DIGITAL RESEARCH

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

CP/M CONTEXT EDITOR (ED)

CP/M VERSION _____

SER. # _____

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

100H: DECLARE BDOS LITERALLY '0005H', BOOT LITERALLY '0';
    /*      COPYRIGHT (C) 1976
            DIGITAL RESEARCH
            BOX 579 PACIFIC GROVE
            CALIFORNIA 93950
    */
DECLARE COPYRIGHT DATA ('COPYRIGHT (C) 1976, 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 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>.LIB UNTIL <ENDFILE>  AND
                     INSERT INTO TEXT
     S               SEARCH FOR FIRST STRING, REPLACE BY SECOND STRING
     T               TYPE LINES
     U               MOVE UP LINES (SAME AS -NP)
     W               WRITE LINES OF TEXT TO FILE
     Z               SLEEP FOR 1/2 SECOND (USED IN MACROS TO STOP DISPLAY)
     <CR>             MOVE UP OR DOWN AND PRINT ONE LINE

         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>NDELTA<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>  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
```

```
121>                (OCCURS IF SEARCH STRING CANNOT BE FOUND).
122>      LETTER O    CANNOT OPEN (FILENAME).LIB IN R COMMAND
123>
124>      THE ERROR CHARACTER IS ALSO ACCOMPANIED BY THE LAST CHARACTER
125>      SCANNED WHEN THE ERROR OCCURRED.                        */
126>
127>NDECLARE LIT LITERALLY 'LITERALLY',
128>      DCL LIT 'DECLARE',
129>      PROC LIT 'PROCEDURE',
130>      ADDR LIT 'ADDRESS',
131>      CTLL LIT '0CH',
132>      CTLU LIT '15H',  /* LINE DELETE IN INSERT MODE */
133>      LCA  LIT '110$0001B',  /* LOWER CASE A */
134>      LCZ  LIT '111$1010B',  /* LOWER CASE Z */
135>      ENDFILE LIT '1AH';     /* CP/M END OF FILE */
136>
137>DECLARE MAXA ADDRESS INITIAL(0006H),  /* MONITOR LOCATION */
138>      MAX ADDRESS,                    /* FILLED FROM MAXA */
139>      MAXM ADDRESS,                   /* MINUS 1 */
140>      HMAX ADDRESS,             /* HALF THE VALUE OF MAXM */
141>      MAXB BASED MAXA ADDRESS;  /* TEMP TO COMPUTE ADDR */
142>
143>NDECLARE NBUF LITERALLY '7',  /* NUMBER OF BUFFERS-1 */
144>      BUFS LITERALLY '1024',     /* (NBUF+1) * 128 */
145>      BDISKA ADDRESS INITIAL(4),   /* BOOT DISK ADDRESS */
146>      BDISK BASED BDISKA BYTE,     /* BOOT DISK VALUE */
147>      FCBA ADDRESS INITIAL(5CH),
148>      FCB BASED FCBA (33) BYTE,   /* DEFAULT FCB AT 5CH */
149>      BUFA ADDRESS INITIAL(80H),       /* DISK IO BUFFER ADDRESS */
150>      RFCB (33) BYTE /* READER FILE CONTROL BLOCK */
151>          INITIAL(0, /* FILE NAME */ '          ',
152>                     /* FILE TYPE */ 'LIB'),
153>      RBP BYTE,                        /* READ BUFFER POINTER */
154>      BUFF BASED BUFA (33) BYTE,       /* DISKIO BUFFER */
155>      SFCB BASED FCBA (33) BYTE, /* SOURCE FILE CONTROL */
156>      SBUFF(BUFS) BYTE,          /* SOURCE BUFFER */
157>      DFCB (33) BYTE,            /* DEST FILE CONTROL BLOCK */
158>      DBUFF(BUFS) BYTE,          /* DESTINATION BUFFER */
159>      NSOURCE ADDRESS,           /* NEXT SOURCE CHARACTER */
160>      NDEST   ADDRESS;           /* NEXT DESTINATION CHAR */
161>
162>DECLARE SDISK BYTE,      /* SOURCE FILE DISK */
163>      DDISK BYTE;         /* DESTINATION FILE DISK */
164>  /* IO SECTION */
165>MON1: PROCEDURE(F,A); DECLARE F BYTE, A ADDRESS;
166>      GO TO BDOS;
167>      END MON1;
168>
169>MON2: PROCEDURE(F,A) BYTE; DECLARE F BYTE, A ADDRESS;
170>      GO TO BDOS;
171>      END MON2;
172>
173>READCHAR: PROCEDURE BYTE; RETURN MON2(1,0);
174>      END READCHAR;
175>
176>NDECLARE TRUE LITERALLY '1', FALSE LITERALLY '0',
177>      FOREVER LITERALLY 'WHILE TRUE',
178>      CR LITERALLY '13',
179>      LF LITERALLY '10',
180>      WHAT LITERALLY '63';
```


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


```
181>
182>^PRINTCHAR: PROCEDURE(CHAR);
183>      DECLARE CHAR BYTE;
184>      CALL MON1(2,CHAR);
185>      END PRINTCHAR;
186>
187>CRLF: PROCEDURE;
188>      CALL PRINTCHAR(CR); CALL PRINTCHAR(LF);
189>      END CRLF;
190>
191>^PRINT: PROCEDURE(A);
192>      DECLARE A ADDRESS;
193>      CALL CRLF; CALL MON1(9,A);
194>      END PRINT;
195>
196>READ: PROCEDURE(A);
197>      DECLARE A ADDRESS;
198>      CALL MON1(10,A);
199>      END READ;
200>
201>NDECLARE DCNT BYTE;
202>
203>
204>OPEN: PROCEDURE(FCB);
205>       DECLARE FCB ADDRESS;
206>       DCNT = MON2(15,FCB);
207>       END OPEN;
208>
209>CLOSE: PROCEDURE(FCB);
210>      DECLARE FCB ADDRESS;
211>      DCNT = MON2(16,FCB);
212>      END CLOSE;
213>
214>SEARCH: PROCEDURE(FCB);
215>      DECLARE FCB ADDRESS;
216>      DCNT = MON2(17,FCB);
217>      END SEARCH;
218>
219>>DELETE: PROCEDURE(FCB);
220>      DECLARE FCB ADDRESS;
221>      CALL MON1(19,FCB);
222>      END DELETE;
223>
224>NDISKREAD: PROCEDURE(FCB) BYTE;
225>      DECLARE FCB ADDRESS;
226>      RETURN MON2(20,FCB);
227>      END DISKREAD;
228>
229>NDISKWRITE: PROCEDURE(FCB) BYTE;
230>      DECLARE FCB ADDRESS;
231>      RETURN MON2(21,FCB);
232>      END DISKWRITE;
233>
234>MAKE: PROCEDURE(FCB);
235>      DECLARE FCB ADDRESS;
236>      DCNT = MON2(22,FCB);
237>      END MAKE;
238>
239>RENAME: PROCEDURE(FCB);
240>      DECLARE FCB ADDRESS;
```

