```
%BEGIN      ;!   LEVEL 0
!   ** DEFINE SYMBOLIC I/O STREAMS **
%OWNINTEGER  COMMAND=0, REPORT=0, MACROS=5
%OWNINTEGERARRAY  INSTREAM(1:4)= 1,2,3,4
%OWNINTEGERARRAY  OUTSTREAM(1:4)= 1,2,3,4
!
%OWNINTEGER  BBOT=0, W=0, X=0, Y=512, Z=512, STAR=10001
%INTEGERARRAY  B(BBOT:Z-1)       ;! COMMAND BUFFER + MACRO DEFNS.
%OWNINTEGER  STKTOP=100
%INTEGERARRAY  ST(0:STKTOP)      ;!  DPDA STACK
! DEFINE FUNCTION CODES FOR NEST=( AND UN-NEST=) OPERATIONS
%OWNINTEGER  LBRCODE=1, RBRCODE=3  ;!  NOTLBR=2
!
%OWNINTEGER  AMIN=1, AMAX = 1000
   AMAX=FREESTORE-800
%INTEGERARRAY  AA(AMIN:AMAX)     ;!  EDITOR TEXT BUFFER
!
!
   %FAULT 30 -> TEST             ;!  TRAP EXECUTION TIME ERROR
   %FAULT 31 -> CLEAR TTY        ;!  TRAP DECODE TIME ERROR
!
   %ROUTINESPEC INITIALISE
   INITIALISE; %PRINTTEXT 'EDIT5B   10/4/72'; NEWLINE
   -> READ NEW COMMAND
!
!
!
!
!
!
!
!
!
!
%ROUTINE  ERROR(%INTEGER N)   ;!  DIAGNOSE ERRORS
%SWITCH  S(0:14)
   %ROUTINE  PRINT STRING
      PRINT SYMBOL( B(W+N) ) %FOR N = 1,1,(B(W)&8_177)-1
      SPACE
   %END
!
   SELECT OUTPUT(REPORT)
   -> S(N) %IF N >= 0
   %PRINTTEXT '*ERROR '; WRITE(N,1); -> END
!
S(0): PRINT STRING; -> QUERY
S(1): PRINT STRING; %PRINTTEXT 'NOT ID.'; -> END
S(2): PRINT STRING; %PRINTTEXT '#N, 1<=N<=4'; -> QUERY
S(3): PRINT STRING; -> S7
S(4): PRINT STRING; %PRINTTEXT ''''-'' IN MACRO'; -> QUERY
S(5): PRINT STRING; %PRINTTEXT '/TEXT/'; -> QUERY
S(6): %PRINTTEXT '['; -> QUERY
S(7):S7: %PRINTTEXT 'PARAM. TYPE'; -> QUERY
S(8): %PRINTTEXT ''''=/'' OR ''']'' MISSING'; -> QUERY
S(9): PRINT STRING; %PRINTTEXT 'IS PROTECTED'; -> END
S(10): PRINT STRING; -> S13
S(11): PRINT STRING; %PRINTTEXT '-'; -> QUERY
S(12): %PRINTTEXT 'BRACKETS'; -> QUERY
S(13):S13: %PRINTTEXT 'TOO LONG'; -> END
S(14): %PRINTTEXT '!'
```

```
!
QUERY: %PRINTTEXT ' ?'
END:   NEWLINE
   %MONITOR 31
%END
!
%ROUTINE  MOVE(%INTEGERARRAYNAME  AA,%INTEGER  P,Q,N)
!  MOVE AA(P:Q) TO AA(P+N:Q+N)
   %IF N > 0 %THEN:
      AA(P+Q+N) = AA(P+Q) %FOR  Q = Q-P,-1,0
   %ELSE:
      AA(P+Q+N) = AA(P+Q) %FOR  Q = 0,1,Q-P
   %FINISH
%END
!
!
!
!
!
!
!  ** THE COMPLETE COMMAND DECODER IS CONTAINED IN THE FOLLOWING
!     %BEGIN - %END  BLOCK **
!
CLEAR TTY:
      %IF INPUT # 0 %THEN:
         SKIP SYMBOL %UNTIL NEXT SYMBOL = NL
      %FINISH
READ NEW COMMAND:
   %FAULT9 -> CLEAR TTY
!
%BEGIN   ;!  LEVEL 1
%INTEGER  LVL, BRACKETS, ESCAPE CELL, MODIFIER,LAST TEXT
%OWNINTEGER  NAME MASK=8_177, BODY MASK=8_377
%OWNINTEGER  QUERY=8_400000, INVERT=8_200000, DUMMY=8_100000
%OWNINTEGER  PROTECT = 8_200, RERUN = 0
!                                  A,B, C,D, E, F, G, H,I, J, K,
%OWNINTEGERARRAY  PRIM('A':'Z')= 6,7,-1,8,11,-1,14,-1,9,-1,-1,
!     L, M, N, O, P, Q, R, S, T, U,V, W, X, Y, Z
     -1,12,-1,-1,13,-1,-1,10,-1,15,5,-1,-1,-1,16
!
%ROUTINESPEC    EXPAND(%INTEGER  SOURCE BODY,MACRO DEFNS,PARM BASE,STP)
!
L1:PROMPT('>'); SKIP SYMBOL %AND -> L1 %IF NEXT SYMBOL = NL
   %IF PROTECT # 0 %AND NEXT SYMBOL = '*' %THEN:
      SKIP SYMBOL; SKIP SYMBOL    ;! PROTECTED DEFINITIONS ARE...
                                  ;! ...TERMINATED BY NL-*-NL
      PROTECT = 0; SELECT INPUT(COMMAND)
   %ELSE:
      W = BBOT; X = Y-1; ESCAPE CELL = Z
      LVL = -1; BRACKETS = 0; LAST TEXT = 0
      EXPAND(0,DUMMY+Y,-4,0)
      ERROR(12) %IF BRACKETS # 0     ;!  USER TYPING ERROR
   %FINISH
   -> L1 %IF LVL = 0  ;! IF IT WAS A MACRO DEFN.
                      ;! (SEE 'CONTROL(6):' IN ROUTINE 'EXPAND')
   RERUN = 1          ;! DECODER BUFFER HOLDS COMPLETE COMMAND
!
%ROUTINE  EXPAND(%INTEGER SOURCE BODY,MACRO DEFNS,PARM BASE,STP)
%INTEGER  P,NEXT
   %ROUTINE  WPERROR(%INTEGER  N); W = P; ERROR(N); %END
```

```
121     %OWNINTEGER  I,J,K,Q,R,PARM
122     %OWNINTEGER  CODE WORD, DELIMITER
123     %SWITCH  SW(1:3), CONTROL(0:6)
124     %ROUTINESPEC  READ NUMBER(%INTEGERNAME X)
125     !
126        %INTEGERFN  STRING DELIMITER
127           %RESULT = NEXT %IF NEXT='/' %OR NEXT='.' %C
128                                       %OR NEXT=':' %OR NEXT=''''
129           %RESULT = \NEXT
130        %END
131     !
132        %ROUTINE  SET NEXT
133           %IF LVL = 0 %THEN:
134              READ SYMBOL(NEXT) %IF NEXT # NL
135           %ELSE:
136              %IF SOURCE BODY # MACRO DEFNS&8 77777 %THEN:
137                 NEXT = B(SOURCE BODY); SOURCE BODY = SOURCE BODY+1
138              %ELSE  NEXT = NL
139           %FINISH
140        %END
141     !
142        %ROUTINE  REMAP ALPHA
143           NEXT = NEXT-32 %IF 'A' <= NEXT-32 <= 'Z'
144        %END
145     !
146        %PREDICATE  STRING IS EMPTY
147           %TRUE %IF NEXT = NL
148           %FALSE
149        %END
150        !
151        %ROUTINE  PUSHBR(%INTEGER  TYPE)
152           STP = STP+4; BRACKETS = BRACKETS+1
153           ERROR(13) %IF STP > STKTOP
154           ! CHAIN ESCAPE CELL ONTO LIST
155           B(X-3) = ESCAPE CELL; ESCAPE CELL = X-3
156           ST(STP+1) = LVL
157           ST(STP+2) = TYPE
158           ST(STP+3) = X
159           X = X-4; ERROR(13) %IF X < W
160        %END
161        !
162        %ROUTINE  PLANT(%INTEGER  I)
163           ERROR(13) %IF X <= W
164           B(X) = I; X = X-1
165        %END
166        !
167        %ROUTINE  POPBR(%INTEGER  TYPE,REPEAT)
168        %INTEGER  K,L,M
169           ! TEST FOR DECODER STACKING ERROR
170           ERROR(-1) %IF STP < 0
171           BRACKETS = BRACKETS-1
172           PLANT(RBRCODE)
173           TYPE = ST(STP+2)!TYPE
174           K = ST(STP+3)
175           %IF TYPE&QUERY # 0 %OR BRACKETS = 0 %THEN:
176              M = 0; M = -(BRACKETS+1) %IF TYPE&QUERY # 0
177              %WHILE ESCAPE CELL < K %DO:
178                 L = ESCAPE CELL; ESCAPE CELL = B(L)
179                 B(L) = M
180              %REPEAT
```

```
181         %FINISH
182         L = LBRCODE; L = L+1 %IF TYPE&INVERT # 0
183         B(K) = L          ;!  PLANT CODE FOR '(' OR '\('
184         B(K-1) = X        ;!  LENGTH CELL
185         B(K-2) = REPEAT   ;!  LOOP COUNT
186         STP = STP-4
187      %END
188      !
189      !
190      !
191      !
192      !
193      !
194      !
195      !
196      !
197      !
198      !
199      !
200      !
201      ! USE 'PUSHBR' TO INSERT AN EXTRA LAYER OF BRACKETS AROUND
202      ! THE STRING CURRENTLY BEING EXPANDED IF:
203      ! 1) THE STRING IS THE COMMAND STRING FROM THE USER RATHER
204      !     THAN  A PARTIALLY EXPANDED VERSION OF IT.
205      ! 2) '\' OR '?' TO QUALIFIED THE CALL ON A MACRO. E.G. D3?
206      !     THIS EXTRA LAYER TO AVOIDS INTERACTION BETWEEN LEVELS.
207      ! FLAG BIT IN 'MACRO DEFNS' MARKS THIS EXTRA LEVEL SO THAT
208      ! IT CAN BE POPPED ON EXIT FROM ROUTINE.
209      LVL = LVL+1
210      MODIFIER = 0
211      PUSHBR(MACRO DEFNS) %IF MACRO DEFNS & 8 700000 # 0
212      NEXT = ' '          ;!   CONTENTS ARE CURRENTLY UNDEFINED
213      %UNTIL STRING IS EMPTY %DO;
214         SET NEXT %WHILE NEXT = ' '
215   %INTEGERFN CHAR
216   %OWNINTEGERARRAY AA(1:6)= ')', '(', ',', '\', '!', '['
217   %INTEGER I
218      %FOR I=1,1,6 %DO:
219         %RESULT = I %IF AA(I) = NEXT
220      %REPEAT
221      %RESULT = 0
222   %END
223         -> CONTROL(CHAR)
224   !
225   !
226   !
227   !
228   !
229   CONTROL(0):      ! ** READ IN COMMAND ATOM IDENTIFIER **
230   %ROUTINE  READ COMMAND ATOM ID
231      RERUN = 0          ;!  PREVIOUS COMMAND NOW BECOMES UNREUSABLE
232      %IF NEXT = '$' %THEN:
233         I = W
234         %UNTIL %NOT 'A' <= NEXT <= 'Z' %DO:
235            I = I+1; B(I) = NEXT
236            SET NEXT; REMAP ALPHA
237         %REPEAT
238      %ELSE:
239         REMAP ALPHA
240         I = W+1; B(I) = NEXT; SET NEXT
```

```
241         %FINISH
242         B(W) = (I-W+1) + PROTECT
243         ERROR(0) %UNLESS 'A' <= B(I) <= 'Z'
244         ! ** COULD INSERT PACKING ROUTINE HERE:  MUST RESET B(W) **
245     %END
246               READ COMMAND ATOM ID
247               !
248               ! ** NOW LOOK IT UP IN DEFN. TABLE **
249     %ROUTINE  LOOK UP
250         P = MACRO DEFNS&8_77777; J = B(W) & NAME MASK
251         %WHILE  P # Z %DO:
252             %IF B(P) & NAME MASK = J %THEN:
253                 %FOR K = 1,1,J-1 %DO:
254                     -> L1 %IF B(P+K) # B(W+K)
255                 %REPEAT
256                 ! IDENTIFIER FOUND:  SET P,Q AND R TO POINT TO THE
257                 ! HEAD OF NAME, END OF NAME/HEAD OF BODY AND END OF
258                 ! BODY RESPECTIVELY.
259                 Q = (B(P)&NAME MASK) + P
260                 R = (B(Q)&BODY MASK) + Q
261                 %RETURN
262                 !
263             %ELSE:
264     L1:         P = ( B(P) & NAME MASK ) + P
265                 P = ( B(P) & BODY MASK ) + P
266             %FINISH
267         %REPEAT
268         ! IDENTIFIER ISN'T IN TABLE.
269         ! INDEX INTO THE PRIMITIVE FNS. TABLE IF:
270         ! 1) IDENTIFIER IS A SINGLE LETTER STRING, AND:
271         ! 2) IT CAME FROM A MACRO BODY, NOT THE TELETYPE DIRECTLY.
272         !    FAILURE OTHERWISE.
273         ! SET Q = P TO IDENTIFY PRIMITIVES IN THE OUTSIDE WORLD.
274         !
275         %IF LVL > 0 %AND J = 2 %THEN P = PRIM(B(W+1)) %ELSE P = -1
276         Q = P
277     %END
278     !
279               LOOK UP; ERROR(1) %IF P < 0        ;! IDENTIFIER NOT FOUND
280               %IF P # Q %THEN:
281                   ! IT'S A MACRO CALL:  SET UP PARAMETERS USING
282                   ! CODE WORD THEN CALL 'EXPAND' RECURSIVELY.
283                   CODE WORD = B(Q) >> 8
284                   PARM = 0; DELIMITER = 0
285                   %WHILE CODE WORD # 0 %DO:
286                       K = CODE WORD & 3; CODE WORD = CODE WORD >> 2
287                       -> SW(K)
288     !
289     !       ** READ NUMERICAL PARAMETER **
290     %ROUTINE  READ NUMBER(%INTEGERNAME X)
291         I = 0; J = +1
292         %IF NEXT = '-' %OR NEXT = '+' %THEN:
293             J = -1 %IF NEXT = '-'; SET NEXT
294         %FINISH
295         SET NEXT %WHILE NEXT = ' '
296         %IF NEXT = '*' %THEN:
297             I = STAR; SET NEXT; -> L1
298         %FINISH
299         %IF NEXT = '#' %AND LVL # 0 %THENSTART
300             SET NEXT; K = NEXT-'0'; SET NEXT
```

```
301          !  *TEST PARAMETER OFFSET*
302          WPERROR(2) %UNLESS 1 <= K <= ST(PARM BASE)
303          K = ST(PARM BASE + K)
304          ! *WRONG PARAMETER TYPE*
305          WPERROR(3) %IF K & (8_600000) # 8_200000
306          I = B(K & 8_77777)     ;! 'AND' OFF TYPE INFORMATION
307       %ELSE;
308          %WHILE '0' <= NEXT <= '9' %DO;
309             I = (((I<<2)+I)<<1) + (NEXT-'0')    ;! I=10*I+(NEXT-'0')
310             SET NEXT
311          %REPEAT
312          I = STAR-1 %IF I >= STAR
313       %FINISH
314       I = 1 %IF I = 0
315    L1:I = -I %IF J < 0
316       X = I
317    %END
318    !
