H  LIN  1074
0000H  1DB6H  LIN  1077
0000H  1DC5H  LIN  1079
0000H  1DDBH  LIN  1081
0000H  1DE0H  LIN  1083
0000H  1DEBH  LIN  1085
0000H  1DF0H  LIN  1087
0000H  1E1EH  LIN  1090
0000H  1E21H  LIN  1093
0000H  1E28H  LIN  1095
0000H  1E34H  LIN  1097
0000H  1E48H  LIN  1099
0000H  1E55H  LIN  1102
0000H  1E60H  LIN  1104
0000H  1E66H  LIN  1106
0000H  1E69H  LIN  1108
0000H  1E7AH  LIN  1110
0000H  1E83H  LIN  1112
0000H  1EA4H  LIN  1114
0000H  1EA9H  LIN  1116
0000H  1EAEH  LIN  1118
0000H  1EBCH  LIN  1121
0000H  1EC2H  LIN  1124
0000H  1ED1H  LIN  1127
0000H  011BH  LIN  1129
0000H  0127H  LIN  1131
0000H  0146H  LIN  1133
0000H  0156H  LIN  1136
0000H  0164H  LIN  1139
0000H  0176H  LIN  1141
0000H  0185H  LIN  1143
0000H  0190H  LIN  1145
0000H  019EH  LIN  1148
0000H  01A4H  LIN  1150
0000H  01B6H  LIN  1153
0000H  01C0H  LIN  1156
0000H  01CDH  LIN  1158
0000H  01DBH  LIN  1160
0000H  01F1H  LIN  1163
0000H  01F9H  LIN  1165
0000H  0219H  LIN  1167
0000H  0223H  LIN  1169
0000H  022EH  LIN  1171
0000H  0239H  LIN  1173
0000H  0246H  LIN  1175
0000H  0252H  LIN  1177
0000H  025EH  LIN  1179
0000H  026AH  LIN  1181

0000H  1D3DH  LIN  1050
0000H  1D51H  LIN  1052
0000H  1D56H  LIN  1054
0000H  1D61H  LIN  1056
0000H  1D66H  LIN  1058
0000H  1D70H  LIN  1060
0000H  1D7EH  LIN  1062
0000H  1D83H  LIN  1064
0000H  1D88H  LIN  1066
0000H  1D90H  LIN  1068
0000H  1D98H  LIN  1070
0000H  1D9DH  LIN  1072
0000H  1DB3H  LIN  1075
0000H  1DB8H  LIN  1078
0000H  1DCFH  LIN  1080
0000H  1DDDH  LIN  1082
0000H  1DE5H  LIN  1084
0000H  1DEEH  LIN  1086
0000H  1DF3H  LIN  1089
0000H  1E1EH  LIN  1091
0000H  1E2BH  LIN  1094
0000H  1E2EH  LIN  1096
0000H  1E3EH  LIN  1098
0000H  1E4BH  LIN  1100
0000H  1E5CH  LIN  1103
0000H  1E60H  LIN  1105
0000H  1E66H  LIN  1107
0000H  1E7AH  LIN  1109
0000H  1E7DH  LIN  1111
0000H  1E8AH  LIN  1113
0000H  1EA7H  LIN  1115
0000H  1EABH  LIN  1117
0000H  1EB7H  LIN  1120
0000H  1EC0H  LIN  1122
0000H  1EC7H  LIN  1125
0000H  0102H  LIN  1128
0000H  0122H  LIN  1130
0000H  0138H  LIN  1132
0000H  014FH  LIN  1135
0000H  0159H  LIN  1138
0000H  016CH  LIN  1140
0000H  017CH  LIN  1142
0000H  018AH  LIN  1144
0000H  0197H  LIN  1147
0000H  01A1H  LIN  1149
0000H  01A9H  LIN  1152
0000H  01B9H  LIN  1154
0000H  01C7H  LIN  1157
0000H  01D2H  LIN  1159
0000H  01ECH  LIN  1161
0000H  01F4H  LIN  1164
0000H  020BH  LIN  1166
0000H  0220H  LIN  1168
0000H  0228H  LIN  1170
0000H  0234H  LIN  1172
0000H  023CH  LIN  1174
0000H  0248H  LIN  1176
0000H  0254H  LIN  1178
0000H  0260H  LIN  1180
0000H  0270H  LIN  1182
```