```
241>      CALL MON1(23,FCB);
242>      END RENAME;
243>
244>DECLARE (MAXLEN,COMLEN) BYTE, COMBUFF(128) BYTE,
245>      (TCBP,CBP) BYTE;
246>
247>READCOM: PROCEDURE;
248>      MAXLEN = 128; CALL READ(.MAXLEN);
249>      END READCOM;
250>
251>BREAK$KEY: PROCEDURE BYTE;
252>      IF MON2(11,0) THEN
253>          DO; /* CLEAR CHAR */
254>          CALL MON1(1,0); RETURN TRUE;
255>          END;
256>      RETURN FALSE;
257>      END BREAK$KEY;
258>
259>CSELECT: PROCEDURE BYTE;
260>      /* RETURN CURRENT DRIVE NUMBER */
261>      RETURN MON2(25,0);
262>      END CSELECT;
263>
264>SELECT: PROCEDURE(DISK);
265>      DECLARE DISK BYTE;
266>      /* SET DRIVE NUMBER */
267>      CALL MON1(14,DISK);
268>      END SELECT;
269>
270>SETDMA: PROCEDURE(A);
271>      DECLARE A ADDRESS;
272>      /* SET DMA ADDRESS */
273>      CALL MON1(26,A);
274>      END SETDMA;
275>
276>LIFTHEAD: PROCEDURE;
277>      /* LIFT HEAD ON SA 3900 DISK */
278>      CALL MON1(12,0);
279>      END LIFTHEAD;
280>   /*  INPUT / OUTPUT BUFFERING ROUTINES */
281>
282>MOVE: PROCEDURE(S,D,N);
283>      DECLARE (S,D) ADDRESS, N BYTE;
284>      DECLARE A BASED S BYTE, B BASED D BYTE;
285>          DO WHILE (N := N - 1) <> 255;
286>          B = A; S = S + 1; D = D + 1;
287>          END;
288>      END MOVE;
289>
290>FERR: PROCEDURE;
291>      CALL CRLF;
292>      CALL PRINT ('FILE ERROR$');
293>      CALL CRLF;
294>      GO TO BOOT;
295>      END FERR;
296>
297>SETTYPE: PROCEDURE(A);
298>      DECLARE A ADDRESS;
299>      CALL MOVE(A,.DFCB+9,3);
300>      END SETTYPE;
```



```
301>
302>SETUP: PROCEDURE;
303>      NSOURCE = LENGTH(SBUFF); NDEST = 0;
304>      SFCB(12) = 0; SFCB(32) = 0;
305>      /* REEL AND RECORD ZEROED */
306>      /* COPY NAME TO DESTINATION FCB */
307>      CALL MOVE(FCBA,.DFCB,33);
308>      /* SOURCE AND DESTINATION DISKS SET */
309>      CALL SELECT(SDISK);
310>      CALL OPEN(FCBA);
311>      IF DCNT = 255 THEN
312>          DO; CALL MAKE(FCBA);
313>          IF DCNT = 255 THEN CALL FERR;
314>          CALL PRINT('NEW FILE$');
315>          CALL CRLF;
316>          END;
317>      CALL SETTYPE('BAK');
318>      CALL DELETE(.DFCB);
319>      IF SDISK <> DDISK THEN
320>          DO; /* REMOVE BAK FILES FROM DESTINATION DISK */
321>          CALL SELECT(DDISK);
322>          CALL DELETE(.DFCB);
323>          END;
324>      CALL SETTYPE('$$$');
325>      CALL DELETE(.DFCB);
326>      CALL MAKE(.DFCB);
327>      DFCB(32) = 0;  /* NEXT RECORD IS ZERO */
328>      IF DCNT = 255 THEN CALL FERR;
329>      /* THE TEMP FILE IS NOW CREATED */
330>      END SETUP;
331>
332>FILLSOURCE: PROCEDURE;
333>      DECLARE I BYTE;
334>      ZN: PROCEDURE;
335>          NSOURCE = 0;
336>          END ZN;
337>
338>       CALL ZN;
339>      CALL SELECT(SDISK);
340>          DO I = 0 TO NBUF;
341>          CALL SETDMA(.SBUFF(NSOURCE));
342>          IF (DCNT := DISKREAD(FCBA)) <>  0 THEN
343>              DO; IF DCNT >  1 THEN CALL FERR;
344>              SBUFF(NSOURCE) = ENDFILE;
345>              I = NBUF;
346>              END;
347>          ELSE
348>              NSOURCE = NSOURCE + 80H;
349>          END;
350>      CALL ZN;
351>      END FILLSOURCE;
352>
353>GETSOURCE: PROCEDURE BYTE;
354>      DECLARE B BYTE;
355>      IF NSOURCE >  = LENGTH(SBUFF) THEN CALL FILLSOURCE;
356>      IF (B := SBUFF(NSOURCE)) <>  ENDFILE THEN
357>          NSOURCE = NSOURCE + 1;
358>      RETURN B;
359>      END GETSOURCE;
360>
```