319    !
320    SW(1):    ! READ ONLY +VE NUMERICAL VALUES
321                WPERROR(11) %IF NEXT = '-'
322    SW(2):    ! READ +VE OR -VE
323                READ NUMBER(K); J = BBOT ;! READ NUMBER CORRUPTS J
324                %WHILE J # W %DO;
325                   -> L2 %IF B(J) = K     ;!  GOT A MATCH YET?
326                   J = J+1
327                %REPEAT
328                B(W) = K; W = W+1          ;! CURRENT 'J' = OLD 'W'
329             L2:J = J+8_200000
330                -> L3
331    !
332    SW(3):          !  ** READ TEXT PARAMETER **
333                    SET NEXT %WHILE NEXT = ' '
334                    %IF NEXT = '#' %AND LVL # 0 %THENSTART
335                       SET NEXT; K = NEXT-'0'; SET NEXT
336                       !  *PARAMETER OFFSET OUT OF RANGE*
337                       WPERROR(2) %UNLESS 1 <= K <= ST(PARM BASE)
338                       K = K + PARM BASE
339                       ! *WRONG PARAMETER TYPE*
340                       WPERROR(3) %IF ST(K)&(8_600000) # 8_400000
341                       J = ST(K)
342                    %ELSE;
343                       %IF NEXT = '-' %AND LAST TEXT # 0 %THEN;
344                          ERROR(4) %IF LVL # 0    ;! *LEGAL ONLY FROM TT*
345                          J = LAST TEXT; SET NEXT; -> L3
346                       %FINISH
347                       REMAP ALPHA
348                       %IF NEXT = 'N' %THEN;   ;! REPLACES QUOTED NEWLINE
349                          J = W+8_400000; B(W) = 1; B(W+1) = NL; W = W+2
350                          SET NEXT; -> L3
351                       %FINISH
352                       !  *TEST FOR ILLEGAL STRING DELIMITER*
353                       ERROR(5) %UNLESS NEXT = STRING DELIMITER
354                       DELIMITER = NEXT
355                       I = W
356                       %UNTIL  NEXT = DELIMITER %DO;
357                          ! *TEST FOR MISSING CLOSING TEXT DELIMITER*
358                          WPERROR(5) %IF STRING IS EMPTY
359                          REMAP ALPHA     ;!  ** UPPERCASE CONVERSION? **
360                          B(I) = NEXT; I = I+1
```

```
361                     SET NEXT
362                 %REPEAT
363                 ! SKIP OVER TRAILING TEXT DELIMITER
364                 SET NEXT %UNLESS CODE WORD & 3 = 3
365                 J = W + 8_400000; LAST TEXT = J
366                 B(W) = I-W-1; W = I
367             %FINISH
368             -> L3
369 !
370 !
371 !
372 L3:             PARM = PARM+1; ST(STP+PARM+4) = J
373             %REPEAT     ;!  WHILE CODE WORD # 0
374             ST(STP+4) = PARM          ;! RECORD NO. OF PARAMETERS
375             SET NEXT %WHILE NEXT = ' '
376             %IF NEXT = '?' %THEN:
377                 MODIFIER = MODIFIER + QUERY; SET NEXT
378             %FINISH
379             EXPAND(Q+1,MODIFIER+R,STP+4,STP+PARM+5)
380         %ELSE:
381             ! 'P' HOLDS NUMERIC CODE FOR PRIMITIVE FN.
382             ! CAN ONLY OCCUR IF LVL > 0 ( SEE  'LOOK UP' )
383             ! PRIMITIVES CAN HAVE UP TO FOUR PARAMETERS BUT NOTE
384             ! ALL CONVERSION MUST HAVE BEEN PREVIOUSLY DONE WHEN
385             ! PASSING PARAMETERS IN GENUINE MACRO CALLS, I.E.
386             ! BY RECURSIVE CALLS ON 'EXPAND'.  FOR EXAMPLE,
387             ! -#1 IS NOT LEGAL WHEN PLANTING A CALL ON
388             ! A PRIMITIVE.
389             PLANT(P)              ;!  STORE FN CODE
390             %WHILE NEXT = '#' %DO:
391                 SET NEXT; K = NEXT-'0'; SET NEXT
392                 WPERROR(2) %UNLESS 1 <= K <= ST(PARM BASE)
393                 PLANT( ST(PARM BASE + K) )
394             %REPEAT
395         %FINISH
396         -> REPEAT
397 !
398 CONTROL(1):         ! **  ')'  **
399       SET NEXT %UNTIL NEXT # ' '
400       READ NUMBER(PARM); SET NEXT %WHILE NEXT = ' '
401       %IF !PARM! = STAR %THEN POPBR(QUERY,PARM) %ELSE:
402           %IF NEXT # '?' %THEN POPBR(0,PARM) %ELSE:
403               POPBR(QUERY,PARM); SET NEXT
404           %FINISH
405       %FINISH
406       -> REPEAT
407 !
408 CONTROL(2):         ! **  '('  **
409       RERUN = 0         ;!  PREVIOUS COMMAND IS NOT NOW REUSABLE
410       PUSHBR(0); SET NEXT
411       -> REPEAT
412 !
413 CONTROL(3):         ! ** ','  **
414       SET NEXT      ;!  SKIP OVER ','
415       PLANT(RBRCODE)
416       K = ST(STP+3)    ;!  START OF CURRENT BRACKETTING LEVEL
417       I = ((X&8_1777)<<5) + (BRACKETS&8_37)
418       %WHILE ESCAPE CELL < K %DO:
419           J = ESCAPE CELL; ESCAPE CELL = B(J)
420           B(J) = I
```

```
421         %REPEAT
422         K = ESCAPE CELL; ESCAPE CELL = X; PLANT(K)
423         -> REPEAT
424     !
425     CONTROL(4):         ! ** '\'  **
426         SET NEXT %UNTIL NEXT # ' '
427         %IF NEXT # '(' %THEN MODIFIER = INVERT %ELSE:
428            PUSHBR(INVERT); SET NEXT
429         %FINISH
430         -> REPEAT
431     !
432     CONTROL(5):         ! ** '!' **
433         SET NEXT %UNTIL NEXT # ' '
434         READ NUMBER(PARM)
435         K = 0
436         K = -BRACKETS %AND SET NEXT %IF NEXT = '?'
437         !  'STAR' IS A SPECIAL NUMBER WHICH BY ITS VALUE IMPLIES
438         !  '*' OR '?'.  'READ NUMBER' CAN ONLY SUPPLY THIS VALUE
439         !  IN THIS CONTEXT SO NO AMBIGUITY ARISES.
440         K = -BRACKETS %IF PARM = STAR
441         ERROR(14) %UNLESS STRING IS EMPTY %AND RERUN # 0
442         B(Y-3) = PARM
443         B(Y-4) = K
444         BRACKETS = 0; LVL = -1
445         %RETURN
446     !
447     CONTROL(6):     !       ** DEFINE NEW MACRO **
448         ! *DEFINITIONS CANNOT BE CONCATENATED:  CHECK THIS*
449         ERROR(6) %UNLESS W = BBOT %AND X = Y-5
450         ! Y-5 ABOVE IS Y-1 REALLY BUT ALLOWS FOR LVL ZERO BRACKET
451         SET NEXT %UNTIL NEXT # ' '
452         READ COMMAND ATOM ID
453         K = B(W)!!PROTECT
454         ! *MACRO NAME LONGER THAN NAME MASK-2*
455         ERROR(10) %IF K & NAME MASK # K
456         W = W+K
457         SET NEXT %WHILE NEXT = ' '
458         CODE WORD = 0
459         %IF NEXT = '(' %THEN:   ;! ...GENERATE PARAMETER CODE-WORD
460            SET NEXT     ;!     SKIP OVER '('
461            %OWNINTEGERARRAY  PARMTYPE(1:3)= '+', '-', 'T'
462            %UNTIL NEXT = ')' %DO:
463               ! *MAX. OF 4 PARAMS./MACRO: TEST FOR MORE*
464               ERROR(7) %IF CODE WORD & 8_003400 # 0
465               REMAP ALPHA     ;!  PUT LETTERS IN STANDARD FORM
466               %FOR K = 1,1,3 %DO:
467                  %IF PARMTYPE(K) = NEXT %THEN:
468                     CODE WORD = (CODE WORD << 2) + K; -> L4
469                  %FINISH
470               %REPEAT
471               ERROR(7)          ;!  *ILLEGAL TYPE SPECIFIER*
472            L4:SET NEXT
473            %REPEAT
474            K = 0
475            %WHILE CODE WORD # 0 %DO:
476               K = (K<<2) + (CODE WORD & 3)
477               CODE WORD = CODE WORD >> 2
478            %REPEAT
479            CODE WORD = K << 8
480            SET NEXT %UNTIL NEXT # ' '
```

```
481         %FINISH
482         %IF NEXT # '=' %THEN ERROR(8) %ELSE:  ;! *'=' MISSING*
483            SET NEXT %UNTIL NEXT # ' '
484         %FINISH
485         I = W
486         %WHILE NEXT # NL %DO:
487            REMAP ALPHA; I = I+1; B(I) = NEXT; SET NEXT
488         %REPEAT
489         ! TEST FOR AND THROW AWAY THE CLOSING ']'
490         ERROR(8) %UNLESS B(I) = ']'          ;!      *']' MISSING*
491         ! ** PACKING ROUTINE CAN BE INSERTED HERE IF REQUIRED **
492         !  SETTING 'I' APPROPRIATELY WILL AUTOMATICALLY SET ALL
493         !  DERIVATIVE VALUES CORRECTLY.
494         K = I-W; B(W) = K + CODE WORD
495         W = BBOT
496         ERROR(10) %IF K & BODY MASK # K ;! *MACRO BODY TOO LONG*
497         LOOK UP    ;!   SEARCH EXISTING MACRO TABLE FOR NEW ID  ...
498         W = I      ;!   ... THEN RESET 'W'
499   !  MACROS ARE OF TWO TYPES, PROTECTED AND UNPROTECTED.  THE
500   !  FORMER ARE LOADED BEFORE CONTROL IS PASSED TO THE USER AND
501   !  DEFINE THE BASIC COMMAND SET FOR THAT RUN OF THE EDITOR;
502   !  THEY CANNOT BE DYNAMICALLY ALTERED.  UNPROTECTED MACROS ARE
503   !  CREATED BY THE USER TO MEET A PARTICULAR TRANSIENT NEED AND
504   !  CAN BE ALTERED AS REQUIRED.
505   !     IN ORDER THAT ANY NAME AVAILABLE TO THE USER HAVE A UNIQUE
506   !  MEANING, UNPROTECTED MACROS ARE DESTRUCTIVELY RE-DEFINED
507   !  RATHER THAN BEING PUSHED DOWN.  SINCE PROTECTED MACROS CANNOT
508   !  BE RE-DEFINED DYNAMICALLY, THIS PROBLEM DOES NOT ARISE WITH
509   !  THEM: TO AVOID PROBLEMS WHICH WOULD ARISE IF UNOFFICIAL
510   !  INTERMEDIATE NAMES, USED IN IMPLEMENTING THE BASIC EDITOR
511   !  COMMANDS WERE SUBSEQUENTLY ACCESSED BY ITS USERS, WITHIN THE
512   !  PROTECTED MACROS ONLY, THE SAME IDENTIFIER CAN BE USED
513   !  TO REFER TO DIFFERENT MACRO BODIES.  SINCE THE LOOK-UP ONLY
514   !  SEARCHES THAT PART OF THE TABLE DEFINED BEFORE THAT MACRO
515   !  WHOSE BODY IS CURRENTLY BEING EXPANDED, AND EXITS AS SOON AS
516   !  A MATCH IS OBTAINED, THE MEANING IS IN FACT UNAMBIGUOUS.  THE
517   !  USER ONLY HAS ACCESS TO THE MOST RECENTLY DEFINED MACRO WITH
518   !  A PARTICULAR NAME.
519   !
520         %IF P > 0 %AND PROTECT = 0 %THEN:
521            ! IF NAME PRESENT(P>0) AND INITIAL LOAD DONE(PROTECT=0)
522            ! *COMPLAIN IF PRIMITIVE OR PROTECTED*
523            W = BBOT %AND ERROR(9) %IF P=0 %OR B(P) # B(BBOT)
524            K = (R-P) - (W-BBOT)    ;! SIZE DIFFERENCE( +VE/-VE )
525            MOVE(B,Y,P-1,K); Y = Y+K
526            MOVE(B,BBOT,W-1,R-W)
527         %ELSE:
528            ! ADD NEW IDENTIFIER
529            Y = Y+(BBOT-W)
530            MOVE(B,BBOT,W-1,Y-BBOT)
531         %FINISH
532         BRACKETS = 0    ;!  TO PREVENT SUBSEQUENT FAILURE
533         %RETURN    ;!  N.B.  LEAVES  LVL=0, NOT -1, ON RETURN TO
534                    ;!  CALLER.  THIS IS USED TO DISTINGUISH MACRO
535                    ;!  DEFINING COMMANDS FROM OTHERS.
536   !
537   REPEAT: %REPEAT
538      POPBR(0,1) %IF MACRO DEFNS & 8_700000 # 0
539      LVL = LVL-1
540   %END     ;!  OF %ROUTINE EXPAND
```

```
%END     ;!  OF LEVEL 1 %BEGIN BLOCK
!
!
!
! ** START OF EXECUTING ALGORITHM FOR PROGRAM ASSEMBLED ABOVE **
!
%INTEGER  COND MASK,OK EXIT,ERROR WORD,LOOP START,LOOP COUNT
%INTEGER  PC,STP,CURRENT FN, COND
%INTEGER  I,J,K
%OWNINTEGER  OK = 1, FAIL = 0
%SWITCH  FN(1:16)
!
!
   PC = Y-1; STP = 0; -> START
!
TEST:  -> ERROR EXIT %IF COND # COND MASK
XCT NEXT: -> READ NEW COMMAND %IF STP = 0 ;! DONE ON EMPTY STACK
START:  CURRENT FN = B(PC); PC = PC-1
   -> FN(CURRENT FN)
!
!
%ROUTINE  SAVE AND RESET        ;!  ** PUSH RUN-TIME STACK **
   ST(STP+1) = COND MASK
   ST(STP+2) = LOOP START
   ST(STP+3) = LOOP COUNT
   ST(STP+4) = ERROR WORD
   COND MASK = I
   OK EXIT = B(PC); PC = PC-1
   LOOP COUNT = B(PC); PC = PC-1
   LOOP START = PC; ERROR WORD = PC; PC = PC-1
   STP = STP+5; ST(STP) = OK EXIT
%END
!