```
0000H  0272H  LIN   1183
0000H  027DH  LIN   1185
0000H  028EH  LIN   1187
0000H  029CH  LIN   1189
0000H  02B3H  LIN   1191
0000H  02C1H  LIN   1194
0000H  02CEH  LIN   1197
0000H  02D8H  LIN   1199
0000H  02E0H  LIN   1201
0000H  02E8H  LIN   1203
0000H  02F3H  LIN   1205
0000H  0300H  LIN   1208
0000H  0303H  LIN   1210
0000H  0310H  LIN   1213
0000H  031CH  LIN   1216
0000H  0328H  LIN   1218
0000H  0330H  LIN   1221
0000H  0349H  LIN   1223
0000H  0351H  LIN   1226
0000H  035BH  LIN   1229
0000H  0381H  LIN   1232
0000H  038CH  LIN   1234
0000H  039DH  LIN   1236
0000H  03AAH  LIN   1239
0000H  03BDH  LIN   1242
0000H  03C7H  LIN   1245
0000H  03CCH  LIN   1248
0000H  03D2H  LIN   1250
0000H  03DCH  LIN   1253
0000H  03E2H  LIN   1255
0000H  03EEH  LIN   1257
0000H  03F0H  LIN   1260
0000H  03FAH  LIN   1263
0000H  040AH  LIN   1265
0000H  0416H  LIN   1268
0000H  0420H  LIN   1270
0000H  0428H  LIN   1272
0000H  043AH  LIN   1274
0000H  0449H  LIN   1276
0000H  044EH  LIN   1278
0000H  045AH  LIN   1281
0000H  0460H  LIN   1283
0000H  0465H  LIN   1286
0000H  046CH  LIN   1289
0000H  0472H  LIN   1291
0000H  0486H  LIN   1293
0000H  049BH  LIN   1296
0000H  04A4H  LIN   1298
0000H  04B3H  LIN   1302
0000H  04C2H  LIN   1305
0000H  04D2H  LIN   1308
0000H  04E0H  LIN   1311
0000H  04EDH  LIN   1313
0000H  0501H  LIN   1318
0000H  050BH  LIN   1320
0000H  051FH  LIN   1322
0000H  0529H  LIN   1324
0000H  052CH  LIN   1326
0000H  0536H  LIN   1329
0000H  0549H  LIN   1332

0000H  0277H  LIN   1184
0000H  0287H  LIN   1186
0000H  0295H  LIN   1188
0000H  02AAH  LIN   1190
0000H  02BAH  LIN   1192
0000H  02C8H  LIN   1195
0000H  02D1H  LIN   1198
0000H  02E0H  LIN   1200
0000H  02E5H  LIN   1202
0000H  02EDH  LIN   1204
0000H  02FDH  LIN   1207
0000H  0303H  LIN   1209
0000H  030DH  LIN   1212
0000H  0315H  LIN   1214
0000H  0322H  LIN   1217
0000H  032EH  LIN   1219
0000H  033BH  LIN   1222
0000H  034EH  LIN   1225
0000H  0351H  LIN   1227
0000H  037CH  LIN   1231
0000H  0387H  LIN   1233
0000H  0398H  LIN   1235
0000H  03A0H  LIN   1238
0000H  03B1H  LIN   1240
0000H  03C4H  LIN   1244
0000H  03CAH  LIN   1246
0000H  03CFH  LIN   1249
0000H  03D9H  LIN   1252
0000H  03DFH  LIN   1254
0000H  03EBH  LIN   1256
0000H  03F0H  LIN   1259
0000H  03F7H  LIN   1262
0000H  0404H  LIN   1264
0000H  040DH  LIN   1266
0000H  041BH  LIN   1269
0000H  0423H  LIN   1271
0000H  0434H  LIN   1273
0000H  0440H  LIN   1275
0000H  044CH  LIN   1277
0000H  044EH  LIN   1280
0000H  045DH  LIN   1282
0000H  0463H  LIN   1285
0000H  0465H  LIN   1287
0000H  046FH  LIN   1290
0000H  0481H  LIN   1292
0000H  0488H  LIN   1294
0000H  04A1H  LIN   1297
0000H  04A6H  LIN   1300
0000H  04B9H  LIN   1303
0000H  04D2H  LIN   1306
0000H  04D9H  LIN   1310
0000H  04E6H  LIN   1312
0000H  04FEH  LIN   1315
0000H  0508H  LIN   1319
0000H  0512H  LIN   1321
0000H  0524H  LIN   1323
0000H  052CH  LIN   1325
0000H  0533H  LIN   1328
0000H  053BH  LIN   1331
0000H  054CH  LIN   1333
```