```
361>WRITEDEST: PROCEDURE;
362>     /* WRITE OUTPUT BUFFER UP TO (NOT INCLUDING) NDEST.
363>     LOW 7 BITS OF NDEST ARE ZERO */
364>     DECLARE (I,N) BYTE;
365>     ZN: PROCEDURE;
366>         NDEST = 0;
367>         END ZN;
368>
369>     CALL SELECT(DDISK);
370>     IF LOW((N := SHR(NDEST,7) - 1)) = 255 THEN RETURN;
371>     CALL ZN;
372>         DO I = 0 TO N;
373>         CALL SETDMA(.DBUFF(NDEST));
374>         IF DISKWRITE(.DFCB) <> 0 THEN CALL FERR;
375>         NDEST = NDEST + 128;
376>         END;
377>     CALL ZN;
378>     END WRITEDEST;
379>
380>^PUTDEST: PROCEDURE(B);
381>     DECLARE B BYTE;
382>     IF NDEST >= LENGTH(DBUFF) THEN CALL WRITEDEST;
383>     DBUFF(NDEST) = B;
384>     NDEST = NDEST + 1;
385>     END PUTDEST;
386>
387>FINIS: PROCEDURE;
388>     MOVEUP: PROCEDURE;
389>     CALL MOVE(.DFCB,.DFCB+16,16);
390>     END MOVEUP;
391>
392>     /* CLEAR OUTPUT */
393>         DO WHILE (LOW(NDEST) AND 7FH) <> 0;
394>         CALL PUTDEST(ENDFILE);
395>         END;
396>     CALL WRITEDEST;
397>
398>     CALL CLOSE(.DFCB);
399>     IF DCNT = 255 THEN CALL FERR;
400>     /* RENAME OLD FILE TO BAK */
401>     CALL SETTYPE(.'BAK'); CALL MOVEUP;
402>     CALL SELECT($DISK);
403>     CALL MOVE(FCBA,.DFCB,16);
404>     CALL RENAME(.DFCB);
405>     CALL MOVEUP;
406>     /* RENAME $$$ TO OLD NAME */
407>     CALL SETTYPE(.'$$$');
408>     CALL SELECT(DDISK);
409>     CALL RENAME(.DFCB);
410>     END FINIS;
411>
412>
413>NDECLARE
414>     LPP LIT '23',          /* LINES PER PAGE */
415>     FORWARD LIT '1',
416>     BACKWARD LIT '0',
417>     RUBOUT LIT '07FH',
418>     FOUND LIT '23H',
419>     MACSIZE LIT '128',      /* MAX MACRO SIZE */
420>     SCRSIZE LIT '100',      /* SCRATCH BUFFER SIZE */
421>     COMSIZE LIT 'ADDRESS';  /* DETERMINES MAX COMMAND NUMBER*/
422>
423>NDCL MACRO(MACSIZE) BYTE,
424>     SCRATCH(SCRSIZE) BYTE,      /* SCRATCH BUFFER FOR F,N,S */
425>     (UBP, WBE, WBJ)  BYTE,      /* END OF F STRING, S STRING, J STRING */
426>     (FLAG, MP, MI, XP, COLUMN) BYTE,
427>     MT COMSIZE;
428>
429>NDCL (START, RESTART, OVERCOUNT, OVERFLOW, RESET, BADCOM) LABEL;
430>
431>NDCL INSERTING BYTE,    /* TRUE IF INSERTING CHARACTERS */
432>     READBUFF  BYTE;    /* TRUE IF END OF READ BUFFER */
433>
434>DECLARE
435>     TAB LITERALLY '11Q',
436>     EOS LITERALLY '0FFH';
437>
438>
439>TTYCHAR: PROCEDURE(CHAR);
440>     DECLARE CHAR BYTE;
441>     IF CHAR >= ' ' THEN COLUMN = COLUMN + 1;
442>     IF CHAR = CR THEN COLUMN = 0;
443>     CALL PRINTCHAR(CHAR);
444>     END TTYCHAR;
445>
446>^PRINTC: PROCEDURE(CHAR);
447>     DECLARE (CHAR,I,J) BYTE;
448>     I = CHAR = TAB AND 7 - (COLUMN AND 7);
449>     IF CHAR = TAB THEN CHAR = ' ';
450>         DO J = 0 TO I;
451>         CALL TTYCHAR(CHAR);
452>         END;
453>     END PRINTC;
454>
455>^PRINTNMAC: PROCEDURE(CHAR);
456>     DECLARE CHAR BYTE;
457>     /* PRINT IF NOT IN MACRO EXPANSION */
458>     IF MP <> 0 THEN RETURN;
459>     CALL PRINTC(CHAR);
460>     END PRINTNMAC;
461>
462>
463>
464>
465>NDECLARE TRANSLATE BYTE, /* TRUE IF TRANSLATION TO UPPER CASE */
466>     UPPER BYTE;         /* TRUE IF GLOBALLY TRANLATING TO UC */
467>
468>LOWERCASE: PROCEDURE(C) BYTE;
469>     DECLARE C BYTE;
470>     /* RETURN TRUE IF LOWER CASE ALPHABETIC */
471>     RETURN C >= LCA AND C <= LCZ;
472>     END LOWERCASE;
473>
474>UTRAN: PROCEDURE(C) BYTE;
475>     DECLARE C BYTE;
476>     /* TRANSLATE TO UPPER CASE IF ALPHABETIC LOWER AND TRANSLATE */
477>     IF TRANSLATE AND LOWERCASE(C) THEN C = C AND 10111111B;
478>     RETURN C;
479>     END UTRAN;
480>
```


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

```
481>READC: PROCEDURE BYTE;
482>     /* MAY BE MACRO EXPANSION */
483>     IF MP > 0 THEN
484>         DO;
485>         IF BREAK$KEY THEN GO TO RESET;
486>         IF XP >= MP THEN
487>             DO; /* START AGAIN */
488>             IF MT <> 0 THEN
489>                 DO; IF (MT:=MT-1) = 0 THEN
490>                     GO TO RESET;
491>                 END;
492>             XP = 0;
493>             END;
494>         RETURN UTRAN(MACRO(XP:=XP + 1) - 1);
495>         END;
496>     IF INSERTING THEN RETURN UTRAN(READCHAR);
497>
498>     /* GET COMMAND LINE */
499>     IF READBUFF THEN
500>         DO; READBUFF = FALSE; CALL LIFTHEAD;
501>         CALL PRINTC('*');
502>         CALL READCOM; CBP = 0;
503>         CALL PRINTC(LF);
504>         COLUMN = 0;
505>         END;
506>     IF (READBUFF:= CBP = COMLEN ) THEN COMBUFF(CBP) = CR;
507>     RETURN UTRAN(COMBUFF(CBP:= CBP +1) -1));
508>     END READC;
509>
510>READFILE: PROCEDURE BYTE;
511>     IF RBP >= 80H THEN
512>         DO; CALL SELECT(SDISK);
513>         IF DISKREAD(.RFCB) <> 0 THEN RETURN ENDFILE;
514>         RBP = 0;
515>         END;
516>     RETURN UTRAN(BUFF(RBP:= RBP + 1) - 1));
517>     END READFILE;
518>
519>GETMEM: PROC(I) BYTE;
520>     DCL I ADDR;
521>     RETURN MEMORY(I);
522>     END GETMEM;
523>
524>PUTMEM: PROC(I,J);
525>     DCL I ADDR, J BYTE;
526>     MEMORY(I) = J;
527>     END PUTMEM;
528>
529>TRANSFER: PROC(I,J);
530>     DCL (I,J) ADDR;
531>     CALL PUTMEM(I,GETMEM(J));
532>     END TRANSFER;
533>
534>DCL (DISTANCE, TDIST) COMSIZE,
535>     (DIRECTION, CHAR) BYTE,
536>     ( FRONT, BACK, FIRST, LAST) ADDR;
537>
538>SETFF: PROCEDURE;
539>     DISTANCE = 0FFFFH;
540>     END SETFF;
```