%ROUTINE RESTORE OLD CONTEXT    ;!  ** POP RUN-TIME STACK **
   PC = ST(STP)
   STP = STP-5
   ! *TEST FOR EXECUTION-TIME STACKING ERROR*
   ERROR(-2) %IF STP < 0
   OK EXIT = ST(STP)
   COND MASK = ST(STP+1)
   LOOP START = ST(STP+2)
   LOOP COUNT = ST(STP+3)
   ERROR WORD = ST(STP+4)
%END
!
!
FN(1):  I = OK; -> L1    ;!  '('
FN(2):  I = FAIL         ;!  '\('
L1: SAVE AND RESET
   -> XCT NEXT
!
!
!
!
FN(3):  ;!      ')'
         LOOP COUNT = LOOP COUNT - 1
         %IF LOOP COUNT <= 0 %THEN RESTORE OLD CONTEXT %ELSE:
            ERROR WORD = LOOP START; PC = LOOP START - 1
        %FINISH
         -> XCT NEXT
```

```
601   !
602   FN(4):  ERROR(-99)              ;! SPARE CONTROL FUNCTION WHICH IN
603                                   ;! FACT WAS NOT ULTIMATELY REQUIRED.
604   !
605   ERROR EXIT:    ;!  SOFTWARE INTERRUPTS COME HERE WHEN A FN. FAILS
606      ERROR WORD = B(ERROR WORD)
607      %IF ERROR WORD > 0 %THEN:          ;! A COMMA: RESET PC AND STP
608         !  STP = 5*(ERROR WORD & 8_37)
609         STP = ERROR WORD & 8_37; STP = (STP<<2) + STP
610         ERROR WORD = ERROR WORD >> 5
611         PC = ERROR WORD - 1             ;! PC ADVANCES BACKWARD
612         COND = OK
613      %ELSE:
614         %IF ERROR WORD = 0 %THEN:       ;!  ...AN UNFORESEEN FAILURE
615            %PRINTTEXT '*FAILED'; NEWLINE; -> READ NEW COMMAND
616         %FINISH
617         !  STP = 5*!ERROR WORD!         ;! ...A FORESEEN FAILURE
618         STP = -ERROR WORD; STP = (STP<<2) + STP
619         RESTORE OLD CONTEXT
620      %FINISH
621      -> XCT NEXT
622   !
623   !
624   !
625   !         ** EDITING ROUTINES FOLLOW **
626   !
627   %OWNINTEGER  MAX LINE SIZE = 100
628   %INTEGER  S,T,U,V,CP,LP,LB END,LB LINES,ALTERED,CHANGED
629   %INTEGER  LINE ORIGIN, LINE OFFSET
630   %INTEGER  P1,P2
631   %INTEGER  IN, OUT
632   !
633   %ROUTINE INITIALISE
634   ! THIS ROUTINE IS CALLED AT THE BEGINNING BUT PUT HERE SO'S IT
635   ! CAN REFERENCE VARIABLES NOT DECLARED AT THE POINT OF CALL.
636      S = AMIN; T = S; AA(T) = 0
637      U = AMAX; V = U; AA(V) = 0
638      LINE ORIGIN = 0; LINE OFFSET = 1; LB LINES = 0
639      ALTERED = 0; CHANGED = 0
640      LP = 0
641      IN = 1; OUT = 3
642      SELECT INPUT(MACROS); SELECT OUTPUT(REPORT)
643   %END
644   !
645   %PREDICATE  GAP IS TOO NARROW
646      ! ROUTINES  'UNPACK'  AND  'READ LINE'  BELOW BOTH ASSUME
647      ! THAT SUFFICIENT FREE BUFFER SPACE IS AVAILABLE AND DO NOT
648      ! THEMSELVES CHECK.  USE 'GAP IS TOO NARROW' TO DO THIS
649      ! TEST WHEREVER APPROPRIATE
650      %FALSE %IF U-T > MAX LINE SIZE + 10
651      %TRUE
652   %END
653   !
654   !  'PACK' AND 'UNPACK' CAN BE CHANGED TO GIVE A HIGHER PACKING
655   !  DENSITY IN THE LINE RECORDS STORED BY THE MAIN BUFFER WITHOUT
656   !  THE REST OF THE EDITOR BEING AFFECTED AT ALL:  BUT NOTE THAT
657   !  IF ONE IS CHANGED, THE OTHER MUST BE TOO.
658   %ROUTINE  PACK
659   %INTEGER  I,K
660      I = T
```

```
661         %IF ALTERED # 0 %THEN:
662             CHANGED = 1
663             !  TWO MARKERS, 'ALTERED' AND 'CHANGED' ARE USED.  THESE
664             ! BEING NON-ZERO DENOTE RESPECTIVELY THAT (1) THE CURRENT
665             ! LINE HAS BEEN MODIFIED AND MUST BE RE-PACKED, NOT
666             ! MERELY COPIED, AND (2) THAT THE TEMPORARY FILE CURRENTLY
667             ! BEING WRITTEN HAS BEEN MODIFIED RELATIVE TO THE PREVIOUS
668             ! ONE AND MUST THEREFORE BE COPIED IN ITS ENTIRETY WHEN
669             ! GOING BACKWARDS, NOT MERELY REWOUND AND RE-READ.
670             K = AA(LP)
671             %WHILE K # NL %DO:
672                 I = I+1 %AND AA(I) = K %IF K # 0  ;! SKIP NULLS
673                 LP = LP+1; K = AA(LP)
674             %REPEAT
675             ! ** PACKING ROUTINE COULD BE INSERTED HERE OR IN THE LOOP
676             !    ABOVE:  'I' POINTS TO THE LAST CELL OCCUPIED **
677             %IF LP = LB END %THEN LP = 0 %AND ALTERED = 0 %ELSE:
678                 K = I-LP                    ;!  'T' IS RESET TO 'I' LATER
679                 %IF K # 0 %THEN:
680                     MOVE(AA,LP,LB END,K)
681                     LP = LP+K; LB END = LB END + K
682                 %FINISH
683                 LP = LP+1  ;! BUMP PAST LINK CELL TO START OF NEXT LINE
684             %FINISH
685         %ELSE:
686             %FOR K = U+1,1,U+AA(U)-1 %DO:
687                 I = I+1; AA(I) = AA(K)
688             %REPEAT
689             U = U+AA(U); LP = 0          ;!  'ALTERED'  ALREADY IS ZERO
690         %FINISH
691         I = I+1
692         AA(I) = I-T; T = I     ;!  NOTE RELOCATABLE POINTERS
693         LB LINES = LB LINES + 1; LINE OFFSET = LINE OFFSET + 1
694     %END
695     !
696     %ROUTINE  MARK LB
697         %IF ALTERED = 0 %THEN:
698             ERROR(-3) %IF LB LINES < 0       ;! *-VE NO. OF LINES?*
699             U = U+AA(U)
700             ALTERED = 1
701         %FINISH
702     %END
703     !
704     %ROUTINE  UNPACK
705     %INTEGER  I,K,X
706         X = LP
707         %IF X = 0 %THEN:
708             LP = T+1; LB END = LP; CP = LP
709         %FINISH
710         ! MUST UNPACK NULL LINES('NL' ONLY) SO %FOR WON'T DO
711         I = U+1; K = U+AA(U)
712         %WHILE I # K %DO:
713             AA(LB END) = AA(I)
714             LB END = LB END + 1; I = I+1
715         %REPEAT
716         !  ** UNPACKING ROUTINE CAN BE INSERTED HERE OR IN THE
717         !     PRECEDING LOOP **
718         AA(LB END) = NL
719         %IF X=0 %THEN ALTERED = 0 %ELSE U = U+AA(U) %AND ALTERED = 1
720         LB LINES = LB LINES - 1
```

```
%END
!
%ROUTINE  READ LINE(%INTEGER  STREAM)
%INTEGER  S,K
   PACK %WHILE LP # 0
   COND = OK
   SELECT INPUT(INSTREAM(STREAM)) %IF STREAM # COMMAND
   %FAULT9 -> L1
   LP = T+1; LB END = T; K = LB END + MAX LINE SIZE - 1
   %UNTIL S = NL %DO:
      READ SYMBOL(S)
      S = S-32 %IF 'A' <= S-32 <= 'Z' ;! ** UPPERCASE LETTERS **
      LB END = LB END + 1
      AA(LB END) = S
      -> L1 %IF LB END = K
   %REPEAT
   -> L2
!
L1:COND = FAIL %AND -> L3 %IF LB END < LP
   LB END = LB END + 1; AA(LB END) = NL
L2:ALTERED = 1
L3:SELECT INPUT(COMMAND)
%END
!
!
%ROUTINE  DUMP LINE(%INTEGER  STREAM)
%INTEGER  I
   SELECT OUTPUT(OUTSTREAM(STREAM)) %IF STREAM # REPORT
   %FOR I = LP,1,LB END %DO:
      PRINT SYMBOL(8_136) %IF STREAM = REPORT %AND I = CP # LP
      PRINT SYMBOL( AA(I) ) %IF AA(I) # 0
   %REPEAT
   SELECT OUTPUT(REPORT)
%END
!
%ROUTINE  NORMALISE
%INTEGER  I,K,X
   ! SPLITS BUFFER UP SO THAT 'CURRENT LINE' IS LINE WHOSE
   ! LENGTH CELL IS POINTED TO BY 'LP'.
   !
   %IF S <= LP <= T %THEN:
      %IF AMAX-V # 0 %THEN:
         MOVE(AA,U,V,AMAX-V)
         U = U+AMAX-V; V = AMAX  ;!  I.E. V = V+AMAX-V
      %ELSE:
         %IF V = U %THEN:
            U = AMAX; V = U; AA(V) = 0
         %FINISH
      %FINISH
      !   SPLIT BLOCK S:T
      ! REVERSE LENGTH CELL POINTERS FIRST
      K = AA(T); X = T
      %WHILE T # S %AND T >= LP %DO:
         T = T-K; I = AA(T)
         AA(T) = K; K = I
      %REPEAT
      MOVE(AA,T,X-1,U-X) %IF U-X # 0
      U = U+T-X; AA(T) = K ;! RESTORE ORIGINAL AA(T)
      %IF S # T %THEN:
         MOVE(AA,S,T,AMIN-S) %IF AMIN-S # 0
```

```
781             T = T+AMIN-S; S = AMIN    ;!  I.E. S = S+AMIN-S
782          %FINISH
783       %ELSE:
784          %IF U <= LP <= V %THEN:
785             %IF AMIN-S # 0 %THEN:
786                MOVE(AA,S,T,AMIN-S)
787                T = T+AMIN-S; S = AMIN
788             %ELSE:
789                %IF S = T %THEN:
790                   S = AMIN; T = S; AA(T) = 0
791                %FINISH
792             %FINISH
793             %IF LP # U %THEN:    ;!  SPLIT BLOCK U:V
794                ! REVERSE POINTERS FIRST
795                K = AA(U); X = U
796                %UNTIL U = LP %DO:
797                   U = U+K; I = AA(U)
798                   AA(U) = K; K = I
799                %REPEAT
800                MOVE(AA,X+1,U,T-X) %IF T-X # 0
801                T = T+U-X; AA(U) = K    ;! RESTORE ORIGINAL AA(U)
802             %FINISH
803             %IF U # V %THEN:
804                MOVE(AA,U,V,AMAX-V) %IF AMAX-V # 0
805                U = U+AMAX-V; V = AMAX
806             %FINISH
807          %FINISH
808       %FINISH
809       LP = 0
810    %END
811    !
812    ! THE FOLLOWING ROUTINE CENTRES THE WINDOW ON THE TARGET LINE
813    ! WHOSE NUMBER IS PASSED AS A PARAMETER:  1 <= TARGET LINE <= N
814    ! WHERE THERE ARE N LINES IN THE FILE.
815    ! THIS ROUTINE RETURNS THE BUFFER TO THE CALLING PROGRAM IN GOOD
816    ! ORDER WITH FREE SPACE AVAILABLE FOR INSERTIONS AND SO ON:
817    ! MOVE TO(CURRENT LINE) IS THEREFORE USED TO CHECK THAT SPACE IS
818    ! AVAILABLE AND OBTAIN IT IF NOT.
819    ! THE OPERATION OF THIS ROUTINE AFFECTS THE EFFICIENCY OF THE
820    ! WHOLE PROGRAM QUITE CONSIDERABLY AND THE STRATEGIES EMPLOYED
821    ! HERE WILL DEPEND CONSIDERABLY ON THE OPERATING ENVIRONMENT.
822    ! ANY SERIOUS IMPLEMENTATION OF THIS PROGRAM WILL THEREFORE
823    ! ALMOST INEVITABLY REQUIRE THE REWRITING OF 'MOVE TO'.
824    %ROUTINE  MOVE TO(%INTEGER  TARGET LINE)
825    %INTEGER  I,K
826    %OWNINTEGER  AV LINES , STEP
827    %OWNINTEGER  CLOSE=16    ;!  CLOSE = NUMERIC CODE FOR PRIM('Z')
828    !
829       %ROUTINE  FILL UP(%INTEGER  LIMIT)
830          LP = V; NORMALISE
831          %WHILE AMAX-T > LIMIT %DO:
832             READ LINE(IN); %RETURN %IF COND = FAIL
833             PACK
834          %REPEAT
835       %END
836       !
837       %ROUTINE  EMPTY OUT(%INTEGER  N)
838          LP = S; NORMALISE
839          COND = FAIL
840          %UNTIL N <= 0 %DO:
```

```
841             LP = 0; %RETURN %IF U = V
842             UNPACK; DUMP LINE(OUT); U = U+AA(U)
843             LINE OFFSET=LINE OFFSET-1; LINE ORIGIN=LINE ORIGIN+1
844             N = N-1
845         %REPEAT
846         COND = OK
847     %END
848     !
849     TARGET LINE = 1 %IF TARGET LINE <= 0
850     PACK %WHILE LP # 0
851 !
