```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE ED
OBJECT MODULE PLACED IN :F1:ED.OBJ
COMPILER INVOKED BY:  PLM80 :F1:ED.PLM DEBUG OPTIMIZE PAGEWIDTH(80)


   1          ED:
              DO;
                  /* MODIFIED FOR .PRL OPERATION MAY, 1979 */
                  /* MODIFIED FOR OPERATION WITH CP/M 2.0 AUGUST 1979 */
   2    1     DECLARE
                  /* JMP EDCOMMAND - 3 (TO ADDRESS LXI SP) */
                  EDJMP BYTE DATA(0C3H),
                  EDADR ADDRESS DATA(.EDCOMMAND-3);

   3    1     DECLARE
                  BDISK     BYTE    EXTERNAL, /* BOOT DISK 0004H */
                  MAXB      ADDRESS EXTERNAL, /* MAX BASE 0006H */
                  FCB (33)  BYTE    EXTERNAL, /* FCB 005CH */
                  BUFF (128)BYTE.   EXTERNAL, /* BUFFER 0080H */
                  SECTSHF   LITERALLY '7',    /* SHL(1,SECTSHF) = SECTSIZE */
                  SECTSIZE  LITERALLY '80H';  /* SECTOR SIZE */

   4    1     MON1: PROCEDURE(F,A) EXTERNAL;
   5    2         DECLARE F BYTE, A ADDRESS;
   6    2         END MON1;

   7    1     MON2: PROCEDURE(F,A) BYTE EXTERNAL;
   8    2         DECLARE F BYTE, A ADDRESS;
   9    2         END MON2;

  10    1     BOOT: PROCEDURE EXTERNAL;
                  /* SYSTEM REBOOT */
  11    2         END BOOT;

                /* 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
                          DIGITAL RESEARCH
                          BOX 579 PACIFIC GROVE
                          CALIFORNIA 93950
                  */
  12    1     DECLARE COPYRIGHT(*) BYTE DATA
                  (' COPYRIGHT (C) 1979, DIGITAL RESEARCH ');

                /* 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 STRIN
```

```
-   G,
                    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>.LIB UNTIL <ENDFILE> AN
-   D
                    INSERT INTO TEXT
        S           SEARCH FOR FIRST STRING, REPLACE BY SECOND STRIN
-   G
        T           TYPE LINES
        U           TRANSLATE TO UPPER CASE (-U CHANGES TO NO TRANSL
-   ATE)
        W           WRITE LINES OF TEXT TO FILE
        X           TRANSFER (XFER) LINES TO TEMP FILE
        Z           SLEEP FOR 1/2 SECOND (USED IN MACROS TO STOP DIS
-   PLAY)
        <CR>        MOVE UP OR DOWN AND PRINT ONE LINE
```



```
   IN GENERAL, THE EDITOR ACCEPTS SINGLE LETTER COMMANDS WITH OPT
-   IONAL
INTEGER VALUES PRECEDING THE COMMAND.  THE EDITOR ACCEPTS BOTH UPP
-   ER AND LOWER
CASE COMMANDS AND VALUES, AND PERFORMS TRANSLATION TO UPPER CASE U
-   NDER THE FOL-
LOWING CONDITIONS.  IF THE COMMAND IS TYPED IN UPPER CASE, THEN TH
-   E 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 E
-   CHOED 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).  I
-   F YOU ARE
OPERATING WITH AN UPPER CASE ONLY TERMINAL, THEN OPERATION IS AUTO
-   MATIC.
SIMILARLY, IF YOU ARE OPERATING WITH A LOWER CASE TERMINAL, AND TR
-   ANSLATION
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 VALU
-   E
       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>0TT<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 COMMAND R MUST BE FOLLOWED BY A FILE NAME (WITH ASSUMED 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 0T PRINTS THE FIRST PART OF THE CURRENT LINE, AND THE COMMAND 0L MOVES THE REFERENCE TO THE BEGINNING OF THE CURRENT LINE. THE COMMAND 0P PRINTS THE CURRENT PAGE ONLY, WHILE THE COMMAND 0Z 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> I
- S
<ENDFILE> OR <CR>. THE COMMANDS C1 ... CN FOLLOWING THE M A
- RE
EXECUTED N TIMES, STARTING AT THE CURRENT POSITION IN THE BUF
- FER.
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 CHARACTE
- RS:

SYMBOL	ERROR CONDITION
GREATER	FREE MEMORY IS EXHAUSTED - ANY COMMAND CAN BE ISS - UED WHICH DOES NOT INCREASE MEMORY REQUIREMENTS.
QUESTION	UNRECOGNIZED COMMAND OR ILLEGAL NUMERIC FIELD
POUND	CANNOT APPLY THE COMMAND THE NUMBER OF TIMES SPEC - FIED (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.
*/

```
13  1       DECLARE LIT LITERALLY 'LITERALLY',
                DCL LIT 'DECLARE',
                PROC LIT 'PROCEDURE',
                ADDR LIT 'ADDRESS',
                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 */
```

```
                ENDFILE LIT '1AH';    /* CP/M END OF FILE */

14   1      DECLARE
                MAX ADDRESS,                    /* .MEMORY(MAX)=0 (END) */
                MAXM ADDRESS,                   /* MINUS 1 */
                HMAX ADDRESS;                   /* = MAX/2 */

15   1      DECLARE
                RO LITERALLY '9',  /* R/O FILE INDICATOR */
                SY LITERALLY '10', /* SYSTEM FILE ATTRIBUTE */
                EX LITERALLY '12', /* EXTENT NUMBER POSITION */
                UB LITERALLY '13', /* UNFILLED BYTES */
                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),
                XFCBE BYTE AT(.XFCB(EX)),  /* XFCB EXTENT */
                XFCBM BYTE AT(.XFCB(MD)),  /* MODULE NUMBER */
                XFCBR BYTE AT(.XFCB(NR)),  /* XFCB RECORD # */
                XBUFF (SECTSIZE) BYTE, /* XFER BUFFER */
                XBP BYTE,              /* XFER POINTER */
                XFERON BYTE,           /* TRUE IF XFER ACTIVE */

                NBUF BYTE,                  /* NUMBER OF BUFFERS */
                BUFFLENGTH ADDRESS,         /* NBUF * SECTSIZE */
                SFCB (FS) BYTE AT(.FCB),    /* SOURCE FCB = DEFAULT FCB */
                SBUFFADR ADDRESS,           /* SOURCE BUFFER ADDRESS */
                SBUFF BASED SBUFFADR (128) BYTE, /* SOURCE BUFFER */

                DFCB (FS) BYTE,           /* DEST FILE CONTROL BLOCK */
                DFUB BYTE AT(.DFCB(UB)),  /* UNFILLED BYTES IN LAST RECORD */
                DBUFFADR ADDRESS,         /* DESTINATION BUFFER ADDRESS */
                DBUFF BASED DBUFFADR (128) BYTE, /* DEST BUFFER */
                NSOURCE ADDRESS,          /* NEXT SOURCE CHARACTER */
                NDEST   ADDRESS;          /* NEXT DESTINATION CHAR */

16   1      DECLARE SDISK BYTE,     /* SOURCE FILE DISK */
                DDISK BYTE;          /* DESTINATION FILE DISK */

             /* IO SECTION */

17   1      READCHAR: PROCEDURE BYTE; RETURN MON2(1,0);
19   2          END READCHAR;

20   1      DECLARE TRUE LITERALLY '1', FALSE LITERALLY '0',
                FOREVER LITERALLY 'WHILE TRUE',
                CR LITERALLY '13',
                LF LITERALLY '10',
                WHAT LITERALLY '63';

21   1      DECLARE
               PRINTSUPPRESS BYTE: /* TRUE IF PRINT SUPPRESSED */
```

```
22   1      PRINTCHAR: PROCEDURE(CHAR);
23   2          DECLARE CHAR BYTE;
24   2          IF PRINTSUPPRESS THEN RETURN;
26   2          CALL MON1(2,CHAR);
27   2          END PRINTCHAR;

28   1      DECLARE
               COLUMN  BYTE, /* CONSOLE COLUMN POSITION */
               SCOLUMN BYTE, /* STARTING COLUMN IN "I" MODE */
               TCOLUMN BYTE, /* TEMP DURING BACKSPACE */
               QCOLUMN BYTE; /* TEMP DURING BACKSPACE */

29   1      TTYCHAR: PROCEDURE(CHAR);
30   2          DECLARE CHAR BYTE;
31   2          IF CHAR >= ' ' THEN COLUMN = COLUMN + 1;
33   2          IF CHAR = LF THEN COLUMN = 0;
35   2          CALL PRINTCHAR(CHAR);
36   2          END TTYCHAR;

37   1      BACKSPACE: PROCEDURE;
               /* MOVE BACK ONE POSITION */
38   2          IF COLUMN = 0 THEN RETURN;
40   2          CALL TTYCHAR(CTLH); /* COLUMN = COLUMN - 1 */
41   2          CALL TTYCHAR(' '); /* COLUMN = COLUMN + 1 */
42   2          CALL TTYCHAR(CTLH); /* COLUMN = COLUMN - 1 */
43   2          COLUMN = COLUMN - 2;
44   2          END BACKSPACE;

45   1      PRINTABS: PROCEDURE(CHAR);
46   2          DECLARE (CHAR,I,J) BYTE;
47   2          I = CHAR = TAB AND 7 - (COLUMN AND 7);
48   2          IF CHAR = TAB THEN CHAR = ' ';
50   2              DO J = 0 TO I;
51   3              CALL TTYCHAR(CHAR);
52   3              END;
53   2          END PRINTABS;

54   1      GRAPHIC: PROCEDURE(C) BYTE;
55   2          DECLARE C BYTE;
               /* RETURN TRUE IF GRAPHIC CHARACTER */
56   2          IF C >= ' ' THEN RETURN TRUE;
58   2          RETURN C = CR OR C = LF OR C = TAB;
59   2          END GRAPHIC;

60   1      PRINTC: PROCEDURE(C);
61   2          DECLARE C BYTE;
62   2          IF NOT GRAPHIC(C) THEN
63   2              DO; CALL PRINTABS('^');
65   3              C = C + '@';
66   3              END;
67   2          CALL PRINTABS(C);
68   2          END PRINTC;