```
0000H  054CH  LIN  1334
0000H  055DH  LIN  1337
0000H  0560H  LIN  1339
0000H  056AH  LIN  1343
0000H  057DH  LIN  1345
0000H  058FH  LIN  1348
0000H  05A7H  LIN  1350
0000H  05B3H  LIN  1352
0000H  05C2H  LIN  1355
0000H  05C7H  LIN  1357
0000H  05D3H  LIN  1359
0000H  05DFH  LIN  1362
0000H  05F3H  LIN  1365
0000H  0605H  LIN  1368
0000H  060BH  LIN  1372
0000H  0615H  LIN  1375
0000H  061DH  LIN  1378
0000H  0627H  LIN  1381
0000H  062CH  LIN  1383
0000H  0636H  LIN  1386
0000H  0642H  LIN  1389
0000H  0647H  LIN  1392
0000H  0651H  LIN  1395
0000H  0657H  LIN  1397
0000H  0662H  LIN  1400
0000H  066EH  LIN  1402
0000H  067EH  LIN  1405
0000H  068AH  LIN  1407
0000H  068CH  LIN  1409
0000H  0696H  LIN  1412
0000H  0699H  LIN  1414
0000H  06A3H  LIN  1417
0000H  06A5H  LIN  1419
0000H  06AFH  LIN  1422
0000H  06B4H  LIN  1424
0000H  06C0H  LIN  1426
0000H  06CEH  LIN  1430
0000H  06D6H  LIN  1432
0000H  06E0H  LIN  1435
0000H  06E6H  LIN  1437
0000H  06E8H  LIN  1439
0000H  06F6H  LIN  1442
0000H  070CH  LIN  1445
0000H  071EH  LIN  1448
0000H  072BH  LIN  1450
0000H  0730H  LIN  1452
0000H  0748H  LIN  1454
0000H  074FH  LIN  1456
0000H  0764H  LIN  1460
0000H  076AH  LIN  1462
0000H  076FH  LIN  1465
0000H  078EH  LIN  1469
0000H  0799H  LIN  1471
0000H  07A2H  LIN  1473
0000H  07ACH  LIN  1475
0000H  07B6H  LIN  1477
0000H  07BDH  LIN  1479
0000H  07C3H  LIN  1481
0000H  07CDH  LIN  1484
0000H  07D7H  LIN  1486

0000H  0556H  LIN  1336
0000H  0560H  LIN  1338
0000H  0567H  LIN  1342
0000H  0578H  LIN  1344
0000H  0586H  LIN  1347
0000H  059DH  LIN  1349
0000H  05ACH  LIN  1351
0000H  05B8H  LIN  1354
0000H  05C5H  LIN  1356
0000H  05CCH  LIN  1358
0000H  05D5H  LIN  1360
0000H  05E6H  LIN  1363
0000H  05FEH  LIN  1366
0000H  0608H  LIN  1369
0000H  060EH  LIN  1373
0000H  0618H  LIN  1376
0000H  0624H  LIN  1380
0000H  062AH  LIN  1382
0000H  0633H  LIN  1385
0000H  063DH  LIN  1388
0000H  0645H  LIN  1391
0000H  064EH  LIN  1394
0000H  0654H  LIN  1396
0000H  065EH  LIN  1398
0000H  0669H  LIN  1401
0000H  0675H  LIN  1404
0000H  0684H  LIN  1406
0000H  068CH  LIN  1408
0000H  0693H  LIN  1411
0000H  0699H  LIN  1413
0000H  06A0H  LIN  1416
0000H  06A5H  LIN  1418
0000H  06ACH  LIN  1421
0000H  06B2H  LIN  1423
0000H  06BBH  LIN  1425
0000H  06C7H  LIN  1428
0000H  06D3H  LIN  1431
0000H  06D9H  LIN  1434
0000H  06E3H  LIN  1436
0000H  06E8H  LIN  1438
0000H  06EFH  LIN  1441
0000H  0705H  LIN  1443
0000H  0713H  LIN  1447
0000H  0724H  LIN  1449
0000H  072EH  LIN  1451
0000H  0741H  LIN  1453
0000H  074DH  LIN  1455
0000H  0756H  LIN  1458
0000H  0767H  LIN  1461
0000H  076DH  LIN  1464
0000H  0787H  LIN  1467
0000H  0793H  LIN  1470
0000H  079FH  LIN  1472
0000H  07A9H  LIN  1474
0000H  07B3H  LIN  1476
0000H  07B8H  LIN  1478
0000H  07C0H  LIN  1480
0000H  07CAH  LIN  1483
0000H  07D4H  LIN  1485
0000H  07D9H  LIN  1487
```