852 L1:EMPTY OUT(LB LINES >> 1) %IF U-T < MAX LINE SIZE + 5
853 L2:%IF LINE ORIGIN < TARGET LINE < LINE ORIGIN + LB LINES %THEN:
854         K = TARGET LINE - (LINE ORIGIN + LINE OFFSET)
855         LINE OFFSET = LINE OFFSET + K
856         %IF K < 0 %THEN:
857             LP = T; I = -1
858             LP = LP-AA(LP) %AND I = I-1 %WHILE I # K
859         %ELSE:
860             LP = U; I = 0
861             LP = LP+AA(LP) %AND I = I+1 %WHILE I # K
862         %FINISH
863         -> L3 %IF LP = V
864         %IF K<0 %THEN STEP=0 %ELSE STEP=((AV LINES>>1)+STEP)>>1
865         NORMALISE; %RETURN
866     %FINISH
867 L3:%IF TARGET LINE > LINE ORIGIN %THEN:
868         EMPTY OUT( (TARGET LINE-LINE ORIGIN) %C
869                     -(((AVLINES<<1)+AVLINES)>>2)+STEP )
870         FILL UP( (MAX LINE SIZE + 5) << 1 )
871         %IF COND=OK %THEN AV LINES=(AV LINES + LB LINES)>>1 %ELSE:
872             %IF TARGET LINE  > LINE ORIGIN + LB LINES %THEN:
873                 LP = 0; %RETURN %IF B(ERROR WORD) # 0
874                 %PRINTTEXT '**END**'; NEWLINE; %MONITOR 31
875             %FINISH
876         %FINISH
877     %ELSE:
878         %IF CHANGED # 0 %THEN:  ;! DON'T COPY IF UNNECESSARY
879             %UNTIL COND = FAIL %DO:
880                 EMPTY OUT(LB LINES << 1)
881                 FILL UP( MAX LINE SIZE << 1 )
882             %REPEAT
883             EMPTY OUT(LB LINES << 1)
884         %FINISH
885         SELECT INPUT(INSTREAM(IN)); CLOSE INPUT
886         SELECT OUTPUT(OUTSTREAM(OUT)); CLOSE OUTPUT
887         SELECT INPUT(COMMAND); SELECT OUTPUT(REPORT)
888         PRINT SYMBOL('*'); K = OUT-2
889         %IF K < 0 %THEN %PRINTTEXT 'FILE' %ELSE:
890             PRINT SYMBOL('.'); PRINT SYMBOL('0'+K)
891         %FINISH
892         %PRINTTEXT ' COMPLETE'; NEWLINE
893         LINE ORIGIN = 0; LINE OFFSET = 1; LB LINES = 0
894         %IF CHANGED # 0 %THEN:
895             IN = 4 %IF IN = 1
896             K = OUT; OUT = IN; IN = K
897             OUT = 1 %IF CURRENT FN = CLOSE
898             FILL UP( (AMAX-AMIN) >> 1 )
899         %FINISH
900         CHANGED = 0
```

```
901         %FINISH
902         -> L2
903     %END
904     !
905     %ROUTINE  COMPARE STRINGS(%INTEGER  PNTR)
906     %OWNINTEGER  R,S
907     %INTEGER  I
908         ! INITIALISE IF PNTR >= 0
909         ! SEARCH START IS SPECIFIED IMPLICITLY BY P1
910         ! P2 SET ON SUCCESSFUL EXIT TO SYMBOL AFTER MATCHED STRING
911         !
912         %IF PNTR >= 0 %THEN:
913             MOVE TO(LINE ORIGIN + LINE OFFSET) %IF LP = 0
914             R = PNTR; S = PNTR+B(PNTR)
915             P1 = P1+1 %WHILE AA(P1) = 0
916             COND = FAIL
917         %FINISH
918         I = R; P2 = P1
919         %WHILE I # S %DO:
920             I = I+1
921             P2 = P2+1 %WHILE AA(P2) = 0
922             !  ** WILL REQUIRE CASE CONVERSION OF LETTERS HERE IF
923             !       NON-CASE-DEPENDENT STRING MATCHING IS DESIRED **
924             %RETURN %IF AA(P2) # B(I)
925             P2 = P2+1
926         %REPEAT
927         COND = OK
928     %END
929     !
930     %ROUTINE  SET P1 P2(%INTEGER  PNTR)
931         ! SETS P1,P2 TO TARGET STRING IF IT APPEARS TO RIGHT OF CP
932         ! IN CURRENT LINE
933         ! 'COND' IS SET BY 'STRINGS MATCH'
934         !
935         !
936         !
937         P1 = CP
938         COMPARE STRINGS(PNTR); %RETURN %IF COND = OK %OR LP = 0
939         %WHILE AA(P1) # NL %DO:
940             P1 = P1+1 %UNTIL AA(P1) # 0
941             COMPARE STRINGS(-1); %RETURN %IF COND = OK
942         %REPEAT
943     %END
944     !
945     %INTEGERFN  PARM
946     %INTEGER  P,Q
947         P = B(PC); Q = P & 8_77777; PC = PC-1
948         %RESULT = Q %IF P & 8_600000 = 8 400000   ;! A TEXT STRING
949         %RESULT = B(Q)                ;!  OTHERWISE, MUST BE A NUMBER
950     %END
951     !
952     %ROUTINE  STEP(%INTEGER  N) ;! GO N POSITIONS LEFT(-), RIGHT(+)
953     %INTEGER  I,INCR,LIMIT
954         MOVE TO(LINE ORIGIN + LINE OFFSET) %IF LP = 0
955         %IF N # 0 %THEN:
956             N = -N; INCR = 1; LIMIT = LB END
957             INCR = -INCR %AND LIMIT = LP %IF N > 0
958             %IF !N! = STAR %THEN I = LIMIT %ELSE:
959                 I = CP; COND = FAIL
960                 %WHILE N # 0 %DO:
```

```
961                 %RETURN %IF I = LIMIT    ;!  ... WITH COND = FAIL ...
962                 I = I+INCR %UNTIL AA(I) # 0;
963                 N = N+INCR
964              %REPEAT
965           %FINISH
966        %FINISH
967        CP = I; COND = OK
968     %END
969     !
970     %ROUTINE  ERASE(%INTEGER  FROM,TO)
971     ! DELETE AA(FROM:TO-1) IF FROM < TO
972        AA(FROM) = 0 %AND FROM = FROM+1 %WHILE FROM < TO
973        MARK LB
974     %END
975     !
976     !
977     !  'INSERT' FAILS IF NO SPACE IS AVAILABLE FOR THE INSERTION
978     %ROUTINE  INSERT(%INTEGER  PARM)
979     %INTEGER  I,K,N
980        N = B(PARM); %RETURN %IF N = 0
981        %IF LP = 0 %THEN:  ;! CREATE NULL LINE IF BUFFER IS EMPTY
982           LP = T+1; LB END = LP; CP = LB END; AA(CP) = NL
983        %FINISH
984        MARK LB
985        K = CP
986        K = K-1 %WHILE K # LP %AND AA(K-1) = 0
987        %IF CP-K < N %THEN:
988           CP = CP+1 %WHILE AA(CP) = 0
989           %IF CP-K < N %THEN:
990              I = N-(CP-K)
991              COND = FAIL %AND %RETURN %IF LBEND+I >= U
992              MOVE(AA,CP,LB END,I); LB END = LB END + I
993           %FINISH
994        %FINISH
995        CP = K+N; PARM = PARM+1
996        AA(K+I) = B(PARM+I) %FOR I = 0,1,N-1
997        PACK %IF B(PARM) = NL %AND B(PARM-1) = 1
998        COND = OK
999     %END
1000    !
1001    !
1002    !  ** DEFNS OF EDITING OPERATIONS FOLLOW **
1003    !
1004    FN(5):    ;!   'VERIFY'     (V)
1005       P1 = CP
1006       COMPARE STRINGS(PARM)
1007       -> TEST
1008    !
1009    FN(6):    ;!   'AFTER'      (A)
1010       SET P1 P2(PARM)     ;!  SETS 'COND'
1011       CP = P2 %IF COND = OK
1012       -> TEST
1013    !
1014    FN(7):    ;!   'BEFORE'     (B)
1015       SET P1 P2(PARM)
1016       CP = P1 %IF COND = OK
1017       -> TEST
1018    !
1019    FN(8):    ;!   'DELETE'     (D)
1020       SET P1 P2(PARM)
```

```
1021        %IF COND = OK %THEN:
1022            MARK LB; CP = P1
1023            %IF CP = LB END %THEN:
1024                MOVE TO(LINE ORIGIN + LINE OFFSET) %IF U = V
1025                UNPACK
1026            %ELSE  ERASE(P1,P2)
1027        %FINISH
1028        -> TEST
1029    !
1030    FN(9):    ;!    'INSERT'     (I)
1031        INSERT(PARM)     ;!   SETS 'COND'
1032        -> TEST
1033    !
1034    FN(10):   ;!    'STEP'       (S)
1035        STEP(PARM); -> TEST
1036    !
1037    FN(11):   ;!    'ERASE'      (E)
1038        J = PARM; K = CP
1039        STEP(J); ERASE(K,CP) %IF COND = OK
1040        -> TEST
1041    !
1042    FN(12):   ;!    'MOVE'       (M)
1043        MOVE TO(LINE ORIGIN + LINE OFFSET + PARM)
1044        UNPACK %IF U # V
1045        COND = OK
1046        -> TEST
1047    !
1048    FN(13):   ;!    'PRINT'      (P)
1049        K = PARM-1
1050        %IF LP = 0 %THEN:
1051            MOVE TO(LINE ORIGIN + LINE OFFSET) %IF U = V
1052            UNPACK
1053        %FINISH
1054        CP = LP %IF K # 0
1055        DUMP LINE(0)
1056        %WHILE K > 0 %DO:
1057            PACK; MOVE TO(LINE ORIGIN + LINE OFFSET) %IF U = V
1058            UNPACK; DUMP LINE(0)
1059            K = K-1
1060        %REPEAT
1061        COND = OK
1062        -> TEST
1063    !
1064    FN(14):   ;!    'GET'        (G)
1065        J = PARM; J = 0 %IF J # 2
1066        %FOR I = 1,1,PARM %DO:     ;! N.B.  'PARM' IS ONLY CALLED ONCE
1067            MOVE TO(LINE ORIGIN + LINE OFFSET) %IF GAP IS TOO NARROW
1068            PROMPT(':'); READ LINE(J)
1069            %IF COND=FAIL %OR ( AA(LP)=':' %AND LBEND=LP+1 ) %THEN:
1070                COND = FAIL; LP = 0; -> TEST
1071            %FINISH
1072            PACK
1073        %REPEAT
1074        COND = OK
1075        -> TEST
1076    !
1077    FN(15):   ;!    'UP TO'      (U)
1078        SET P1 P2(PARM)
1079        MARK LB %AND ERASE(CP,P1) %IF COND = OK
1080        -> TEST
```

```
1081    !
1082    FN(16):    ;!    $CLOSE(1)/$REWIND(3)        (Z)
1083        %IF PARM = 1 %THEN:
1084            MOVE TO(O)        ;!  COPY UP REST OF CURRENT PASS
1085            CHANGED = 1; LINE ORIGIN = 1; MOVE TO(O)  ;! FINAL OUTPUT
1086        %ELSE:
1087            SELECT INPUT( INSTREAM(2) ); CLOSE INPUT
1088            SELECT INPUT(COMMAND)
1089            -> TEST
1090        %FINISH
1091    !
1092    %ENDOFPROGRAM
```

```
%BEGIN     ;!   LEVEL 0
!   ** DEFINE SYMBOLIC I/O STREAMS **
%OWNINTEGER  COMMAND=0, REPORT=0, MACROS=5
%OWNINTEGERARRAY  INSTREAM(1:4)= 1,2,3,4
%OWNINTEGERARRAY  OUTSTREAM(1:4)= 1,2,3,4
!
%OWNINTEGER  BBOT=0, W=0, X=0, Y=512, Z=512, STAR=10001
%INTEGERARRAY  B(BBOT:Z-1)       ;! COMMAND BUFFER + MACRO DEFNS.
%OWNINTEGER  STKTOP=100
%INTEGERARRAY  ST(0:STKTOP)      ;!  DPDA STACK
! DEFINE FUNCTION CODES FOR NEST=( AND UN-NEST=) OPERATIONS
%OWNINTEGER  LBRCODE=1, RBRCODE=3   ;!  NOTLBR=2
!
%OWNINTEGER  AMIN=1, AMAX = 1000
   AMAX=FREESTORE-800
%INTEGERARRAY  AA(AMIN:AMAX)     ;!  EDITOR TEXT BUFFER
!
!
   %FAULT 30 -> TEST             ;!  TRAP EXECUTION TIME ERROR
   %FAULT 31 -> CLEAR TTY        ;!  TRAP DECODE TIME ERROR
!
   %ROUTINESPEC INITIALISE
   INITIALISE; %PRINTTEXT 'EDIT5B   10/4/72'; NEWLINE
   -> READ NEW COMMAND
!
!
!
!
!
!
!
!
!
!
%ROUTINE  ERROR(%INTEGER N)    ;!  DIAGNOSE ERRORS
%SWITCH  S(0:14)
   %ROUTINE  PRINT STRING
      PRINT SYMBOL( B(W+N) ) %FOR N = 1,1,(B(W)&8_177)-1
      SPACE
   %END
!
   SELECT OUTPUT(REPORT)
   -> S(N) %IF N >= 0
   %PRINTTEXT '*ERROR '; WRITE(N,1); -> END
!
S(0): PRINT STRING; -> QUERY
S(1): PRINT STRING; %PRINTTEXT 'NOT ID.'; -> END
S(2): PRINT STRING; %PRINTTEXT '#N, 1<=N<=4'; -> QUERY
S(3): PRINT STRING; -> S7
S(4): PRINT STRING; %PRINTTEXT ''')-'' IN MACRO'; -> QUERY
S(5): PRINT STRING; %PRINTTEXT '/TEXT/'; -> QUERY
S(6): %PRINTTEXT '('; -> QUERY
S(7):S7: %PRINTTEXT 'PARAM. TYPE'; -> QUERY
S(8): %PRINTTEXT ''')='' OR '')'' MISSING'; -> QUERY
S(9): PRINT STRING; %PRINTTEXT 'IS PROTECTED'; -> END
S(10): PRINT STRING; -> S13
S(11): PRINT STRING; %PRINTTEXT '-'; -> QUERY
S(12): %PRINTTEXT 'BRACKETS'; -> QUERY
S(13):S13: %PRINTTEXT 'TOO LONG'; -> END
S(14): %PRINTTEXT '!'
```

```
61    !
62    QUERY: %PRINTTEXT ' ?'
63    END:   NEWLINE
64       %MONITOR 31
65    %END
66    !
67    %ROUTINE  MOVE(%INTEGERARRAYNAME  AA,%INTEGER  P,Q,N)
68    !  MOVE AA(P:Q) TO AA(P+N:Q+N)
69       %IF N > 0 %THEN:
70          AA(P+Q+N) = AA(P+Q) %FOR  Q = Q-P,-1,0
71       %ELSE:
72          AA(P+Q+N) = AA(P+Q) %FOR  Q = 0,1,Q-P
73       %FINISH
74    %END
75    !
76    !
77    !
78    !
79    !
80    !
81    !  ** THE COMPLETE COMMAND DECODER IS CONTAINED IN THE FOLLOWING
82    !       %BEGIN - %END  BLOCK **
83    !
84    CLEAR TTY:
85          %IF INPUT # 0 %THEN:
86             SKIP SYMBOL %UNTIL NEXT SYMBOL = NL
87          %FINISH
88    READ NEW COMMAND:
89       %FAULT9 -> CLEAR TTY
90    !
91    %BEGIN    ;!  LEVEL 1
92    %INTEGER  LVL, BRACKETS, ESCAPE CELL, MODIFIER,LAST TEXT
93    %OWNINTEGER  NAME MASK=8_177, BODY MASK=8_377
94    %OWNINTEGER  QUERY=8_400000, INVERT=8_200000, DUMMY=8_100000
95    %OWNINTEGER  PROTECT = 8_200, RERUN = 0
96    !                                  A,B, C,D, E, F, G, H,I, J, K,
97    %OWNINTEGERARRAY  PRIM('A':'Z')= 6,7,-1,8,11,-1,14,-1,9,-1,-1,
98    !       L, M, N, O, P, Q, R, S, T, U,V, W, X, Y, Z
99          -1,12,-1,-1,13,-1,-1,10,-1,15,5,-1,-1,-1,16
100   !