```
541>
542>NDISTZERO: PROCEDURE BYTE;
543>     /* RETURN TRUE IF DISTANCE IS ZERO */
544>     RETURN DISTANCE = 0;
545>     END DISTZERO;
546>
547>ZERODIST: PROCEDURE;
548>     DISTANCE = 0;
549>     END ZERODIST;
550>
551>NDISTNZERO: PROCEDURE BYTE;
552>     /* CHECK FOR ZERO DISTANCE AND DECREMENT */
553>     IF NOT DISTZERO THEN
554>         DO; DISTANCE = DISTANCE - 1;
555>         RETURN TRUE;
556>         END;
557>     RETURN FALSE;
558>     END DISTNZERO;
559>
560>SETLIMITS: PROC;
561>     DCL (I,K,L,M) ADDR, (MIDDLE,LOOPING) BYTE;
562>     IF DIRECTION = BACKWARD THEN
563>         DO; DISTANCE = DISTANCE+1; I = FRONT; L = 0; K = 0FFFFH;
564>         END; ELSE
565>         DO; I = BACK; L = MAXM; K = 1;
566>         END;
567>
568>     LOOPING = TRUE;
569>         DO WHILE LOOPING;
570>             DO WHILE (MIDDLE:= I <> L) AND
571>                 GETMEM(M:=I+K) <> LF;
572>             I = M;
573>             END;
574>         LOOPING = (DISTANCE:= DISTANCE - 1) <> 0;
575>         IF NOT MIDDLE THEN
576>             DO; LOOPING = FALSE;
577>             I = I - K;
578>             END; ELSE
579>         IF LOOPING THEN I = M;
580>         END;
581>
582>     IF DIRECTION = BACKWARD THEN
583>         DO; FIRST= I; LAST = FRONT - 1;
584>         END; ELSE
585>         DO; FIRST = BACK + 1; LAST = I + 1;
586>         END;
587>     END SETLIMITS;
588>
589>SETPTRS: PROCEDURE;
590>     IF DIRECTION = FORWARD THEN BACK=LAST; ELSE FRONT=FIRST;
591>     END SETPTRS;
592>
593>
594>INCFRONT: PROC; FRONT = FRONT + 1;
595>     END INCFRONT;
596>INCBACK: PROCEDURE; BACK = BACK + 1;
597>     END INCBACK;
598>NDECFRONT: PROC; FRONT = FRONT - 1;
599>     END DECFRONT;
600>NDECBACK: PROC; BACK = BACK - 1;
```


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

```
601>      END DECBACK;
602>
603>MOVER: PROC;
604>      IF DIRECTION = FORWARD THEN
605>          DO WHILE BACK < LAST; CALL INCBACK;
606>          CALL TRANSFER(FRONT,BACK); CALL INCFRONT;
607>          END; ELSE
608>          DO WHILE FRONT > FIRST; CALL DECFRONT;
609>          CALL TRANSFER(BACK,FRONT); CALL DECBACK;
610>          END;
611>      END MOVER;
612>
613>MOVELINES: PROC;
614>      CALL SETLIMITS;
615>      CALL MOVER;
616>      END MOVELINES;
617>
618>SETCLIMITS: PROC;
619>      IF DIRECTION = BACKWARD THEN
620>          DO; LAST = BACK;
621>          IF DISTANCE > FRONT THEN
622>              FIRST = 1; ELSE FIRST = FRONT - DISTANCE;
623>          END; ELSE
624>          DO; FIRST = FRONT;
625>          IF DISTANCE >= MAX - BACK THEN
626>              LAST = MAXM; ELSE LAST = BACK + DISTANCE;
627>          END;
628>      END SETCLIMITS;
629>
630>READLINE: PROCEDURE;
631>      DECLARE B BYTE;
632>      /* READ ANOTHER LINE OF INPUT */
633>      CTRAN: PROCEDURE(B) BYTE;
634>          DECLARE B BYTE;
635>          /* CONDITIONALLY TRANSLATE TO UPPER CASE ON INPUT */
636>          IF UPPER THEN RETURN UTRAN(B);
637>          RETURN B;
638>          END CTRAN;
639>      DO FOREVER;
640>      IF FRONT >= BACK THEN GO TO OVERFLOW;
641>      IF (B := CTRAN(GETSOURCE)) = ENDFILE THEN
642>          DO; CALL ZERODIST; RETURN;
643>          END;
644>      CALL PUTMEM(FRONT,B);
645>      CALL INCFRONT;
646>      IF B = LF THEN RETURN;
647>      END;
648>      END READLINE;
649>
650>WRITELINE: PROCEDURE;
651>      /* WRITE ONE LINE OUT */
652>      DECLARE B BYTE;
653>          DO FOREVER;
654>          IF BACK >= MAXM THEN /* EMPTY */
655>              DO; CALL ZERODIST; RETURN;
656>              END;
657>          CALL INCBACK;
658>          CALL PUTDEST(B:=GETMEM(BACK));
659>          IF B = LF THEN RETURN;
660>          END;
```



```
661>      END WRITELINE;
662>
663>WRHALF: PROCEDURE;
664>      /* WRITE LINES UNTIL AT LEAST HALF THE BUFFER IS EMPTY */
665>      CALL SETFF;
666>          DO WHILE DISTNZERO;
667>          IF HMAX >= (MAXM - BACK) THEN CALL ZERODIST; ELSE
668>          CALL WRITELINE;
669>          END;
670>      END WRHALF;
671>
672>WRITEOUT: PROCEDURE;
673>      /* WRITE LINES DETERMINED BY 'DISTANCE',
674>      CALLED FROM W AND E COMMANDS */
675>      DIRECTION = BACKWARD; FIRST = 1; LAST = BACK;
676>      CALL MOVER;
677>      IF DISTZERO THEN CALL WRHALF;
678>      /* DISTANCE = 0 IF CALL WRHALF */
679>          DO WHILE DISTNZERO;
680>          CALL WRITELINE;
681>          END;
682>      IF BACK < LAST THEN
683>          DO; DIRECTION = FORWARD; CALL MOVER;
684>          END;
685>      END WRITEOUT;
686>
687>CLEARMEM: PROCEDURE;
688>      /* CLEAR MEMORY BUFFER */
689>      CALL SETFF;
690>      CALL WRITEOUT;
691>      END CLEARMEM;
692>
693>TERMINATE: PROCEDURE;
694>      /* CLEAR BUFFERS */
695>      CALL CLEARMEM;
696>          DO WHILE (CHAR := GETSOURCE) <> ENDFILE;
697>          CALL PUTDEST(CHAR);
698>          END;
699>      CALL FINIS;
700>      END TERMINATE;
701>
702>
703>INSERT: PROCEDURE;
704>      /* INSERT CHAR INTO MEMORY BUFFER */
705>      IF FRONT = BACK THEN GO TO OVERFLOW;
706>      CALL PUTMEM(FRONT,CHAR); CALL INCFRONT;
707>      END INSERT;
708>
709>SCANNING: PROCEDURE BYTE;
710>      /* READ A CHARACTER AND CHECK FOR ENDFILE OR CR */
711>      RETURN NOT ((CHAR := READC) = ENDFILE OR
712>                  (CHAR = CR AND NOT INSERTING));
713>      END SCANNING;
714>
715>COLLECT: PROCEDURE;
716>      /* READ COMMAND BUFFER AND INSERT CHARACTERS INTO
717>      SCRATCH 'TIL NEXT CONTROL-Z OR CR FOR  FIND, NEXT, JUNT, OR
718>      SUBSTITUTE COMMANDS - FILL AT UBE AND INCREMENT UBE SO IT
719>      ADDRESSES NEXT EMPTY POSITION OF SCRATCH */
720>      SETSCR: PROCEDURE;
```