69   1      CRLF: PROCEDURE;
70   2          CALL PRINTC(CR); CALL PRINTC(LF);
72   2          END CRLF;
```

```
 73   1      PRINTM: PROCEDURE(A);
 74   2          DECLARE A ADDRESS;
 75   2          CALL MON1(9,A);
 76   2          END PRINTM;

 77   1      PRINT: PROCEDURE(A);
 78   2          DECLARE A ADDRESS;
 79   2          CALL CRLF;
 80   2          CALL PRINTM(A);
 81   2          END PRINT;

 82   1      READ: PROCEDURE(A);
 83   2          DECLARE A ADDRESS;
 84   2          CALL MON1(10,A);
 85   2          END READ;

 86   1      DECLARE DCNT BYTE;


 87   1      OPEN: PROCEDURE(FCB);
 88   2           DECLARE FCB ADDRESS;
 89   2           DCNT = MON2(15,FCB);
 90   2           END OPEN;

 91   1      CLOSE: PROCEDURE(FCB);
 92   2          DECLARE FCB ADDRESS;
 93   2          DCNT = MON2(16,FCB);
 94   2          END CLOSE;

 95   1      SEARCH: PROCEDURE(FCB);
 96   2          DECLARE FCB ADDRESS;
 97   2          DCNT = MON2(17,FCB);
 98   2          END SEARCH;

 99   1      DELETE: PROCEDURE(FCB);
100   2          DECLARE FCB ADDRESS;
101   2          CALL MON1(19,FCB);
102   2          END DELETE;

103   1      DISKREAD: PROCEDURE(FCB) BYTE;
104   2          DECLARE FCB ADDRESS;
105   2          RETURN MON2(20,FCB);
106   2          END DISKREAD;

107   1      DISKWRITE: PROCEDURE(FCB) BYTE;
108   2          DECLARE FCB ADDRESS;
109   2          RETURN MON2(21,FCB);
110   2          END DISKWRITE;

111   1      MAKE: PROCEDURE(FCB);
112   2          DECLARE FCB ADDRESS;
113   2          DCNT = MON2(22,FCB);
114   2          END MAKE;

115   1      RENAME: PROCEDURE(FCB);
116   2          DECLARE FCB ADDRESS;
```

```
117   2          CALL MON1(23,FCB);
118   2          END RENAME;

119   1      DECLARE (MAXLEN,COMLEN) BYTE, COMBUFF(128) BYTE,
                 (TCBP,CBP) BYTE;

120   1      READCOM: PROCEDURE;
121   2          MAXLEN = 128; CALL READ(.MAXLEN);
123   2          END READCOM;

124   1      BREAK$KEY: PROCEDURE BYTE;
125   2          IF MON2(11,0) THEN
126   2              DO; /* CLEAR CHAR */
127   3              CALL MON1(1,0); RETURN TRUE;
129   3              END;
130   2          RETURN FALSE;
131   2          END BREAK$KEY;

132   1      CSELECT: PROCEDURE BYTE;
                 /* RETURN CURRENT DRIVE NUMBER */
133   2          RETURN MON2(25,0);
134   2          END CSELECT;

135   1      SELECT: PROCEDURE(DISK);
136   2          DECLARE DISK BYTE;
                 /* SET DRIVE NUMBER */
137   2          CALL MON1(14,DISK);
138   2          END SELECT;

139   1      SETDMA: PROCEDURE(A);
140   2          DECLARE A ADDRESS;
                 /* SET DMA ADDRESS */
141   2          CALL MON1(26,A);
142   2          END SETDMA;

143   1      REBOOT: PROCEDURE;
144   2          IF XFERON THEN CALL DELETE(.XFCB);
146   2          CALL BOOT;
147   2          END REBOOT;

148   1      DECLARE /* LINE COUNTERS */
                 BASELINE ADDRESS,       /* CURRENT LINE */
                 RELLINE  ADDRESS,       /* RELATIVE LINE IN TYPEOUT */
                 LINESET BYTE;           /* TRUE IF LINE #'S PRINTED */

               /*  INPUT / OUTPUT BUFFERING ROUTINES */

               /* THE PL/M BUILT-IN PROCEDURE "MOVE" IS 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;
                 */

149   1      ABORT: PROCEDURE(A);
```

```
150   2          DECLARE A ADDRESS;
151   2          CALL PRINT(A);
152   2          CALL CRLF;
153   2          CALL REBOOT;
154   2          END ABORT;

155   1      FERR: PROCEDURE;
156   2          CALL CLOSE(.DFCB); /* ATTEMPT TO CLOSE FILE FOR LATER RECOVERY
                  */
157   2          CALL ABORT (.('DISK OR DIRECTORY FULL$'));
158   2          END FERR;

159   1      SETTYPE: PROCEDURE(A);
160   2          DECLARE A ADDRESS;
161   2          CALL MOVE(3,A,.DFCB+9);
162   2          END SETTYPE;

163   1      SETUP: PROCEDURE;
164   2          NSOURCE = BUFFLENGTH; NDEST = 0;
166   2          SFCB(EX), SFCB(MD), SFCB(NR) = 0;
                  /* REEL AND RECORD ZEROED */
                  /* COPY NAME TO DESTINATION FCB */
167   2          CALL MOVE(33,.FCB,.DFCB);
                  /* 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 */
168   2          IF SDISK <> DDISK THEN
169   2              DO; CALL SELECT(DDISK);
171   3              CALL SEARCH(.FCB);
172   3              IF DCNT <> 255 THEN /* SOURCE FILE PRESENT ON DEST DISK */
173   3                  CALL ABORT(.('FILE EXISTS, ERASE IT$'));
174   3              END;
175   2          CALL SELECT(SDISK);
176   2          CALL OPEN(.FCB);
177   2          IF DCNT = 255 THEN
178   2              DO; CALL MAKE(.FCB);
180   3              IF DCNT = 255 THEN CALL FERR;
182   3              CALL PRINT(.('NEW FILE$'));
183   3              CALL CRLF;
184   3              END; ELSE
185   2          IF ROL(FCB(RO),1) THEN
186   2              DO;
187   3              CALL PRINT(.('** FILE IS READ/ONLY **$'));
188   3              CALL CRLF;
189   3              END; ELSE
190   2          IF ROL(FCB(SY),1) THEN
191   2              CALL ABORT(.('"SYSTEM" FILE NOT ACCESSIBLE$'));
                  CALL SETTYPE(.('BAK'));
193   2          CALL DELETE(.DFCB);
194   2          IF SDISK <> DDISK THEN
195   2              DO; /* REMOVE BAK FILES FROM DESTINATION DISK */
196   3              CALL SELECT(DDISK);
197   3              CALL DELETE(.DFCB);
```

```
198   3             END;
199   2         CALL SETTYPE(.('$$$'));
200   2         CALL DELETE(.DFCB);
201   2         CALL MAKE(.DFCB);
202   2         DFCB(32) = 0;  /* NEXT RECORD IS ZERO */
203   2         IF DCNT = 255 THEN CALL FERR;
                /* THE TEMP FILE IS NOW CREATED */
205   2         BASELINE = 1; /* START WITH LINE 1 */
206   2         END SETUP;

207   1     XCLEAR: PROCEDURE;
                /* CLEAR THE XFER FILE PARAMETERS */
208   2         XFERON, XFCBE, XFCBR, XBP = 0;
209   2         END XCLEAR;

210   1     SETXDMA: PROCEDURE;
211   2         CALL SELECT(SDISK);
212   2         CALL SETDMA(.XBUFF);
213   2         END SETXDMA;

214   1     FILLSOURCE: PROCEDURE;
215   2         DECLARE I BYTE;
216   2         ZN: PROCEDURE;
217   3             NSOURCE = 0;
218   3             END ZN;

219   2          CALL ZN;
220   2         CALL SELECT(SDISK);
221   2             DO I = 0 TO NBUF;
222   3             CALL SETDMA(SBUFFADR+NSOURCE);
223   3             IF (DCNT := DISKREAD(.FCB)) <> 0 THEN
224   3                 DO; IF DCNT > 1 THEN CALL FERR;
227   4                 SBUFF(NSOURCE) = ENDFILE;
228   4                 I = NBUF;
229   4                 END;
                    ELSE
230   3                 NSOURCE = NSOURCE + SECTSIZE;
231   3             END;
232   2         CALL ZN;
233   2         END FILLSOURCE;

234   1     GETSOURCE: PROCEDURE BYTE;
235   2         DECLARE B BYTE;
236   2         IF NSOURCE >= BUFFLENGTH THEN CALL FILLSOURCE;
238   2         IF (B := SBUFF(NSOURCE)) <> ENDFILE THEN
239   2             NSOURCE = NSOURCE + 1;
240   2         RETURN B;
241   2         END GETSOURCE;

242   1     WRITEDEST: PROCEDURE;
                /* WRITE OUTPUT BUFFER UP TO (NOT INCLUDING) NDEST.
                LOW 7 BITS OF NDEST ARE ZERO */
243   2         DECLARE (I,N) BYTE;
244   2         ZN: PROCEDURE;
245   3             NDEST = 0;
246   3             END ZN;
```

```
247   2          CALL SELECT(DDISK);
248   2          IF LOW((N := SHR(NDEST,SECTSHF) - 1)) = 255 THEN RETURN;
250   2          CALL ZN;
251   2              DO I = 0 TO N;
252   3              CALL SETDMA(DBUFFADR+NDEST);
253   3              IF DISKWRITE(.DFCB) <> 0 THEN CALL FERR;
255   3              NDEST = NDEST + SECTSIZE;
256   3              END;
257   2          CALL ZN;
258   2          END WRITEDEST;

259   1      PUTDEST: PROCEDURE(B);
260   2          DECLARE B BYTE;
261   2          IF NDEST >= BUFFLENGTH THEN CALL WRITEDEST;
263   2          DBUFF(NDEST) = B;
264   2          NDEST = NDEST + 1;
265   2          END PUTDEST;

266   1      PUTXFER: PROCEDURE(C);
267   2          DECLARE C BYTE;
                  /* WRITE C TO XFER FILE */
268   2          IF XBP >= SECTSIZE THEN /* BUFFER OVERFLOW */
269   2              DO; CALL SETXDMA;
271   3              IF DISKWRITE(.XFCB) <> 0 THEN CALL FERR;
273   3              XBP = 0;
274   3              END;
275   2          XBUFF(XBP) = C; XBP = XBP + 1;
277   2          END PUTXFER;