101   %ROUTINESPEC    EXPAND(%INTEGER  SOURCE BODY,MACRO DEFNS,PARM BASE,STP)
102   !
103   L1:PROMPT('>'); SKIP SYMBOL %AND -> L1 %IF NEXT SYMBOL = NL
104      %IF PROTECT # 0 %AND NEXT SYMBOL = '*' %THEN:
105         SKIP SYMBOL; SKIP SYMBOL    ;! PROTECTED DEFINITIONS ARE...
106                                     ;! ...TERMINATED BY NL-*-NL
107         PROTECT = 0; SELECT INPUT(COMMAND)
108      %ELSE:
109         W = BBOT; X = Y-1; ESCAPE CELL = Z
110         LVL = -1; BRACKETS = 0; LAST TEXT = 0
111         EXPAND(0,DUMMY+Y,-4,0)
112         ERROR(12) %IF BRACKETS # 0     ;!  USER TYPING ERROR
113      %FINISH
114      -> L1 %IF LVL = 0  ;! IF IT WAS A MACRO DEFN.
115                         ;! (SEE 'CONTROL(6):' IN ROUTINE 'EXPAND')
116      RERUN = 1          ;! DECODER BUFFER HOLDS COMPLETE COMMAND
117   !
118   %ROUTINE  EXPAND(%INTEGER SOURCE BODY,MACRO DEFNS,PARM BASE,STP)
119   %INTEGER  P,NEXT
120      %ROUTINE  WPERROR(%INTEGER  N); W = P; ERROR(N); %END
```

```
121    %OWNINTEGER  I,J,K,Q,R,PARM
122    %OWNINTEGER  CODE WORD, DELIMITER
123    %SWITCH  SW(1:3), CONTROL(0:6)
124    %ROUTINESPEC  READ NUMBER(%INTEGERNAME X)
125    !
126        %INTEGERFN  STRING DELIMITER
127            %RESULT = NEXT %IF NEXT='/' %OR NEXT='.' %C
128                                        %OR NEXT=':' %OR NEXT=''''
129            %RESULT = \NEXT
130        %END
131    !
132        %ROUTINE  SET NEXT
133            %IF LVL = 0 %THEN:
134                READ SYMBOL(NEXT) %IF NEXT # NL
135            %ELSE:
136                %IF SOURCE BODY # MACRO DEFNS&8 77777 %THEN:
137                    NEXT = B(SOURCE BODY); SOURCE BODY = SOURCE BODY+1
138                %ELSE  NEXT = NL
139            %FINISH
140        %END
141    !
142        %ROUTINE  REMAP ALPHA
143            NEXT = NEXT-32 %IF 'A' <= NEXT-32 <= 'Z'
144        %END
145    !
146        %PREDICATE  STRING IS EMPTY
147            %TRUE %IF NEXT = NL
148            %FALSE
149        %END
150        !
151        %ROUTINE  PUSHBR(%INTEGER  TYPE)
152            STP = STP+4; BRACKETS = BRACKETS+1
153            ERROR(13) %IF STP > STKTOP
154            ! CHAIN ESCAPE CELL ONTO LIST
155            B(X-3) = ESCAPE CELL; ESCAPE CELL = X-3
156            ST(STP+1) = LVL
157            ST(STP+2) = TYPE
158            ST(STP+3) = X
159            X = X-4; ERROR(13) %IF X < W
160        %END
161        !
162        %ROUTINE  PLANT(%INTEGER  I)
163            ERROR(13) %IF X <= W
164            B(X) = I; X = X-1
165        %END
166        !
167        %ROUTINE  POPBR(%INTEGER  TYPE,REPEAT)
168        %INTEGER  K,L,M
169            ! TEST FOR DECODER STACKING ERROR
170            ERROR(-1) %IF STP < 0
171            BRACKETS = BRACKETS-1
172            PLANT(RBRCODE)
173            TYPE = ST(STP+2)!TYPE
174            K = ST(STP+3)
175            %IF TYPE&QUERY # 0 %OR BRACKETS = 0 %THEN:
176                M = 0; M = -(BRACKETS+1) %IF TYPE&QUERY # 0
177                %WHILE ESCAPE CELL < K %DO:
178                    L = ESCAPE CELL; ESCAPE CELL = B(L)
179                    B(L) = M
180                %REPEAT
```

```
181         %FINISH
182         L = LBRCODE; L = L+1 %IF TYPE&INVERT # 0
183         B(K) = L          ;!  PLANT CODE FOR '(' OR '\('
184         B(K-1) = X        ;!  LENGTH CELL
185         B(K-2) = REPEAT   ;!  LOOP COUNT
186         STP = STP-4
187      %END
188      !
189      !
190      !
191      !
192      !
193      !
194      !
195      !
196      !
197      !
198      !
199      !
200      !
201      ! USE 'PUSHBR' TO INSERT AN EXTRA LAYER OF BRACKETS AROUND
202      ! THE STRING CURRENTLY BEING EXPANDED IF:
203      ! 1) THE STRING IS THE COMMAND STRING FROM THE USER RATHER
204      !     THAN  A PARTIALLY EXPANDED VERSION OF IT.
205      ! 2) '\' OR '?' TO QUALIFIED THE CALL ON A MACRO. E.G. D3?
206      !     THIS EXTRA LAYER TO AVOIDS INTERACTION BETWEEN LEVELS.
207      ! FLAG BIT IN 'MACRO DEFNS' MARKS THIS EXTRA LEVEL SO THAT
208      ! IT CAN BE POPPED ON EXIT FROM ROUTINE.
209      LVL = LVL+1
210      MODIFIER = 0
211      PUSHBR(MACRO DEFNS) %IF MACRO DEFNS & 8 700000 # 0
212      NEXT = ' '          ;!   CONTENTS ARE CURRENTLY UNDEFINED
213      %UNTIL STRING IS EMPTY %DO:
214         SET NEXT %WHILE NEXT = ' '
215   %INTEGERFN CHAR
216   %OWNINTEGERARRAY AA(1:6)= ')', '(', ',', '\', '!', '['
217   %INTEGER I
218      %FOR I=1,1,6 %DO:
219         %RESULT = I %IF AA(I) = NEXT
220      %REPEAT
221      %RESULT = 0
222   %END
223         -> CONTROL(CHAR)
224   !
225   !
226   !
227   !
228   !
229   CONTROL(0):      ! ** READ IN COMMAND ATOM IDENTIFIER **
230   %ROUTINE  READ COMMAND ATOM ID
231      RERUN = 0          ;!  PREVIOUS COMMAND NOW BECOMES UNREUSABLE
232      %IF NEXT = '$' %THEN:
233         I = W
234         %UNTIL %NOT 'A' <= NEXT <= 'Z' %DO:
235            I = I+1; B(I) = NEXT
236            SET NEXT; REMAP ALPHA
237         %REPEAT
238      %ELSE:
239         REMAP ALPHA
240         I = W+1; B(I) = NEXT; SET NEXT
```

```
241         %FINISH
242         B(W) = (I-W+1) + PROTECT
243         ERROR(0) %UNLESS 'A' <= B(I) <= 'Z'
244         ! ** COULD INSERT PACKING ROUTINE HERE:  MUST RESET B(W) **
245     %END
246               READ COMMAND ATOM ID
247               !
248               ! ** NOW LOOK IT UP IN DEFN. TABLE **
249     %ROUTINE  LOOK UP
250         P = MACRO DEFNS&8_77777; J = B(W) & NAME MASK
251         %WHILE  P # Z %DO:
252             %IF B(P) & NAME MASK = J %THEN:
253                 %FOR K = 1,1,J-1 %DO:
254                     -> L1 %IF B(P+K) # B(W+K)
255                 %REPEAT
256                 ! IDENTIFIER FOUND:  SET P,Q AND R TO POINT TO THE
257                 ! HEAD OF NAME, END OF NAME/HEAD OF BODY AND END OF
258                 ! BODY RESPECTIVELY.
259                 Q = (B(P)&NAME MASK) + P
260                 R = (B(Q)&BODY MASK) + Q
261                 %RETURN
262                 !
263             %ELSE:
264     L1:         P = ( B(P) & NAME MASK ) + P
265                 P = ( B(P) & BODY MASK ) + P
266             %FINISH
267         %REPEAT
268         ! IDENTIFIER ISN'T IN TABLE.
269         ! INDEX INTO THE PRIMITIVE FNS. TABLE IF:
270         ! 1) IDENTIFIER IS A SINGLE LETTER STRING, AND:
271         ! 2) IT CAME FROM A MACRO BODY, NOT THE TELETYPE DIRECTLY.
272         !    FAILURE OTHERWISE.
273         ! SET Q = P TO IDENTIFY PRIMITIVES IN THE OUTSIDE WORLD.
274         !
275         %IF LVL > 0 %AND J = 2 %THEN P = PRIM(B(W+1)) %ELSE P = -1
276         Q = P
277     %END
278     !
279               LOOK UP; ERROR(1) %IF P < 0        ;! IDENTIFIER NOT FOUND
280               %IF P # Q %THEN:
281                   ! IT'S A MACRO CALL:  SET UP PARAMETERS USING
282                   ! CODE WORD THEN CALL 'EXPAND' RECURSIVELY.
283                   CODE WORD = B(Q) >> 8
284                   PARM = 0; DELIMITER = 0
285                   %WHILE CODE WORD # 0 %DO:
286                       K = CODE WORD & 3; CODE WORD = CODE WORD >> 2
287                       -> SW(K)
288     !
289     !       ** READ NUMERICAL PARAMETER **
290     %ROUTINE  READ NUMBER(%INTEGERNAME X)
291         I = 0; J = +1
292         %IF NEXT = '-' %OR NEXT = '+' %THEN:
293             J = -1 %IF NEXT = '-'; SET NEXT
294         %FINISH
295         SET NEXT %WHILE NEXT = ' '
296         %IF NEXT = '*' %THEN:
297             I = STAR; SET NEXT; -> L1
298         %FINISH
299         %IF NEXT = '#' %AND LVL # 0 %THENSTART
300             SET NEXT; K = NEXT-'0'; SET NEXT
```

```
301         !  *TEST PARAMETER OFFSET*
302         WPERROR(2) %UNLESS 1 <= K <= ST(PARM BASE)
303         K = ST(PARM BASE + K)
304         ! *WRONG PARAMETER TYPE*
305         WPERROR(3) %IF K & (8_600000) # 8_200000
306         I = B(K & 8_77777)     ;! 'AND' OFF TYPE INFORMATION
307      %ELSE:
308         %WHILE '0' <= NEXT <= '9' %DO:
309            I = (((I<<2)+I)<<1) + (NEXT-'0')    ;! I=10*I+(NEXT-'0')
310            SET NEXT
311         %REPEAT
312         I = STAR-1 %IF I >= STAR
313      %FINISH
314      I = 1 %IF I = 0
315   L1:I = -I %IF J < 0
316      X = I
317   %END
318   !
319   !
320   SW(1):    ! READ ONLY +VE NUMERICAL VALUES
321               WPERROR(11) %IF NEXT = '-'
322   SW(2):    ! READ +VE OR -VE
323               READ NUMBER(K); J = BBOT ;! READ NUMBER CORRUPTS J
324               %WHILE J # W %DO:
325                  -> L2 %IF B(J) = K     ;!  GOT A MATCH YET?
326                  J = J+1
327               %REPEAT
328               B(W) = K; W = W+1          ;! CURRENT 'J' = OLD 'W'
329            L2:J = J+8_200000
330               -> L3
331   !
332   SW(3):       !  ** READ TEXT PARAMETER **
333               SET NEXT %WHILE NEXT = ' '
334               %IF NEXT = '#' %AND LVL # 0 %THENSTART
335                  SET NEXT; K = NEXT-'0'; SET NEXT
336                  !  *PARAMETER OFFSET OUT OF RANGE*
337                  WPERROR(2) %UNLESS 1 <= K <= ST(PARM BASE)
338                  K = K + PARM BASE
339                  ! *WRONG PARAMETER TYPE*
340                  WPERROR(3) %IF ST(K)&(8_600000) # 8_400000
341                  J = ST(K)
342               %ELSE:
343                  %IF NEXT = '-' %AND LAST TEXT # 0 %THEN:
344                     ERROR(4) %IF LVL # 0    ;! *LEGAL ONLY FROM TT*
345                     J = LAST TEXT; SET NEXT; -> L3
346                  %FINISH
347                  REMAP ALPHA
348                  %IF NEXT = 'N' %THEN:   ;! REPLACES QUOTED NEWLINE
349                     J = W+8_400000; B(W) = 1; B(W+1) = NL; W = W+2
350                     SET NEXT; -> L3
351                  %FINISH
352                  !  *TEST FOR ILLEGAL STRING DELIMITER*
353                  ERROR(5) %UNLESS NEXT = STRING DELIMITER
354                  DELIMITER = NEXT
355                  I = W
356                  %UNTIL  NEXT = DELIMITER %DO:
357                     ! *TEST FOR MISSING CLOSING TEXT DELIMITER*
358                     WPERROR(5) %IF STRING IS EMPTY
359                     REMAP ALPHA     ;!  ** UPPERCASE CONVERSION? **
360                     B(I) = NEXT; I = I+1
```

```
361                     SET NEXT
362                 %REPEAT
363                 ! SKIP OVER TRAILING TEXT DELIMITER
364                 SET NEXT %UNLESS CODE WORD & 3 = 3
365                 J = W + 8_400000; LAST TEXT = J
366                 B(W) = I-W-1; W = I
367             %FINISH
368             -> L3
369 !
370 !
371 !
372 L3:             PARM = PARM+1; ST(STP+PARM+4) = J
373             %REPEAT     ;!  WHILE CODE WORD # 0
374             ST(STP+4) = PARM          ;! RECORD NO. OF PARAMETERS
375             SET NEXT %WHILE NEXT = ' '
376             %IF NEXT = '?' %THEN:
377                 MODIFIER = MODIFIER + QUERY; SET NEXT
378             %FINISH
379             EXPAND(Q+1,MODIFIER+R,STP+4,STP+PARM+5)
380         %ELSE:
381             ! 'P' HOLDS NUMERIC CODE FOR PRIMITIVE FN.
382             ! CAN ONLY OCCUR IF LVL > 0 ( SEE  'LOOK UP' )
383             ! PRIMITIVES CAN HAVE UP TO FOUR PARAMETERS BUT NOTE
384             ! ALL CONVERSION MUST HAVE BEEN PREVIOUSLY DONE WHEN
385             ! PASSING PARAMETERS IN GENUINE MACRO CALLS, I.E.
386             ! BY RECURSIVE CALLS ON 'EXPAND'.  FOR EXAMPLE,
387             ! -#1 IS NOT LEGAL WHEN PLANTING A CALL ON
388             ! A PRIMITIVE.
389             PLANT(P)              ;!  STORE FN CODE
390             %WHILE NEXT = '#' %DO:
391                 SET NEXT; K = NEXT-'0'; SET NEXT
392                 WPERROR(2) %UNLESS 1 <= K <= ST(PARM BASE)
393                 PLANT( ST(PARM BASE + K) )
394             %REPEAT
395         %FINISH
396         -> REPEAT
397 !