```
0000H  07D9H  LIN   1488
0000H  07E3H  LIN   1492
0000H  07ECH  LIN   1494
0000H  07F6H  LIN   1496
0000H  0801H  LIN   1498
0000H  081AH  LIN   1500
0000H  081FH  LIN   1502
0000H  0839H  LIN   1505
0000H  0851H  LIN   1507
0000H  0858H  LIN   1509
0000H  085DH  LIN   1511
0000H  0870H  LIN   1514
0000H  087AH  LIN   1516
0000H  0899H  LIN   1519
0000H  08A4H  LIN   1521
0000H  08AEH  LIN   1524
0000H  0888H  LIN   1526
0000H  08D2H  LIN   1529
0000H  08D8H  LIN   1531
0000H  08E0H  LIN   1533
0000H  08E9H  LIN   1535
0000H  08F1H  LIN   1537
0000H  08FFH  LIN   1541
0000H  0906H  LIN   1544
0000H  090CH  LIN   1546
0000H  0916H  LIN   1548
0000H  0931H  LIN   1550
0000H  0940H  LIN   1552
0000H  0945H  LIN   1555
0000H  094CH  LIN   1557
0000H  0958H  LIN   1560
0000H  096AH  LIN   1562
0000H  0976H  LIN   1565
0000H  0984H  LIN   1567
0000H  098DH  LIN   1571
0000H  099DH  LIN   1574
0000H  09B0H  LIN   1576
0000H  09C5H  LIN   1578
0000H  09CFH  LIN   1581
0000H  09E1H  LIN   1583
0000H  09F1H  LIN   1585
0000H  09F6H  LIN   1588
0000H  0A04H  LIN   1592
0000H  0A15H  LIN   1596
0000H  0A1AH  LIN   1598
0000H  0A24H  LIN   1601
0000H  0A27H  LIN   1605

0000H  07E0H  LIN   1491
0000H  07E6H  LIN   1493
0000H  07EFH  LIN   1495
0000H  07F9H  LIN   1497
0000H  080FH  LIN   1499
0000H  081DH  LIN   1501
0000H  0825H  LIN   1503
0000H  084CH  LIN   1506
0000H  0854H  LIN   1508
0000H  085DH  LIN   1510
0000H  0868H  LIN   1513
0000H  0877H  LIN   1515
0000H  0893H  LIN   1518
0000H  089EH  LIN   1520
0000H  08A7H  LIN   1522
0000H  08B1H  LIN   1525
0000H  08C8H  LIN   1527
0000H  08D5H  LIN   1530
0000H  08D8H  LIN   1532
0000H  08E6H  LIN   1534
0000H  08ECH  LIN   1536
0000H  08FDH  LIN   1539
0000H  08FFH  LIN   1542
0000H  0909H  LIN   1545
0000H  0913H  LIN   1547
0000H  0928H  LIN   1549
0000H  093CH  LIN   1551
0000H  0943H  LIN   1553
0000H  0945H  LIN   1556
0000H  0951H  LIN   1558
0000H  0965H  LIN   1561
0000H  0971H  LIN   1564
0000H  097DH  LIN   1566
0000H  098BH  LIN   1568
0000H  0998H  LIN   1572
0000H  09A2H  LIN   1575
0000H  09BEH  LIN   1577
0000H  09CFH  LIN   1579
0000H  09D2H  LIN   1582
0000H  09EAH  LIN   1584
0000H  09F4H  LIN   1587
0000H  09FDH  LIN   1590
0000H  0A0EH  LIN   1595
0000H  0A18H  LIN   1597
0000H  0A1AH  LIN   1599
0000H  0A27H  LIN   1604
```



```
MEMORY MAP OF MODULE SCD
READ FROM FILE ED.LNK
WRITTEN TO FILE ED.

MODULE START ADDRESS  PARAGRAPH = 0000H  OFFSET = 0102H
SEGMENT MAP

START     STOP      LENGTH ALIGN NAME            CLASS

00000H    01ED2H     1ED3H   G   CODE            CODE
10000H    10107H     0108H   G   DATS            DATA
10108H    10418H     0311H   W   DATA            DATA
```

```
1041AH    10481H    0068H   W   STACK           STACK
10482H    1058FH    013EH   W   CONST           CONST
105C0H    105C0H    0000H   G   ??SEG
105C0H    105C0H    0000H   W   MEMORY          MEMORY


GROUP MAP

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