278   1      FINIS: PROCEDURE;
279   2          MOVEUP: PROCEDURE;
280   3          CALL MOVE(16,.DFCB,.DFCB+16);
281   3          END MOVEUP;

                  /* CLEAR OUTPUT */
282   2          DFUB = 0 ; /* SET UNFILLED BYTES - USED FOR ISIS-II COMPATIBIL
          -    ITY */
283   2                  DO WHILE (LOW(NDEST) AND 7FH) <> 0;
284   3                  DFUB = DFUB + 1; /* COUNTS UNFILLED BYTES IN LAST RECORD *
          -    /
285   3                  CALL PUTDEST(ENDFILE);
286   3                  END;
287   2          CALL WRITEDEST;

288   2          CALL CLOSE(.DFCB);
289   2          IF DCNT = 255 THEN CALL FERR;
                  /* RENAME OLD FILE TO BAK */
291   2          CALL SETTYPE(.('BAK')); CALL MOVEUP;
293   2          CALL SELECT(SDISK);
294   2          CALL MOVE(16,.FCB,.DFCB);
295   2          CALL RENAME(.DFCB);
296   2          CALL MOVEUP;
                  /* RENAME $$$ TO OLD NAME */
297   2          CALL SETTYPE(.('$$$'));
298   2          CALL SELECT(DDISK);
299   2          CALL RENAME(.DFCB);
300   2          END FINIS;
```

```
301   1       DECLARE
                  LPP LIT '23',           /* LINES PER PAGE */
                  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*/

302   1       DCL MACRO(MACSIZE) BYTE,
                  SCRATCH(SCRSIZE) BYTE,     /* SCRATCH BUFFER FOR F,N,S */
                  (WBP, WBE, WBJ)  BYTE,     /* END OF F STRIN , S STRING, J STR
            -   ING */
                  (FLAG, MP, MI, XP) BYTE,
                  MT COMSIZE;

303   1       DCL (START, RESTART, OVERCOUNT, OVERFLOW, RESET, BADCOM) LABEL;

304   1       DCL INSERTING BYTE,    /* TRUE IF INSERTING CHARACTERS */
                  READBUFF  BYTE;    /* TRUE IF END OF READ BUFFER */

305   1       DECLARE
                  EOS LITERALLY '0FFH';



306   1       PRINTNMAC: PROCEDURE(CHAR);
307   2           DECLARE CHAR BYTE;
                  /* PRINT IF NOT IN MACRO EXPANSION */
308   2           IF MP <> 0 THEN RETURN;
310   2           CALL PRINTC(CHAR);
311   2           END PRINTNMAC;



312   1       DECLARE TRANSLATE BYTE, /* TRUE IF TRANSLATION TO UPPER CASE */
                  UPPER BYTE;        /* TRUE IF GLOBALLY TRANLATING TO UC */

313   1       LOWERCASE: PROCEDURE(C) BYTE;
314   2           DECLARE C BYTE;
                  /* RETURN TRUE IF LOWER CASE ALPHABETIC */
315   2           RETURN C >= LCA AND C <= LCZ;
316   2           END LOWERCASE;

317   1       UCASE: PROCEDURE(C) BYTE;
318   2           DECLARE C BYTE;
                  /* TRANSLATE C TO UPPER CASE */
319   2           IF LOWERCASE(C) THEN RETURN C AND 5FH;
321   2           RETURN C;
322   2           END UCASE;

323   1       UTRAN: PROCEDURE(C) BYTE;
324   2           DECLARE C BYTE;
```

```
                    /* TRANSLATE TO UPPER CASE IF ALPHABETIC LOWER AND TRANSLATE *
                - /
 325   2              IF TRANSLATE THEN RETURN UCASE(C);
 327   2              RETURN C;
 328   2              END UTRAN;

 329   1          PRINTVALUE: PROCEDURE(V);
                        /* PRINT THE LINE VALUE V */
 330   2              DECLARE (D,ZERO) BYTE,
                            (K,V) ADDRESS;
 331   2              K = 10000;
 332   2              ZERO = FALSE;
 333   2                  DO WHILE K <> 0;
 334   3                  D = LOW(V/K); V = V MOD K;
 336   3                  K = K / 10;
 337   3                  IF ZERO OR  D <> 0 THEN
 338   3                      DO; ZERO = TRUE;
 340   4                      CALL PRINTC('0'+D);
 341   4                      END; ELSE
 342   3                  CALL PRINTC(' ');
 343   3                  END;
 344   2              END PRINTVALUE;

 345   1          PRINTLINE: PROCEDURE(V);
 346   2              DECLARE V ADDRESS;
 347   2              IF NOT LINESET THEN RETURN;
 349   2              CALL PRINTVALUE(V);
 350   2              CALL PRINTC(':');
 351   2              CALL PRINTC(' ');
 352   2              IF INSERTING THEN CALL PRINTC(' '); ELSE
 354   2                  CALL PRINTC('*');
 355   2              END PRINTLINE;


 356   1          PRINTBASE: PROCEDURE;
 357   2              CALL PRINTLINE(BASELINE);
 358   2              END PRINTBASE;

 359   1          PRINTNMBASE: PROCEDURE;
 360   2              IF MP <> 0 THEN RETURN;
 362   2              CALL PRINTBASE;
 363   2              END PRINTNMBASE;

 364   1          READC: PROCEDURE BYTE;
                        /* MAY BE MACRO EXPANSION */
 365   2              IF MP > 0 THEN
 366   2                  DO;
 367   3                  IF BREAK$KEY THEN GO TO OVERCOUNT;
 369   3                  IF XP >= MP THEN
 370   3                      DO; /* START AGAIN */
 371   4                      IF MT <> 0 THEN
 372   4                          DO; IF (MT:=MT-1) = 0 THEN
 374   5                              GO TO OVERCOUNT;
 375   5                          END;
 376   4                      XP = 0;
 377   4                      END;
 378   3                  RETURN UTRAN(MACRO((XP := XP + 1) - 1));
```

```
379   3             END;
380   2         IF INSERTING THEN RETURN UTRAN(READCHAR);

                /* GET COMMAND LINE */
382   2         IF READBUFF THEN
383   2             DO; READBUFF = FALSE;
385   3             IF LINESET AND COLUMN = 0 THEN
386   3                 DO;
387   4                 IF BACK >= MAXM THEN
388   4                     CALL PRINTLINE(0); ELSE
389   4                     CALL PRINTBASE;
390   4                 END; ELSE
391   3             CALL PRINTC('*');
392   3             CALL READCOM; CBP = 0;
394   3             CALL PRINTC(LF);
395   3             COLUMN = 0;
396   3             END;
397   2         IF (READBUFF := CBP = COMLEN ) THEN COMBUFF(CBP) = CR;
399   2         RETURN UTRAN(COMBUFF((CBP := CBP +1) -1));
400   2         END READC;

401   1       SETRDMA: PROCEDURE;
                /* SET READ LIB DMA ADDRESS */
402   2         CALL SELECT(SDISK);
403   2         CALL SETDMA(.BUFF);
404   2         END SETRDMA;

405   1       READFILE: PROCEDURE BYTE;
406   2         IF RBP >= SECTSIZE THEN
407   2             DO; CALL SETRDMA;
409   3             IF DISKREAD(.RFCB) <> 0 THEN RETURN ENDFILE;
411   3             RBP = 0;
412   3             END;
413   2         RETURN UTRAN(BUFF((RBP := RBP + 1) - 1));
414   2         END READFILE;

415   1       DCL (DISTANCE, TDIST) COMSIZE,
                (DIRECTION, CHAR) BYTE,
                ( FRONT, BACK, FIRST, LAST) ADDR;

416   1       SETFF: PROCEDURE;
417   2         DISTANCE = OFFFFH;
418   2         END SETFF;

419   1       DISTZERO: PROCEDURE BYTE;
                /* RETURN TRUE IF DISTANCE IS ZERO */
420   2         RETURN DISTANCE = 0;
421   2         END DISTZERO;

422   1       ZERODIST: PROCEDURE;
423   2         DISTANCE = 0;
424   2         END ZERODIST;

425   1       DISTNZERO: PROCEDURE BYTE;
                /* CHECK FOR ZERO DISTANCE AND DECREMENT */
426   2         IF NOT DISTZERO THEN
427   2             DO; DISTANCE = DISTANCE - 1;
```

```
429   3             RETURN TRUE;
430   3             END;
431   2         RETURN FALSE;
432   2         END DISTNZERO;

433   1     SETLIMITS: PROC;
434   2         DCL (I,K,L,M) ADDR, (MIDDLE,LOOPING) BYTE;
435   2         RELLINE = 1; /* RELATIVE LINE COUNT */
436   2         IF DIRECTION = BACKWARD THEN
437   2             DO; DISTANCE = DISTANCE+1; I = FRONT; L = 0; K = 0FFFFH;
442   3             END; ELSE
443   2             DO; I = BACK; L = MAXM; K = 1;
447   3             END;

448   2         LOOPING = TRUE;
449   2             DO WHILE LOOPING;
450   3                 DO WHILE (MIDDLE := I <> L) AND
                              MEMORY(M:=I+K) <> LF;
451   4                 I = M;
452   4                 END;
453   3             RELLINE = RELLINE - 1;
454   3             LOOPING = (DISTANCE := DISTANCE - 1) <> 0;
455   3             IF NOT MIDDLE THEN
456   3                 DO; LOOPING = FALSE;
458   4                 I = I - K;
459   4                 END; ELSE
460   3             IF LOOPING THEN I = M;
                      END;

463   2         IF DIRECTION = BACKWARD THEN
464   2             DO; FIRST = I; LAST = FRONT - 1;
467   3             END; ELSE
468   2             DO; FIRST = BACK + 1; LAST = I + 1;
471   3             END;
472   2         END SETLIMITS;

473   1     INCFRONT: PROC; FRONT = FRONT + 1;
475   2         END INCFRONT;
476   1     INCBACK: PROCEDURE; BACK = BACK + 1;
478   2         END INCBACK;
479   1     DECFRONT: PROC; FRONT = FRONT - 1;
481   2         END DECFRONT;
482   1     DECBACK: PROC; BACK = BACK - 1;
484   2         END DECBACK;
485   1     INCBASE: PROCEDURE;
486   2         BASELINE = BASELINE + 1;
487   2         END INCBASE;