398 CONTROL(1):         ! **  ')'  **
399       SET NEXT %UNTIL NEXT # ' '
400       READ NUMBER(PARM); SET NEXT %WHILE NEXT = ' '
401       %IF !PARM! = STAR %THEN POPBR(QUERY,PARM) %ELSE:
402           %IF NEXT # '?' %THEN POPBR(0,PARM) %ELSE:
403               POPBR(QUERY,PARM); SET NEXT
404           %FINISH
405       %FINISH
406       -> REPEAT
407 !
408 CONTROL(2):         ! **  '('  **
409       RERUN = 0         ;!  PREVIOUS COMMAND IS NOT NOW REUSABLE
410       PUSHBR(0); SET NEXT
411       -> REPEAT
412 !
413 CONTROL(3):         ! ** ','  **
414       SET NEXT      ;!  SKIP OVER ','
415       PLANT(RBRCODE)
416       K = ST(STP+3)    ;!  START OF CURRENT BRACKETTING LEVEL
417       I = ((X&8_1777)<<5) + (BRACKETS&8_37)
418       %WHILE ESCAPE CELL < K %DO:
419           J = ESCAPE CELL; ESCAPE CELL = B(J)
420           B(J) = I
```

```
421         %REPEAT
422         K = ESCAPE CELL; ESCAPE CELL = K; PLANT(K)
423         -> REPEAT
424   !
425   CONTROL(4):         ! ** '\'  **
426         SET NEXT %UNTIL NEXT # ' '
427         %IF NEXT # '(' %THEN MODIFIER = INVERT %ELSE:
428            PUSHBR(INVERT); SET NEXT
429         %FINISH
430         -> REPEAT
431   !
432   CONTROL(5):         ! ** '!' **
433         SET NEXT %UNTIL NEXT # ' '
434         READ NUMBER(PARM)
435         K = 0
436         K = -BRACKETS %AND SET NEXT %IF NEXT = '?'
437         !  'STAR' IS A SPECIAL NUMBER WHICH BY ITS VALUE IMPLIES
438         !  '*' OR '?'.  'READ NUMBER' CAN ONLY SUPPLY THIS VALUE
439         !  IN THIS CONTEXT SO NO AMBIGUITY ARISES.
440         K = -BRACKETS %IF PARM = STAR
441         ERROR(14) %UNLESS STRING IS EMPTY %AND RERUN # 0
442         B(Y-3) = PARM
443         B(Y-4) = K
444         BRACKETS = 0; LVL = -1
445         %RETURN
446   !
447   CONTROL(6):     !        ** DEFINE NEW MACRO **
448         ! *DEFINITIONS CANNOT BE CONCATENATED:  CHECK THIS*
449         ERROR(6) %UNLESS W = BBOT %AND X = Y-5
450         ! Y-5 ABOVE IS Y-1 REALLY BUT ALLOWS FOR LVL ZERO BRACKET
451         SET NEXT %UNTIL NEXT # ' '
452         READ COMMAND ATOM ID
453         K = B(W)!!PROTECT
454         ! *MACRO NAME LONGER THAN NAME MASK-2*
455         ERROR(10) %IF K & NAME MASK # K
456         W = W+K
457         SET NEXT %WHILE NEXT = ' '
458         CODE WORD = 0
459         %IF NEXT = '(' %THEN:   ;! ...GENERATE PARAMETER CODE-WORD
460            SET NEXT     ;!     SKIP OVER '('
461            %OWN INTEGER ARRAY  PARMTYPE(1:3)= '+', '-', 'T'
462            %UNTIL NEXT = ')' %DO:
463               ! *MAX. OF 4 PARAMS./MACRO: TEST FOR MORE*
464               ERROR(7) %IF CODE WORD & 8_003400 # 0
465               REMAP ALPHA     ;!  PUT LETTERS IN STANDARD FORM
466               %FOR K = 1,1,3 %DO:
467                  %IF PARMTYPE(K) = NEXT %THEN:
468                     CODE WORD = (CODE WORD << 2) + K; -> L4
469                  %FINISH
470               %REPEAT
471               ERROR(7)          ;!  *ILLEGAL TYPE SPECIFIER*
472            L4:SET NEXT
473            %REPEAT
474            K = 0
475            %WHILE CODE WORD # 0 %DO:
476               K = (K<<2) + (CODE WORD & 3)
477               CODE WORD = CODE WORD >> 2
478            %REPEAT
479            CODE WORD = K << 8
480            SET NEXT %UNTIL NEXT # ' '
```

```
481         %FINISH
482         %IF NEXT # '=' %THEN ERROR(8) %BLSE:  ;! *'=' MISSING*
483            SET NEXT %UNTIL NEXT # ' '
484         %FINISH
485         I = W
486         %WHILE NEXT # NL %DO:
487            REMAP ALPHA; I = I+1; B(I) = NEXT; SET NEXT
488         %REPEAT
489         ! TEST FOR AND THROW AWAY THE CLOSING ']'
490         ERROR(8) %UNLESS B(I) = ']'          ;!      *']' MISSING*
491         ! ** PACKING ROUTINE CAN BE INSERTED HERE IF REQUIRED **
492         !  SETTING 'I' APPROPRIATELY WILL AUTOMATICALLY SET ALL
493         !  DERIVATIVE VALUES CORRECTLY.
494         K = I-W; B(W) = K + CODE WORD
495         W = BBOT
496         ERROR(10) %IF K & BODY MASK # K ;! *MACRO BODY TOO LONG*
497         LOOK UP    ;!   SEARCH EXISTING MACRO TABLE FOR NEW ID  ...
498         W = I      ;!   ... THEN RESET 'W'
499   !  MACROS ARE OF TWO TYPES, PROTECTED AND UNPROTECTED.  THE
500   !  FORMER ARE LOADED BEFORE CONTROL IS PASSED TO THE USER AND
501   !  DEFINE THE BASIC COMMAND SET FOR THAT RUN OF THE EDITOR;
502   !  THEY CANNOT BE DYNAMICALLY ALTERED.  UNPROTECTED MACROS ARE
503   !  CREATED BY THE USER TO MEET A PARTICULAR TRANSIENT NEED AND
504   !  CAN BE ALTERED AS REQUIRED.
505   !     IN ORDER THAT ANY NAME AVAILABLE TO THE USER HAVE A UNIQUE
506   !  MEANING, UNPROTECTED MACROS ARE DESTRUCTIVELY RE-DEFINED
507   !  RATHER THAN BEING PUSHED DOWN.  SINCE PROTECTED MACROS CANNOT
508   !  BE RE-DEFINED DYNAMICALLY, THIS PROBLEM DOES NOT ARISE WITH
509   !  THEM: TO AVOID PROBLEMS WHICH WOULD ARISE IF UNOFFICIAL
510   !  INTERMEDIATE NAMES, USED IN IMPLEMENTING THE BASIC EDITOR
511   !  COMMANDS WERE SUBSEQUENTLY ACCESSED BY ITS USERS, WITHIN THE
512   !  PROTECTED MACROS ONLY, THE SAME IDENTIFIER CAN BE USED
513   !  TO REFER TO DIFFERENT MACRO BODIES.  SINCE THE LOOK-UP ONLY
514   !  SEARCHES THAT PART OF THE TABLE DEFINED BEFORE THAT MACRO
515   !  WHOSE BODY IS CURRENTLY BEING EXPANDED, AND EXITS AS SOON AS
516   !  A MATCH IS OBTAINED, THE MEANING IS IN FACT UNAMBIGUOUS.  THE
517   !  USER ONLY HAS ACCESS TO THE MOST RECENTLY DEFINED MACRO WITH
518   !  A PARTICULAR NAME.
519   !
520         %IF P > 0 %AND PROTECT = 0 %THEN:
521            ! IF NAME PRESENT(P>0) AND INITIAL LOAD DONE(PROTECT=0)
522            ! *COMPLAIN IF PRIMITIVE OR PROTECTED*
523            W = BBOT %AND ERROR(9) %IF P=0 %OR B(P) # B(BBOT)
524            K = (R-P) - (W-BBOT)   ;! SIZE DIFFERENCE( +VE/-VE )
525            MOVE(B,Y,P-1,K); Y = Y+K
526            MOVE(B,BBOT,W-1,R-W)
527         %ELSE:
528            ! ADD NEW IDENTIFIER
529            Y = Y+(BBOT-W)
530            MOVE(B,BBOT,W-1,Y-BBOT)
531         %FINISH
532         BRACKETS = 0   ;!  TO PREVENT SUBSEQUENT FAILURE
533         %RETURN   ;!  N.B.  LEAVES  LVL=0, NOT -1, ON RETURN TO
534                   ;!  CALLER.  THIS IS USED TO DISTINGUISH MACRO
535                   ;!  DEFINING COMMANDS FROM OTHERS.
536   !
537   REPEAT: %REPEAT
538      POPBR(0,1) %IF MACRO DEFNS & 8_700000 # 0
539      LVL = LVL-1
540   %END    ;!  OF %ROUTINE EXPAND
```

```
%END    ;!  OF LEVEL 1 %BEGIN BLOCK
!
!
!
! ** START OF EXECUTING ALGORITHM FOR PROGRAM ASSEMBLED ABOVE **
!
%INTEGER  COND MASK,OK EXIT,ERROR WORD,LOOP START,LOOP COUNT
%INTEGER  PC,STP,CURRENT FN, COND
%INTEGER  I,J,K
%OWNINTEGER  OK = 1, FAIL = 0
%SWITCH  FN(1:16)
!
!
   PC = Y-1; STP = 0; -> START
!
TEST:  -> ERROR EXIT %IF COND # COND MASK
XCT NEXT: -> READ NEW COMMAND %IF STP = 0 ;! DONE ON EMPTY STACK
START:  CURRENT FN = B(PC); PC = PC-1
   -> FN(CURRENT FN)
!
!
%ROUTINE  SAVE AND RESET        ;!  ** PUSH RUN-TIME STACK **
   ST(STP+1) = COND MASK
   ST(STP+2) = LOOP START
   ST(STP+3) = LOOP COUNT
   ST(STP+4) = ERROR WORD
   COND MASK = I
   OK EXIT = B(PC); PC = PC-1
   LOOP COUNT = B(PC); PC = PC-1
   LOOP START = PC; ERROR WORD = PC; PC = PC-1
   STP = STP+5; ST(STP) = OK EXIT
%END
!
%ROUTINE RESTORE OLD CONTEXT    ;!  ** POP RUN-TIME STACK **
   PC = ST(STP)
   STP = STP-5
   ! *TEST FOR EXECUTION-TIME STACKING ERROR*
   ERROR(-2) %IF STP < 0
   OK EXIT = ST(STP)
   COND MASK = ST(STP+1)
   LOOP START = ST(STP+2)
   LOOP COUNT = ST(STP+3)
   ERROR WORD = ST(STP+4)
%END
!
!
FN(1):  I = OK; -> L1    ;!  '('
FN(2):  I = FAIL         ;!  '\('
L1: SAVE AND RESET
   -> XCT NEXT
!
!
!
!
FN(3):  ;!      ')'
          LOOP COUNT = LOOP COUNT - 1
          %IF LOOP COUNT <= 0 %THEN RESTORE OLD CONTEXT %ELSE:
             ERROR WORD = LOOP START; PC = LOOP START - 1
         %FINISH
          -> XCT NEXT
```

```
601   !
602   FN(4):  ERROR(-99)              ;! SPARE CONTROL FUNCTION WHICH IN
603                                   ;! FACT WAS NOT ULTIMATELY REQUIRED.
604   !
605   ERROR EXIT:    ;!  SOFTWARE INTERRUPTS COME HERE WHEN A FN. FAILS
606      ERROR WORD = B(ERROR WORD)
607      %IF ERROR WORD > 0 %THEN:          ;! A COMMA: RESET PC AND STP
608         !  STP = 5*(ERROR WORD & 8_37)
609         STP = ERROR WORD & 8_37; STP = (STP<<2) + STP
610         ERROR WORD = ERROR WORD >> 5
611         PC = ERROR WORD - 1             ;! PC ADVANCES BACKWARD
612         COND = OK
613      %ELSE:
614         %IF ERROR WORD = 0 %THEN:       ;!  ...AN UNFORESEEN FAILURE
615            %PRINTTEXT '*FAILED'; NEWLINE; -> READ NEW COMMAND
616         %FINISH
617         !  STP = 5*!ERROR WORD!         ;! ...A FORESEEN FAILURE
618         STP = -ERROR WORD; STP = (STP<<2) + STP
619         RESTORE OLD CONTEXT
620      %FINISH
621      -> XCT NEXT
622   !
623   !
624   !
625   !         ** EDITING ROUTINES FOLLOW **
626   !
627   %OWNINTEGER  MAX LINE SIZE = 100
628   %INTEGER  S,T,U,V,CP,LP,LB END,LB LINES,ALTERED,CHANGED
629   %INTEGER  LINE ORIGIN, LINE OFFSET
630   %INTEGER  P1,P2
631   %INTEGER  IN, OUT
632   !
633   %ROUTINE INITIALISE
634   ! THIS ROUTINE IS CALLED AT THE BEGINNING BUT PUT HERE SO'S IT
635   ! CAN REFERENCE VARIABLES NOT DECLARED AT THE POINT OF CALL.
636      S = AMIN; T = S; AA(T) = 0
637      U = AMAX; V = U; AA(V) = 0
638      LINE ORIGIN = 0; LINE OFFSET = 1; LB LINES = 0
639      ALTERED = 0; CHANGED = 0
640      LP = 0
641      IN = 1; OUT = 3
642      SELECT INPUT(MACROS); SELECT OUTPUT(REPORT)
643   %END
644   !
645   %PREDICATE  GAP IS TOO NARROW
646      ! ROUTINES  'UNPACK'  AND  'READ LINE'  BELOW BOTH ASSUME
647      ! THAT SUFFICIENT FREE BUFFER SPACE IS AVAILABLE AND DO NOT
648      ! THEMSELVES CHECK.  USE 'GAP IS TOO NARROW' TO DO THIS
649      ! TEST WHEREVER APPROPRIATE
650      %FALSE %IF U-T > MAX LINE SIZE + 10
651      %TRUE
652   %END
653   !
654   !  'PACK' AND 'UNPACK' CAN BE CHANGED TO GIVE A HIGHER PACKING
655   !  DENSITY IN THE LINE RECORDS STORED BY THE MAIN BUFFER WITHOUT
656   !  THE REST OF THE EDITOR BEING AFFECTED AT ALL:  BUT NOTE THAT
657   !  IF ONE IS CHANGED, THE OTHER MUST BE TOO.
658   %ROUTINE  PACK
659   %INTEGER  I,K
660      I = T
```

```
661       %IF ALTERED # 0 %THEN:
662          CHANGED = 1
663          !  TWO MARKERS, 'ALTERED' AND 'CHANGED' ARE USED.  THESE
664          ! BEING NON-ZERO DENOTE RESPECTIVELY THAT (1) THE CURRENT
665          ! LINE HAS BEEN MODIFIED AND MUST BE RE-PACKED, NOT
666          ! MERELY COPIED, AND (2) THAT THE TEMPORARY FILE CURRENTLY
667          ! BEING WRITTEN HAS BEEN MODIFIED RELATIVE TO THE PREVIOUS
668          ! ONE AND MUST THEREFORE BE COPIED IN ITS ENTIRETY WHEN
669          ! GOING BACKWARDS, NOT MERELY REWOUND AND RE-READ.