```
721>          SCRATCH(UBE) = CHAR;
722>          IF (UBE := UBE + 1) > LAST(SCRATCH) THEN GO TO OVERFLOW;
723>          END SETSCR;
724>          DO WHILE SCANNING;
725>          IF CHAR = CTLL THEN
726>              DO; CHAR = CR; CALL SETSCR;
727>              CHAR = LF;
728>              END;
729>          IF CHAR = 0 THEN GO TO RESET;
730>          CALL SETSCR;
731>          END;
732>      END COLLECT;
733>
734>FIND: PROCEDURE(PA,PB) BYTE;
735>      DECLARE (PA,PB) BYTE;
736>      /* FIND THE STRING IN SCRATCH STARTING AT PA AND ENDING AT PB */
737>      DECLARE J ADDRESS;
```



```
749>      RETURN MATCH;
750>      END FIND;
751>
752>SETFIND: PROCEDURE;
753>      /* SETUP THE SEARCH STRING FOR F,N, AND S COMMANDS */
754>      UBE = 0; CALL COLLECT; UBP = UBE;
755>      END SETFIND;
756>
757>CHKFOUND: PROCEDURE;
758>      /* CHECK FOR FOUND STRING IN F AND S COMMANDS */
759>      IF NOT FIND(0,UBP) THEN /* NO MATCH */ GO TO OVERCOUNT;
760>      END CHKFOUND;
761>
762>
763>
764>SETRFCB: PROCEDURE;
765>      /* PLACE CHAR INTO READ FILE CONTROL BLOCK AND INCREMENT */
766>      RFCB((RBP := RBP + 1) - 1) = CHAR;
767>      END SETRFCB;
768>
769>TYPELINES: PROCEDURE;
770>      DCL I ADDR;
771>      CALL SETLIMITS;
772>      IF GETMEM(FIRST - 1) = LF AND COLUMN <> 0 THEN
773>          DO; CALL PRINTC(CR); CALL PRINTC(LF);
774>          END;
775>          DO I = FIRST TO LAST;
776>          IF BREAK$KEY THEN
777>              GO TO RESET;
778>          CALL PRINTC(GETMEM(I));
779>          END;
780>      END TYPELINES;
781>
782>SETLPP: PROCEDURE;
783>      /* SET DISTANCE TO LINES PER PAGE */
784>      DISTANCE = LPP;
785>      END SETLPP;
```

```
786>
787>SAVEDIST: PROCEDURE;
788>      TDIST = DISTANCE;
789>      END SAVEDIST;
790>
791>RESTDIST: PROCEDURE;
792>      DISTANCE = TDIST;
793>      END RESTDIST;
794>
795>^PAGE: PROCEDURE;
796>      DECLARE I BYTE;
797>      CALL SAVEDIST;
798>      CALL SETLPP;
799>      CALL MOVELINES;
800>      I = DIRECTION;
801>      DIRECTION = FORWARD;
802>      CALL SETLPP;
803>      CALL TYPELINES;
804>      DIRECTION = I;
805>      IF LAST = MAXN OR FIRST = 1 THEN CALL ZERODIST;
806>          ELSE CALL RESTDIST;
807>      END PAGE;
808>
809>WAIT: PROCEDURE;
810>      /* 1/2 SECOND TIME OUT */
811>      DECLARE I BYTE;
812>          DO I = 0 TO 19;
813>          IF BREAK$KEY THEN GO TO RESET;
814>          CALL TIME(250);
815>          END;
816>      END WAIT;
817>
818>SETFORWARD: PROCEDURE;
819>      DIRECTION = FORWARD;
820>      DISTANCE = 1;
821>      END SETFORWARD;
822>
823>AAPPHALF: PROCEDURE;
824>      /* APPEND 'TIL BUFFER IS AT LEAST HALF FULL */
825>      CALL SETFF; /* DISTANCE = 0FFFFH */
826>          DO WHILE DISTNZERO;
827>          IF FRONT >= HMAX THEN CALL ZERODIST; ELSE
828>          CALL READLINE;
829>          END;
830>      END APPHALF;
831>
832>INSCRLF: PROCEDURE;
833>      /* INSERT CR LF CHARACTERS */
834>      CHAR = CR; CALL INSERT;
835>      CHAR = LF; CALL INSERT;
836>      END INSCRLF;
837>
838>TESTCASE: PROCEDURE;
839>      DECLARE T BYTE;
840>      /* TEST FOR UPPER OR LOWER CASE COMMAND AND SET TRANSLATE
841>      FLAG (USED TO DETERMINE IF CHARACTERS WHICH FOLLOW GO TO UPPER */
842>      TRANSLATE = TRUE;
843>      T = LOWERCASE(CHAR);
844>      CHAR = UTRAN(CHAR);
845>      TRANSLATE = UPPER OR NOT T;
```

```
846)     END TESTCASE;
847)
848)READCTRAN: PROCEDURE;
849)     /* SET TRANSLATE TO FALSE AND READ NEXT CHARACTER */
850)     TRANSLATE = FALSE;
851)     CHAR = READC;
852)     CALL TESTCASE;
853)     END READCTRAN;
854)
855)SINGLECOM: PROCEDURE(C) BYTE;
856)     /* RETURN TRUE IF COMMAND IS ONLY CHARACTER, NOT IN MACRO */
857)     DECLARE C BYTE;
858)     RETURN CHAR = C AND COMLEN = 1 AND MP
859)     END SINGLECOM;
860)
861)
862)
863)
864)
865)
866)
867)  /* INITIALIZE THE SYSTEM */
868)     MAX = MAXB - .MEMORY - 1;
869)     MEMORY(MAX) = 0; /* STOPS MATCH AT END OF BUFFER */
870)     MAXM = MAX - 1;
871)     HMAX = SHR(MAXM,1);
872)     /* INITIALLY ASSUME NO TRANSLATE TO UPPER CASE */
873)     UPPER = FALSE;
874)     /* GET SOURCE AND DESTINATION DISKS */
875)     IF (FCB(1) = ' ') OR (FCB(17) <> ' ') THEN CALL FERR;
876)     IF (SDISK := FCB) = 0 THEN SDISK = CSELECT; ELSE
877)         DO; SDISK = SDISK - 1; FCB = 0; /* CLEAR DISK NAME */
878)         END;
879)     IF (DDISK := FCB(16)) = 0 THEN DDISK = SDISK; ELSE
880)         DDISK = DDISK - 1;
881)
882)RESTART:
883)     CALL SETUP;
884)     MEMORY = LF;
885)     FRONT = 1; BACK = MAXM;
886)     COLUMN = 0;
887)     GO TO START;
888)
889)OVERCOUNT: FLAG = POUND; GO TO RESET;
890)
891)BADCOM: FLAG = WHAT; GO TO RESET;
892)
893)OVERFLOW: /* ARRIVE HERE ON OVERFLOW CONDITION (I,F,S COMMAND) */
894)     FLAG = '>';
895)
896)RESET: /* ARRIVE HERE ON SYSTEM RESET OR OVERFLOW CONDITION */
897)     CALL PRINTC(CR); CALL PRINTC(LF);
898)     CALL PRINTC(CHAR);
899)     CALL PRINTC(FLAG);
900)     CALL CRLF;
901)
902)START:
903)     READBUFF = TRUE;
904)     MP = 0;
905)
```