488   1     MEM$MOVE: PROC(MOVEFLAG);
489   2         DECLARE (MOVEFLAG,C) BYTE;
                  /* MOVE IF MOVEFLAG IS TRUE */
490   2         IF DIRECTION = FORWARD THEN
491   2             DO WHILE BACK < LAST; CALL INCBACK;
493   3             IF MOVEFLAG THEN
494   3                 DO;
495   4                 IF (C := MEMORY(BACK)) = LF THEN CALL INCBASE;
```

```
497   4                 MEMORY(FRONT) = C; CALL INCFRONT;
499   4                 END;
500   3             END; ELSE
501   2             DO WHILE FRONT > FIRST; CALL DECFRONT;
503   3             IF (C := MEMORY(FRONT)) = LF THEN BASELINE = BASELINE - 1;
505   3             IF MOVEFLAG THEN
506   3                 DO; MEMORY(BACK) = C; CALL DECBACK;
509   4                 END;
510   3             E D;
511   2         END MEM$MOVE;

512   1     MOVER: PROC;
513   2         CALL MEM$MOVE(TRUE);
514   2         END MOVER;

515   1     SETPTRS: PROC;
516   2         CALL MEM$MOVE(FALSE);
517   2         END SETPTRS;

518   1     MOVELINES: PROC;
519   2         CALL SETLIMITS;
520   2         CALL MOVER;
521   2         END MOVELINES;

522   1     SETCLIMITS: PROC;
523   2         IF DIRECTION = BACKWARD THEN
524   2             DO; LAST = BACK;
526   3             IF DISTANCE > FRONT THEN
527   3                 FIRST = 1; ELSE FIRST = FRONT - DISTANCE;
529   3             END; ELSE
530   2             DO; FIRST = FRONT;
532   3             IF DISTANCE >= MAX - BACK THEN
533   3                 LAST = MAXM; ELSE LAST = BACK + DISTANCE;
535   3             END;
536   2         END SETCLIMITS;

537   1     READLINE: PROCEDURE;
538   2         DECLARE B BYTE;
                  /* READ ANOTHER LINE OF INPUT */
539   2         CTRAN: PROCEDURE(B) BYTE;
540   3             DECLARE B BYTE;
                      /* CONDITIONALLY TRANSLATE TO UPPER CASE ON INPUT */
541   3             IF UPPER THEN RETURN UTRAN(B);
543   3             RETURN B;
544   3             END CTRAN;
545   2         DO FOREVER;
546   3         IF FRONT >= BACK THEN GO TO OVERFLOW;
548   3         IF (B := CTRAN(GETSOURCE)) = ENDFILE THEN
549   3             DO; CALL ZERODIST; RETURN;
552   4             END;
553   3         MEMORY(FRONT) = B;
554   3         CALL INCFRONT;
555   3         IF B = LF THEN
556   3             DO; CALL INCBASE;
558   4             RETURN;
559   4             END;
560   3         END;
```

```
561   2          END READLINE;

562   1      WRITELINE: PROCEDURE;
                 /* WRITE ONE LINE OUT */
563   2          DECLARE B BYTE;
564   2              DO FOREVER;
565   3              IF BACK >= MAXM THEN /* EMPTY */
566   3                  DO; CALL ZERODIST; RETURN;
569   4                  EN ;
570   3              CALL INCBACK;
571   3              CALL PUTDEST(B:=MEMORY(BACK));
572   3              IF B = LF THEN
573   3                  DO; CALL INCBASE;
575   4                  RETURN;
576   4                  EN ;
577   3              END;
578   2          END WRITELINE;

579   1      WRHALF: PROCEDURE;
                  /* WRITE LINES UNTIL AT LEAST HALF THE BUFFER IS EMPTY */
580   2          CALL SETFF;
581   2              DO WHILE DISTNZERO;
582   3              IF HMAX >= (MAXM - BACK) THEN CALL ZERODIST; ELSE
584   3              CALL WRITELINE;
585   3              END;
586   2          END WRHALF;

587   1      WRITEOUT: PROCEDURE;
                 /* WRITE LINES DETERMINED BY 'DISTANCE',
                 CALLED FROM W AND E COMMANDS */
588   2          DIRECTION = BACKWARD; FIRST = 1; LAST = BACK;
591   2          CALL MOVER;
592   2          IF DISTZERO THEN CALL WRHALF;
                 /* DISTANCE = 0 IF CALL WRHALF */
594   2              DO WHILE DISTNZERO;
595   3              CALL WRITELINE;
596   3              END;
597   2          IF BACK < LAST THEN
598   2              DO; DIRECTION = FORWARD; CALL MOVER;
601   3              END;
602   2          END WRITEOUT;

603   1      CLEARMEM: PROCEDURE;
                 /* CLEAR MEMORY BUFFER */
604   2          CALL SETFF;
605   2          CALL WRITEOUT;
606   2          END CLEARMEM;

607   1      TERMINATE: PROCEDURE;
                 /* CLEAR BUFFERS */
608   2          CALL CLEARMEM;
609   2              DO WHILE (CHAR := GETSOURCE) <> ENDFILE;
610   3              CALL PUTDEST(CHAR);
611   3              END;
612   2          CALL FINIS;
613   2          END TERMINATE;
```

```
614   1       INSERT: PROCEDURE;
                  /* INSERT CHAR INTO MEMORY BUFFER */
615   2           IF FRONT = BACK THEN GO TO OVERFLOW;
617   2           MEMORY(FRONT) = CHAR; CALL INCFRONT;
619   2           IF CHAR = LF THEN CALL INCBASE;
621   2           END INSERT;

622   1       SCANNING: PROCEDURE BYTE;
                  /* READ A CHARACTER AND CHECK FOR ENDFILE OR CR */
623   2           RETURN NOT ((CHAR := READC) = ENDFILE OR
                                (CHAR = CR AND NOT INSERTING));
624   2           END SCANNING;

625   1       COLLECT: PROCEDURE;
                  /* READ COMMAND BUFFER AND INSERT CHARACTERS INTO
                  SCRATCH 'TIL NEXT CONTROL-Z OR CR FOR  FIND, NEXT, JUXT, OR
                  SUBSTITUTE COMMANDS - FILL AT WBE AND INCREMENT WBE SO IT
                  ADDRESSES NEXT EMPTY POSITION OF SCRATCH */
626   2           SETSCR: PROCEDURE;
627   3               SCRATCH(WBE) = CHAR;
628   3               IF (WBE := WBE + 1) >= SCRSIZE THEN GO TO OVERFLOW;
630   3               END SETSCR;
631   2               DO WHILE SCANNING;
632   3               IF CHAR = CTLL THEN
633   3                   DO; CHAR = CR; CALL SETSCR;
636   4                   CHAR = LF;
637   4                   END;
638   3               IF CHAR = 0 THEN GO TO BADCOM;
640   3               CALL SETSCR;
641   3               END;
642   2           END COLLECT;

643   1       FIND: PROCEDURE(PA,PB) BYTE;
644   2           DECLARE (PA,PB) BYTE;
                  /* FIND THE STRING IN SCRATCH STARTING AT PA AND ENDING AT PB
          -    */
645   2           DECLARE J ADDRESS,
                      (K, MATCH) BYTE;
646   2           J = BACK ;
647   2           MATCH = FALSE;
648   2               DO WHILE NOT MATCH AND (MAXM > J);
649   3               LAST,J = J + 1; /* START SCAN AT J */
650   3               K = PA ; /* ATTEMPT STRING MATCH AT K */
651   3                   DO WHILE SCRATCH(K) = MEMORY(LAST) AND
                                NOT (MATCH := K = PB);
                        /* MATCHED ONE MORE CHARACTER */
652   4                   K = K + 1; LAST = LAST + 1;
654   4                   END;
655   3               END;
656   2           IF MATCH THEN /* MOVE STORAGE */
657   2               DO; LAST = LAST - 1; CALL MOVER;
660   3               END;
661   2           RETURN MATCH;
662   2           END FIND;

663   1       SETFIND: PROCEDURE;
```

```
                    /* SETUP THE SEARCH STRING FOR F,N, AND S COMMANDS */
664   2             WBE = 0; CALL COLLECT; WBP = WBE;
667   2             END SETFIND;

668   1         CHKFOUND: PROCEDURE;
                    /* CHECK FOR FOUND STRING IN F AND S COMMANDS */
669   2             IF NOT FIND(0,WBP) THEN /* NO MATCH */ GO TO OVERCOUNT;
671   2             END CHKFOUND;



672   1         SETRFCB: PROCEDURE;
                    /* PLACE CHAR INTO READ FILE CONTROL BLOCK AND INCREMENT */
673   2             RFCB((RBP := RBP + 1) - 1) = UCASE(CHAR);
674   2             END SETRFCB;

675   1         PRINTREL: PROCEDURE;
676   2             CALL PRINTLINE(BASELINE+RELLINE);
677   2             END PRINTREL;

678   1         TYPELINES: PROCEDURE;
679   2             DCL I ADDR;
680   2             DCL C BYTE;
681   2             CALL SETLIMITS;
                    /* DISABLE THE * PROMPT */
682   2             INSERTING = TRUE;
683   2             IF DIRECTION = FORWARD THEN
684   2                 DO; RELLINE = 0; I = FRONT;
687   3                 END; ELSE
688   2                 I = FIRST;
689   2             IF (C := MEMORY(I-1)) = LF AND COLUMN <> 0 THEN
690   2                 CALL CRLF;
691   2                 DO I = FIRST TO LAST;
692   3                 IF C = LF THEN
693   3                     DO;
694   4                     CALL PRINTREL;
695   4                     RELLINE = RELLINE + 1;
696   4                     IF BREAK$KEY THEN GO TO OVERCOUNT;
698   4                     END;
699   3                 CALL PRINTC(C:=MEMORY(I));
700   3                 END;
701   2             END TYPELINES;

702   1         SETLPP: PROCEDURE;
                    /* SET DISTANCE TO LINES PER PAGE */
703   2             DISTANCE = LPP;
704   2             END SETLPP;

705   1         SAVEDIST: PROCEDURE;
706   2             TDIST = DISTANCE;
707   2             END SAVEDIST;

708   1         RESTDIST: PROCEDURE;
709   2             DISTANCE = TDIST;
710   2             END RESTDIST;