670          K = AA(LP)
671          %WHILE K # NL %DO:
672              I = I+1 %AND AA(I) = K %IF K # 0  ;! SKIP NULLS
673              LP = LP+1; K = AA(LP)
674          %REPEAT
675          ! ** PACKING ROUTINE COULD BE INSERTED HERE OR IN THE LOOP
676          !     ABOVE:  'I' POINTS TO THE LAST CELL OCCUPIED **
677          %IF LP = LB END %THEN LP = 0 %AND ALTERED = 0 %ELSE:
678             K = I-LP                    ;!  'T' IS RESET TO 'I' LATER
679             %IF K # 0 %THEN:
680                MOVE(AA,LP,LB END,K)
681                LP = LP+K; LB END = LB END + K
682             %FINISH
683             LP = LP+1  ;! BUMP PAST LINK CELL TO START OF NEXT LINE
684          %FINISH
685       %ELSE:
686          %FOR K = U+1,1,U+AA(U)-1 %DO:
687             I = I+1; AA(I) = AA(K)
688          %REPEAT
689          U = U+AA(U); LP = 0          ;!  'ALTERED'  ALREADY IS ZERO
690       %FINISH
691       I = I+1
692       AA(I) = I-T; T = I     ;!  NOTE RELOCATABLE POINTERS
693       LB LINES = LB LINES + 1; LINE OFFSET = LINE OFFSET + 1
694    %END
695    !
696    %ROUTINE  MARK LB
697       %IF ALTERED = 0 %THEN:
698          ERROR(-3) %IF LB LINES < 0       ;! *-VE NO. OF LINES?*
699          U = U+AA(U)
700          ALTERED = 1
701       %FINISH
702    %END
703    !
704    %ROUTINE  UNPACK
705    %INTEGER  I,K,X
706       X = LP
707       %IF X = 0 %THEN:
708          LP = T+1; LB END = LP; CP = LP
709       %FINISH
710       ! MUST UNPACK NULL LINES('NL' ONLY) SO %FOR WON'T DO
711       I = U+1; K = U+AA(U)
712       %WHILE I # K %DO:
713          AA(LB END) = AA(I)
714          LB END = LB END + 1; I = I+1
715       %REPEAT
716       !  ** UNPACKING ROUTINE CAN BE INSERTED HERE OR IN THE
717       !     PRECEDING LOOP **
718       AA(LB END) = NL
719       %IF X=0 %THEN ALTERED = 0 %ELSE U = U+AA(U) %AND ALTERED = 1
720       LB LINES = LB LINES - 1
```

```
721   %END
722   !
723   %ROUTINE  READ LINE(%INTEGER  STREAM)
724   %INTEGER  S,K
725      PACK %WHILE LP # 0
726      COND = OK
727      SELECT INPUT(INSTREAM(STREAM)) %IF STREAM # COMMAND
728      %FAULT9 -> L1
729      LP = T+1; LB END = T; K = LB END + MAX LINE SIZE - 1
730      %UNTIL S = NL %DO:
731         READ SYMBOL(S)
732         S = S-32 %IF 'A' <= S-32 <= 'Z' ;! ** UPPERCASE LETTERS **
733         LB END = LB END + 1
734         AA(LB END) = S
735         -> L1 %IF LB END = K
736      %REPEAT
737      -> L2
738   !
739   L1:COND = FAIL %AND -> L3 %IF LB END < LP
740      LB END = LB END + 1; AA(LB END) = NL
741   L2:ALTERED = 1
742   L3:SELECT INPUT(COMMAND)
743   %END
744   !
745   !
746   %ROUTINE  DUMP LINE(%INTEGER  STREAM)
747   %INTEGER  I
748      SELECT OUTPUT(OUTSTREAM(STREAM)) %IF STREAM # REPORT
749      %FOR I = LP,1,LB END %DO:
750         PRINT SYMBOL(8_136) %IF STREAM = REPORT %AND I = CP # LP
751         PRINT SYMBOL( AA(I) ) %IF AA(I) # 0
752      %REPEAT
753      SELECT OUTPUT(REPORT)
754   %END
755   !
756   %ROUTINE  NORMALISE
757   %INTEGER  I,K,X
758      ! SPLITS BUFFER UP SO THAT 'CURRENT LINE' IS LINE WHOSE
759      ! LENGTH CELL IS POINTED TO BY 'LP'.
760      !
761      %IF S <= LP <= T %THEN:
762         %IF AMAX-V # 0 %THEN:
763            MOVE(AA,U,V,AMAX-V)
764            U = U+AMAX-V; V = AMAX  ;!  I.E. V = V+AMAX-V
765         %ELSE:
766            %IF V = U %THEN:
767               U = AMAX; V = U; AA(V) = 0
768            %FINISH
769         %FINISH
770         !   SPLIT BLOCK S:T
771         ! REVERSE LENGTH CELL POINTERS FIRST
772         K = AA(T); X = T
773         %WHILE T # S %AND T >= LP %DO:
774            T = T-K; I = AA(T)
775            AA(T) = K; K = I
776         %REPEAT
777         MOVE(AA,T,X-1,U-X) %IF U-X # 0
778         U = U+T-X; AA(T) = K ;! RESTORE ORIGINAL AA(T)
779         %IF S # T %THEN;
780            MOVE(AA,S,T,AMIN-S) %IF AMIN-S # 0
```

```
781             T = T+AMIN-S; S = AMIN    ;!  I.E. S = S+AMIN-S
782          %FINISH
783       %ELSE:
784          %IF U <= LP <= V %THEN:
785             %IF AMIN-S # 0 %THEN:
786                MOVE(AA,S,T,AMIN-S)
787                T = T+AMIN-S; S = AMIN
788             %ELSE:
789                %IF S = T %THEN:
790                   S = AMIN; T = S; AA(T) = 0
791                %FINISH
792             %FINISH
793             %IF LP # U %THEN:    ;!  SPLIT BLOCK U:V
794                ! REVERSE POINTERS FIRST
795                K = AA(U); X = U
796                %UNTIL U = LP %DO:
797                   U = U+K; I = AA(U)
798                   AA(U) = K; K = I
799                %REPEAT
800                MOVE(AA,X+1,U,T-X) %IF T-X # 0
801                T = T+U-X; AA(U) = K    ;! RESTORE ORIGINAL AA(U)
802             %FINISH
803             %IF U # V %THEN:
804                MOVE(AA,U,V,AMAX-V) %IF AMAX-V # 0
805                U = U+AMAX-V; V = AMAX
806             %FINISH
807          %FINISH
808       %FINISH
809       LP = 0
810    %END
811    !
812    ! THE FOLLOWING ROUTINE CENTRES THE WINDOW ON THE TARGET LINE
813    ! WHOSE NUMBER IS PASSED AS A PARAMETER:  1 <= TARGET LINE <= N
814    ! WHERE THERE ARE N LINES IN THE FILE.
815    ! THIS ROUTINE RETURNS THE BUFFER TO THE CALLING PROGRAM IN GOOD
816    ! ORDER WITH FREE SPACE AVAILABLE FOR INSERTIONS AND SO ON:
817    ! MOVE TO(CURRENT LINE) IS THEREFORE USED TO CHECK THAT SPACE IS
818    ! AVAILABLE AND OBTAIN IT IF NOT.
819    ! THE OPERATION OF THIS ROUTINE AFFECTS THE EFFICIENCY OF THE
820    ! WHOLE PROGRAM QUITE CONSIDERABLY AND THE STRATEGIES EMPLOYED
821    ! HERE WILL DEPEND CONSIDERABLY ON THE OPERATING ENVIRONMENT.
822    ! ANY SERIOUS IMPLEMENTATION OF THIS PROGRAM WILL THEREFORE
823    ! ALMOST INEVITABLY REQUIRE THE REWRITING OF 'MOVE TO'.
824    %ROUTINE  MOVE TO(%INTEGER  TARGET LINE)
825    %INTEGER  I,K
826    %OWNINTEGER  AV LINES , STEP
827    %OWNINTEGER  CLOSE=16    ;!  CLOSE = NUMERIC CODE FOR PRIM('Z')
828    !
829       %ROUTINE  FILL UP(%INTEGER  LIMIT)
830          LP = V; NORMALISE
831          %WHILE AMAX-T > LIMIT %DO:
832             READ LINE(IN); %RETURN %IF COND = FAIL
833             PACK
834          %REPEAT
835       %END
836       !
837       %ROUTINE  EMPTY OUT(%INTEGER  N)
838          LP = S; NORMALISE
839          COND = FAIL
840          %UNTIL N <= 0 %DO:
```

```
841             LP = 0; %RETURN %IF U = V
842             UNPACK; DUMP LINE(OUT); U = U+AA(U)
843             LINE OFFSET=LINE OFFSET-1; LINE ORIGIN=LINE ORIGIN+1
844             N = N-1
845          %REPEAT
846          COND = OK
847       %END
848       !
849       TARGET LINE = 1 %IF TARGET LINE <= 0
850       PACK %WHILE LP # 0
851    !
852    L1:EMPTY OUT(LB LINES >> 1) %IF U-T < MAX LINE SIZE + 5
853    L2:%IF LINE ORIGIN < TARGET LINE < LINE ORIGIN + LB LINES %THEN:
854          K = TARGET LINE - (LINE ORIGIN + LINE OFFSET)
855          LINE OFFSET = LINE OFFSET + K
856          %IF K < 0 %THEN:
857             LP = T; I = -1
858             LP = LP-AA(LP) %AND I = I-1 %WHILE I # K
859          %ELSE:
860             LP = U; I = 0
861             LP = LP+AA(LP) %AND I = I+1 %WHILE I # K
862          %FINISH
863          -> L3 %IF LP = V
864          %IF K<0 %THEN STEP=0 %ELSE STEP=((AV LINES>>1)+STEP)>>1
865          NORMALISE; %RETURN
866       %FINISH
867    L3:%IF TARGET LINE > LINE ORIGIN %THEN:
868          EMPTY OUT( (TARGET LINE-LINE ORIGIN) %C
869                          -(((AVLINES<<1)+AVLINES)>>2)+STEP )
870          FILL UP( (MAX LINE SIZE + 5) << 1 )
871          %IF COND=OK %THEN AV LINES=(AV LINES + LB LINES)>>1 %ELSE:
872             %IF TARGET LINE  > LINE ORIGIN + LB LINES %THEN:
873                LP = 0; %RETURN %IF B(ERROR WORD) # 0
874                %PRINTTEXT '**END**'; NEWLINE; %MONITOR 31
875             %FINISH
876          %FINISH
877       %ELSE:
878          %IF CHANGED # 0 %THEN:   ;! DON'T COPY IF UNNECESSARY
879             %UNTIL COND = FAIL %DO:
880                EMPTY OUT(LB LINES << 1)
881                FILL UP( MAX LINE SIZE << 1 )
882             %REPEAT
883             EMPTY OUT(LB LINES << 1)
884          %FINISH
885          SELECT INPUT(INSTREAM(IN)); CLOSE INPUT
886          SELECT OUTPUT(OUTSTREAM(OUT)); CLOSE OUTPUT
887          SELECT INPUT(COMMAND); SELECT OUTPUT(REPORT)
888          PRINT SYMBOL('*'); K = OUT-2
889          %IF K < 0 %THEN %PRINTTEXT 'FILE' %ELSE:
890             PRINT SYMBOL('.'); PRINT SYMBOL('0'+K)
891          %FINISH
892          %PRINTTEXT ' COMPLETE'; NEWLINE
893          LINE ORIGIN = 0; LINE OFFSET = 1; LB LINES = 0
894          %IF CHANGED # 0 %THEN:
895             IN = 4 %IF IN = 1
896             K = OUT; OUT = IN; IN = K
897             OUT = 1 %IF CURRENT FN = CLOSE
898             FILL UP( (AMAX-AMIN) >> 1 )
899          %FINISH
900          CHANGED = 0
```

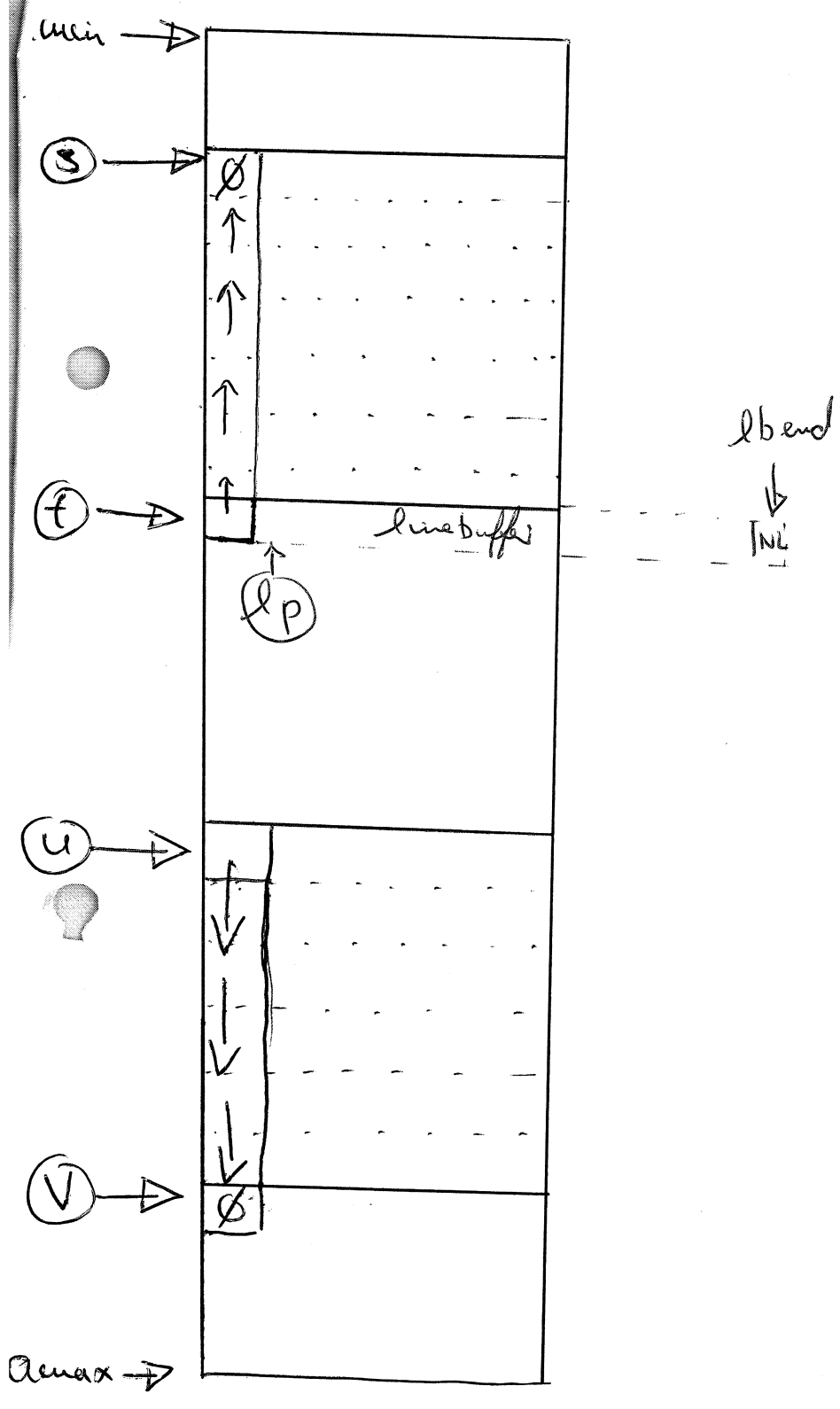
```
901         %FINISH
902         -> L2
903     %END
904     !