```
906)
907)     DO FOREVER; /* OR UNTIL THE POWER IS TURNED OFF */
908)
909)     /* ************************************************************
910)     SIMPLE COMMANDS (CANNOT BE PRECEDED BY DIRECTION/DISTANCE);
911)             E       END THE EDIT NORMALLY
912)             H       MOVE TO HEAD OF EDITED FILE
913)             I       INSERT CHARACTERS
914)             O       RETURN TO THE ORIGINAL FILE
915)             R       READ FROM LIBRARY FILE
916)             Q       QUIT EDIT WITHOUT CHANGES TO ORIGINAL FILE
917)        ************************************************************ */
918)
919)
920)
921)     INSERTING = FALSE;
922)     CALL READCTRAN;
923)     MI = CBP; /* SAVE STARTING ADDRESS FOR <CR> COMMAND */
924)     IF SINGLECOM('E') THEN
925)         DO; CALL TERMINATE;
926)         IF SDISK <> DDISK THEN BDISK = DDISK;
927)         GO TO BOOT;
928)         END; ELSE
929)
930)
931)     IF SINGLECOM('H') THEN /* GO TO TOP */
932)         DO; CALL TERMINATE;
933)         CHAR = DDISK; DDISK = SDISK; SDISK = CHAR;
934)         /* PING - PONG DISKS */
935)         GO TO RESTART;
936)         END; ELSE
937)
938)
939)     IF CHAR = 'I' THEN /* INSERT CHARACTERS */
940)         DO;
941)         INSERTING = (CBP = COMLEN) AND (MP = 0);
942)         DO WHILE SCANNING;
943)             DO WHILE CHAR <> 0;
944)             IF CHAR = CTLU THEN /* LINE DELETE */
945)                 DO; CALL PRINTC(CR); CALL PRINTC(LF);
946)                 /* ELIMINATE LINE */ DIRECTION = BACKWARD;
947)                 DISTANCE = 0; CALL SETLIMITS; CALL SETPTRS;
948)                 END; ELSE
949)             IF CHAR = RUBOUT THEN
950)                 DO; IF FRONT = 1 THEN GO TO RESET;
951)                 CALL DECFRONT; CALL PRINTC(GETMEM(FRONT));
952)                 END; ELSE
953)             IF CHAR = CTLL THEN CALL INSCRLF; ELSE
954)                 CALL INSERT;
955)             IF CHAR = CR THEN
956)                 CALL PRINTMAC(CHAR,*LF); ELSE CHAR=0;
957)             END;
958)         END;
959)         IF CHAR <> ENDFILE THEN /* MUST HAVE STOPPED ON CR */
960)             CALL INSCRLF;
961)         END; ELSE
962)
963)
964)     IF SINGLECOM('O') THEN /* FORGET THIS EDIT */
965)         GO TO RESTART; ELSE
```

```
966)
967)
968)     IF CHAR = 'R' THEN
969)         DO; /* READ FROM 'LIB' FILE */
970)         TRANSLATE = TRUE;
971)         RBP = 1;
972)             DO WHILE SCANNING;
973)             IF RBP > 8 THEN GO TO OVERCOUNT;
974)             CALL SETRFCB;
975)             END;
976)         CHAR = ' ';
977)             DO WHILE RBP <= 8;
978)             CALL SETRFCB;
979)             END;
980)         RFCB(12), RFCB(32) = 0; /* FILL REEL, AND NEXT RECORD */
981)         CALL OPEN( RFCB); RBP = 80H;
982)         IF DCNT = 255 THEN
983)             DO; FLAG = '0'; GO TO RESET;
984)             END;
985)             DO WHILE (CHAR := READFILE) <>  ENDFILE;
986)             IF UPPER THEN CHAR = UTRAN(CHAR);
987)             CALL INSERT;
988)             END;
989)         END; ELSE
990)
991)
992)     IF SINGLECOM('Q') THEN
993)         DO; CALL DELETE( DFCB); GO TO BOOT;
994)         END; ELSE
995)
996)
997)
998)     /* MAY BE A COMMAND WHICH HAS AN OPTIONAL DIRECTION AND DISTANCE */
999)         DO; /* SCAN A SIGNED INTEGER VALUE (IF ANY) */
1000)        DCL I BYTE;
1001)        CALL SETFORWARD;
1002)
1003)        IF CHAR = '-' THEN
1004)            DO; CALL READCTRAN; DIRECTION = BACKWARD;
1005)            END;
1006)
1007)        IF CHAR = POUND THEN
1008)            DO; CALL SETFF; CALL READCTRAN;
1009)            END; ELSE
1010)
1011)        IF (I:=CHAR-'0') <= 9 THEN
1012)            DO; DISTANCE = I; CALL READCTRAN;
1013)                DO WHILE (I := CHAR - '0') <= 9;
1014)                DISTANCE = SHL(DISTANCE,3) + SHL(DISTANCE,1) + I;
1015)                CALL READCTRAN;
1016)              END;
1017)            END;
1018)        IF DISTZERO THEN DIRECTION = BACKWARD;
1019)        /* DIRECTION AND DISTANCE ARE NOW SET */
1020)
1021)
1022)    /* ************************************************************
1023)    MAY BE A COMMAND WHICH HAS DIRECTION AND DISTANCE SPECIFIED.
1024)             B       BEGINNING/BOTTOM OF BUFFER
1025)             C       MOVE CHARACTER POSITIONS
1026)             D       DELETE CHARACTERS
1027)             K       KILL LINES
1028)             L       MOVE LINE POSITION
1029)             P       PAGE UP OR DOWN (LPP LINES AND PRINT)
1030)             T       TYPE LINES
1031)             U       OPPOSITE L
1032)            <CR>     MOVE UP OR DOWN LINES AND PRINT LINE
1033)    ************************************************************ */
1034)
1035)
1036)    IF CHAR = 'B' THEN
1037)        DO; DIRECTION = 1 - DIRECTION;
1038)        FIRST = 1; LAST = MAXM; CALL MOVER;
1039)        END; ELSE
1040)
1041)
1042)    IF CHAR = 'C' THEN
1043)        DO; CALL SETCLIMITS; CALL MOVER;
1044)        END; ELSE
1045)
1046)
1047)    IF CHAR = 'D' THEN
1048)        DO; CALL SETCLIMITS;
1049)        CALL SETPTRS; /* SETS BACK/FRONT */
1050)        END; ELSE
1051)
1052)
1053)    IF CHAR = 'K' THEN
1054)        DO; CALL SETLIMITS;
1055)        CALL SETPTRS;
1056)        END; ELSE
1057)
1058)
1059)    IF CHAR = 'L' THEN CALL MOVELINES; ELSE
1060)
1061)    IF CHAR = 'P' THEN /* PAGE MODE PRINT */
1062)        DO; IF DISTZERO THEN
1063)            DO; DIRECTION = FORWARD;
1064)            CALL SETLPP; CALL TYPELINES;
1065)            END; ELSE
1066)            DO WHILE DISTNZERO; CALL PAGE;
1067)            CALL WAIT;
1068)            END;
1069)        END; ELSE
1070)
1071)
1072)    IF CHAR = 'T' THEN
1073)        CALL TYPELINES; ELSE
1074)
1075)
1076)
1077)    IF CHAR = 'U' THEN
1078)        UPPER = DIRECTION = FORWARD; ELSE
1079)
1080)
1081)    IF CHAR = CR THEN /* MAY BE MOVE/TYPE COMMAND */
1082)        DO;
1083)        IF MI = 1 AND MP = 0 THEN /* FIRST COMMAND */
1084)            DO; CALL MOVELINES; CALL SETFORWARD; CALL TYPELINES;
1085)            END;
```


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