711   1         PAGE: PROCEDURE;
```

```
712   2          DECLARE I BYTE;
713   2          CALL SAVEDIST;
714   2          CALL SETLPP;
715   2          CALL MOVELINES;
716   2          I = DIRECTION;
717   2          DIRECTION = FORWARD;
718   2          CALL SETLPP;
719   2          CALL TYPELINES;
720   2          DIRECTION = I;
721   2          IF LAST = MAXM OR FIRST = 1 THEN CALL ZERODIST;
723   2              ELSE CALL RESTDIST;
724   2          END PAGE;

725   1      WAIT: PROCEDURE;
                  /* 1/2 SECOND TIME OUT */
726   2          DECLARE I BYTE;
727   2              DO I = 0 TO 19;
728   3              IF BREAK$KEY THEN GO TO RESET;
730   3              CALL TIME(250);
731   3              END;
732   2          END WAIT;

733   1      SETFORWARD: PROCEDURE;
734   2          DIRECTION = FORWARD;
735   2          DISTANCE = 1;
736   2          END SETFORWARD;

737   1      APPHALF: PROCEDURE;
                  /* APPEND 'TIL BUFFER IS AT LEAST HALF FULL */
738   2          CALL SETFF; /* DISTANCE = 0FFFFH */
739   2              DO WHILE DISTNZERO;
740   3              IF FRONT >= HMAX THEN CALL ZERODIST; ELSE
742   3              CALL READLINE;
743   3              END;
744   2          END APPHALF;

745   1      INSCRLF: PROCEDURE;
                  /* INSERT CR LF CHARACTERS */
746   2          CHAR = CR; CALL INSERT;
748   2          CHAR = LF; CALL INSERT;
750   2          END INSCRLF;

751   1      TESTCASE: PROCEDURE;
752   2          DECLARE T BYTE;
                  /* TEST FOR UPPER OR LOWER CASE COMMAND AND SET TRANSLATE
                  FLAG (USED TO DETERMINE IF CHARACTERS WHICH FOLLOW GO TO UPPER
          -    */
753   2          TRANSLATE = TRUE;
754   2          T = LOWERCASE(CHAR);
755   2          CHAR = UTRAN(CHAR);
756   2          TRANSLATE = UPPER OR NOT T;
757   2          END TESTCASE;

758   1      READCTRAN: PROCEDURE;
                  /* SET TRANSLATE TO FALSE AND READ NEXT CHARACTER */
759   2          TRANSLATE = FALSE;
760   2          CHAR = READC;
```

```
761   2          CALL TESTCASE;
762   2          END READCTRAN;

763   1      SINGLECOM: PROCEDURE(C) BYTE;
                 /* RETURN TRUE IF COMMAND IS ONLY CHARACTER, NOT IN MACRO */
764   2          DECLARE C BYTE;
765   2          RETURN CHAR = C AND COMLEN = 1 AND MP = 0;
766   2          END SINGLECOM;

767   1      SINGLERCOM: PROCEDURE(C) BYTE;
768   2          DECLARE C BYTE;
                 /* RETURN TRUE IF COMMAND IS ONLY CHARACTER, NOT IN MACRO, AND
                 THE OPERATOR HAS RESPONDED WITH 'Y' TO A Y/N REQUEST */
769   2          IF SINGLECOM(C) THEN
770   2              DO; CALL CRLF; CALL PRINTCHAR(C);
773   3              CALL MON1(9,.('-(Y/N)',WHAT,'$'));
774   3              C = UCASE(READCHAR); CALL CRLF;
776   3              IF C <> 'Y' THEN GO TO START;
778   3              RETURN TRUE;
779   3              END;
780   2          RETURN FALSE;
781   2          END SINGLERCOM;
```



```
           /* INITIALIZE THE SYSTEM */

782   1      EDCOMMAND: /* PAST LXI SP,STACK */
                 /* I/O BUFFER REGION IS 1/8 AVAILABLE MEMORY */
                 NBUF = SHR(MAX := MAXB - .MEMORY,SECTSHF+3) - 1;
                 /* NBUF IS NUMBER OF BUFFERS - 1 */
783   1          BUFFLENGTH = SHL(DOUBLE(NBUF+1),SECTSHF+1);
                 /* NOW SET MAX AS REMAINDER OF FREE MEMORY */
784   1          IF BUFFLENGTH + 1024 > MAX THEN
785   1              DO; CALL PRINT(.('NO MEMORY$'));
787   2              CALL BOOT;
788   2              END;
                 /* REMOVE BUFFER SPACE AND 00 AT END OF MEMORY VECTOR */
789   1          MAX = MAX - BUFFLENGTH - 1;
                 /* RESET BUFFER LENGTH FOR I AND O */
790   1          BUFFLENGTH = SHR(BUFFLENGTH,1);
791   1          SBUFFADR = MAXB - BUFFLENGTH;
792   1          DBUFFADR = SBUFFADR - BUFFLENGTH;
793   1          MEMORY(MAX) = 0; /* STOPS MATCH AT END OF BUFFER */
794   1          MAXM = MAX - 1;
795   1          HMAX = SHR(MAXM,1);
                 /* NO TRANSLATE, WITH LINE NUMBERS */
796   1          UPPER, PRINTSUPPRESS = FALSE;
797   1          LINESET = TRUE;
                 /* GET SOURCE AND DESTINATION DISKS */
798   1          IF (FCB(1) = ' ') OR (FCB(17) <> ' ') THEN CALL FERR;
800   1          IF (SDISK := FCB(0)) = 0 THEN SDISK = CSELECT; ELSE
802   1              DO; SDISK = SDISK - 1; FCB(0) = 0; /* CLEAR DISK NAME */
805   2              END;
```

```
806   1          IF (DDISK := FCB(16)) = 0 THEN DDISK = SDISK; ELSE
808   1              DDISK = DDISK - 1;
                   /* CLEAR THE XFER FILE */
809   1          CALL XCLEAR;

810   1      RESTART:
               CALL SETUP;
811   1          MEMORY(0) = LF;
812   1          FRONT = 1; BACK = MAXM;
814   1          COLUMN = 0;
815   1          GO TO START;

816   1      OVERCOUNT: FLAG = POUND; GO TO RESET;

818   1      BADCOM: FLAG = WHAT; GO TO RESET;

820   1      OVERFLOW: /* ARRIVE HERE ON OVERFLOW CONDITION (I,F,S COMMAND) */
               FLAG = '>';

821   1      RESET: /* ARRIVE HERE ON ERROR CONDITION */
               PRINTSUPPRESS = FALSE;
822   1          CALL PRINT(.('BREAK "$'));
823   1          CALL PRINTC(FLAG);
824   1          CALL PRINTM(.('" AT $'));
825   1          CALL PRINTC(CHAR);
826   1          CALL CRLF;


827   1      START:
               READBUFF = TRUE;
828   1          MP = 0;


829   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
                    ************************************************************
          -    *** */


830   2          INSERTING = FALSE;
831   2          CALL READCTRAN;
832   2          MI = CBP; /* SAVE STARTING ADDRESS FOR <CR> COMMAND */
833   2          IF SINGLECOM('E') THEN
834   2              DO; CALL TERMINATE;
836   3              IF SDISK <> DDISK THEN /* CHANGE DISKS */
                           /* USER CODE IN HIGH NIBBLE */
837   3                  BDISK = (BDISK AND 0F0H) OR (DDISK AND 0FH);
```

```
838  3              CALL REBOOT;
839  3              END; ELSE


840  2          IF SINGLECOM('H') THEN /* GO TO TOP */
841  2              DO; CALL TERMINATE;
843  3              CHAR = DDISK; DDISK = SDISK; SDISK = CHAR;
                    /* PING - PONG DISKS */
846  3              GO TO RESTART;
847  3              END; ELSE


848  2          IF CHAR = 'I' THEN /* INSERT CHARACTERS */
849  2              DO;
850  3              IF (INSERTING := (CBP = COMLEN) AND (MP = 0)) THEN
851  3                  CALL PRINTNMBASE;
852  3              SCOLUMN = COLUMN; /* STARTING COLUMN POSITION */
853  3              DO WHILE SCANNING;
854  4                  DO WHILE CHAR <> 0;
855  5                  IF CHAR=CTLU OR CHAR=CTLX OR CHAR=CTLR THEN
                           /* LINE DELETE OR RETYPE */
856  5                      DO;
857  6                      DISTANCE = 0; DIRECTION = BACKWARD;
                            /* ELIMINATE OR REPEAT THE LINE */
859  6                      IF CHAR = CTLR THEN
860  6                          DO; CALL CRLF;
862  7                          CALL TYPELINES;
863  7                          END; ELSE
                            /* LINE DELETE */
864  6                          DO; CALL SETLIMITS; CALL SETPTRS;
867  7                          IF CHAR = CTLU THEN
868  7                              DO; CALL CRLF; CALL PRINTNMBASE;
871  8                              END; ELSE
                            /* MUST BE CTLX */
872  7                              DO WHILE COLUMN > SCOLUMN;
873  8                              CALL BACKSPACE;
874  8                              END;
875  7                          END;
876  6                      END; ELSE
877  5                  IF CHAR = CTLH THEN
878  5                      DO;
879  6                      IF (TCOLUMN := COLUMN) > 0 THEN
880  6                          CALL PRINTNMAC(' '); /* RESTORE AFT BACKSP */
881  6                      IF FRONT > 1 AND TCOLUMN > SCOLUMN THEN
882  6                          DO;
883  7                          IF MEMORY(FRONT-1) <> LF THEN
884  7                              DO; /* CHARACTER CAN BE ELIMINATED */
885  8                              CALL DECFRONT;
886  8                              PRINTSUPPRESS = TRUE;
                                    /* BACKSPACE CHARACTER ACCEPTED */
887  8                              COLUMN = 0;
888  8                              DISTANCE = 0; DIRECTION = BACKWARD;
890  8                              CALL TYPELINES;
891  8                              PRINTSUPPRESS = FALSE;
                                    /* COLUMN POSITION NOW RESET */
892  8                              IF (QCOLUMN := COLUMN) < SCOLUMN THEN
893  8                                  QCOLUMN = SCOLUMN;
```