905     %ROUTINE  COMPARE STRINGS(%INTEGER  PNTR)
906     %OWNINTEGER  R,S
907     %INTEGER  I
908         ! INITIALISE IF PNTR >= 0
909         ! SEARCH START IS SPECIFIED IMPLICITLY BY P1
910         ! P2 SET ON SUCCESSFUL EXIT TO SYMBOL AFTER MATCHED STRING
911         !
912         %IF PNTR >= 0 %THEN:
913             MOVE TO(LINE ORIGIN + LINE OFFSET) %IF LP = 0
914             R = PNTR; S = PNTR+B(PNTR)
915             P1 = P1+1 %WHILE AA(P1) = 0
916             COND = FAIL
917         %FINISH
918         I = R; P2 = P1
919         %WHILE I # S %DO:
920             I = I+1
921             P2 = P2+1 %WHILE AA(P2) = 0
922             !  ** WILL REQUIRE CASE CONVERSION OF LETTERS HERE IF
923             !       NON-CASE-DEPENDENT STRING MATCHING IS DESIRED **
924             %RETURN %IF AA(P2) # B(I)
925             P2 = P2+1
926         %REPEAT
927         COND = OK
928     %END
929     !
930     %ROUTINE  SET P1 P2(%INTEGER  PNTR)
931         ! SETS P1,P2 TO TARGET STRING IF IT APPEARS TO RIGHT OF CP
932         ! IN CURRENT LINE
933         ! 'COND' IS SET BY 'STRINGS MATCH'
934         !
935         !
936         !
937         P1 = CP
938         COMPARE STRINGS(PNTR); %RETURN %IF COND = OK %OR LP = 0
939         %WHILE AA(P1) # NL %DO:
940             P1 = P1+1 %UNTIL AA(P1) # 0
941             COMPARE STRINGS(-1); %RETURN %IF COND = OK
942         %REPEAT
943     %END
944     !
945     %INTEGERFN  PARM
946     %INTEGER  P,Q
947         P = B(PC); Q = P & 8_77777; PC = PC-1
948         %RESULT = Q %IF P & 8_600000 = 8_400000    ;! A TEXT STRING
949         %RESULT = B(Q)                  ;!  OTHERWISE, MUST BE A NUMBER
950     %END
951     !
952     %ROUTINE  STEP(%INTEGER  N) ;! GO N POSITIONS LEFT(-), RIGHT(+)
953     %INTEGER  I,INCR,LIMIT
954         MOVE TO(LINE ORIGIN + LINE OFFSET) %IF LP = 0
955         %IF N # 0 %THEN:
956             N = -N; INCR = 1; LIMIT = LB END
957             INCR = -INCR %AND LIMIT = LP %IF N > 0
958             %IF !N! = STAR %THEN I = LIMIT %ELSE:
959                 I = CP; COND = FAIL
960                 %WHILE N # 0 %DO:
```

```
961                 %RETURN %IF  I = LIMIT    ;!  ... WITH COND = FAIL ...
962                 I = I+INCR %UNTIL AA(I) # 0
963                 N = N+INCR
964               %REPEAT
965             %FINISH
966          %FINISH
967          CP = I; COND = OK
968       %END
969       !
970       %ROUTINE  ERASE(%INTEGER  FROM,TO)
971       ! DELETE AA(FROM:TO-1) IF FROM < TO
972          AA(FROM) = 0 %AND FROM = FROM+1 %WHILE FROM < TO
973          MARK LB
974       %END
975       !
976       !
977       !  'INSERT' FAILS IF NO SPACE IS AVAILABLE FOR THE INSERTION
978       %ROUTINE  INSERT(%INTEGER  PARM)
979       %INTEGER  I,K,N
980          N = B(PARM); %RETURN %IF N = 0
981          %IF LP = 0 %THEN:  ;! CREATE NULL LINE IF BUFFER IS EMPTY
982             LP = T+1; LB END = LP; CP = LB END; AA(CP) = NL
983          %FINISH
984          MARK LB
985          K = CP
986          K = K-1 %WHILE K # LP %AND AA(K-1) = 0
987          %IF CP-K < N %THEN:
988             CP = CP+1 %WHILE AA(CP) = 0
989             %IF CP-K < N %THEN:
990                I = N-(CP-K)
991                COND = FAIL %AND %RETURN %IF LBEND+I >= U
992                MOVE(AA,CP,LB END,I); LB END = LB END + I
993             %FINISH
994          %FINISH
995          CP = K+N; PARM = PARM+1
996          AA(K+I) = B(PARM+I) %FOR I = 0,1,N-1
997          PACK %IF B(PARM) = NL %AND B(PARM-1) = 1
998          COND = OK
999       %END
1000      !
1001      !
1002      !  ** DEFNS OF EDITING OPERATIONS FOLLOW **
1003      !
1004      FN(5):    ;!   'VERIFY'     (V)
1005         P1 = CP
1006         COMPARE STRINGS(PARM)
1007         -> TEST
1008      !
1009      FN(6):    ;!   'AFTER'      (A)
1010         SET P1 P2(PARM)     ;!  SETS 'COND'
1011         CP = P2 %IF COND = OK
1012         -> TEST
1013      !
1014      FN(7):    ;!   'BEFORE'     (B)
1015         SET P1 P2(PARM)
1016         CP = P1 %IF COND = OK
1017         -> TEST
1018      !
1019      FN(8):    ;!   'DELETE'     (D)
1020         SET P1 P2(PARM)
```

```
1021        %IF COND = OK %THEN:
1022           MARK LB; CP = P1
1023           %IF CP = LB END %THEN:
1024              MOVE TO(LINE ORIGIN + LINE OFFSET) %IF U = V
1025              UNPACK
1026           %ELSE  ERASE(P1,P2)
1027        %FINISH
1028        -> TEST
1029     !
1030     FN(9):    ;!    'INSERT'      (I)
1031        INSERT(PARM)     ;!    SETS 'COND'
1032        -> TEST
1033     !
1034     FN(10):   ;!    'STEP'        (S)
1035        STEP(PARM); -> TEST
1036     !
1037     FN(11):   ;!    'ERASE'       (E)
1038        J = PARM; K = CP
1039        STEP(J); ERASE(K,CP) %IF COND = OK
1040        -> TEST
1041     !
1042     FN(12):   ;!    'MOVE'        (M)
1043        MOVE TO(LINE ORIGIN + LINE OFFSET + PARM)
1044        UNPACK %IF U # V
1045        COND = OK
1046        -> TEST
1047     !
1048     FN(13):   ;!    'PRINT'       (P)
1049        K = PARM-1
1050        %IF LP = 0 %THEN:
1051           MOVE TO(LINE ORIGIN + LINE OFFSET) %IF U = V
1052           UNPACK
1053        %FINISH
1054        CP = LP %IF K # 0
1055        DUMP LINE(0)
1056        %WHILE K > 0 %DO:
1057           PACK; MOVE TO(LINE ORIGIN + LINE OFFSET) %IF U = V
1058           UNPACK; DUMP LINE(0)
1059           K = K-1
1060        %REPEAT
1061        COND = OK
1062        -> TEST
1063     !
1064     FN(14):   ;!    'GET'         (G)
1065        J = PARM; J = 0 %IF J # 2
1066        %FOR I = 1,1,PARM %DO:     ;! N.B.  'PARM' IS ONLY CALLED ONCE
1067           MOVE TO(LINE ORIGIN + LINE OFFSET) %IF GAP IS TOO NARROW
1068           PROMPT(':'); READ LINE(J)
1069           %IF COND=FAIL %OR ( AA(LP)=':' %AND LBEND=LP+1 ) %THEN:
1070              COND = FAIL; LP = 0; -> TEST
1071           %FINISH
1072           PACK
1073        %REPEAT
1074        COND = OK
1075        -> TEST
1076     !
1077     FN(15):   ;!    'UP TO'       (U)
1078        SET P1 P2(PARM)
1079        MARK LB %AND ERASE(CP,P1) %IF COND = OK
1080        -> TEST
```

```
1081    !
1082    FN(16):    ;!    $CLOSE(1)/$REWIND(3)    .   (Z)
1083       %IF PARM = 1 %THEN:
1084          MOVE TO(O)       ;!  COPY UP REST OF CURRENT PASS
1085          CHANGED = 1; LINE ORIGIN = 1; MOVE TO(O)  ;! FINAL OUTPUT
1086       %ELSE:
1087          SELECT INPUT( INSTREAM(2) ); CLOSE INPUT
1088          SELECT INPUT(COMMAND)
1089          -> TEST
1090       %FINISH
1091    !
1092    %ENDOFPROGRAM
```

array A



```
| SWITCH TABLE FOR IMP EDITOR: <SYMBOL> => <VALUE>
|    '(' =>  1
|    ')' =>  2
|    ',' =>  3
|    '¬' =>  4
|    '*' =>  5
|    '?' =>  6
|    '/' =>  7
|    '.' =>  8
|    ''' =>  9
|    ' ' => 10
|     NL => 11
|    '+' => 12
|    '-' => 13
|
| ALL DIGITS => 14
|
| LETTERS(UPPER & LOWER CASE) RUN CONSECUTIVELY FROM 15
|    'A' => 15    'B' => 16    'C' => 17    'D' => 18    'E' => 19
|    'F' => 20    'G' => 21    'H' => 22    'I' => 23    'J' => 24
|    'K' => 25    'L' => 26    'M' => 27    'N' => 28    'O' => 29
|    'P' => 30    'Q' => 31    'R' => 32    'S' => 33    'T' => 34
|    'U' => 35    'V' => 36    'W' => 37    'X' => 38    'Y' => 39
|    'Z' => 40
|
%CONSTSHORTINTEGER NVAL= 14
%CONSTBYTEINTEGERARRAY  CTAB(0:127) = 0,
       0,  0,  0,  0,  0,  0,  0,  0,  0, 11,  0,  0,  0,  0,  0,
       0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
       0, 10,  0,  0,  0,  0,  0,  0,  9,  1,  2,  5, 12,  3, 13,
       8,  7, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14,  0,  0,  0,
       0,  0,  6,  0, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25,
      26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40,
       0,  4,  0,  0,  0,  0, 15, 16, 17, 18, 19, 20, 21, 22, 23,
      24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38,
      39, 40,  0,  0,  0,  0,  0
%SWITCH  SW(0: 40)
```

```
*.1 COMPLETE
*FAILED
>p

>[ $line(++)=((i/-/)#1(i/ /)#2)4 in]
>$line 13 4
>m-p

 F A U L T  32  736    IN: 0    OUT: 0   SIZE: 5948
 STOPPED  AT  LINE  710

NO NEW DT1 XXX TMP

*e edit5a
          m705p7
       ! ONE AND MUST THEREFORE BE COPIED IN ITS ENTIRETY WHEN
       ! COING BACKWARDS, NOT MERELY REWOUND AND RE-READ.
       K = AA(LP)
       %WHILE K # NL %DO:
            I = I+1 %AND AA(I) = K %IF K # 0  ;! SKIP NULLS
            LP = LP+1; K = AA(LP)
       %REPEAT
>?

IMP 15

*r n,,.1,.2,.9/xxx,,.1,.2
EDIT5A   -  AMAX= 735
>
>[ $line(++)=((i/-/)#1(i/ /)#2)4]\ in]
>
>
>g3
:top
:bottom
::
*FAILED
>m-2p
TOP
>mp
*.1 COMPLETE
*.2 COMPLETE
*.2 COMPLETE
*.1 COMPLETE
*.1 COMPLETE
*.2 COMPLETE
*.2 COMPLETE
*.1 COMPLETE
*.1 COMPLETE
*.2 COM

 ?

IMP 15

*r n,,.1,.2,.9/,,.1,.2
EDIT5A   -  AMAX= 912
>
>
>g3
:top
:bottom
::
*FAILED
>m-*
>p
```

```
*r n,,.1,.2,.9/xxx,,.1,.2
EDIT5A   -  AMAX= 735
>
>[$line(++)=((i/-/)#1(i/ /)#2)4]\ in]
>
>
>g3
:top
:bottom
::
*FAILED
>m-2p
TOP
>mp
*.1 COMPLETE
*.2 COMPLETE
*.2 COMPLETE
*.1 COMPLETE
*.1 COMPLETE
*.2 COMPLETE
*.2 COMPLETE
*.1 COMPLETE
*.1 COMPLETE
*.2 COM

 ?

IMP 15

*r n,,.1,.2,.9/,,.1,.2
EDIT5A   -  AMAX= 912
>
>
>g3
:top
:bottom
::
*FAILED
>m-*
>p
TOP
>m
*.1 COMPLETE
*.2 COMPLETE
*.2 COMPLETE
*.1 COMPLETE
*.1 COMPLETE
*.2 COMPLETE
*.2 COMPLETE
*.1 COMPLETE
*.1 COMPLETE
*.2 COMPLETE
*.2 COMPLETE
*.1 COMPLETE
*.1 COMPLETE
*.2 COMPLETE
MOVE(3):  TARGET,ORIGIN,OFFSET,LBLINES,AVLINES,STEP= 2 0 3 2 0 0
MOVE(3):  TARGET,ORIGIN,OFFSET,LBLINES,AVLINES,STEP= 2 2 1 0 0 0
MOVE(4):  TARGET,ORIGIN,OFFSET,LBLINES,AVLINES,STEP= 2 2 1 0 0 0
*.2 COMPLETE
MOVE(3):  TARGET,ORIGIN,OFFSET,LBLINES,AVLINES,STEP= 2 2 1 0 0 0
MOVE(4):  TARGET,ORIGIN,OFFSET,LBLINES,AVLINES,STEP= 2 2 1 0 0 0
*.1 COMPLETE
MOVE(3):  TARGET,ORIGIN,OFFSET,LBLINES,AVLINES,STEP= 2 0 3 2 0 0
MOVE(3):  TARGET,ORIGIN,OFFSET,LBLINES,AVLINES,STEP= 2 2 1 0 0 0
MOVE(4):  TARGET,ORIGIN,OFFSET,LBLINES,AVLINES,STEP= 2 2 1 0 0 0
*.1 COMPLETE
MOVE(3):  TARGET,ORIGIN,OFFSET,LBLINES,AVLINES,STEP= 2 2 1 0 0 0
MOVE(4):  TARGET,ORIGIN,OFFSET,LBLI
?
```