```
1006>           END; ELSE
1007>
1008>       IF DIRECTIONN = FORWARD OR DISTZERO THEN
1009>       DO;
1010>
1011>   /* ************************************************************
1012>   COMMANDS WHICH ALLOW ONLY A PRECEDING NUMBER:
1013>            A       APPEND LINES
1014>            F       FIND NTH OCCURRENCE
1015>            M       APPLY MACRO
1016>            N       SAME AS F WITH AUTOSCAN THROUGH FILE
1017>            S       PERFORM N SUBSTITUTIONS
1018>            W       WRITE LINES TO OUTPUT FILE
1019>      ************************************************************ */
1020>
1021>
1022>
1023>       IF CHAR = 'A' THEN
1024>           DO;
1025>           FIRST = FRONT; LAST = MAXM; CALL MOVER;
1026>           /* ALL STORAGE FORWARD */
1027>           IF DISTZERO THEN CALL APPHALF;
1028>           /* DISTANCE = 0 IF APPHALF CALLED */
1029>               DO WHILE DISTNZERO;
1030>               CALL READLINE;
1031>               END;
1032>           DIRECTION = BACKWARD; CALL MOVER;
1033>           /* POINTERS REPOSITIONED */
1034>           END; ELSE
1035>
1036>
1037>       IF CHAR = 'F' THEN
1038>           DO; CALL SETFIND; /* SEARCH STRING SCANNED
1039>           AND SETUP BETWEEN 0 AND UBP-1 IN SCRATCH */
1040>               DO WHILE DISTNZERO; CALL CHKFOUND;
1041>               END;
1042>           END; ELSE
1043>
1044>
1045>       IF CHAR = 'J' THEN /* JUXTAPOSITION OPERATION */
1046>           DO; DECLARE T ADDRESS;
1047>           CALL SETFIND; CALL COLLECT;
1048>           UBJ = UBE; CALL COLLECT;
1049>           /* SEARCH FOR STRING 0 - UBP-1, INSERT STRING UBP TO UBJ-1,
1050>           AND THEN DELETE UP TO STRING UBJ TO UBE-1 */
1051>           DO WHILE DISTNZERO; CALL CHKFOUND;
1052>           /* INSERT STRING */ MI = UBP - 1;
1053>               DO WHILE (MI := MI + 1) < UBJ;
1054>               CHAR = SCRATCH(MI); CALL INSERT;
1055>               END;
1056>           T = FRONT; /* SAVE POSITION FOR DELETE */
1057>           IF NOT FIND(UBJ,UBE) THEN GO TO OVERCOUNT;
1058>           /* STRING FOUND, SO MOVE IT BACK */
1059>           FIRST = FRONT - (UBE - UBJ);
1060>           DIRECTION = BACKWARD; CALL MOVER;
1061>           /* NOW REMOVE THE INTERMEDIATE STRING */
1062>           FRONT = T;
1063>           END;
1064>       END; ELSE
1065>
1066>
```



```
1147>       IF CHAR = 'M' AND MP = 0 THEN /* MACRO DEFINITION */
1148>           DO; XP = 255;
1149>           IF DISTANCE = 1 THEN CALL ZERODIST;
1150>               DO WHILE (MACRO(XP := XP + 1) = READC) <> CR;
1151>               END;
1152>           MP = XP; XP = 0; MT = DISTANCE;
1153>           END; ELSE
1154>
1155>
1156>       IF CHAR = 'N' THEN
1157>           DO; /* SEARCH FOR STRING WITH AUTOSCAN */
1158>           CALL SETFIND; /* SEARCH STRING SCANNED */
1159>               DO WHILE DISTNZERO;
1160>               /* FIND ANOTHER OCCURRENCE OF STRING */
1161>                   DO WHILE NOT FIND(0,UBP); /* NOT IN BUFFER */
1162>                   IF BREAK$KEY THEN GO TO RESET;
1163>                   CALL SAVEDIST; CALL CLEARMEM;
1164>                   /* MEMORY BUFFER WRITTEN */
1165>                   CALL APPHALF;
1166>                   DIRECTION = BACKWARD; FIRST = 1; CALL MOVER;
1167>                   CALL RESTDIST; DIRECTION = FORWARD;
1168>                   /* MAY BE END OF FILE */
1169>                   IF BACK >= MAXM THEN GO TO OVERCOUNT;
1170>                   END;
1171>               END;
1172>           END; ELSE
1173>
1174>
1175>       IF CHAR = 'S' THEN /* SUBSTITUTE COMMAND */
1176>           DO; CALL SETFIND;
1177>           CALL COLLECT;
1178>           /* FIND STRING FROM 0 TO UBP-1, SUBSTITUTE STRING
1179>           BETWEEN UBP AND UBE-1 IN SCRATCH */
1180>               DO WHILE DISTNZERO;
1181>               CALL CHKFOUND;
1182>               /* FRONT AND BACK NOW POSITIONED AT FOUND
1183>               STRING - REPLACE IT */
1184>               FRONT = FRONT - (MI := UBP); /* BACKED UP */
1185>                   DO WHILE MI < UBE;
1186>                   CHAR = SCRATCH(MI);
1187>                   MI = MI + 1; CALL INSERT;
1188>                   END;
1189>               END;
1190>           END; ELSE
1191>
1192>
1193>       IF CHAR = 'W' THEN
1194>           CALL WRITEOUT; ELSE
1195>
1196>       IF CHAR = 'Z' THEN /* SLEEP */
1197>           DO;
1198>           IF DISTZERO THEN
1199>               DO; IF READCHAR = ENDFILE THEN GO TO RESET;
1200>               END;
1201>               DO WHILE DISTNZERO; CALL WAIT;
1202>               END;
1203>           END; ELSE
1204>       IF CHAR <> 0 THEN /* NOT BREAK LEFT OVER FROM STOP */
1205>       /* DIRECTION FORWARD, BUT NOT ONE OF THE ABOVE */
1206>       GO TO BADCOM;
```