```
894   8                         COLUMN = TCOLUMN; /* ORIGINAL VALUE */
895   8                             DO WHILE COLUMN > QCOLUMN;
896   9                             CALL BACKSPACE;
897   9                             END;
898   8                         TCOLUMN = COLUMN;
899   8                         END;
900   7                     END;
901   6                 CHAR = 0;
902   6                 COLUMN = TCOLUMN;
903   6                 END; ELSE
904   5             IF CHAR = RUBOUT THEN
905   5                 DO; IF FRONT = 1 THEN GO TO RESET;
908   6                 CALL DECFRONT; CALL PRINTC(CHAR:=MEMORY(FRONT));
910   6                 IF CHAR = LF THEN BASELINE=BASELINE-1;
912   6                 CHAR = 0;
913   6                 END; ELSE
                      /* NOT A SPECIAL CASE */
914   5                 DO; IF NOT GRAPHIC(CHAR) THEN
916   6                     DO; CALL PRINTNMAC('^');
918   7                     CALL PRINTNMAC(CHAR + '@');
919   7                     END;
920   6                 IF CHAR = CTLL THEN
921   6                     CALL INSCRLF; ELSE
922   6                     DO; /* COLUMN COUNT GOES UP IF GRAPHIC */
                              /* COMPUTE OUTPUT COLUMN POSITION */
923   7                     IF MP = 0 THEN
924   7                         DO; IF CHAR >= ' ' THEN
926   8                         COLUMN = COLUMN + 1; ELSE
927   8                         IF CHAR = TAB THEN
928   8                         COLUMN = COLUMN + (8 - (COLUMN AND 111B));
                              END;
930   7                     CALL INSERT;
931   7                     END;
932   6                 END;
933   5             IF CHAR = LF THEN CALL PRINTNMBASE;
935   5             IF CHAR = CR THEN
936   5                 CALL PRINTNMAC(CHAR:=LF); ELSE CHAR = 0;
938   5             END;
939   4         END;
940   3         IF CHAR <> ENDFILE THEN /* MUST HAVE STOPPED ON CR */
941   3             CALL INSCRLF;
942   3         IF INSERTING AND LINESET THEN CALL CRLF;
944   3         END; ELSE


945   2     IF SINGLERCOM('O') THEN /* FORGET THIS EDIT */
946   2         GO TO RESTART; ELSE


947   2     IF CHAR = 'R' THEN
948   2         DO; DECLARE I BYTE;
                  /* READ FROM LIB FILE */
950   3         RBP = 1; CALL SETRDMA;
952   3             DO WHILE SCANNING;
953   4             IF RBP > 8 THEN GO TO OVERCOUNT;
955   4             CALL SETRFCB;
956   4             END;
```

```
 957   3              CHAR = ' ';
 958   3              IF (FLAG := RBP = 1) THEN /* READ FROM XFER FILE */
 959   3                  DO;
 960   4                  CALL MOVE(8,.XFCB(1),.RFCB(1));
 961   4                  CALL CLOSE(.XFCB);
 962   4                  END; ELSE /* LIB NAME SPECIFIED */
 963   3                  DO WHILE RBP <= 8;
 964   4                  CALL SETRFCB;
 965   4                  END;
 966   3              RFCB(12), RFCB(32) = 0; /* FILL REEL, AND NEXT RECORD */
 967   3              CALL OPEN(.RFCB); RBP = SECTSIZE;
 969   3              IF DCNT = 255 THEN
 970   3                  DO; FLAG = '0'; GO TO RESET;
 973   4                  END;
 974   3                  DO WHILE (CHAR  = READFILE) <> ENDFILE;
 975   4                  CALL INSERT;
 976   4                  END;
 977   3              I = 0;
 978   3              IF FLAG THEN /* MAY BE XFER DATA IN BUFFER */
 979   3                  DO WHILE I < XBP;
 980   4                  CHAR = XBUFF(I); I = I + 1;
 982   4                  CALL INSERT;
 983   4                  END;
 984   3              END; ELSE


 985   2          IF SINGLERCOM('Q') THEN
 986   2              DO; CALL DELETE(.DFCB); CALL REBOOT;
 989   3              END; ELSE


                    /* MAY BE A COMMAND WHICH HAS AN OPTIONAL DIRECTION AND DISTAN
          -     CE */
 990   2              DO; /* SCAN A SIGNED INTEGER VALUE (IF ANY) */
 991   3              DCL I BYTE;

 992   3              DIGIT: PROCEDURE BYTE;
 993   4                  RETURN (I := CHAR - '0') <= 9;
 994   4                  END DIGIT;

 995   3              NUMBER: PROCEDURE;
 996   4                  DISTANCE = 0;
 997   4                      DO WHILE DIGIT;
 998   5                      DISTANCE = SHL(DISTANCE,3) +
                                       SHL(DISTANCE,1) + I;
 999   5                      CALL READCTRAN;
1000   5                      END;
                        /* RETURN WITH DISTANCE = NUMBER, CHAR = NEXT */
1001   4                  END NUMBER;

1002   3              RELDISTANCE: PROCEDURE;
1003   4                  IF DISTANCE > BASELINE THEN
1004   4                      DO; DIRECTION = FORWARD;
1006   5                      DISTANCE = DISTANCE - BASELINE;
1007   5                      END; ELSE
1008   4                      DO; DIRECTION = BACKWARD;
```

```
1010   5                    DISTANCE = BASELINE - DISTANCE;
1011   5                    END;
1012   4                END RELDISTANCE;

1013   3            CALL SETFORWARD;

1014   3            IF CHAR = '-' THEN
1015   3                DO; CALL READCTRAN; DIRECTION = BACKWARD;
1018   4                END;

1019   3            IF CHAR = POUND THEN
1020   3                DO; CALL SETFF; CALL READCTRAN;
1023   4                END; ELSE

1024   3            IF DIGIT THEN
1025   3                DO; CALL NUMBER;
                        /* MAY BE ABSOLUTE LINE REFERENCE */
1027   4                IF CHAR = ':' THEN
1028   4                    DO; CHAR = 'L';
1030   5                    CALL RELDISTANCE;
1031   5                    END;
1032   4                END; ELSE

1033   3            IF CHAR = ':' THEN /* LEADING COLON */
1034   3                DO; CALL READCTRAN; /* CLEAR THE COLON */
1036   4                CALL NUMBER;
1037   4                CALL RELDISTANCE;
1038   4                IF DIRECTION = FORWARD THEN
1039   4                    DISTANCE = DISTANCE + 1;
1040   4                END;

                        IF DISTZERO THEN 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
                         ***************************************************
                -   *** */

1043   3            IF CHAR = 'B' THEN
1044   3                DO; DIRECTION = 1 - DIRECTION;
1046   4                FIRST = 1; LAST = MAXM; CALL MOVER;
1049   4                END; ELSE
```

```
1050   3              IF CHAR = 'C' THEN
1051   3                  DO; CALL SETCLIMITS; CALL MOVER;
1054   4                  END; ELSE


1055   3              IF CHAR = 'D' THEN
1056   3                  DO; CALL SETCLIMITS;
1058   4                  CALL SETPTRS; /* SETS BACK/FRONT */
1059   4                  END; ELSE


1060   3              IF CHAR = 'K' THEN
1061   3                  DO; CALL SETLIMITS;
1063   4                  CALL SETPTRS;
1064   4                  END; ELSE


1065   3              IF CHAR = 'L' THEN CALL MOVELINES; ELSE

1067   3              IF CHAR = 'P' THEN /* PAGE MODE PRINT */
1068   3                  DO; IF DISTZERO THEN
1070   4                      DO; DIRECTION = FORWARD;
1072   5                      CALL SETLPP; CALL TYPELINES;
1074   5                      END; ELSE
1075   4                      DO WHILE DISTNZERO; CALL PAGE;
1077   5                      CALL WAIT;
1078   5                      END;
1079   4                  END; ELSE


1080   3              IF CHAR = 'T' THEN
1081   3                  CALL TYPELINES; ELSE


1082   3              IF CHAR = 'U' THEN
1083   3                  UPPER = DIRECTION = FORWARD; ELSE

1084   3              IF CHAR = 'V' THEN
1085   3                  DO; /* OV DISPLAYS BUFFER STATE */
1086   4                  IF DISTZERO THEN
1087   4                      DO; CALL PRINTVALUE(BACK-FRONT);
1089   5                      CALL PRINTC('/');
1090   5                      CALL PRINTVALUE(MAXM);
1091   5                      CALL CRLF;
1092   5                      END; ELSE
1093   4                  LINESET = DIRECTION = FORWARD;
1094   4                  END; ELSE

1095   3              IF CHAR = CR THEN /* MAY BE MOVE/TYPE COMMAND */
1096   3                  DO;
1097   4                  IF MI = 1 AND MP = 0 THEN /* FIRST COMMAND */
1098   4                      DO; CALL MOVELINES; CALL SETFORWARD; CALL TYPELINE
          -   S;
1102   5                      END;
1103   4                  END; ELSE
```

```
1104   3              IF DIRECTION = FORWARD OR DISTZERO THEN
1105   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
                       ***************************************************************
          -    *** */


1106   4                  IF CHAR = 'A' THEN
1107   4                      DO; DIRECTION = FORWARD;
1109   5                      FIRST = FRONT; LAST = MAXM; CALL MOVER;
                              /* ALL STORAGE FORWARD */
1112   5                      IF DISTZERO THEN CALL APPHALF;
                              /* DISTANCE = 0 IF APPHALF CALLED */
1114   5                          DO WHILE DISTNZERO;
1115   6                          CALL READLINE;
1116   6                          END;
1117   5                      DIRECTION = BACKWARD; CALL MOVER;
                              /* POINTERS REPOSITIONED */
1119   5                      END; ELSE


1120   4                  IF CHAR = 'F' THEN
1121   4                      DO; CALL SETFIND; /* SEARCH STRING SCANNED
                              AND SETUP BETWEEN 0 AND WBP-1 IN SCRATCH */
1123   5                          DO WHILE DISTNZERO; CALL CHKFOUND;
1125   6                          END;
1126   5                      END; ELSE


1127   4                  IF CHAR = 'J' THEN /* JUXTAPOSITION OPERATION */
1128   4                      DO; DECLARE T ADDRESS;
1130   5                      CALL SETFIND; CALL COLLECT;
1132   5                      WBJ = WBE; CALL COLLECT;
                              /* SEARCH FOR STRING 0 - WBP-1, INSERT STRING WBP TO W
          -    BJ-1,
                              AND THEN DELETE UP TO STRING WBJ TO WBE-1 */
1134   5                      DO WHILE DISTNZERO; CALL CHKFOUND;
1136   6                      /* INSERT STRING */ MI = WBP - 1;
1137   6                          DO WHILE (MI := MI + 1) < WBJ;
1138   7                          CHAR = SCRATCH(MI); CALL INSERT;
1140   7                          END;
1141   6                      T = FRONT; /* SAVE POSITION FOR DELETE */
1142   6                      IF NOT FIND(WBJ,WBE) THEN GO TO OVERCOUNT;
                              /* STRING FOUND, SO MOVE IT BACK */
```

```
1144   6                FIRST = FRONT - (WBE - WBJ);
1145   6                DIRECTION = BACKWARD; CALL MOVER;
                        /* NOW REMOVE THE INTERMEDIATE STRING */
1147   6                FRONT = T;
1148   6                END;
1149   5            END; ELSE


1150   4            IF CHAR = 'M' AND MP = 0 THEN /* MACRO DEFINITION */
1151   4                DO; XP = 255;
1153   5                IF DISTANCE = 1 THEN CALL ZERODIST;
1155   5                    DO WHILE (MACRO(XP := XP + 1) := READC) <> CR;
1156   6                    END;
1157   5                MP = XP; XP = 0; MT = DISTANCE;
1160   5                END; ELSE


1161   4            IF CHAR = 'N' THEN
1162   4                DO; /* SEARCH FOR STRING WITH AUTOSCAN */
1163   5                CALL SETFIND; /* SEARCH STRING SCANNED */
1164   5                    DO WHILE DISTNZERO;
                            /* FIND ANOTHER OCCURRENCE OF STRING */
1165   6                        DO WHILE NOT FIND(0,WBP); /* NOT IN BUFFER */
1166   7                        IF BREAK$KEY THEN GO TO RESET;
1168   7                        CALL SAVEDIST; CALL CLEARMEM;
                                /* MEMORY BUFFER WRITTEN */
1170   7                        CALL APPHALF;
1171   7                        DIRECTION = BACKWARD; FIRST = 1; CALL MOVER;
1174   7                        CALL RESTDIST; DIRECTION = FORWARD;
                                /* MAY BE END OF FILE */
1176   7                        IF BACK >= MAXM THEN GO TO OVERCOUNT;
1178   7                        END;
1179   6                    END;
1180   5                END; ELSE


1181   4            IF CHAR = 'S' THEN /* SUBSTITUTE COMMAND */
1182   4                DO; CALL SETFIND;
1184   5                CALL COLLECT;
                        /* FIND STRING FROM 0 TO WBP-1, SUBSTITUTE STRING
                        BETWEEN WBP AND WBE-1 IN SCRATCH */
1185   5                    DO WHILE DISTNZERO;
1186   6                    CALL CHKFOUND;
                            /* FRONT AND BACK NOW POSITIONED AT FOUND
                            STRING - REPLACE IT */
1187   6                    FRONT = FRONT - (MI := WBP); /* BACKED UP */
1188   6                        DO WHILE MI < WBE;
1189   7                        CHAR = SCRATCH(MI);
1190   7                        MI = MI + 1; CALL INSERT;
1192   7                        END;
1193   6                     END;
1194   5                END; ELSE


1195   4            IF CHAR = 'W' THEN
1196   4                CALL WRITEOUT; ELSE
```

```
1197   4              IF CHAR = 'X' THEN /* TRANSFER LINES */
1198   4                  DO;
1199   5                  CALL SETXDMA;
1200   5                  IF DISTZERO THEN /* CLEAR THE FILE */
1201   5                      DO; CALL XCLEAR;
1203   6                      CALL DELETE(.XFCB);
1204   6                      END; ELSE
                           /* TRANSFER LINES */
1205   -                      DO; DECLARE I  DDRESS;
1207   6                      IF NOT XFERON THEN /* CREATE XFER FILE */
1208   6                          DO; CALL XCLEAR;
1210   7                          XFERON = TRUE;
1211   7                          CALL DELETE(.XFCB); /* OLD VERSION GONE */
1212   7                          CALL MAKE(.XFCB);
1213   '                          IF DCNT =  55 THEN CALL FERR;
1215   7                          END;
1216   6                      CALL SETLIMITS;
1217   6                          DO I = FIRST TO LAST;
1218   7                          CALL PUTXFER(MEMORY(I));
1219   7                          END;
1220   6                      END;
1221   5                  END; ELSE

1222   4              IF CHAR = 'Z' THEN /* SLEEP */
1223   4                  DO;
1224   5                  IF DISTZERO THEN
1225   5                      DO; IF READCHAR = ENDFILE THEN GO TO RESET;
1228   6                      END;
1229   5                      DO WHILE DISTNZERO; CALL WAIT;
1231   6                      END;
1232   5                  END; ELSE
1233   4              IF CHAR <> 0 THEN /* NOT BREAK LEFT OVER FROM STOP */
                       /* DIRECTION FORWARD, BUT NOT ONE OF THE ABOVE */
1234   4              GO TO BADCOM;


                       END; ELSE /* DIRECTION NOT FORWARD */
1236   3                  GO TO BADCOM;
1237   3              END;
1238   2          END;
1239   1      END;
               EOF
```


MODULE INFORMATION:

```
     CODE AREA SIZE     = 18AAH   6314D
     VARIABLE AREA SIZE = 02E0H    736D
     MAXIMUM STACK SIZE = 0020H     32D
     1630 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```

```
0000 ED#
0000 ED#
0AF3 17     0AF3 18     0AFC 19     0AFC 22     0B00 24
0B07 25     0B08 26     0B13 27     0B14 29     0B18 31
0B20 32     0B24 33     0B2C 34     0B31 35     0B38 36
0B39 37     0B39 38     0B41 39     0B42 40     0B47 41
0B4C 42     0B51 43     0B56 44     0B57 45     0B5B 47
0B73 48     0B78 49     0B7D 50     0B8C 51     0B93 52
0B9A 53     0B9B 54     0B9F 56     0BA7 57     0BAA 58
0BCB 59     0BCB 60     0BCF 62     0BDA 64     0BDF 65
0BE7 66     0BE7 67     0BEE 68     0BEF 69     0BEF 70
0BF4 71     0BF9 72     0BFA 73     0C00 75     0C09 76
0C0A 77     0C10 79     0C13 80     0C1B 81     0C1C 82
0C22 84     0C2B 85     0C2C 87     0C32 89     0C3E 90
0C3F 91     0C45 93     0C51 94     0C52 95     0C58 97
0C64 98     0C65 99     0C6B 101    0C74 102    0C75 103
0C7B 105    0C85 106    0C85 107    0C8B 109    0C95 110
0C95 111    0C9B 113    0CA7 114    0CA8 115    0CAE 117
0CB7 118    0CB8 120    0CB8 121    0CBD 122    0CC3 123
0CC4 124    0CC4 125    0CD0 127    0CD8 128    0CDB 129
0CDB 130    0CDE 131    0CDE 132    0CDE 133    0CE7 134
0CE7 135    0CEB 137    0CF6 138    0CF7 139    0CFD 141
0D06 142    0D07 143    0D07 144    0D0E 145    0D14 146
0D17 147    0D18 149    0D1E 151    0D26 152    0D29 153
0D2C 154    0D2D 155    0D2D 156    0D33 157    0D39 158
0D3A 159    0D40 161    0D54 162    0D55 163    0D55 164
0D5B 165    0D61 166    0D70 167    0D80 168    0D8A 170
0D91 171    0D97 172    0D9F 173    0DA5 174    0DA5 175
0DAC 176    0DB2 177    0DBA 179    0DC0 180    0DC8 181
0DCB 182    0DD1 183    0DD4 184    0DD7 185    0DDF 187
0DE5 188    0DE8 189    0DEB 190    0DF3 191    0DF9 192
0DFF 193    0E05 194    0E0F 196    0E16 197    0E1C 198
0E1C 199    0E22 200    0E28 201    0E2E 202    0E33 203
0E3B 204    0E3E 205    0E44 206    0E45 207    0E45 208
0E59 209    0E5A 210    0E5A 211    0E61 212    0E67 213
0E68 214    0ED3 216    0ED3 217    0ED9 218    0E68 219
0E6B 220    0E72 221    0E81 222    0E8E 223    0E9C 225
0EA5 226    0EA8 227    0EB2 228    0EB8 229    0EBB 230
0EC5 231    0ECF 232    0ED2 233    0EDA 234    0EDA 236
0EE6 237    0EE9 238    0EFA 239    0F01 240    0F05 241
0F05 242    0F66 244    0F66 245    0F6C 246    0F05 247
0F0C 248    0F20 249    0F21 250    0F24 251    0F33 252
0F40 253    0F4B 254    0F4E 255    0F58 256    0F62 257
0F65 258    0F6D 259    0F71 261    0F7D 262    0F80 263
0F8C 264    0F93 265    0F94 266    0F98 268    0FA0 270
0FA3 271    0FAE 272    0FB1 273    0FB6 274    0FB6 275
0FC3 276    0FCA 277    0FCB 278    103B 279    103B 280
104B 281    0FCB 282    0FD0 283    0FDB 284    0FE2 285
0FE7 286    0FEA 287    0FED 288    0FF3 289    0FFB 290
0FFE 291    1004 292    1007 293    100E 294    101E 295
1024 296    1027 297    102D 298    1034 299    103A 300
104C 306    1050 308    1058 309    1059 310    1060 311
1061 313    1065 315    1079 316    1079 317    107D 319
1088 320    108E 321    1092 322    1092 323    1096 325
109D 326    10A5 327    10A9 328    10A9 329    10AB 331
```