```
1207>
1208>
1209>          END; ELSE /* DIRECTION NOT FORWARD */
1210>             GO TO BADCOM;
1211>          END;
1212>       END;
1213>EOF
```

```
317>MEMORY.....1F00H
318>COPYRIGHT.....0106H
319>MAYA.....14C2H
320>MAX.....14C4H
321>MAXM.....14C6H
322>NHMAX.....14C8H
323>BDISKA.....14CAH
324>FCBA.....14CCH
325>BUFA.....14CEH
326>RFCB.....14D1H
327>PBP.....14F2H
328>SBUFF.....14F3H
329>NDFCB.....18F3H
330>NDBUFF.....1914H
331>NSOURCE.....1D14H
332>NDEST.....1D16H
333>SDISK.....1D19H
334>DDDISK.....1D1AH
335>MONI.....012AH
336>F.....1D1BH
337>AA.....1D1CH
338>MON2.....013CH
339>F.....1D1FH
340>AA.....1D20H
341>READCHAR.....0142H
342>PPRINTCHAR.....014CH
343>CHAR.....1D23H
344>CRLF.....0159H
345>PPRINT.....0164H
346>AA.....1D24H
347>READ.....0179H
348>AA.....1D26H
349>NDCNT.....1D28H
350>OPEN.....0188H
351>FCB.....1D2AH
352>CLOSE.....019DH
353>FCB.....1D2CH
354>SEARCH.....01B1H
355>FCB.....1D2EH
356>NDELETE.....01C5H
357>FCB.....1D30H
358>NDISKREAD.....01D5H
359>FCB.....1D32H
360>NDISKWRITE.....01E5H
361>FCB.....1D34H
362>MAKE.....01F5H
363>FCB.....1D36H
364>RENAME.....0209H
365>FCB.....1D38H
366>MAXLEN.....1D3BH
367>COMLEN.....1D3CH
368>COMBUFF.....1D3DH
369>TCBP.....1D5DH
370>CBP.....1D8EH
371>READCOM.....0219H
372>NBREAKKEY.....0225H
373>CSELECT.....0240H
374>SELECT.....024AH
375>NDISK.....1DBFH
376>SETDMA.....0257H
377>AA.....1DC0H
378>LIFTHEAD.....0267H
379>MOVE.....0271H
380>S.....1DC2H
381>ND.....1DC4H
382>N.....1DC7H
383>FERR.....029EH
384>SETTYPE.....02BCH
385>AA.....1DC8H
386>SETUP.....02DDH
387>FILLSOURCE.....03A8H
388>I.....1DC8H
389>ZN.....03A3H
390>GETSOURCE.....0413H
391>B.....1DCCH
392>WRITEDEST.....0440H
393>I.....1DCDH
394>N.....1DCEH
395>ZN.....0443H
396>PPUTDEST.....04B0H
397>B.....1DCFH
398>FINIS.....04DAH
399>MOVEUP.....04DDH
400>MACRO.....1DD0H
401>SCRATCH.....1E50H
402>WBP.....1EB4H
403>WBE.....1EB5H
404>WBJ.....1EB6H
405>FLAG.....1EB7H
406>MP.....1EB8H
407>MI.....1EB9H
408>XP.....1EBAH
409>COLUMN.....1EBBH
410>MT.....1EBCH
411>START.....0E83H
412>RESTART.....0E60H
413>OVERCOUNT.....0E83H
414>OVERFLOW.....0E93H
415>RESET.....0E98H
416>BADCOM.....0E88H
417>INSERTING.....1EBFH
418>READBUFF.....1EC0H
419>TTYCHAR.....0569H
420>CHAR.....1EC1H
421>PPRINTC.....0589H
422>CHAR.....1EC2H
423>I.....1EC3H
424>J.....1EC4H
425>^PRINTNMAC.....05C7H
426>CHAR.....1EC5H
427>TRANSLATE.....1EC6H
428>UPPER.....1EC7H
429>LOWERCASE.....05D9H
430>C.....1EC8H
431>UTRAN.....05E9H
432>C.....1EC9H
433>READC.....0602H
434>READFILE.....06BEH
435>GETMEM.....06F0H
436>I.....1ECAH
437>PPUTMEM.....0700H
438>I.....1ECCH
439>J.....1ECFH
440>TRANSFER.....071AH
441>I.....1ED0H
442>J.....1ED2H
443>DDISTANCE.....1ED4H
444>TDIST.....1ED6H
445>NDIRECTION.....1ED8H
446>CHAR.....1ED9H
447>FRONT.....1EDAH
448>BACK.....1EDCH
449>FIRST.....1EDEH
450>LAST.....1EE0H
451>SETFF.....0736H
452>DDISTZERO.....073FH
453>ZERODIST.....0750H
454>NDISTNZERO.....0759H
455>SETLIMITS.....076DH
456>I.....1EE2H
457>K.....1EE4H
458>L.....1EE6H
459>M.....1EE8H
460>MIDDLE.....1EEAH
461>LOOPING.....1EECH
462>SETPTRS.....0875H
463>INCFRONT.....0895H
464>INCBACK.....089DH
465>NDECFRONT.....08A5H
466>NDECBACK.....08ADH
467>MOVER.....08B5H
468>MOVELINES.....0908H
469>SETCLIMITS.....0912H
470>READLINE.....0993H
471>B.....1EEDH
472>CTRAN.....0996H
473>B.....1EEEH
474>WRITELINE.....09EEH
475>B.....1EEFH
476>URHALF.....0A22H
477>WRITEOUT.....0A55H
478>CLEARMEM.....0A9AH
479>TERMINATE.....0AA1H
480>INSERT.....0AB8H
481>SCANNING.....0AE0H
482>COLLECT.....0AFEH
483>SETSCR.....0B01H
484>FIND.....0B5DH
485>PPA.....1EF0H
486>PPB.....1EF1H
487>J.....1EF2H
488>K.....1EF4H
489>MATCH.....1EF5H
490>SETFIND.....0BE8H
491>CHKFOUND.....0BF5H
492>SETRFCB.....0C0EH
493>TYPELINES.....0C21H
494>I.....1EF6H
495>SETLPP.....0C92H
496>SAVEDIST.....0C9E
497>RESTDIST.....0CA6
498>PPAGE.....0CB2
499>I.....1EF9
500>WAIT.....0D2C
501>I.....1EFA
502>SETFORWARD.....0D32
503>AAPPHALF.....0D7F
504>INSCRLF.....0D87
505>TESTCASE.....0D78
506>T.....1EFB
507>READCTRAN.....0D95
508>SINGLECOM.....0DAB
509>C.....1EFC
510>I.....1EFD
511>T.....1EFE
```