10B5 332
10BA 333
10C6 334
10D4 335
10E1 336
10EF 337
10FF 339
1104 340
110D 341
1110 342
1115 343
1118 344
1119 345
111F 347
1127 348
1128 349
1130 350
1135 351
113A 352
1141 353
1149 354
114E 355
114F 356
114F 357
1157 358
1158 359
1158 360
1160 361
1161 362
1164 363
1165 364
1165 365
116E 367
1175 368
1178 369
1182 371
118E 373
119E 374
11A1 375
11A1 376
11A6 377
11A6 378
11BA 379
11BA 380
11C1 381
11C9 382
11D0 384
11D5 385
11E5 387
11F1 388
11FA 389
11FD 390
1200 391
1205 392
1208 393
120D 394
1212 395
1217 396
1217 397
1228 398
1233 399
1247 400
1247 401
1247 402
124E 403
1254 404
1255 405
1255 406
125D 408
1260 409
126B 410
126E 411
1273 412
1273 413
1287 414
1287 416
1287 417
128D 418
128E 419
128E 420
129B 421
129B 422
129B 423
12A1 424
12A2 425
12A2 426
12AA 428
12B1 429
12B4 430
12B4 431
12B7 432
12B7 433
12B7 435
12BD 436
12C5 438
12CC 439
12D2 440
12D8 441
12DC 442
12DF 444
12E5 445
12EB 446
12F1 447
12F1 448
12F6 449
12FD 450
132A 451
1330 452
1333 453
133A 454
134D 455
1355 457
135A 458
1368 459
136B 460
1372 461
1378 462
137B 463
1383 465
1389 466
1390 467
1393 469
139A 470
13A1 471
13A1 472
13A2 473
13A2 474
13A9 475
13AA 476
13AA 477
13B1 478
13B2 479
13B2 480
13B9 481
13BA 482
13BA 483
13C1 484
13C2 485
13C2 486
13C9 487
13CA 488
13CE 490
13D6 491
13E2 492
13E5 493
13EC 495
13FC 496
13FF 497
140A 498
140D 499
140D 500
1413 501
141F 502
1422 503
1432 504
1439 505
1440 507
144B 508
144E 509
144E 510
1451 511
1452 512
1452 513
1457 514
1458 515
1458 516
145D 517
145E 518
145E 519
1461 520
1464 521
1465 522
1465 523
146D 525
1473 526
147F 527
1488 528
1494 529
1497 531
149D 532
14AF 533
14B8 534
14C3 535
14C3 536
14C4 537
1504 539
1508 541
150F 542
1517 543
151B 544
14C4 545
14C4 546
14D0 547
14D3 548
14E2 550
14E5 551
14E6 552
14E6 553
14F1 554
14F4 555
14FC 557
14FF 558
1500 559
1500 560
1503 561
151B 562
151B 564
151B 565
1527 567
152A 568
152B 569
152B 570
152E 571
153D 572
1545 574
1548 575
1549 576
1549 577
154C 578
154D 579
154D 580
1550 581
1557 582
1569 583
156F 584
1572 585
1575 586
1576 587
1576 588
157B 589
1581 590
1587 591
158A 592
1591 593
1594 594
159B 595
159E 596
15A1 597
15AD 599
15B2 600
15B5 601
15B5 602
15B6 603
15B6 604
15B9 605
15BC 606
15BD 607
15BD 608
15C0 609
15CB 610
15D2 611
15D5 612
15D8 613
15D9 614
15D9 615
15E6 616
15E9 617
15F4 618
15F7 619
15FF 620
1602 621
1603 622
1603 623
1624 624
1624 625
1652 626
1652 627
165F 628
166B 629
1638 635
1624 631
162B 632
1633 634

SERIAL # _____

1640 637	1640 638	1648 639	164B 640	164E 641
1651 642	166F 643	1675 646	167B 647	1680 648
1696 649	16A0 650	16A6 651	16D3 652	16DA 653
16E1 654	16E4 655	16E7 656	16EE 658	16F5 659
16F8 660	16F8 661	16FC 662	16FC 663	16FC 664
1701 665	1704 666	170A 667	170B 668	170B 669
1719 670	171C 671	171D 672	171D 673	1737 674
1738 675	1738 676	1745 677	1746 678	1746 681
1749 682	174E 683	1756 685	175C 686	1762 687
1765 688	176B 689	178C 690	178F 691	17A1 692
17A9 694	17AC 695	17B3 696	17BA 697	17BD 698
17BD 699	17CC 700	17D9 701	17DA 702	17DA 703
17E0 704	17E1 705	17E1 706	17E7 707	17E8 708
17E8 709	17EE 710	17EF 711	17EF 713	17F2 714
17F5 715	17F8 716	17FE 717	1803 718	1806 719
1809 720	180F 721	1830 722	1836 723	1839 724
183A 725	183A 727	1848 728	184F 729	1852 730
1857 731	1861 732	1862 733	1862 734	1867 735
186D 736	186E 737	186E 738	1871 739	1878 740
1884 741	188A 742	188D 743	1890 744	1891 745
1891 746	1896 747	1899 748	189E 749	18A1 750
18A2 751	18A2 753	18A7 754	18B1 755	18BB 756
18C5 757	18C6 758	18C6 759	18CB 760	18D1 761
18D4 762	18D5 763	18D9 765	18FC 766	18FC 767
1900 769	190B 771	190E 772	1915 773	191D 774
1927 775	192A 776	1932 777	1935 778	1938 779
1938 780	193B 781	01C0 782	01DA 783	01E9 784
01F9 786	01FF 787	0202 788	0202 789	0211 790
021E 791	022A 792	0236 793	023F 794	0246 795
0253 796	025D 797	0262 798	027A 799	027D 800
0288 801	0291 803	0298 804	029D 805	029D 806
02A8 807	02B1 808	02B8 809	02BB 810	02BE 811
02C3 812	02C9 813	02CF 814	02D4 815	02D7 816
02DF 817	02E2 818	02EA 819	02ED 820	02F5 821
02FD 822	0303 823	030A 824	0310 825	0317 826
031A 827	0322 828	0327 829	0327 830	032C 831
032F 832	0335 833	033E 835	0341 836	034B 837
035C 838	035F 839	0362 840	036B 842	036E 843
0374 844	037A 845	0380 846	0383 847	0386 848
038E 850	03AB 851	03AE 852	03B4 853	03BB 854
03C3 855	03E7 857	03ED 858	03F2 859	03F9 861
03FC 862	03FF 863	0402 865	0405 866	0408 867
0410 869	0413 870	0416 871	0419 872	0423 873
0426 874	0429 875	0429 876	042C 877	0434 879
0441 880	0446 881	045F 883	046D 885	0470 886
0475 887	047A 888	0480 889	0485 890	0488 891
048D 892	049A 893	04A0 894	04A6 895	04B0 896
04B3 897	04B6 898	04BC 899	04BC 900	04BC 901
04C1 902	04C7 903	04CA 904	04D2 906	04DE 907
04E1 908	04E4 909	04F3 910	04FB 911	0502 912
0507 913	050A 915	0516 917	051B 918	0524 919
0524 920	052C 921	0532 923	053A 925	0542 926
054C 927	0554 928	0562 929	0562 930	0565 931
0565 932	0565 933	056D 934	0570 935	0578 936
0585 937	058A 938	058D 939	0590 940	0598 941

```
059B 942    05A6 943    05A9 944    05AC 945    05B5 946
05B8 947    05C0 950    05C5 951    05C8 952    05CF 953
05D8 954    05DB 955    05DE 956    05E1 957    05E6 958
05F5 960    0605 961    060B 962    060E 963    0617 964
061A 965    061D 966    0627 967    062D 968    0632 969
063A 971    063F 972    0642 973    0642 974    064D 975
0650 976    0653 977    0658 978    065F 979    0669 980
0676 981    067D 982    0680 983    0683 984    0686 985
068F 987    0695 988    0698 989    193B 992    193B 993
194A 994    194A 995    194A 996    1950 997    1957 998
196F 999    1972 1000   1975 1001   1976 1002   1976 1003
1982 1005   1987 1006   1995 1007   1998 1009   199D 1010
19A9 1011   19A9 1012   069B 1013   069E 1014   06A6 1016
06A9 1017   06AE 1018   06AE 1019   06B6 1021   06B9 1022
06BC 1023   06BF 1024   06C6 1026   06C9 1027   06D1 1029
06D6 1030   06D9 1031   06D9 1032   06DC 1033   06E4 1035
06E7 1036   06EA 1037   06ED 1038   06F5 1039   06FC 1040
06FC 1041   0703 1042   0708 1043   0710 1045   0717 1046
071D 1047   0723 1048   0726 1049   0729 1050   0731 1052
0734 1053   0737 1054   073A 1055   0742 1057   0745 1058
0748 1059   074B 1060   0753 1062   0756 1063   0759 1064
075C 1065   0764 1066   076A 1067   0772 1069   0779 1071
077E 1072   0781 1073   0784 1074   0787 1075   078E 1076
0791 1077   0794 1078   0797 1079   079A 1080   07A2 1081
07A8 1082   07B0 1083   07BE 1084   07C6 1086   07CD 1088
07DB 1089   07E0 1090   07E8 1091   07EB 1092   07EE 1093
07F9 1094   07FC 1095   0804 1097   081C 1099   081F 1100
0822 1101   0825 1102   0825 1103   0828 1104   083B 1106
0843 1108   0848 1109   084E 1110   0854 1111   0857 1112
085E 1113   0861 1114   0868 1115   086B 1116   086E 1117
0873 1118   0876 1119   0879 1120   0881 1122   0884 1123
088B 1124   088E 1125   0891 1126   0894 1127   089C 1130
089F 1131   08A2 1132   08A8 1133   08AB 1134   08B2 1135
08B5 1136   08BC 1137   08CA 1138   08D7 1139   08DA 1140
08DD 1141   08E3 1142   08F3 1143   08F6 1144   0906 1145
090B 1146   090E 1147   0914 1148   0917 1149   091A 1150
0932 1152   0937 1153   0943 1154   0946 1155   0961 1156
0964 1157   096A 1158   096F 1159   0979 1160   097C 1161
0984 1163   0987 1164   098E 1165   099C 1166   09A3 1167
09A6 1168   09A9 1169   09AC 1170   09AF 1171   09B4 1172
09BA 1173   09BD 1174   09C0 1175   09C5 1176   09D1 1177
09D4 1178   09D7 1179   09DA 1180   09DD 1181   09E5 1183
09E8 1184   09EB 1185   09F2 1186   09F5 1187   0A06 1188
0A10 1189   0A1D 1190   0A24 1191   0A27 1192   0A2A 1193
0A2D 1194   0A30 1195   0A38 1196   0A3E 1197   0A46 1199
0A49 1200   0A50 1202   0A53 1203   0A59 1204   0A5C 1207
0A64 1209   0A67 1210   0A6C 1211   0A72 1212   0A78 1213
0A80 1214   0A83 1215   0A83 1216   0A86 1217   0A98 1218
0AA3 1219   0AB0 1220   0AB0 1221   0AB3 1222   0ABB 1224
0AC2 1226   0ACA 1227   0ACD 1228   0ACD 1229   0AD4 1230
0AD7 1231   0ADA 1232   0ADD 1233   0AE5 1234   0AE8 1235
0AEB 1236   0AEE 1237   0AEE 1238   0AF1 1239
0000 MODULE#
```